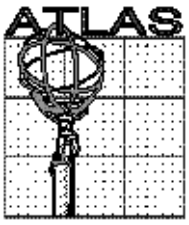

ATLAS
High-Level Trigger, Data Acquisition and Controls

Technical Design Report

Issue:	1
Revision:	0
Reference:	ATLAS TDR-016
Created:	12 November 2002
Last modified:	30 June 2003
Prepared By:	ATLAS HLT/DAQ/DCS Group

ATLAS Collaboration

Armenia
Yerevan Physics Institute, Yerevan

Australia
Research Centre for High Energy Physics, Melbourne University, Melbourne
University of Sydney, Sydney

Austria
Institut für Experimentalphysik der Leopold-Franzens-Universität Innsbruck, Innsbruck

Azerbaijan Republic
Institute of Physics, Azerbaijan Academy of Science, Baku

Republic of Belarus
Institute of Physics, National Academy of Science, Minsk
National Centre for Particle and High Energy Physics, Minsk

Brazil
Universidade Federal do Rio de Janeiro, COPPE / EE / IF, Rio de Janeiro

Canada
University of Alberta, Edmonton
University of Carleton / C.R.P.P., Carleton
Group of Particle Physics, University of Montreal, Montreal
Department of Physics, University of Toronto, Toronto
Simon Fraser University, Burnaby, BC
TRIUMF, Vancouver
Department of Physics, University of British Columbia, Vancouver
University of Victoria, Victoria

CERN
European Laboratory for Particle Physics (CERN), Geneva

China
Joint Cluster formed by IHEP Beijing, USTC Hefei, University of Nanjing and University of Shandong

Czech Republic
Academy of Sciences of the Czech Republic, Institute of Physics and Institute for Computer Science, Prague
Charles University in Prague, Faculty of Mathematics and Physics, Prague
Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Faculty of Mechanical Engineering, Prague

Denmark
Niels Bohr Institute, University of Copenhagen, Copenhagen

France
Laboratoire d'Annecy-le-Vieux de Physique des Particules (LAPP), IN2P3-CNRS, Annecy-le-Vieux
Laboratoire de Physique Corpusculaire, Université Blaise Pascal, IN2P3-CNRS, Clermont-Ferrand
Laboratoire de Physique Subatomique et de Cosmologie de Grenoble, IN2P3-CNRS Université Joseph Fourier, Grenoble
Centre de Physique des Particules de Marseille, IN2P3-CNRS, Marseille
Laboratoire de l'Accélérateur Linéaire, IN2P3-CNRS, Orsay
LPNHE, Universités de Paris VI et VII, IN2P3-CNRS, Paris
Commissariat à l'Energie Atomique (CEA), DSM / DAPNIA, Centre d'Etudes de Saclay, Gif-sur-Yvette

Georgia
Institute of Physics of the Georgian Academy of Sciences and Tbilisi State University, Tbilisi

Germany
Physikalisches Institut, Universität Bonn, Bonn
Institut für Physik, Universität Dortmund, Dortmund
Fakultät für Physik, Albert-Ludwigs-Universität, Freiburg
Kirchhoff-Institut für Physik der Universität Heidelberg, Heidelberg
Institut für Physik, Universität Mainz, Mainz
Lehrstuhl für Informatik V, Universität Mannheim, Mannheim
Sektion Physik, Ludwig-Maximilian-Universität München, Munich
Max-Planck-Institut für Physik, Munich
Fachbereich Physik, Universität Siegen, Siegen
Fachbereich Physik, Bergische Universität, Wuppertal

Greece
Athens National Technical University, Athens
Athens University, Athens
Aristotle University of Thessaloniki, High Energy Physics Department and Department of Mechanical Engineering, Thessaloniki

Israel
Department of Physics, Technion, Haifa
Raymond and Beverly Sackler Faculty of Exact Sciences, School of Physics and Astronomy, Tel-Aviv University, Tel-Aviv
Department of Particle Physics, The Weizmann Institute of Science, Rehovot

Italy
Dipartimento di Fisica dell'Università della Calabria e I.N.F.N., Cosenza
Laboratori Nazionali di Frascati dell'I.N.F.N., Frascati
Dipartimento di Fisica dell'Università di Genova e I.N.F.N., Genova
Dipartimento di Fisica dell'Università di Lecce e I.N.F.N., Lecce
Dipartimento di Fisica dell'Università di Milano e I.N.F.N., Milan
Dipartimento di Scienze Fisiche, Università di Napoli 'Federico II' e I.N.F.N., Naples
Dipartimento di Fisica Nucleare e Teorica dell'Università di Pavia e I.N.F.N., Pavia
Dipartimento di Fisica dell'Università di Pisa e I.N.F.N., Pisa
Dipartimento di Fisica dell'Università di Roma I 'La Sapienza' e I.N.F.N., Roma
Dipartimento di Fisica dell'Università di Roma II 'Tor Vergata' e I.N.F.N., Roma
Dipartimento di Fisica dell'Università di Roma III 'Roma Tre' e I.N.F.N., Roma
Dipartimento di Fisica dell'Università di Udine, Gruppo collegato di Udine I.N.F.N. Trieste, Udine

Japan
Hiroshima Institute of Technology, Hiroshima
Department of Physics, Hiroshima University, Higashi-Hiroshima
KEK, High Energy Accelerator Research Organisation, Tsukuba
Department of Physics, Faculty of Science, Kobe University, Kobe
Department of Physics, Kyoto University, Kyoto
Kyoto University of Education, Kyoto
Nagasaki Institute of Applied Science, Nagasaki
Naruto University of Education, Naruto
Department of Physics, Faculty of Science, Okayama University, Okayama
Department of Computer Science, Ritsumeikan University, Kusatsu
Department of Physics, Faculty of Science, Shinshu University, Matsumoto
International Center for Elementary Particle Physics, University of Tokyo, Tokyo
Physics Department, Tokyo Metropolitan University, Tokyo
Department of Applied Physics System, Tokyo University of Agriculture and Technology, Tokyo
Institute of Physics, University of Tsukuba, Tsukuba

Morocco
Faculté des Sciences Aïn Chock, Université Hassan II, Casablanca, and Université Mohamed V, Rabat

Netherlands
FOM - Institute SAF NIKHEF and University of Amsterdam / NIKHEF
University of Nijmegen / NIKHEF, Nijmegen

Norway
University of Bergen, Bergen
University of Oslo, Oslo

Poland
Henryk Niewodniczanski Institute of Nuclear Physics, Cracow
Faculty of Physics and Nuclear Techniques of the University of Mining and Metallurgy, Cracow

Portugal
Laboratório de Instrumentação e Física Experimental de Partículas (LIP), Lisbon, in collaboration with:
Faculdade de Ciências de Universidade de Lisboa (FCUL), University of Coimbra, University Católica-Figueira da Foz, and University Nova de Lisboa

Romania
National Institute for Physics and Nuclear Engineering, Institute of Atomic Physics, Bucharest

Russia
Institute for Theoretical and Experimental Physics (ITEP), Moscow
P.N. Lebedev Institute of Physics, Moscow
Moscow Engineering and Physics Institute (MEPhI), Moscow
Moscow State University, Moscow
Budker Institute of Nuclear Physics (BINP), Novosibirsk
State Research Center of the Russian Federation - Institute for High Energy Physics (IHEP), Protvino
Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg

JINR
Joint Institute for Nuclear Research , Dubna

Republic of Serbia
Institute of Physics, University of Belgrade, Belgrade

Slovak Republic
Bratislava University, Bratislava, and Institute of Experimental Physics of the Slovak Academy of Sciences, Kosice

Slovenia
Jozef Stefan Institute and Department of Physics, University of Ljubljana, Ljubljana

Spain
Institut de Física d'Altes Energies (IFAE), Universidad Autónoma de Barcelona, Bellaterra, Barcelona
Physics Department, Universidad Autónoma de Madrid, Madrid
Instituto de Física Corpuscular (IFIC), Centro Mixto Universidad de Valencia - CSIC, Valencia and Instituto de Microelectrónica de Barcelona, Bellaterra, Barcelona

Sweden
Fysika institutionen, Lunds universitet, Lund
Royal Institute of Technology (KTH), Stockholm
University of Stockholm, Stockholm
Uppsala University, Department of Radiation Sciences, Uppsala

Switzerland
Laboratory for High Energy Physics, University of Bern, Bern
Section de Physique, Université de Genève, Geneva

Taiwan
Institute of Physics, Academia Sinica, Taipei

Turkey
Department of Physics, Ankara University, Ankara
Department of Physics, Bogaziçi University, Istanbul

United Kingdom
School of Physics and Astronomy, The University of Birmingham, Birmingham
Cavendish Laboratory, Cambridge University, Cambridge
Department of Physics and Astronomy, University of Glasgow, Glasgow
Department of Physics, Lancaster University, Lancaster
Department of Physics, Oliver Lodge Laboratory, University of Liverpool, Liverpool
Department of Physics, Queen Mary and Westfield College, University of London, London
Department of Physics, Royal Holloway, University of London, Egham
Department of Physics and Astronomy, University College London, London
Department of Physics and Astronomy, University of Manchester, Manchester
Department of Physics, Oxford University, Oxford
Rutherford Appleton Laboratory, Chilton, Didcot
Department of Physics, University of Sheffield, Sheffield

United States of America
State University of New York at Albany, New York
Argonne National Laboratory, Argonne, Illinois
University of Arizona, Tucson, Arizona
Department of Physics, The University of Texas at Arlington, Arlington, Texas
Lawrence Berkeley Laboratory and University of California, Berkeley, California
Physics Department of the University of Boston, Boston, Massachusetts
Brandeis University, Department of Physics, Waltham, Massachusetts
Brookhaven National Laboratory (BNL), Upton, New York
University of Chicago, Enrico Fermi Institute, Chicago, Illinois
Nevis Laboratory, Columbia University, Irvington, New York
Department of Physics, Duke University, Durham, North Carolina
Department of Physics, Hampton University, Virginia
Department of Physics, Harvard University, Cambridge, Massachusetts
Indiana University, Bloomington, Indiana
Iowa State University, Ames, Iowa
University of California, Irvine, California
Massachusetts Institute of Technology, Department of Physics, Cambridge, Massachusetts
Michigan State University, Department of Physics and Astronomy, East Lansing, Michigan
University of Michigan, Department of Physics, Ann Arbor, Michigan
Department of Physics, New Mexico University, Albuquerque, New Mexico
Northern Illinois University, DeKalb, Illinois
Ohio State University, Columbus, Ohio
Department of Physics and Astronomy, University of Oklahoma
Department of Physics, University of Pennsylvania, Philadelphia, Pennsylvania
University of Pittsburgh, Pittsburgh, Pennsylvania
Department of Physics and Astronomy, University of Rochester, Rochester, New York
Institute for Particle Physics, University of California, Santa Cruz, California
Department of Physics, Southern Methodist University, Dallas, Texas
State University of New York at Stony Brook, New York
Tufts University, Medford, Massachusetts
High Energy Physics, University of Illinois, Urbana, Illinois
Department of Physics, Department of Mechanical Engineering, University of Washington, Seattle, Washington
Department of Physics, University of Wisconsin, Madison, Wisconsin

Acknowledgements

The authors would like to thank Mario Ruggier for preparing the template upon which this document is based and the DocSys group for their help in using it.

Contents

Part 1

Global View

1 Overview

1.1 Document overview

This Technical Design Report (TDR) for the High-level Trigger (HLT), Data Acquisition (DAQ) and Controls of the ATLAS experiment builds on the earlier documents published on these systems: Trigger Performance Status Report [1-1], DAQ, EF, LVL2 and DCS Technical Progress Report [1-2], and HLT/DAQ/DCS Technical Proposal [1-3]. Much background and preparatory work relevant to this TDR is referenced in the above documents. In addition, a large amount of detailed technical documentation has been produced in support of this TDR. These documents are referenced in the appropriate places in the following chapters.

This section introduces the overall organization of the document. The following sections give an overview of the principal system requirements and functions, as well as a brief description of the principal data types used in the TDAQ system.

The document has been organized into four parts:

- Part I — Global View

  Chapters 2, 3 and 4 address the principal system and experiment parameters which define the main requirements of the HLT, DAQ and Controls system. The global system operations, and the physics requirements and event selection strategy are also addressed. Chapter 5 defines the overall architecture of the system and analyses the requirements of its principal components, while Chapters 6 and 7 address more specific fault-tolerance and monitoring issues.

- Part II — System Components

  This part describes in more detail the principal components and functions of the system. Chapter 8 addresses the final prototype design and performance of the Data Flow component. These are responsible for the transport of event data from the output of the detector Read Out Links (ROLs) via the HLT system (where event selection takes place) to mass storage. Chapter 9 explains the decomposition of the HLT into a second level trigger (LVL2) and an Event Filter (EF) component. It details the design of the data flow within the HLT, the specifics of the HLT system supervision, and the design and implementation of the Event Selection Software (ESS). Chapter 10 addresses the Online Software which is responsible for the run control and DAQ supervision of the entire Trigger/DAQ (TDAQ) and detector systems during data taking. It is also responsible for miscellaneous services such as error reporting, run parameter accessibility, and histogramming and monitoring support. Chapter 11 describes the Detector Control System (DCS), responsible for the control and supervision of all the detector hardware and of the services and the infrastructure of the experiment. The DCS is also the interface point for information exchange between ATLAS and the LHC accelerator. Chapter 12 draws together the various aspects of experiment control detailed in previous chapters and examines several use-cases for the overall operation and control of the experiment, including: data-taking operations, calibration runs, and operations required outside data-taking.

- Part III — System Performance

  Chapter 13 addresses the physics selection. The tools used for physics selection are described along with the event-selection algorithms and their performance. Overall HLT

output rates and sizes are also discussed. A first analysis of how ATLAS will handle the first year of running from the point of view of TDAQ is presented. Chapter 14 discusses the overall performance of the HLT/DAQ system from various points of view, namely the HLT performance as evaluated in dedicated testbeds, the overall performance of the TDAQ system in a testbed of ~10% ATLAS size, and functional tests of the system in the detector test-beam environment. Data from these various testbeds are also used to calibrate a detailed discrete-event -simulation model of data flow in the full-scale system.

- Part IV — Organization and Planning

  Chapter 15 discusses quality-assurance issues and explains the software-development process employed. Chapter 16 presents the system costing and staging scenario. Chapter 17 presents the overall organization of the project and general system-resource issues. Chapter 18 presents the short-term HLT/DAQ work-plan for the next phase of the project as well as the global development schedule up to LHC turn-on in 2007.

1.2 Main system requirements

This section presents some of the principal requirements on the HLT/DAQ system from several points of view. The response to these requirements in terms of the system design is then presented in the later chapters of the document.

1.2.1 Rate requirements and event-selection strategy

The LHC proton bunches will cross at a frequency of 40 MHz. At the machine's design luminosity of 1×10^{34} cm^{-2} s^{-1}, on average about 23 inelastic proton–proton collisions will be produced at each bunch crossing. The level-1 trigger (LVL1) [1-4] will make the first level of event selection, reducing the initial event rate to less than about 75 kHz. Operation at up to about 100 kHz is possible with somewhat increased deadtime. The HLT must reduce the event rate further to *O*(100) Hz. Each selected event will have a total size of ~1.5 Mbyte giving a required storage capability of a few hundred Mbyte/s.

The ATLAS baseline assumes a maximum LVL1 rate of 75 kHz. However, for the purposes of the system design and costing, a maximum LVL1 rate of 100 kHz has been assumed in this document, in order to ensure that if ATLAS decides to run at this LVL1 rate, that the HLT/DAQ system will be able to handle it.

ATLAS aims to use inclusive triggers as much as possible in its physics-selection strategy in order to maximise its physics coverage and to be as open as possible to new and possibly un-foreseen signatures. The configuration of the entire trigger system, including LVL1, must be sufficiently flexible that one can easily change both algorithms and their thresholds between data-taking runs. The system also needs to be able to respond rapidly to the consequences of changes in the beam conditions in terms of performance and stability.

1.2.2 Detector readout requirements

The HLT/DAQ system is required to handle data coming in parallel from the detector readout drivers (RODs) over some 1600 point-to-point readout links (ROLs), each having a maximum

bandwidth of 160 Mbyte/s. Taking into account estimates of the bandwidth that will be used by each detector, the total readout bandwidth to be dealt with after the LVL1 trigger is more than ~160 Gbyte/s. This number depends on the detector occupancy as well as the LVL1 trigger rate and will vary considerably according to the luminosity being delivered by the LHC. Other important aspects bearing on the total readout bandwidth are the data compression and zero suppression schemes which are still under study in some of the detector systems. A more detailed description of the detectors' readout parameters can be found in Chapter 2.

1.2.3 Functional and operational requirements

The TDAQ system must be designed and constructed in such a way as to provide ATLAS with highly reliable and efficient data-taking and event-selection capabilities. The huge investment in both financial and human resources for the ATLAS detector itself, and also for the LHC machine, means that data-taking operations must be established and optimized as soon as possible after first collisions in the LHC in 2007, in order to capitalize on this investment.

From early stages in the development of the ATLAS HLT/DAQ system, elements of the system (in particular those concerned with data acquisition) have been used and tested in a test beam environment, providing the data acquisition functionality and performance needed for the detector data taking. A major effort has been made to minimize the functional divergence between the system used at the test beam and that being developed for the final experiment. Apart from providing a real-life, albeit scaled down, testing facility for the HLT/DAQ system, this policy has the advantage of familiarizing the detector communities in ATLAS with the HLT/DAQ system at an early stage. Some of the elements of the system (those closest to the detector readout, and their associated control and supervision functions) will be required by the detectors during their commissioning phases, both above and below ground. Requiring that the detectors be able to use and give feedback on the HLT/DAQ system well in advance of this, therefore offers considerable advantages both to TDAQ and to the detector communities in terms of easing the installation and commissioning phase of the experiment.

An essential requirement on the HLT/DAQ system which will be particularly important in the commissioning and installation phase is the ability to partition the system into several independent but fully-functional entities. It must be possible for several detectors and/or several parts of a given detector to be triggered and to take data in parallel and independently of each other. This is in order to facilitate and render as parallel as possible the detector debugging and commissioning operations. During physics running, it will be necessary to have the capability to run a partition of an element of a given detector in test mode, to help track down a fault, while the rest of the ATLAS detector is taking data normally.

The DCS is an integral part of the TDAQ system and assumes a particular role in assuring the coherent, safe operation and monitoring of all components of the ATLAS detector. Although it is highly integrated with other parts of the system, the DCS has the particular requirements of being operational 24 hours a day, seven days a week and of being highly fault tolerant. The principal elements of the DCS must be installed and commissioned in time for the first detector commissioning operations which will begin in early 2005. These elements must be able to operate in a stand-alone mode i.e. without depending on other parts of the HLT/DAQ system being available.

Constraints of floor space and cooling capacity, in particular in the underground experimental cavern and adjoining service rooms, limit the number of racks available to the HLT/DAQ system. The proposed location of the HLT/DAQ racks is presented in Section 5.5.11.

1.2.4 Requirements due to the long expected lifetime of ATLAS

The installation and commissioning phase of ATLAS [1-5] will take in excess of four years and the experiment is expected to take data for fifteen years or more. This timescale puts a strong premium on the requirement for a highly modular and cost-effective system design. This facilitates the replacement or upgrading of specific elements of the system in a manner that will have little or no side effects on neighbouring elements.

Experience has shown that custom electronics is more difficult and expensive to maintain in the long term than comparable commercial products. The use of commercial computing and network equipment, and the adoption of commercial protocol standards such as Ethernet, wherever appropriate and possible, is a requirement which will help us to maintain the system for the full lifetime of the experiment. The adoption of widely-supported commercial standards and equipment at the outset will also enable us to benefit from future improvements in technology by rendering equipment replacement and upgrade relatively transparent. An additional benefit of such an approach is the highly-competitive commercial market which offers high performance at low cost.

1.3 System components and functions

In this section, the principal components and functions of the baseline HLT/DAQ system are described very briefly in order to give the reader an overview of the system before proceeding to the subsequent chapters where detailed descriptions are given. A schematic diagram is presented in Figure 1-1. HLT/DAQ can be broken down into four principal systems, namely:

- The Data Flow system — responsible for receiving the detector data, serving of a subset of data to the HLT system, and transporting the data for selected events to mass storage
- The HLT system — responsible for the post-LVL1 event selection and filtering involving a rate reduction of a factor of several hundred, and for the classification of all accepted events
- The Online system — responsible for all aspects of experiment and TDAQ operation and control during data-taking, and during testing and calibration runs
- The DCS — responsible for the coherent and safe operation of the ATLAS detector, as well as the interface with external systems and services including the LHC itself.

The Online system is implicitly understood to be connected to all elements in Figure 1-1, and the DCS to all hardware elements which need to be monitored and controlled.

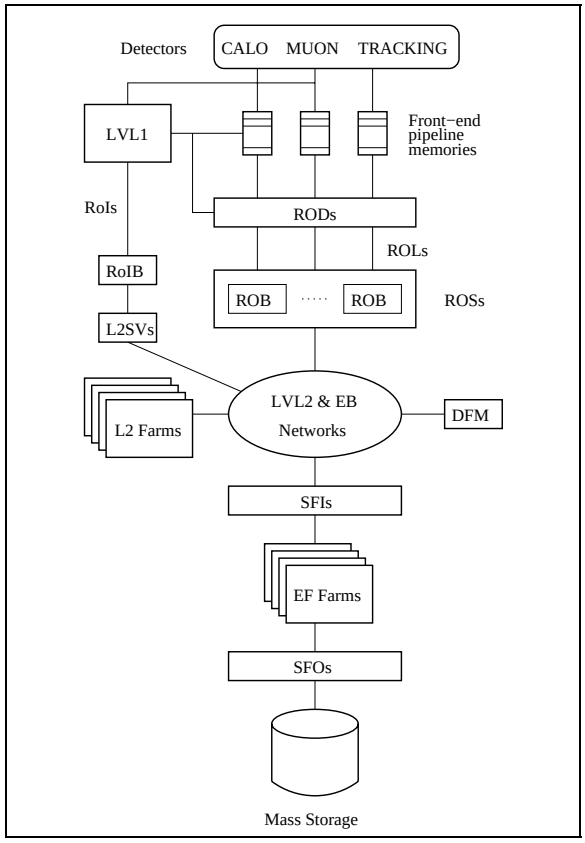

Figure 1-1 Principal components of the Data Flow and HLT systems

1.3.1 The Data Flow system

ATLAS decided early on to define the boundary between the detector readout and the data acquisition to be at the input of the Read Out Buffers (ROBs) [1-6]. The ROLs, for which there is an agreed ATLAS standard, transport data fragments of LVL1-accepted events from the detectors' RODs to the ROBs. The Read Out Systems (ROS's) each contain several ROBs.

Requested data fragments from selected ROBs are served to the LVL2 trigger element of the HLT system. Event data fragments for LVL2-accepted events are then built, on the initiation of the Data Flow Manager (DFM), from the ROBs, across a switched Ethernet network, into a complete event by one of the ~100 Sub-Farm Interfaces (SFIs). The SFIs then serve the complete events to the second element of the HLT system, the Event Filter (EF). Events selected by the EF for final archiving in preparation for offline reconstruction and primary analysis are passed to permanent storage via the final element of the Data Flow system, the Sub-Farm Output (SFO).

Most of the element interconnection in the Data Flow system is done using standard Gigabit Ethernet network and switching technology. The maximum network bandwidth capacity required for building events accepted by the LVL2 trigger is expected to be ~5 Gbyte/s, much less than the aggregate data rate into the RODs for events retained by LVL1.

1.3.1.1 ROD-crate data acquisition

The detector ROD crates contain a variety of modules, including: the RODs themselves, modules used for control of the Front-end Electronics and for processing event data upstream of the RODs, as well as one or more ROD Crate Processors (RCPs). They may also contain reduced-functionality ROD prototypes in laboratory setups or at test beams. Each ROD Crate is connected by a local-area Ethernet network to one or more ROD Crate Workstations (RCWs).

The detector communities need common DAQ functionality at the level of the ROD Crate in several environments: in laboratory setups; for performing tests during assembly of their detectors; at test beams; and also in-situ at the experiment during commissioning and when running with beam. The ROD-Crate DAQ [1-7] provides this functionality and is part of the HLT/DAQ system itself. It comprises all common software for operating one or more ROD Crates, and it runs inside the ROD Crate (on RCPs) as well as on the RCWs. It provides the functionality for configuration and control, ROD emulation, monitoring, calibration at the level of the ROD Crate, and event building across multiple ROD Crates. The system is described in more detail in Chapter 8.

1.3.2 The High-Level Trigger system

The HLT system is comprised of the LVL2 trigger and the Event Filter. Although these will both be built using farms of standard PCs interconnected by Ethernet networks, they differ in several important respects. A combination of high rejection power with fast, limited precision algorithms using modest computing power in LVL2, and modest rejection power with slower, high precision algorithms using more extensive computing power in the EF is a cost-effective and flexible way to implement the HLT.

The LVL2 trigger must work at the LVL1 accept rate with an average event treatment time of ~10 ms. In order to operate within this time budget, the LVL2 trigger will use a sequence of highly optimized trigger selection algorithms which operate on only a fraction (typically ~2%) of the event data. The LVL1 trigger identifies regions in the detector where it has found interesting features, so-called Regions Of Interest (RoIs) [1-4]. The RoI Builder (RoIB) combines the RoI information from the various parts of the LVL1 trigger and passes it to the LVL2 Supervisor (L2SV). The L2SV allocates an event to one of the computing nodes in the LVL2 farm and transfers the RoI information for the event to the allocated processor (see Figure 1-1). These RoIs are then used to seed the LVL2 algorithms. This enables the algorithms to select precisely the region of the detector in which the interesting features reside and therefore the ROBs from which to request the data for analysis. Data requests may be done several times per event by different HLT algorithms that progressively refine the selection. At each stage in the processing an event may be rejected. Each LVL2 processor can treat several events concurrently.

The LVL2 trigger decisions are communicated, via the L2SV, to the DFM for event deletion or building. It should be noted that all the data for a given event are stored in the ROBs during the LVL2 processing and, for selected events, until the event building process is completed.

The Event Filter has to work at the LVL2 accept rate with an average event treatment time of ~1 s. Compared to LV2, more sophisticated reconstruction and trigger algorithms, tools adapted from those of the offline, and more complete and detailed calibration information are used here in making the selection. The EF receives fully built events from the SFI and so the entirety of the data is available locally for analysis. All the selection processing for a given event is done in a

single processor of the EF processor farm. Events not selected by the EF are deleted and those accepted are passed to the SFO for transmission to mass storage.

The scope, complexity, degree of generality, and precision of measured quantities (including p_T, E_T, and isolation variables) of the LVL2 and EF algorithms is different. The overall framework of the ESS in which they operate has been designed in such a way that all the algorithms may be developed in the same (offline) development environment and have the same data-interface definition. The detailed implementation of this interface is, however, different in each case. This approach has the major advantage of having a high degree of development commonality and flexibility of scope across the spectrum of the HLT and the offline, as well as facilitating performance comparisons.

1.3.3 The Online Software system

The Online Software system is responsible for configuring, controlling, and monitoring the TDAQ system, but excludes any management, processing, or transportation of event data. It is a framework which provides the glue between the various elements of the DAQ, HLT and DCS systems, and defines interfaces to those elements. It also includes information-distribution services and access to configuration and other meta-data databases.

An important part of the Online software is to provide the services that enable the TDAQ and detector systems to start up and shut down. It is also responsible for the synchronization of the the entire system, and the supervision of processes. Verification and diagnostics facilities help with the early detection of problems. The configuration services provide the framework for storing the large amount of information required to describe the system topology, including hardware and software components. During data taking, access is provided to monitoring tasks, histograms produced in the TDAQ system, and also the errors and diagnostics messages sent by different applications. One or more user interfaces display the available information and enable the user to configure and control the TDAQ system.

1.3.4 The Detector Control system

The DCS supervises all hardware of the experimental set-up, including all detector systems of ATLAS and the common experimental infrastructure. It also communicates with external systems like the infrastructure services of CERN and most notably with the LHC accelerator.

Safety aspects are treated by DCS only at the least-severe level. This concerns mainly questions of sequencing operations or requiring specific conditions to prevail before allowing certain procedures to be executed. Tools for interlocks, both in hardware and in software, are also provided by DCS. Monitoring and prevention of situations which could cause major damage to the detector or even endanger people's lives are the responsibility of a dedicated Detector Safety System (DSS), and the CERN-wide safety and alarm system, respectively. DCS interacts with both of these systems.

All actions initiated by the operator and all errors, warnings, and alarms concerning the hardware of the detector are handled by DCS. It provides online status information to the level of detail required for global system operation. The interaction of detector experts with their detector is also done via DCS. DCS continuously monitors all operational parameters, gives guidance

to the operator, and signals any abnormal behaviour. It must also have the capability to take appropriate action automatically if necessary and to bring the detector to a safe state.

Concerning the operation of the experiment, close interaction with the DAQ system is of prime importance. Good-quality physics data requires detailed synchronization between the DAQ system and DCS. Both systems are complementary in as far that the DAQ deals with the data describing a physics event (characterized by an event number), while DCS treats all data connected with the hardware of the detector related to the operational state of the detector when the data were taken (categorized by a time interval). The correlation between them is required for offline analysis.

Some parts of the detector will operate continuously because any interruption is costly in time or money, or may even be detrimental to the performance of that detector. Hence supervision by DCS is needed continuously. The DAQ system in contrast is required only when physics data are taken or during specific monitoring, calibration, or testing runs. Therefore DCS needs complete operational independence. This must, however, not result in boundaries which limit functionality or performance. Therefore both DAQ and DCS share elements of a common software infrastructure.

1.4 Data types

The TDAQ system has to deal with several broad classes of data which possess different characteristics. These classes are introduced here briefly and discussed in more detail later in the document.

- Detector control data

  The DCS system will produce large amounts of system status and hardware-parameter data at frequencies of up to of order 1 Hz. However, if the system being monitored is in a stable condition, it can be expected that these values will change little over periods of several hours or more. The important quantity to monitor in many cases will therefore be variations of the values read rather than the values themselves, thus reducing the need to transport and store large quantities of very similar data across the DCS and online control networks. A more detailed discussion of DCS data handling can be found in Chapter 11.

- Event data

  Event data are those read out from the detectors following a trigger, with the addition of the data produced by the various stages of the trigger itself while processing the event. The detailed internal format of these data is defined by the characteristics of each detector's readout electronics. However, the data are surrounded by headers and trailers in a 'raw data event' format that is common across the experiment [1-8]. Event data are represented physically as a single stream of bytes, which is subsequently referred to as a 'bytestream'. Prior to their use in the HLT, bytestream data are reformatted and converted into quantities more suitable for the selection algorithms (see Chapter 9). Bytestream data from events accepted by the HLT will be stored in mass storage in preparation for the prompt reconstruction — the first step in the offline analysis. Event data are transported by the Data Flow central network.

- Configuration data

  These data are used to configure the TDAQ and detector systems in preparation for data-taking and they define the precise configuration of the system used for a particular run.

Examples of this data type include electronics parameters and thresholds, module mapping, software executables, network parameters, trigger parameters and thresholds, high-voltage values, etc. A given data-taking run is defined by an unique set of configuration data. The run's configuration data are stored in and accessed from a dedicated in-memory database during data taking. Configuration data are transported over the Online control network.

- Conditions data

  Conditions data include all data that are relevant for the complete analysis of any given selection of event data and include data from some of the above categories. Conditions data will be stored for the lifetime of the experiment and are labelled by Interval-of-Validity (IOV) tags. The conditions data for a given run will include all the configuration data used for that run, the associated DCS data, detailed detector calibration data, magnet calibration, etc. Conditions data will be produced and accessed in many different areas of the experiment, both online and offline. Conditions data in the TDAQ system are transported over the DCS and Online control networks.

- Online statistics and monitoring data

  This type of data will be produced in large quantities by both the detectors and the TDAQ system during data taking. They are similar in some respects to DCS data in that in many cases, the monitoring and observation of the variation of parameters is more important than the parameter values themselves. This data type will be transported by both the central Data Flow network (in the case of data being sampled from the main event data flow for monitoring purposes e.g. to produce detector wire-maps etc.) and the Online control network. This subject is addressed in detail in Chapter 7.

1.5 Long-term issues and perspectives

A more detailed view of HLT/DAQ planning for the immediate future is described in Chapter 18, but it is useful to set out the principal elements of the overall ATLAS planning here. In the current ATLAS installation schedule [1-5], the initial detector will be completed in December 2006, ready for the first LHC collisions in April 2007. A cosmic ray run is planned in Autumn 2006. In its first year of operation, the target is for the LHC to attain a luminosity of 2×10^{33} cm^{-2} s^{-1}. The installation schedule of the HLT/DAQ system is dictated both by constraints coming from the installation of the system itself, as well as by the detector installation and commissioning schedule. In particular, the needs of the detectors for HLT/DAQ services during that time should be fulfilled.

The first elements of the HLT/DAQ system will be required by the detectors in early 2005, and the ROD-crate DAQ system (Section 1.3.1.1) in late 2004. The required HLT/DAQ elements are those directly involved in the initial detector readout, such as the ROBs, as well as some specific associated software elements and the complete DCS system. These components are well advanced, with in many cases final prototypes or production modules being available now or at the latest by the end of 2003. The use of these elements in the ATLAS test beam has already been noted as one important element in their development. Elements of the system further down the chain, in particular the HLT farms, their interconnections, and software will be required later. The description of these elements in this TDR documents the current status of development of the system. It is clearly desirable to purchase computing and network hardware as late as possible, while being consistent with the ATLAS schedule, to benefit from the latest technological de-

velopments. It is vital that maximum flexibility be kept in both the system schedule and design in order to facilitate this.

The increasing size and timescale of high-energy physics experiments means that their associated software is becoming increasingly complex. The software will have to be maintained and supported over periods of fifteen years or more by people who will come and go from the experiment on a much shorter timescale. In order to ease the long-term maintenance issues, a big emphasis in HLT / DAQ is being given to the use of a well-defined and appropriate software-development process (see Chapter 15), to coherent error handling and fault tolerance (see Chapter 6), and to documentation.

The use of test beams to validate and test the designs and prototype implementations of elements of the system has already been mentioned. However, this offers testing on a very small scale compared to that of the final experiment (test-beam implementations typically have just a few VME-bus crates and their associated support services). Therefore, it is desirable to test the system also on testbeds which are as large as possible within budget constraints. Furthermore, experience shows that many performance and functionality issues will only appear when testing on larger prototypes and testbeds. Nevertheless these testbeds may only represent some 10–20% of the final system size and so the extrapolation of measurements from them to the full system via advanced modelling techniques is also a vital element in the overall system-design validation. Testbed and modelling results for the system are presented and discussed in Chapter 14.

As mentioned at the beginning of this chapter, several design and technical documents precede this TDR. Further design documents will be produced by ATLAS before the start of data taking, notably the offline computing TDR which is expected in some two years from now. The offline computing TDR document will present a continuation and more complete analysis of some aspects addressed already in this TDR. In particular it will further address the areas of the Event Selection Software and performance, the definition of and access to conditions (calibration) data online, and the architecture of the mass storage and prompt reconstruction system, which immediately follow the output of the EF.

1.6 Glossary

A complete glossary can be found in Appendix B. This section addresses a few general nomenclature issues to assist the reader. For the purposes of this document, the following principal definitions are assumed:

- Detector — one of the several principal detectors of ATLAS such as the muon detector, the calorimeter detector, and tracking detector.
- The ATLAS Detector — the integrated complete set of detectors used to form the ATLAS experiment.
- Sub-detector — component parts of detectors such as a liquid argon calorimeter or the muon RPCs.
- System — the component systems of the TDAQ are the LVL1 trigger together with those defined in Section 1.3.
- Subsystem — the component parts of any of the above TDAQ systems, e.g., the LVL2 trigger, the LVL1 muon trigger.

- TDAQ — comprises the LVL1 trigger, HLT and DAQ (including DCS).
- HLT / DAQ — the TDAQ excluding the LVL1 trigger.

1.7 References

1-1 *ATLAS Trigger Performance Status Report*, CERN / LHCC / 98-15 (1998)

1-2 *ATLAS DAQ, EF, LVL2 and DCS Technical Progress Report*, CERN / LHCC / 98-16 (1998)

1-3 *ATLAS High-Level Triggers, DAQ and DCS Technical Proposal*, CERN / LHCC / 2000-17 (2000)

1-4 *ATLAS Level-1 Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

1-5 ATLAS Installation,
http: / / cern.ch / Atlas / TCOORD / Activities / TcOffice / Scheduling / Installation / ATLASinstallation.html

1-6 *Trigger & DAQ interfaces with front-end systems: requirement document version 2.0*, ATLAS EDMS Note, ATL-D-ES-0013 ()
https: / / edms.cern.ch / document / 384600 /

1-7 ROD Crate DAQ Task Force, *Data Acquisition for the ATLAS ReadOut Driver Crate (ROD Crate DAQ)*, ATLAS EDMS Note, ATL-D-ES-0007 ()
https: / / edms.cern.ch / document / 344713 /

1-8 *The raw event format in the ATLAS Trigger & DAQ*, ATLAS Internal Note, ATL-DAQ-98-129 (1998)

2 Parameters

This chapter presents the parameters relevant to the HLT/DAQ system. These include the detector-readout and the trigger-selection parameters for the correct dimensioning of the Data Flow system and for understanding the data volumes that will need to be stored. These are the subjects of the first two sections. Many of the parameter values have been extracted from a static model of the system that depends on certain input parameters such as the LVL1 trigger rate. More details are given in Appendix A**.**

Other important parameters for the correct definition of the system are those coming from the monitoring and calibration requirements. These are discussed in the following two sections.

The last section is dedicated to the DCS parameters. The extent of configuration-data traffic for start-up, configuration, and re-configuration of possible faulty elements, and the type of data produced are discussed.

2.1 Detector readout parameters

The ATLAS detector consists of three main detection systems: the Inner Detector, the Calorimeters, and the Muon Spectrometer. These systems are divided into sub-detectors.

The Inner Detector consists of three sub-detectors: Pixels, SCT and TRT [2-1], [2-2]. The Pixels sub-detector consists of semiconductor detectors with pixel readout. It is divided into two endcaps, an innermost barrel ‘B-layer’, and two outer barrel layers. All are divided into ϕ regions. The SCT sub-detector is built from silicon microstrip detectors. It is sub-divided into two endcaps and a barrel part. The latter is sub-divided into two regions, one for positive and the other for negative η. The TRT sub-detector is a tracking detector built from straw tubes and radiator. It identifies highly-relativistic particles by means of transition radiation.

The ATLAS Calorimeters consist of a Liquid Argon System and a Tilecal System. The LAr System comprises the barrel electromagnetic, endcap electromagnetic, endcap hadron, and forward calorimeters, which use liquid argon as active medium [2-3]. The barrel and two extended-barrel hadronic calorimeters, together form the Tilecal calorimeter. They use plastic scintillator tiles, read out via optical fibres and photomultipliers, as active medium and iron as absorber [2-4].

The Muon Spectrometer is divided into a barrel part and two endcaps. The barrel consists of precision chambers based on Monitored Drift Tubes (MDTs), and trigger chambers based on Resistive Plate Chambers (RPCs). The two endcaps contain both MDTs and another type of trigger chamber: Thin Gap Chambers (TGCs). Furthermore at large pseudo-rapidity and close to the interaction point, Cathode Strip Chambers (CSCs) are used in the forward regions [2-5].

The LVL1 Trigger is another source of data for the DAQ system — it provides intermediate and final results from the trigger processing that is combined with the detector data to form the complete event [2-6].

The organization of the ATLAS detector readout is specified in Table 2-1 in terms of partitions, data sources (the RODs), Read Out Links (ROLs), and Read Out Systems (ROS), assuming that a maximum of 12 ROLs can be connected to a single ROS sub-system. These numbers are almost final, but small changes may still occur due to further developments in the detector RODs. The

information in the table is based mainly on information provided by the sub-detector groups during the Third ATLAS ROD Workshop held in Annecy in November 2002 [2-7]. The partitions used coincide with the TTC partitions described in [2-6]. A ROD module, ROD crate, and ROS are associated with a single partition. Each partition can function independently.

In the following, the official ATLAS coordinate system will be used, therefore the definitions of this system are briefly summarized [2-8].

The X axis is horizontal and points from the Interaction Point (IP) to the LHC ring centre. The Z axis is parallel to the beam direction. The Y axis is perpendicular to X and to Z.

The ϕ angle is measured from the polar X axis with positive values in the anti-clockwise direction. The pseudorapidity η is measured from the Y axis, positive towards Z-positive.

A and C are the labels used to identify the two sides of any ATLAS component with respect to the pseudorapidity $\eta = 0$. They correspond to the convention of the two sides of the ATLAS cavern. If the Z axis is defined as the one along the beam direction, when looking from inside the LHC ring, the positive Z is in the left direction. The positive Z is identified as side A. The negative Z is in the right direction and is identified as side C.

Table 2-1 The distribution of the RODs and ROS sub-systems (assuming at maximum 12 ROLs per ROS sub-system) per detector per partition

Detector	Partition	RODs	ROD crates	ROLs	ROS sub-systems	ROLs per ROS sub-system
Pixel Detector	B Layer	44	3	44	4	$3 \times 12 + 8$
	Disks	12	1	12	1	1×12
	Layer 1 + 2	38 + 26	4	38 + 26	6	$5 \times 12 + 4$
SCT	Barrel A	22	2	22	2	$1 \times 12 + 10$
	Barrel C	22	2	22	2	$1 \times 12 + 10$
	Endcap A	24	2	24	2	2×12
	Endcap C	24	2	24	2	2×12
TRT	Barrel A	32	3	32	3	$2 \times 12 + 8$
	Barrel C	32	3	32	3	$2 \times 12 + 8$
	Endcap A	84	7	84	7	7×12
	Endcap C	84	7	84	7	7×12
LAr	EMBarrel A	56	4	224	19	$18 \times 12 + 8$
	EMBarrel C	56	4	224	19	$18 \times 12 + 8$
	EMEC A	35	3	138	12	$11 \times 12 + 6$
	EMEC C	35	3	138	12	$11 \times 12 + 6$
	FCAL	4	1	16	2	$1 \times 12 + 4$
	HEC	6	1	24	2	2×12

Table 2-1 The distribution of the RODs and ROS sub-systems (assuming at maximum 12 ROLs per ROS sub-system) per detector per partition

Tilecal	Barrel A	8	1	16	2	1 × 12 + 4
	Barrel C	8	1	16	2	1 × 12 + 4
	Ext Barrel A	8	1	16	2	1 × 12 + 4
	Ext Barrel C	8	1	16	2	1 × 12 + 4
MDT	Barrel A	48	4	48	4	4 × 12
	Barrel C	48	4	48	4	4 × 12
	Endcap A	48	4	48	4	4 × 12
	Endcap C	48	4	48	4	4 × 12
CSC	Endcap A	8 + 8	1	16	2	1 × 12 + 4
	Endcap C	8 + 8	1	16	2	1 × 12 + 4
RPC	Barrel A	16	1	16	2	1 × 12 + 4
	Barrel C	16	1	16	2	1 × 12 + 4
TGC	Endcap A	8	1	8	1	1 × 8
	Endcap C	8	1	8	1	1 × 8
LVL1 Muon Trigger	MIROD	1	1	1	5	4 × 12 + 8
LVL1 Calorimeter Trigger (RoI, CP, JEP and PP RODs belong to the same partition)	RoI	6	2	6		
	CP	4		16		
	JEP	4		16		
	PP	4	8	16		
	CTP	1	1	1		
Total	33	960	90	1600	144	

It is worth mentioning that the initial ATLAS detector will lack some components for initial data taking with the first LHC collisions in Spring 2007 because of funding limitations. The necessity of staging the detectors has an impact on the number of ROLs required in the initial period for some detectors. The Pixel sub-detector will initially be missing Layer 1 [2-2], the TRT sub-detector will initially not have the two end-cap ‘C’-wheels [2-1], and the CSC sub-detector will lack eight chambers per endcap [2-5]. In addition, for the LAr sub-detector the instrumentation of the RODs will be staged by using only half of the Digital Signal Processor (DSP) boards and rearranging the ROD output to reduce by a factor two the number of ROLs [2-3]. This initial staging scenario is summarized in Table 2-2.

Table 2-2 Number of RODs and ROLs for the initial and final ATLAS detector

	Initial detector		Final detector	
Sub-detector	**Number of RODs**	**Number of ROLs**	**Number of RODs**	**Number of ROLs**
Pixel	82	82	120	120
SCT	92	92	92	92
TRT	192	192	232	232
LAr	192	382	192	764
Tilecal	32	64	32	64
MDT	192	192	192	192
CSC	16	16	32	32
RPC	32	32	32	32
TGC	16	16	16	16
LVL1	24	56	24	56

2.2 Trigger and data flow parameters

Table 2-3 presents an overview of typical values of data-transfer and request rates, and of throughput for a LVL1 rate of 100 kHz at start-up and design luminosity (we recall that the ATLAS architecture supports a LVL1 accept rate of 75 kHz, upgradeable to 100 kHz). Detailed simulation of the LVL1 trigger, with menu settings chosen to cover the ATLAS physics programme (see Chapter 4) within tight cost constraints, shows that the expected accept rate at LHC start-up luminosity (2×10^{33} cm^{-2} s^{-1}) will be ~25 kHz (see Section 13.5), and that at design luminosity (1×10^{34} cm^{-2} s^{-1}) it will be ~40 kHz. Note that there are very large uncertainties on these rate estimates mainly due to uncertainties on the underlying cross-sections and the background conditions, but also from the model of the detector and the trigger used in the simulation. A simulation of the LVL2 trigger at start-up luminosity indicates a mean rejection factor of ~30 relative to LVL1 [2-9]. The numbers presented in the table are therefore estimates of the required design and performance capabilities of the system, and allow for a ‘safety factor’ of 4 in the LVL1 accept rate for start-up luminosity, and 2.5 at design luminosity.

The data needed for the LVL2 trigger and the type of processing performed, depend on the Regions of Interest (RoIs) supplied by the LVL1 trigger. Each of the four different types of RoI (‘muon’, ‘electron/gamma’, ‘jet’, and ‘tau/hadron’) has its own characteristic type of processing. The processing consists of several steps and after each step a decision is taken on whether data from other sub-detectors within the RoI should be requested for further analysis. The data rates can be estimated with the help of information on the type, rate, sizes, and locations of the RoI, and on the mapping of the detector onto the ROLs. More details are provided in Appendix A.

The size of the HLT farms and the number of Sub-Farm Inputs (SFIs) have been estimated, assuming the use of 8 GHz dual-CPU PCs for the LVL2 and EF processors, and 8 GHz single-CPU computer servers for the SFIs. At a LVL1 rate of 100 kHz, about 500 dual-CPU machines would be needed for LVL2. About 50–100 SFIs would be required, assuming an input bandwidth per

Table 2-3 Typical values and rates in the HLT/DAQ system for a 100 kHz LVL1 trigger rate

	Start-up luminosity	Design luminosity
Average number of ROLs supplying data needed by LVL2, per LVL1 accept	17.9	16.2
Average number of groups of 12 ROLs supplying data needed by LVL2, per LVL1 accept	9.1	8.5
Maximum throughput requested per ROL (MByte / s)	8.9	8.4
Maximum LVL2 request rate for data from a single ROL (kHz)	7.4	6.3
Maximum throughput requested per group of 12 ROLs (MByte /s) [a]	70	68
Maximum LVL2 request rate per group of 12 ROLs (kHz)	21	19
Total throughput LVL2 traffic (Gbyte/s)	1.6	1.5
Event Building rate (kHz)	3.0	3.5
Total throughput traffic to Event Builder (Gbyte/s)	3.6	5.3

a. Fragments of the same event input via different ROLs are assumed to be requested by a single message and to be concatenated and output also as a single message.

SFI of ~70 Mbyte/s. For the Event Filter, approximately 1600 dual-CPU machines would be needed for a ~3 kHz event-building rate. More details on these estimates are provided in Chapter 14 and its references.

2.3 Monitoring requirements

Monitoring may be done locally to the HLT/DAQ or detector systems, and/or remotely, which would require the transfer of event fragments on a sampling basis to remote analysis software. The concepts of sources and destinations of the monitoring traffic are introduced and a preliminary list is reported in Table 2-4. Investigations are under way to identify the principal sources and destinations of monitoring traffic and the network traffic that monitoring data-transfer will generate.

The following sources of monitoring data can be identified:

- ROD crates (and LVL1 trigger crates)
- ROS
- SFI
- HLT processors (LVL2, EF).

Possible destinations are:

- Workstations, e.g. in the main Control Room (SCX1), dedicated to monitoring specific sub-systems.

- Online monitoring farm (possibly a sub-set of the EF Farm). Although the precise location of this is not yet defined, monitoring results from the Online Monitoring farm will be made available in the main ATLAS Control Room.

More details on monitoring during data taking can be found in Chapter 7 and in Ref. [2-6].

Table 2-4 summarizes the present knowledge of the relations between the sources and the destinations for monitoring. This is based on a survey where detector and physics groups were asked to quantify their monitoring needs, details of which can be found in Ref. [2-7]. Some figures for the expected traffic generated by monitoring are still missing, but the ones quoted here should be considered as a reasonable upper limit. The transport medium used for event fragments used for monitoring coming from the ROS is not yet clear. Most other monitoring data will be transferred on the Control network (i.e. standard TCP/IP on Fast or Gigabit Ethernet LAN).

Table 2-4 Monitoring sources and destinations

Source / Destination	ROD/ROB	ROS	SFI	Trigger processors
Dedicated workstations	• event fragments (~5 Mbyte/s) • histograms, scalers, files, numbers from operational monitoring (several Gbyte once every hour)	• event fragments (some hundreds of Mbyte/s) • histograms (few Mbyte/s, surges of ~6 Gbyte every hour)	• histograms (some tens of Mbyte/s)	• histograms (some Mbyte/s)
Online Farm	• calibration data (The size of the muon data in particular is not yet fully decided.	• could also be done at the SFI level	• events • calibration data (several tens of Mbyte once a day)	• rejected events (≤ 1% of the total bandwidth)

2.4 Calibration requirements

The calibration of sub-detectors is another important source of data for the experiment. For each sub-detector and each type of calibration, the data may flow from the front-end electronics, through the RODs, to the ROD Crate Controller or to the same Data Flow elements used during the normal data taking, i.e. to the ROS and up to the Sub-Farm Output.

The data volumes and bandwidths involved for each sub-detector calibration are still under study and it is difficult at this stage to make a complete evaluation. Nevertheless a few examples can be discussed.

The calibrations discussed in this section concerns the sub-detector electronics and reference systems. The in situ calibration of the detector with dedicated physics channels and cosmic rays are discussed elsewhere (see Ref. [2-10]).

An example of sub-detector electronics calibration is the LAr charge injection. This calibration makes use of special calibration boards to inject a known charge in the electronics. The data can have a first treatment in the LAr RODs and in this case only the result of the calibration, a few

calculated quantities and a few histograms, are transferred to storage via the ROD crate Processor (see Section 1.3.1.1). The amount of data produced has been estimated to be around 50 Mbyte. The time needed for a complete calibration is under study but it is approximately several minutes. There is a second mode of operation for this LAr calibration in which not only the results, but also the time-samples are sent to the RODs. In this case about 50 Gbyte of data are produced and a possible scenario for the data transfer is to send all the information through the ROLs to the storage as for the physics data.

Most of the calibration procedures require dedicated runs, but there are some that can be performed while taking data in physics mode, during the assigned LHC empty bunches (e.g. the Tilecal laser calibration). Other calibrations of sub-detector reference systems do not interact with DAQ and will make use of dedicated data-acquisition systems, e.g. the Tilecal radioactive-source system, the SCT and the MDT alignment systems [2-11].

2.5 DCS parameters

DCS deals with two categories of parameters: input parameters, which are used to set up both the DCS itself and the detector hardware, and output parameters, which are the values of the measurements and the status of the hardware of the experiment. The ATLAS-wide configuration database (ConfDB) will be used to store the first class, while parameters of the second class will be stored in the ATLAS conditions database (CondDB).

Two different types of set-up parameters are needed by DCS: static data, defining hardware and software of the DCS set-up, and variable data, describing the operation of the detector. The associated data volume is large because of the very many separate systems and the very high number of elements to be configured, of the order of 250 000. However, a high data-transfer rate is not required as the operations are not time-critical and will normally be performed at low rate during shutdown periods.

The variable data are used to configure the sub-detectors. Depending on the beam conditions of the LHC, different sets of operational parameters will be required for certain parts of the detector. The different types of data-taking run also require different operational parameters. All these sets of dynamic configuration data are loaded at boot-up time into the relevant DCS station. This operation is therefore also not time-critical. During running of ATLAS, updates of sub-sets may be needed. Hence access to the ConfDB is required at all times and it is important to guarantee the consistency of the data between the database and the configuration running in the sub-detector systems.

The output data of DCS are the measurements of the operational parameters of the detector and the infrastructure of the experiment, as well as the conditions and status of systems external to ATLAS (see Section 11.9), most notably the LHC accelerator. Much of these data are structured in very small entities; they essentially consist of the triplet definition, time, and value. The update frequency can be tuned individually and many quantities need only be stored when they change by more than a fixed amount. Nevertheless the total data volume is high and may reach 1 Gbyte per day. These data can be sent to the CondDB asynchronously, i.e. they can be held for small periods of time within DCS.

2.6 References

2-1 *ATLAS Inner Detector Technical Design Report*, CERN / LHCC / 97-16 and 97-17 (1997)

2-2 *ATLAS Pixel Detector Technical Design Report*, CERN / LHCC / 98-013 (1998)

2-3 *ATLAS Liquid Argon Calorimeter Technical Design Report*, CERN / LHCC / 96-41 (1996)

2-4 *ATLAS Tile Calorimeter Technical Design Report*, CERN / LHCC / 96-042 (1996)

2-5 *ATLAS Muon Spectrometer Technical Design Report*, CERN / LHCC / 97-022 (1997)

2-6 *ATLAS Level-1 Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

2-7 *3rd ATLAS ROD Workshop*, (2002)
http: / / wwwlapp.in2p3.fr / ROD2002 / ,
http: / / agenda.cern.ch / age?a021794

2-8 *ATLAS Technical Co-ordination Technical Design Report*, CERN / LHCC / 99-01 (1999)

2-9 *ATLAS HLT/DAQ/DCS Technical Proposal*, CERN / LHCC / 2000-17 (1999)

2-10 *Detector performance and physics commissioning with physics data*,
http: / / cern.ch / gianotti / commissioning.html

2-11 *ATLAS Technical Co-ordination, Detector Interface Group, DIG Forums*,
http: / / cern.ch / Atlas / GROUPS / DAQTRIG / DIG

3 System Operations

3.1 Introduction

This chapter describes how the TDAQ system will be used. While the main use of the ATLAS TDAQ system naturally refers to the periods of data taking, there are also operational aspects related to those periods of time when the LHC machine is off.

The chapter starts with a definition of what conditions have to be met in order for TDAQ to perform certain operations: for instance what conditions have to be met in order to take data from the detector. The definitions are given in terms of relationships between system states.

A large part of the chapter is dedicated to defining the data taking period, the run. In particular it discusses the different types of run, what operations are defined during a run and the transitions between one run and another.

The requirement for the TDAQ system to support multiple concurrent runs on different parts of the detector, partitions, is discussed next.

3.2 TDAQ states

In the context of the ATLAS experiment, operations involve three main actors: the ATLAS detector (the detector for short in the following), the LHC machine and the TDAQ system. The high-level operation of the experiment, and the relationships between the main actors defined above, are described in terms of states. A state is a concept which summarises what a part of the experiment is capable of doing at a certain point in time. States are also useful to describe how actors relate one to another. The further development of the concept of states, their refinement in terms of sub-states and their detailed relationships, is the subject of Chapter 12.

Three main states are useful to describe the way TDAQ can operate:

- Initial: in this state the TDAQ system is not capable of performing any useful operation for the experiment (apart from being able to communicate with the sub-systems for the purpose of initialization and configuration). The only operations allowed on TDAQ are those which bring it to a situation where data taking can be performed (see below). This may be the state into which TDAQ is for example after a power failure, or TDAQ may revert to this state in order to re-initialise large parts of ATLAS.
- Configured: in this state the TDAQ system is ready, provided other conditions are met (for example related to the detector or the LHC machine), to initiate a data taking session. This means that the various parts and components have been properly initialised and set. For the purpose of detailing the initialisation procedures, the configured state may be reached by means of intermediate states, related for instance to loading software into processors, configuring components, etc.
- Running: in this state the TDAQ system is taking data from the detector.

For the purpose of the global operation of the experiment, two states are associated to the detector:

- Ready: the detector can be used for data taking.
- Not Ready (or $\overline{\text{Ready}}$): the detector cannot be used for data taking.

Three states may also summarise, for the purpose of operating TDAQ, the conditions of the LHC machine:

- Stable-Beams: the LHC machine is in a condition which allows safe operation of the detector and beams are available to produce physics events.
- Detector-Safe: the LHC machine is in a condition which allows safe operation of the detector and beams are not available.
- Non Stable-Beams ($\overline{\text{Stable-Beams}}$): the LHC machine is not in a condition which allows operation of the ATLAS detector.

The diagrams of Figure 3-1 indicate the inter-relationships of the detector, machine and TDAQ states both for stable and non-stable beam conditions.

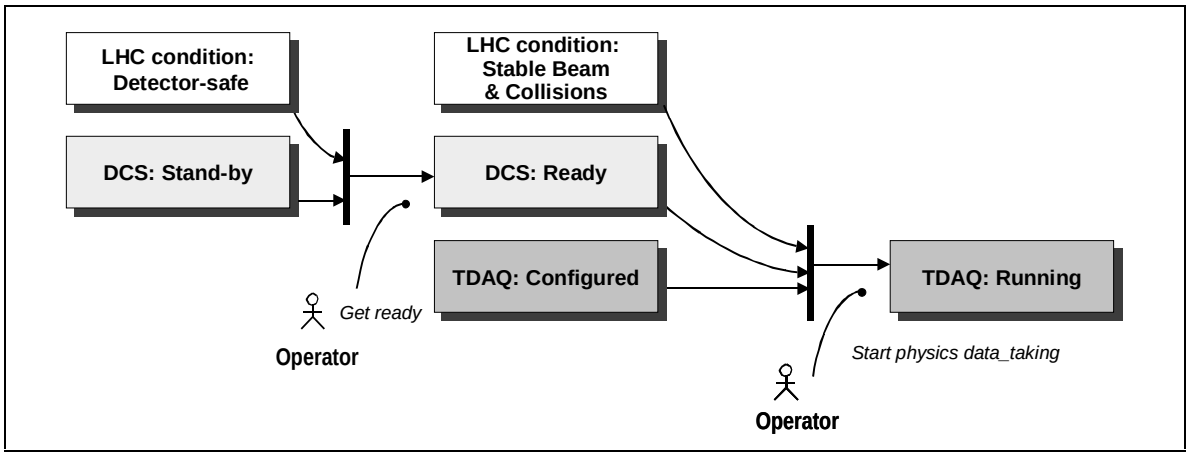

Figure 3-1 Inter-relationship of the detector, machine and TDAQ states for stable and non-stable bean conditions

3.3 The run

3.3.1 Run definition

A run is defined as a period of data taking in a TDAQ partition with a defined set of stable conditions (as defined in Chapter 1) related to quality of physics.

Whilst it is difficult to give today an exhaustive list of changes in running conditions that will force the starting of a new run, a number of items come immediately to mind: the parameters defining or affecting the selectivity of the triggers (LVL1, LVL2 and EF); the set of sub-detectors participating to the TDAQ partition, the operational parameters of sub-detectors. A modification of any of the above conditions forces a new run, that is the events following the change of the conditions are tagged with a new run number.

Conditions whose change force a new run are stored in a conditions database, whose contents are saved to permanent storage prior to the start of a new run.

Changes which do not force a new run include, for example, the removal or the insertion of processors or the disabling of FE channels (insofar as the physics is not affected)[1]. Those changes which do not force a new run number may not be stored in the conditions database. The change may be entered for example in an electronic experiment logbook if it affects the performance of the TDAQ system (e.g. removal of an EF processor) or tagged into the event if it affects the data from the detector. As an example of the latter: the removal of a detector or DAQ buffer may be flagged by appropriately setting a 'quality flag' in the fragment header.

A run, when conditions do not change, may extend throughout an entire machine fill.

3.3.2 Run Identification

The run number uniquely identifies a run (today it is a 32-bit number); it associates an event to a set of stable conditions. A run number is unique and is never re-used during the lifetime of the experiment. A run number is generated by a central service.

The proposed mechanism to tag events with the run number is based on the distribution of the run number (at the beginning of a run) to the ROD crate controllers. The RODs will then insert the run number into the fragment header for each event.

Any event fragment is partly identified, anywhere in the system, by its associated run number.

3.3.3 Event identification

Up to acceptance by LVL2 (or event building in the case of a partition without LVL2) an event is identified by an extended (32-bit) LVL1 ID (L1ID, generated by LVL1 as a 24-bit number and extended to 32 bits).

A Global Event Number (GID) uniquely identifies an event, accepted by the LVL2 trigger, within a run. It is generated by a central element (the DFM) after the LVL2 decision and it is made available, to be inserted into the event, to the element responsible for building the full event (the SFI).

Therefore an event, during the lifetime of ATLAS, is uniquely identified by the pair of 32-bit numbers consisting of the run number and the Global Event Number.

3.3.4 Performance requirement

The ATLAS TDAQ system is required to minimise its contribution to the experiment down time; although a quantitative definition of this requirement is not yet available, one can anticipate that the experiment down time due to TDAQ must be well below 1%.

There are two contributions to system down-time which are relevant to the subject of this chapter:

1. For example the number of FE channels (or ROBs) which can be removed from the read-out without affecting the physics is bounded by some threshold which is sub-detector dependent. When the amount of unavailable read-out exceeds the threshold, the physics is affected and the run should be stopped.

- the time spent by the system to initiate (start) or terminate (stop) a run, and
- the down time when coping with malfunctioning TDAQ components.

The two contributions above have an important impact on:

- how a run is defined, that is which configuration and parameter changes force a new run and which changes do not force a new run,
- how the transition between runs should be implemented, and
- how faults should be handled during a run.

3.3.5 Categories of runs

A data taking session on the ATLAS detector may be started for different purposes. A broad categorization of different types of runs follows.

Physics run: to record event data from the detector for the purpose of physics analysis. The participating actors in this kind of run are: all or part of the ATLAS detector, a fully functional (i.e. including the high level triggers) TDAQ, the DCS, the LHC machine, and the LVL1 trigger (including the Central Trigger Processor).

Detector calibration with beams: to calibrate (part of) an ATLAS sub-detector using data produced by the LHC beam in the sub-detector. The participating actors are a sub-set of those related to the physics run above: a sub-set of the ATLAS detector (a sub-detector partition, a complete sub-detector or some combination of sub-detectors), the LHC machine in 'stable beam' state, the first level trigger (see below) but not the high level triggers. However one can anticipate that the event filter infrastructure (e.g. a sub-set of the event filter farm) will be used for the purpose of running calibration software at the level of the complete event. As regards the first level trigger, the proper sub-set of the TTC system and the Local Trigger Processor for the sub-detector in question will be part of the data taking sub-system (a TDAQ partition as defined in Section 3.4).

Detector calibration without beams: this type of run needs the same actors as for detector calibration with beams above. There are two main differences, however: the LHC machine has to be in a 'detector safe' state, i.e. such that the detector may be safely switched on, and (depending on the detector being calibrated) the complete TDAQ functionality, or only the DCS functionality may be needed.

(Sub-)Detector commissioning: a type of run intended to put a sub-detector (and eventually the whole ATLAS detector) into an operational state. This may include also the initial run with cosmic rays. This type of run is similar to a calibration run with the exception that a 'stable beam' state for the LHC machine is emulated, in the case the LHC machine is not yet available. The high level triggers, as well as the Central Trigger Processor, may be part of a commissioning-like run, e.g. during the commissioning stage of the experiment.

The categories of runs above refer to the way the TDAQ system is globally set up for data taking. ***Calibration during a physics run*** is also required. This refers to a TDAQ system, set for a 'physics run', which also deals with special triggers, such as calibration triggers fired by a sub-detector during empty LHC bunches. These special triggers are marked as such, i.e. identified by a proper trigger type, and are dealt accordingly by the TDAQ system: for example a calibra-

tion trigger is always accepted by LVL2 and may be routed, by the Event Builder, to a specific Event Filter sub-farm for the purpose of being treated by dedicated software.

3.3.6 Operations during a Run

A TDAQ run is bracketed by two commands: one to start it and one to stop it (in the following referred to as 'Start' and 'Stop').

Start: the TDAQ system is in the 'configured' state (Section 3.2), the command is distributed to all the TDAQ elements. Any TDAQ element performs its own 'run start' sequence and then completes the transition to the 'running' state. When all the TDAQ elements have completed their transition to the 'running' state, the TDAQ system as a whole enters the 'running' state. At this point the first level trigger is enabled and events may start to flow in the system.

Stop: first the LVL1 triggers are disabled, then the command is distributed to all the TDAQ elements, each of them completes its task in an orderly way, in particular each element completes the processing of all the events in its queues and optionally produces an 'end of run' summary. Upon completion of its task, the TDAQ element re-enters the 'configured' state. When all the TDAQ elements have completed their transition to the stopped state, the TDAQ system as a whole enters the stopped state.

A need is foreseen for a 'Pause' command to interrupt data taking in order to perform some operation on the detector that will not force a new run. This could involve changes before LVL1 triggers are generated. The global system state associated to the temporary interruption of a run is called 'Paused'.

Two commands are available to respectively enter and exit the Paused state: ***pause*** and ***continue***. Pause and continue commands may be issued: by an operator or by software (viz. an expert system).

Pause: when the command is issued the LVL1 triggers are blocked, by raising the global busy signal. All TDAQ elements are issued with Pause command. Each element will execute it locally as soon as the handling of the current event is terminated (i.e. TDAQ elements will not empty their buffers before entering the paused state.) TDAQ completes the transition to Paused as soon as all the TDAQ elements have entered the Paused state.

Continue: when the continue command is issued: all TDAQ elements are issued the continue command, each element returns to the running state. TDAQ completes the transition to the running state as soon as all the TDAQ elements have returned to the running state. At this point LVL1 triggers are unblocked.

Another special command will be available for a running TDAQ system, the Abort command. This command is reserved for very special cases and it entails a fast termination of the run; for example TDAQ elements will not complete the processing of events in their buffers.

3.3.7 Transition between Runs

Prior to the start of a run, or as the immediate consequence of a run stop command, the LVL1 Busy is asserted. The LVL1 Busy is removed upon the transition to the running state. This latter

implies that all[1] TDAQ elements have completed their local execution of the run start command.

The completion of a run is also a process which needs synchronous local processing of all[1] the TDAQ elements: they receive the stop command, complete the processing of the contents of their buffers, produce end of run statistics etc. and leave the running state.

In addition to the run control command, which signals a TDAQ element when a run is requested to complete, a mechanism is necessary to determine when the last fragment or event of the terminating run has been processed. This mechanism cannot be part of the run control as it is tightly related to the flow of the event data in the detector front-end buffers and in the TDAQ system. A time-out will be used for this purpose: a TDAQ element will consider that the last event has been processed when 1) it has received the 'stop run' command and 2) it has not received events for a certain time (for example a time of the order of 10 seconds).

The transition between two runs (i.e. stopping the previous and starting the next) includes two potentially time consuming processes:

- the completion of the processing of the contents of all the fragment / event buffers in the system: front-end buffers, RODs, ROBs, LVL2 and EF nodes;
- the synchronisation of all[1] the TDAQ elements to complete the transition stopped / running or running / stopped. That is, before the TDAQ partition may complete a state transition, all[1] the TDAQ elements have to have completed the transition locally.

Under certain conditions the values of some parameters such as LVL1 trigger masks, thresholds and pre-scaling factors or sub-detector calibration operating parameters, may need to change relatively often, maybe several times per machine fill, with each change forcing a new run.

This could happen in the case of calibration runs, when some detector operating parameter may be required to change frequently.

In these cases the transition between runs as defined in Section 3.3.6 is not adequate in terms of the potentially long TDAQ system down time. A more efficient transition between runs is required and we define:

Checkpoint

> a transition in a running TDAQ system, triggered by a change in conditions or by an operator, which 1) results in the following events to be tagged with a new run number and 2) does not need the synchronisation, via run control start / stop commands, of all TDAQ elements.

The checkpoint transition is intended for those changes in conditions which require that events be correlated to the new conditions via a new run number but the change has a light implication on most of TDAQ. It is a mechanism to associate a new run number to events characterised by new conditions with minimal synchronisation within TDAQ.[2]

1. It maybe envisaged that, in the case of the LVL2 and the EF, only a (to be defined) percentage of the farm needs to successfully perform the transition. The rest may do it 'in the background' and join the new run afterwards.
2. It should be noted that, for a transient time, events belonging to more than one run could be simultaneously present in the system. In particular given that LVL2 accepts are not time ordered, an EF node might have to process events belonging to two (or in principle even more) different runs.

A checkpoint transition is started automatically by the TDAQ control system when certain conditions are modified; it may also be initiated manually by an operator or automatically by some other software component (viz. an expert system). The level-1 triggers are blocked upon entering the checkpoint transition and re-enable prior to completing the checkpoint transition.

The main feature of the checkpoint transition is the fact that events keep flowing in the system continuously. Therefore a mechanism is needed for a TDAQ element to detect when the new run begins. That is when the new run number becomes applicable, when the Global Event ID should be reset to 0 and when a TDAQ element should perform run completion processing and the initialisation necessary for a new run (for example a LVL2 processor may require to read the new conditions). The run number may be used for this purpose, i.e. a TDAQ element recognises a new run whenever a piece of data (fragment or full event) is tagged with a new run number. TDAQ elements may therefore perform the 'transition' from the old to the new run at their own pace and time. It is remarked that the same mechanism is also applicable to analysis and monitoring software dealing with a statistical sample of the event data: e.g. a monitoring program recognises a new run whenever it samples an event with a new run number (with the caveat that, as for EF processing units, programs sampling events after LVL2 might have to handle events belonging to more than one run).

3.4 Partitions and related operations

A list of the different ways the TDAQ system may be subdivided follows; each definition corresponds to a specific function [3-2].

- **TTC Partition**. A TTC partition includes a part of the TTC system and a corresponding part of the ROD-BUSY feedback tree. A TTC partition corresponds to a single TTCvi module. The concept of a TTC partition is already present in the LVL1 system [3-3].
- **TDAQ resource**. A TDAQ resource is a part of the ATLAS TDAQ system which can be individually disabled (masked out of the ATLAS TDAQ), and possibly enabled, without stopping the data taking process. A single ROB and a single HLT processing unit are examples of TDAQ resources.
- **TDAQ segment**. A TDAQ segment is defined as a set of TDAQ system elements that can be configured and controlled[1] independently from the rest of the TDAQ system. A TDAQ segment can be dynamically removed from / inserted into an active TDAQ partition without stopping the run. An event filter sub-farm and a single ROD crate are examples of TDAQ segments.
- **TDAQ partition**. It is a sub-set of the ATLAS TDAQ system for the purpose of data taking.The full function of the ATLAS TDAQ is available to a sub-set of the ATLAS detector. The data taking from the LAr EMB A sub-detector, including the corresponding TTC partition, the read-out associated to the EMB A sub-detector, part of the event filter sub-farm (to run e.g. calibration software) constitutes an example of TDAQ partition.

The last three definitions introduce three independent concepts for the ATLAS TDAQ system: resources can be disabled, segments can be removed and operated independently, and partitions are fully functional TDAQ systems running on a sub-set of the ATLAS detector. There

1. That is the segment is capable of receiving commands from the TDAQ control system.

are elements which are neither a resource nor a segment (for example the RoIB), therefore the classification above does not define a hierarchy within the ATLAS TDAQ.

The term **partition** is reserved for the concept of a TDAQ partition. An equivalent formulation for a TDAQ partition is that the TDAQ system must be capable of running as multiple (fully functional[1]), possibly concurrent, instances each operating on sub-sets of the ATLAS detector. There is a direct correspondence between TDAQ partitions and TTC partitions: these latter define how TDAQ is physically partitionable. For example the LAr sub-detector has six TTC partitions associated to it, hence no more than six independent TDAQ partitions may be run concurrently on the LAr detector.

There exists one TDAQ Partition which covers the whole ATLAS TDAQ system. That TDAQ Partition is used for physics data taking at the experiment. A TDAQ Partition always includes some FE elements of one or more sub-detectors in order to provide data[2]. A TDAQ Partition may stop at the level of one (or more) ROD crate(s) (ROD Crate DAQ), one ROS, one or more SFIs or one or more Event Filter sub-farms.

TDAQ partitions are properly (i.e. such that they result in a system which is capable of data taking) defined sets of TDAQ components. TDAQ partitions may be:

- Defined: the process of relating together the required TDAQ components in order to obtain a runnable system, including the definition of the sub-set of detector read-out associated to the TDAQ partition. The definition process will make sure that the definition of the TDAQ partition is consistent (i.e. it contains all that is needed). At the level of the definition, there is no need for different TDAQ partitions to be disjoint: for example two different TDAQ partitions may be defined to share parts of TDAQ (e.g. the read-out or part of the event filter farm).
- Activated: a defined TDAQ partition may be activated, this means that the TDAQ components associated to it are booked for use. In this case the system will check that the TDAQ partition being activated will not require any component which is already booked by another active TDAQ partition. In other words TDAQ partitions are required to be independent at the time of activation.
- Deactivated: the booking of the TDAQ components associated to the TDAQ partition is released. Other TDAQ partitions that may need those elements can at this moment be activated.
- Joined: the components from two (or more) existing (i.e. defined) TDAQ partitions can be merged together in order to make a larger (i.e. including more sub-detectors) TDAQ partition.
- Split: this is the reverse operation, with respect to joining TDAQ partitions. Two, or more, smaller TDAQ partitions are created out of an existing one.

It is remarked that the ‘Join’ and ‘Split’ operations do not affect the pre-existing TDAQ partitions: that is to say those which are merged and, respectively, split.

1. That is the complete functionality of the DAQ system is available to run a subset of the detector.
2. In one exceptional case, that of the DAQ partition, the TDAQ partition may include simulated input data instead of FE elements.

3.5 Operations outside a run

Outside a run, the operations allowed on the TDAQ system fall into two main categories: those operations which are performed between runs (with or without the LHC machine on) and those which are performed during the LHC shutdown period. More detail is available in Chapter 12.

Operations between runs: they typically fall in the range of initialisation (e.g. firmware loading), configuration (e.g. setting up of application parameter values) and partition management.

Operations during shutdown: there will be the need to run the TDAQ system for calibration, commissioning and test purposes, and the need for continuous operation of the DCS system.

3.6 Error handling strategy

The ATLAS TDAQ system will consist of a large number of hardware and software elements: a few thousand processors, each running possibly several software processes. Today a model for the Mean Time To Failure (MTTF) of the full TDAQ, or any of its components, is not available. However it is safe to assume that malfunctioning[1] in a system of such a size may happen at a non negligible rate. Under this assumption and taking into account the requirement of maximising the TDAQ up-time, a two-pronged strategy to address faults in the system is presented. The distinction is made between faults which are, respectively are not, fatal for a data taking session.

- The fault happens in an element in such a way that this latter can be removed from the running system without affecting the physics, that is to say once the element is removed, it is still meaningful to continue the run. It is the responsibility of the sub-system, to which the element belongs, to remove the element transparently without stopping the run. When necessary the sub-system will tag subsequent events with a ‘quality flag’ which marks events as ‘degraded’. Examples are: a ROD, a ROB (both require the tagging with a ‘quality flag’), a LVL2 or an EF processor or even an entire farm (which do not require the quality flag).
- The fault happens in an element which is either essential (e.g. the DFM or the RoIB of the baseline) or such that it must respond to a high rate of requests (e.g. the ROS today). A fault in one of these elements is either fatal for the run or it may potentially generate a long and disrupting sequence of errors in related parts of TDAQ (for example, a ROS which fails to respond will generate a large number of time-out errors in the LVL2 system, which in the most optimistic scenario will degrade performance). The ‘simpler’ fault tolerance as defined above (i.e. the ‘self-healing’ capability of a sub-system) is not applicable in this case. These TDAQ elements have to be identified and ought to be designed with a higher degree of reliability (i.e. predicting a reasonable MTTF for them).

1. Hardware fault (e.g. a faulty fan, power supply, disk, etc.) or software fault (e.g. a process aborting).

3.7 Data bases

The issue of data bases is central to the whole ATLAS TDAQ system, as the means to permanently record data (event data but also the detector and machine status) and to permanently hold the information necessary to initialise, configure and run the system. In this respect, and as discussed in Section 1.4, there is a number of broad categories of data to be permanently stored:

- Configuration data: information necessary to the configuration of TDAQ components and the related software as well as the ATLAS detector hardware. The management of this information is under the responsibility of TDAQ. The configuration information is used prior to the start of a run, during the TDAQ initialisation and configuration phases and while the run is being started. The configuration information is mostly static during data taking will only be changed in the case of serious hardware and/or software problems. In this latter case the TDAQ components which need the updated information will be notified of the changes (so as to be able to reconfigure themselves).
- Conditions data: this is, in some sense, the 'offline database', that is to say it contains all the information necessary to the reconstruction and analysis software as well as the information concerning the detector behaviour (as recorded by DCS). In particular it contains information necessary for the initialisation and correct functioning of the HLT algorithms. The responsibility for this database is outside TDAQ, the latter being a user of the database. TDAQ acts both as a consumer of the conditions database, as regards initialisation and configuration of the HLT processes for example, and a producer, in the case of DCS and calibration procedures (both produce information, e.g. detector status, which has to be stored as conditions).
- Event data: the data relative to the events produced in the ATLAS detector. This information is produced by TDAQ, which is responsible to record them on local, permanent storage. The contents of the local, permanent storage are transferred later (possibly asynchronously with respect to the data taking process) to an ATLAS central permanent event data repository.
- Monitoring data: TDAQ performs operational (i.e. of TDAQ itself) and event (i.e. of the detector) monitoring. Results, for example in the form of histograms, will be stored for later use (e.g. comparisons).
- Logbook data: the historical record of the experiment which includes the information produced by TDAQ, e.g. state transition, and via TDAQ, e.g. error conditions.

Table 3-1 Configuration and conditions data volume requirements for some TDAQ components

Component	Number (order of magnitude)	Configuration data volume at initialisation per individual component	Conditions data volume at initialization per individual component	Operational monitoring data rate
ROD Crate	100	few kbytes	N/A	O(1) kbyte/s
ROS	150	few kbytes	N/A	O(1) kbyte/s
LVL2 Processing Unit	500	few kbytes	50 Mbyte	O(1) kbyte/s
Event Filter Processing Unit	1600	few kbytes	100 Mbyte	O(1) kbyte/s

The amount of potential data producers and consumers, that is to say the total number of TDAQ components which are users of data bases, is of the order of several thousands (from ROD crates to Event Filter processors). Each is expected to consume and/or produce variable amounts of data (from kbytes to several tens of Mbytes); an order of magnitude view of the problem, for some TDAQ component, is shown in Table 3-1.

3.8 References

3-1 ATLAS TDAQ/DCS Global Issues Working Group, *Run and States*, ATLAS Internal Note, ATL-COM-DAQ-2003-003 (2002)

3-2 ATLAS TDAQ/DCS Global Issues Working Group, *Partitioning*, ATLAS Internal Note, ATL-COM-DAQ-2003-004 (2002)

3-3 *ATLAS Level-1 Trigger Technical Design Report*, CERN/LHCC/98-14 (1998)

4 Physics selection strategy

This chapter provides an overview of the strategy for the online selection of events in ATLAS. The challenge faced at the LHC is to reduce the interaction rate of about 1 GHz at the design luminosity of $1 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ online by about seven orders of magnitude to an event rate of O(100 Hz) going to mass storage. Although the emphasis in this document will be on the contribution of the HLT to the reduction in rate, the final overall optimization of the selection procedure also includes LVL1.

The first section describes the requirements defined by the physics programme of ATLAS. This is followed by a discussion of the approach taken for the selection at LVL1 and HLT. Next, a brief overview of the major selection signatures and their relation to the various detector components of ATLAS is given. Then, an overview of the various parts of the trigger menu for running at an initial luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ is presented, together with a discussion of the expected physics coverage. The discussion in this chapter concentrates on the initial luminosity regime; the selection strategy for the design luminosity phase will crucially depend on the observations and measurements during the first years of data taking. This is followed by a description of how changes in the running conditions are going to be addressed, and finally ideas for the strategy of determining trigger efficiencies from the data alone are presented.

Details on the implementation of the event-selection strategy, in terms of the software framework to perform the selection, can be found in Section 9.5. More information on selection-algorithm implementations and their performance in terms of signal efficiency and background rejection are given in Chapter 13. Finally, Chapter 14 addresses the issue of system performance of the online selection, presenting our current understanding of the resources (e.g. CPU time, network bandwidth) needed to implement the selection strategy presented in this chapter.

4.1 Requirements

The ATLAS experiment has been designed to cover the physics in proton–proton collisions with a centre-of-mass energy of 14 TeV at LHC. Amongst the primary goals are the understanding of the origin of electroweak symmetry breaking, which might manifest itself in the observation of one or more Higgs bosons, and the search for new physics beyond the Standard Model. For the latter it will be of utmost importance to retain sensitivity to new processes which may not have been modelled. The observation of new heavy objects with masses of O(TeV) will involve very high-p_{T} signatures and should not pose any problem for the online selection. The challenge is the efficient and unbiased selection of lighter objects with masses of O(100 GeV). In addition, precision measurements of processes within and beyond the Standard Model are to be made. These precision measurements will also provide important consistency tests for signals of new physics. An overview of the variety of physics processes and the expected performance of ATLAS can be found in [4-1]. Most of the selection criteria used in the assessment of the physics potential of ATLAS are based on the selection of at most a few high-p_{T} objects, such as charged leptons, photons, jets (with or without b-tagging), or other high-p_{T} criteria such as missing and total transverse energy. Furthermore, ATLAS expects to take data during the heavy-ion running of the LHC.

The online event-selection strategy has to define the proper criteria to cover efficiently the physics programme foreseen for ATLAS, while at the same time providing the required reduction in event rate at the HLT. Guidance on the choice of online selection criteria has been obtained from

the variety of analyses assessing the ATLAS physics potential, aiming for further simplification to a very few, mostly inclusive, criteria.

Event selection at LHC faces a huge range in cross-section values for various processes, as shown in Figure 4-1. The interaction rate is dominated by the inelastic part of the total cross-section with a cross-section of about 70 mb. The inclusive production of b-quarks occurs with a cross-section of about 0.6 mb, corresponding to a rate of about 6 MHz for design luminosity. It is worth noting that the cross-section for inclusive W production, including the branching ratio for the leptonic decays to an electron or a muon, leads to a rate of about 300 Hz at design luminosity. The rate of some rare signals will be much smaller, e.g. the rate for the production of a Standard Model Higgs boson with a mass of 120 GeV for the rare-decay mode into two photons will be below 0.001 Hz. The selection strategy has to ensure that such rare signals will not be missed, while at the same time reducing the output rate of the HLT to mass storage to an acceptable value.

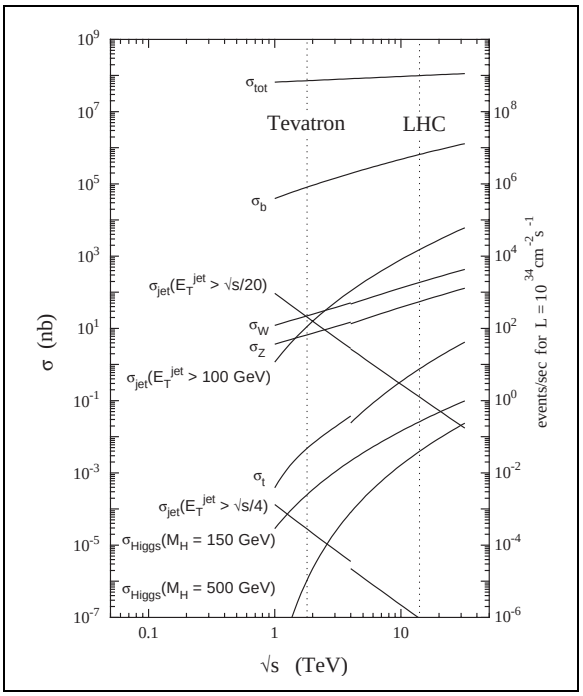

Figure 4-1 Cross-section and rates (for a luminosity of 1×10^{34} cm^{-2} s^{-1}) for various processes in proton–(anti)proton collisions, as a function of the centre-of-mass energy.

The online selection thus has to provide a very efficient and unbiased selection, maintaining the physics reach of the ATLAS detector. It should be extremely flexible in order to operate in the challenging environment of the LHC, with up to about 23 inelastic events per bunch crossing at design luminosity. Furthermore, it has also to provide a very robust, and, where possible, redundant selection. It is highly desirable to reject fake events or background processes as early as possible in order to optimize the usage of the available resources. Presently the selection is based on rather simple criteria, while at the same time making use of the ATLAS capabilities to reject most of the fake signatures for a given selection. It is, however, mandatory to have additional tools such as exclusive criteria or more elaborate object definitions available for the online selection.

4.2 Selection criteria

In order to guarantee optimal acceptance to new physics within the current paradigm of particle physics, we have taken an approach based on emphazising the use of inclusive criteria for the online selection, i.e. having signatures mostly based on single- and di-object high-p_T triggers. Here 'high-p_T' refers to objects such as charged leptons with transverse momenta above O(10 GeV). The choice of the thresholds has to be made in such a way that a good overlap with the reach of the Tevatron and other colliders is guaranteed, and there is good sensitivity to new light objects, e.g. Higgs bosons. Enlarging this high-p_T selection to complement the ATLAS physics potential requires access to signatures involving more exclusive selections, such as requiring the presence of several different physics objects or the use of topological criteria. A fur-

ther example is the use of charged particles with transverse momenta of O(1 GeV) for the selection of b-hadron physics.

The selection at the HLT will in most cases be seeded by the information already obtained at LVL1 (i.e. Regions-of-Interest, RoIs) and will exploit the complementary features of the LVL2 trigger and the EF selection. At LVL2, a fast rejection has to be achieved, using dedicated algorithms to meet the latency constraints. These algorithms will, in most cases, require only a few per cent of the event data, thanks to the guidance of RoIs from LVL1. The selection signatures will be refined at the EF, where the full event is available for analysis, using more precise and detailed calibration and alignment parameters and having fewer constraints on the latency. Also the EF processing can profit from the results of the earlier trigger stages, for example, using the results of LVL2 for seeding the EF processing. Furthermore, the EF will provide classification of the accepted events, in order to facilitate offline analyses. This might involve the reconstruction of further objects, which are not used for the event selection.

Although the selection will be predominantly based on simple and inclusive signatures to allow for optimization at the analysis level, more refined selection tools have to be used in addition. These include the use of more elaborate algorithms, e.g. b-tagging of jets, and the application of more exclusive criteria, in order to enrich the available samples for certain physics processes, for example. Furthermore, it is highly desirable to select events with several complementary criteria, in order to better control possible biases due to the online selection.

4.3 Physics objects for event selection

The ATLAS trigger relies on the concept of physics ‘objects’ (muons, electrons, jets, etc.). Candidate objects are typically first identified and crudely reconstructed at LVL1. Processing in the HLT progressively refines the reconstruction, rejecting fake objects and improving the precision on measured parameters such as E_{T}.

The physics objects used in the HLT can be based on information from all sub-detectors of ATLAS, at full granularity. As mentioned above, the difference in the reconstruction of these objects between LVL2 and the EF refers mostly to the complexity of the algorithms interpreting the raw data, and on the level of detail and accuracy of the alignment and calibration information used. The EF has the full event at its disposal to the search for these objects, although it is anticipated that the processing will typically be initiated using the detailed LVL2 result to seed the algorithms.

ATLAS, as a multi-purpose detector, will have charged-particle tracking in the Inner Detector covering the pseudorapidity region of $|\eta| < 2.5$ inside a solenoidal magnetic field of 2 T and fine-grained calorimeter coverage for $|\eta| < 2.4$, especially in the electromagnetic compartments, which extends up to $|\eta| < 3.2$. The forward calorimeter completes the calorimetric coverage up to $|\eta|$ of 4.9 for the measurement of missing transverse energy and forward jets. The coverage of the muon system extends up to $|\eta| = 2.4$ for the trigger chambers and up to $|\eta| = 2.7$ for the precision muon chambers. More details on the various components and their expected performance can be found in [4-1].

The following overview briefly summarizes the most important physics objects foreseen for use in the HLT. More details on the concrete implementation of the selection algorithms and their expected performance are given in Chapter 13.

- Electron (within $|\eta| < 2.5$): the selection criteria for electrons will include a detailed shower-shape analysis in the fine-grained electromagnetic compartments of the LAr calorimeters (within the range of $|\eta| < 2.4$), a search for high-p_T tracks, and a match between the clusters and tracks. Further refinement is possible via the identification and correct treatment in case the electron has undergone Bremsstrahlung, the requirement of a transition radiation signal from the transition radiation tracker (TRT), and the application of isolation criteria.
- Photon (within $|\eta| < 2.5$): the selection of photons will also be based on a detailed calorimeter shower-shape analysis, including the requirement of isolation, and possibly on the use of a veto against charged tracks, after it has been checked that the photon did not convert in the material of the Inner Detector into an e^+e^- pair.
- Muon (within $|\eta| < 2.4$): the muon selection will in a first step make use of the external muon spectrometer to determine the muon momentum and charge. A refinement of this information will then be obtained by searching for tracks in the Inner Detector and matching and combining these candidates with the muon track segment. Isolation criteria may also be applied, using, for example, information from the calorimeters.
- Tau (within $|\eta| < 2.5$): the selection of taus in their hadronic decay modes will use calorimeter shower-shape analysis to identify narrow hadronic jets. These can be matched to one or more tracks found in the Inner Detector. As above, isolation criteria may also be applied, using, for example, information from the calorimeters.
- Jet (normally within $|\eta| < 3.2$): the jet selection will be based mostly on calorimeter information, which might be refined by including the information from matching charged tracks. A search can also be made for jets in the forward calorimeter, covering $3.2 < |\eta| < 4.9$.
- b-tagged jet (within $|\eta| < 2.5$): the selection will start from jets already selected. The associated tracks found in the Inner Detector are used to search, for example, for large values of the impact parameters or for the presence of secondary vertices, and/or for soft (i.e. low-p_T) leptons.
- E_T^{miss} (within $|\eta| < 4.9$): the definition of missing transverse energy will be based on the full calorimeter data, allowing for improvements via the inclusion of information from observed muons.
- Total ΣE_T (within $|\eta| < 4.9$): again the calculation will be based on the full calorimeter information, with additional corrections possible from reconstructed muons. An alternative definition of the total ΣE_T can be obtained using only reconstructed jets.

An example of an exclusive selection, which requires a complex approach, is the case of b-hadron physics. Here it is necessary to reconstruct online in a (semi-)exclusive way the b-hadron decay modes of interest. This requires the reconstruction and identification of low-p_T charged hadrons and leptons (electrons and muons), where guidance from LVL1 is not always available with optimal efficiency. At LVL1 the trigger will demand the presence of at least one low-p_T muon.

4.4 Trigger menu

In this section, the present understanding of the trigger menu for running at an initial peak luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ is presented. Distinction is made between different parts of this trigger menu which are discussed separately:

- inclusive physics triggers, which form the backbone of the online selection and are chosen to guarantee the coverage of a very large fraction of the ATLAS physics programme;
- prescaled physics triggers, which will extend the physics coverage for ATLAS, by having, for example, inclusive selections with lower thresholds to enlarge the kinematic reach, and provide samples for understanding background processes and detector performance;
- exclusive physics triggers, which will also extend the physics coverage for ATLAS; and
- dedicated monitor and calibration triggers, not already contained in one of the above items, for improving the understanding of the performance of the ATLAS detector, based on physics events not needed otherwise for physics measurements. Furthermore, specific selections might be used to monitor the machine luminosity.

The description of these four parts of the current trigger menu is followed by a discussion of the physics coverage achieved, including indications about dependence of the acceptance for several physics processes on the threshold values. A more detailed discussion of the various parts of the trigger menu can be found in [4-2].

The derivation of the trigger menus and the threshold values starts from the physics-analysis requirements, followed by an assessment of the rejection capabilities at the various selection stages, and finally taking into account estimates of the total HLT output bandwidth. This procedure is iterative in order to include existing information, e.g. from studies of the LVL1 trigger (see Ref. [4-3]), or from past studies of the HLT performance, as documented, for example, in Ref. [4-4].

Not discussed in any detail in this document are possible trigger selections for the study of very-forward physics or heavy-ion interactions, as these additional aspects of the ATLAS physics potential are presently only under investigation. The flexibility in the online selection will, however, allow these physics processes to be addressed. As the excellent capabilities of ATLAS for identifying high-p_{T} signatures are expected to play an important role in the study of these physics processes (especially for the study of heavy-ion interactions), the selection strategy in this document should be extremely useful for these environments as well.

It should be noted that the menus will evolve continuously, benefiting from a better understanding of the detector, and the experience gained when commissioning the experiment. Further progress in the understanding of the Standard Model and future discoveries prior to the start of the LHC might influence the contents of the trigger menu.

In the narrative that follows, labels of the form ‘NoXXi’ will be used to identify specific trigger items. Here ‘N’ is the minimum number of objects required, and ‘o’ indicates the type of the selection (‘e’ for electron, ‘γ’ for photon, ‘μ’ for muon, ‘τ’ for a τ hadron, ‘j’ for jet, ‘b’ for a b-tagged jet, ‘xE’ for missing transverse energy, ‘E’ for total transverse energy, and ‘jE’ for the total transverse energy obtained using only jets). ‘XX’ gives the threshold in transverse energy (in units of GeV), and ‘i’ indicates an isolation requirement. As an example, 2μ20i refers to the requirement of at least two muons, with an E_{T} threshold of 20 GeV each, fulfilling isolation criteria. The thresholds indicate the transverse-energy value above which the selection has good efficiency for true objects of the specified type. The exact value for the efficiency obtained depends on the

implementation of the algorithm and the details of the criteria applied, examples of which are given in Chapter 13.

A comprehensive assessment of the expected rates for the trigger menu will be given in Section 13.5, both for LVL1 and for the HLT, including the expected total data rate for recording the accepted events to mass storage.

4.4.1 Physics triggers

Table 4-1 gives an overview of the major selection signatures needed to guarantee the physics coverage for the initial running at a peak luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$.

Table 4-1 Trigger menu, showing the inclusive physics triggers. The notation for the selection signatures and the definition of the thresholds are explained in Section 4.4.

Selection signature	Examples of physics coverage
e25i	$W \rightarrow e\nu$, $Z \rightarrow ee$, top production, $H \rightarrow WW^{(*)}/ZZ^{(*)}$, W',Z'
2e15i	$Z \rightarrow ee$, $H \rightarrow WW^{(*)}/ZZ^{(*)}$
μ20i	$W \rightarrow \mu\nu$, $Z \rightarrow \mu\mu$, top production, $H \rightarrow WW^{(*)}/ZZ^{(*)}$, W',Z'
2μ10	$Z \rightarrow \mu\mu$, $H \rightarrow WW^{(*)}/ZZ^{(*)}$
γ60i	direct photon production, $H \rightarrow \gamma\gamma$
2γ20i	$H \rightarrow \gamma\gamma$
j400	QCD, SUSY, new resonances
2j350	QCD, SUSY, new resonances
3j165	QCD, SUSY
4j110	QCD, SUSY
τ60i	charged Higgs
μ10 + e15i	$H \rightarrow WW^{(*)}/ZZ^{(*)}$, SUSY
τ35i + xE45	$qqH(\tau\tau)$, $W \rightarrow \tau\nu$, $Z \rightarrow \tau\tau$, SUSY at large tan β
j70 + xE70	SUSY
xE200	new phenomena
E1000	new phenomena
jE1000	new phenomena
$2\mu6 + \mu^+\mu^-$ + mass cuts	rare b-hadron decays ($B \rightarrow \mu\mu X$) and $B \rightarrow J/\psi\,(\psi')X$

A large part of the physics programme will rely heavily on the inclusive single- and di-lepton triggers, involving electrons and muons. Besides selecting events from Standard Model processes — such as production of W and Z bosons, gauge-boson pairs, $t\bar{t}$ pairs, and the Higgs boson — they provide sensitivity to a very large variety of new physics possibilities, for example new heavy gauge bosons (W', Z'), supersymmetric particles, large extra dimensions (via the Drell-

Yan di-lepton spectrum), and Higgs bosons in extensions of the Standard Model such as the MSSM. These triggers also select particle decays involving τ's where the τ decays leptonically.

The inclusive single- and di-photon triggers will select a light Higgs boson via its decay $H \rightarrow \gamma\gamma$ as well as some exotic signatures (e.g. technicolour).

The coverage for supersymmetry is extended by using the jet + missing-transverse-energy signatures as well as multi-jet selections; the multi-jet selection is especially relevant in case of R-parity violation. The inclusive single- and di-jet triggers will, for example, be used in the search for new resonances decaying into two jets. Further sensitivity to supersymmetry at large values of tan β will be provided by signatures involving a hadronic τ selection.

Rare b-hadron decays and b-hadron decays involving final states with a J/ψ are selected by a di-muon signature (requiring opposite charges) and additional invariant-mass cuts.

4.4.2 Prescaled physics triggers

Table 4-2 shows a prototype for additional contributions to the trigger menu in the form of prescaled physics triggers. These triggers extend the physics coverage of the online selection by extending the kinematic reach of various measurements towards smaller values, e.g. of the transverse momentum in a process.

Table 4-2 Examples of additional prescaled physics triggers

Selection signature	Examples of physics motivation
single jets (8 thresholds between 20 and 400 GeV)	inclusive jet cross-section measurement
di-jets (7 thresholds between 20 and 350 GeV)	di-jet cross-section measurements
three jets (6 thresholds between 20 and 165 GeV)	multi-jet cross-section measurements
four jets (5 thresholds between 20 and 110 GeV)	multi-jet cross-section measurements
single electrons (5–6 thresholds between 7 and 25 GeV)	inclusive electron cross-section
di-electrons (2 thresholds between 5 and 15 GeV)	
single muons (6 thresholds between 5 and 20 GeV)	inclusive muon cross-section
di-muons (2 thresholds between 5 and 10 GeV)	
single photons (7–8 thresholds between 7 and 60 GeV)	inclusive photon cross-section
di-photons (2 thresholds between 7 and 20 GeV)	
taus (4 thresholds between 25 and 60 GeV)	
di-tau (2 thresholds between 25 and 35 GeV)	$Z \rightarrow \tau\tau$ selection
xE (5 thresholds between 45 and 200 GeV)	
E (3 thresholds between 400 and 1000 GeV)	
jE (3 thresholds between 400 and 1000 GeV)	
filled-bunch-crossing random trigger	minimum bias events, trigger efficiency

A typical example for the application of these trigger selections is the measurement of the jet cross-section over the full kinematic range, starting from the lowest achievable E_T values up to the region covered by the un-prescaled inclusive jet trigger. In addition, these pre-scaled triggers together, with the pre-scaled minimum bias selection, will be crucial in determining trigger efficiencies from the data, as discussed in Section 4.6.

The prescale factors to be applied to most of the selections shown in Table 4-2 will have to be determined on the basis of the desired statistical accuracy for the given application, taking into account the total available data bandwidth and computing resources. The E_T ranges for the thresholds should be seen as indicative only. The aim will be to cover the widest possible range, i.e. extending the coverage from the thresholds un-prescaled selections down to the lowest possible ones for a given signature. The values of the prescale factors will evolve with time. These triggers will be of particularly high importance in the early phases of the data taking after the start-up of the LHC.

4.4.3 Exclusive physics triggers

A list of further selections using more exclusive criteria is presented in Table 4-3.

Table 4-3 Examples of additional exclusive physics triggers

Selection signature	Physics motivation
e20i + xE25	$W \rightarrow e\nu$
μ20 + γ10i	lepton flavour violation
e / μ + jet	
forward jet + xE	invisible Higgs decays
b-jet (various multiplicities)	
μ8 + B-decays	b-hadron physics

An example is the extension of the selection for b-hadron physics to involve more decay modes. As discussed in more detail in Ref. [4-1] and Ref. [4-4], ATLAS offers the possibility of performing several measurements of CP-violation in the b-hadron system. The selection strategy relies on selecting $b\bar{b}$ production via the semi-muonic decay of one of the b-quarks, and then making a (semi-)exclusive selection of the decays of interest based on reconstructing low-p_T charged particles in the Inner Detector. The best efficiency would be achieved by reconstructing the tracks at LVL2 without guidance from LVL1. However, in view of resource limitations, RoI-driven reconstruction is also being considered.

For the first data taking, it will be essential that the full detector is read out to mass storage for all triggers in order to have the full spectrum of information available for offline studies. It is, however, envisaged that at a later stage some of the above prescaled triggers might no longer require recording of the full detector information, and thus more bandwidth could be made available for further selection criteria.

4.4.4 Monitor and calibration triggers

The selection signatures presented in Table 4-1, Table 4-2 and Table 4-3 will provide a huge sample of physics events which will be of extreme importance for the understanding and continuous monitoring of the detector performance. In particular, the leptonic decays of the copiously-produced Z bosons will be of great help. One will use $Z \rightarrow ee$ events to determine the absolute energy scale for electrons and to intercalibrate the electromagnetic parts of the calorimeter. The muonic decay will set the absolute momentum scale, both in the Inner Detector and in the muon spectrometer. In addition, the inclusive electron and muon triggers will select $t\bar{t}$ events via the semi-leptonic decay of one of the top quarks. This sample can be used to set the jet energy scale, using di-jet decays of the W's produced in the top quark decays, and also to determine the b-tagging efficiency.

Table 4-4 Examples of specific monitor and calibration triggers, based on physics events, which are not covered in Table 4-1, Table 4-2 and Table 4-3

Selection signature	Example for application
random trigger	zero bias trigger
unpaired-bunch-crossing random trigger	background monitoring
e25i loose cuts	trigger monitoring
e25	trigger monitoring
μ20i loose cuts	trigger monitoring
μ20	trigger monitoring
γ60i loose cuts	trigger monitoring
γ60	trigger monitoring
τ60 loose cuts	trigger monitoring
e^+e^- + Z mass, loose cuts	monitor flavour subtraction
$\mu^+\mu^-$ + Z mass, loose cuts	monitor flavour subtraction
$\mu^+\mu^-$ + Y mass	calibration

Large inclusive samples of muons are the starting point for the Inner Detector alignment; these will also be used to study energy loss in the calorimeters and to align the muon detectors. Large inclusive samples of electrons will be used to understand the electromagnetic energy scale of the calorimeter, using the precise momentum measurements from the Inner Detector, and to understand the energy–momentum matching between the Inner Detector and the calorimeter. The calorimeter inter-calibration, especially for the hadronic part, will benefit from inclusive (di-)jet samples by making use of the energy-balance technique.

Further examples of specific monitor and calibration triggers are given in Table 4-4. The random trigger on unpaired bunch crossing will select bunch crossings, where only one of the two packets is actually filled, and can thus be used to monitor beam-related backgrounds. The selection of oppositely charged lepton pairs, with an invariant mass compatible with the Z-boson mass, and loose cuts (especially on the isolation, which should be in this case defined identical for electrons and muons), will be an important control sample for the analyses where the contribution from different (lepton) flavours needs to be compared.

Monitor and calibration triggers provide examples of selections which do not always require the full detector information to be read out to mass storage. For some monitoring aspects, it could be envisaged to record only the results of the processing of the event in the EF, and not to store the raw data. However, it is anticipated that this would only be done after stable operation of the detector and the machine has been reached.

4.4.5 Physics coverage

In this section, examples will be given to indicate the sensitivity of the physics coverage to the thresholds used in the selection signatures. Indications on the redundancy achieved by the full set of selection signatures proposed are also given.

As an example, the search for the Standard Model Higgs boson in the decay mode to $H \rightarrow b\bar{b}$ will be discussed, where H is produced in association with a $t\bar{t}$ pair. The proposed selection criteria for this mode involve the requirement of a lepton from the semi-leptonic decay of one of the top quarks. In Ref. [4-1], the study was based on the assumption of a lepton-p_T threshold of 20 GeV for both electrons and muons. Table 4-5 shows the impact of raising one or both of these thresholds on the expected significance for signal observation.

Table 4-5 Example of loss in significance for the associated production of $t\bar{t}H$ (for a Standard Model Higgs boson with a mass of 120 GeV) for an integrated luminosity of 30 fb^{-1}

$p_T(e) >$	20 GeV	25 GeV	30 GeV	30 GeV	35 GeV
$p_T(\mu) >$	20 GeV	20 GeV	20 GeV	40 GeV	25 GeV
$S/\sqrt{B}$	**1**	**0.98**	**0.96**	**0.92**	**0.92**

Another example is the inclusive single-photon trigger, with a present threshold of 60 GeV. This selection can be used to search for technicolour via the production of a techni-omega, ω_T, which would be detected via its decay $\omega_T \rightarrow \gamma\pi^0{}_T \rightarrow \gamma b\bar{b}$. As shown in Ref. [4-1], for a mass $M(\omega_T) = 500$ GeV, the offline analysis would make a cut on the photon transverse energy of $E_T(\gamma) > 50$ GeV. This shows that a further raising of the single-photon threshold would impact the discovery potential. In addition, the overlap with the Tevatron reach up to techni-omega masses of about 400 GeV might not be assured.

Di-jet events will be used to search for new resonances decaying into two jets. The expected reach [4-5] of the CDF Experiment at Tevatron for an integrated luminosity of 15 fb^{-1} should cover E6 di-quarks with masses up to 700 GeV, heavy W' / Z' bosons with masses up to 850 GeV, techni-rho mesons, ρ_T, with masses up to 900 GeV, excited quarks q* with masses up to 950 GeV, and axigluons with masses up to 1250 GeV. The transverse-energy spectrum of the jets from the decay of resonance with a mass in the range discussed above would lead to a Jacobian peak at 350–630 GeV in E_T. An inclusive single-jet trigger with $E_T > 400$ GeV will not provide adequate coverage for this kinematic region and needs to be supplemented by a di-jet trigger with lower thresholds.

Further selections presently under study could enlarge the acceptance for some Higgs production and decay modes. The production of Higgs bosons via vector-boson fusion leads to the presence of non-central low-p_T jets that can be used as a tag. Requiring that two such jets be present in an event and that they are separated significantly in rapidity could allow, for example, the lepton p_T thresholds to be lowered, and thus increase the significance for Higgs obser-

vation. A second, related example is the search for a Higgs boson with an invisible decay mode, where a trigger requirement of two tag jets and missing transverse energy could be used.

The overall optimization of the online selection will be refined until the first collisions are recorded. Experience with the first real data might well force one to significantly revise this optimization and, in any case, it will have be refined continuously throughout the lifetime of the experiment. The following aspects have to be taken into account:

- the evolving understanding of physics;
- the effect of the foreseen thresholds on the acceptance for known (and unknown) physics processes;
- the rate of events after the selection (one should note that, even in the absence of fake signatures, there is a large rate of high-p_{T} events that one would like to retain — e.g. W and Z leptonic decays);
- the inclusiveness of the selection (one should note that a more exclusive selection does not necessarily imply a significantly reduced output rate — selections for many final states will have to be considered in order to achieve good coverage); and
- the available resources for the online selection, and the subsequent offline storage and processing of the data.

Various possibilities are available to control the output rate of the HLT. These include changes to the threshold values, changes to, or introduction of, pre-scale factors, phasing in of more exclusive selections and, where possible, tightening of selection cuts in the physics-object definition. Some of these schemas can also be applied at LVL1 in order to reduce the input rate to the HLT in case insufficient resources for the treatment the events are available.

It is important to keep in mind that less inclusive selections are targeted towards specific model predictions and are thus not desirable for unbiased searches for new physics. They will, however, be useful to increase the accumulated statistics for specific processes, e.g. when thresholds for the inclusive selection will have to be raised further. The introduction of additional biases at the trigger level due to less inclusive selection should be avoided where possible, since they might influence the accuracy of various precision measurements.

4.5 Adapting to changes in running conditions

An efficient mechanism is needed to adapt the trigger menu to changes in the running and operational conditions of the experiment. These changes will include a decrease in the luminosity during a coast, changes in machine-background conditions, and changes in detector performance affecting the selection criteria. Procedures must be put in place to react to these changes effectively and thus maintain a robust selection process. It is important to keep the correct history of all changes to the trigger configuration; this will be done by changing the run number whenever there is a significant change (see Section 3.3).

Luminosity changes

During a coast[1] in the LHC (estimated duration up to ~14 hours) the luminosity may drop by a factor of two or more. To take advantage of the available bandwidth and processing power in the HLT system, the trigger criteria should be adjusted to maintain an approximately constant event rate in the HLT. This could be achieved by including more trigger selections in the trigger menu, by reducing prescale factors for selections with lower-p_T thresholds, by adding more exclusive selections, or even by changing, i.e. lowering, the thresholds of the inclusive physics triggers. In this way, the physics coverage of ATLAS will be extended during a coast. An example is the case of b-hadron physics, where the inclusive trigger menu discussed above for $L = 2 \times 10^{33}\ cm^{-2}\ s^{-1}$ is restricted to select only final states with at least two oppositely-charged muons. As the luminosity drops below the initial peak value, other decay modes (e.g. fully hadronic or involving two low-p_T electrons) could be added.

The sizeable change in luminosity during a coast will imply changes in the average number of pile-up events present in the same bunch crossing, which might influence the optimal choice of threshold values (or isolation cuts) for various criteria in the definition of the selection objects. More studies are needed to assess whether simple changes of prescale factors are sufficient. It is important to have an accurate monitoring of the luminosity, which possibly requires additional dedicated trigger selections.

Background conditions

One will have to foresee adjustments for changes in the operating conditions, such as sudden increases in backgrounds or the appearance of hot channels, leading to certain triggers firing at a larger rate than is acceptable. Furthermore, machine-related backgrounds might increase suddenly and impact on the rate for certain selection signatures. Possible remedies in this case will be either to disable this selection or to significantly increase the prescale factor.

Mechanisms for adaptation

From an operational point of view, changes to pre-scale values must be simple and quick, whereas changes to threshold values might imply more complex re-initialization procedures. Changes in the prescale factors could be done dynamically in an automated fashion, however, it might be preferable to perform these changes less frequently in order to simplify the calculation of integrated luminosities for cross-section measurements.

Other remarks

The above list of changes in conditions is obviously incomplete and there will be many outside causes for changes to a stable operation, to which the online selection has to react and adapt in order to preserve the physics coverage of ATLAS.

4.6 Determination of trigger efficiencies

As far as possible, trigger efficiencies should be derived from the data. In this section, only a few basic ideas will be described, more details need to be worked out, e.g. addressing the statistical precision that can be obtained on the efficiency determination. No explicit distinction be-

1. A coast is the period when there is beam after a machine fill.

tween LVL1 and the HLT selections will be made, although the efficiency determination will be done separately for each trigger (sub-)level as well as globally. Furthermore, the efficiency of the selection must be monitored carefully throughout the lifetime of ATLAS, which implies that low-threshold triggers, from which the efficiencies are calculated, have to be kept running with pre-scale factors set to provide sufficient statistical precision. In the following, a qualitative overview of various possible procedures is given. The emphasis will be on trying to determine the efficiencies in several ways, in order to minimize systematic uncertainties.

Bootstrap procedure

One possibility is a 'bootstrap' procedure, starting off with a selection of minimum-bias events and using those to study the turn-on curve for very low-E_T triggers on jets, electrons, photons, muons, and so on. Next, the turn-on curves for electron triggers with higher E_T thresholds, for example, are studied using samples of events selected by an electron trigger with the lower (and thus already understood) threshold. The same holds for all other object types.

Selection of objects by independent signatures

For the HLT, it will be possible to foresee selection of objects by independent signatures in order to study the trigger efficiency of a particular step in the selection sequence, e.g. by selecting a sample with high-p_T inner-detector tracks in order to study the calorimeter or the muon trigger selections in more detail. Given the absence of a track trigger at LVL1, this will not be possible for the first stage of the selection.

Di-object samples

A further possibility is to use di-object samples which have been selected online by an inclusive single-object requirement and then reconstructed offline. These events can be used to study the trigger response for the second object, which was not required for the online selection. Examples of samples that can be used to study the lepton-trigger efficiences are $Z \rightarrow ll$ decays which are plentifully produced, and also $J/\psi \rightarrow ll$ or the $\Upsilon \rightarrow ll$ decays which might prove very useful for the region of lower transverse momenta. The same technique can be applied to events where the two objects are of different types. For example, events with $Z \rightarrow ll$ plus jets could be used to measure the efficiency of the jet trigger.

Required statistics

It cannot be emphasized enough that the amount of data needed to obtain a proper understanding of the online selection is not going to be negligible in the start-up phase and indeed throughout the lifetime of ATLAS. However, it will play an important role in assuring the potential of the experiment for discoveries and precision physics measurements. A detailed assessment of the expected needs will be done in the next couple of years. In the present menus only 10% of the total rate of the HLT is assigned to such triggers, which is smaller than the fractions used by current running experiments and might not be sufficient. During Tevatron Run I, as much as 30% of the accepted triggers at the second stage for CDF were pre-scaled physics (and calibration) triggers. In the case of D0, for the highest luminosities, about 20% of the triggers were pre-scaled and about 25% were monitoring and calibration triggers. Since D0 kept all thresholds fixed during a machine coast and only adjusted the pre-scale factors to always fill up the available bandwidth, the fraction of pre-scaled triggers increased towards the end of coast, where it could make up to 90% of the total rate.

4.7 References

4-1 *ATLAS Detector and Physics Performance TDR*, CERN / LHCC / 99-14 and 99-15 (1999)

4-2 T. Schoerner-Sadenius and S. Tapprogge (eds.), *ATLAS Trigger Menus for the LHC Start-up Phase*, ATL-COM-DAQ-2003-007 (2003)

4-3 *ATLAS Level-1 Trigger TDR*, CERN / LHCC / 98-14 (1998)

4-4 *ATLAS HLT, DAQ and DCS Technical Proposal*, CERN / LHCC / 2000-017 (2000)

4-5 *The CDF II Detector Technical Design Report*, FERMILAB-Pub-96 / 390-E (1996)

5 Architecture

This chapter addresses the architecture of the ATLAS HLT/DAQ system. It starts with a description of how the HLT/DAQ system is positioned with respect to both the other systems of the experiment and external systems such as the LHC machine and CERN technical services. This is followed by a description of the organization of the HLT/DAQ system in terms of the functions it provides and how they are organiz\ed in terms of sub-systems.

The system architecture, represented in terms of implementation-independent components, each associated to a specific function, and their relationships is then presented, together with the mapping of the HLT/DAQ sub-systems onto the architecture. This generic architecture has also been used to evaluate the overall system cost, discussed in Chapter 16. Finally, the concrete implementation (baseline) of the architecture is described.

5.1 TDAQ context

The context diagram of the HLT/DAQ is shown in Figure 5-1. It illustrates the inter-relations between the HLT/DAQ system seen as a whole and elements external to it that are directly related to data acquisition and triggering. It also illustrates the type of data which is exchanged in each case. The associated interfaces are introduced in Section 5.2.3, and discussed in more detail in Part 2 of the present document.

The LVL1 trigger provides LVL2 with RoI information needed to guide the LVL2 selection and processing; this interface is discussed in detail in Part 2. The TTC system [5-1] provides signals associated with events that are selected by the LVL1 trigger. RODs associated with the detectors provide event fragments for all events that are selected by the LVL1 trigger. In addition, the LVL1 system contains RODs that provide LVL1 trigger-decision data to be read out for the selected bunch crossings. The LVL1 trigger system, the TTC system and all the ROD systems need to be configured by the DAQ system, e.g. at the start of each run. These components are shown in Figure 5-1.

Interfaces to external systems are also illustrated in Figure 5-1. These connect to the LHC machine, in order to exchange information on beam parameters with the detectors, to control voltages for example; to the experimental infrastructure, to monitor temperatures of racks for example; and to the CERN technical infrastructure (such as the rack cooling water supply). The TDAQ interface for all these external systems is the DCS. Also shown are the interfaces relating to long-term storage of event data retained by the HLT prior to offline analysis, and non-event data which have to be stored such as alignment and calibration constants, configuration parameters, etc.

5.2 HLT/DAQ functional analysis

This section analyses the HLT/DAQ system in terms of the basic functions required, the building blocks and sub-systems that provide these functions, and their internal and external interfaces.

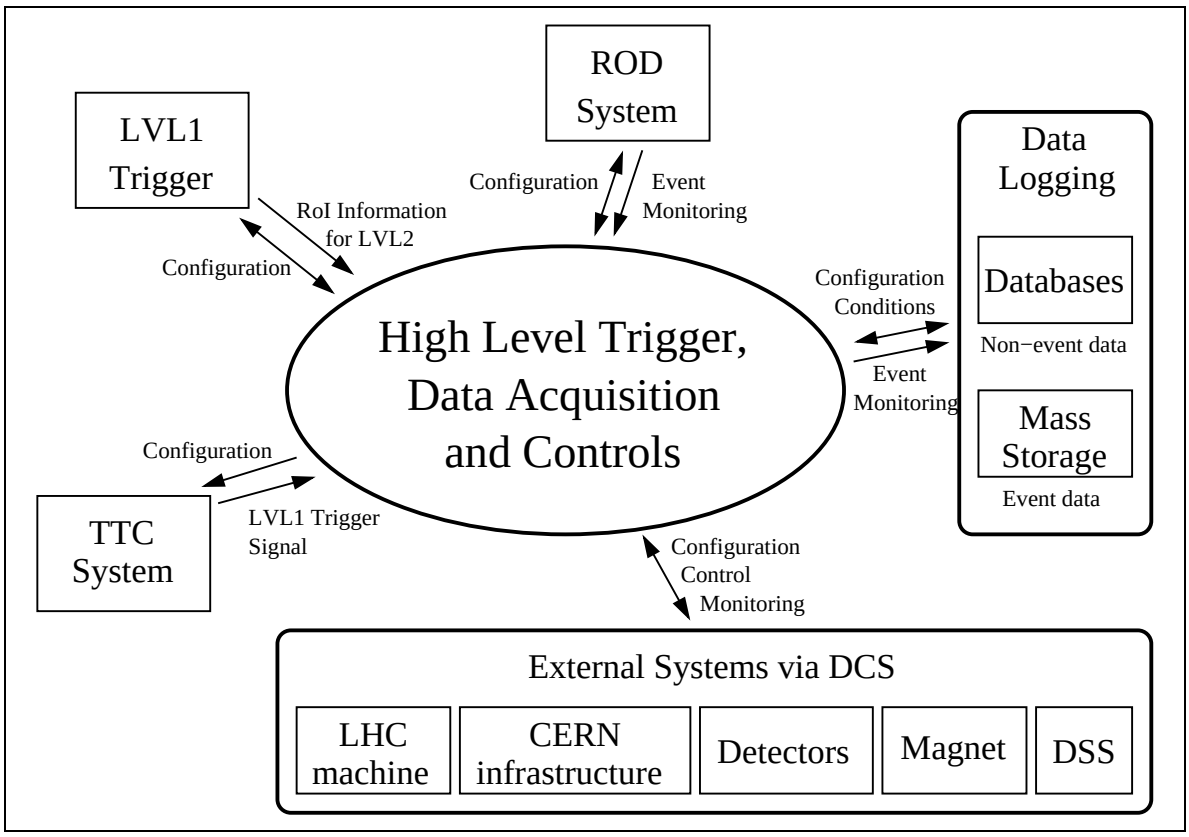

Figure 5-1 Context diagram

5.2.1 Functional decomposition

The HLT/DAQ system provides the ATLAS experiment with the capability of *moving* the detector data, e.g. physics events, from the detector to mass storage; *selecting* those events that are of interest for physics studies; and *controlling* and *monitoring* the whole experiment.

The following functions are identified:

- ***Detector readout***: for events, i.e. bunch crossings, selected by the LVL1 trigger, the data associated with the relevant bunch crossing are passed through detector-specific modules (RODs) that arrange them in formatted event fragments before sending them on to the HLT/DAQ system. An event arriving at the input to the HLT/DAQ system, i.e. at the ROBs, is therefore split into a number of fragments. Quantitatively, there are ~1600 RODs each of which sends one event fragment per event, i.e. at the LVL1 rate of up to 100 kHz.
- ***Movement of event data***: event fragments buffered in the ROBs have to be moved to the HLT and, for selected events, to mass storage. This is a complex process that involves moving both small amounts (typically ~2% of the full event) of data per event at the LVL1 trigger rate (data in RoIs for the LVL2 trigger) and the full event, i.e. ~1.5 Mbyte, at the rate of the LVL2 trigger (few kHz).
- ***Event selection and storage***: the HLT system is responsible for reducing the rate of events, and selecting those potentially of interest for offline analysis. Events selected by the HLT system are written to permanent storage for offline reconstruction and analysis. The data rate for selected events should not exceed a manageable level of a few hundred Mbyte/s.

- ***Controls***: TDAQ and detector online control includes the capability to configure, operate and control the experiment (detector, infrastructure, TDAQ) during data taking, in testing and calibration runs, and also to control certain vital service systems that remain active in periods when most of the detector and the LHC are shut down.
- ***Monitoring***: online monitoring includes the capability to monitor the state and behaviour (operational monitoring), and the performance (detector and physics monitoring) of all parts of ATLAS both during physics data taking and when calibration and testing operations are in progress.

5.2.2 HLT/DAQ building blocks

The HLT/DAQ system is designed to provide the above functions using the following building blocks:

- ***ROD crate DAQ***. This is the building block that provides the data acquisition functions for a ROD crate. During detector commissioning, debugging, or testing it allows data taking to be performed on a single ROD crate. It is itself built from the building blocks described in the following paragraphs and during normal experiment operations the ROD crate DAQ is operated as an integral part of the overall DAQ.
- ***Readout***. The Readout is a building block associated to functions of detector readout and the movement of event data. It provides for the receiving and buffering of event fragments coming from the RODs. The depth of the required buffers is determined by the duration of the LVL2 processing, plus the duration of event building (for events accepted by LVL2) and the time taken to remove[1] the events from the system. In addition, the Readout provides a sub-set of the buffered event fragments to the LVL2 trigger and all[2] event fragments to the event building.

  It also provides the first stage in the HLT/DAQ where event fragments from more than a single ROD can be coherently sampled (with respect to the EL1ID) for the provision of Monitoring functionality.
- ***LVL2 processing***. As outlined in Chapter 1, the LVL2 trigger uses the RoI mechanism [5-1] to selectively read out only the parts of each event that it needs to make the selection. Starting from RoI information supplied by the LVL1 trigger, appropriate event fragments are requested from the Readout and used to decide on the acceptance or rejection of the event. Event fragments are requested on the basis of the LVL1 event identifier and the η–ϕ region. The selection proceeds in several steps (with several matching data requests) and an event is rejected if at any given step none of the algorithmic criteria for all available signatures are fulfilled.
- ***RoI Collection***. This is the building block that, in conjunction with the Readout, provides the movement of a sub-set of event data to the LVL2 processing. It maps the η–ϕ region into Readout identifiers, collects the event fragments from the Readout, and provides the LVL2 processing with the resulting single data structure representing the RoI data.

1. An event is removed from the ROS either if it is rejected by the LVL2 or following the completion of the event building process for those events that were accepted by the LVL2.
2. It is also foreseen that the quantity of event fragments sent to the event building may depend on the type of event.

- ***Event Builder***. This building block provides, for those events passing the LVL2 processing selection criteria, for the movement of the event fragments out of the ROS. It builds, buffers, and formats a complete event as a single data structure. Following the building operation, the EB subsequently provides the event to the Event Filter.

  It is also the first stage in the HLT/DAQ where complete event fragments may be sampled for the provision of Monitoring functionality.

- ***Event Filter***. This is the final level of event rate and data volume reduction. With the LVL2 processing it provides the event selection functionality described in Section 5.2.1. It executes complex selection algorithms on complete events provided to it by the Event Building. It also provides an additional point in the HLT/DAQ where complete events may be used for the purposes of monitoring.

- ***Controls***. This is the building block that provides the configuration, control, and monitoring of the ATLAS experiment for and during the process of data taking. In conjunction with the Detector control and monitoring (see below) it provides the control functionality.

- ***Detector control and monitoring***. The detector is brought to and maintained in an operational state by this building block. It also performs the monitoring of the state of the detector using the information provided by detector sensors, e.g. flow meters and temperature monitors. In addition it receives and monitors information provided to ATLAS by external systems, e.g. the LHC machine, CERN technical infrastructure.

- ***Information services***. This building block provides for the exchange of information between the Controls building block and all other building blocks for the purposes of configuration, control, and monitoring. It provides information management, error handling and, makes available event fragments, sampled by the Readout and Event Builder, for the purpose of event-data-based operational monitoring of the detector.

- ***Databases***. Like the Information services this is a building block that supports the Controls and Detector control and monitoring building blocks to provide the functionality of configuration. It provides for the experiment's configuration description and its access by all building blocks. In addition, it allows the recording and accessing of information pertaining to the experiment's operation during data-taking.

5.2.3 HLT/DAQ interfaces

The HLT/DAQ system interfaces to a variety of other subsystems inside ATLAS, as well as external systems that are not under the experiment's control. The following sub-sections describe these interfaces with particular reference to the systems being connected, the interface responsibilities, and the data exchanged across the interface. References to more detailed documentation on the interfaces, including data formats are also given.

The interfaces can be split into two classes, those to other systems in ATLAS, and those to external systems. Table 5-1 summarizes the characteristics of the former.

Table 5-1 Overview of interfaces between HLT/DAQ and other ATLAS systems

Interface	Data Rate	Data Volume	Data Type
LVL1–LVL2	100 kHz	~1 kbyte / event	RoI information
Detector-specific non-physics trigger	few kHz	few words	Trigger signals
LVL1 & Detector Front-ends	100 kHz	~150 Gbyte / s	Raw data
Detector Monitoring interface TDAQ – detectors	few Hz	few Mbyte / s	Raw, processed, and control data
Conditions Database	Intermittent	~100 Mbyte / run	System status
Mass Storage Interface	~200 Hz	~300 Mbyte / s	Raw data + LVL2 and EF results

5.2.3.1 Interfaces to other parts of ATLAS

5.2.3.1.1 LVL1–LVL2 trigger interface

Although part of the TDAQ system, the LVL1 trigger is an external sub-system from the point of view of the DAQ and the HLT components.

A direct interface from the LVL1 trigger is provided through the RoIB that receives data from all of the LVL1 subsystems, the Calorimeter trigger, the Muon trigger and the CTP for events retained by LVL1. This information is combined on a per-event basis inside the RoIB, at the LVL1 rate, and passed to the supervisor of the LVL2 system, the L2SV. The interface has to run at full LVL1 speed, i.e. up to 100 kHz. The detailed specification of the LVL1–LVL2 interface is described in Ref. [5-2].

As mentioned above, there are RODs associated with the LVL1 trigger that receive detailed information on the LVL1 processing, e.g. intermediate results, for the events that are selected. The data from the corresponding ROBs are included in the event that is built following a LVL2-accept decision and is therefore accessible for the Event Filter processing. They could also be accessed earlier by the LVL2 trigger.

Trigger control signals generated by the LVL1 trigger are interfaced to the rest of the TDAQ system, as well as to the detector readout systems, by the TTC system documented in detail in Ref. [5-1].

5.2.3.1.2 Detector-specific triggers

For test beams, during integration, installation and commissioning, and also for calibration runs, it will be necessary to trigger the DAQ for a sub-set of the full system, i.e. a partition, independent of the LVL1 CTP and LVL2, in parallel with other ongoing activities. Detectors are provided with Local Trigger Processors (LTPs) [5-3] that offer the necessary common functions for such running independently of the LVL1 CTP. Triggers provided independently of the CTP are

referred to as detector-specific triggers. A DFM is needed for each partition to initiate the building of partial events in that partition.

Any detector-specific trigger will communicate via the TTC system with its corresponding DFM. The DFM component therefore requires a TTC input and a mode where it will work independently from LVL2. A back-pressure mechanism throttles the detector-specific trigger via the ROD-busy tree.

5.2.3.1.3 Interface to the detector front-ends

The detector front-end systems provide the raw data for each event that LVL1 accepts. The detector side of the interface is the ROD, while the TDAQ side is the ROB. The connection between the two is the ROL.

From the point of view of the detector, i.e. ROD, the interface follows the S-LINK specification ([5-4]). Implementation details of the ROL can change as long as this specification is followed. All RODs support a standard mezzanine card to hold the actual interface, either directly or on a rear-transition module. Data flow from the RODs to the ROBs, while only the link flow control is available in the reverse direction.

This interface has to work at the maximum possible LVL1 accept rate, i.e. up to 100 kHz and at 160 MByte / s.

5.2.3.1.4 TDAQ access to the Conditions Databases

The conditions databases are expected to come from an LHC Computing Grid applications area project, with any ATLAS-specific implementation supported by the Offline Software group. The Online Software system will provide interfaces to the conditions databases for all TDAQ systems and detector applications that require them online. It remains to be studied how accessing the conditions database will affect the HLT performance itself and how frequently such access will need to occur. This is an area that will be addressed in more detail in the Offline Computing TDR which is currently planned to be published in 2005.

The conditions database will store all time-varying information of the ATLAS detector that is required both for reconstructing and analysing the data offline and for analysing and monitoring the time variation of detector parameters and performance. Components of the HLT / DAQ system will write information into the database and read from it. In most cases read access will occur at configuration time, but the HLT may need to access the database more often.

5.2.3.1.5 Mass Storage

Events that have passed the EF will be written to mass storage. The design and support of the mass storage service will be centrally provided at CERN. However, in the first step of data storage, the Sub Farm Output (SFO) component will stream data directly to files on local disks. These will be large enough to accommodate typically a day of autonomous ATLAS data-taking in the event of failures in the network connecting ATLAS to the CERN mass storage system, or possible failures in the offline prompt reconstruction system. The event-data files are stored in a standard format (see Refs. [5-5] and [5-6]) and libraries are provided to read these files in offline applications. In normal running, data will be continuously sent from the local disk storage to the CERN mass storage system.

5.2.3.2 External interfaces

5.2.3.2.1 Interface to the LHC machine, safety systems, and the CERN technical infrastructure

All communications between the LHC machine, the general safety systems, the CERN technical infrastructure, the detector safety system, the magnets, and the TDAQ are done via DCS, with the exception of fast signals (such as the LHC 40 MHz clock and orbit signal) that are handled in the LVL1 trigger. These communication mechanisms and interfaces are described in Chapter 11.

5.3 Architecture of the HLT/DAQ system

This section presents the global architecture of the ATLAS HLT/DAQ system. The architecture builds upon the work presented in previous documents that we have submitted to the LHCC: the ATLAS Technical Proposal [5-7], the DAQ, EF, LVL2 and DCS Technical Progress Report [5-8], and the HLT/DAQ/DCS Technical Proposal (TP) [5-9]. The architecture is further developed by the requirements and performance studies, design, and development of prototypes and their exploitation in test beams, done since the TP. Table 5-2 summarizes the performance required from the different TDAQ system functions.

Table 5-2 Required system performance capabilities for a 100 kHz LVL1 accept rate

Function	Input requirements	Output requirements
Detector readout	~1600 event fragments of size typically 1 kbyte at 100 kHz	Few per cent of input event fragments to LVL2 at 100 kHz; ~1600 event fragments at ~3 kHz to EB
LVL2	Few per cent of event fragments at 100 kHz	100 kHz decision rate (~3 kHz accept rate)
EB	~1600 event fragments at ~3 kHz	~3 kHz and ~4.5 Gbyte/s
EF	~3 kHz and ~4.5 Gbyte/s	~200 Hz and ~300 Mbyte/s

The numbers presented in this table are based on a 100 kHz LVL1 accept rate (we recall that the ATLAS baseline assumes a LVL1 accept rate of 75 kHz, upgradeable to 100 kHz), and therefore represent an upper limit in the required design and performance capabilities of the system. Detailed simulations of the LVL1 trigger (see Section 13.5) show that its accept rate at LHC start-up luminosity (2×10^{33} cm^{-2} s^{-1}) will be ~25 kHz. Simulations of the HLT at this luminosity indicate a LVL2 rejection factor of ~30. This rejection has been linearly extrapolated to 100 kHz to give the numbers in the table. The rate at the output of the EF is, however, quoted as that expected from the above simulations. The output capacity of the EF could be increased, but will be limited finally by offline data processing and storage capabilities.

The architecture is presented following the functional breakdown given in the previous sections. The architectural components are described from the functional point of view, without reference to possible implementations at this stage. The baseline implementation of this architecture is presented in Section 5.5.

5.3.1 Architectural components

Figure 5-2 depicts the architectural components defined in the following text.

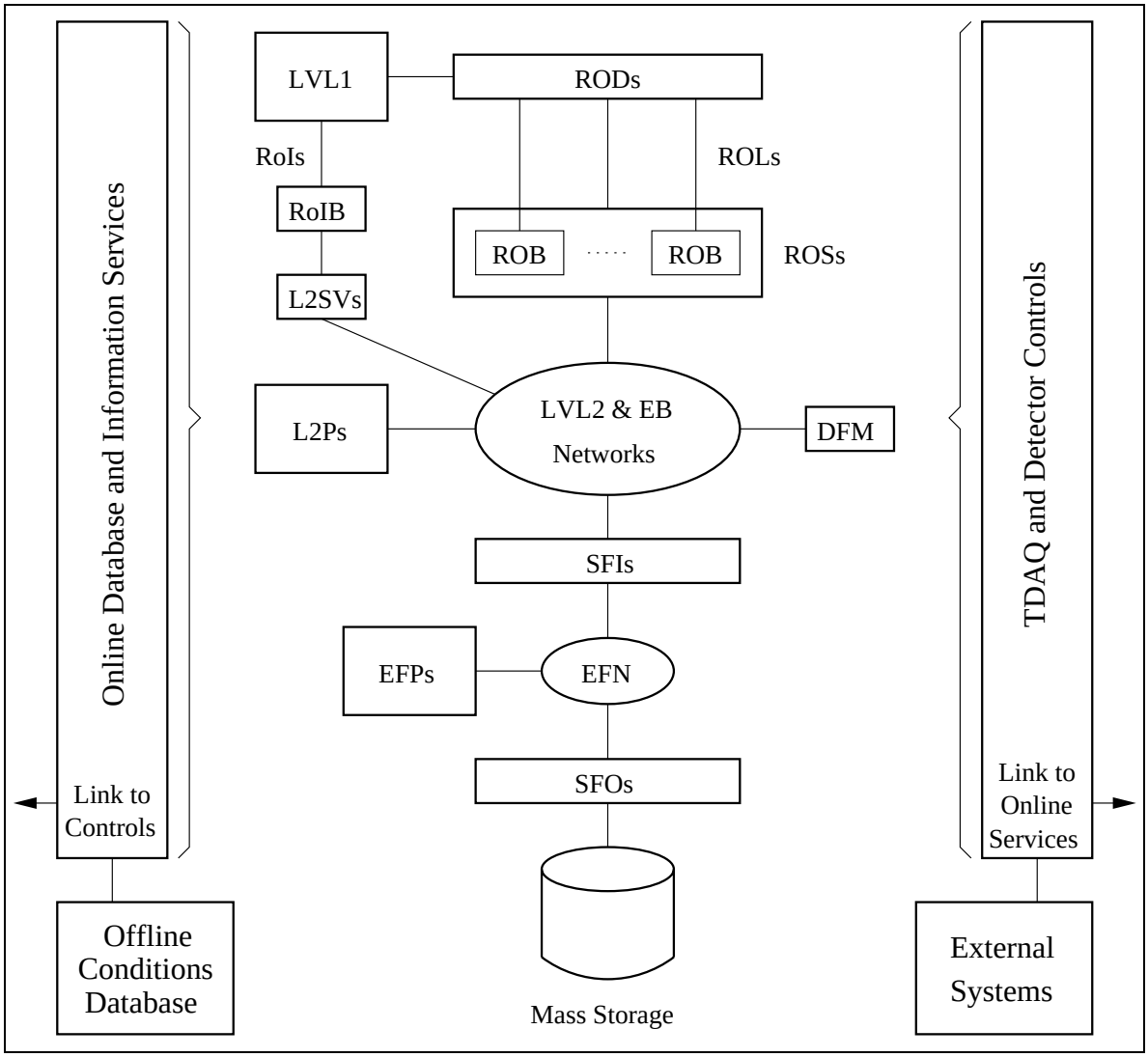

Figure 5-2 General architectural components and their relations

5.3.1.1 Detector readout

The communication link from the detector readout units (RODs), to the ROBs is the ROL. Each ROL carries one ROD fragment per event. The ROL must be able to transport data at a rate equal to the average event fragment size times the maximum LVL1 rate (possibly up to 160 Mbyte/s). In the calculation, the 'average event fragment size' should be taken as the mean over LVL1 triggers for the ROD that sees the largest event fragments on average.

ROBs are the memory buffers located at the receiving end of the ROLs, there being one ROL associated to one ROB. Several (typically two or four) ROBs are physically implemented on a ROBin card and several ROBins can be located in a ROS — the number depending on the implementation option (see Section 5.5.4). The ROS and its component ROBs provide two functions: buffering of event data and serving requests for event data from the HLT.

The ROBins are of a unique design used by all the ATLAS sub-detectors[1]. The detector event fragments are sent from the RODs and stored in the ROBs. The ROBs buffer the event fragment for the time needed by LVL2 to reject or select the event, plus that needed by the EB to build selected events.

The ROS is the component for serving data from the ROBs to LVL2 and the EB. In order to reduce the number of connections into the LVL2 and EB networks, it funnels a number of ROBs into a single port for each network. This ROB multiplexing capability is known as the ROB–ROS merging (RRM) function. The ROS also includes a software local controller of the online software (see Section 10.5), for the purpose of controlling data taking and local monitoring.

5.3.1.2 LVL2

The LVL2 trigger uses RoI information, provided by LVL1 via the RoIB, to request relevant event fragments from the ROS's (at the granularity of individual ROBs). Using these data, it produces a decision on the event and delivers the decision together with data it produced during its algorithmic processing back to the Data Flow components.

The RoIB receives information from LVL1 following each LVL1 trigger, allocates a LVL2 Supervisor (L2SV - see below) for the event, and transmits aggregate LVL1 information, after formatting, to the allocated supervisor. The RoIB will run at the rate of the LVL1 trigger. The L2SV then assigns, according to a load-balancing algorithm, a LVL2 Processor (L2P) from a pool under its control to process the event, and forwards the LVL1 information provided by the RoIB to an event handling process, the LVL2 processing unit (L2PU), running on the L2P[2]. The L2PU, using this LVL1 information, requests event fragments from the ROS, processes the RoI data, and makes a decision to accept or reject the event. In the course of processing the event data, additional requests for event fragments may be issued in several steps. The L2PU may, according to a pre-defined algorithm, decide to flag for accept, events which have in fact been rejected by the algorithm processing. This would be done in order to monitor the LVL2 selection process either in the EF or offline. The L2PU may also effect any required prescaling of selected channels. The final accept / reject decision is sent back to the L2SV. If an event is rejected, the decision is passed to the ROS via the DFM in order to remove the event from the ROBs. If an event is accepted, the decision is forwarded to the DFM, which then initiates the event-building operation for the event.

A switching network, the L2N (LVL2 Network) component, links the ROS's, the L2SV, and the L2Ps. The L2Ps will be organized into sub-farms in a manner which optimizes the use of the L2N.

Online software local controllers are associated to the RoIB, the L2SVs, and L2P sub-farms for the purpose of controlling and monitoring the LVL2 trigger.

1. Contrary to the ROD, which is a specialized detector-specific module, there is a single ROBin implementation for the whole experiment. RODs for the different detectors implement detector-dependent functions. For example, digital signal processing is implemented in the case of the LAr sub-detector.
2. The L2SV may also assign a given L2P on the basis of the LVL1 trigger type.

5.3.1.3 Event Builder

Events accepted by LVL2 are fully assembled and formatted in the EB's destination nodes, the SFIs. The DFM component is informed by the L2SV of the LVL2 decision. For each accepted event, the DFM, according to a load-balancing algorithm and other selective allocation considerations (e.g. special triggers and monitoring - see below), allocates an SFI. For rejected events and for events that have been successfully built, the DFM initiates a procedure to clear these events from the ROSs.

When the LVL2 system is not in use, for example during a detector calibration run, the DFM provides a LVL2-bypass mechanism. It is informed when new events are available for building, directly by the LVL1 CTP system or the Local Trigger Processor (see Section 5.2.3.1.2) of the detector being calibrated, via the TTC network. In the event of several TDAQ partitions being run in parallel, each running partition has a dedicated DFM for initiating event building within that partition.

The DFM has the additional capability to assign SFIs on the basis of pre-defined criteria, e.g. forced-accepted LVL2 events, special LVL1/LVL2 trigger types (as defined by the detector systems) etc.

The SFI allocated by the DFM requests and receives event data from the ROSs then builds and formats the event in its memory. It notifies the DFM when a complete event has been built, correctly or otherwise, e.g. when expected event fragments are missing. In the latter case, the SFI attempts corrective action, by trying to re-access the missing fragments. Built events are buffered in the SFIs in order to be served to the EF. For efficiency reasons, the SFI can build more than one event in parallel.

A switching network, the EBN (Event Builder Network) component[1], links ROSs, SFIs and the DFM. The network enables the building of events concurrently into all the SFIs at an overall rate of a few kilohertz.

Online software local controllers are associated to the DFM and SFIs for the purpose of controlling and monitoring the event building.

5.3.1.4 Event Filter

The EF comprises a large farm of processors, the EFP components. Each EFP deals with complete events served by the SFIs, as opposed to the selected event data used by the LVL2 trigger. From the architectural point of view the EF is a general computing tool for analysis of complete events — either events produced by the full ATLAS detector or the set of event data associated to a detector partition. In fact, it is also envisaged to use all or part of the EF for offline computing purposes, outside of ATLAS data-taking periods.

Each EFP runs an EF data flow control program (EFD) that receives built events from the SFIs. Several independent Processing Tasks (PTs) continuously process events allocated to them on demand by the EFD. Using offline-like event reconstruction and selection algorithms, the PT processes the event and produces a final trigger decision. When a given PT has completed the

1. The logically separate LVL2 and EB networks (which fulfil two different functions) could be implemented on a single physical network; in particular, this might be the case in the early phase of the experiment when the full performance is not required.

processing of an event, it requests a new one to be transferred from an SFI via the EFD. Data generated by the PTs during processing are appended to the complete raw event, if accepted, by the EFD. Accepted events are classified and moved to respective Sub-Farm Output buffers (SFOs), where they are written into local disk files. Completed files are accessed by a mass storage system for permanent storage (Section 5.2.3.1.5). Note that EFDs may send events to one of several parallel SFO output streams for further dedicated analysis, e.g. express analysis, calibration, debugging.

The EFPs are organized in terms of clusters or sub-farms, with each sub-farm being associated to one or more SFI and SFO. The minimum granularity of EFPs as seen from partitions is the sub-farm. SFIs, SFOs and EFPs are interconnected via a switching network, the EFN (EF Network) component.

The EF sub-farms may also be used for purposes other than event triggering and classification. A sub-farm may be allocated to running detector calibration and monitoring procedures which require events possessing specific characteristics. As discussed above, the DFM can assign events to dedicated SFIs from which EF sub-farms can request those events for dedicated analyses.

Monitoring and control of the EF itself is performed by a number of local controllers that interface to the online system.

5.3.1.5 Online Software system

The Online Software system encompasses the software to configure, control, and monitor the TDAQ system (known as the TDAQ control system), but excludes the management, processing, and transportation of physics data. Examples of the online software services are local control processes that interface to other TDAQ components, as noted above, general process management, run control, configuration data storage and access, monitoring, and histogramming facilities. A number of services are also provided to support the various types of information exchange between TDAQ software applications. Information can be shared between applications in the same sub-system, across different sub-systems, and between TDAQ systems and detectors.

TDAQ and detectors use the configuration databases to store the parameters and dependencies that describe their system topology and the parameters used for data-taking. The offline conditions databases are used to read and to store data that reflect the conditions under which the event data were taken.

The software infrastructure integrates the online services with the rest of TDAQ into a coherent software system. The hardware infrastructure comprises the Online Software Farm (OSF), the computers on which the services run, and an Online Software switching network (OSN) that interconnects the OSF with other TDAQ components. These include the DCS for control and information exchange as well as detector components that require access to online software facilities. The OSF will include machines dedicated to monitoring event data and database servers that hold the software and firmware for all of the TDAQ system — there will be servers local to clusters of computers that perform a common function, as well as central, backup servers.

Online software, services, and infrastructure are described in detail in Chapter 10.

5.3.1.6 Detector Control System

The DCS has a high degree of independence from the rest of the HLT/DAQ system, being required to be able to run at all times, even if HLT/DAQ is not available. At the level of detail suitable for this chapter, the DCS is a single component, without internal structure. It is interfaced to the rest of the system via the Online Software system. The interplay between DCS and TDAQ control is discussed in Section 5.3.2.

DCS is the only ATLAS interface with the LHC machine, with the exception of that for fast signals (which is handled directly by the LVL1 system). A detailed definition and description of the DCS architecture, interfaces, and implementation is given in Chapter 11.

5.3.2 Overall experimental control

The overall control of ATLAS includes the monitoring and control of the operational parameters of the detector and of the experiment infrastructure, as well as the supervision of all processes involved in the event readout and selection. This control function is provided by a complementary and coherent interaction between the TDAQ control system and the DCS. Whilst the TDAQ control is only required when taking data or during calibration and testing periods, the DCS has to operate continuously to ensure the safe operation of the detector, and a reliable coordination with the LHC control system and essential external services. DCS control services are therefore available for the sub-detectors at all times. The TDAQ control has the overall control during data-taking operations.

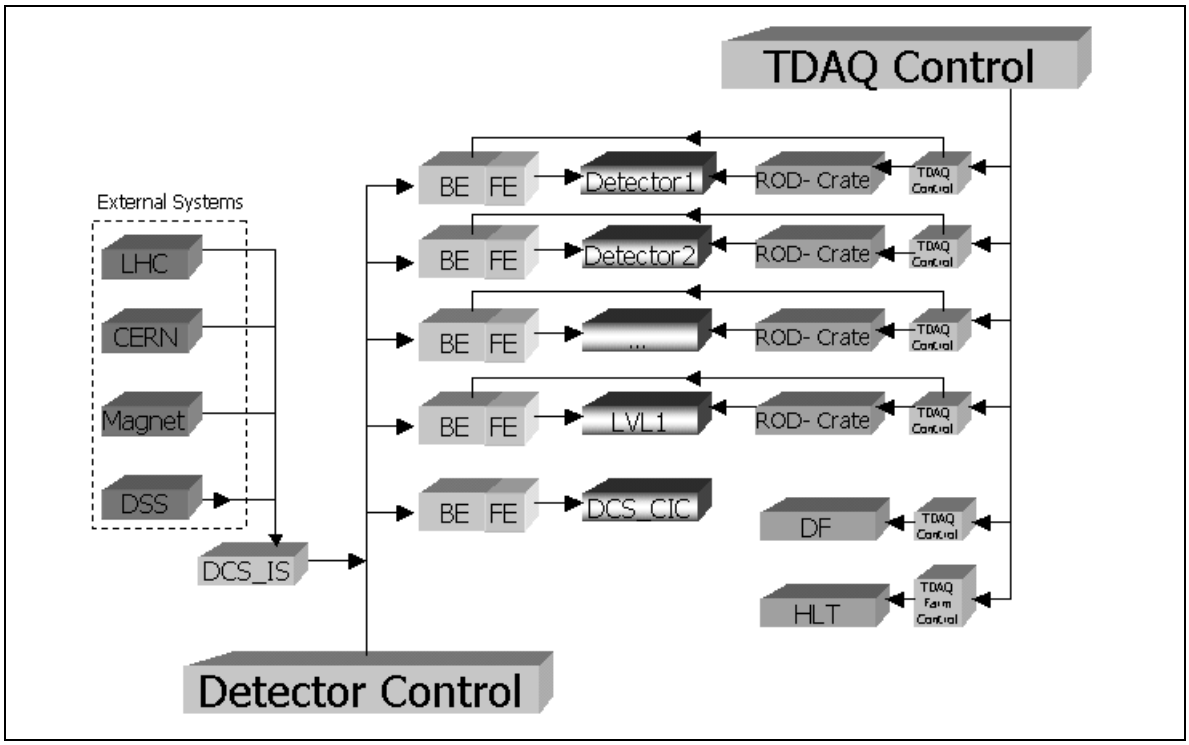

Figure 5-3 Logical experiment controls flow

There is a two-way exchange of information between DCS and the TDAQ control via mechanisms defined and provided by the Online Software (see Chapter 10). DCS will report informa-

tion about the status and readiness of various components that it controls and monitors to the Online system. The Online system will provide both configuration information and issue commands related to run control to DCS. Figure 5-3 presents the logical experiment control flow. In the figure, the lines represent the bi-directional information exchange between the systems, while the arrows on the lines indicate the direction of the command flow. The TDAQ control and DCS actions are coordinated at the sub-detector level. Each sub-detector, as well as the HLT and Data Flow systems, has TDAQ control elements responsible for the data-taking control. DCS control elements, called the Back-End (BE), control the sub-detector Front-End (FE) and LVL1 trigger equipment. DCS also controls the hardware infrastructure common to all the detectors, the DCS-Common Infrastructure Controls (DCS-CIC). The interaction with the LHC machine, which is explained in Chapter 11, and other external services is handled by DCS via the Information Service (DCS-IS). Experiment control is addressed in more detail in Chapter 12.

5.4 Partitioning

As already discussed in Chapter 3, partitioning refers to the capability of providing the functionality of the complete TDAQ to a sub-set of the ATLAS detector; for example, the ability to read out all or part of a specific detector with a dedicated local trigger in one partition, while the other detectors are taking data in one or more independent partitions.

The definition of the detector sub-set defines which ROBs belong to the partition (because of the connectivity between RODs and ROBs, and the fact that RODs are grouped into TTC partitions). For example, if a partition of the muon MDT chambers is required, the ROBs associated to the MDT RODs will be contained in the partition. Downstream of the ROBs a partition is realised by assigning part of TDAQ (EBN, SFI, EF, OSF and networking) to the partition — this is a resource-management issue. Some examples of TDAQ components that may have to be assigned to a TDAQ partition are a sub-set of the ROBs, as mentioned above, and a sub-set of the SFIs. The precise resource allocation in a particular case will depend on what the partition is required to do. For example, in order to calibrate an entire detector, all that detector's allocated ROBs, a fraction of the event building bandwidth and several EF sub-farms as well as online software control and monitoring functions may be required.

As regards the transport of the data to the allocated resources, the DFM allocates SFIs responsible for event building from a sub-set of ROSs. In order for this to happen, in the case of partitions associated to non-physics runs (i.e. when there is no LVL2) but that require functionality beyond the ROS, the DFM must receive, via the TTC, the triggering information for the relevant partition. Hence the need for full connectivity between the DFMs and the TTC partitions.

5.5 Implementation of the architecture

5.5.1 Overview

This section defines a concrete implementation for each of the architectural components described in the previous sections. The choices for what we call the baseline implementation have been guided by the following criteria:

- The existence of working prototypes.

- Performance measurements that either fulfil the final ATLAS specifications today or can be safely extrapolated to the required performance on the relevant timescale (e.g. extrapolating CPU speed of PCs according to Moore's law).
- The availability of a clear evolution and staging path from a small initial system for use in test beams, to the full ATLAS system for high-luminosity running.
- The overall cost-effectiveness of the implementation and the existence of a cost-effective staging scenario.
- The possibility to take advantage of future technological changes over the lifetime of the experiment.

The proposed baseline implementation is a system that could be built with today's technology and achieves the desired performance. It is expected that technological advances in the areas of networking and computing will continue at the current pace over the next few years; these will simplify various aspects of the proposed architecture and its implementation. Optimization in the area of the ROB I/O (see Section 5.5.4) remains to be performed prior to freezing details of the implementation.

By making use of commercial off-the-shelf (COTS) components based on widely-supported industrial standards wherever possible, the architecture will be able to take advantage of any future improvements from industry in a straightforward way. Only four custom components are foreseen in the final HLT/DAQ/DCS system as such[1]: the DCS ELMB module; the RoIB, of which only a single instance is needed; the ROL implementation; and the ROBin, which implements the ROL destination and the ROB functionality. Components of the TTC system are also custom elements but are common across the entire experiment (and indeed used in other LHC experiments).

The component performance and overall system performance figures which help to justify the proposed implementation can be found in Part 3 of this document.

Figure 5-4 depicts the baseline implementation of the system. The Online Software system is implicitly understood to be connected to all elements in the figure, and the DCS to all hardware elements that need to be monitored and controlled. Table 5-3 lists the principal assumptions from which the size of the implementation has been determined. Table 5-4 presents more details of the estimated size of the system; it lists the components that make up the baseline implementation and, for each one, gives the number of units and the associated technology.

5.5.2 Categories of components

The implementation calls for the use of a small number of categories of components, which are either custom or commercial, as follows:

- ***Buffers***. These are used to decouple the different parts of the system: detector readout, LVL2, EB and EF. Because of the parallelism designed into the system, buffers fulfilling the same function, e.g. ROBs, operate concurrently and independently.

1. The ROL source cards and the ELMB (see Chapter 11) are custom components that are specified and produced under the responsibility of the TDAQ project. However, these are integrated in the detector systems.

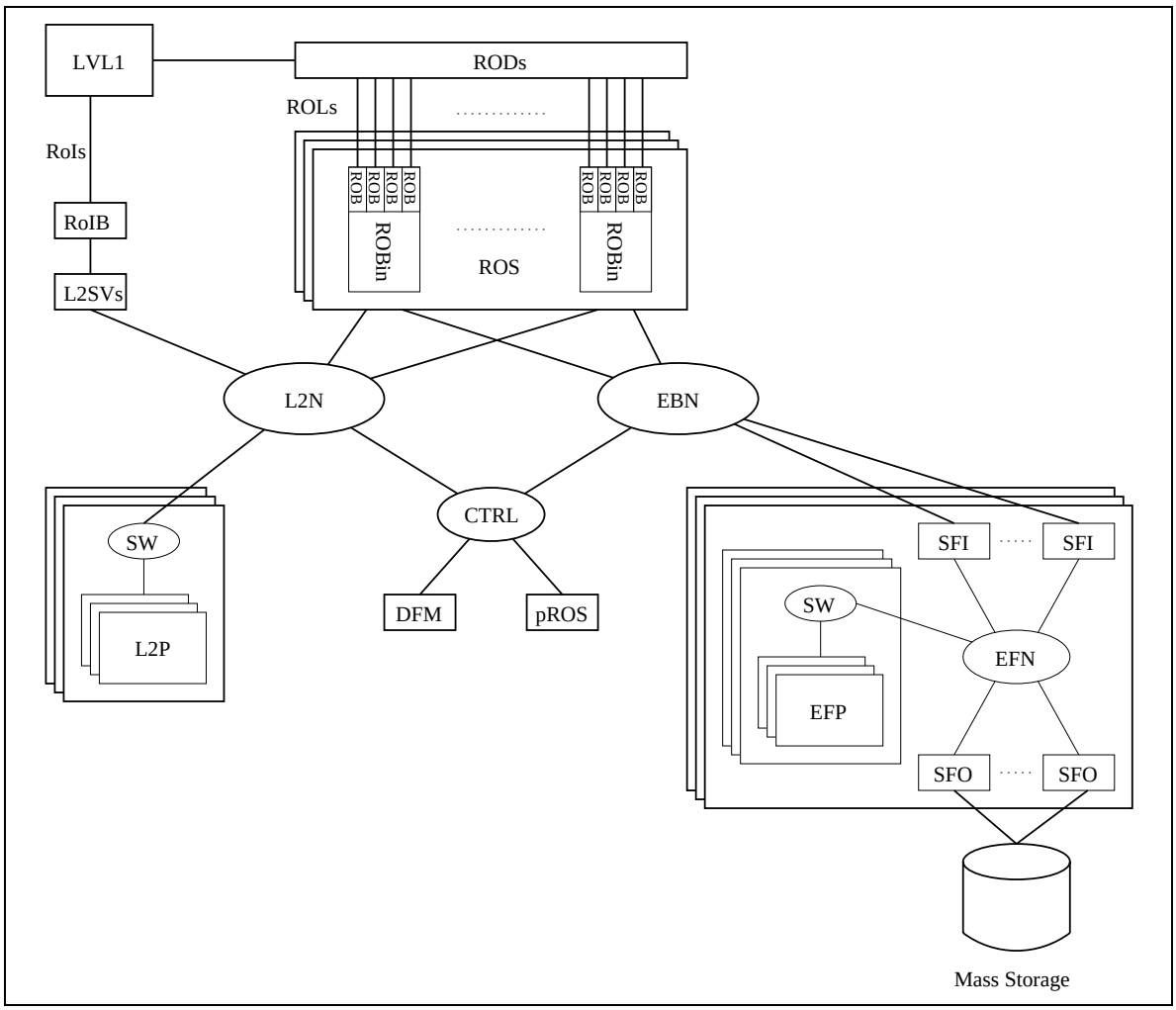

Figure 5-4 Baseline implementation of the Data Flow and High-Level Trigger

Table 5-3 Assumptions

Parameter	Assumed value	Comments
LVL1 rate	100 kHz	HLT/DAQ designed at this LVL1 rate
Event size (maximum)	2 Mbyte	Assumes maximum fragment size for all sub-detectors during normal high-luminosity running
Event size (average)	1.5 Mbyte	Assumed average value
LVL2 rejection	~30	EB rate of 3.3 kHz
RoI data volume	2% of total event size	Detector data within an RoI - average value
L2P rate (events/s)	~100	LVL2 decisions/s/processor Assume dual-CPU (8 GHz) machines
EFP rate (events/s)	~1	EF decisions/s/processor Assumed dual-CPU (8 GHz) machines

- ***Processors***. These run event selection algorithms, monitor and control the system. They are typically organized in farms, i.e. groups of processors performing the same function.

Table 5-4 Baseline implementation and size at a 100 kHz Level-1 rate

Component	Number of elements	Technology	Comments
ROL	~1600 links	Custom	Follows S-Link specification.
ROBin	~400 ROBins	Custom module	One ROBin multiplexes and buffers data from 4 ROLs.
ROS[a]	~60 switches ~60 PCs	Gigabit Ethernet Industrial PC	10(ROBins)x4(uplinks) concentrating factor. PC houses ~10 ROLs. ROBins are addressed individually via the concentrator switch.
	~150 PCs	Industrial PC	Multiplexes 3 ROBins. Relays RoI requests to appropriate ROBin. Builds super-fragment for the EB.
LVL2 Network	~650 ports[b]	Gigabit Ethernet	Connects ROS, L2P, L2SV, DFM.
EB Network	~250 ports	Gigabit Ethernet	Connects ROS, SFI, DFM.
RoIB	1 unit	Custom Module	
L2SV	~10	Dual-CPU PC[c]	One L2SV every 10 kHz of Level-1 trigger rate.
DFM	~35	Dual-CPU PC[c]	One per TTC partition.
L2P	~500	Dual-CPU PC[c]	Run LVL2 selection algorithms.
SFI	~90	Dual-CPU PC[c,d]	Build (and buffer) complete events.
EFP	~1600	Dual-CPU PC[c]	Run EF selection algorithms.
EF Network	~1700 ports	Gigabit Ethernet	Connects SFI, EFP and SFO.
SFO	~30	Dual-CPU PC[c] and ~1 Tbyte disk storage	Buffers events accepted by EF and stores them on permanent storage.
File Servers	~100	Dual-CPU PC[c] with ~1 Tbyte of disk space	Holds copy of databases and software. Local to a group of functionally homogeneous elements (e.g. a group of EFP).
Database servers	~2	RAID based file server	Hold master copy of databases, initialization data and down-loadable software.
Online farm	~50	Standard PCs	Operates and monitors the experiment. Runs online services.
Local control switch	~100	Gigabit Ethernet	Connects a group of ~30 elements (e.g. a LVL2 subfarm) to the central control switch.
Central control switch	250 ports	Gigabit Ethernet	Connects online farm, local control switches, and other system elements.

a. The ROS row indicates the number of components for both the switch-based (top numbers in the row) and bus-based ROS (bottom numbers).

b. The size of connectivity for the LVL2 network. The actual size of the central switch will be greatly reduced by the use of small concentrator switches (e.g. 5 L2Ps into a single central switch port).

c. Assume 8 GHz CPU clock rate.

d. At 8 GHz clock speed, the SFI may require only a single-CPU PC.

- ***Supervisors***. These are generally processors dedicated to coordinating concurrent activities, in terms of assigning events to processors and buffers at the different levels: the LVL2 trigger (supervised by the RoIB and L2SV units), the EB (supervised by the DFM) and the EF (supervised internally by its data flow control).
- ***Communication systems***. These connect buffers and processors to provide a path for transporting event data or a path to control and operate the overall system. Communication systems are present at different locations in the system. Some of them provide a multiplexing function, i.e. they concentrate a number of input links into a smaller number of output links. Depending on how the architecture is physically realised, a multiplexer may have a physical implementation (e.g. a switch, or a bus) or not (viz. a one-to-one connection, without multiplexing).

5.5.3 Custom components

5.5.3.1 The ELMB

The Embedded Local Monitor Board designed for use by the DCS system (see Chapter 11).

5.5.3.2 The ReadOut Link

The ROL will be implemented based on the S-LINK protocol [5-4]. It is discussed in detail in Section 8.1.2. The RODs will host the link source cards, either as mezzanines mounted directly on the RODs or on rear-transition modules. The ROBins, located near the RODs in USA15, will host several link destination cards. About 1600 links will be needed.

5.5.3.3 The ROBin

The ROBin is implemented as a PCI board receiving data from a number of ROLs. Each ROBin has a PCI interface and a Gigabit Ethernet output interface [5-10]. The high-speed input data paths from the RODs are handled by an FPGA. The buffers (associated to each input ROL) will be big enough to deal with the system latency (LVL2 decision time, time to receive the clear command after a LVL2 reject, and time to build the complete event following a LVL2 accept). A PowerPC CPU is available on each ROBin. The prototype ROBin is currently implemented on a single PCI board implementing two input links. The final ROBin design is expected to support four ROL channels.

5.5.3.4 The Region-of-Interest Builder

The RoIB design is modular and scalable and is custom designed and built. It is described in detail in Ref. [5-11]. The link interface from LVL1 to the RoIB follows the S-LINK specification. There are inputs from eight different sources to the RoIB (Section 5.2.3.1.1). Data flow from the LVL1 system to the RoIB, with the link interface providing only flow control in the reverse direction. Asserting XOFF is the only way for the RoIB to stop the trigger. The output of the RoIB is sent via S-LINK (or possibly via Gigabit Ethernet) to one of the L2SV processors.

5.5.4 ReadOut System

The ROS is implemented as a rack-mounted PC. Each ROS contains several ROBins, and has connections to the LVL2 and EB networks. Each ROBin multiplexes up to four ROLs into a single physical output channel. The ROS multiplexes the ROBin outputs onto the central LVL2 and EB networks (the RRM function, see Section 5.3.1.1). The RRM multiplexing factor depends on the physical number of links between the ROS and each of the central networks. The maximum factor can be calculated from external parameters, in particular the average RoI size, the peak RoI fragment request rate per ROB, and the LVL2 rejection power.

The RRM function may be implemented in two different ways:

- ***The bus-based ROS*** — the ROS PC contains three ROBins, each connected to its own PCI-bus and each having four ROL inputs to four ROBs, and one PCI output. This output is connected to the ROS's PCI-buses, which implement the RRM function. Requests, coming from LVL2 or the EB, for event fragments in the ROB, are handled directly by the ROS PC which aggregates the event fragments over its PCI-buses before dispatching them to the LVL2 or the EB. Gigabit Ethernet interfaces connect the ROS to the LVL2 and EB networks.
- ***The switch-based ROS*** — the ROS PC houses typically 10 ROBins, each having four ROL inputs to four ROBs, and one Gigabit Ethernet output. The RRM function is implemented typically using a 10 x 4 Gigabit Ethernet switch, which concentrates the ROBin outputs directly into four Gigabit Ethernet outputs: two for the LVL2 network and two for the EB network[1]. The switch-based ROS is understood to include the switch as well as the PC. The ROS PC itself does not play any role in the process of transferring data between the ROBs, and the LVL2 and EB. It is solely responsible via its PCI-bus for the physical housing of the ROBins, their initialization and control, and some monitoring functions.

Bus-based and switch-based ROSs are depicted in Figure 5-5. Preliminary studies have already been made on both implementation options, details can be found in Refs. [5-12] and [5-13]. The current ROBin prototype allows access to the ROB data via both the PCI-bus and the Gigabit Ethernet interfaces. The optimization of the ROS architecture, as indicated previously, will be the subject of post-TDR studies using this prototype. The final decision on an optimized design for the ROS architecture will be taken based on the results of these studies on a timescale compatible with the production of the ROBins (see Chapter 18).

As noted above, the ROBins, and therefore the ROSs will be located underground in the USA15 cavern.

5.5.5 Networks

All networks use Gigabit Ethernet technology despite the fact that in some cases, for example for the EFN, Fast Ethernet (100 Mbyte/s) would be sufficient to satisfy the requirements today. Considerations of overall network uniformity, technology evolution, and expected cost evolution justify this choice.

1. Whether one, two, or more outputs are used each to the EB & LVL2 networks will depend on how many switches are used to implement these networks.

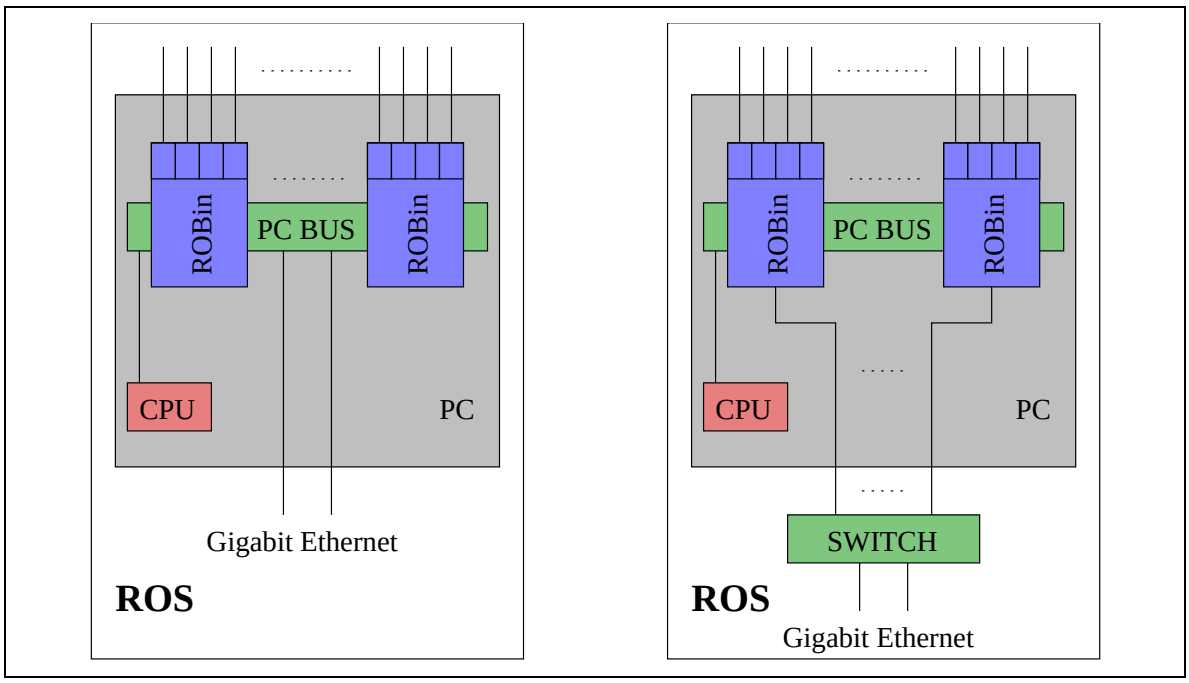

Figure 5-5 Bus and switched based ROSs

Following the first level of ROB concentration in the ROS (see Section 5.5.4), the topology of the EB and LVL2 networks consists of central switches, connecting sources (i.e. ROSs) and destinations (e.g. SFIs and L2Ps). These central networks will, logically, be monolithic in the sense that each network will connect to all of its sources and destinations. However, they may be physically organized either in terms of large monolithic switches or in terms of combined small switches. The detailed network topology will be fixed at the time of implementation.

Given the number of L2Ps (typically ~500 dual CPUs), small concentrating switches are used around the L2Ps to reduce the number of ports on the central switches. This is possible since the bandwidth per processor is much less than Gigabit Ethernet link capacity.

The organization of the OSN reflects the organization of the online farm. It is hierarchical, with networks local to specific functional elements (e.g. an EF sub-farm) and up-links to a central network providing the full connectivity. Typically, the OSN will be organized in terms of small (~40 ports) switches, local to groups of components with common functions (e.g. L2Ps), which connect through a large (~200 ports) central switch.

The current baseline for the overall EF network is a set of small independent networks (EFNs) each connecting a set of EF nodes with a number of SFIs and SFOs. This scheme allows flexibility in choosing the number of EF nodes for each cluster. The input rate capability can be increased by simply adding more SFI nodes to a given sub-farm. The EF network is organized in two layers. The lower layer consists of EF nodes clustered via small switches (labelled ‘SW’ in Figure 5-4, ~20 ports). The higher layer connects together a number of these EF clusters to one or more SFIs and SFOs via an EFN switch.

One possible network architecture implementation and topology, using the switch-based ROS (see Section 5.5.4) as an example, is given in Ref. [5-13].

5.5.6 Supervisory components

The LVSV and DFM supervisory components are implemented by high-performance, rack-mounted, dual-CPU PCs running Linux, connected to the EB and LVL2 networks. A custom PCI-to-S-LINK interface or standard Gigabit Ethernet technology may be used to connect the L2SV to the RoIB, and standard Gigabit Ethernet technology to connect the L2SV to the rest of the system. The connection between the DFM and the TTC network has not been defined yet — it could be based, for example, on a network technology or on a dedicated PCI-to-TTC-receiver interface.

Measurements done so far show that at the nominal LVL1 rate (100 kHz), only a small number of L2SVs (~10) are required. One DFM is required for each active concurrent TDAQ partition (see Section 5.4) — this includes the main data-taking partition. All DFMs except for the one running the main data-taking partition will need to receive detector-specific triggers directly or indirectly from the relevant TTC partition. In the main partition, the DFM receives its input from the L2SVs. The maximum foreseen number of partitions in ATLAS is 35. In practise, the number of active concurrent TDAQ partitions that require a DFM and in use at any time is likely to be less than this. A back-pressure mechanism will throttle the detector-specific trigger rate via the ROD-busy tree if it becomes unmanageable.

5.5.7 HLT processors

LVL2 and EF processors will be normal rack-mounted PCs running Linux. Dual-CPU systems are the most appropriate solution at the moment, although that may change in the future. We estimate that there will be ~500 LVL2 processors and ~1600 EF processors in the final (100 kHz) system (assuming 8 GHz clock speed), but initial setups during commissioning, and for the initial data-taking when the luminosity will be below the LHC design luminosity, will be much smaller. LVL2 and EF processors are COTS components and can be replaced or added at any time. Computing performance is more important than I/O capacity in the EF nodes.

Although 1U servers currently offer an attractive implementation for the farms, blade servers, which house hundreds of processors in a rack together with local switches, have a number of potential advantages, e.g. lower power requirements, higher density, and may offer a more attractive solution of sufficient maturity by the time the farms are purchased.

Studies are currently under way (see Ref. [5-14]) to investigate the possibility and implications of siting some EF sub-farms at locations outside CERN (e.g. at ATLAS institutes or computing centres). This would have the advantage of being able to benefit from existing computing equipment, both during the initial phases of the experiment, when only a fraction of the EF equipment at CERN will have been purchased, and throughout the lifetime of the experiment for non-critical applications such as data monitoring.

5.5.8 Event-building processors

The SFIs are again rack-mounted PCs running Linux. The event-building process requires a large CPU capacity to handle the I/O load and the event assembly. Again dual-CPU systems are the most cost-effective solution at the moment. The SFI components require no special hardware apart from a second Gigabit Ethernet interface that connects them to the EF network. Roughly 90 units are envisaged for the final system.

5.5.9 Mass Storage

The SFOs write events to the disks in a series of files, and provide buffering if the network connection to the CERN mass storage system is down. Assuming that the SFOs have to buffer ~1 day of event data, with an average event size of 1.5 Mbyte per event at a rate of ~200 Hz, they will need a total of ~26 Tbyte of disk storage. Today, relatively cheap PC servers can be bought with > 1 Tbyte of IDE disk storage. The SFOs will therefore consist of normal PCs, with a housing that allows the addition of large disk arrays — approximately 30 units are assumed for the final system.

5.5.10 Online farm

The online farm provides diverse functions related to control, monitoring, supervision, and information / data services. It will be organized hierarchically in terms of controllers and database servers local to the TDAQ functional blocks (for example the ROS or an EF sub-farm) and clusters of processors providing global functions (experiment control, general information services, central database servers).

The processors for experiment control and general information services will be standard PCs organized in small farms of ~20 PCs each. Local file servers are also standard PCs, with the addition of a large local disk storage (~1 Tbyte), while the central database servers are large file servers with several TByte of RAID disk storage.

At the time of system initialization, quasi-simultaneous access to the configuration and conditions databases will be required by many hundreds of processes, in some cases requiring *O*(100) Mbyte data each. This calls for a high-performance in-memory database system and a structured organization of the database servers to spread the I/O load sufficiently widely that the system can be initialized and configured in a reasonable time. It is proposed to organize the database infrastructure as follows:

- ***Local data servers***: database servers that hold a copy of the conditions and configurations databases to be accessed locally by an homogeneous sub-set of the system (e.g. an EF sub-farm). A copy of specific software and miscellaneous data that is to be downloaded into the components in the sub-set is also held on the local server.
- ***Central server***: a fault tolerant, redundant database server that holds the master copy of the TDAQ software and configuration as well as a copy of the conditions data. Local servers are updated from the central server at appropriate intervals. The central server may both update and be updated by the central offline conditions database.

The control of database write access and synchronization – in particular in the case of conditions data where both online and offline clients may wish to write to the database at the same time – is a complex problem that has not yet been addressed in detail. This issue will be studied in common with the offline community as well as with other experiments with the aim of arriving at a common solution.

5.5.11 Deployment

Functionally homogeneous components, e.g. EF processors, are organized in terms of standard ATLAS sub-farm racks. Each rack contains, in addition to the computing components, a local

file server; a local online switch (to connect all the components to the online central switch); a rack controller; and the necessary power and cooling distribution. We are investigating, with other LHC experiments, cooling systems for racks with horizontal air-flow suitable for rack-mounted PCs. The number of components per rack depends on the size of the racks. Table 5-5 summarizes the organization of the different racks, the number of racks of a specified content per function and their physical location.

Table 5-5 Rack organization and location

Unit type	Number of units	Composition	Location
ROS rack	~15	~15 ROSs Local online switch Local file server Rack controller	USA15 – underground
HLT rack	~20 for LVL2 ~80 for EF	~30 Processing Units Local online switch Local file server Rack controller	SDX 15 – surface 50% of EF racks possibly located in the CERN computer centre
SFI rack	3–4	~30 SFIs / rack Local online switch Local file server Rack controller	SDX 15 – surface
SFO rack	6–7	~5 SFO (+ Disk space) Local online switch Local file server Rack controller	SDX 15 – surface
Online rack	3	~20–30 Processors Local online switch Local file server Rack controller	SDX 15 – surface
Central switches	3	~256 ports per switch	SDX 15 – surface

5.6 Scalability and staging

The performance profile anticipated for the ATLAS TDAQ system between the detector installation and the availability of the LHC nominal luminosity, is summarized in Table 5-6. It is expressed as the required LVL1 rate in kilohertz.

The scalability of the TDAQ system is discussed on the basis of the above performance profile. Prior to the ATLAS start-up, the TDAQ system will provide the full detector read out for the purpose of commissioning the experiment and the cosmic run, and a minimal HLT system appropriate to fulfilling the requirements for this period (few kilohertz).

The detector readout system must be fully[1] available at the time of the detector commissioning (i.e. in 2005) irrespective of the rate performance, since all parts of the detector must be connect-

1. Some 25% of the detector ROLs, corresponding to the part of the ATLAS detector which is staged, will not be installed at the start-up of ATLAS but later: currently planned for 2008.

Table 5-6 TDAQ required performance profile

Phase	Date (TDAQ ready by)	Performance (LVL1 rate)
ATLAS Commissioning	2005	N / A[a]
Cosmic-ray run	2006	N / A[a.]
ATLAS start-up	2007	37.5 kHz
Full performance	2008	75 kHz
Final performance	2009	100 kHz

a. Provide full detector read out but minimal HLT capability.

ed. In contrast, the processing power in the HLT depends on the maximum LVL1 rate that must be accepted. Since the LVL2 and EF farms, as well as SFIs, SFOs, etc, are implemented using standard computer equipment connected by Gigabit Ethernet networks, they can be extended as financial resources become available.

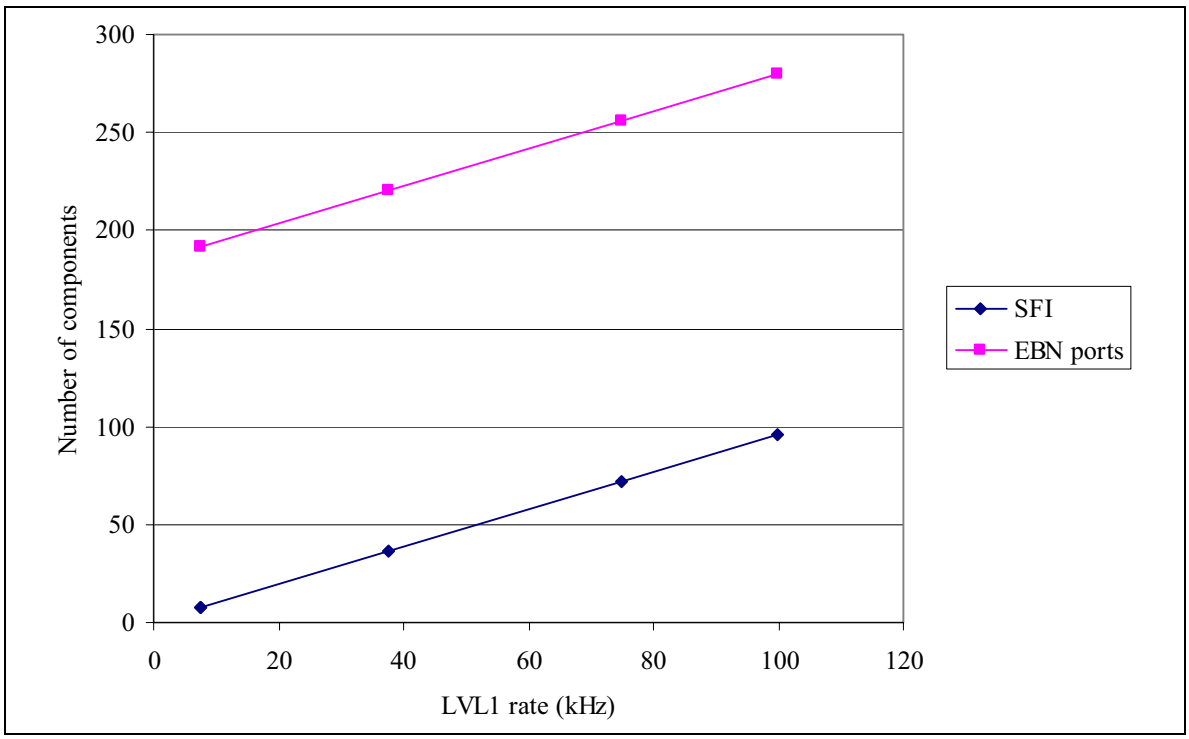

Figure 5-6 Scaling of the EB sub-system

The strategy is therefore the staging of the farms and associated networks. In the latter case, additional central switches (if a topology based on multiple central switches is chosen) or ports (for monolithic central switches) will be added to support the additional HLT nodes. The same argument, although on a smaller scale, applies to SFIs and SFOs.

The scaling of the TDAQ system size, as a function of the LVL1 trigger rate, is depicted in Figure 5-6 for the Event Builder (number of SFIs and number of EBN ports), Figure 5-7 for the Level-2 sub-system (number of L2Ps and number of L2N ports), and Figure 5-8 for the Event Filter (number of EFPs and ports of the Event Filter network).

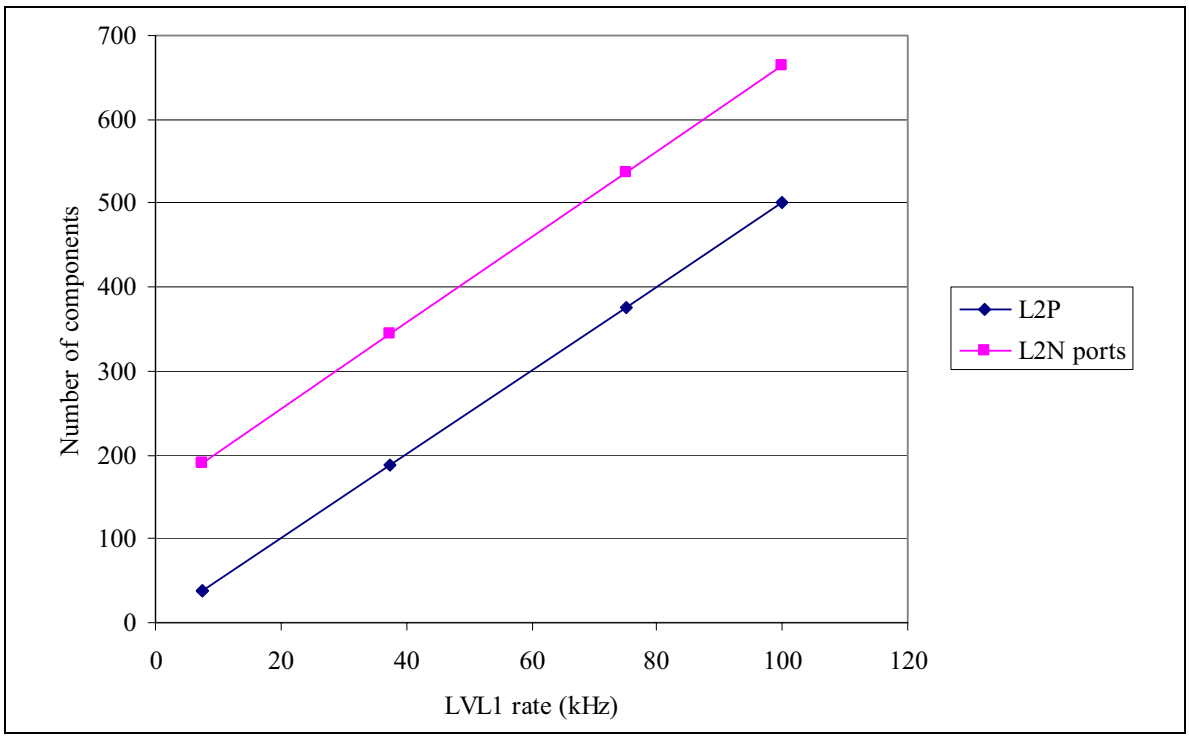

Figure 5-7 LVL2 system scaling

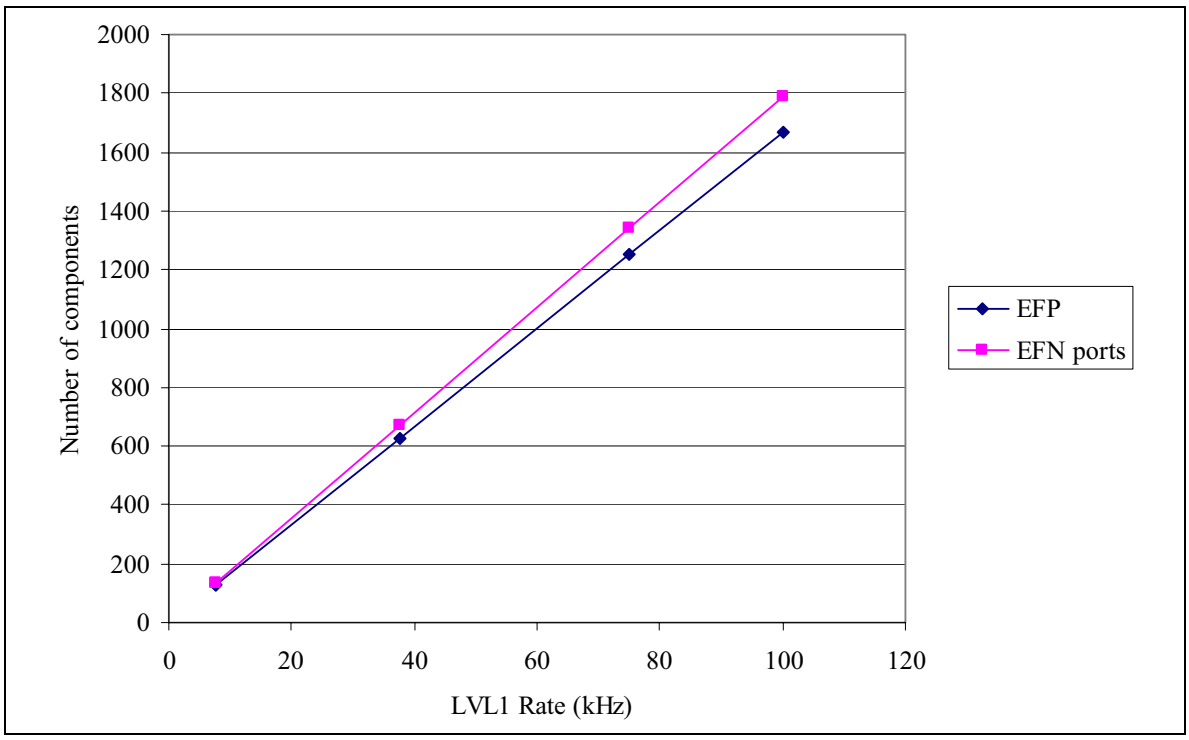

Figure 5-8 EF system scaling

5.7 References

5-1 *ATLAS Level-1 Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

5-2 *Specification of the LVL1 / LVL2 trigger interface,* ATLAS EDMS Note, ATL-D-ES-0003 ()
http://edms.cern.ch/document/107485/

5-3 *Proposal for a Local Trigger Processor,* ATLAS EDMS Note, ATL-DA-ES-0033,
http://edms.cern.ch/document/374560/

5-4 *Recommendations of the Detector Interface Group - ROD Working Group,* ATLAS EDMS Note, ATL-D-ES-0003 ()
http://edms.cern.ch/document/332389/

5-5 *The raw event format in the ATLAS Trigger & DAQ,* ATLAS Internal Note, ATL-DAQ-98-129 (1998)

5-6 *Event Storage Requirements,* ATLAS EDMS Note, ATL-DQ-ES-0041 ()
https://edms.cern.ch/document/391572/

5-7 ATLAS Collaboration, *ATLAS: Technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN*, CERN/LHCC/94-43 (1994)

5-8 *ATLAS DAQ, EF, LVL2 and DCS Technical Progress Report*, CERN/LHCC/98-16 (1998)

5-9 ATLAS Collaboration, *ATLAS High-Level Triggers, DAQ and DCS Technical Proposal,* CERN/LHCC/2000-017 (2000)

5-10 *Summary of Prototype ROBins,* ATLAS EDMS Note, ATL-DQ-ER-0001 ()
http://edms.cern.ch/document/382933/

5-11 *A Prototype RoI Builder for the Second Level Trigger of ATLAS Implemented in FPGAs,* ATLAS Internal Note, ATL-DAQ-99-016 (1999)

5-12 B. Gorini et al., ROS Test Report, *in preparation*

5-13 *ATLAS TDAQ: A Network-based Architecture,* ATLAS EDMS Note, ATL-DQ-EN-0014 ()
https://edms.cern.ch/document/391592/

5-14 K. Korcyl et al., *Network performance measurements as part of feasibility studies on moving part of the ATLAS Event Filter to off-site Institutes,* 1st European Across Grids Conference, Universidad de Santiago de Compostela (2003)

6 Fault tolerance and error handling

6.1 Fault-tolerance and error-handling strategy

Error handling and fault tolerance are concerned with the behaviour of the TDAQ system in the case of failures of its components [6-1]. Failure, here, means the inability of a component, be it hardware or software, to perform its intended function.

The overall goal is to *maximize system up-time, data-taking efficiency and data quality* for the ATLAS detector. This is achieved by designing a *robust* system that will keep functioning even when parts of it are not working properly.

Complete fault tolerance is a desired system property which does not imply that each component must be able to tolerate every conceivable kind of error. The best way for the system to achieve its overall goal may well be to simply reset or reboot a component which is in an error state. The optimal strategy depends on the impact that the faulty component has on data taking, the frequency of the error, and the amount of effort necessary to make the component more fault tolerant.

The fault-tolerance and error-handling strategy is based on a number of basic principles:

- The number of single points of failure in the design itself must be minimized. Where unavoidable, provide redundancy to quickly replace failing components. This might consist of spare parts of custom hardware or simply making sure that critical software processes can run on off-the-shelf hardware which can be easily replaced.
- Failing components must affect as little as possible the functioning of other components.
- Failures should be handled in a hierarchical way, where local measures are taken to correct failures. Local recovery mechanisms will not make important decisions, e.g. to stop the run, but pass the information on to higher levels.
- Errors are reported in a standardized way to make it easy to automate detection and handling of well-defined error situations (e.g. with an expert system).
- Errors are automatically logged and the record made available for later analysis if necessary. Where the error affects data quality the necessary information will be stored in the conditions database.

The following actions can be distinguished.

Error detection describes how a component finds out about failures either in itself or neighbouring components. Errors are classified in a standardized way and may be transient or permanent. A component should be able to recover from transient errors by itself once the cause of the error disappears.

Error response describes the immediate action taken by the component when it detects an error. This action will typically allow the component to keep working, maybe with reduced functionality. Applications which can sensibly correct errors that are generated internally or occur in hardware or software components they are responsible for should correct them directly. In many cases the component itself will not be able to take the action necessary to cure failures in a

neighbouring component. Even if the component is unable to continue working, this should not be a fatal error for the TDAQ system if it is not a single point of failure.

Error reporting describes how failure conditions are reported to applications interested in the error condition. The mechanism will be a standardized service which all components use. The receiver of the error message might be persons (like a member of the shift crew or an expert) or an automated expert system.

Error recovery describes the process of bringing the faulty component back into a functioning state. This might involve manual intervention by the shift crew, an expert, or an automated response initiated by the expert system. The time-scale of this phase will typically be longer than the previous ones and can range from seconds to days (e.g. in the case of the replacement of a piece of hardware which requires access to controlled areas).

Error prevention describes the measures to be taken to prevent the errors from being introduced in hardware or software. Good software engineering, the use of standards, training, testing, and the availability and use of diagnostic tools help in reducing the error level in the TDAQ system.

6.2 Error definition and identification

In order to respond to error conditions it is important to have a clearly-defined TDAQ-wide classification scheme that allows proper identification of errors. It is assumed that error conditions are detected by DataFlow applications, controllers, event-selection software, and monitoring tasks. These conditions may be caused by failures of hardware they control, of components that they communicate with, or these may occur internally.

If the anomalous condition cannot be corrected immediately an error message is issued. Error messages are classified according to severity. The classification is necessarily based on local judgements; it is left to human / artificial intelligence to take further action, guided by the classification and additional information provided by the applications that detect the errors.

Additional information consists of a unique TDAQ-wide identifier (note that status and return codes, if used, are internal to the applications), determination of the source, and additional information needed to repair the problem. Messages are directed to an Error-Reporting Service, never directly to the application that may be at the origin of the fault.

For successful fault management it is essential that correct issuing of error messages be enforced in all TDAQ applications.

6.3 Error-reporting mechanism

Applications encountering a fault make use of an error-reporting facility to send an appropriate message to the TDAQ system. The facility is responsible for message transport, message filtering, and message distribution. Optional and mandatory attributes are passed with the messages. The facility allows receiving applications to request messages according to the error severity or other qualifiers, independent of origin of the messages. A set of commonly used qualifiers will be recommended. These can, for example, include the detector name, the failure type (e.g. hardware or software), or finer granularity indicators like ‘Gas’, ‘HV’ etc. Along with mandato-

ry qualifiers like process name and id, injection date and time, and processor identification, they provide a powerful and flexible system logic for the filtering and distribution of messages.

Automatic message suppression at the sender level is foreseen to help avoid avalanches of messages in the case of major system failures. A count on the suppressed messages is kept. Message suppression at the level of the message-reporting system is also possible.

An error-message database is foreseen to help with the standardization of messages including their qualifiers. A help facility can be attached which allows the operator to get detailed advice for the actions to be taken for a given failure.

6.4 Error diagnosis and system-status verification

Regular verification of the system status and its correct functioning will be a vital operation in helping to avoid the occurrence of errors. A customizable diagnostics and verification framework will allow verification of the correct status of the TDAQ system before starting a run or between runs, automatically or on request. It will make use of a suite of custom test programs, specific to each component type, in order to diagnose faults.

6.5 Error recovery

Error-recovery mechanisms describe the actions which are undertaken to correct any important errors that a component has encountered. The main goal is to keep the system in a running state and minimize the consequences for data taking.

There will be a wide range of error-recovery mechanisms, depending on the subsystem and the exact nature of the failure. The overall principle is that the recovery from a failure should be handled as close as possible to the component where the failure occurred. This allows failures to be dealt with in subsystems without necessarily involving any action from other systems. This decentralizes the knowledge required to react appropriately to a failure and it allows experts to modify the error handling in their specific subsystem without having to worry about the consequences for the full system.

If a failure cannot be handled at a given level, it will be passed on to a higher level in a standardized way. While the higher level will not have the detailed knowledge to correct the error, it will be able to take a different kind of action which is not appropriate at the lower level. For example, the higher level might be able to pause the run and draw the attention of the member of the shift crew to the problem, or it might take a subfarm out of the running system and proceed without it.

The actual reaction to a failure will depend strongly on the type of error. The same error condition, for example timeouts on requests, may lead to quite different actions depending on the type of component in which the error occurs. A list of possible reactions is given in Section 6.8.

Each level in the hierarchy will have different means to correct failures. Only the highest levels will be able to pause data taking or to stop a run.

The functionality for automatic recovery by an expert system will be integrated into the hierarchical structure of the TDAQ control framework and can optionally take automatic action for

the recovery of a fault. It provides multi-level decentralized error handling and allows actions on failures at a low level. A knowledge base containing the appropriate actions to be taken will be established at system installation time. Experience from integration tests and in test-beam operation will initially provide the basis for such a knowledge base. Each supervision node can contain rules customized to its specific role in the system.

6.6 Error logging and error browsing

Every reported failure will be logged in a local or central place for later retrieval and analysis. The time of occurrence and details on the origin of the failure will also be stored to help in determining the cause and to build failure statistics, which should lead to the implementation of corrective actions and improvements of the system. Corresponding browsing tools will be provided.

6.7 Data integrity

One of the major Use Cases for error handling and fault tolerance concerns the communication between sources and and destinations in the system.

Given the size of the TDAQ systems, there is a significant possibility that at any given moment one of the sources of data may not be responding. Each element in the DataFlow can be generally seen as both a source and a destination.

Because of electronics malfunctioning, e.g. a fault in the power supply for a number of front-end channels, it may happen that a source temporarily stops sending data. In this case the best strategy for the error handling is to have a destination that is able to understand, after a timeout and possible retries (asking for data), that the source is not working. In this case the data fragment will be missing and the destination will build an empty fragment with proper flagging of the error in the event header. The destination will issue an error message.

There are cases of communications errors where there is no need for a time-out mechanism. This is, for example, the case of a ReadOut Link fault due to Link Down. The S-LINK protocol signals this situation, the receiving ROS immediately spots it, builds an empty fragment, and reports the link fault. A similar mechanism can also be envisaged for the Front-End electronics to ROD communication that is also established via point-to-point links.

There are many situations where a time-out mechanism is mandatory, for example when the communication between source and destination is using a switched network. In this case possible network congestion may simulate a source fault (e.g. through packet loss). Switched networks carrying event data at high rates are present between the ROS and the Event Building, the ROS and the LVL2, and the Event Building and the Event Filter.

The choice of the correct time-out limits can only be made once the system is fully assembled with the final number of nodes connected to the switches. Only then can the normal operation timing be evaluated via dedicated measurements with real and simulated data. The system timeouts may have to be tuned differently when the system is used for physics and calibration runs. For example, in calibration mode the time for getting a data fragment may be higher due to the bigger amount of data to be transferred from a source to a destination.

6.8 Fault-tolerance and failure strategy for hardware components

6.8.1 The source end of ReadOut Links

The ROL is implemented as an S-LINK [6-2] LSC mezzanine installed on the ROD module, or installed on the ROD rear transition module, or integrated into the ROD rear transition module. On the receiver end of the ROL, multiple S-LINK input channels are integrated onto a ROBin card.

The ROL design uses small form-factor pluggable (SFP) [6-3] fibre optic transceiver modules. In the case of a failure of a fibre optic transceiver, a replacement part can therefore simply be plugged in from the front-panel, avoiding the need to power off the ROD crate or the ROS PC. The only implication is that fragments from the faulty ROL will be lost during the operation. Device qualification data from the fibre optic transceiver vendor indicates that fewer than 1.5% of the devices will fail over a 10 year period. Sufficient spare parts will be available for replacement.

In the case of a failure of an LSC mezzanine card or a ROD rear transition module, the crate will have to be powered off, in order to replace the faulty cards. This will take about five minutes, not counting the time to shut down and reboot the ROD crate. It is difficult to estimate the failure rate of the LSC mezzanine card or the ROD rear transition modules; however, it is expected to be a rare occurrence and sufficient spares will be available for replacement.

In the case of a failure of one of the ROL input channels integrated on the ROBin card, the complete card needs to be replaced, which implies a reboot of the ROS PC with the faulty card. Again this is expected to be a rare occurrence.

6.8.2 The ReadOut System

The ReadOut System (ROS) is the system that receives, buffers, and dispatches data fragments coming from the Readout links. The ROS is implemented as a number of individual units (up to around 150). Each unit is composed of an industrial PC housing a number of ROBin modules.

The ROBin modules are custom PCI devices receiving data from a number of Readout links (between 2 and 4 according to the implementation options) and providing the buffering. The ROBin boards are based on standard commercial components. Their failure rate is expected to be very low, and detailed studies will be performed when the first prototypes are available.

In the case of a failure of a ROBin module, the full ROS PC will need to be powered off. An adequate number of spare modules will be available to expedite repairs. The data of all the ROLs connected to the ROS (at least 12) will be unreadable for the full intervention period. An alternative intervention approach will be to dynamically mask out the ROBin module from the ROS, thus losing only the data of the ROLs connected to the module, and to delay the module replacement until the next natural pause in data taking. The intervention approach will have to be decided on a case-by-case basis, according to the importance of the ROLs connected to the faulty ROBin module.

The PCs used to house the ROBin modules will be commercial rack-mounted industrial PCs. The selection criteria for the PCs will require high reliability with such features as redundant power supplies. Every ROS unit is completely independent from the others. Thus the ROS sys-

tem itself does not need to be reconfigured in the case of a failure with a particular unit / PC, and it can work with any number of missing PCs. However, any hardware intervention on a ROS PC will imply the loss of all the data from the connected ROLs, and the need to reconfigure the DAQ accordingly. For example, the TTC system will have to be reconfigured to ignore any busy signal coming from the RODs which were connected to the missing ROS. The replacement of a full ROS PC is expected to be, in the best case, at least one hour.

From the software point of view the ROS is being designed to have a lower failure rate than the high-level trigger applications to cause a negligible impact on data acquisition efficiency. In particular the ROS software does not make use of any dynamic memory allocation scheme (all needed memory is statically pre-allocated), to eliminate memory leaks. Moreover the communication between the ROS and the HLT applications is based on a request-response, connectionless protocol, thus permitting the ROS to be insensitive to any problems occurring in the HLT applications.

6.8.3 The RoI Builder

The LVL1 system produces details about the type and position of the RoIs for all accepted events. The RoI Builder combines and reformats the RoI data it receives from the LVL1 Trigger and passes the RoI information corresponding to each event accepted by LVL1 to a LVL2 Supervisor. The RoI Builder is a critical, unique component using custom hardware. It is a highly modular system. It consists of two types of cards (input and builder cards). The cards and a controlling PC will reside in a standard VME crate. The card count is small for this system, but the cards are custom built. Since the system is critical for taking data, the cost of maintaining a substantial pool of replacements is low compared to the risk of not having adequate spares. The current plan is to include a reserve of spares and to run with some hot spares in the crate being used during data taking. If a builder board fails during data taking it can be replaced with a hot spare by simply reconfiguring the system via the DAQ. It is expected that laboratory space at CERN will be set aside for repair and that a backup system (crate, builder, and input boards) will be kept there in the event of more catastrophic whole system failure (e.g. a power supply failure that causes damage to multiple components in the crate).

6.8.4 Network

The topology of the TDAQ networks is ‘star based’, using several large central switches, and many concentrating switches, which are attached to the central switches using fibre Gigabit Ethernet links. A catastrophic failure of the central switches would stop the experiment, while a failure of a concentrating switch would lose part of the detector data or the processing farm attached to it.

Therefore, the central switches should have redundant power supplies, redundant CPU modules, and a redundant or at least hot swappable switching matrix. If the switch line cards are not hot swappable, there should be a few (one or two) standby line cards, in order to minimize the down time of the switch in the case of a line card failure. To ease technical support and availability of spare parts, the central switches should be as similar as possible (ideally identical).

The use of multiple central switches in the Data Collection network improves the fault tolerance. In the case of a total failure of one central switch, the system can still operate at a lower rate, until the failing switch is repaired or replaced. A concentrating switch failure does not

bring down the entire network. If a ROS concentrating switch fails, then the corresponding part of the detector is unreadable, and one may decide to stop the experiment, or continue running with reduced detector coverage. If a processor farm concentrating switch (such as the L2PU or the EF farms concentrating switches) fails, the corresponding farm will be excluded from the system configuration, and the system will continue to operate with a slightly lower processing capacity. Failures in concentrating switches can be dealt with by keeping an adequate number of spares and replacing the faulty units.

6.8.5 Rack-mounted PCs

The HLT/DAQ system will contain very large numbers of rack-mounted PCs. The reliability of modern PC hardware is generally high and particularly so for most rack-mounted PCs, as these are aimed at server markets where reliability is given a higher priority than for individual desktop machines. Indeed part of the cost premium of rack-mounted PCs is due to the greater care taken over the choice of components, better cooling, and ease of maintenance when something does go wrong.

Nevertheless with so many machines, faults will be an issue both in the hardware and the software.

For the hardware the most likely issues will be disk faults, cooling problems, and power supply failures. During normal data taking many of the PCs should not be accessing disk data, indeed for the most time-critical applications any such accesses risk a substantial downgrading of the performance. For these machines disk access is required for booting the PC, for configuration of the applications, and for recording of the monitoring data. All of these could be done across the network from a local file server (it is planned that each rack of PCs will have such a file server), but it remains to be seen whether this gives a better solution than using a local disk for some of the data accesses. Other machines require very high levels of disk access (the file servers, event store, etc), and for these RAID arrays are assumed. These not only reduce the susceptibility to disk errors, but also allow faulty drives to be replaced without stopping the host PC.

Cooling is a critical issue and several precautions are being elaborated. Studies are under way, in collaboration with the other LHC experiments, to investigate the efficiency of water-cooled heat exchangers (suitable for horizontal air-flow) to be placed at the back of the racks. All of the PCs will include on-board temperature sensors, and data from these will be monitored as part of the general farm management so that operator intervention can be requested if required. In addition the PCs include features to reduce the clock speed if over-heating occurs, if this is not sufficient the PC will shut down.

For the most critical PCs, redundant power supplies are foreseen. This includes the ROS PCs, the central file servers, and the DFM, since failures of any of these would cause a major impact on data taking. For the processors in the HLT farms where the failure of a single PC would lead to just a small reduction in data-rate, no such provision is planned. Should there be a power failure to the building, only the file-servers and event store are planned to be on UPS so that these machines can be shut down in an orderly way, thus reducing the risk of corrupting the file system. Putting all of the machines onto UPS would reduce recovery time and avoid some hardware failures, but at the cost of a very large UPS system. Also the faster recovery time from a power failure would be of little benefit as it will take a while for the detector to be ready for data-taking; during this time the bulk of the machines can be rebooted and reconfigured.

All of the above hardware problems are likely to require operator intervention at some point, but the system is designed to minimize the impact of the failure and to make replacement of the faulty components as simple as possible: ensuring, for example, that individual PCs can be removed from racks with minimal disturbance to cabling.

Faults due to software problems are likely to be much more frequent than hardware problems. Operating system (Linux) crashes should represent only a very small fraction of software errors, but care will be taken to ensure that these are not exacerbated by inadequate machine resources (e.g. insufficient memory, inadequate disk size for log files). Most software-related failures will be in the applications or individual threads within applications.

6.9 Use Cases

In the following, a short list is given of possible errors and reactions at different levels (from inside an application to system wide), and the impact on data taking.

- Symptom: Readout Link is down (LDOWN is active on S-LINK), status register and LEDs show error.
    - Action: Online Software should perform a reset of the ROL. If the ROL stays down after reset, prepare to mask BUSY from the ROD in order to continue taking data. Warn the person on shift that the ROL has a fault.
    - Impact: Data from the ROD connected to the ROL will be missing from the event. The ROS will produce empty ROB fragments with a specific error status flag.
- Symptom: Readout Link Data transmission error (bit set in trailing control word of fragment).
    - Action: Flag the Event Fragment, increment the error counter in the ROS.
    - Impact: The HLT will decide, depending on the type of error, whether to use or reject the fragment.
- Symptom: timeout for requests to a ROS inside a LVL2 node.
    - Action: Retry a configurable number of times.
    - Impact: Parts of an event might be missing. If not successful, the LVL2 trigger might not be able to run its intended algorithms and the event has to be force-accepted. If the error persists, the data-taking efficiency might drop because the event building will be busy with forced-accept events.
- Symptom: a dataflow component reports that a ROS times out repeatedly.
    - Action: Pause the run, remotely reset the ROS component and if successful resume the run. If not successful, inform all concerned components that this ROS is no longer available and inform the higher level (who might decide to stop the run and take other measures like calling an expert to re-configure the DAQ).
    - Impact: Data missing in every event.
- Symptom: a LVL2 supervisor event assignment to a LVL2 node times out.
    - Action: Retry a configurable number of times. If this happens repeatedly, take the node out of the LVL2 Supervisor's scheduler list and report to a higher level.

    - Impact: Available LVL2 processing power reduced but it might be possible to re-connect the LVL2 node if the fault can be repaired. The failing event could be forcibly accepted for more detailed offline analysis (as assigning the same event to another LVL2 node could possibly bring this node down too).
- Symptom: none of the nodes in an EF subfarm can be reached via the network (e.g. in the case of a switch failure).
    - Action: Take all affected nodes out of any scheduling decisions (e.g. in the DFM) to prevent further timeouts. Inform higher level about the situation.
    - Impact: Data-taking-rate capability is reduced.
- Symptom: HLT algorithm or data converter crashes (LVL2 or Event Builder).
    - Action: Similar to communications failure; the crashed node is temporarily taken out of the partition and brought back up again; the offending event is saved for offline scrutiny. Repeated crashes of many nodes may be indicative of faulty software, possibly the result of running with a new version or an untested trigger menu. In this case the run may need to be stopped and more drastic action (reverting to an older software version or different trigger menu) is required.
    - Impact: For sporadic errors the impact is similar to communications failure. The not unlikely case of running with faulty software leads to a longer loss due to change of run. The trigger may be less selective if one is forced to run with a reduced trigger menu.

As can be seen, the same error condition (e.g. timeouts for requests) leads to quite different actions depending on the type of component. Each ROS is unique in that its failure leads to some non-recoverable data loss. A LVL2 node on the other hand can be easily replaced with a different node of the same kind.

6.10 References

6-1 A. Bogaerts et. al., *ATLAS TDAQ Fault Tolerance and Error Handling Policy and Recommendations*, ATL-D-OR-0002 ()
http://edms.cern.ch/document/392826/

6-2 *S-LINK Interface Specification*,
http://hsi.web.cern.ch/HSI/s-link/spec/spec/s-link.pdf

6-3 *Small Form-factor Pluggable (SFP) Transceiver MultiSource Agreement (MSA)*,
http://schelto.com/SFP/SFP%20MSA.pdf

7 Monitoring

7.1 Overview

Fast and efficient monitoring is essential during data-taking periods as well as during the commissioning of the detector. The quality of the data sent to permanent storage must be monitored continuously. Any malfunctioning part of the experiment must be identified as soon as possible. The types of monitoring information may be events, fragments of events, or any other kind of information (histograms, counters, status flags, etc.). These may come from the hardware, processors, or network elements, either directly or via the DCS. Some malfunctions can be detected by the observation of a single piece of information and could be performed at the source of the information. Infrastructure has to be provided to process the monitoring information and bring the result to the end user (normally the shift crew).

The collection of monitoring information can be done at different places in the DataFlow part of the TDAQ system: ROD crate, ROS, and SFI. Additional monitoring information can be provided by the LVL2 trigger and by the Event Filter since they will decode data, compute tracks and clusters. LVL2 and the EF will count relevant quantities for simple event statistics, for monitoring the functioning of the various trigger stages, and for tracking their selection power.

Monitoring information that is collected may be processed locally, e.g. in the ROD module, or in dedicated processes running in the Event Filter. Additional processing may be done by a dedicated Online Monitoring Farm possibly also used for calibration activities. Results will be sent to the shift crew in the control room.

It is clear that monitoring cannot be precisely defined at this stage of the development of the experiment and should therefore be kept as flexible as possible. The ideas presented in this section are the result of a discussion that has started recently at the level of the whole ATLAS community [7-1], [7-2]. The ideas are bound to evolve during the preparation of the experiment, as well as during its lifetime.

7.2 Monitoring sources

7.2.1 DAQ data flow monitoring

7.2.1.1 Front-end and ROD monitoring

Here, sub-detector front-end electronics and ROD-module-specific monitoring is addressed. It deals with:

- Data integrity monitoring. Sampled event fragments must be collected at the level of the ROD and sent to dedicated monitoring processes to check the consistency of the data and the headers. Comparisons at the different levels of the transfers must be done. Use of the Data Collection network is preferred for this operation.

- Operational monitoring. Throughput and similar information, scaler counters, histograms produced by the RODs must be transferred to the end user for further analysis via the control network.
- Hardware monitoring, in cooperation with DCS. Additional monitoring of the hardware, not accessible to DCS (e.g. internal functioning of PCBs) may be provided at this level.

7.2.1.2 Data Collection monitoring

Here, it is DAQ-specific monitoring which is addressed. The same points as those listed in the previous section will be addressed, at the level of the Data Collection (ROB, Event Builder, SFI, SFO):

- data integrity monitoring
- operational monitoring (throughput and similar, scaler histograms)
- hardware monitoring.

7.2.2 Trigger monitoring

7.2.2.1 Trigger decision

The general philosophy of the trigger decision monitoring is to re-evaluate the decision of each trigger level on samples of both accepted and rejected events in order to confirm the quality of the decision.

7.2.2.1.1 LVL1 decision

The LVL1 decision (for LVL1 accepts) is cross-checked when doing the LVL2 processing. In addition, a pre-scaled sample of minimum-bias LVL1 triggers will be passed to dedicated processing tasks (possibly in a dedicated partition of the Event Filter or in an Online Monitoring Farm).

7.2.2.1.2 LVL2 decision

The LVL2 decision (for LVL2 accepts) is cross-checked when doing the EF processing and pre-scaled samples of events rejected at LVL2 will be passed to the EF. Detailed information on the processing in LVL2 is appended to the event (via the pROS) for accepted and force-accepted events. This will be available at the EF for further analysis.

7.2.2.1.3 EF decision

Detailed information will be appended to the event, for a sub-set of accepted and rejected events for further offline analysis.

7.2.2.1.4 Classification monitoring

For monitoring, classification is a very important output of both LVL2 and EF processing. It consists of a 128-bit bitmap that records which signatures in the trigger menu were passed. Histograms will be filled locally on the processors where the selection is performed. Considering the accept rates of LVL2 for EF, and assuming a sampling rate of ~0.1 Hz, a 1 byte depth is sufficient for the histograms. For both LVL2 and EF farms, the bandwidth for the transfer of the histograms is estimated to be ~3.5 kbyte / s.

7.2.2.1.5 Physics monitoring

In addition to classification monitoring, the simplest approach to monitoring the quality of the physics that is sent to permanent storage is to measure the rates for some physics channels. It can be performed easily in the EF. A part of the result of these monitoring operations will be appended to the event bytestream for offline analysis. Other data will be sent to the operator via the standard Online Software media for an online analysis and display.

An interesting approach to the physics monitoring could consist of a set of prescaled physics triggers with relaxed thresholds to monitor both the trigger decision and the effect of the trigger sharpness. Further studies will be performed to explore this approach.

Histograms of the rates for every item of the trigger menu as a function of time will be recorded, with the relevant variables with which they must be correlated (e.g. the instantaneous luminosity). Such histograms can very quickly give evidence of any malfunctions, although their interpretation may be quite tricky.

Well-known physics channels will be monitored so that one will permanently compare the observed rates with the expected ones. The list of such channels will be established in collaboration with the physics groups.

Information coming from the reconstruction algorithms executed for selection purposes in the EF may be of interest. One will monitor, for example, the number of tracks found in a given detector or the track quality at primary and secondary vertex on a per-event basis. There again, a comparison with reference histograms may be of great help in detecting malfunctioning. Physics quantities will be monitored, e.g. mass-plots of W and Z, n-Jet rates, reconstructed vertex location, quality of muon-tracks, quality of calorimeter clusters, and quality of ID tracks. Input is required from offline reconstruction groups.

7.2.2.2 Operational monitoring

This section covers all aspects related to the ‘system’, e.g. the transportation of the events or event fragments, the usage of computing resources, etc.

7.2.2.2.1 LVL1 operational monitoring

The integrity and correct operation of the LVL1 trigger hardware will be monitored by processes running in trigger crate CPUs, and also by monitoring tasks in the Event Filter.

The LVL1 trigger is the only place where every bunch crossing is processed and where a crude picture of the beam conditions can be found. For example, the calorimeter trigger fills histo-

grams in hardware of the trigger-tower hit rates and spectra. This is done for every trigger tower and can quickly identify hot channels which, if necessary, can be suppressed. Hardware monitoring is used to check the integrity of links between different parts of the LVL1 trigger, e.g. between the calorimeter trigger preprocessor and the cluster and jet/energy-sum processors. The Central Trigger Processor (CTP) will be monitored mainly at the ROD level, using internal scalers and histograms. It will include beam monitoring, including scaling of the trigger inputs on a bunch-to-bunch basis.

After the Event Builder, monitoring tasks, running in the Event filter or in a dedicated Monitoring Farm, will check for errors in the trigger processors on a sampling basis. This will be at a low rate, but will have high diagnostic power since details of the LVL1 caluclations can be checked. Event Filter tasks will also produce various histograms of trigger rates, their correlation, and history.

7.2.2.2.2 LVL2 operational monitoring

The LVL2 selection software runs within the framework provided by the Data Collection (DC) in the L2PU. It will therefore use the DC infrastructure and hence the monitoring tools foreseen for this system. The following relevant aspects of LVL2, will be monitored:

- trigger, data, and error rates
- CPU activity
- queue occupancies (load balancing).

Other valuable pieces of information for monitoring which could be performed by LVL2 processing units are:

- LVL2 selectivity per LVL1 trigger type
- number of RoIs per RoI type
- RoI occupancies per sub-detector
- RoI specific hit-maps per sub-detector.

In contrast to these data which do not require much CPU power, the monitoring of the quality of the data by LVL2 processors is not envisaged. Indeed, the available time budget is limited because of the necessity to release data from the ROB. Monitoring a fraction of the events in the L2PU is not desirable since this would introduce large variations in LVL2 latencies as well as possible points of weakness in the LVL2 system. The necessary monitoring of the LVL2 quality is therefore delegated to the downstream monitoring facilities, i.e. the EF (or online monitoring farm) and the offline analysis. One should, however, discuss very carefully the opportunity for the L2PU to fill some histograms, possibly read at the end of the run, so that high statistics information is given. This information can not reasonably be obtained by using events selected by forced accepts on a pre-sampled basis. The evaluation of the extra CPU load for such operations should be made.

7.2.2.2.3 EF operational monitoring

The monitoring of the data flow in the Event Filter will be done primarily at the level of the Event Filter data flow (EFD) process. Specific EFD tasks, part of the main data flow, will be in charge of producing relevant statistics in terms of throughput at the different levels of the data

flow. They have no connection with other processes external to EFD. The detailed list of information of interest for the end user has not yet been finalized and will continue to evolve throughout the lifetime of the experiment.

Among the most obvious parameters which are going to be monitored, one might quote:

- the number of events entering the Farm
- the number of events entering each sub-farm
- the number of events entering each processing host
- the number of events entering each processing task
- the number of events selected by each processing task, as a function of the physics channels present in the trigger menu
- the same statistics as above at the level of the processing host, the sub-farm, and the Farm.

Other statistics may be of interest such as the size of the events, as a function of different parameters (the time, the luminosity of the beam, and the physics channel).

The results of the data flow monitoring will be sent to the operator via standard Online Software media (e.g. IS or Histogram Service in the present implementation).

7.2.2.2.4 PESA SW operational monitoring

A first list of parameters which could be monitored for debugging purposes and comparison with modelling results can be given:

- time spent in each algorithm
- frequency at which each algorithm is called
- number of steps in the step sequencer before rejection
- info and debug messages issued by the PESA SW
- number of active input/output trigger elements.

Some of these points could be monitored during normal data taking.

Profiling tools, such as NetLogger for coarse measurements and TAU, have already been studied in the context of LVL2, and their use on a larger scale will be considered by the PESA software team.

It is intended to make use of the ATHENA Histogramming service, which should therefore be interfaced to the EF infrastructure. Some control mechanisms should be provided to configure the various monitoring options and to operate on the histograms (e.g. to reset them after transfer).

7.2.3 Network monitoring

Monitoring the TDAQ network activity is essential during the data-taking period. A very large number of switches (several hundred) will be included in the network. Their management and

monitoring is crucial. A first experience has been gained with the testbed used for network and data flow studies. Requirements are presented in [7-3].

Preliminary studies have been undertaken for tool evaluation. A first evaluation of the tool used by CERN-IT (*Spectrum*, from Aprisma) has been made. It has been found quite mature and it offers most of the expected functionality. Nice graphical interfaces are available giving an accurate and explicit view of the system (Figure 7-1). Extra work will be required to integrate the tool with the Online Software environment.

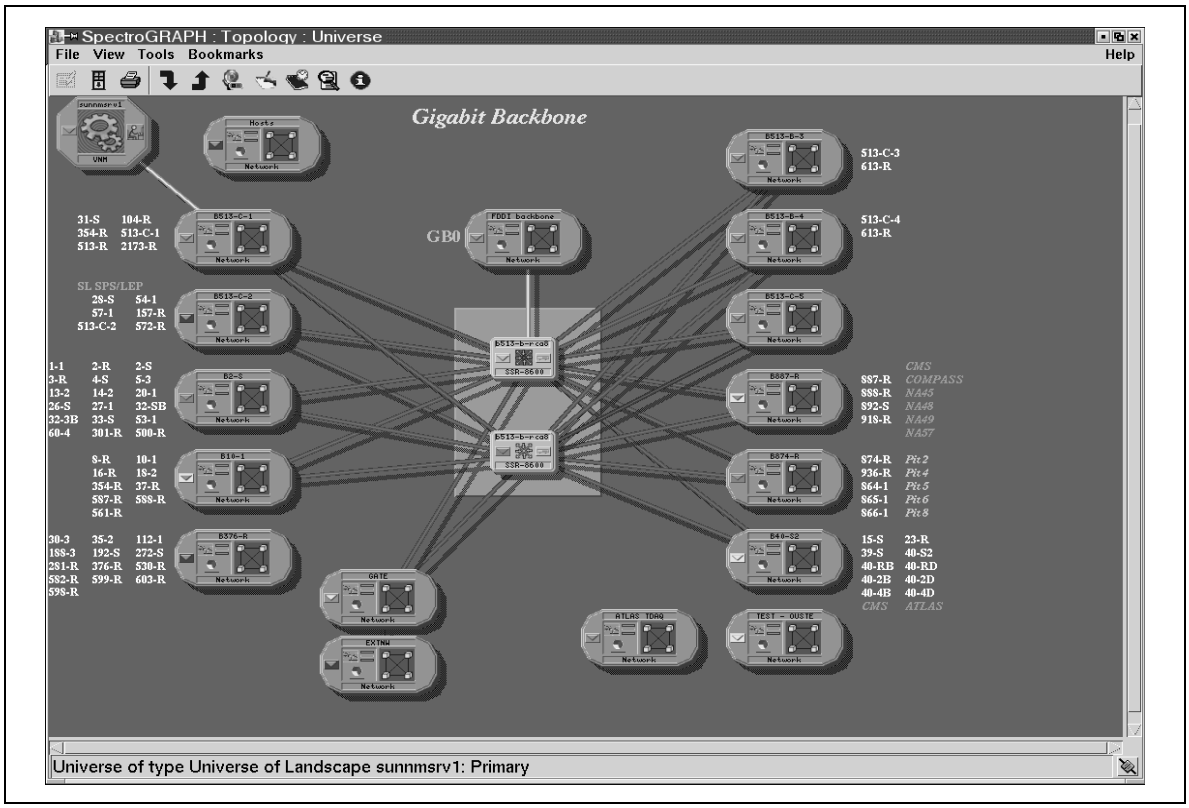

Figure 7-1 Example of a user interface of the Spectrum tool

7.2.4 Detector monitoring

The detector monitoring can be done at different places in the DataFlow part of the TDAQ system: ROD Crate, ROS, and SFI. Moreover, additional monitoring can be provided by the LVL2 trigger and by the Event Filter since these programs will decode data, compute tracks and clusters, count relevant quantities for simple event statistics, and monitor the functioning of the various trigger levels and their selection power.

The ROD level is the first place where the monitoring of the data quality and integrity can be easily done. The computing power provided, for example, by the DSPs installed directly on the ROD board allows sophisticated calculations to be performed and histograms to be filled. The sending of these histograms to analysis workstations will then be performed by the ROD crate CPU (using the Online Software tools running on this CPU).

Some detectors will need a systematic monitoring action at the beginning of the run to check the integrity of the system. This concept has already been introduced at the test beam by the Pixel

sub-detector. At the beginning of the run and at the end there are special events with start and end-of-run statistics. The need for this first check of the data at the ROD level is driven by the huge amount of information. If monitored later, the back tracking of possible problems would be complicated. The frequency of this activity, for normal operation, can be limited to the start and end of run.

Larger parts of the detector are available at the ROS level (compared to the ROD level), and monitoring at this level is therefore considered as a potentially interesting facility. A correlation between ROS crates is not seen as necessary because such a correlation may be obtained at the SFI level. Event fragments sampled at the level of the ROS could be routed to dedicated workstations operated by the shift crew.

When information from several detectors is needed, the natural place to monitor it is after the Event Builder. The SFI is the first place were fully-assembled events are available. The monitoring at the level of the SFI is the first place in the data flow chain where the calorimeter, muon spectrometer, and LVL1 information are available. This can be used, for example, to cross-check the data for LVL1 calorimeter trigger towers against the more-precise, fine-granularity calorimeter readout. Moreover at the SFI level a first correlation among sub-detectors is possible and is seen as extremely useful.

When the monitoring requires some reconstruction operations, it seems natural to try to save computing resources by re-using the results obtained for selection purposes and therefore to execute this activity in the framework of the EF. In addition, some detectors plan to perform monitoring at the level of event decoding, i.e. in the bytestream conversion service, and to fill histograms during the reconstruction phase associated with the selection procedure in the EF. These histograms should be sent regularly to the shift operators and archived. More sophisticated monitoring operations might require longer execution times. However, since CPU power available in the EF should be kept as a first priority for selection, it seems more efficient to have a dedicated monitoring farm running beside the EF.

Finally, some monitoring activities such as calibration and alignment checks may require events with a special topology selected at the level of the Event Filter. For instance, the Inner Detector group plans to perform the alignment of the tracking system online. This requires thousands of selected events, to be either stored on a local disk or fed directly to the processing task. Then CPU-intensive calculations are required to invert matrices which may be as large as $30\,000 \times 30\,000$. With a cluster consisting of 16 PCs (as available in 2007, i.e. 8 GHz dual CPU PC with 1 Gbyte of fast memory and a 64-bit floating point arithmetic unit), this can be completed in less than one hour. A very efficient monitoring of the tracker alignment can therefore be performed. Similar requirements are made by the Muon Spectrometer for the purpose of monitoring and calibration operations.

7.3 Tools for monitoring

This section describes where and how (i.e. with which tools) monitoring operations will be performed.

7.3.1 Online Software services

The Online Software (see Chapter 10) provides a number of services which can be used as a monitoring mechanism independent of the main data flow stream. The main responsibility of these services is to transport the monitoring data requests from the monitoring destinations to the monitoring sources, and to transport the monitoring data back from the sources to the destinations.

There are four services provided for different types of monitoring information:

- **Event Monitoring Service** — responsible for the transportation of physics events or event fragments sampled from well-defined points in the data flow chain and delivered to analysis applications. These will examine and analyse the data in order to monitor the state of the data acquisition and the quality of the physics data in the experiment.
- **Information Service** — responsible for the exchange of user-defined information between TDAQ applications and is aimed at being used for the operational monitoring. It can be used to monitor the status and various statistics data of the TDAQ sub-systems and their hardware or software elements.
- **Histogramming Service** — a specialization of the Information Service with the aim of transporting histograms. It accepts several commonly used histogram formats (ROOT histograms for example) as the type of information which can be sent from the monitoring sources to the destinations.
- **Error Reporting Service** — provides transportation of the error messages from the software applications which detect these errors to the applications responsible for their monitoring and handling.

Each service offers the most appropriate and efficient functionality for a given monitoring data type and provides specific interfaces for both monitoring sources and destinations.

7.3.2 Computing resources for monitoring

7.3.2.1 Workstations in SCX1

It is foreseen to have several workstations in the the SCX1 Control Room near the experiment pit. These workstations will be owned and operated by the sub-detector crews who are on shift. They will receive, via the Ethernet network, the results coming from operations performed in ROD and ROS crates as well as event fragments. Whether the network is a dedicated one or the general-purpose one is still an open question. These workstations will perform subsequent treatment such as histogram merging or event display. They may possibly delegate processing to machines in remote sites if available local CPU power is not sufficient. Results of monitoring operations will be made available to the whole shift crew using the dedicated Online Software services.

7.3.2.2 Monitoring in the Event Filter

From the beginning of the design of the EF, it has been foreseen to perform some monitoring activities there, in addition to the ones related directly to the selection operation. EF is indeed the first place in the data-selection chain where the full information about the events is available.

Decisions from the previous levels of the trigger system can be checked for accepted events and for those that would normally have been rejected, but were retained on the basis of a forced accept. Information coming from the reconstruction phase, which generally requires a large amount of CPU power, can rather easily be re-used, leading to large savings in terms of computing resources.

Monitoring in the EF can be performed in different places:

- directly in the filtering tasks (which raises the problem of the robustness of the monitoring code) and,
- in dedicated monitoring tasks running in the context of the Event Filter (here one should think of passing the information gathered in the filtering task to take advantage of the CPU power already used).

As already stated, the first priority of the EF must be the selection procedure which should not be jeopardized by introducing some CPU-intensive applications in the processing host. Monitoring in the EF should therefore be reserved to light-weight applications which would profit most from re-using pieces of information produced by EF Processing Tasks.

7.3.2.3 Monitoring after the Event Filter

In order to perform the CPU-intensive monitoring activities such as the ones described at the end of Section 7.2.4, a dedicated Online Farm should be provided. Such a farm is also necessary to perform the various calibration tasks which do not require the full offline framework. It would be fed by events specially selected in the EF, as well as directly by the general DataFlow through one or more dedicated SFIs (so that it may receive events selected at previous stages of the data acquisition chain). Such specially selected events may be events rejected at LVL1 or LVL2 (on a prescaled basis), or events tagged for calibration purposes (physical as well as non physical events, e.g. generated by a pulser or corresponding to empty bunches).

If such an Online Farm were to be used, one would require that the Data Flow Manager be able to route events towards specific SFIs according to a tag set at various levels of the data acquisition chain (front end, LVL1, or LVL2). The DataFlow framework developed for the Event Filter seems to be well suited for the distribution of the events to the different applications. Moreover, a uniform approach for the EF and the Online Farm would bring some flexibility for the global computing power usage. Indeed intensive monitoring is likely to be a higher priority during commissioning or debugging phases compared to physics quality, and conversely during the stable physics-data-taking phase. The size of this Online Farm is still to be evaluated.

7.3.2.4 Network requirements

An indication of the network requirement for the purpose of monitoring can be found in Table 2-4 of Chapter 2, which shows the quantity of monitoring data foreseen between the various sources and destination of this data.

7.4 Archiving monitoring data

Data which are produced by monitoring activities should be archived by some bookkeeping service so that they can be cross-checked offline with more detailed analysis. One must also store events whose acceptance has been forced at any level of the selection chain. These events are necessary to evaluate precisely the acceptance of the trigger. Details of the archiving procedure will be established in cooperation with the Offline Computing group.

7.5 References

7-1 B. Di Girolamo et al., *Introduction to Monitoring in TDAQ*, ATL-D-OR-001 () https://edms.cern.ch/document/382428/

7-2 B. Di Girolamo et al., *ATLAS Monitoring Requirements*, in preparation

7-3 M. Zurek, *Network Management Requirements*, ATL-DQ-ES-0045 () https://edms.cern.ch/document/391577/

Part 2

System Components

8 Data-flow

This chapter presents the results of studies that have been performed, using the final prototype implementations of the DataFlow components, to support the choice of the baseline DataFlow architecture presented in Chapter 5. Results of measurements are presented on the performance of the main components and on the functionalities that they provide in collaboration: detector readout, message passing, RoI collection and event building. The results of measurements are presented in terms of sustained rates, e.g. the event building rate, versus various system parameters, for example the size of the system. The measurements were performed on a testbed whose description is in given in Section 14.2.2. In this same section the overall performance of the DataFlow is presented.

This section also presents details of the design and implementation of the DataFlow components, building upon the descriptions so far presented, see Chapter 5.

8.1 Detector readout and event fragment buffering

8.1.1 ROD crate data acquisition

ROD Crate DAQ comprises all the software and hardware to operate one or more ROD Crates and is deployed on the ROD Crate Processor (RCP)[1] of the ROD Crate and a controlling PC [8-1]. It provides the functionality of configuration, control, and monitoring of one or more ROD crate systems independently of the rest of the DataFlow system. This fulfils the detector community's requirements of operational independence during their commissioning and calibration procedures. This model also allows the timescale for the deployment and commissioning of the DataFlow to be decoupled from that of the detector's readout needs. During normal experiment operations the same ROD crate DAQ is operated as an integral part of the overall DAQ.

1. The RCP is implemented by a Single board Computer (SBC).

A block diagram of ROD crate DAQ is shown in Figure 8-1. Event data flows into the RODs via

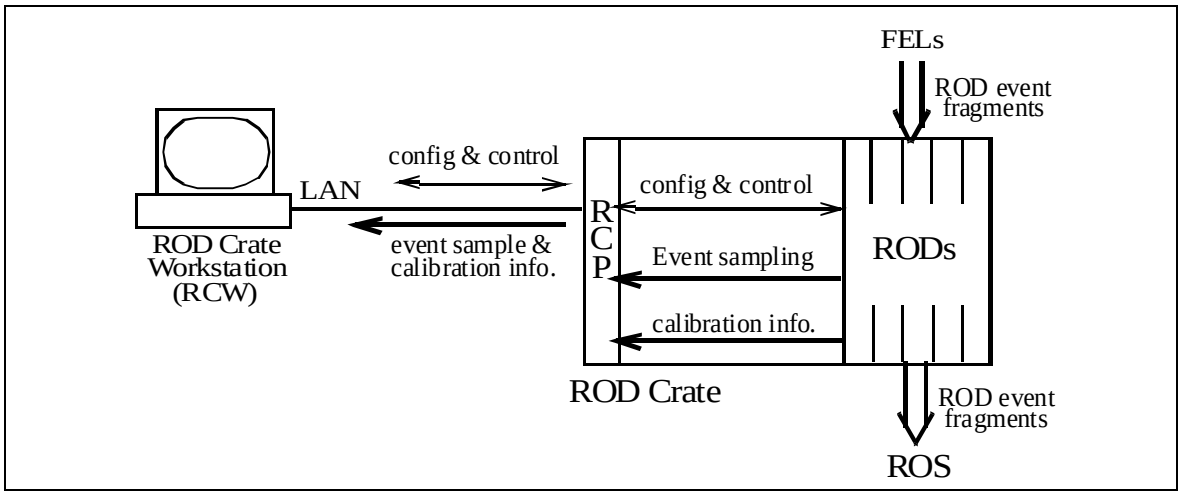

Figure 8-1 Block diagram of the context and contents of ROD Crate DAQ

the FELs and out via the S-LINK. ROD Crate DAQ provides for the sampling of ROD event fragments within this flow. The sampling may be from a ROD, a set of RODs[1] within the same crate, or one or more ROD crates. In addition to the sampling of ROD event fragments, ROD crate DAQ also provides for the retrieval of the results of calibration procedures performed within the RODs. Subsequently the sampled data may be further analysed, recorded to mass storage, or, in the case of calibration data, written to the conditions database. The detector requirements on the sampling rate is O(Hz) [8-1]. This non-demanding requirement allows the sampling from one or more ROD crates to be done over a LAN using TCP/IP, thus simplifying certain aspects of the design and implementation.

The framework of ROD Crate DAQ is organized into four layers: hardware, operating system, low-level services, and high-level tasks. The latter are based on a skeleton that implements detector-independent functionality, and, for an identified set of functions, may be extended to meet detector-specific requirements.

ROD Crate DAQ re-uses DataFlow and Online software where possible. The ROD Crate controller is an extension of the Local Controller developed for the ROS, see Section 8.1.3.1, and as such implements all the functionality provided by the Online software for configuration, control, and monitoring. In addition, other ROS software modules are used to provide the high-level task skeleton and low-level services. A prototype system is being developed, now in conjunction with detector-specific developers, and a first distribution is scheduled for June 2003 [8-2].

8.1.2 ReadOut link

Sub-detectors transmit event data accepted by the LVL1 over FELs and use RODs to multiplex the data. Each of the sub-detectors has different requirements and consequently the implementation of the ROD varies between sub-detectors. The guidelines for designing the ROD are set

1. The coherent sampling from one or more RODs depends on the implementation of the ROD.

out in the Trigger & DAQ Interfaces with Front-End Systems: Requirement Document [8-3]. The purpose of the ROL is to connect the sub-detectors to the TDAQ system and it is responsible for transmitting error-free data from the output of the ROD to the input of the ROS, i.e. the RoBIn.

The ROL requirements have been stable since the High-level Triggers, DAQ and DCS Technical Proposal TP [8-4]:

- 32 bit data at 40.08 MHz, (~ 160 Mbyte / s)
- A control bit to identify the start and end of an event
- Xon / Xoff flow control
- Error detection, error rate < 10^{-12}
- A maximum length of 300 m.

To ensure homogeneity, the output of the ROD is defined by the S-LINK specification [8-5]. In addition, the raw event format [8-6] defines the order and content of the information transmitted from the ROD. At the other end of the ROL, the ROS inputs are identical for all sub-detectors and also conform to the S-LINK standard.

The S-LINK specification has been stable since 1996. It is an interface definition; it only defines a protocol and recommends connector pin-out. As shown in Figure 8-2, the ROD end of the ROL is called the Link Source Card (LSC) and the receiver end, implemented by the RoBIn, the Link Destination Card (LDC) and they are connected by optical fibres. Event data flows from the LSC to the LDC on the forward channel. Flow control information, i.e. the ROS can stop the ROD sending data if input buffers are almost full, flows from the LDC to the LSC on the return channel.

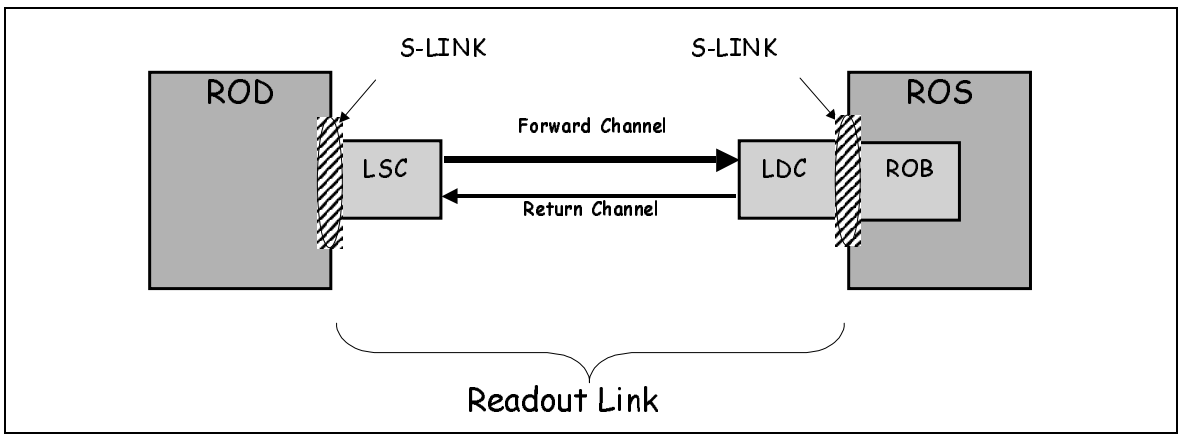

Figure 8-2 The relationship between the S-LINK and the ROL

The DIG–ROD Working Group have also recommended that the LSC be placed on a mezzanine card to facilitate support and upgradeability [8-7]. The form factor of these mezzanine cards is based on the CMC [8-8] standard.

The LSC plugs onto the S-LINK connector on the ROD (or its associated rear transition card). For the forward channel, a Field Programmable Gate Array (FPGA) handles the protocol and delivers words to a serial / deserializer (SERDES) chip which performs parallel-to-serial data conversion and encoding. The output of the SERDES drives an optical transceiver that in turn feeds the optical fibre. The operation of the receiving card, the LDC, is a mirror image of the LSC.

Various prototype implementations of the ROL have been built to prove the concept and measure the performance. A previous version of the ROL, the ODIN, used a physical layer that was based on the Hewlett Packard G-LINKs (HDMP–1032/1034). These have also been used successfully in test-beams and eighty of these ROLs are being used in the COMPASS experiment. However, the maximum bandwidth is limited, by the G-LINK, to 128 Mbyte/s. Following the second ROD Workshop, the requirements of the ROL were increased to 160 Mbyte/s and a second version of this link was designed which used two G-LINKs chips per channel. This raised the cost as two pairs of fibres and associated connectors are required.

Another recommendation of the ROD Working Group was to build a ROL that would use only one pair of fibres. This has been achieved by using 2.5 Gbit/s components in the current design, the High-speed Optical Link for ATLAS (HOLA) [8-9]. In this implementation a small FPGA, the EP20K30E APEX 20K, handles the S-LINK protocol. The SERDES chip is a Texas Instruments TLK2501 running at 2.5 Gbit/s for both the forward and the return channels (one per card). For the optical transceiver, the Small Form Factor Pluggable Multimode 850 nm 2.5 Gbit/s with LC Connectors is recommended, e.g. the Infineon V23818-N305-B57. The use of pluggable components allows the optical components to be changed in case of failure.

Test equipment has been developed for the ROD/ROL/ROS. This includes an emulator that can be placed on the ROD to check that the ROD conforms to the S-LINK specification. Similarly, an emulator exists that can be placed on a ROS to emulate a ROL connection. The emulators allow ROD, ROL and ROS designs to be tested at full bandwidth and errors to be introduced in a controlled manner. The HOLA was produced and tested in 2002 and satisfies all requirements of the ROL.

In addition, for the purposes of exploitation in laboratory test set-ups and in test-beams, i.e. further testing, cards exist which allow the ROL to be interfaced to the PCI Bus in a PC. Performance measurements of this interface [8-10] have shown that data can be transferred into a PC at 160 Mbyte/s using a single ROL input. Modifications to the firmware have allowed the emulation of an interface with four ROL inputs. Measurements using this emulator have demonstrated a bandwidth of 450 Mbyte/s into a PC. The latest version of the interface, the FILAR, has been produced and tested in 2003. It incorporates four ROLs and could be tested during the course of 2003 test-beams.

The purchase of the cards, in small quantities, is handled by the CERN stores. For quantities required for ATLAS a tendering process will be initiated in 2003 thus ensuring the availability of larger quantities during 2004. The production schedule will be adapted to the requirements of the sub-detectors. Those responsible have been asked by the DIG to provide estimates of quantities for the years up to the start of the LHC. Maintenance and short-term loans of equipment will be handled by CERN.

8.1.3 ReadOut subsystem

8.1.3.1 High-Level Design

The ROS has three major components: the RoBIn, the IOManager and the LocalController. Figure 8-3 shows the relationship between the three ROS components and other relevant TDAQ components. A complete high-level design covering both the software and hardware aspects of the prototype ROS can be found in Ref. [8-11], only a summary is presented here.

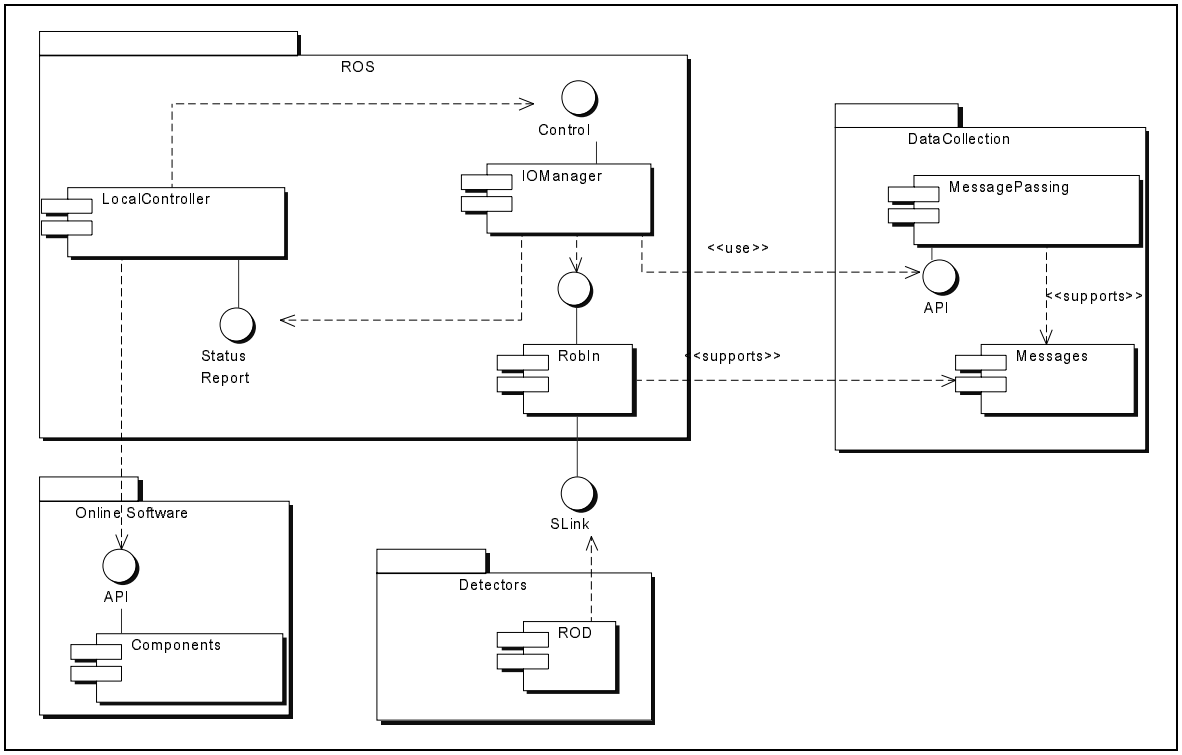

Figure 8-3 Main ROS components and their relationship with the other TDAQ components

The RoBIn component provides the temporary buffering of the individual ROD event fragments produced by the RODs, it must receive ROD event fragments at the LVL1 accept rate, i.e. 100 kHz. All incoming ROD event fragments are subsequently buffered for the duration of the LVL2 trigger decision and, for approximately 3% of the events, the duration of the event building. In addition, it must provide ROD event fragments to the LVL2 trigger, and, for events accepted by the LVL2 trigger, ROD event fragments to the Event Building.

Because of these requirements the baseline RoBIn is custom-designed and built. The design of the prototype is described in Section 8.1.3.2.

The IOManager is deployed in the bus-based ROS scenario. It services the requests for data by the L2PUs and the SFIs. According to the criteria specified in the data request, the IOManager collects one or more ROB fragments from one or more RoBIns, builds a ROS fragment and sends it to the destination specified in the original data request. The IOManager also receives from the DFM the request to release buffer space occupied by ROD event fragments in the RoBIns. This message is subsequently relayed to the RoBIns. So as to maximize the overall performance of the ROS, the design of the IOManager allows a number of data requests and releases to

be handled concurrently. This has proven to provide significant performance improvements [8-13] and is achieved by implementing the IOManager as a multithreaded software process.

In the switch-based ROS scenario there is no IOManager as each RoBIn receives data request and release messages directly from the L2PUs, SFIs and DFM via its direct connection to the DataFlow network.

The LocalController provides the interface between the ROS and the Online software. It configures, controls and monitors all components within a ROS. Monitoring covers both the operational monitoring, e.g. buffer page utilization and queue size, and the provision of a sample of the event fragments flowing through the ROS for the purposes of detector monitoring. This functionality requires the use of Online services that are not subject to the same demanding performance requirements as the IOManager. Thus the LocalController separates the non performant control, configuration and monitoring functionality from the more demanding data handling functionality. The design of the LocalController is based on a generic framework encapsulating ROS-specific functionality; thus allowing the generic LocalController to be re-used within ROD crate DAQ.

8.1.3.2 Design of the RoBIn

As described in Chapter 5 the RoBIn is located at the boundary between the detectors and the ROS. Its context is shown in Figure 8-3. Within this context it provides the functionality of:

- receiving ROD event fragments from the ROL
- buffering ROD event fragments
- sending ROD event fragments, on request, to the L2PUs and SFIs
- releasing buffer space, on request from the DFM.

The final prototype of this component, described here, takes into account the experience and results of studies from previous prototyping studies [8-12], [8-13] and the requirements on it are documented in the ROS-URD [8-14]. Its complete design and implementation are described in [8-15], [8-16] and [8-18].

Referring to Figure 8-4, the primary functions of the prototype RoBIn (receive, buffer, send and release) are mapped onto a small number of specialized building blocks. It supports two ROLs, the data from which are stored in a separate buffers. All functions related to the receiving of ROD event fragments from the ROLs, i.e. operations occurring at up to 100 kHz, are realized in an FPGA. A CPU is used to implement the functions of memory management, servicing of data requests, and operational monitoring.

The baseline architecture allows I/O between the ROS and other DataFlow components to be performed via two I/O paths. These I/O paths may be used exclusively or together. As described in Section 5.5.4, one of these scenarios is termed the bus-based ROS, while the other is termed the switch-based ROS. These terms reflect the fact that in each case data from a number of RoBIns are collected exclusively via a PCI bus or an Ethernet switch.

To allow these two I/O scenarios to be further studied, the prototype RoBIn features both a PCI bus and a Gigabit Ethernet interface. The basic set of services, e.g. data request and clears, that the prototype RoBIn provides via these interfaces is defined via a single software interface [8-18], and those operations related to a specific I/O interface have been encapsulated in separate modules. It is envisaged that the design of the final RoBIn will be realized by removing and not

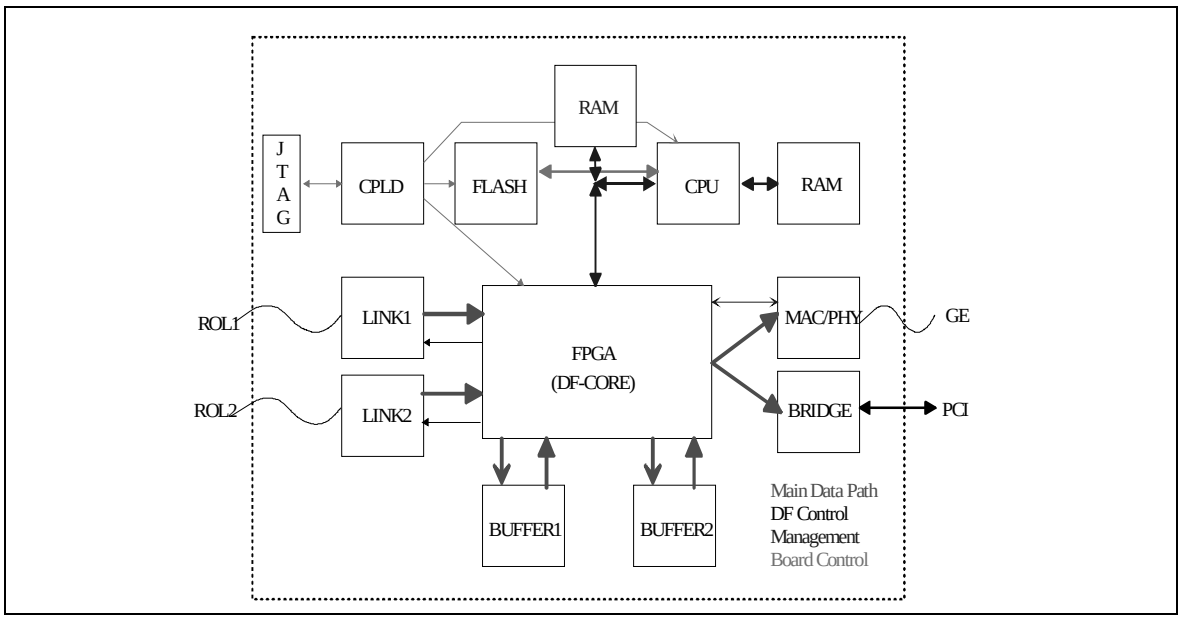

Figure 8-4 Schematic diagram of the final prototype RoBIn

by adding functionality, e.g. all buffer management performed by the FPGA or the removal of the Ethernet interface.

8.1.3.3 Implementation and performance of the ROS

The deployment of the bus-based ROS is shown in Figure 8-5. It consists of two types of nodes: a ROS PC and the prototype RoBIn. The former is a desktop PC running the Linux operating system and has a Gigabit Ethernet connection to the DataFlow network and a Fast Ethernet connection for the purpose of communication with the Online system. In addition, it has four 64 bit / 33 MHz and 3.3 V PCI bus slots. These slots are used to host three prototype RoBIns. The IOManager via the Message Passing interface (see Section 8.3.1.3) receives data requests and release messages, and returns ROS event fragments to the High-Level Trigger components.

Figure 8-6 shows the deployment of the switch-based ROS. Similarly to the bus-based ROS it consists of two types of nodes: a ROS PC and the prototype RoBIn. However, in this case the ROS PC node has not Gigabit Ethernet connection to the DataFlow network and is used solely for the purposes of configuration, control and monitoring. Data requests and clears are received directly by the RoBIns via its Gigabit network interface, i.e. without passing via the IOManager.

Measurements have been made on the performance of these two scenarios. However, the production schedule of the prototype RoBIn has not allowed the prototype RoBIn to be available for the measurements presented in this chapter. In its absence measurements have been performed with emulators (simple hardware devices providing a subset, I/O, of the RoBIn functionality). For the bus-based ROS measurements the emulator was based on the MPRACE board [8-19]. These boards have the same physical PCI bus interface as the prototype RoBIn, and thus provide a very accurate emulation of the final device with respect to I/O over a PCI bus. However, their internal performance is limited to half the PCI bus throughput. For the switch-based ROS the Gigabit Ethernet testers developed for the evaluation Gigabit Ethernet have been used [8-20]. Note that neither flavour of emulator receives ROD event fragments via S-LINK. ROS event fragments are generated on request and sent to the requester via PCI bus or gigabit Ethernet.

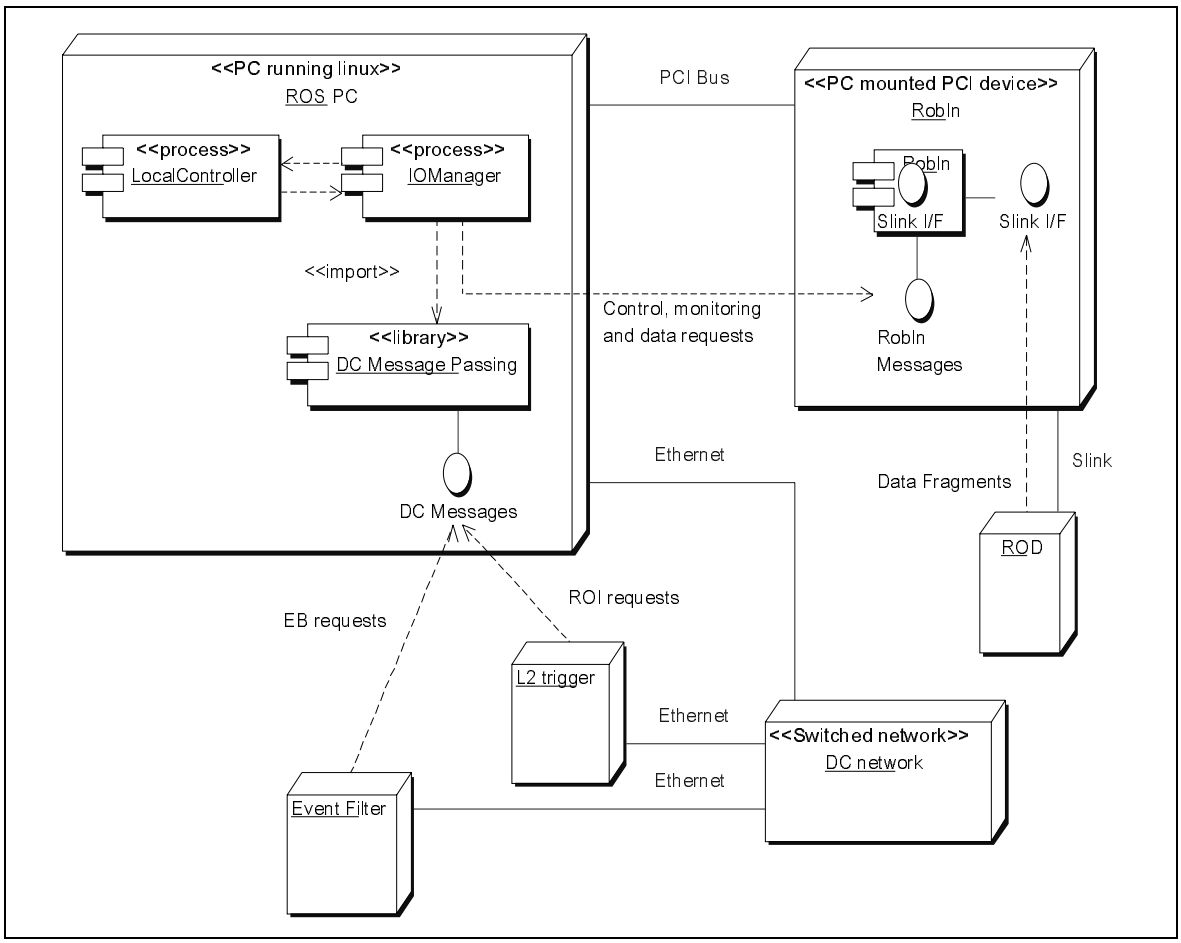

Figure 8-5 Deployment of the bus-based ROS in the baseline DataFlow

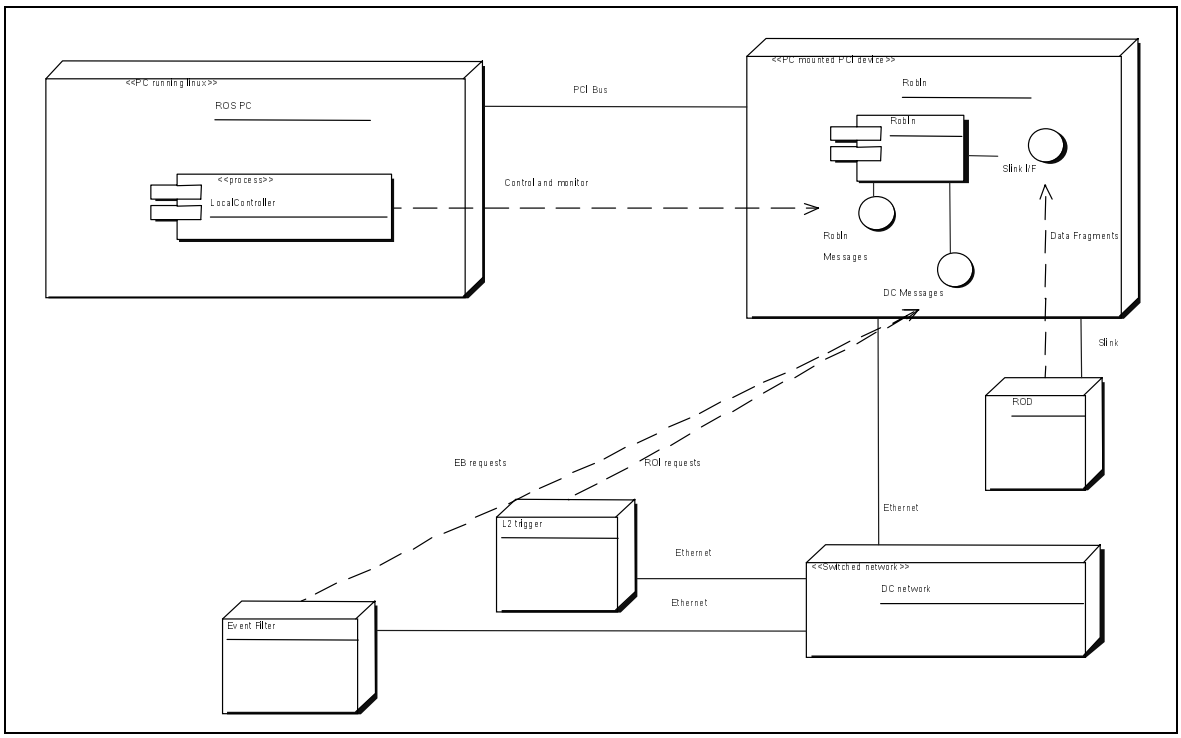

Figure 8-6 Deployment of the switch-based ROS

The main results obtained from studies of the bus-based scenario are presented here, while the main results of switch-based ROS measurements are presented in conjunction with the results on RoI collection and event building, see Section 8.3.2.2 and Section 8.3.2.3. More detailed results are documented elsewhere [8-21].

Figure 8-7 shows the set-up for the bus-based ROS testbed. In this testbed an IOManager was deployed on a standard PC having a single 2.4 GHz Xeon processor and a PCI bus configuration including four 66 MHz / 64 byte PCI buses, running RedHat Linux 7.3. The PC was equipped with three MPRACE boards, each emulating a RoBIn with four ROLs, giving a total of twelve ROLs per ROS.

The testbed has been operated in a standalone configuration, where the IOManager generated triggers internally and the ROS Fragments produced discarded, and in a configuration where the IOManager was receiving real data request messages from a test program via the network and sending back the ROS Fragments to the requesting process. The first configuration allows the measurement of the performance of the internal ROS operations (scheduling of concurrent data requests, collection of data fragments from the PCI RobIns, building combined data fragments). The second configuration includes the overhead of the network I/O associated with receiving data requests and sending data fragments.

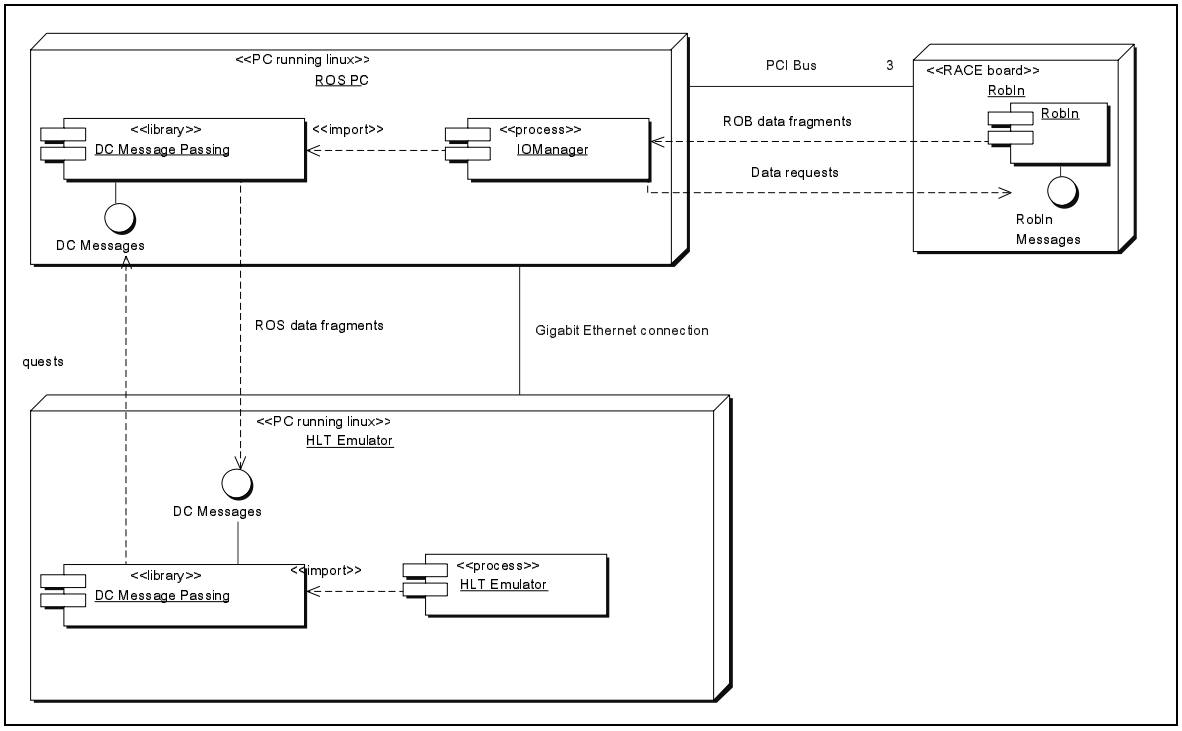

Figure 8-7 The bus-based ROS performance testbed

In the final ATLAS set-up, ROS modules will be exposed to a rate of EB data requests depending on the LVL2 rejection power, and to different rates of RoI data requests depending on which detector channels they are connected to. In addition to the data request messages the ROS will receive messages containing the list of events that have to be removed from its buffers.

The test procedure involved measuring the maximum LVL1 accept rate that a ROS could sustain as a function of the LVL2 rejection factor. The measurements have been carried out both for

average and maximum RoI data request scenarios referred to as ‘standard’ and ‘hotspot’ ROS in the remainder of this section.

The ROI data request rates and throughputs for standard and hotspot ROSs have been derived from the paper model results presented in Appendix A. A standard ROS receives RoI data requests at a rate corresponding to ~12% of the LVL1 rate, each requesting one kilobyte of data from a single ROL. A hotspot ROS instead will receive RoI data requests at a rate corresponding to ~20% of the LVL1 rate, each requesting data from one or two ROLs, for a mean output fragment size of ~1.4 kbyte.

Figure 8-8 shows the maximum sustained LVL1 rate as a function of the fraction of accepted events, i.e. the inverse of the LVL2 rejection power, for both a standard ROS and a hotspot ROS.

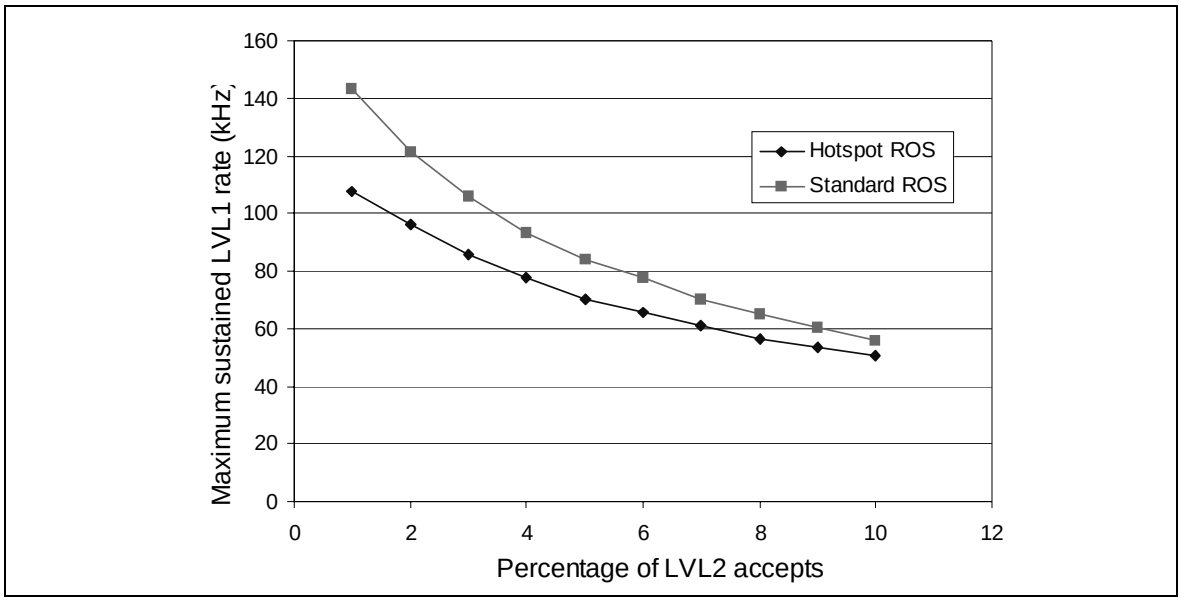

Figure 8-8 The bus-based ROS sustained LVL1 rate as a function of the event-building rate for different values of the data volume requested by the LVL2 trigger

It can be seen that for an accept percentage of 4%, i.e. a LVL2 reduction power of twenty-five, a hotspot ROS can sustain up to 78 kHz of LVL1 rate while a standard ROS can sustain up to 94 kHz. It can also be observed that a standard ROS sustains a LVL1 accept rate of 75 kHz for an accept fraction of up to ~6.5%, i.e. a LVL2 reduction power of sixteen.

In Figure 8-9 the maximum sustainable LVL1 rate, as shown in Figure 8-8 for a hotspot ROS, is compared to the internal performance of the ROS as measured running without network I/O. For the upper curve and for an accept percentage of 4%, the fragment flow from the MPRACE modules is ~150 Mbyte/s and increases up to ~215 Mbyte/s for an accept percentage of 10%, confirming the observation that the PCI bus can be driven to data rates much larger than those required of the final system.

Figure 8-10 shows the results of a more complex test set-up where four bus-based ROSs, each emulating the input from twelve ROLs, were connected through a Gigabit Ethernet switch to a number of SFIs, requesting data as quickly as possible. One can observe that the ROSs reach a sustained rate of 6 kHz maximum performance when receiving requests for data from seven SFIs. The maximum sustained event building rate corresponds to a total data throughput from

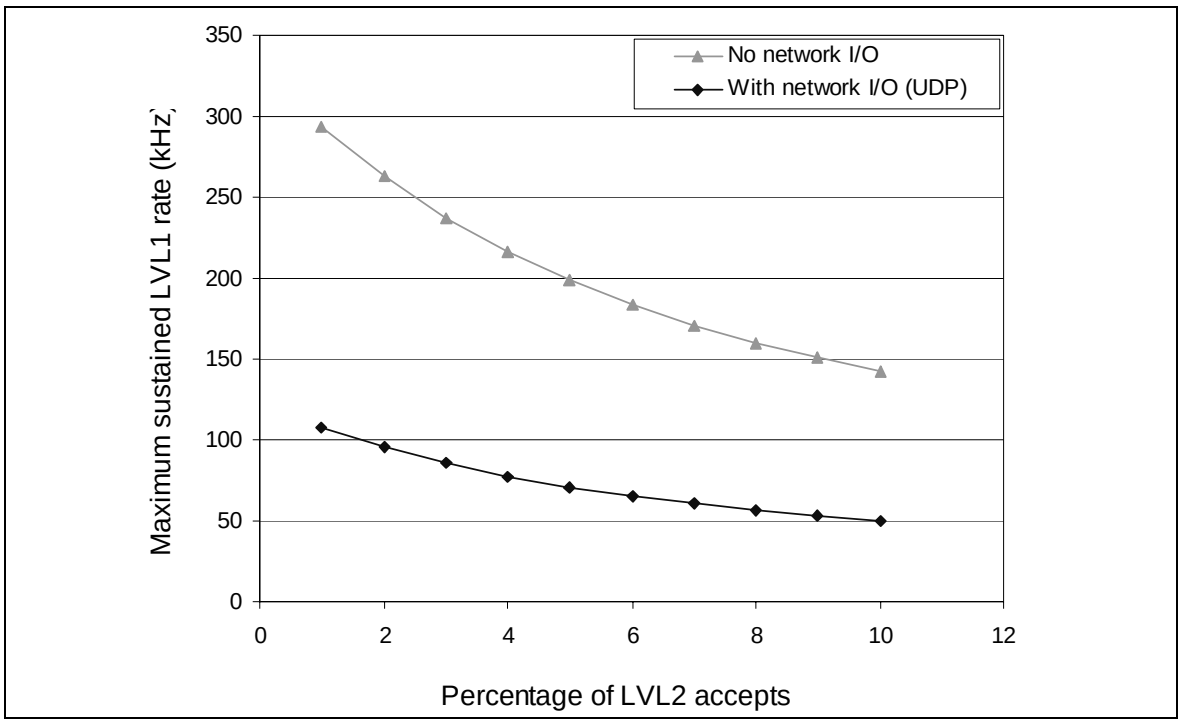

Figure 8-9 The bus-based ROS sustained LVL1 rate with and without network I/O

the ROSs to the SFIs, of 450 MByte/s. In other words the bus-based ROS is able to drive its output link to 110 MByte/s, i.e. almost the nominal link speed limit.

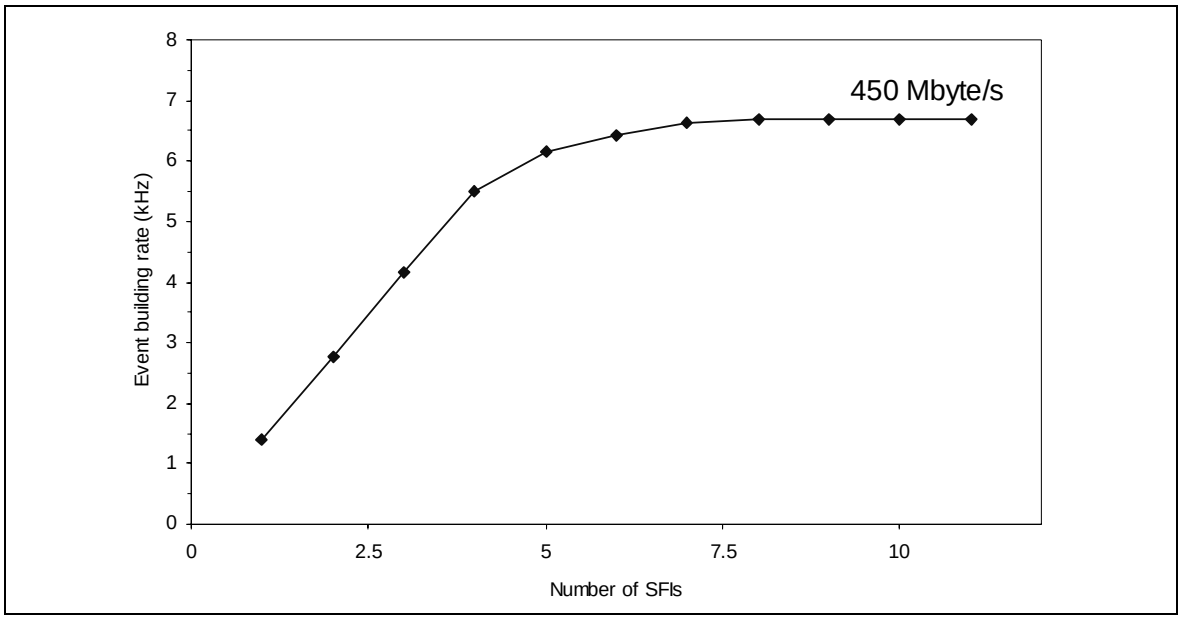

Figure 8-10 The event building rate sustained by four bus-based ROSs each emulating twelve ROLs, i.e a total of 48 ROLs

In summary, it has been shown that the bus-based ROS equipped with twelve ROLs on three RoBIn emulators meets the worst-case performance requirements (hotspot ROS) and exceeds the requirements by 20% in the normal case (standard ROS), with current off-the-shelf single-processor Linux PCs. The performance of the bus-based ROS, with real or emulated network

traffic, is far from being limited by the PCI bus bandwidth and performance gains may be achieved in the near future, due to improvement of PC Hardware, e.g. the replacing of today's 2.4 GHz CPUs with the 4 GHz CPUs that are expected to be commonly available by the end of next year.

In addition significant improvements in performance are expected from on-going software optimization studies: the use of new Linux kernels and/or thread libraries; possible software architecture changes to exploit multi-processor machines; and the optimization of the high-level layers of the message passing software.

8.1.3.4 pROS

The Pseudo-ROS receives the detailed result records of the L2PUs for accepted events and participates in the event building process, such that the LVL2 detailed result appears within the full event record, see Section 9.2.5. As the name indicates, it provides ROS functionality specifically for the L2PU. As its input rate is given by the rate of LVL2 accepted events O(3 kHz) and the estimated size of the LVL2 detailed result is O(1 kbyte), it is purely a software process, on a PC, receiving event fragments via an Ethernet connection. That is to say that unlike the ROS the I/O demands do not warrant the deployment of a RoBIn. From the point of view of the SFI there is no difference between the pROS and the ROS and it is estimated that a single pseudo-ROS is sufficient for the final system. The requirements and design of the pROS are described in Refs. [8-22] and [8-23].

8.2 Boundary and interface to the LVL1 trigger

The LVL2 trigger is seeded by the RoIs identified by the LVL1 trigger. The information from LVL1 includes the selected trigger type and the details of where in η and φ the trigger objects (e/γ, μ, etc.) that caused the event to be accepted originated. The interface between the LVL1 and LVL2 triggers has been specified [8-24] and is implemented by the RoIB component of the DataFlow.

Figure 8-11 shows the RoIB and its connections to the LVL1 system. The RoIs are input to the RoIB on eight separate links at event rates of up to 100 kHz. The main function of the RoIB is to collect the individual RoIs for each LVL1 accept and produce a single data structure which it then relays to the L2SV. To meet the rate requirements, the L2SV is implemented by a small, O(10), processor farm of PCs each of which runs a supervisor process. The RoIB ensures the flow of information between the LVL1 and DataFlow and is also an integral part of the LVL2 trigger. In this section the design and performance of the prototype RoIB are presented. Section 9.2.2 presents those aspects relevant to the correct functioning of the LVL2 trigger, e.g. load balancing.

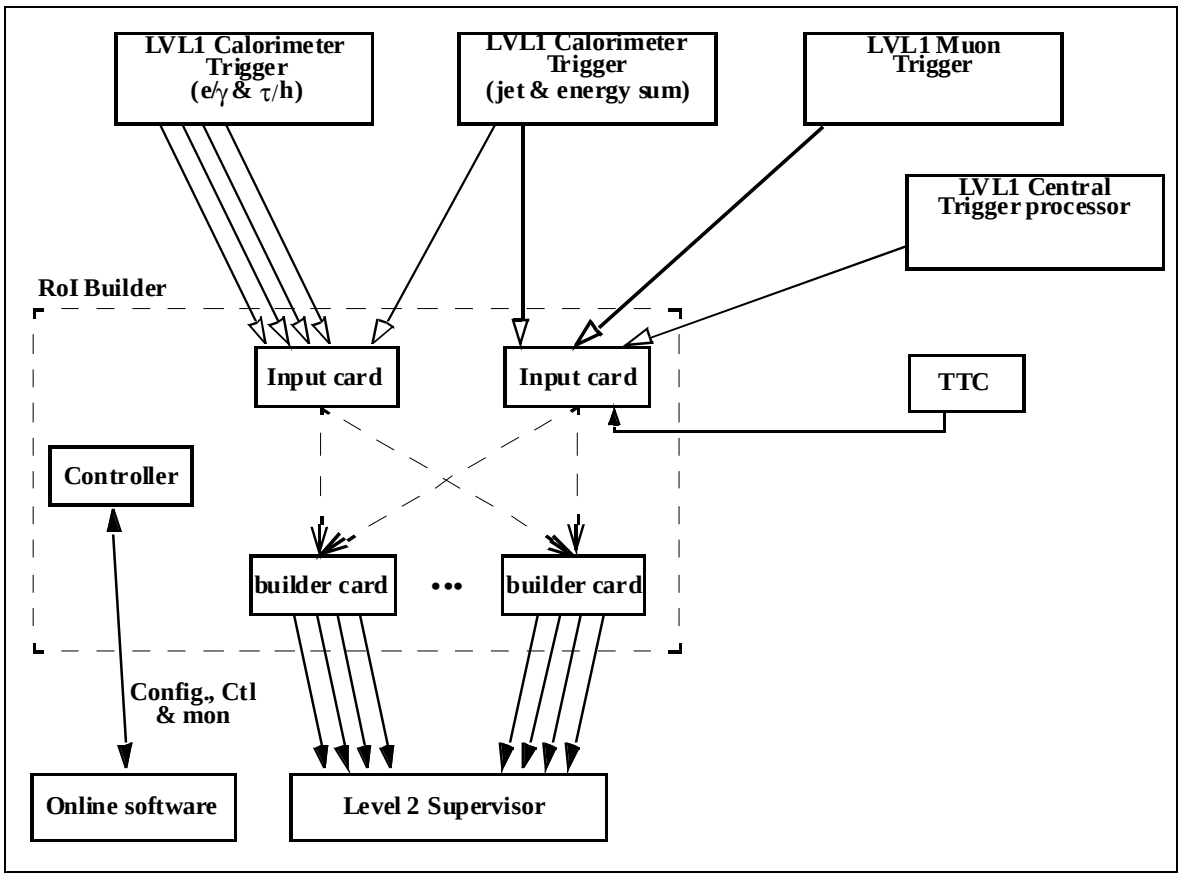

Figure 8-11 RoIB and its connections to the LVL1 system

8.2.1 Region-of-interest builder

Referring to Figure 8-11, each major element of the LVL1 trigger system provides RoI fragments to the RoIB via a point-to-point link. The requirements of this link are equal to those of the ROL and it is therefore envisaged to use S-LINK for its implementation, see Section 8.1.2. In addition to the eight ROLs, a TTC input is also foreseen to allow for, particularly during the debugging and commissioning phases, consistency checks of the extended L1ID contained in the received RoI fragments.

The maximum size of the RoI fragment received on each link, per LVL1 accept, is specified to be 256 byte [8-24]. Additionally, the skew between the arrival time of RoI fragments is specified to be less than one millisecond in the absence of flow control. Per LVL1 accept the RoIB assembles the RoI fragments into a RoI record and sends them to a supervisor processor. The selection of supervisor processor is governed by a load-balancing algorithm.

8.2.1.1 Implementation and performance

The baseline implementation of the RoIB is a VMEbus system which includes a SBC[1] for interfacing with the Online system for the purposes of configuration, control and monitoring. It is composed of two stages: input and assembly. The input stage consists of input cards that receive and buffer the RoI fragments. Each input card is equipped to receive data from up to six ROLs, thus two cards are required in the final system. These cards subsequently send the RoI fragments to ‘builder cards’ in the assembly stage where the RoI fragments are assembled into RoI

1. The same SBC as used in the ROD crates.

records. Per event, the RoI fragments are sent to all 'builder cards', the assignment of each event to a specific builder card will be based on a token-passing mechanism between the builder cards deployed. Each builder card can service up to four supervisor processes. The number of builder cards within the system is not limited and is dictated by the rate that a supervisor process can sustain. The implementation of the baseline architecture foresees ten supervisor processes thus three builder cards.

A prototype of the RoIB has been built and tested during the course of 1999. It was based on a pair of 12U VMEbus cards, an input card capable of handling six S-LINK inputs, and a pair of builder cards able to output to a pair of supervisor processes. This implementation utilized 76 Altera 10K40 FPGAs and 8 10K50 FPGAs. The system and early performance measurements are documented in Ref. [8-25].

Exploitation has shown that combining RoI fragments from several sources using an FPGA-based device is feasible and that a L2SV consisting of four 300 MHz Pentium II PCs was sufficient to receive the RoIB output rate of 100 kHz. Subsequent tests with prototypes of the muon Central Trigger Processor interface and the Central Trigger Processor ROD modules of the LVL1 system have made a start on debugging the component interfaces and have further demonstrated that data input to RoIB could be achieved at the required rates [8-26].

8.3 Control and flow of event data to high-level triggers

8.3.1 Message passing

8.3.1.1 Control and event data messages

The flow of event data between components of the DataFlow system is achieved by the exchange of control messages and subsequent event data messages. This is described in detail in Refs. [8-27] and [8-28], here only its major features are summarized. Figure 8-12 is a sequence diagram describing the handling of an event by the DataFlow components.

The sequence commences with the reception by a supervisor process of the LVL1 Result, which contains the RoI information, from the RoIB. Using a load balancing algorithm the supervisor assigns the event to a L2PU for analysis.

The L2PU receives the LVL1 Result from the L2SV and uses the contained RoI information to seed its processing, see Section 9.2.4. The sequential processing performed by the L2PU results, on average, in 1.6 RoI data requests per event. The selected ROS units service the request for data by responding to the requesting L2PU with a ROS event fragment message. On reaching a decision as to whether the event should be accepted or rejected, the L2PU sends the LVL2 Decision message to the supervisor process. In the case that the event is accepted for further processing by the EF, the L2PU also sends the detailed result of its analysis to the pROS.

The supervisor process receives the LVL2 Decision and forwards a group of them to the DFM. If no LVL2 decision is received within a pre-defined timeout, the supervisor process deems the event to have been accepted by the L2PU and sends a LVL2 Decision to the DFM.

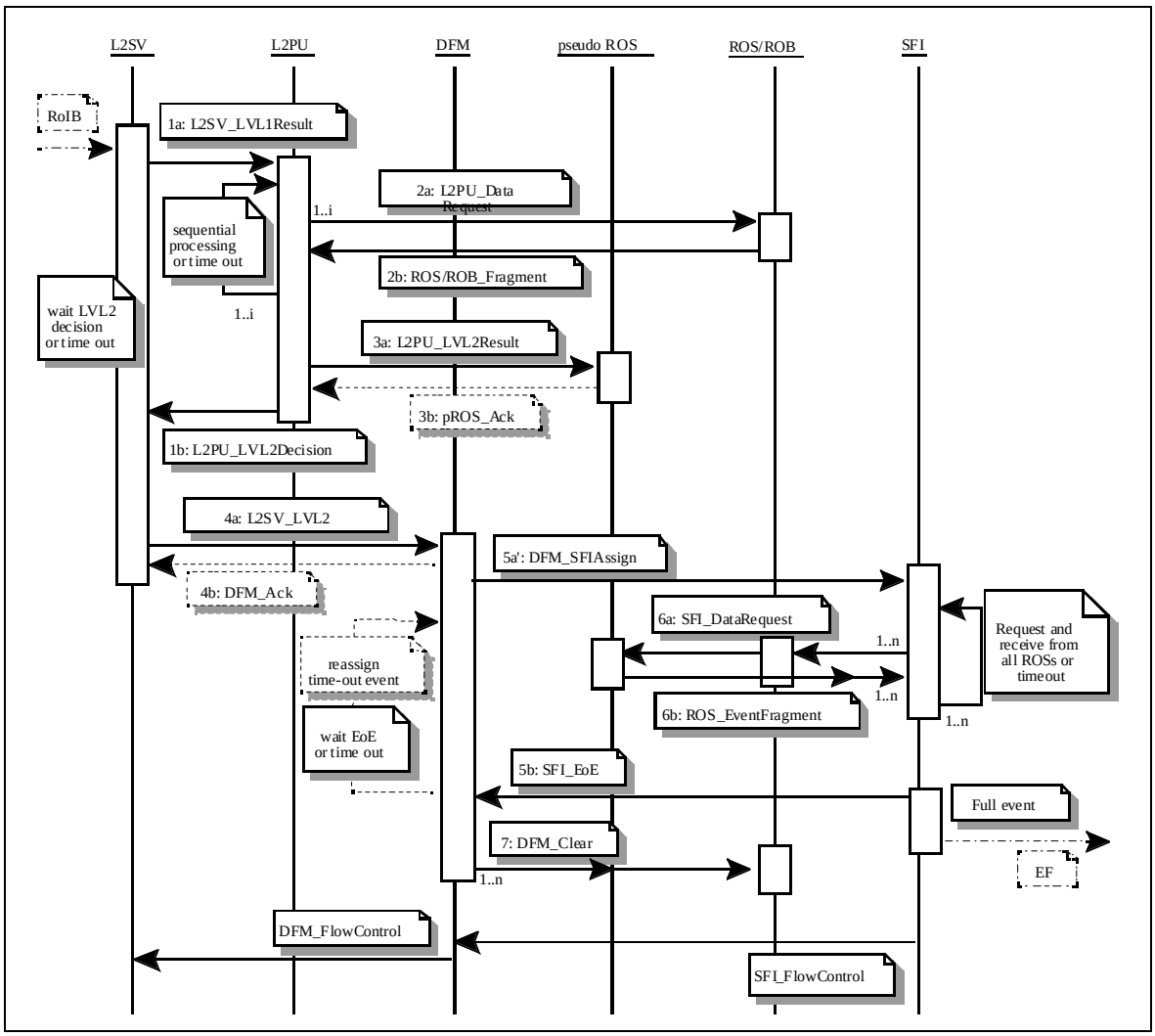

Figure 8-12 Sequence diagram showing the interactions between DataFlow components

On reception of a group of LVL2 Decisions the DFM acts on each decision and in the case of an accepted event, based on a load balancing algorithm, assigns an SFI to perform the building of the event. A group of Clear messages is multicast by the DFM to all ROSs for events, i.e. those rejected by the LVL2 and those successfully built. This message contains the identifiers of events treated by the system.

The SFI builds the event by sequentially requesting event date from each ROS or RoBIn. The choice of sequentially building the event into the SFI using a request-responce, implicitly controls, i.e. Traffic shaping, the flow of Data messages towards the SFI. Therefore removing the dependency on the DataFlow network technology to provide peer-to-peer flow control. The full events built in the SFI are subsequently sent to an EF subfarm for further processing.

Table 8-1 Summarizes the messages, in terms of rate and bandwidth, exchanged between the DataFlow component.

The message rates and bandwidth can be handled by a wide range of link technologies. The choice is dictated by price, long-term availability, support, inter-operability, and suitability for DataFlow. Ethernet in its varieties of 100 Mbit/s and 1000 Mbit/s is the prime candidate and has been evaluated for its suitability for exchange of control and event data messages.

Table 8-1 Average message rates and bandwidth per DataFlow component

Communicating components	Message type	Sender		Receiver		Comment
		Rate (kHz)	Bandwidth (Mbyte/s)	Rate (kHz)	Bandwidth (Mbyte/s)	
LVL1 to RoIB[a]	Data	12.5	26	12.5	26	RoI information
RoIB[b] to L2SV	Data	33	32	7.5	9.8	RoI Record
L2SV to L2PU	Data	7.5	9.8	0.5	0.7	RoI Record
L2PU to ROS	Control [c]	5	0.5	14	1.4	Data requests
	Control [d]	10	1	5	0.5	
ROS to a L2PU	Data [c]	14	28	5	10	Event data
	Data [d]	5	5	10	10	
L2PU to L2SV	Control	0.5	0.05	7.5	0.75	LVL2 decision
L2PU to pROS	Data	3	0.003	0.3	0.3	L2PU processing output
L2SV to DFM	Control	75	0.04	0.75	0.4	LVL2 decision
DFM to a SFI	Control	3	0.3	0.04	0.004	Assignment to an SFI
SFI to ROS	Control [c]	6	0.6	3	0.3	Data requests
	Control [d]	66	6.6	3	0.3	
ROS to a SFI	Data [c]	3	36	6	66	Event data
	Data [d]	3	3	66	66	
SFI to DFM	Control	0.04	0.004	3	0.3	Indicates event built
DFM to ROSs	Control	0.25	0.4	0.25	0.4	Clear events from buffers
SFI to EF	Data	0.04	66	0.04	66	Event data

a. RoIB input port
b. RoIB output port
c. Bus-based ROS
d. Switch-based ROS

8.3.1.2 Ethernet

Extensive studies have been performed on many Ethernet features leading to its adoption as the baseline networking technology in the DataFlow. The features studied have included the characteristics of switches with respect to throughput, packet loss, latency, trunking and Media Access Control (MAC) address table size; Virtual Local Area Network (VLAN) implementation; Flow Control at different levels, i.e. across switch and between Ethernet nodes; Quality of Service (QoS); broadcast and multicast handling; inter-operability of switches from various vendors.

The results of studies of these features are reported in Ref. [8-29] and additional references to further studies performed on Ethernet can be found in Ref. [8-30]. Features of primary importance to the baseline architecture have emerged to be switch throughput, latency, packet loss

and, in the case of the switch based ROS, VLANs. The results of studies of these features are summarized in this chapter.

8.3.1.2.1 Basic switch performance

Switches must meet the throughput requirements of the architecture with a minimum latency and loss of Ethernet frames. The latter results in a degradation of the system's performance as it implies the implementation of timeouts and retries at the application level. Within a switch the loss of frames may occur as a result of buffer overflow, therefore the larger the switches buffers the smaller the probability of frame loss.

The frame loss and latency of a number of switches have been studied for different frame size, without flow control, loads (from 10% to 100% of the line speed), with Constant Bit Rate (CBR) or with Poisson inter-packet gap, using unicast, multicast and broadcast traffic.

Figure 8-13 shows the results of a test performed on two different switches, using raw Ethernet and maximum size frames. These measurements used 30 GE ports, each one sending unicast traffic to all the others with a negative exponential inter-packet gap. Switch 1 became saturated when the offered load exceeded 66% of the line speed. It can further be seen that a slight increase in latency is followed by packet loss, and a significant growth in latency occurs once the switches buffers become full. The second switch (Switch 2 in the figure) in this test performed better, almost achieving line speed.

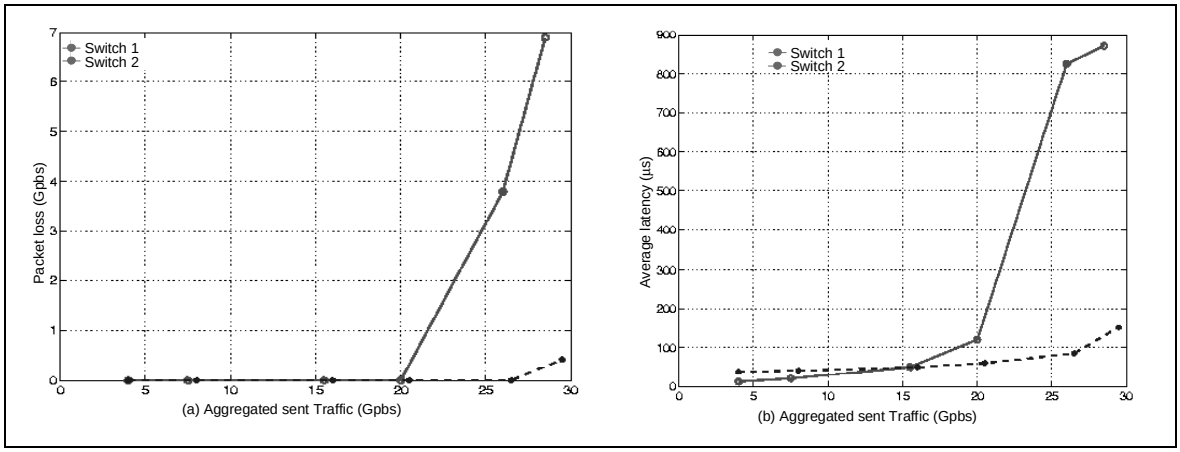

Figure 8-13 Switch measurements for unicast traffic, Poisson inter-packet gap, 1518 byte frames: (a) Frame loss; (b) Average latency

Similar measurements have also been performed using multicast and broadcast traffic. The results show that switch performance is vendor specific and in some cases the maximum throughput is surprisingly low, i.e. less than 10% of the line speed. The avoidance of vendor dependency is one of the reasons for the choice of unicast traffic (a request–response scenario) in the baseline architecture, see Section 8.3.1.1.

In summary, any switch that is to be deployed must operate below a saturation point to avoid latency increase and the subsequent frame loss. This saturation point must be determined by measurement.

The problem of frame loss can be alleviated by the use of Ethernet flow control. However, the propagation of flow control through a switch is manufacturer dependant and not covered by

the Ethernet flow control standard. The propagation of flow control assures a lossless frame transfer, but once congestion appears it may spread over large areas [8-29] leading to a degradation of the switch's overall throughput. In switch's that do not propagate flow control, frame loss will occur if the congestion cannot be absorbed by the switches internal buffers. However, in this case there is no spreading of congestion. Ethernet flow control helps prevent the overflow of buffers within a switch and a network interface card, but it does not completely solve the problem of frame loss as it can also occur when the kernel buffers used to receive frames from a network interface card also overflow. The prevention of frame loss at this level requires that flow control propagates back to, and be used by, the Linux kernel. This functionality is not currently foreseen to be implemented.

8.3.1.2.2 Virtual Local Area Network

The network topology of the proposed architecture using the switch-based ROS, see Section 5-5, contains network loops, see Figure 8-14, which are illegal in the use of Ethernet as they disturb the MAC address tables for unicast frames and result in the continuous sending of multicast and broadcast messages (broadcast storms). In general the Spanning Tree Protocol (STP) is deployed to disable redundant links in a LAN so as to maintain a loop-free topology. In the proposed architecture a loop-free topology will be achieved by using two VLANs, one of each associated to the LVL2 and EB traffic. The extended header of the Ethernet frame may include a VLAN tag immediately after the Ethernet addresses allowing many logical LANs, i.e. VLANs, to coexist on the same physical LAN.

The set-up shown in Figure 8-14 has been used to verify that VLANs eliminate illegal loops, and to ascertain whether the STP can be applied per VLAN. With the STP disabled, tests have shown that VLANs ensured a loop-free topology. With the STP enabled, one of the links in the loop was disabled indicating that the STP is not implemented per VLAN, at least on the switches used in the tests.

In summary, if the STP cannot be applied per VLAN it can be disabled and VLANs alone used to ensure a loop-free topology. Note there are indications that most forthcoming switches will implement a STP that can be applied per VLAN.

Not only do VLANs ensure a loop-free topology, but they also provide additional features: they restrict flooding; provide containment of multicast and broadcast traffic; and provide support for QoS by means of the priority field within the VLAN tag.

8.3.1.3 Design of the message passing component

The requirements of the Message passing layer are detailed in Ref. [8-31]. It is responsible for the transfer of all control and event data messages between the DataFlow components. It imposes no structure on the data that is to be exchanged and its implementation allows the transfer of data with a best-effort guarantee. No re-transmission or acknowledgement of data is done by this layer allowing the API to be implemented over a wide range of technologies without imposing an unnecessary overhead or the duplication of existing functionality. The API supports the sending of both unicast and multicast messages. The latter has to be emulated by the implementation if it is not available, e.g. for TCP.

The design of the Message Passing layer [8-32] defines classes that allow the sending and receiving of messages. The *Node*, *Group* and *Address* classes are used at configuration time to set-up all

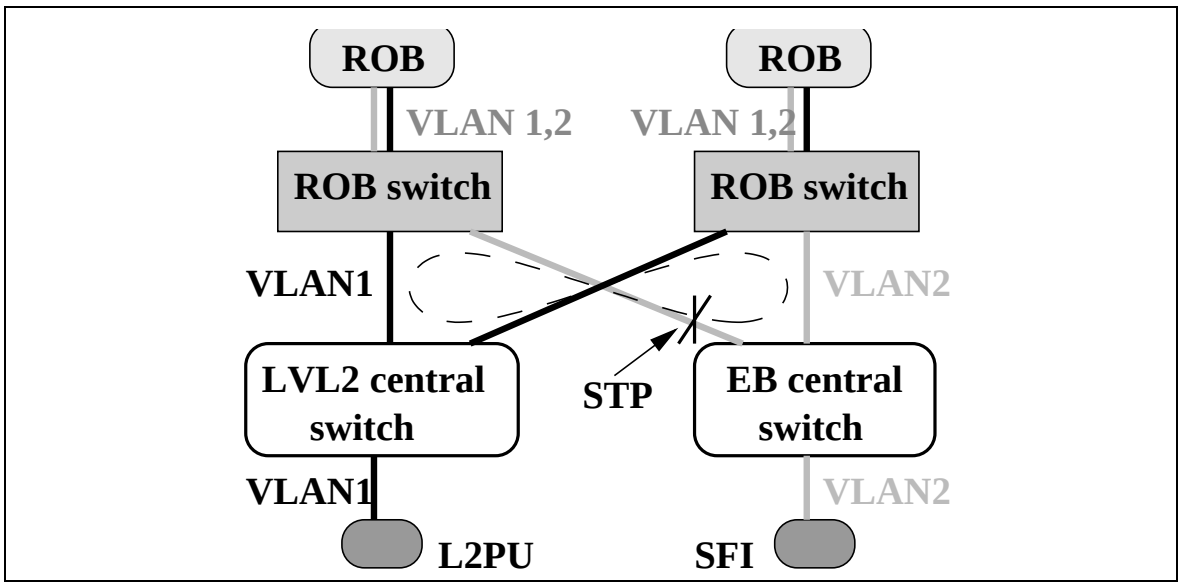

Figure 8-14 VLAN Ethernet loop set-up. The potential loop appears in dashed line

the necessary internal connections. The *Port* class is the central interface for sending data. All user data has to be in part of a *Buffer* object to enable it to be sent or received from a *Port*. The *Buffer* interface allows the addition of user-defined memory locations which are not under the control of the Message Passing layer in order to avoid copying. The Provider class is an internal interface from which different implementations have to inherit. Multiple Provider objects can be active at any given time. A Provider is basically the code to send and receive data over a given protocol / technology, e.g. TCP, UDP or raw Ethernet.

8.3.1.4 Performance of the message passing

The prototype Message Passing layer interface has been implemented over raw Ethernet frames, UDP / IP and TCP / IP. TCP / IP provides additional reliability compared to UDP and raw Ethernet. However, the apropriateness of TCP / IP for the traffic in the DataFlow is debatable [8-33]. Therefore the applications and message flow have been designed to ensure the correct system functioning when an unreliable technology is used, i.e UDP or raw Ethernet.

The raw Ethernet implementation adds message re-assembly on the receiver side to overcome the restriction of the maximum message size being a single Ethernet frame restriction.

Internally all implementations support scatter / gather transmission and reception of data. This allows the building of a logical message out of a message header and additional user data that does not need to be copied inside the application.

Extensive studies of the performance of the Message Passing layer have been performed [8-34]. Here the performance is summarized by drawing a comparison with the performance that can be obtained using standard Linux operating system primitives to stream messages between two PCs, equipped with 2 GHz CPUs, and connected by a point-to-point gigabit Ethernet link.

Figure 8-15 shows the CPU time to receive a message versus the received message size using the Message Passing layer with raw Ethernet and UDP / IP; and using standard Linux operating system primitives with raw Ethernet[1] and UDP / IP. The results obtained for applications using the standard Linux operating system primitives, due to their simplicity, give the upper limits on

the performance that can be achieved with today's versions of the Linux operating system and PCs.

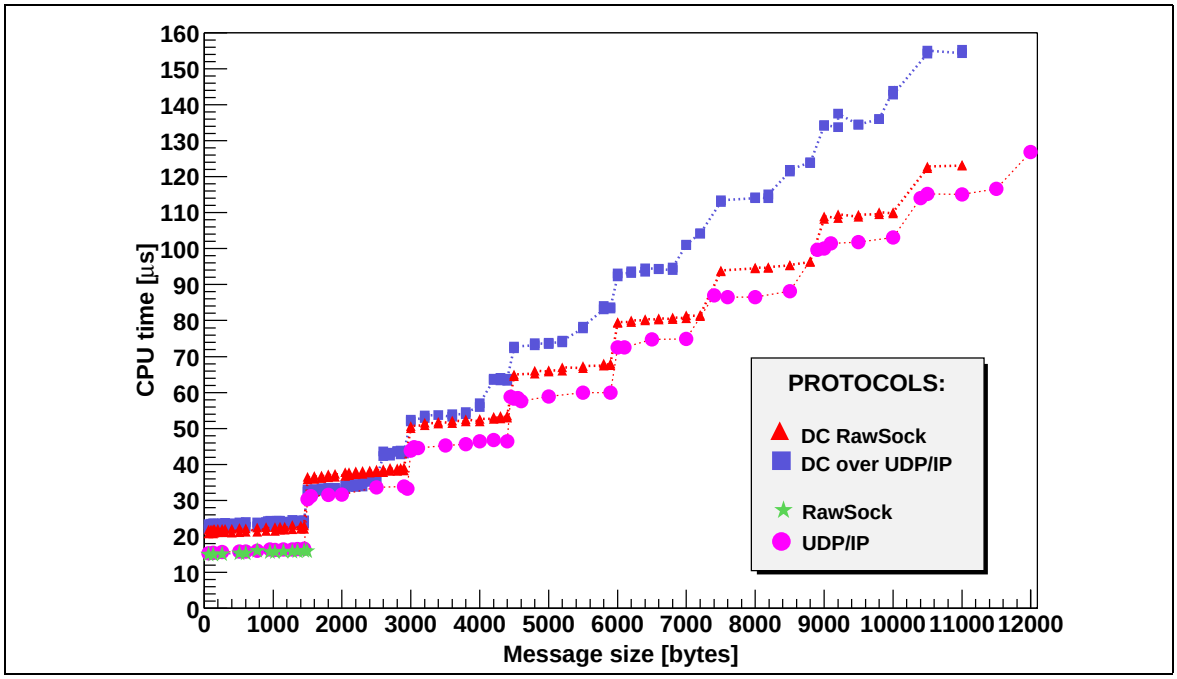

Figure 8-15 The CPU time required to receive a message versus the length of the message

The main feature is a step increase in the time to send a message of ~8 µs at the boundary of multiples of the Ethernet frame size. Within multiples of Ethernet frames the time to send a message varies by less than 1%. Upto message sizes of a single Ethernet frame there is no significant difference in performance between raw Ethernet and UDP/IP and the overhead in receiving a message with Linux operating system primitives is measured to be ~14 µs. The Message Passing layer using raw Ethernet, compared to the performance using Linux operating system primitives and UDP/IP, introduces an additional overhead of 8 µs to the CPU time required to receive a message and this additional time is independent of the message size. The CPU time required to receive a message using the Message Passing layer with UDP/IP increases, compared to that obtained with raw Ethernet. This divergence is due to the reassembly of messages performed in the IP layer.

The CPU time required by a process using the Message Passing layer with UDP/IP (running on a PC with a 2 GHz CPU and Linux kernel version 2.4.20) to receive a message has been categorized with respect to the operating system interrupt handling, the operating system protocol stack, and the Message Passing layer overhead. The results of this categorization are summarized in Table 8-2.

In summary, the CPU time required to receive a single Ethernet frame message is ~22 µs and for multi-Ethernet frame message the CPU time is ~14 µs per frame. Similar measurements performed for the sending of a message show [8-34] that the CPU time to send a single frame message is ~12 ms.

1. The measurements using raw Ethernet system primitives are limited to a maximum message size of 1460 byte, as no re-assembly of larger packets has been implemented.

Table 8-2 Summary of the Message Passing performance with UDP/IP on a 2 GHz PC

Parameter	Time (μs)
Operating system interrupt service	10
Operating system protocol stack	4
Message Passing overhead	8

Over the past few years the CPU time to send and receive messages has decreased substantially. This is largely due to the improvements made in the Linux operating system, which are continuing to be made, e.g. interrupt coalescence, and should lead to proportional improvements in the DataFlow Message Passing layer. The latter adds to the operating system overheads an additional overhead which decrease as CPU speed increases. It is also not excluded that this additional overhead will further decrease as a result of improvements to the design of the Message Passing layer.

8.3.2 Data collection

8.3.2.1 General overview

In the final phase of prototyping leading to the baseline architecture, the movement of event data within the DataFlow was designed and implemented as a single sub-system called the DataCollection. The requirements on this sub-system and a detailed description are given in Refs. [8-35] and [8-36]. It provides for the movement of the LVL1 RoIs to the L2PU (via the L2SV) and the LVL2 result (decision and detailed result) to the EventFilter as well as the EventBuilding and feeding the complete events to the EventFilter. The DataCollection applications (L2SV, L2PU framework, DFM, pROS, SFI and SFO) are based on the design and implementation of a framework that implemented a suite of common services:

- OS Abstraction Layer
- Access to the configuration Database
- Error Reporting
- System Monitoring
- Run Control
- Message Passing.

Services are built from packages following a modular approach. Many of these packages consist only of interfaces whose implementation is provided by other packages that can be changed at configuration or run-time. This clear separation between interfaces and implementations exists down to the lowest levels, e.g. the thread interface and access to system clocks and timers. Examples of the common interfaces are those required between the applications and the Online software for the reporting of errors, access to the configuration database, and for control and operational monitoring. The Message Passing layer, described in Section 8.3.1.3, provides for a common interface for the exchange of control and data messages.

The following sections present the performance of the RoI data collection and Event Building functionality achieved using prototype implementations of the L2SV, L2PUs, pROS, DFM, SFI and SFO based on the DataCollection implementation.

In these performance tests the processors used to run the L2PU as well as the L2SV and ROS emulators were all dual Xeon CPUs of 2.2 GHz interconnected by Gigabit Ethernet.

8.3.2.2 RoI data collection

8.3.2.2.1 Design

The interactions between the RoIB, L2SVs, L2PUs, ROSs and pROS provides the functionality of RoI data collection and were described in Section 8.3.1.1. The collection of RoI data is part of the overall LVL2 functionality, whose detailed description is given in Chapter 9.

8.3.2.2.2 Performance

Each L2SV receives complete RoI information from the RoIB and provides it to one of its associated L2PUs. Figure 8-16 shows the LVL1 rate sustained by a L2SV as a function of the number of L2PUs to which it is associated. In this measurement the PC running the L2SV is also connected to a gigabit Ethernet switch on which is also connected a number of PCs, each of which were executing the L2PU. The L2SV was not connected to a RoIB, instead it emulated the receiving of events from the RoIB. As can be seen from the figure, the L2SV can sustain a LVL1 accept rate of 32 kHz when distributing RoI information to a single L2PU and the dependency on the number of L2PUs is 1%. Thus, based on today's prototype implementation and PCs ten L2SVs are more than adequate to support a LVL1 accept rate of 100 kHz.

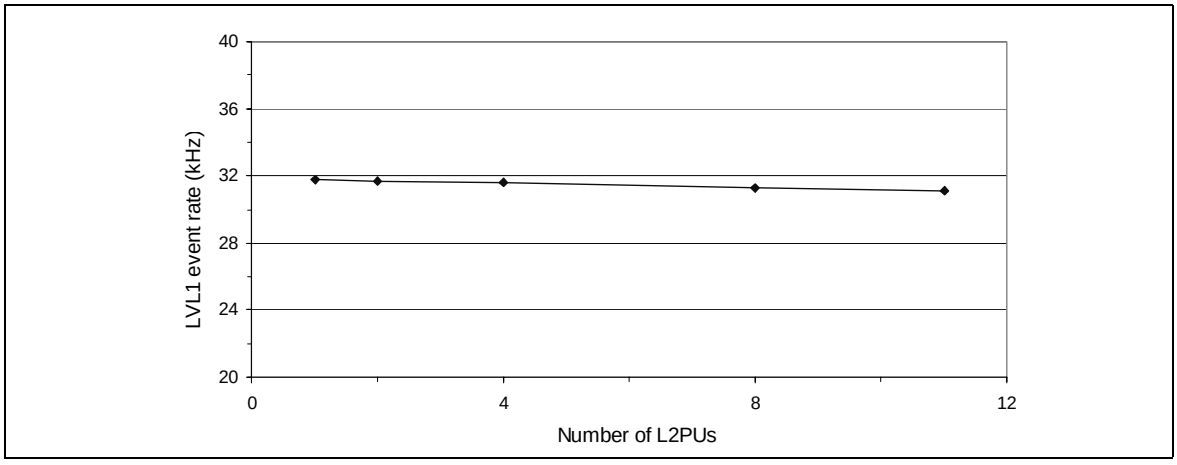

Figure 8-16 The L2SV sustained LVL1 accept rate versus the number of L2PUs

In the baseline architecture the rate at which a L2PU can collect RoI data will depend on the size of the RoI, the number of ROSs over which the RoI is distributed, and the number of worker threads, see Section 9.2.4, that collect RoI data in parallel[1].

1. Each worker thread processes a different event.

Figure 8-17 shows the performance of a single L2PU as the inverse of the LVL1 accept rate sustained by a single L2PU versus the number of worker threads. In this measurement the L2PU collects only the RoI data, i.e. it does not execute any LVL2 algorithm, and the RoI has a fixed size of 16 kbyte. The different curves represent the RoI being collected from 2, 4, 8, or 16 ROSs. For example the upper curve represents the results of collecting a 16 kbyte RoI from twenty-two ROSs, each of which contributes 0.8 kbyte to the RoI.

The results indicate that the optimum number of worker threads, in this set-up, is approximately three and is independent of the number of ROSs from which the RoI is being collected. In addition, the results show that for the conditions of this measurement and for three worker threads, the collection of RoI data contributes less than 10% to the average event processing time of 10 ms.

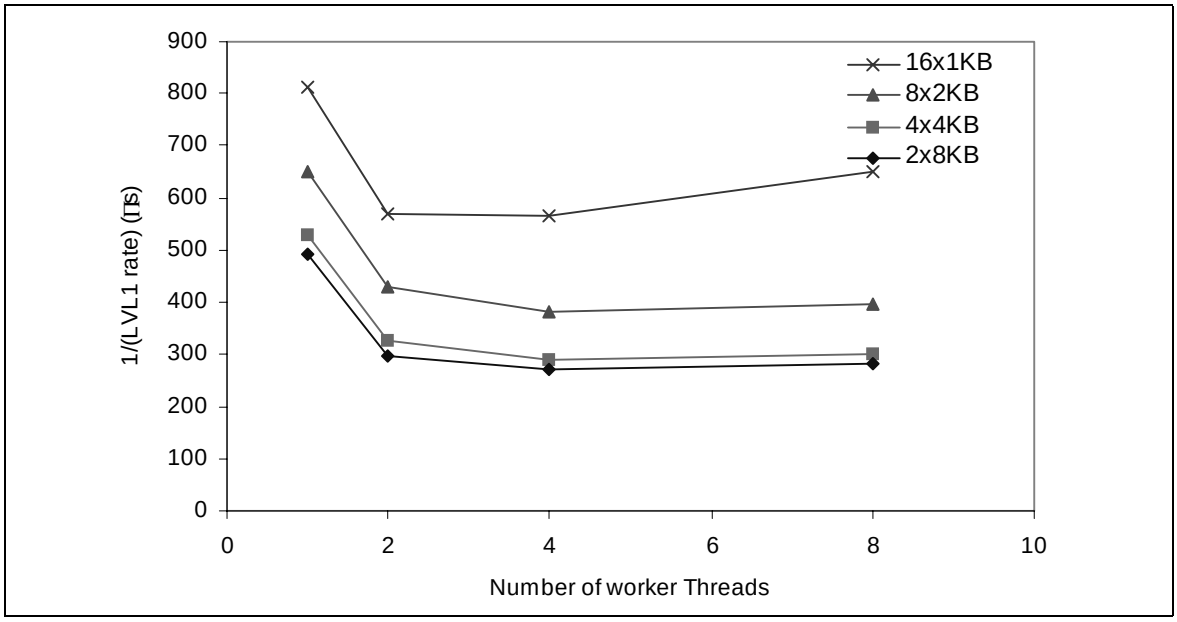

Figure 8-17 The inverse of the LVL1 accept rate sustained by an L2PU collecting a 16 kbyte RoI from 2, 4, 8, and 16 and ROSs versus the number of threads concurrently collecting RoI data

The contribution of RoI data collection to the L2PU event processing time is also shown in Figure 8-18 as a function of the RoI size and the number of ROSs over which the RoI is distributed.

The range of the measurements presented in Figure 8-18 corresponds to the currently expected RoI sizes and their distribution over RoBs, e.g. an e/γ RoI in the Liquid Argon detector is expected to be distributed over 13 to 16 ROBs and have a size of approximately 16 kbyte. It can be observed from the measurement results that the time taken to collect RoI data contributes, in the worst case, less than 10% to the average event processing time budget of 10 ms.

The measurements so far presented have shown the rate at which a single L2PU, without algorithm execution, can collect RoI data. The achieved performance depends on the RoI size and the number of ROBs from which the RoI must be collected. For the system, however, it is also neesary to demonstrate that when many L2PUs are requesting data at the same time from the same set of ROSs the performance of te RoI collection does not degrade unacceptably. For this test L2PU without algorithms were again used, although it should be noted that the request rate for data per L2PU is much higher that it would be if algorithms were included. Thus each L2PU

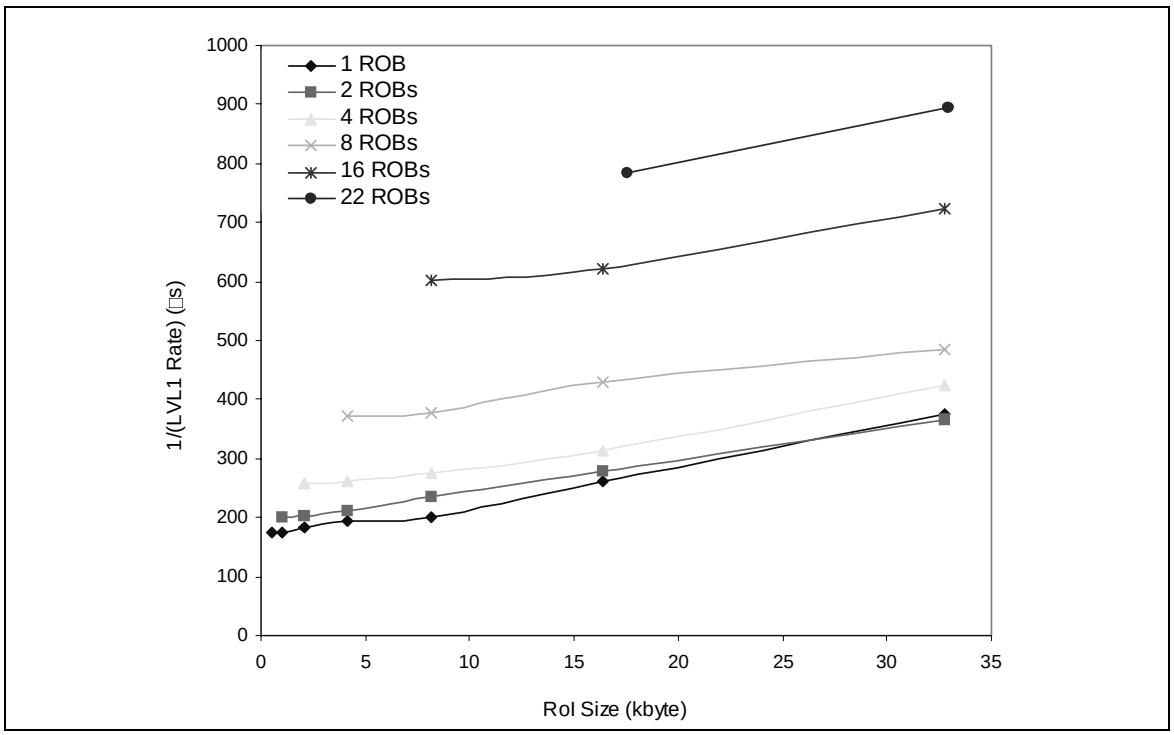

Figure 8-18 Summary of the performance of the RoI data collection for various combinations of RoI sizes and slices

generates more than ten times the normal request rate. Similarly the requests are sent to a small number of ROSs, so that again it is important to ensure that the total request rate did not exceed the capability of the ROSs. The tests were run in a testbed consisting of three L2SVs, four ROSs (as described in Section 8.1.3.3) and upto eleven L2PUs[1]. All the nodes in the test bed were PCs equipped with dual Xeon processors, clocked at 2.2 GHz, inter-connected via a gigabit Ethernet switch. For this test each ROS was configured to contain 12 ROLs, to give a total of 48 ROLs across the four ROSs. For each request the L2PU chose one of the 48 ROBs at random, and in the case of 6 ROBs/RoI and 24 ROBs/RoI requested the following 5 and 23 ROBs respectively, wrapping round across the ROSs as necessary. Figure 8-19 present the results of these measurements. It shows the rate of events handled by each L2PU and the rate of requests to each ROS as a function of the number of L2PUs used in the test bed for an RoI distributed over 1, 6 or 24 ROBs (with 1.5 kbyte per ROB).

The plot shows that the rate per ROB does fall as the number of L2PUs is increased, by ~20% for 1 ROB/RoI and ~40% for 24 ROBs/RoI. However, with 11 L2PUs in the test the request rate per ROS is 17 kHz for 1 ROB/RoI and 8 kHz for 24 ROBs/RoI, both very heavy demands on a ROS. In addition the total RoI request rate in the small test set-up is already 70 kHz and 11 kHz for the two RoI sizes respectively. Given these extreme conditions the fall from scaling can be seen to be relatively modest, and it seems reasonable to assume that with more typical conditions a full-sized system will show acceptable scaling. This interpretation is reinforced by the fact that running similar tests with the ROSs replaced by FPGA ROS emulators shows scaling within better than 10% over the same range of RoI data traffic.

1. Each L2PU having two worker threads.

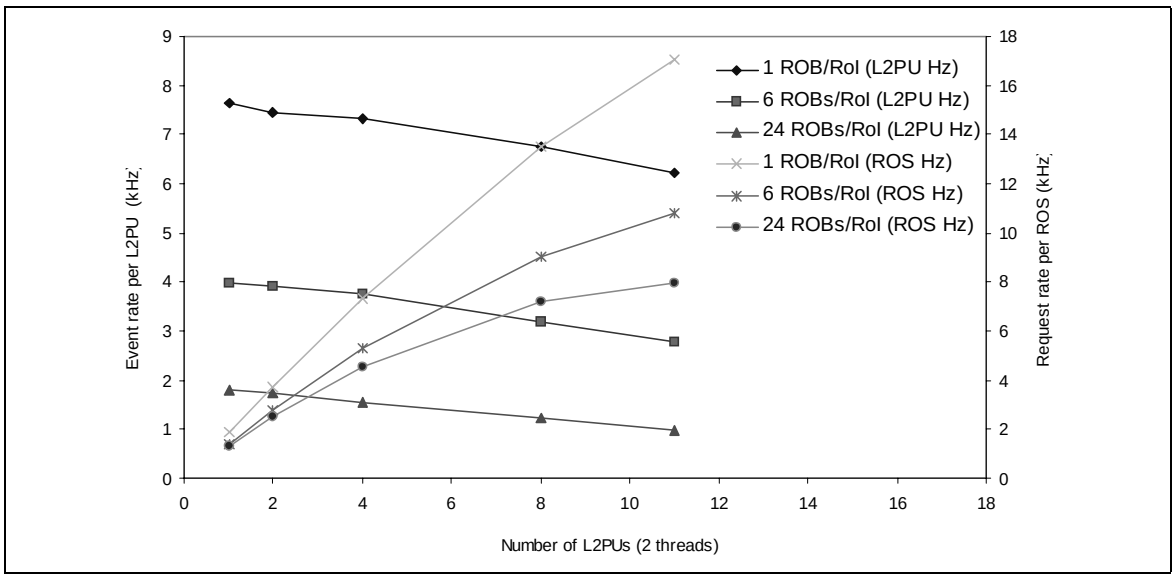

Figure 8-19 the sustained LVL1 event rate and data rate versus the number of L2PUs

8.3.2.3 Event Building

8.3.2.3.1 Design

The interaction between the ROSs, the DFM and the SFIs which implements the event building functionality is shown in Figure 8-12 and explained in Ref. [8-27]. Two scenarios concerning the interaction between DataFlow components in Event Building are being studied:

- *Push* scenario: In this scenario, the DFM assigns an SFI and informs all ROSs, via a multicast mechanism, to send their associated event fragments to that SFI. The SFI acts as an open receiver and builds the complete event out of the individual fragments received.
- *Pull* scenario: In this scenario, the DFM assigns an event to an SFI and SFI then requests from each ROS its event fragment via a unicast messages. The SFI receives from each ROS individually an event fragment and builds the complete event.

There is no difference in the amount of messages being handled on the level of the DFM, the ROSs, or the network, however, the amount of messages to be handled by an individual SFI is doubled in case of the pull scenario, see Table 8-1.

Although a doubling of the message rate at the level of the SFIs may seem problematic, the pull scenario offers the advantage of controlling the flow of traffic it is receiving. In this mode an SFI at any given moment in time never requests more event fragments than it can handle. Thus reducing the probability of congestion within the network and avoiding the loss of fragments in the switch buffers, see Section 8.3.1.2.1.

In the case of the push scenario the amount of traffic entering the switching network is controlled by applying IP QoS. The latter has been implemented in the standard Linux kernel at the IP level removing the necessity of implementing traffic shaping at the level of the ROS application. IP QoS manages the flow of data by employing packet classification and packet scheduling techniques. Packet classification, e.g. Class Base Queuing (CBQ), is used to classify incoming packets into groups, while packet scheduler arranges the scheduling for outgoing packets according to a queuing method and the buffer management selected, e.g. Token Bucket Filter

(TBF). Note that IP QoS as implemented by the Linux kernel is only performed in the message output queues, incoming packets continue to be accepted on a best-effort basis. It is also important to realize that packets are scheduled at best at the rate of the Linux kernel scheduler, which is a configurable parameter. As Event building is to be performed at a rate of ~ 3 kHz, packet scheduling should occur at least at the same rate for the traffic shaping to be effective.

Detailed studies have been made on the use of IP QoS to avoid congestion in the network. The results of these studies are summarized in Section 8.3.2.3.3 and further details can be found in Ref. [8-37].

8.3.2.3.2 Performance using the pull scenario

The building of events is performed by the DFM, SFI and ROSs and is the collecting of event fragments of an event located in up to 1600 different buffers. This has to be performed at a rate of ~ 3 kHz. Detailed studies of the event building have been performed using prototype software, PCs, Ethernet switches and traffic generators, see Refs. [8-38], [8-39], only the principal results are presented here.

The nominal event building rate in the proposed baseline architecture is ~3 kHz and commences with the arrival of LVL2 decisions at the DFM. In the pull scenario, the DFM assigns the events to an SFI, receives notification when an event is built, and sends clears to the ROSs. The performance of the DFM, the sustained event building rate verses the number of SFIs, is shown in Figure 8-20 when the Message Passing layer uses UDP/IP and Raw Ethernet. It can be seen that the prototype implementation of the DFM can sustain an event building rate of at least ~ 20 kHz, almost an order of magnitude greater than the required performance, and as expected a better performance is achieved when the raw Ethernet protocol is used. The different curves in the figure also show the dependence of the DFM performance on the size of the LVL2 decision group. A better performance is achieved the more LVL2 decisions are grouped into a single message. However, the sustained event building rate has a stronger dependency on the number of SFIs in the system. This dependency is attributed to the additional load on the DFM when handling more SFIs. However, in the case of a LVL2 decision group of 100, the receiving of additonal messages determines the behaviour.

The event building performance of the DataFlow is shown in Figure 8-21. This figure shows the sustained event rate as a function of the number of SFIs in the system, and for one to four ROLs per ROS[1]. The total size of the event built is 2.2 Mbyte and Ethernet flow control was not used. It can be seen that the sustained event building rate increases linearly with respect to the number of SFIs in the system. In the case of four ROLs per ROS, every additional SFI increases the overall system performance by ~35 Hz while for a single ROL per ROS the system performance increase by ~30 Hz.

It is worth noting that these measurements were performed without Ethernet flow control and in the case of four ROLS per ROS the sustained event building rate reaches a limitation due to the small buffer sizes of the gigabit Ethernet switch that was used for these measurements and Ethernet flow control not being used, i.e. the switch buffers start to overflow. Similar measurements performed with Ethernet Flow control applied restore the linear behaviour for four ROLs per ROS. In the case of a single ROL per ROS the perfomance is limited by the ROS emulators used in the measurement. Finally it should be noted that these measurements have been per-

1. In this measurement the ROS was emulated using custom hardware

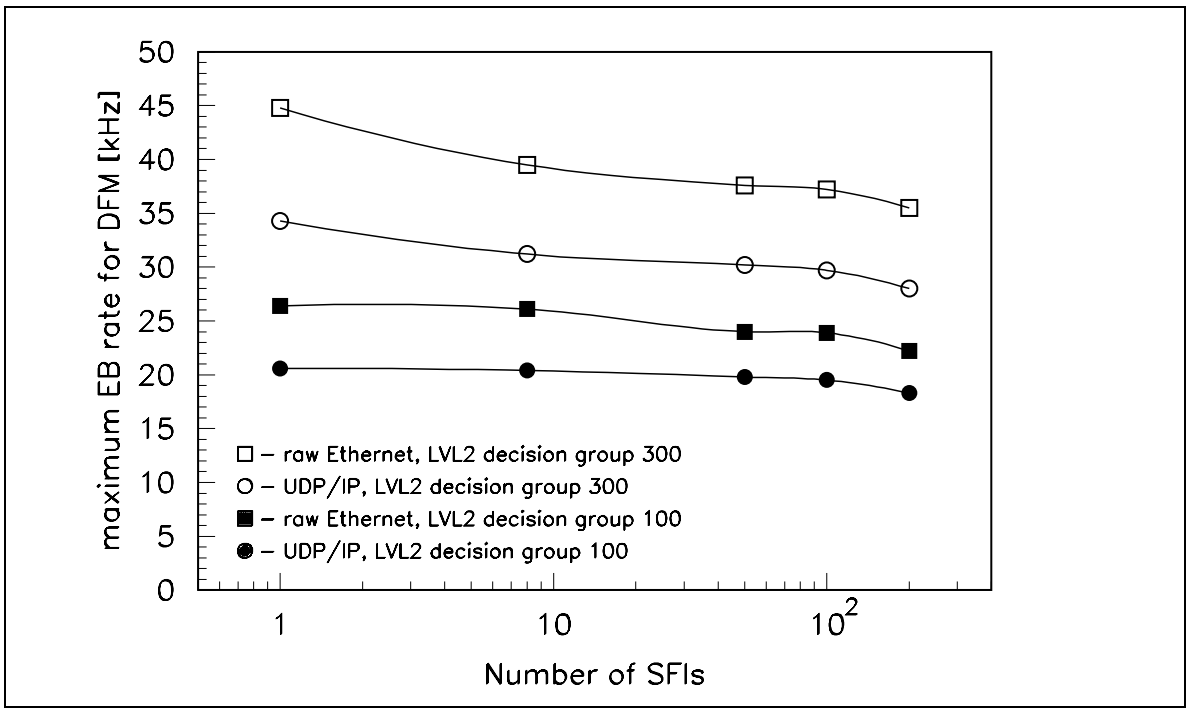

Figure 8-20 The DFM event building rate versus the number of SFIs. Each SFI concurrently builds two events

formed without the SFIs forwarding the built events to the EF. Other measurements, [8-39], have shown that when an SFI does forward the built events to the EF, the sustained event building rate decreases by ~ 30%. This decrease in performance is expected to be minimal on the timescales of ATLAS due to the expected increase CPU power.

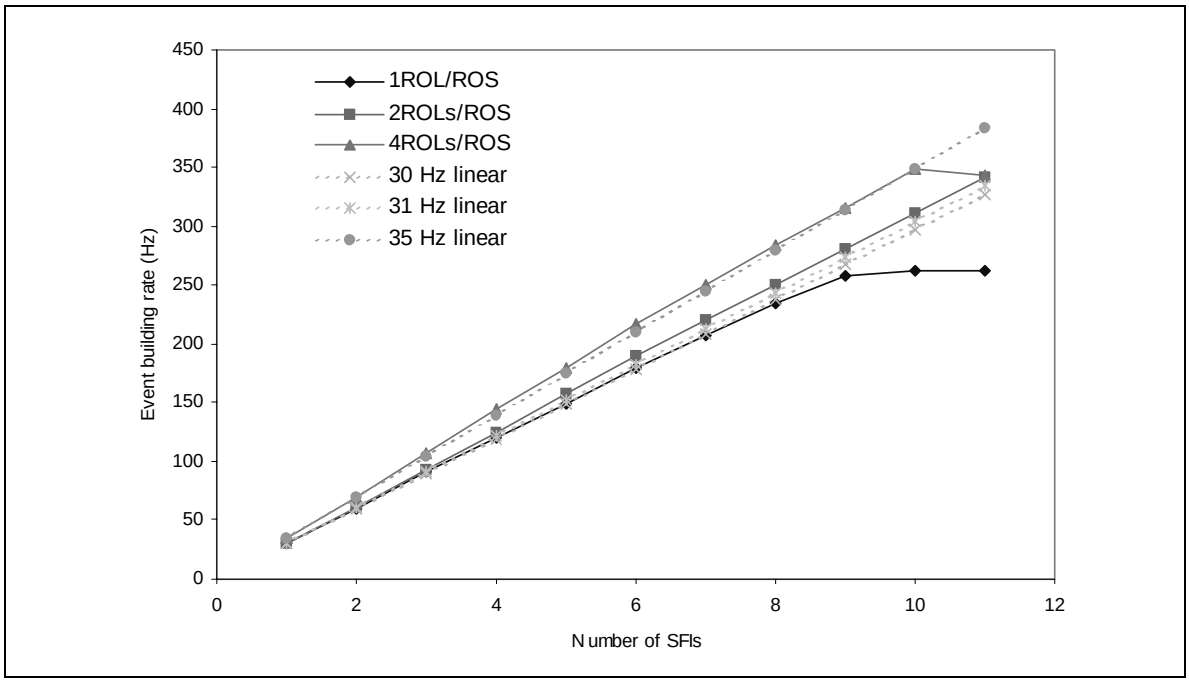

Figure 8-21 The event building rate versus the number of SFIs in the system, without Ethernet flow control

8.3.2.3.3 Performance using the push scenario

The event building performance with QoS applied at the IP level has been measured for the push scenario [8-37] and the Linux kernel scheduling frequency set to ~ 4 kHz. The results of these studies are shown in Figure 8-22. It can be observed that in the case of the push scenario without QoS, packet loss occurs at the SFI when the message size exceeds 4 kbyte. With QoS applied, packet loss is not observed when the QoS is used to limit the output bandwidth at the level of the ROS to 40 Mbit / s.

In the conditions in which no packet loss occurs, the push scenario has a better performance than the pull scenario due to the additional data control messages implied by the pull scenario. However, over the full range of ROS event fragment sizes being studied for event building, the pull scenario performs better.

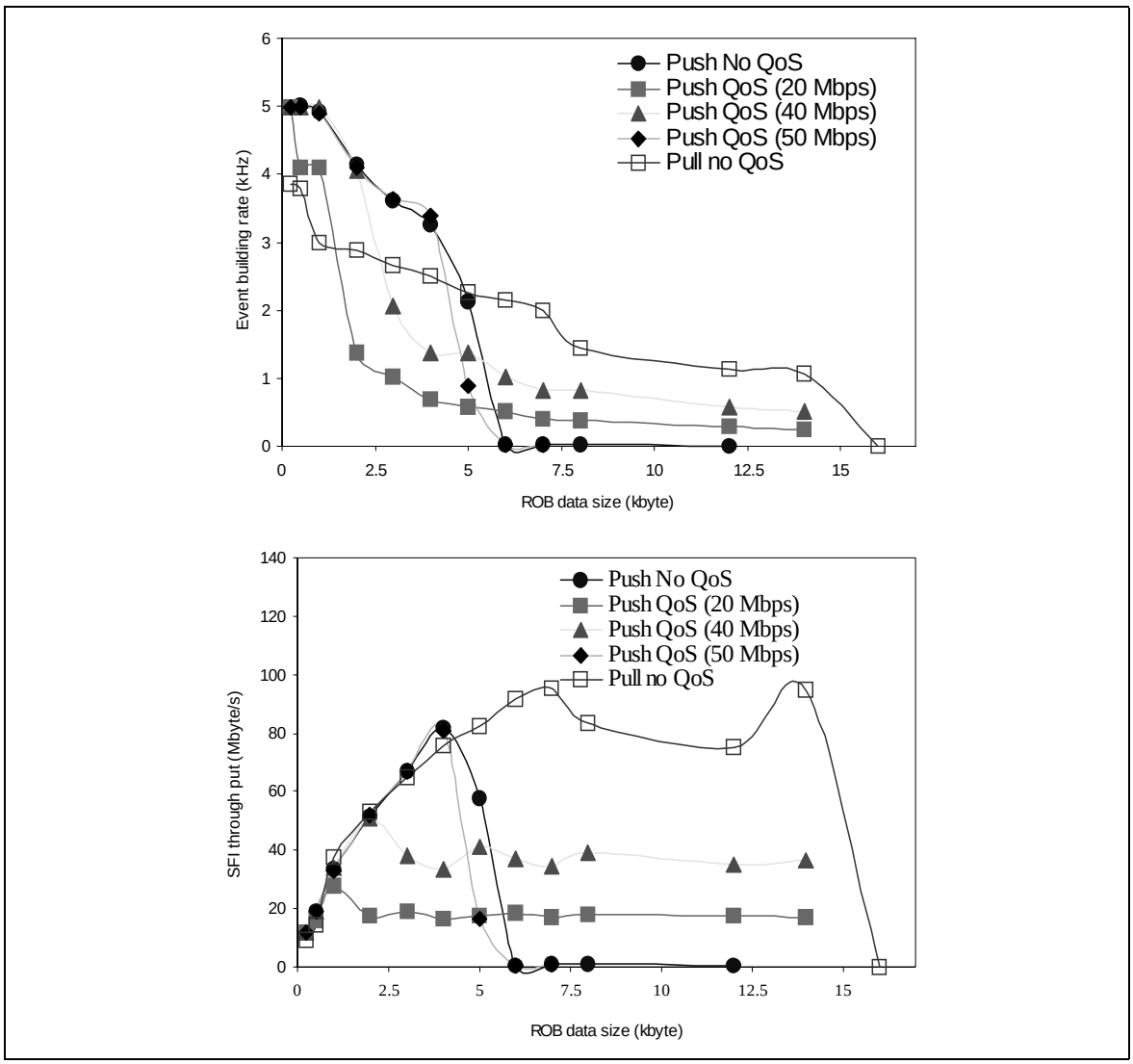

Figure 8-22 (a) Event Building rate and (b) throughput as a function of event fragment size and different QoS parameter values for the push scenario

The results, obtained on a small system, show that QoS as a shaping technique is not always effective for event building in the range required. Whether these conclusions are applicable to the

full-sized system remains to be established. Enhancements in the performance of the pull scenario with QoS applied have still to be investigated.

8.4 Summary

The performance of the main DataFlow components has been presented. It has been shown that the prototype design and implementation of these components meets the requirements on the baseline architecture and thus support the choice of the baseline architecture presented in Chapter 5. Notably, it has been demonstrated that the collection of RoI data, required by the LVL2 trigger, can be done within a time that contributes less than 10% to the overall time budget of the LVL2 event processing. It has also been demonstrated that the performance of the RoI collection functionality scales sufficiently to the rates required of the final system. Similarly, the event building has been demonstrated to sufficiently scale, within the constraints of the testbed, to the performance requirements of the final systems. Departures from linear scaling have been understood and will be addressed in the final design and implementation. Two scenarios for event building were presented leading to the conclusion that the scenario in which the SFI controls the flow of traffic to it results in a better overall performance and avoid the overloading of network links. Results from initial measurements have been shown on the respective performance of the bus- and switch-based ROSs. The relative merits of these two solutions will be further studied when the final prototype RoBIn becomes available.

Further measurements are required on larger testbeds to confirm the results obtained the testbeds currently available and enhance the reliability and predictability of system models, see Section 14.6.2.1.

8.5 References

8-1 ROD Crate DAQ Task Force, *Data Acquisition for the ATLAS ReadOut Driver Crate (ROD Crate DAQ)*, ATL-D-ES-0007, https://edms.cern.ch/document/344713

8-2 R. Spiwoks, *Workplan for ROD Crate DAQ*, ATL-D-MG-0001 () https://edms.cern.ch/document/364357

8-3 *Trigger & DAQ Interfaces with Front-End Systems: Requirement Document*, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/DIG/archive/document/FEdoc_2.5.pdf

8-4 *ATLAS High-level Triggers, DAQ and DCS Technical Proposal*, CERN/LHCC/2000-17, 31 March 2000 http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/SG/TP/tp_doc.html

8-5 The S-LINK *interface Specification*, http://edmsorweb.cern.ch:8001/ceder/doc.info?documnet_id=110828

8-6 C. Bee et al., *The raw event format in the ATLAS Trigger & DAQ*, http://preprints.cern.ch/cgi-bin/setlink?base=atlnot&categ=Note&id=daq-98-129

8-7 Recommendations of the Detector Interface Group - ROD Working Group, ATL-DF-ER-0001 () https://edms.cern.ch/document/332389

8-8 The CMC standard. Common Mezzanine Cards as defined in IEEE P1386/Draft 2.0 04-APR-1995, Standard for a Common Mezzanine Card Family: CMC (the CMC Standard).

8-9 Design specification for HOLA, https://edms.cern.ch/document/330901

8-10 Procedures for Standalone ROD-ROL Testing, G. Lehmann et al., 27 July 2001, ATC-TD-TP-0001 () http://edmsoraweb.cern.ch:8001/cedardoc.info?document_id=320873&version=1

8-11 B. Gorini, *ROS Software Architecture Document*, ATL-DQ-ES-0010 () https://edms.cern.ch/document/364343

8-12 ReadOut Subsystem, *Summary of prototype RoBIns*, ATL-DQ-ER-001 () https://edms.cern.ch/document/382933

8-13 Readout sub-system test report (using DAQ -1.), *in preparation*.

8-14 R.Cranfield et al., *ROS Requirements*, ATL-DQ-ES-0007 () https://edms.cern.ch/document/356336

8-15 B. Green et al., *RobIn TDR-Prototype HLDD*, ATL-DQ-EN-0001 () https://edms.cern.ch/document/356324

8-16 B. Green et al., *RobIn TDR-Prototype DLDD*, ATL-DQ-EN-0002 ()

8-17 https://edms.cern.ch/document/356328

8-18 B. Green et al., *RobIn TDR-Prototype Software Interface*, ATL-DQ-EN-0003 () https://edms.cern.ch/document/356332

8-19 The MPRACE board, http://mp-pc53.informatik.uni-mannheim.de/fpga/

8-20 Gigabit Ethernet testers for ATLAS event building studies, *in preparation*.

8-21 B. Gorini et al., ROS Test Report, *in preparation*.

8-22 P. Werner amd A. Bogaerts, *Pseudo-ROS Requirements*, ATL-DQ-ES-0039 () https://edms.cern.ch/document/391569

8-23 P. Werner, *Pseudo-ROS Design Document*, ATL-DQ-ES-0044 ()
https://edms.cern.ch/document/391576

8-24 M. Abolins et al., *Specification of the LVL1 / LVL2 trigger interface*, ATL-D-ES-0003 ()
https://edms.cern.ch/document/107485

8-25 R. Blair et al., *A Prototype RoI Builder for the Second Level Trigger of ATLAS Implemented in FPGA's*, ATL-DAQ-99-016

8-26 B.M. Barnett et al., *Dataflow integration of the calorimeter trigger CPROD with the RoI Builder and the ROS*, ATL-DA-ER-0016 ()
https://edms.cern.ch/document/326682

8-27 H-P. Beck and C. Haeberli, *High-Level Description of the Flow of Control and Data Messages for the ATLAS TDAQ Integrated Prototypes*, ATL-DQ-ES-0028 ()
https://edms.cern.ch/document/391514

8-28 H-P. Beck and F. Wickens, *The Message Format used by DataCollection in the ATLAS TDAQ Integrated Prototype*, ATL-DQ-ES-0035 ()
https://edms.cern.ch/document/391557

8-29 M. Ciobotaru and S. Stancu, Networking Tests, *in preparation*.

8-30 Networking group activity, *in preparation*.

8-31 R. Hauser and H-P. Beck, *Requirements for the Message Passing Layer* ATL-DQ-ES-0024 ()
https://edms.cern.ch/document/391495

8-32 R. Hauser, *Message Passing Interface in the LVL2 Reference Software*, ATL-DQ-ES-0023 ()
https://edms.cern.ch/document/391490

8-33 R. Huges-Jones, *The use of TCP/IP for Real-Time Messages in ATLAS Trigger/DAQ*, ATL-DQ-EN-0015 ()
https://edms.cern.ch/document/393752

8-34 P. Golonka, *Linux network performance study for the ATLAS DataFlow System*, ATL-DQ-TR-0001 ()
https://edms.cern.ch/document/368844

8-35 R. Hauser and H-P. Beck, *ATLAS TDAQ DataCollection Requirements*, ATL-DQ-ES-0048 ()
https://edms.cern.ch/document/391580

8-36 R. Hauser and H-P. Beck, *ATLAS TDAQ/DCS DataCollection Architectural Design*, ATL-DQ-ES-0046 ()
https://edms.cern.ch/document/391578

8-37 Y. Yasu et. al., *Summary of Studies on ATLAS DataCollection software with QoS*, ATL-DQ-TR-0004 ()
https://edms.cern.ch/document/38292

8-38 M. Gruwe & B. Di Girolamo, *DataFlow performances measurements: the Event Building*, ATL-DQ-TR-0002 ()
https://edms.cern.ch/document/375101

8-39 C. Haeberli et al., *ATLAS DataCollection: Event Building Test Report*, ATL-DQ-TR-0003 ()
https://edms.cern.ch/document/382415

9 High-Level Trigger

9.1 HLT overview

The High-level Trigger (HLT) contains the second and third stages of event selection. It comprises three main parts: The LVL2 system, the Event Filter (EF), and the Event Selection Software (ESS). Section 9.2 describes the components of the LVL2 system, Section 9.3 describes those for the EF, and Section 9.4 describes the operation of the LVL2 and EF systems. Although the algorithms used at LVL2 and the EF are different, it has been decided to use a common software architecture for the event selection code across LVL2, EF and off-line studies. This facilitates use of common infrastructure (such as detector calibration and alignment data) and simplifies off-line studies and development of the HLT algorithms. The common ESS architecture is described in Section 9.5.

The basic structure of the HLT selection chain is shown in Figure 9-1 in a simplified form. The starting point for the HLT is the LVL1 Result. It contains the LVL1 trigger type and the information about primary RoIs that caused the LVL1 accept, plus secondary RoIs not used for the LVL1 accept. Both types of RoIs are used to seed the LVL2 selection. The concept of seeded reconstruction is fundamental, particularly at LVL2 (apart from the special case of B-physics — see Section 13.4.5).

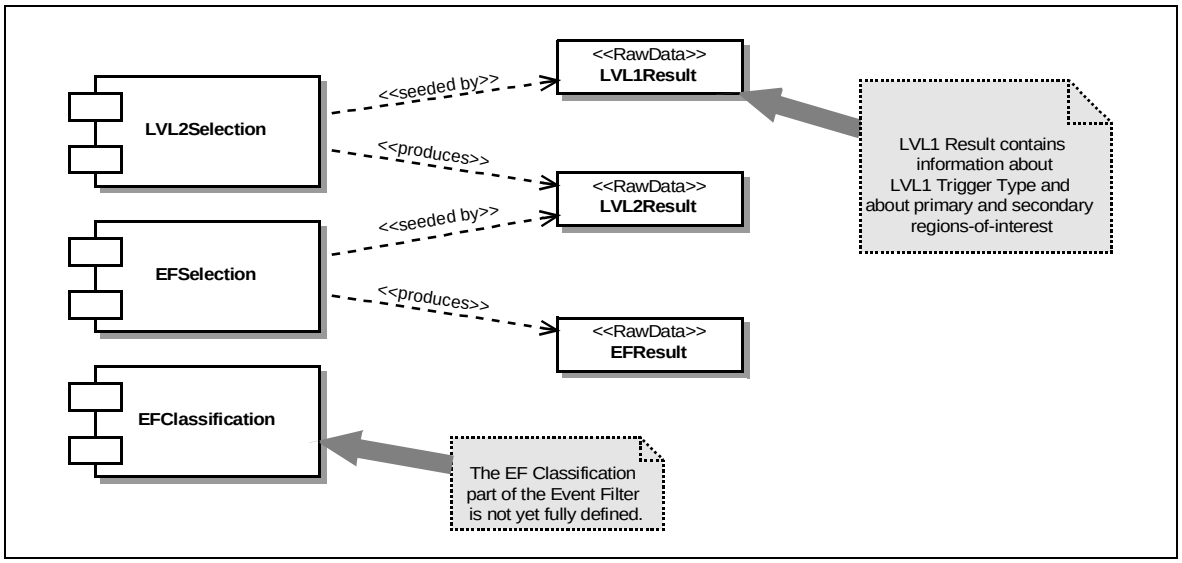

Figure 9-1 The high-level trigger selection chain with the LVL2 and EF selection each seeded by the preceding trigger

The LVL2 Result provides a seed for the EF selection, thus playing a similar role for the EF as does the LVL1 Result for the LVL2. It will also be possible to seed the EF directly with the LVL1 Result in order to study, for example, the LVL2 performance. The EF and the LVL2 Results, containing the physics signatures from the trigger menu that were satisfied and higher level reconstruction objects, will be appended to the raw event data.

The EF classification is yet to be defined in detail. Possibilities considered include special selections for calibration events and for new physics signatures, i.e. a discovery stream. The EF Result can be used to assign tags to the events or even assign them to particular output streams.

The flow of data is as follows. Data for events accepted by the LVL1 trigger are sent from the detector front-end electronics to the ROSs, containing ~1600 ROBs. In parallel, information on the location of RoIs identified by LVL1 is sent to the LVL2 Supervisor to guide the LVL2 event selection. Figure 9-2 shows the exchange of messages that are then used to perform the HLT process.

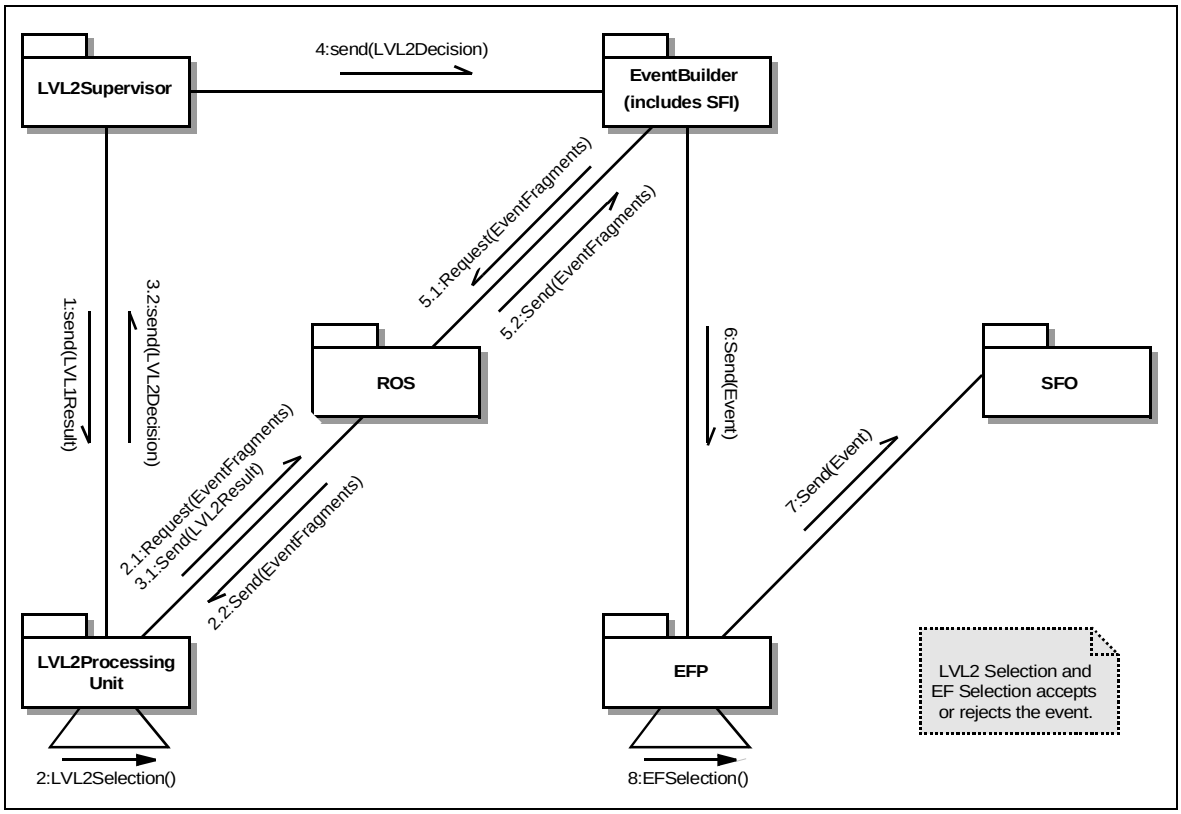

Figure 9-2 The exchange of messages between HLT Components

The LVL2 Supervisor sends the LVL1 Result to the L2PU, where the LVL2 selection is performed. Using the LVL1 Result as guidance, specialized LVL2 algorithms request a sub-set of the event data from the ROSs to perform the event selection. In this way only a few per cent of the event data need to be transferred to the LVL2 system — thus considerably reducing the network bandwidth required. For events accepted by LVL2, details are sent to the ROS (in the LVL2 Result) to be included in the event. The L2PU send the LVL2 Decision back to the LVL2 Supervisor, which forwards them to the Event Builder. Within the Event Builder the SFI assembles each accepted event into a single record, including the LVL2 Result. The built event is then passed to the EFP where off-line algorithms are applied guided by information from the LVL1 and LVL2 triggers to further refine the event selection. Events passing the EF are then passed to the SFO for permanent storage and off-line analysis.

Details of how the HLT system fits into the overall HLT/DAQ architecture have been given in Chapter 5.

9.2 LVL2

9.2.1 Overview

The LVL2 trigger provides the next stage of event selection after the hardware-based LVL1 trigger. It uses RoI guidance received from LVL1 to seed the validation and enhancement of the LVL1 trigger using selected full granularity event data. Components involved in the LVL2 process are the RoI Builder (RoIB), the LVL2 Supervisor (L2SV), the LVL2 Processors (L2P), the ROS and the pROS. The RoIB assembles the fragments of information from the different parts of the LVL1 trigger and transmits the combined record to a L2SV. The L2SV selects a L2P for the event and sends the LVL1 information to that processor and then waits for the LVL2 Decision to be returned. The L2P runs the LVL2 event selection software, requesting event data as required from the ROSs and returns the LVL2 Decision to the L2SV. For events which are to be passed to the EF the L2P also sends a more detailed LVL2 Result to the pROS to be included in the event to be built. Details of the ROS and the DataFlow aspects of the gathering of data within an RoI for LVL2 are described in Chapter 8. Details specific to the LVL2 selection process of all of the other components are given below.

9.2.2 RoI Builder

For each LVL1 accept the various parts of the LVL1 trigger send information on the trigger including the RoI positions and thresholds passed. The RoIB combines these fragments into a single record that is passed to a L2SV. In the baseline design each L2SV will see only a sub-set of the L2Ps, thus the choice of L2SV affects the load-balancing between L2Ps. This routing would normally be on a round-robin basis, but busy L2SVs can be skipped and the design also allows more complex algorithms, including use of the LVL1 trigger type. A fuller description of the RoIB is given in Section 8.2.1.

9.2.3 LVL2 Supervisor

The LVL2 Supervisors (L2SVs) are a small group of processors (of order 10) that supervise the flow of events in LVL2 and mediate between the LVL2 system and the LVL1 system. In order to simplify farm management and to keep software uniform the processors will be similar to those used in the LVL2 farm, however, each Supervisor processor needs an interface (S-LINK in the baseline architecture) to receive the data from the RoIB (see Section 8.2.1). The L2SV application is based on the Data Collection Framework described in Section 8.3.2.

The context of the L2SV is indicated in Figure 9-3. The L2SV receives information on the LVL1 trigger in a single record (LVL1 Data) from the RoIB. It selects a L2P for the event and passes the LVL1 Data to the L2PU (running in the L2P) in the LVL1 Result message. Once the L2PU has decided whether the event should be accepted or rejected it passes back a LVL2 Decision message. If the decision is an accept or if the L2SV has collected a predetermined number of rejects, the LVL2 Decision Group message is sent to the DFM where the DFM coordinates clearing of the buffers and readout to the EB.

In selecting a L2P the Supervisor exercises a load balancing function. Currently the Supervisor is designed to either select L2Ps from its available pool via a simple round-robin algorithm, or to examine the number of events queued and to assign the event to the least loaded processor.

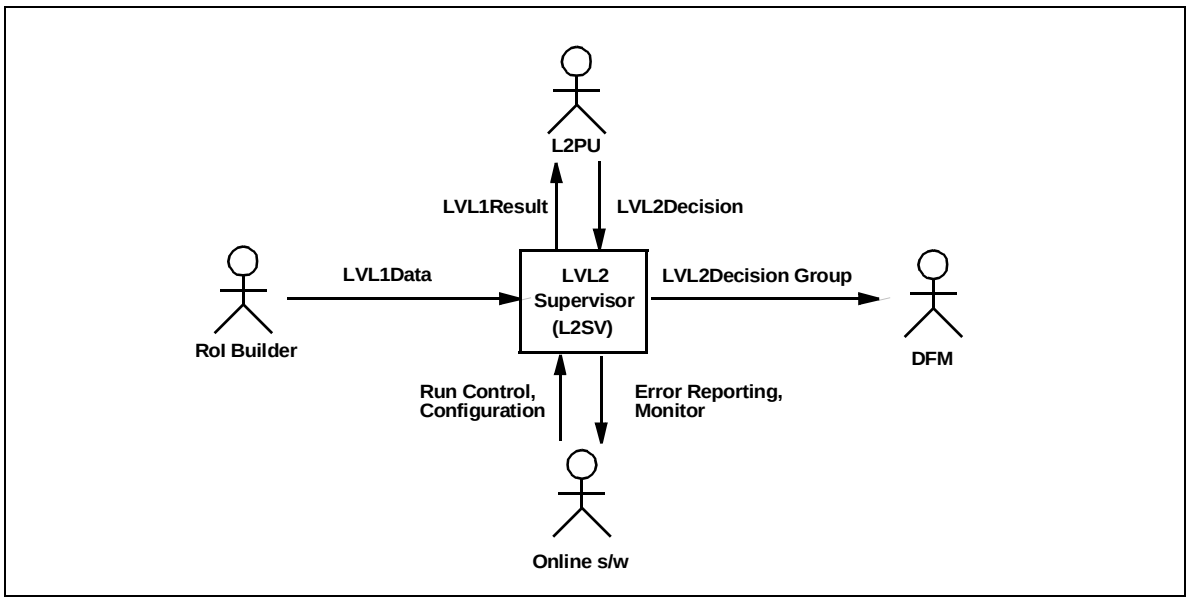

Figure 9-3 Context of LVL2 Supervisor

The L2SV can also use the LVL1 Trigger Type when selecting an L2P, this could be used to restrict specific triggers types to a sub-set of the L2Ps, for example when debugging code for a new trigger. It is also foreseen to automatically accept certain classes of LVL1 triggers without involving the L2Ps, for example detector calibration events. This could be extended to provide unbiased samples to the EF.

The supervisor software is based on the Data Collection Framework described in Section 8.3.2. For normal data taking the LVL1 Data is received from the RoIB, but for diagnostic purposes the Supervisor can also retrieve LVL1 Data from a file or generate it internally.

For further details of the L2SVs see Section 8.3.1 and Section 8.3.2. Details of the Requirements and Design of the L2SV are given in [9-1] and [9-2], respectively.

9.2.4 LVL2 Processors

The LVL2 selection is performed in a farm of processors, today assumed to be dual processor PCs, running at 8 GHz, in 1U rack-mounted format. The nominal budget allowed for the mean processing time for the LVL2 decision per event is ~10 ms. However, there will be large variations from event to event, many events will be rejected in a single selection step in a significantly shorter time than the mean, whilst others will require several sequential steps, and a few events may take many times the mean latency. Each event is handled in a single processing node, requesting selected event data from the ROSs for each RoI only when required by the algorithms. To maintain a high efficiency of processor utilization even when it is waiting for such RoI event data, several events are processed in parallel in each processing node.

The processing is performed in a single application running on each node (i.e. PC host). The application has three main components: the L2PU; the PESA Steering Controller (PSC); and the ESS. The L2PU handles the Data Flow with other parts of HLT/DAQ, including message passing, configuration, control and supervision. The PSC runs inside the L2PU and provides the required environment and services for the ESS. As the L2PU handles the communication with the L2SV and the ROSs, interfaces have to be provided for the various messages between the L2PU

and the ESS. The PSC provides the interface for the LVL1 Result and returns the LVL2 Result (from which the LVL2 Decision is derived). The interface for the RoI event data to be retrieved from the ROS is provided separately and is described in Section 9.2.4.3.

The use of FPGA co-processors [9-3] has been studied for some CPU-intensive algorithms, for example non-guided track finding in the inner detector [9-4]. For the current trigger menus, however, it has been found that their use is not justified and therefore they are not discussed further here.

9.2.4.1 L2PU

The design and implementation of the L2PU is based on the Data Collection Framework described in Section 8.3.2 from which it uses the following services: application control, initialization and configuration, error reporting, application monitoring, message passing, and, for the purpose of performance evaluation, the instrumentation. For further details of the L2PU design see [9-5].

The L2PU communicates with the LVL2 Supervisor from which it receives the RoI information (originating from the LVL1 Trigger) and to which it returns the LVL2 Decision. RoI data (in the form of ROB Fragments) are requested from the ROSs, on instigation of the LVL2 Selection algorithms.

The actual selection algorithms run inside one out of several ‘Worker threads’, each processing one event. This multi-threaded approach has been chosen to avoid stalling the CPU when waiting for requested RoI data to arrive (from ROSs). This also allows efficient use of multi-CPU processors, but requires that LVL2 selection algorithms must be thread-safe. Specific guidelines to LVL2 algorithm developers are given in [9-6]. Some asynchronous services (application monitoring, input of data) are also executed in separate threads.

The LVL2 event selection takes place inside the PSC which has a simple interface to the Data Collection framework: it receives the LVL1 RoI information (LVL1 Result) as input parameter and it returns the LVL2 Result. Figure 9-4 illustrates what happens for each event. The LVL2 Supervisor selects an L2PU and sends the LVL1 Result. This L2PU stores the received LVL1 Result in a shared queue. When a Worker thread becomes available it unqueues an event, starts processing it and produces a LVL2 Result. Finally the LVL2 Decision is derived from the LVL2 Result and returned to the LVL2 Supervisor. For positive decisions, the LVL2 Result is also sent to the pROS (see Section 8.1.3.4 and Section 9.2.5).

When selection algorithms require ROB data, the ‘ROB Data Provider’ is activated, which functions as shown in Figure 9-5. The ROB Data Collector takes a ‘list of RoBs’ as input parameter and returns a ‘list of ROB Fragments’. The ROB Data Collector takes care of sending out requests for data to the appropriate ROSs, waits for all data to arrive, assembles the received ROS Fragments into a list of ROB Fragments that are returned to the caller. As reported in Section 8.3.2.2 the performance for collecting RoI data from ROBs has been measured in testbeds. It exceeds by a large margin the required I/O capacity.

9.2.4.2 PESA Steering Controller (PSC)

The PESA Steering Controller (PSC) is the HLT component that interfaces the L2PU and the ESS. The purpose of the PSC is threefold: to allow the L2PU to host and control selection software developed in the offline framework; to allow the algorithm steering software to be shared

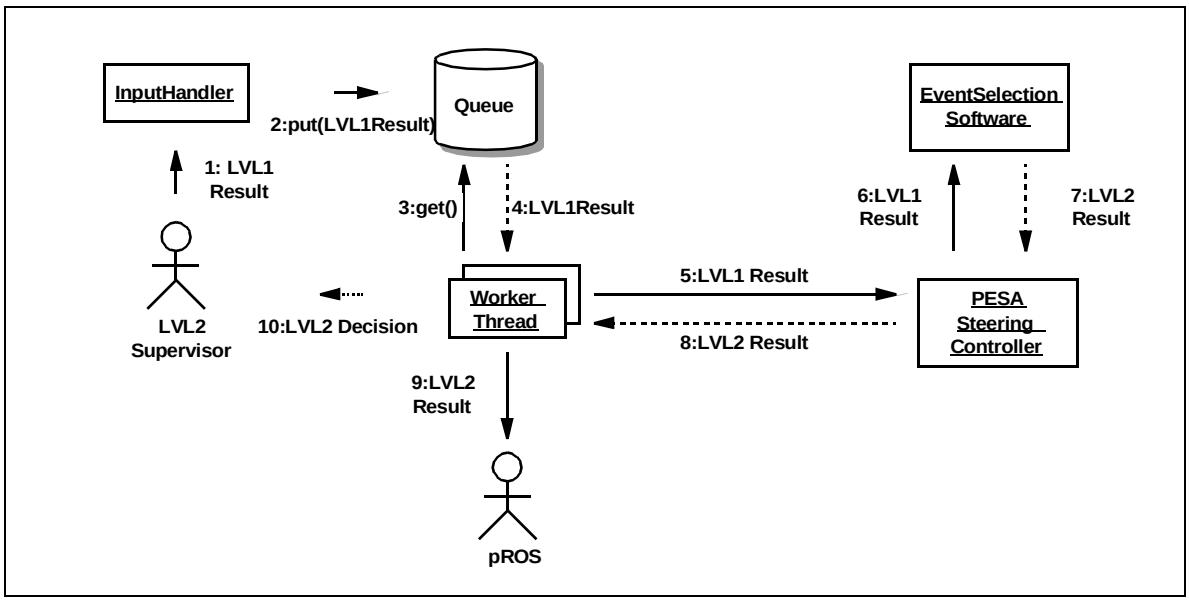

Figure 9-4 Collaboration diagram showing the successive steps that are applied to each event received from LVL1 leading to the LVL2 Decision

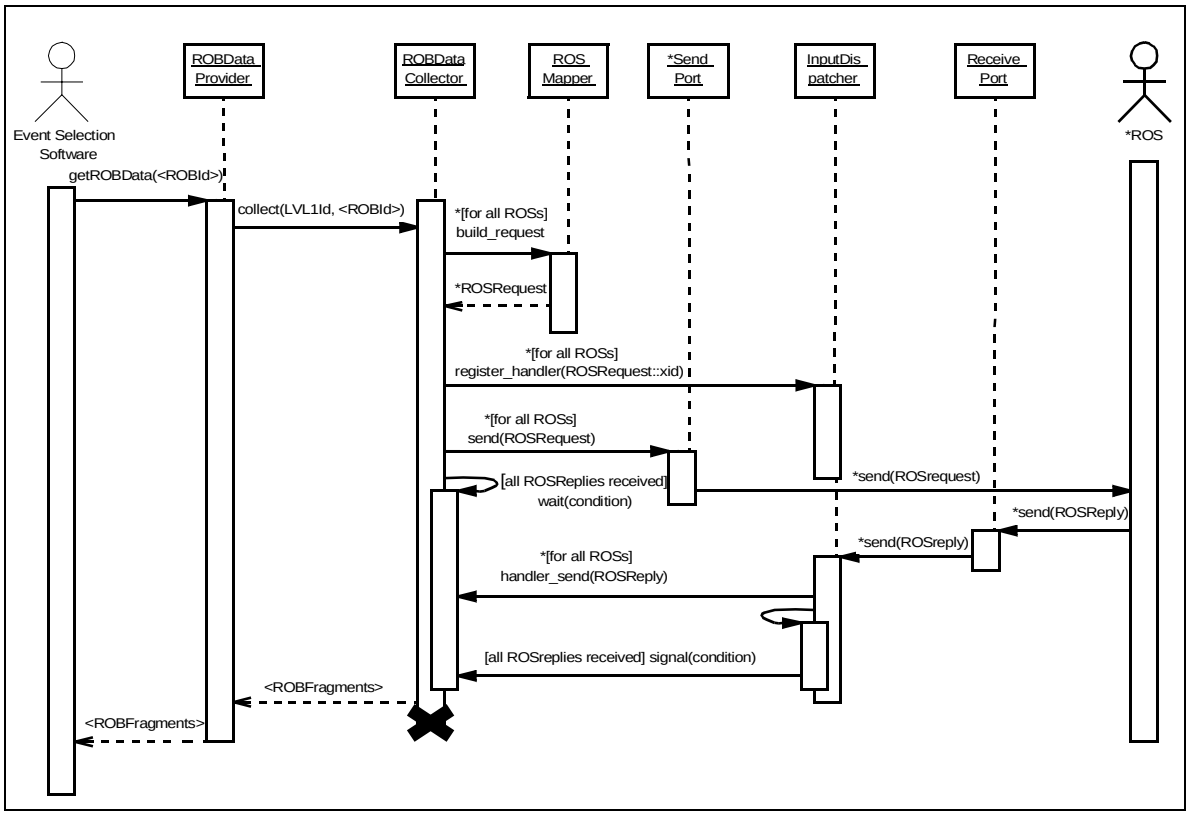

Figure 9-5 Data Collection by the L2PU of the data within a list of ROBs, corresponding to an RoI

with the EF; and to provide a mechanism for transmitting the LVL1 and LVL2 Results between the Data Flow system and the ESS.

The key to the PSC design is to place this interface where the functionality of the LVL2 Data Flow and ESS can be cleanly separated. The location chosen is the Finite State Machine (FSM) of

the L2PU. The PSC is realized as a local ‘state-aware’ replica of the Data Collection’s FSM. It thus provides the means for forwarding changes in the run control state from the LVL2 Data Flow software to the ESS. Since the ESS is being developed in the ATLAS offline framework, Athena [9-7], which is itself based on Gaudi [9-8], the PSC has been designed [9-9] to re-use the framework interfaces defined in Gaudi.

Figure 9-6 illustrates the sequence of interactions of the PSC with the L2PU and the ESS. The figure shows three states: *Configure*, *Start*, and *Stop*. During the *Configure* phase, configuration and conditions meta data are obtained from external databases via an HLT-online interface. These data are then used to configure the ESS and all associated components. As Figure 9-6 (left) shows, during this phase multiple Worker Threads are also set up. After a *Start*, the PSC receives an ‘execute event’ directive with a LVL1 Result as an argument. The PSC then returns (after execution of the ESS) the LVL2 Result directly to the L2PU.

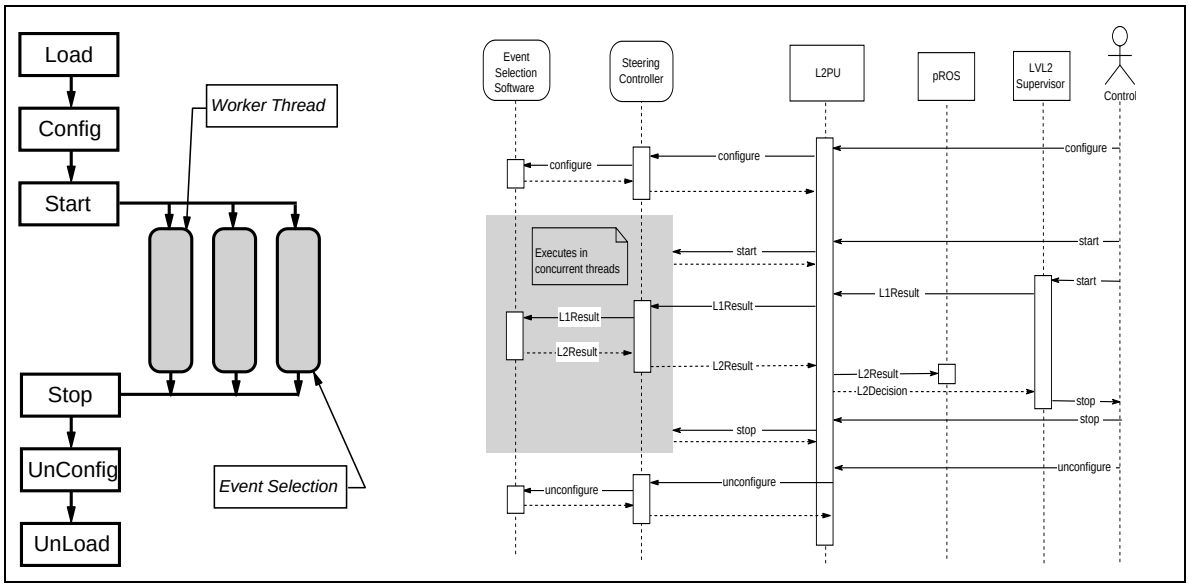

Figure 9-6 The L2PU Finite State Machine (left) and the PESA Steering Controller (right)

An important aspect of this approach is that the LVL2 event handling is managed entirely by the Data Collection framework. The PSC then does not need to interact directly with the input thread, the L2SV, or with the pROS. The requests for event data fragments are hidden behind the ROB Data Provider Service.

After a *Stop*, the PSC terminates algorithm execution. At this stage, run summary information can be produced for the selection process.

Since the ESS executes in multiple worker threads, the PSC must provide a thread-safe environment. At the same time, and in order to provide an easy-to-use framework for offline developers, the PSC must hide all technical details of thread handling and locks. Thread safety has been implemented in the PSC by using Gaudi’s name-based object and service bookkeeping system. Copies of components that need to be thread-safe are created in each worker thread with different labels. The labels incorporate the thread-ID of the worker thread, as obtained from the Data Collection software. The number of threads created by the Data Collection software is transferred to the PSC, which transparently creates the number of required copies. In this scheme, the same configuration can be used in the offline and in the LVL2 environments; the thread-ID collapses to *null* in the offline software.

After integrating the PSC with the Data Collection software, both performance and robustness tests were carried out on a dual-processor 1.533 GHz Athlon machine (for details, see [9-9]). The PSC ran for over 50 hours with three threads with an early version of the selection software prototype [9-10]. The prototype ran successfully on both single and dual-CPU machines, showing it to be thread-safe. A direct measurement of the PSC overhead yielded 13 µs per event, well within the 10 ms nominal LVL2 budget.

9.2.4.3 Interfaces with the Event Selection Software

In Figure 9-7 a package diagram is shown for the ESS running in the L2PU. The communication with external systems and subsystems, including the L2SV and the ROS, are hidden from the ESS. The ESS is initialized as shown in Figure 9-6. The PSC provides the ESS with the LVL1 Result and requests the LVL2 selection. To provide the LVL2 Result the ESS needs to access ROB data fragments and stored meta data. The ROB data requests are sent via the ROB Data Provider Service to the ROB Data Collector. Services are used to access meta data, these include the geometry, conditions, and B-Field map information. Monitoring services include histogramming and messaging.

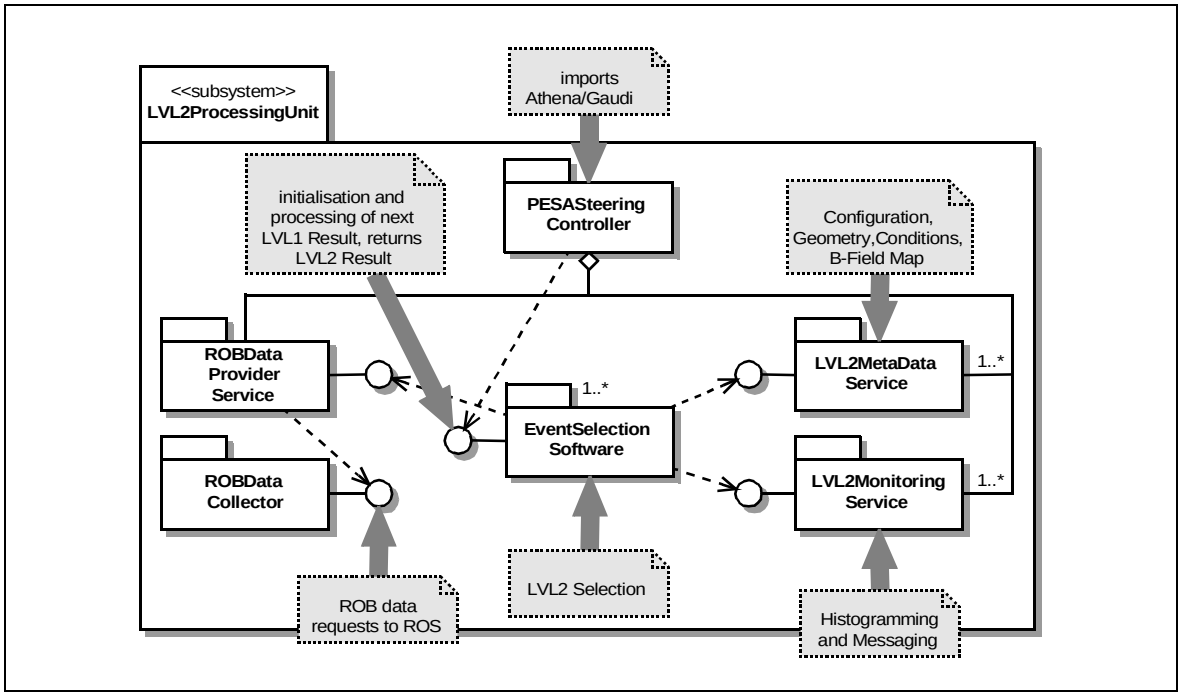

Figure 9-7 The dependencies of the Event Selection Software performing the LVL2 selection in the L2PU

9.2.5 pROS

For all events accepted by LVL2 (whether a normal accept, a forced accept or a pre-scale) details of the LVL2 processing (and the full LVL1 Result received from the Supervisor) are provided by the LVL2 Processor for inclusion in the event. This information is sent via the network as a ROB fragment to the pROS, where it is stored pending a request from the Event Builder. When the event is built the pROS is included with all the other ROSs — thus including the LVL2 Result into the built event, which is passed to the EF. A single such unit is sufficient for the size of LVL2

Result foreseen (not exceeding a few kilobytes) as it only has to operate at the event building rate (~3 kHz). Further details of the pROS are given in Section 8.1.3.4.

9.3 Event Filter

The Event Filter (EF) is the third and last stage of the on-line selection chain. It makes use of the full event information. It will use the offline framework (Athena) and the Event Selection Software, that will execute filtering algorithms directly based on the offline reconstruction.

9.3.1 Overview

The functionality of the EF has been logically distributed between two main entities:

- the Event Handler (EH) performs the activities related to event selection. This includes: the data flow, both between the main DAQ system and the EF, and internally between different parts of the EF; the framework to support various tasks including the Processing Tasks (PT) where the ESS runs.
- the EF Supervisor handles the control operations, in co-ordination with the overall TDAQ control system. Its responsibilities include the monitoring of the EF. For further details see Section 9.4 below.

The EF has been designed so that additional functions can be added without jeopardizing the selection activity. Examples of such extra activities are the global monitoring of the detectors or tasks related to alignment and calibration.

As described in Chapter 5, the baseline architecture of the EF relies on commodity components, namely PCs linked by Ethernet networks. A detailed description of the baseline architecture for EF may be found in [9-11]. The EF Farm is organized in independent sub-farms, in the baseline each one connected to a different SFI. The possibility of dynamic routing between SFIs and sub-farms is also being considered. Several SFOs can be connected to a given sub-farm. A given SFO may be accessed by different sub-farms. The proposed layout is shown on Figure 9-8.

9.3.2 Event Handler

9.3.2.1 Overview

A detailed list of requirements for the EH, based on analysis of the external constraints and use cases, can be found in [9-12].

The EH receives events from the SFIs. The events are distributed to Processing Tasks which run the ESS. Information contained in the event header can be used to route events to specific PTs. Events selected for final storage will be sent to an SFO, which can be selected according to classifications made in the PT, thus providing the possibility of dedicated output streams. Information created during the selection process will be appended to the events sent to permanent storage.

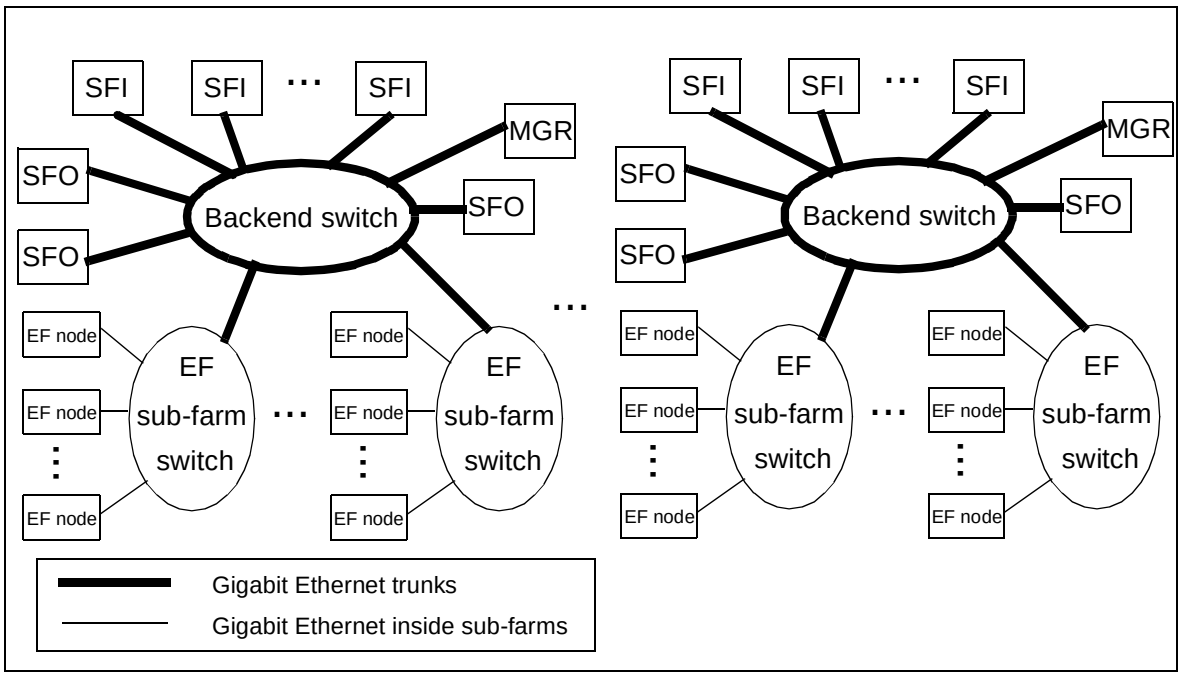

Figure 9-8 Proposed layout for the EF Farm

The design of the EH is based on the following principles: data flow and data processing are separated; the flow of events is data driven, i.e. there is no data flow manager to assign the event to a specified target; data copying on a given processing node is avoided as much as possible to save time and CPU resources.

Data movement between the different phases of the processing chain is provided by the Event Filter Dataflow process (EFD), while the processing is performed in independent Processing Tasks. There is one EFD process per processing node (i.e. per PC of the EF farms). One or several PTs can connect to the EFD at different stages of the processing chain. Event passing is made by a shared memory mapped file using a local disk for storage. Synchronization is maintained via messages using UNIX sockets. Details can be found in [9-13] and [9-14].

9.3.2.2 Event Filter Dataflow

The processing of the events is decomposed into steps that can be configured dynamically. Each step provides a basic function: event input or output, event sorting, event duplication, internal processing (e.g. for monitoring purposes), external processing, etc.

The different stages of the process chain are implemented by ‘tasks’ to provide specific functionality. Examples are tasks providing the interface with the DAQ Data Flow; tasks to perform internal monitoring activity (e.g. counting the events which pass through); tasks to duplicate events to send them in parallel towards different processing paths; tasks to sort events towards different data paths according to internal flags (e.g. the result of the reconstruction and selection process). However, functions such as the event selection (using the ESS) and pre- or post-processing are performed in separate processes and so tasks are included to provide the interface from the EFD process to these external Processing Tasks. An example of an EFD implementation, illustrating the use of these tasks, is given in Figure 9-9.

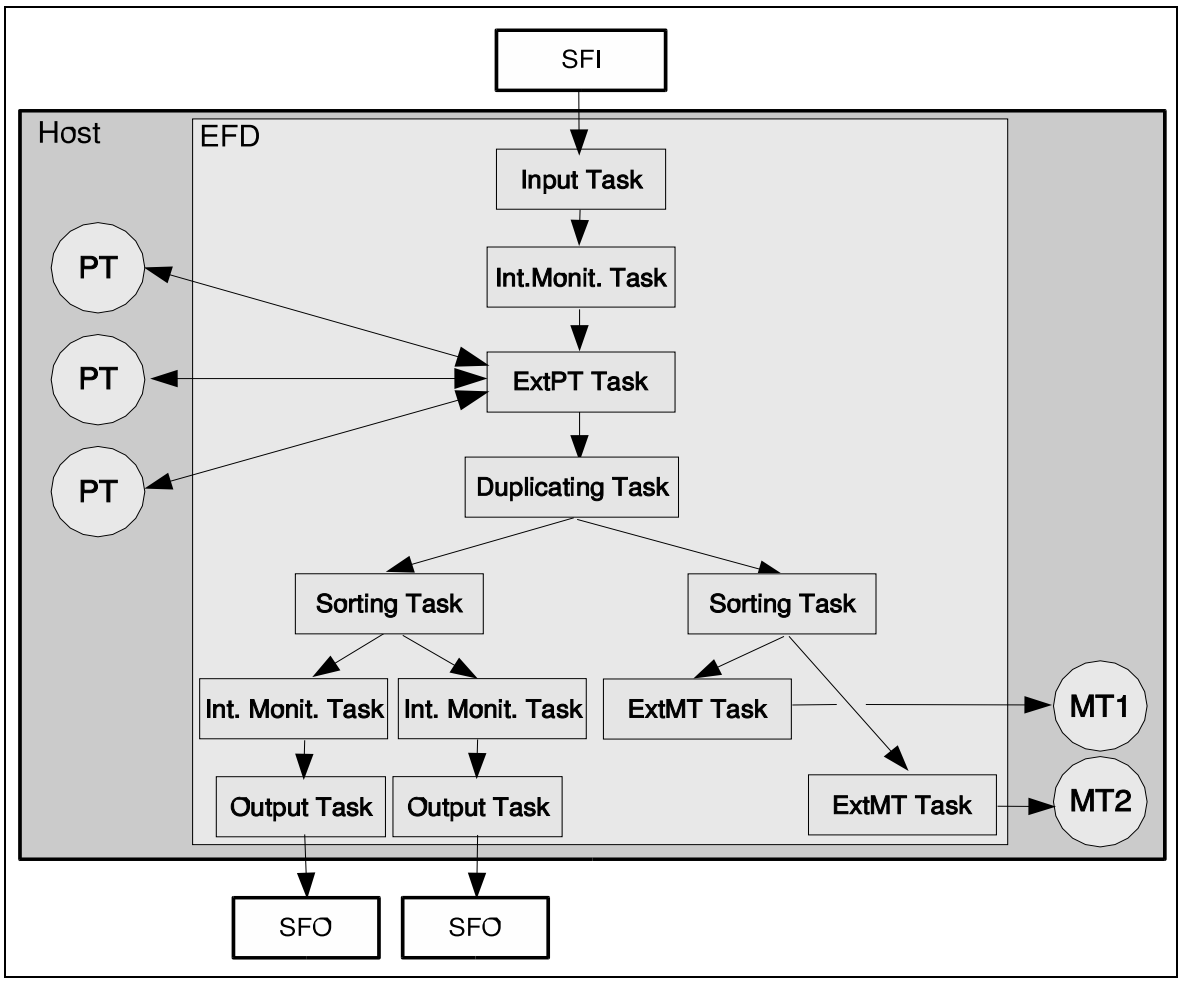

Figure 9-9 An example of an EFD implementation

In this example, an *Input Task* makes the interface with the SFI. Events are counted in an *Internal Monitoring Task*. The *External PT Task* provides the interface for synchronization and communication with PTs performing the event selection. Events which are accepted by the PT are then duplicated. In one path, events are counted and passed to *Output Tasks* to be sent to permanent storage. In the other path, on which prescaling may be applied, events are made available to different monitoring tasks according to the tag they received during selection in the PT.

The `Input Task` maps events into shared memory. For efficiency, event data is not copied between the processing steps, but pointers are passed to the different processing entities. The information produced by the external PTs can be made available to other (monitoring) tasks by storing it in the shared memory. The shared memory segment is mapped to a file which ensures that data is saved by the operating system in case of problems, e.g. a crash of the EFD process. When the EFD is restarted, events received from the Data Flow can be recovered from the shared memory. An automatic recovery procedure allows an event which has caused a crash of the PT to be reprocessed again. The PT crash is detected by the socket hang-up, and the event is tagged as having been already processed. If the event causes a second crash it is sent to a dedicated output channel.

The tasks are daisy chained with each task knowing the identity of the next task to execute for the current event. The chaining mechanism is based on a `Work Assignment` thread which first extracts an event from a Work Queue. Figure 9-10 shows the sequence diagram corresponding

to this mechanism. The `getWork()` method returns an (Event pointer, Task pointer) pair from the Work queue. It then calls the `processEvent` method of the Task with the pointer to the Event. After processing, the method returns the pointer to the next Task, or the `NULL` pointer if it was the last task in the chain. This feature allows the dynamical configuration of the processing chain. New monitoring or diagnostic activities can easily be inserted into the flow of the event treatment.

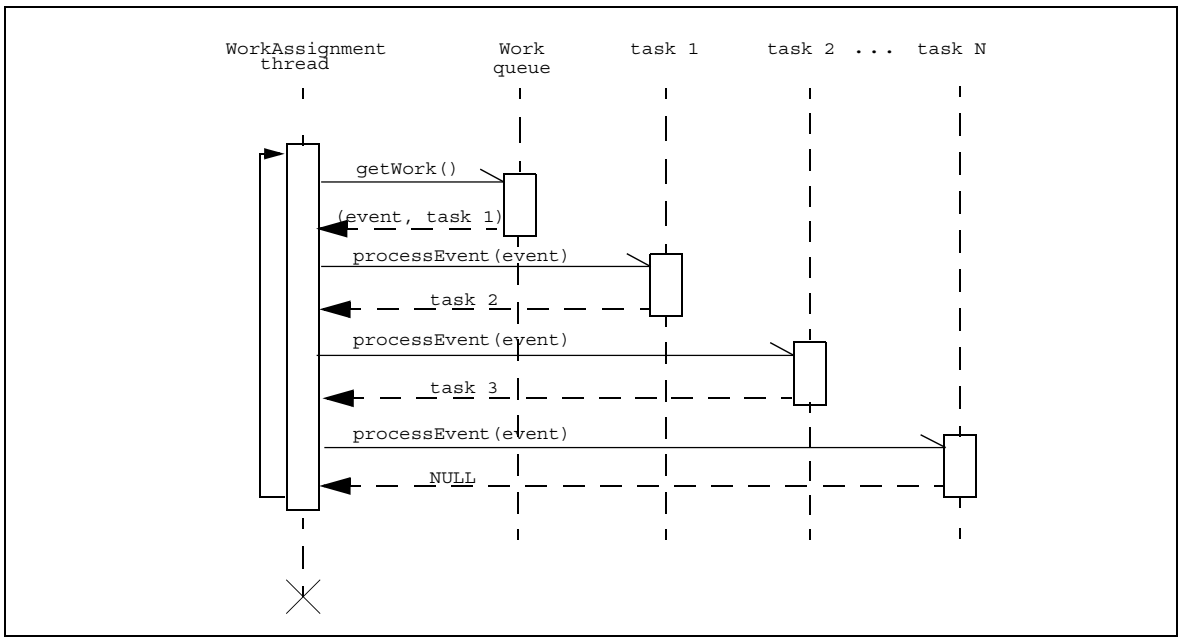

Figure 9-10 Sequence diagram for the work distribution in the EFD

9.3.2.3 Processing Task

Processing Tasks run on every processing node as independent processes. They use the offline framework Athena to run the Event Selection Software for the strategy described in Chapter 4. The sequence diagram shown in Figure 9-11 shows the synchronization mechanism between the Event Selection Software and the framework provided by the Event Filter PT.

Event passing between PT and EFD is done via the shared memory described in the previous section. Synchronization makes use of UNIX sockets. After having connected to the EFD process, the PT can request an event to the ‘External PT Task’. It receives a pointer to a read-only region of shared memory. When processing is completed, PT returns an ‘EF decision’ to the External PT Task in EFD as a string. This EF decision is then used to decide which step will be executed next in the processing chain (event sent to permanent storage, deleted, used for monitoring purposes, etc.). PT can request a writeable block in shared memory, where it can store additional information produced during processing. If the event is to be sent to permanent storage, EFD will append this data to the raw event.

Inside the PT, communication with EFD is done via the Athena standard ROB Data Provider Service. The service uses the interface of the `Byte Stream Input Service`, which handles reading of byte stream data. The converse process is handled by the `Byte Stream Output Service`. Concrete implementations are provided for input and output from and to the Data Flow system. Access to the event in the shared memory makes use of the standard Event Format Library [9-15]. The `Byte Stream Output Service` serializes the information produced

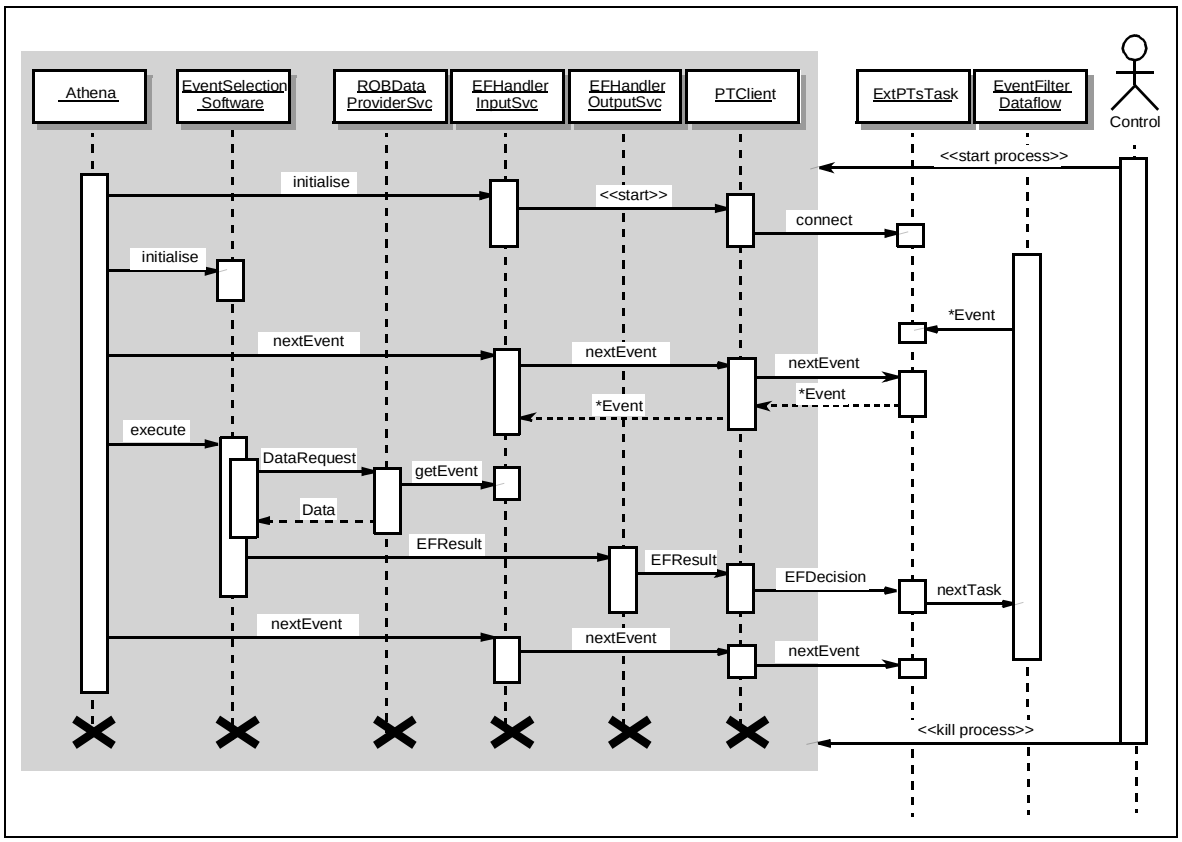

Figure 9-11 Sequence diagram for the interaction between the ESS and the EF PT. The shaded area indicates the parts running in the PT.

by the PT in the shared memory and an EF sub-detector fragment is created following the standard sub-detector structure in the event format for the byte stream data.

9.3.2.3.1 Interfaces with Event Selection Software

In Figure 9-12 a package diagram is shown for the ESS running in the EF Processing Task. The EFD has an interface to the SFI and SFO. The pointer to the event is transmitted to the Processing Task running Athena, which performs the EF selection for the event. No equivalent of the ROB Data Collector is needed by the processing task, because it is carried out after the event building. Instead, the input and output services allow access to the full event in shared memory. The dependencies on external systems and subsystems are hidden from the ESS. It is foreseen to run a single EF selection per Processing Task. The ESS has dependencies on interfaces to the Meta Data and Monitoring Services. The use of Athena as a common framework allows for the same or similar interface definitions running offline or online.

9.3.2.4 Detailed EF Validation tests

Extensive EF validation tests have been performed on PC-based clusters, in Pavia (the GRID-PV cluster) and at CERN (the MAGNI cluster). Details of these tests are given in [9-16]. The key results relating to data flow are described below. Test results dealing with EF supervision are covered in Chapter 12, whilst tests relating to the ESS are presented in Chapter 14.

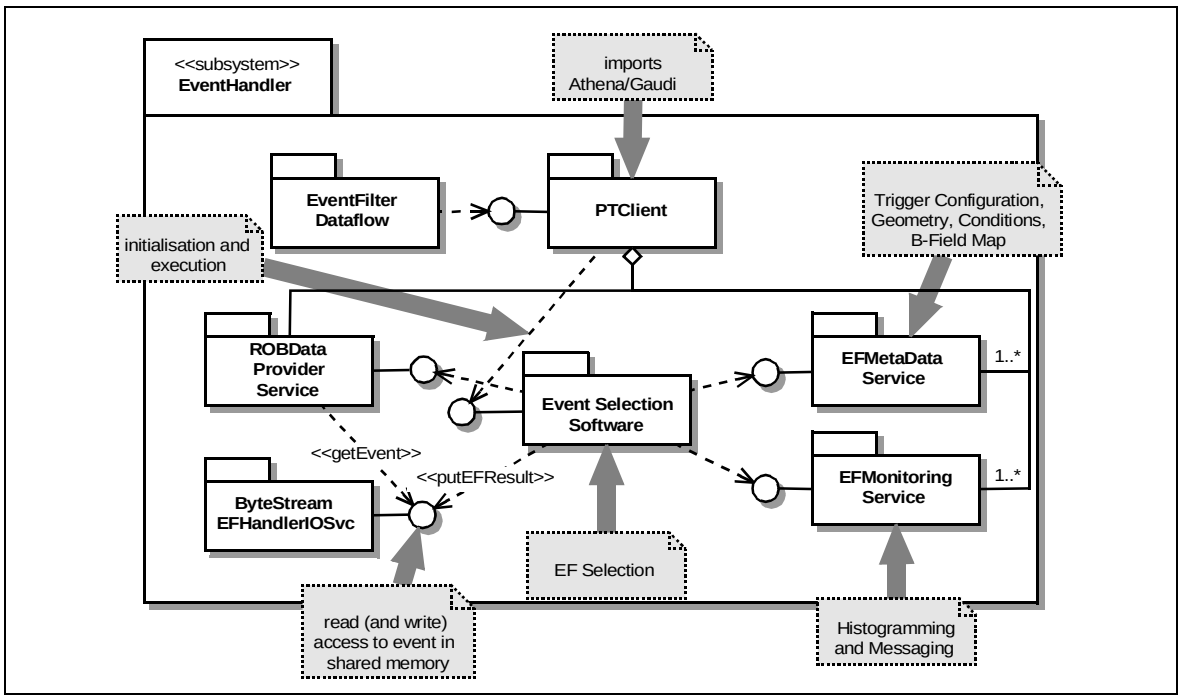

Figure 9-12 Package diagram showing the dependencies of the Event Selection Software performing the EF selection in the Processing Task

9.3.2.4.1 EF data flow stand-alone performance

Two lightweight EFD and PT processes have been used to measure the overhead time of the communication between EFD and PT. The EFD process used contained a single External PT task to provide the PT with dummy data. The PT performed the following sequence of actions: request an event; map the event in the shared memory; send the dummy EF answer; unmap the event. The measured time to perform this sequence is 80 µs on a dual processor Xeon at 2.2 GHz. It does not depend on the size of the event. This overhead is negligible compared to the nominal average processing time (of the order of 1 s).

9.3.2.4.2 EF data flow communication with the main DataFlow

To test the data throughput of the EFD a configuration was used with two external interfaces: an SFI providing dummy events of various sizes and an SFO receiving events (which it discarded immediately). Some tests were run with all processes running on a single machine, other tests had the SFI and SFO running on separate machines. Several values were used for the acceptance of the PT: 0% to give a measure of the SFI to EFD communication; 10%, the expected acceptance of the final system; 100%, the figure used for the EF in the test beam environment, which represents the maximum load for the system.

The tests were performed in Pavia on a cluster of dual Xeon (2.2 GHz) machines connected by Gigabit Ethernet and at CERN on a cluster of dual Pentium 3 (600 MHz) machines connected by Fast Ethernet. The results are summarized in Figure 9-13 and Figure 9-14. For an event size of 1.6 Mbyte the results when running the processes on a single machine of the Pavia cluster show that the combined processing time is ~5 ms per event at 0% acceptance and just over twice this at 100%. Clearly, even without correcting for the time taken by the SFI and SFO, this is small compared to the expected average EF processing time. On the slower MAGNI cluster these

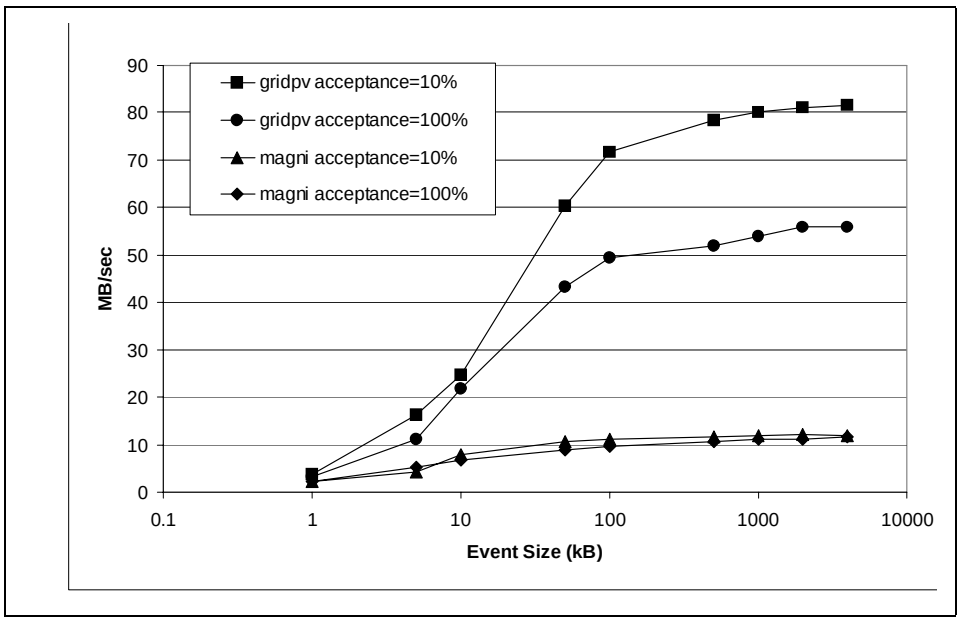

Figure 9-13 EFD throughput measurements (with processes running on different machines)

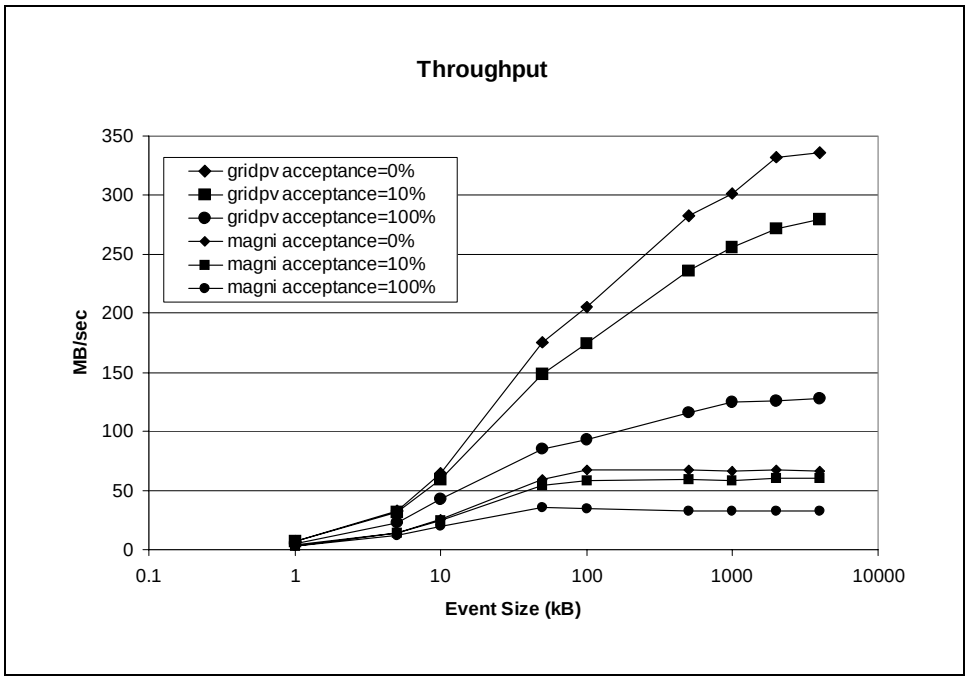

Figure 9-14 EFD throughput measurements (with processes running on the same machine)

times become some four times longer, consistent with the processor speed. When the processes are run on a separate machine the throughput is limited by the network links between the machines and reduces several fold. Nevertheless, even with Fast Ethernet the throughput is entirely adequate and with Gigabit Ethernet the available bandwidth is well in excess of that required.

9.3.2.4.3 Robustness tests

To test the robustness of the EFD a configuration of an SFI, an SFO and the EFD, each on different machines of a 3-node cluster, was used. The processing node also included 4 PTs. The system was run for 10 days without problems and handled $\sim 3 \times 10^9$ dummy events. The test included randomly stopping and starting some of the PTs, which the EFD handled correctly, stopping and resuming delivery of events to the corresponding PT as required.

9.4 HLT operation

9.4.1 Common aspects

The operational analysis of the whole HLT has been described in a dedicated document [9-17] which describes in detail the expected functionality and gives uses cases for the operation. Use cases include start-up, run control, shutdown, steady running, and error conditions.

The HLT Supervision system is responsible for all aspects of software task management and control in the HLT. Mandates of the supervision system include configuration of the HLT software processes, synchronizing the HLT processes with data-taking activities in the rest of the experiment, monitoring the status of HLT processes e.g. checking that they are running, and restarting crashed processes. The Supervision system must provide a user interface for the crew on shift. The interface must be as user friendly as possible while providing the tools for expert work during both the commissioning and steady operation phases. Both the LVL2 and the EF are implemented as hundreds of software processes running on large processor farms, split for reasons of practicality into a number of sub-farms. In view of this the supervision requirements for the two systems are very similar and an integrated HLT supervision system has been developed. It has been implemented using services provided by the Online Software (see Chapter 10).

In addition to standard requirements on robustness and scalability, the design of the Supervision system must be sufficiently flexible to cope with future evolution during the lifetime of the experiment, especially when new hardware and software is used.

The Online Software configuration database is used to describe the HLT in terms of the software processes and hardware (processing nodes) of which it is comprised. The HLT supervision and control system uses information stored in the Online Software configuration database to determine which processes need to be started on which hardware and subsequently monitored and controlled. The smallest set of HLT elements, which can be configured and controlled independently from the rest of the TDAQ system, is the sub-farm. This allows sub-farms to be dynamically included / excluded from partitions during data-taking.

Synchronization with the rest of the TDAQ system is achieved using the Online Software run control system. Each sub-farm has a local run controller, which will interface to the Online Software run control via a farm controller. The controller that collaborates with a sub-farm supervisor, which provides process management and monitoring facilities within the sub-farm. The controller and supervisor cooperate to maintain the sub-farm in the best achievable state by keeping each other informed about changes in supervision or run-control state and by taking appropriate actions, e.g. restarting crashed processes. Where possible, errors should be handled

internally within the HLT processes. Only when they cannot be handled internally should errors be sent to the supervision and control system for further consideration.

The overall supervision of LVL2 and EF is integrated in the HLT Supervision system described in Section 12.3.2.

Software will also be required for farm management, i.e. hardware monitoring, operating system maintenance, code distribution on many different nodes, etc. However, given the commonalities with the requirements of many other farms and the increasing availability of such software elsewhere, it is planned to select the software to be used for this and how it is interfaced to the Supervision system at a later date.

9.4.2 LVL2 operation

The configuration and control of the LVL2 applications in the LVL2 processors are handled by standard Data Collection controllers. The Run Control hierarchy consists of a root top-level controller and one child controller per LVL2 sub-farm. It is convenient to configure the LVL2 Supervisors (which control the flow of events through the LVL2 farm, see Section 9.2.3) so that each LVL2 sub-farm is associated with just one LVL2 Supervisor — although a single Supervisor may send events to several sub-farms. Monitoring of the applications is performed in a parallel tree structure again with one monitor process per sub-farm. In contrast, however, for the Information Service a single IS server is used for all of the LVL2 processors.

9.4.3 EF operation

A detailed list of EF requirements can be found in [9-18], many are common with LVL2 as listed above, however, some derive from remote EF sub-farms and the possibility of event monitoring and calibration tasks.

The EF Supervision also uses a tree-like structure following the baseline architecture described in Section 9.3.1. The Run Control hierarchy consists of a root top-level controller and one child controller per sub-farm. All ancillary duties related to process management are performed by a supervisor server, local to each sub-farm. A local Information Sharing server allows the exchange of information with other sub-systems.

9.4.3.1 Scalability tests

Scalability and performance tests, in the context of EF Supervision, have been performed on the ASGARD cluster at ETH Zurich and on CERN-IT clusters. Detailed results are given in [9-19] and [9-20], respectively. On ASGARD, the supervision system had been implemented with JAVA Mobile Agents technology. It has been possible to control successfully 500 processors. However, the tested product, which was freeware, is now licensed and has therefore been discarded.

An implementation of the Supervision has been made using the tools provided by the Online Software group. This implementation has proved to be able to control some 1000 processors (running on 250 quad-board machines from the CERN-IT cluster). The execution times for the Run Control transitions do not depend strongly on the number of controlled nodes and are less than 3 s for configurations of a size varying between 50 and 250 nodes (see Section 10.6.2).

9.5 Event Selection Software (ESS)

The tasks of the Event Selection Software are ‘event selection’ and ‘event classification’. Abstract objects representing candidates such as electrons, jets, muons, and $J/\psi \rightarrow e^+e^-$ are reconstructed from event data by using a particular set of HLT algorithms and applying appropriate cuts. An event is selected if the reconstructed objects satisfy at least one of the physics signatures given in the Trigger Menu. In both LVL2 and the EF events are rejected if they do not pass any of the specified selection criteria, which are designed to meet the signal efficiency and rate reduction targets of the trigger. From a physics event selection point of view there is no precise boundary between LVL2 and EF. Indeed, flexibility in setting the boundary is important to take advantage of the complementary features of these trigger stages.

The ESS comprises an infrastructure and the selection algorithms. The latter are to be provided either by the PESA group or, in the case of the algorithms for the EF, by the offline reconstruction group. Major parts of the trigger reconstruction will have to be based on offline reconstruction algorithms. This is an important constraint for the design of the ESS.

In the HLT system the ESS will run in the software environments provided by the L2PU and by the PT of the EF, as shown in Figure 9-7 and Figure 9-12, respectively. Hence the ESS needs to comply with the online requirements such as thread safety, online system requirements and services [9-6], as well as the online performance goals.

It is highly desirable though that the ESS is also able to run directly in the offline environment Athena [9-7] to facilitate development of algorithms, to study the boundary between LVL2 and EF, and to allow performance studies for physics analysis. Therefore the ESS needs to comply with the control framework and services that are provided by the offline software architecture. For this reason the Athena framework was chosen as the framework to run the ESS inside the EF PT and in the modified form provided by the PESA Steering Controller inside the L2PU (see Section 9.2.4.3 and Section 9.3.2.3 above).

In the offline, the task of the ESS is to emulate the full online selection chain. Hence three so-called top-level Athena algorithms are required, namely the emulation of the LVL1 trigger and two instances of the Event Selection Software. The LVL1 trigger emulation provides the LVL1 Result. The first instance of the Event Selection Software is configured to execute the LVL2 seeded by the LVL1 Result, the second to execute the EF selection seeded by the LVL2 Result. The details of the prototype implementation of the physics selection chain are given in Chapter 13.

9.5.1 Overview

The design and implementation of the Event Selection Software is based on the requirements and use cases documented in [9-21]. The current prototype implementation follows the analysis and conceptual design discussed in [9-22]. In the following an overview of the subsystem is given and the design and implementation of the basic concepts are discussed.

The ESS is subdivided into four sub-packages, as listed below. These are shown in Figure 9-15 with the most important external software dependencies.

- The **Steering** controls the selection software. It organizes the HLT algorithm processing in the correct order, so that the required data is produced and the trigger decision is obtained. The Steering implements the interface to the PESA steering controller (when run-

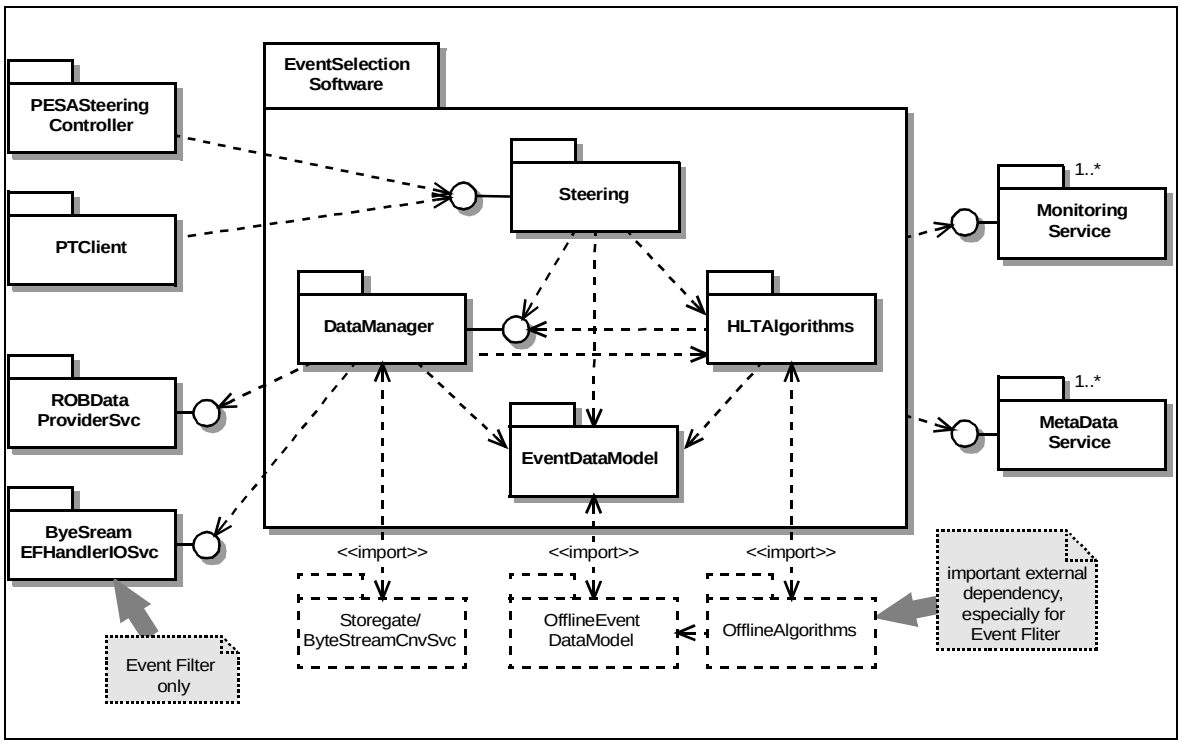

Figure 9-15 A package diagram of the Event Selection Software. Also shown are the dependencies of the sub-packages. The dependencies on the offline Event Data Model and on the offline algorithms are explained in the text.

ning in the L2PU) and to the PT Client (when running in the EF). The same interfaces are used when running in the offline framework Athena.

- The event data is structured following the **Event Data Model** (EDM). The EDM covers all data entities in the event and the relationships between them. The data entities span from the raw data in byte stream format (originating from the detector RODs), the LVL1 Result, and all other reconstruction entities up to the LVL2 and EF Results.
- The **HLT Algorithms** are used by the Steering to process the event and to obtain the data on the basis of which the trigger decision is taken.
- The **Data Manager** handles all event data during the trigger processing. The current implementation is based on Athena Storegate [9-23] to provide the necessary infrastructure for the EDM. The offline Byte Stream Conversion Service is used to implement the HLT algorithm access to the raw data. Different implementations of the ROB Data Provider Service are used for the LVL2, EF, and offline to access the ROB Fragments.

In summary, the EDM covers all event data entities and is used by the Steering, the HLT algorithms and the Data Manager to communicate information about the event. The HLT algorithms build up the event tree in the process of the reconstruction. The result is analysed by the Steering to obtain the trigger decision. The Data Manager supports the EDM. It provides the means of accessing the event data and for developing it as the event is processed. The data access patterns reflect the needs of the HLT algorithms and of the Steering, and the constraints of the online systems. Raw data access by 'Region' is a requirement, especially for the LVL2. In the following subsections more details are given of the Event Selection Software sub-packages.

9.5.2 The Event Data Model sub-package

The LVL2 and the EF selection are implemented as software triggers that select events by means of reconstruction, guided by the RoI information provided by the LVL1 system. Therefore the organization of the data classes and of the object relations are fundamental. The EDM of the Event Selection Software is closely coupled to the offline EDM, especially because offline algorithms are the basis of the EF selection. The EDM is therefore being developed in close contact with the offline EDM, detector, and reconstruction groups. Logically the EDM classes are grouped into five sub-packages:

- **Raw Data** coming from the ROS and the trigger systems are in byte stream format. This includes the LVL1 Result, the LVL2 and EF Results, as well as ROB Data from the sub-detectors and from the LVL1 system. Note that for a given sub-detector several Raw Data formats might be used, e.g. different formats depending on the luminosity. This includes different data content or compression schemes.
- The **Raw Data Objects (RDOs)** are an object representation of the raw data from the different sub-detectors. In the trigger they are used only in the EF where they are created at input to the reconstruction chain. In LVL2, however, creating these objects poses too much overhead and thus in LVL2 the Raw Data is converted directly to Reconstruction Input Objects (RIOs) described below.
- **Features** are all types of reconstruction data derived from the RDOs or from other features, with increasing levels of abstraction. This includes all offline reconstruction EDM classes. They range from relatively simple RIOs (e.g. calibrated calorimeter cells and SCT clusters) up to reconstructed quantities such as tracks, vertices, electrons, or jets.

  A detailed description of the implementation of the Raw Data formats, RDOs, and features for the current prototype are given in Section 13.3 and the associated references.
- **MC Truth** information. Together with the first three sub-packages of the EDM, the Monte Carlo truth is common to the Event Selection Software and the offline reconstruction. It is needed primarily for debugging and performance studies.
- **Trigger Related Data** comprising RoI Objects, the LVL1(LVL2/EF) Trigger Type [9-24], Trigger Elements (TEs) and Signatures. A TE labels a set of Features and associates them with a physical interpretation, such as a particle, missing energy or a jet. TEs are the objects used by the Steering to guide the processing and to extract the trigger decision.

An important aspect of the EDM is the ability to navigate between different objects in the event using the relations between them. This is used during the HLT algorithm processing to analyse and develop the knowledge of the event. Examples of instances of EDM classes and their relations are discussed in Section 9.5.3.1 in the context of the seeding mechanism.

9.5.3 The HLT algorithms sub-package

The task of the HLT algorithms is to analyse the Raw Data and to reconstruct parts of the event according to the guidance from LVL1. This reconstructed data is used by the Steering to derive the trigger decision. The LVL1 RoI-based approach implies a data-driven event reconstruction. Any HLT algorithm in a reconstruction sequence may be executed several times per event, once for each RoI. Therefore a modular architecture of the reconstruction code is necessary. The HLT algorithms are structured into three parts:

- **Data Preparation** comprises algorithmic code to convert the Raw Data into objects that are used as input to reconstruction, include format changes and calibration corrections. This task involves sub-detector-specific information, and hence sub-detector groups are responsible for the implementation and maintenance of the code, taking the HLT requirements into account. In the current design the organization of the Data Preparation is different for LVL2 and the EF. The EF follows closely the offline reconstruction chain, where the Raw Data is first converted into RDOs and these are subsequently processed to obtain SCT Clusters or Calorimeter Cells. In LVL2 the Data Preparation goes in one step from Raw Data to RIOs, thus avoiding the overhead of creating the RDOs.

  The boundary between Data Preparation and subsequent Feature Extraction is not exact, but only Data Preparation algorithms use the Raw Data or Raw Data Objects.

- **Feature Extraction** algorithms operate on abstract Features and Trigger Related Data to refine the event information. Two types of algorithms are distinguished in the Feature extraction sub-package. **Reconstruction Algorithms** process Features and produce new types of Features, just like offline reconstruction algorithms. However, in the trigger, RoI Objects are used to restrict the processing to geometrical regions of the detector. To do this TEs are used by the Steering to 'seed' the reconstruction algorithms. The seeding mechanism is discussed in the next subsection. The second type of algorithms are **Hypothesis Algorithms**. Their task is similar to particle identification. A hypothesis algorithm validates the physics interpretation implied by the label of the TE based on the reconstructed Features. An example is the validation of an 'electron' matching a reconstructed Calorimeter Cluster and a Track.

- A library of **Algorithm Tools** to carry out common tasks such as track-fitting or vertex-finding.

Overviews of Reconstruction Algorithms and Algorithm Tools implemented in the current ESS prototype are given in Section 13.3.3 and Section 13.3.4.

9.5.3.1 The seeding mechanism

Logically the trigger processing starts from a LVL1 RoI using predefined sequences of HLT algorithms. A so-called 'Seeding mechanism' is used to guide the reconstruction to those parts of the event relevant for preparing the trigger decision.

Figure 9-16 shows an example of the evolution of the tree of objects associated to one LVL1 RoI at different stages of the trigger processing. This example starts with an electromagnetic RoI Object from the LVL1 calorimeter trigger. In the first step shown in Figure 9-16(a) an input 'EM' TE that uses the RoI Object is created for the Steering.

Starting from the input TE the first HLT reconstruction step attempts to validate the physics hypothesis of there being an electromagnetic cluster above a certain threshold. For this hypothesis the Steering creates an output TE with the corresponding label. The output TE is linked to the input TE by a 'seeded by' relation. The Steering then executes the cluster-finding HLT algorithm, giving it the output TE as an argument, to validate this output TE. The algorithm navigates via the 'seeded by' and 'uses' relations from the output TE to the input TE and then to the 'LVL1EM' RoI Object to access information about the LVL1 RoI. The algorithm obtains η and ϕ from the RoI and does its pattern recognition work. In the example it creates a Calorimeter Cluster from a set of Calorimeter Cells. These objects are linked to the output TE to record the event information associated with this RoI for later processing. Based on the reconstructed

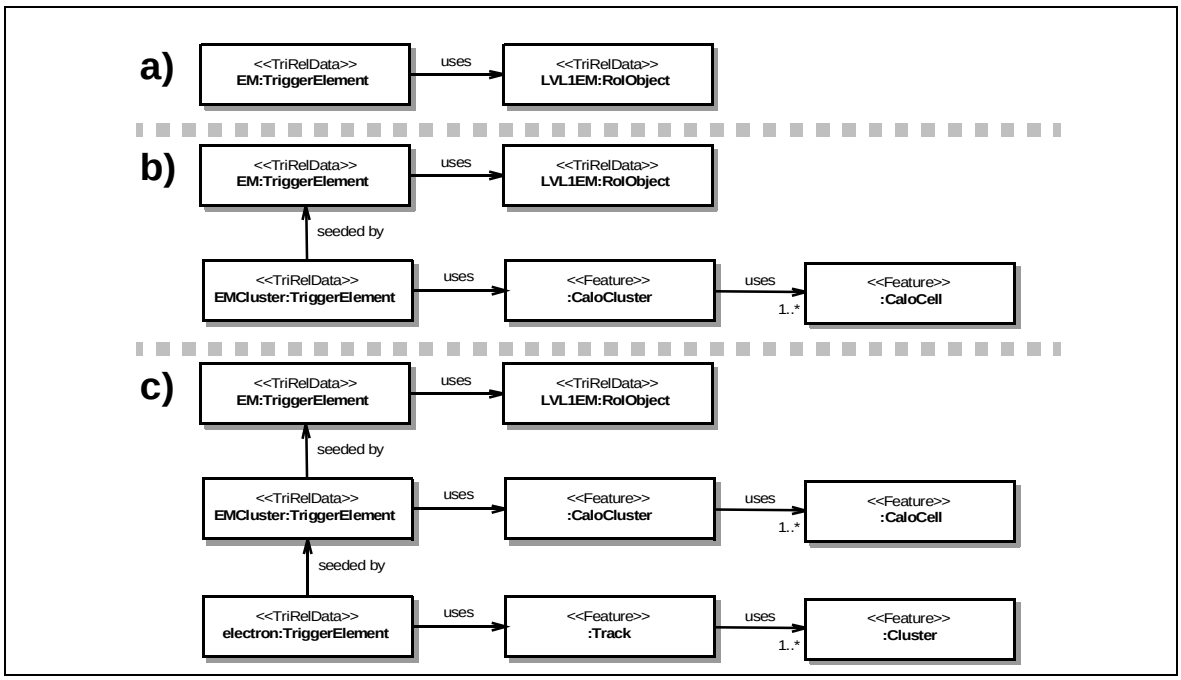

Figure 9-16 Three diagrams showing fragments of the object tree associated to one RoI at different stages of trigger processing

Cluster the algorithm validates the hypothesis if all cuts are passed. The algorithm transmits the result to the steering, by ‘activating’ the output TE in case of a positive decision. The Steering ignores all inactive TEs for further processing. The object tree at the end of the algorithm execution is shown in Figure 9-16(b).

The next algorithm in the example looks for an electron track. It is seeded with a new output TE labelled ‘electron’. The algorithm navigates the full tree of data objects to access the necessary information. To validate the ‘electron’ physics hypothesis it could use the precise position of the calorimeter cluster to reconstruct a Track from a set of SCT Clusters. If a Track is found the ‘electron’ hypothesis is validated and a TE is activated. The resulting object tree is shown in Figure 9-16(c).

The seeding mechanism needs to be general enough to cover all physics use cases. In Figure 9-17 a similar diagram to Figure 9-16(c) is shown for the case of a LVL2 B-physics trigger. Here, the new aspect is the inner detector full scan. In this use case once the initial ‘muon’ has been validated a scan is made of the inner detector to reconstruct all tracks in the event. A TE ‘uses’ the collection of Tracks and seeds, for example, an algorithm that reconstructs an exclusive B-decay into two pions. The result of this would be a TE called ‘B to PiPi’. This TE uses a B-meson and is ‘seeded by’ the ‘Inner Detector Scan’ TE, as shown in the figure.

9.5.4 The steering sub-package

The **Steering** controls the HLT selection. It ‘arranges’ the HLT algorithm processing for the event analysis in the correct order, so that the required data is produced and the trigger decision is obtained. The Steering provides the interface to the PESA Steering Controller for running in the L2PU and to the PT Client for running in the EF Processing Task (Figure 9-15). In the offline the Steering is a top-level Athena algorithm executed directly by the Athena event loop manager.

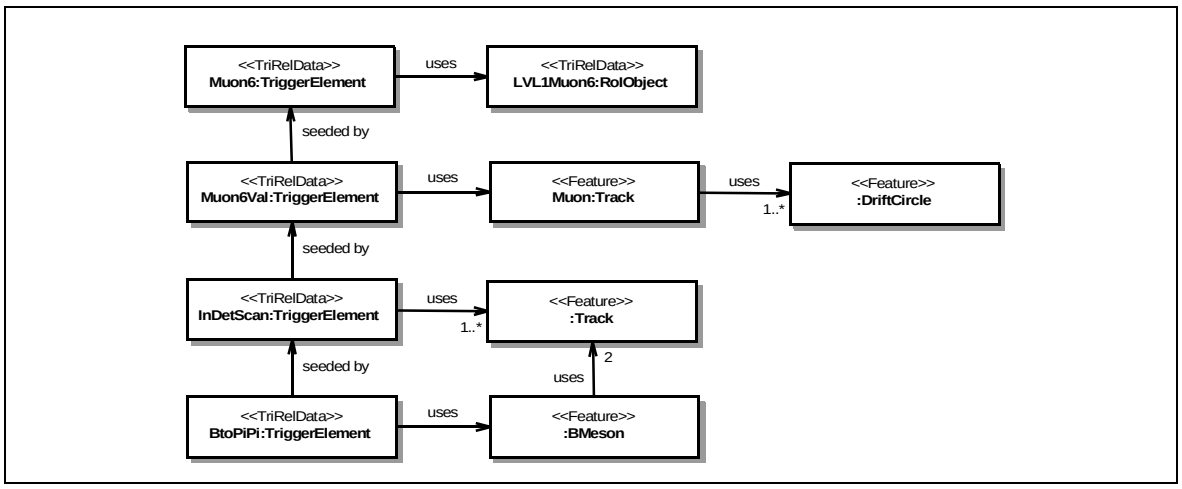

Figure 9-17 A diagram showing a fragment of the event for the case of LVL2 B-Physics trigger reconstruction

The LVL2 and the EF selection is data driven, in that it builds on the result of the preceding trigger level. The Steering has to guide the HLT algorithm processing to the physics-relevant aspects of the event. The Steering uses the seeding mechanism discussed in the previous section to restrict the reconstruction to the parts of the event corresponding to a given LVL1 RoI. Seen from the side of the Steering, the reconstruction of an event in the trigger is a process of refining TEs, as was shown in Figure 9-16 and Figure 9-17.

The ESS has to implement the physics selection strategy discussed in Chapter 4. The HLT selection demands that an event matches at least one physics ‘Signature’. Each Signature is a combination of abstract physical objects like ‘electrons’, ‘muons’, ‘jets’, or ‘missing energy’. It is usually requested that, for example, an electron has a minimal energy and is isolated from a jet. Translated into the ESS a Signature is simply a combination of required Trigger Elements with labels like ‘e25i’. An example for such a menu of Signatures is given in Table 4-1.

Many of the Signatures discussed in Chapter 4 require more than one TE. In this case it is beneficial not to complete the validation of the TEs in turn, as a later one may fail at an early stage of the validation. Thus the Steering arranges the HLT algorithm processing in steps. At each step a part of the reconstruction chain is carried out, starting with the HLT algorithm giving the biggest rejection. At the end of each step a decision is taken, whether the event can still possibly satisfy the TE combination required in the Signature. This concept of ‘step processing’ ensures the earliest possible rejection of events.

9.5.4.1 Implementation of the steering

In Figure 9-18 a class diagram for the Steering sub-package is given. The **Step Controller** inherits from the Athena Algorithm and is the interface to the PESA Steering Controller and the PT Client. It provides the necessary methods to configure the ESS at the beginning of a ‘RUN’, to execute the LVL2 or EF selection on an event, and to end a ‘RUN’.

The task of the **Trigger Configuration** is to provide the Sequence and Menu Tables for the step processing. This task is carried out during the initialization phase (before the start of a ‘RUN’), which is especially important for LVL2.

In LVL2 the Step Controller uses the **LVL1 Conversion** for each event to convert the LVL1 Result (i.e. Raw Data) into RoI Objects and prepares the trigger selection by creating the TEs need-

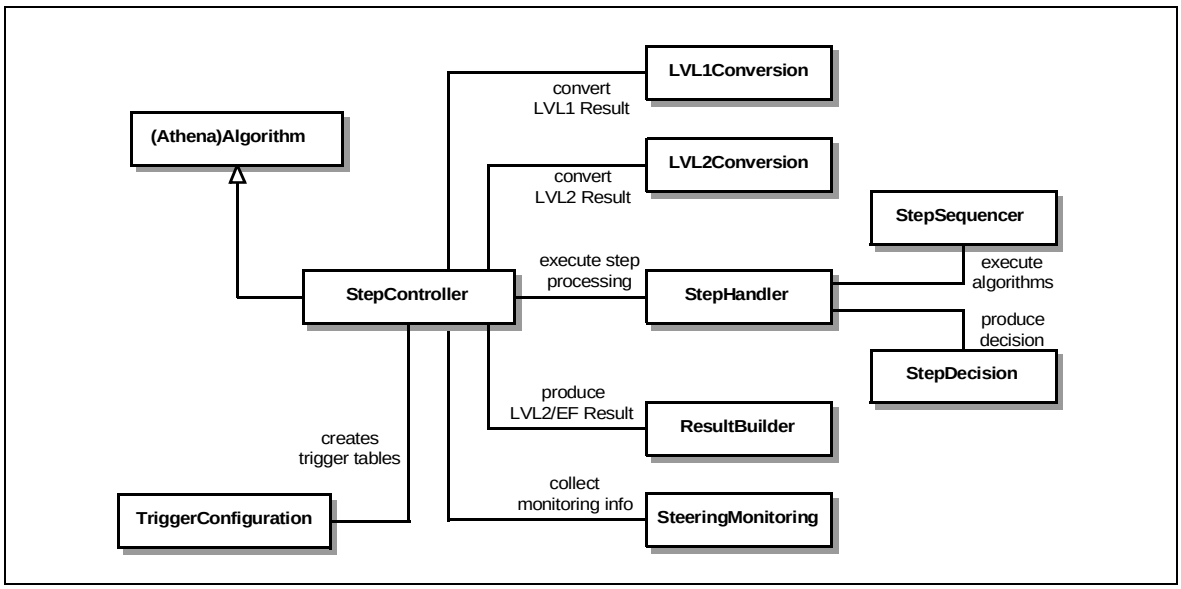

Figure 9-18 A class diagram for the Steering sub-package

ed for the seeding mechanism, as was shown in Figure 9-16(a). In the EF the corresponding **LVL2 Conversion** translates the LVL2 Result to prepare the EF selection.

The trigger processing of each event is performed by the **Step Handler**. For each step it uses the **Step Sequencer** to execute the HLT algorithm from the corresponding Sequences. The Step Handler executes the **Step Decision** to compare the result of the algorithmic processing, given in terms of TEs, with the Signatures to decide whether or not to reject the event. The Step Handler continues to process subsequent steps until the event is rejected or all steps have been executed. The **Result Builder** produces the LVL2 or EF Results, depending on which HLT subsystem the Event Selection Software is running.

The Step Controller uses the **Steering Monitoring** at the end of the event processing to collect summary information for monitoring purposes.

9.5.4.2 The Trigger Configuration

At initialization time the **Trigger Configuration** [9-25],[9-26] configures the ESS to execute the LVL2 or EF selection, depending on the subsystem the software is running in. The ATLAS trigger selection strategy is defined in terms of physics **Signatures** as given in Table 4-1. The Trigger Configuration derives the configuration of the selection from this list of physics Signatures and from the list of available HLT algorithms that are implemented. A recursive algorithm is used that computes the full EF and LVL2 configuration in a top-down approach starting from the final Signatures. In this way a consistent configuration of both HLT selection levels is ensured which logically connects LVL2 and EF. The configuration scheme will be extended in the future to also cover the LVL1 configuration.

Technically the input to the Trigger Configuration are two XML files that are parsed [9-27] into C++ objects. The first XML file contains the list of final physics Signatures in terms of TE labels, as shown in Table 4-1. The second XML file contains a list of available Sequences of HLT algorithms that can be used by the trigger. Each Sequence specifies the required input in terms of TEs, the HLT algorithms to be executed for these seeds, and the hypothesis in terms of an output TE to be validated.

The recursive algorithm to calculate the full configuration is illustrated in Figure 9-19 using as an example a '2-electron' Signature. The final physics Signature '2 × e20i' requires two constituent TEs 'e20i'. Such a TE is the output of a Sequence, which is found in the Sequence list. In the example the input TE of the matching Sequence is 'e20' and the Sequence contains an isolation algorithm. Hence, in order to satisfy the final Signature '2 × e20i' requires two 'e20' TEs in the previous step or in other words, the derived Signature of '2 × e20'. The recursion is continued until the LVL1 input TEs are reached [corresponding to the RoI Objects as shown in Figure 9-16(a)]. In the example from Figure 9-19 this corresponds to four recursion steps.

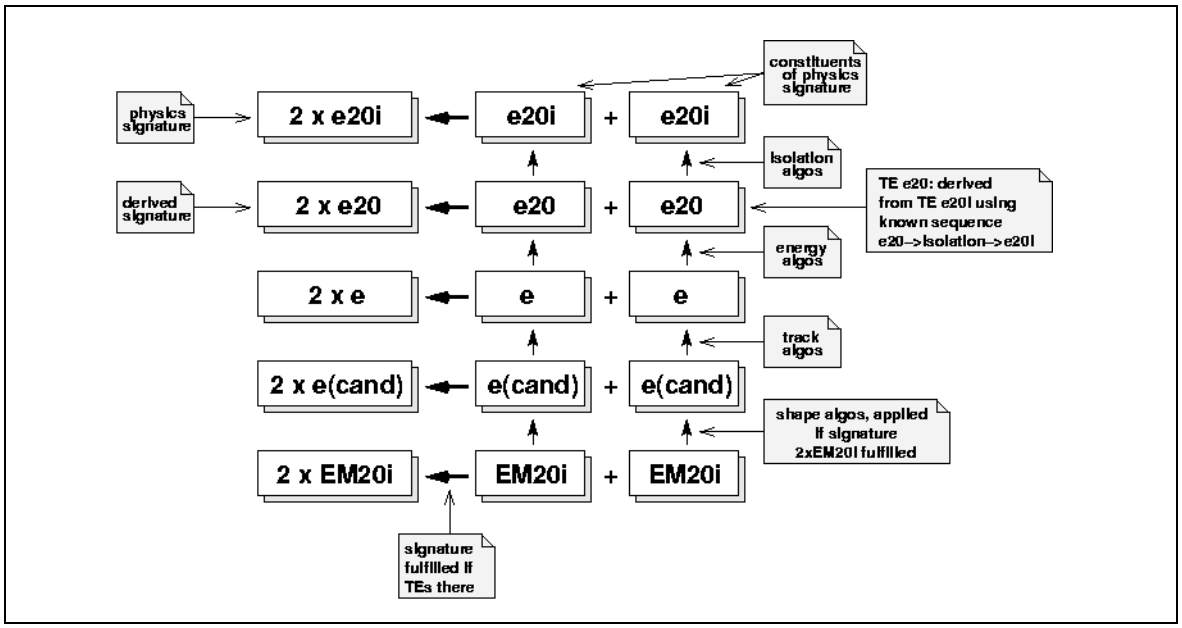

Figure 9-19 Illustration of the recursive configuration of the selection for a '2-electron' trigger

The calculation is done for all final Signature in the full trigger menu and the resulting tables are combined for each trigger step. Different Signatures may involve the same Algorithm or Sequence in a given step. Hence the tables are processed to remove double entries. The boundary between LVL2 and EF is defined by associating Sequences in the input list to either of the two trigger levels. This association is the basis for separating the EF and LVL2 in the output of the Trigger Configuration. Prescaling and forced accept rates are allowed for at the level of the Signatures.

The output is given in the form of pairs of **Sequence Tables** and **Menu Tables** for each trigger step. The class diagram is shown in Figure 9-20. The tables contain the necessary information in terms of TEs and HLT algorithms for the Steering to control the data-driven trigger processing as discussed in the following subsections.

9.5.4.3 The LVL1 conversion

At the start of the LVL2 event processing the LVL1 Result is converted into a form that is suitable for steering the step processing. The Step Controller calls the LVL1 Conversion to decode the LVL1 Result fragment (the output of the RoIB, see Section 13.2). In the fragment the RoI information is encoded into 32-bit words as defined in [9-24]. These words contain bit patterns identifying, for example, the electronics channel from which the RoI came and the threshold passed. This information is uniquely related to the location of the RoI in $\eta-\phi$ space and to the threshold

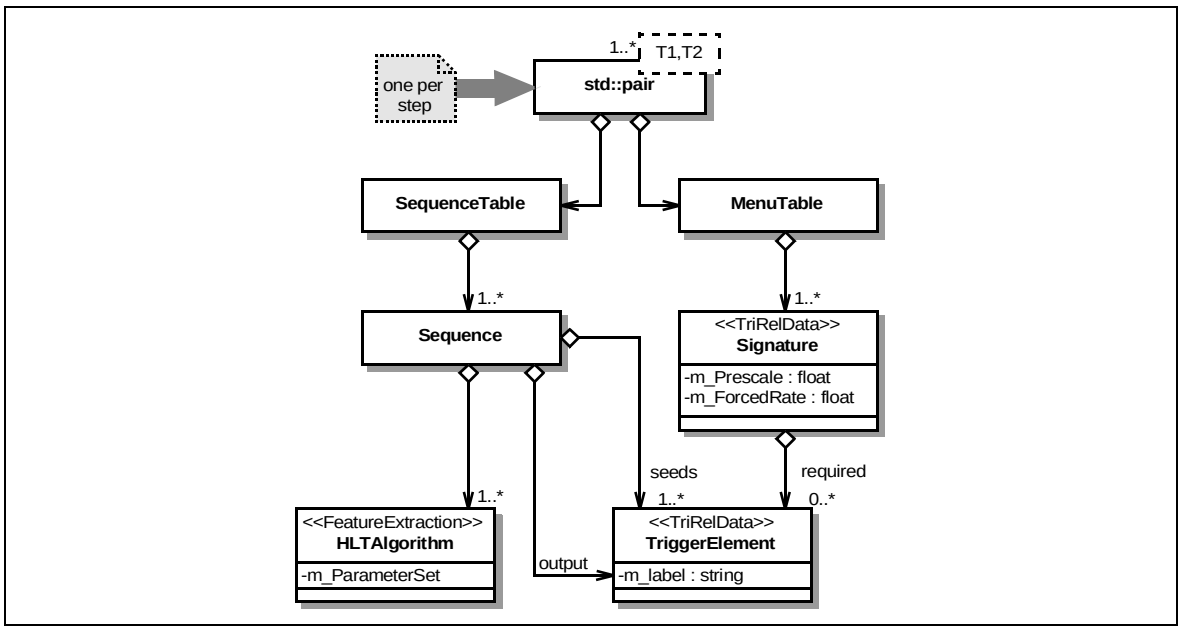

Figure 9-20 A class diagram for output of the Trigger Configuration, showing for each step a pair of a Sequence Table and a Menu Table

value in GeV. To obtain the values the configuration of the LVL1 and subdetector specific mapping information must be used. The RoI Objects are then stored for further processing and for each RoI Object a TE of the corresponding label is created to enable the seeding mechanism as shown in Figure 9-16(a).

9.5.4.4 The Step Handler

After the LVL1 Conversion (or in the EF the LVL2 Conversion) the Step Controller executes the Step Handler. In Figure 9-21 a sequence diagram is shown for the step-by-step processing of an event by the Step Handler. Each step is configured by a pair of a Sequence Table and a Menu Table, as provided from the Trigger Configuration.

The Step Handler first executes the Step Sequencer, which controls the reconstruction part of each step. The Step Sequencer accesses the list of active input TEs from the previous step or the LVL1 Conversion for the first step. The list is compared to the required input TEs of each Sequence in the Sequence Table. For all matching combination of TEs the Step Controller executes the HLT algorithms in the corresponding Sequence. To implement the seeding mechanism discussed in Section 9.5.3.1 it is necessary to created the output TE and the ‘seeded by’ relation to the input TE and then pass the output TE as a seed to the HLT algorithm, as shown in Figure 9-21. The HLT algorithm has to validate the hypothesis of just this output TE. In the scheme currently implemented the HLT algorithm signals a validated hypothesis by ‘activating’ the output TE. It is important to note that for a given event the same algorithm Sequences can be executed multiple times, but on different seeds, depending on the number of matching input TE combinations.

At the end of each step the Step Decision is run to decide whether, based on the result of the Step Sequencer the event is still accepted. The Step Decision compares the list of active output TEs to the required TE combinations for the Signatures in the Menu Table of the step. For each matching TE combination the Step Decision creates a satisfied Signature object that ‘uses’ these

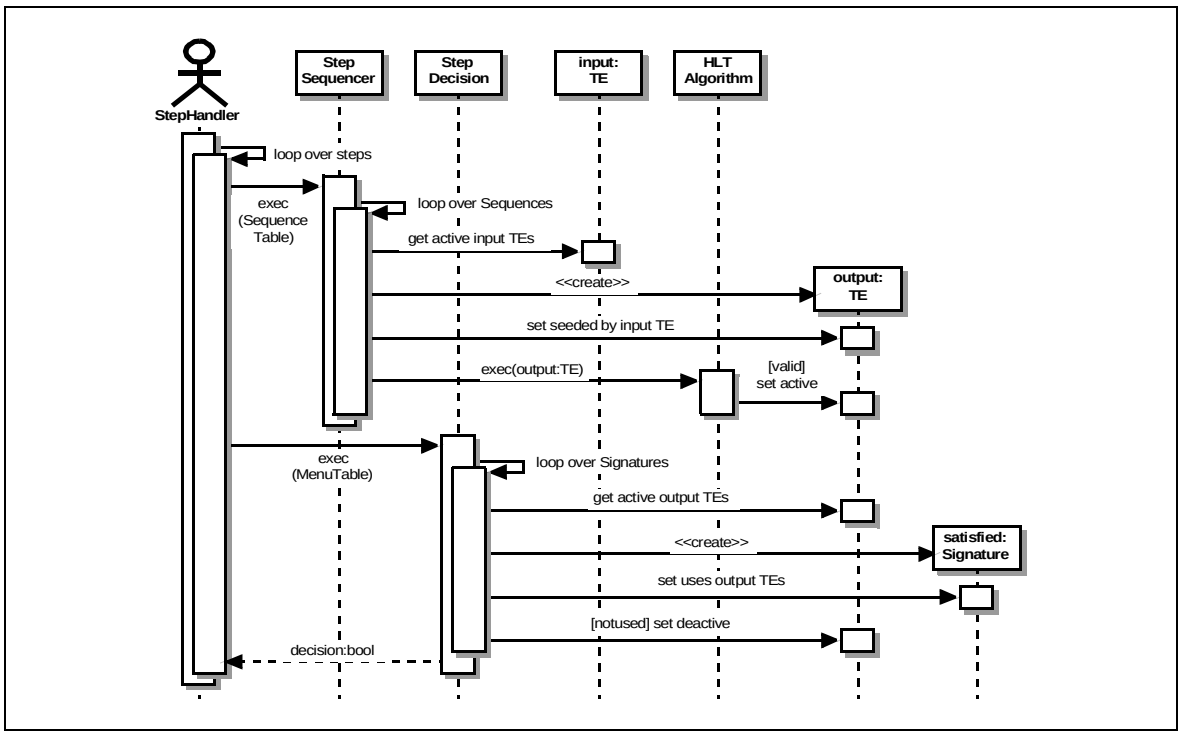

Figure 9-21 A sequence diagram for the Step Handler processing an event. For each step the Step Sequencer and the Step Decision are executed. No interaction with the store, etc., are shown.

TEs. In the next step only those TEs that were used to satisfy a least one Signature are used for further processing, all others are discarded from the event by deactivating them. An event is accepted for the step if at least one Signature was satisfied. The Step Handler continues processing the next step either until the event is rejected by the Step Decision or until all steps are done, in which case the event is finally selected.

9.5.4.5 The Result Builder and the LVL2 Conversion

At the end of the LVL2 processing of an event, if the event was selected, the Result Builder provides the information to seed the EF selection. In the current implementation this includes information about all satisfied Signatures, the associated TEs, and LVL1 RoIs. These are converted into a ROB fragment to be transferred via the pROS to the Event Builder. The structure is kept modular to accommodate a variable number of accepted items and to allow changes.

At the beginning of the EF processing the Step Controller executes the LVL2 Conversion that retrieves the LVL2 Result. It extracts the TE and RoI words to recreate the objects, including the navigational links among them. Thereby it recreates a structure similar to that shown in Figure 9-16(a) as output of the LVL1 Conversion at the beginning of LVL2. The differences are that the TEs are the active TEs of the final LVL2 step and the list of LVL1 RoIs is reduced to the ones ‘used’ in the final decision. After the LVL2 Conversion the EF selection is controlled (and seeded) by the Step Handler in the same way as for LVL2, but using more sophisticated HLT algorithms.

9.5.4.6 The Steering Monitoring

Monitoring of the Steering has several aspects, the debug output of the HLT algorithms, the timing, the rate monitoring, etc. Monitoring Services are used by the Event Selection Software to publish this information. The Steering Monitoring runs at the end of the event and analyses the event in memory.

9.5.5 The Data Manager sub-package

The Data Manager sub-package provides the infrastructure to receive, store, and later access data while processing the event. The transient data store, which is used to implement the Event Data Model, is thus the central part of this sub-package. The Data Manager also provides the means for the Event Selection Software to implement the seeding mechanism and to access Raw Data in restricted geometrical Regions ('retrieve by Region'). The Data Manager is the part of the ESS that is interfaced to the ROB Data Provider, as was shown in Figure 9-15 for the L2PU or for the EF Processing Task, respectively.

9.5.5.1 Implementation

Storegate [9-23] is the part of the Athena infrastructure that provides the transient data store. It was originally designed to handle data for the offline reconstruction and analysis. Storegate provides services for the client application, such as the functionality of a container with infrastructure for data storage, different patterns of data access, and memory management. It is also the part of Athena that supports persistency and implements the interface to the different I/O services. Using the Storegate interface for the implementation of the Data Manger facilitates the use of offline software in the trigger. Initial timing measurements [9-10] have shown that the performance of Storegate's core services satisfies the EF and also the LVL2 requirements. Thus Storegate has been chosen to implement the Data Manager in the ESS. This choice allows for common interfaces for the HLT algorithm to access the data when running online in the trigger or offline for development purposes, as will be discussed in the following.

Figure 9-22 shows the main components of the Data Manager.

- Storegate, as the transient data store, holds the event data during the processing.
- A **Region Selector** that provides an efficient look-up of the Hash Identifiers of all Collections of data that fall within a given geometrical Region of a sub-detector. A Region is typically defined in terms of η and ϕ by the RoI Objects provided by the LVL1 system.
- A **Detector Container** with RDO or RIO Collections for each sub-detector. The granularity of the Collections are optimized for the HLT algorithm processing. For example, a Collection for the Pixel Detector contains the RDOs or RIOs (Clusters) in a 'Detector Element', in this case a module. For the LAr a Collection contains all cells in a sampling of a trigger tower. An HLT algorithm retrieves each required Collection from the Detector Container using a Hash Identifier.
- The **Byte Stream Conversion Service** that provides the means to access the ROB data, to convert it and to provide the Collections of data. The Byte Stream Conversion Service is based on the normal Storegate persistency mechanism [9-23], which is designed to retrieve collections of objects from a persistent data source, i.e. a file or database. In the trigger context, however, this source is replaced by the ROB Data Provider to access the ROB Fragments containing the detector information in serialized format. In LVL2 the ROB

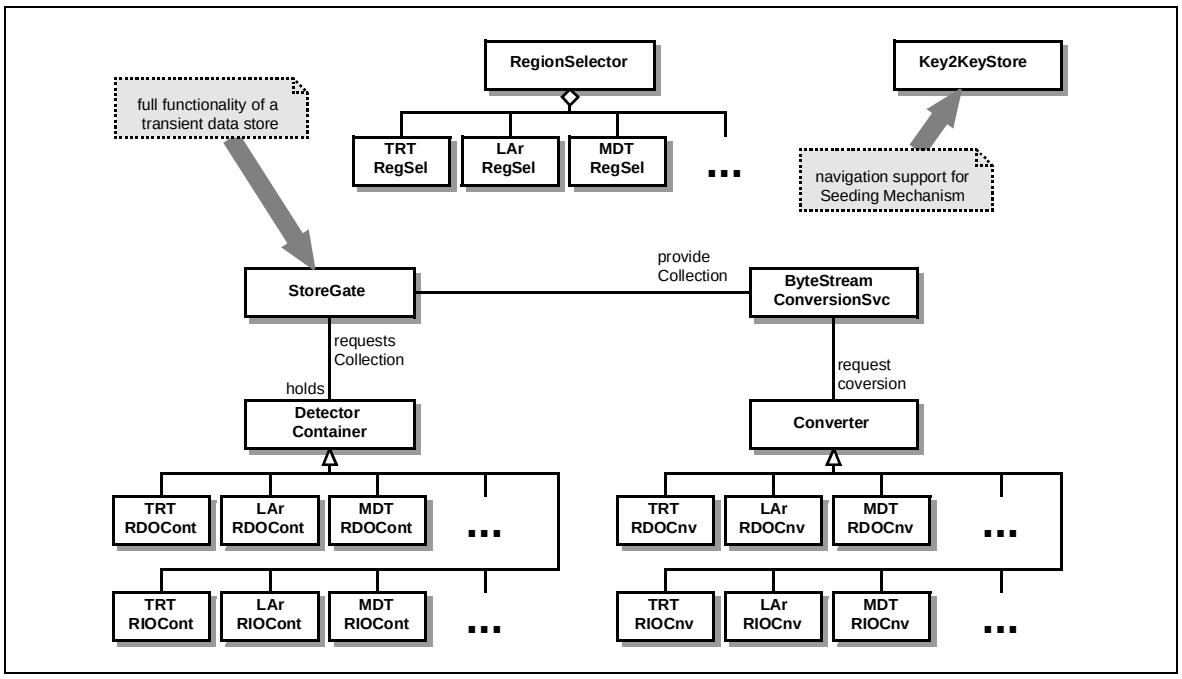

Figure 9-22 The class diagram for the Data Manager sub-package

Fragments have to be requested across the network and in the EF they are retrieved from the complete byte-stream event in memory.

- The **Converter** uses Data Preparation algorithms (as discussed in Chapter 9.5.3) to convert the Raw Data of the ROB fragment into objects to be analysed by the HLT algorithms. The Converters are specific to the detector and the type of data (RDOs for EF or RIOs for LVL2) requested by the HLT algorithm.
- The **Key to Key Store** is a trigger-specific implementation of the object relations that are used for the seeding mechanism (Section 9.5.3.1). It supports the 'seeded by' and 'uses' relations of Trigger Elements, RoI Objects and other EDM information in a generic C++ way that is also compatible with Storegate.

9.5.5.2 The Raw Data Access

The HLT Raw Data access follows the so-called 'London Scheme' [9-28], which uses the components described above. This scheme aims to keep data transfers and CPU time to a minimum by only requesting and unpacking data as it is required; to encapsulate data access so that the algorithm does not have to deal with details such as the byte stream or ROB granularity, and so that the data source can be changed (LVL2, EF, off-line emulation) with no impact on the algorithm. A sequence diagram of the HLT Raw Data access is shown in Figure 9-23. The HLT algorithm defines the geometrical Region from which it needs data. Typically the HLT algorithm uses the position of the LVL1 RoI and defines a region in terms of $\Delta\eta$ and $\Delta\phi$ around it. The geometrical Region is a general concept and is not confined to the original RoI information from LVL1: for example, during the HLT algorithm processing as more refined information becomes available it may be possible to use a more refined volume (e.g. allowing for track curvature) thereby reducing the amount of data to be analysed. The Region Selector [9-29] translates this abstract Region into a list of Hash Identifiers to be used by the HLT algorithm to access the Collections of RDO or RIO data. In the basic London Scheme the Region Selector merely returns the list of Hash Identifiers. However, for efficiency in the LVL2 environment, the Region Selector also pro-

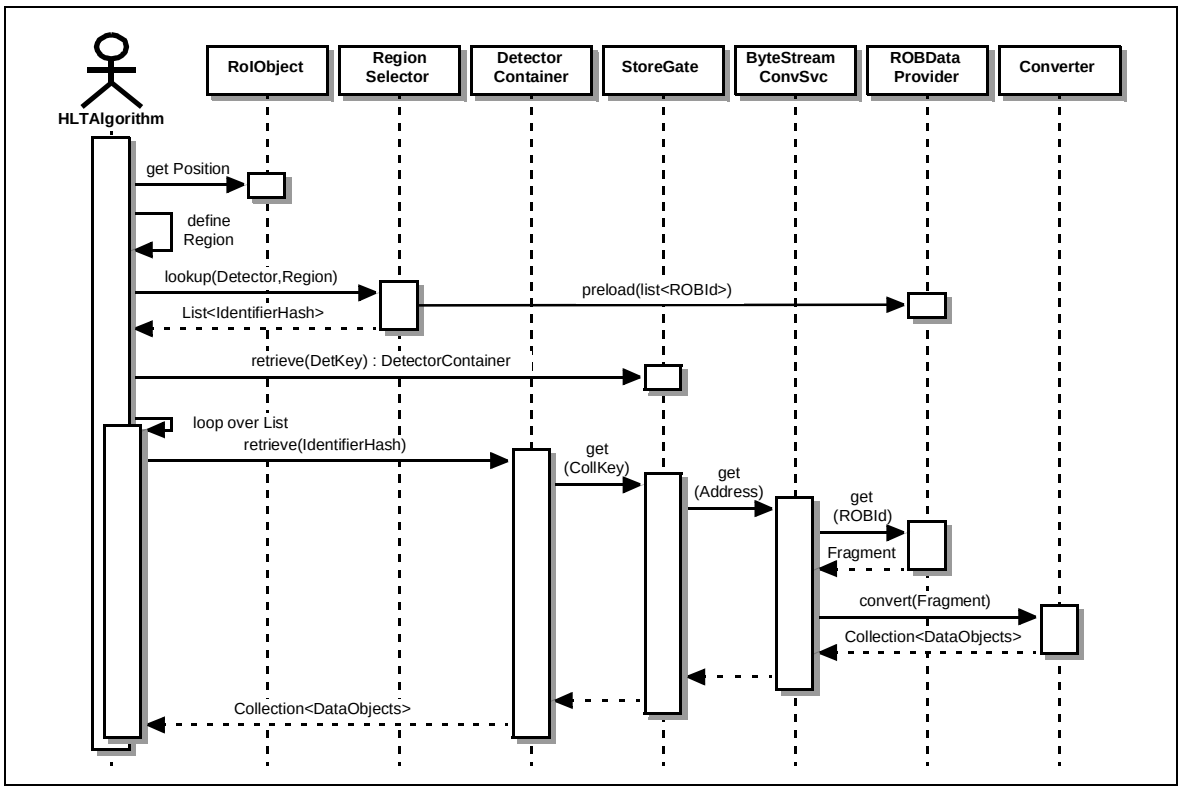

Figure 9-23 A sequence diagram showing the HLT Raw Data Access scheme

vides the list of ROB Fragments to be preloaded from the ROS via the ROB Data Provider. The ROB Data Provider caches the returned ROB fragments for later decoding. This optimization significantly reduces the network latency for ROB data access in LVL2 (see Section 14.2.1).

Having received the list of Hash Identifiers from the Region Selector, the HLT algorithm retrieves the Detector Container for the type of data it requires (for details see [9-30]). In the current prototype the EF requests RDO data and uses the full offline Data Preparation algorithms to obtain the RIO information. The LVL2 directly requests the RIO data in order to avoid the overhead of RDO creation. During the reconstruction process the HLT algorithm loops over the list requesting the individual Collections from the Detector Container. The Detector Container translates the Hash Identifier into an Address of the Collection in Storegate. In the normal case the Collection is not available and hence Storegate requests via its persistency mechanism the Byte Stream Conversion Service to provide the Collection. The Service retrieves the corresponding ROB fragment from the ROB Data Provider and uses either the RDO or RIO Converter to convert the data. The Collection is then stored in Storegate for possible subsequent requests and returned to the HLT algorithm. In practice a ROB fragment contains the data for more than one Collection. It is a matter of optimization either to decode only the requested part or to fully prepare the data of all collections in the first request.

It should be noted that in this scheme the encapsulation in the Region Selector of the translation from an abstract region to the detector identifiers hides the details of the detector geometry from the HLT algorithms. The Storegate persistency mechanism hides the details of the online Raw Data. Thus the scheme provides the means to use offline code in the trigger and to use and develop dedicated LVL2 algorithms in the offline environment.

9.6 Summary

This chapter has described the components used in the HLT event selection, with the main emphasis on functional aspects and implementation. Some performance figures have been given for software components within the processors (the L2P and the EFP), but the overall system performance is largely determined by aspects outside of the scope of this chapter. For LVL2 a major requirement has been achieving adequate performance for RoI Collection, results for this (both for a single L2P and when multiple L2Ps and additional event building capacity are added) have been presented in Chapter 8 and show that this can be done using only a small part of the total LVL2 time budget. For the EF the movement of events through the system has also been demonstrated with performance well within the limits required. However, the validation of the overall architecture also requires that realistic algorithms, including data conversion from realistic input data, can be supported with adequate performance in the software framework described. The algorithms being developed for HLT, and their current performances, are described in Chapter 13. System tests of both LVL2 and the EF are described in Chapter 14.

9.7 References

9-1 J. Schlereth and M. LeVine, *LVL2 Supervisor Requirements*, ATL-DQ-ES-0025 (2001) https://edms.cern.ch/document/391501/

9-2 J. Schlereth, *Level 2 Supervisor Design*, ATL-DQ-ES-0049 (2002) https://edms.cern.ch/document/391581/

9-3 C. Hinkelbein et al., *Prospects of FPGAs for the ATLAS LVL2 Trigger*, ATLAS Internal Note, ATL-DAQ-2000-006 (1999)

9-4 A. Khomich et al., *Timing measurements of some tracking algorithms and suitability of FPGA's to improve the execution speed*, ATLAS Internal Note, ATL-COM-DAQ-2003-006 (2003)

9-5 A. Bogaerts et al., *LVL2 Processing Unit Application Design*, ATL-DH-ES-0009 () https://edms.cern.ch/document/391554/

9-6 A. dos Anjos et al., *Guidelines for offline preparation and testing of LVL2 code*, http://www.cern.ch/steve.armstrong/algorithms/guidelines

9-7 *Athena: User Guide and Tutorial*, http://atlas.web.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/Tech.Doc/Manual/2.0.0-Draft/AthenaUserGuide.pdf

9-8 *Gaudi*, http://cern.ch/proj-gaudi/

9-9 S. Gonzalez et al., *Use of Gaudi in the LVL2 Trigger: The Steering Controller*, ATL-DH-EN-0001 () https://edms.cern.ch/document/371778/

9-10 A. Bogaerts et al., *Initial LVL2 Tests with the Si Tree Algorithm*, ATL-DH-TN-0001 () https://edms.cern.ch/document/375305/

9-11 H.P. Beck et al., *ATLAS TDAQ, A Network-based Architecture*, ATL-DQ-EN-0014 () https://edms.cern.ch/document/391592/

9-12 C. Bee et al., *Event Handler Requirements*, ATL-DH-ES-0003 () https://edms.cern.ch/document/361786/

9-13 C. Meessen et al., *Event Handler Functional Design Analysis*, ATL-DH-ES-0005 ()
https://edms.cern.ch/document/361808/

9-14 C. Bee et al., *Event Handler Design*, ATL-DH-ES-0007 ()
https://edms.cern.ch/document/367089/

9-15 A. dos Anjos et al., *Event Format Library Requirements*, ATL-DQ-EN-0005 ()
https://edms.cern.ch/document/383859/

9-16 F. Touchard et al., *Event Filter infrastructure validation tests*, ATL-DH-TR-0004
https://edms.cern.ch/document/391248/

9-17 F. Touchard et al., *HLT operational analysis and requirements to other sub-systems*, ATL-DH-ES-0006 ()
https://edms.cern.ch/document/363032/

9-18 C. Bee et al., *Supervision requirements*, ATL-DH-ES-0004 ()
https://edms.cern.ch/document/361792/

9-19 C. Bee et al., *ATLAS Event Filter: Test Results for the Supervision Scalability at ETH Zurich*, ATLAS Internal Note, ATL-DAQ-2002-006 (2001)

9-20 S. Wheeler et al., *Test results for the EF Supervision*, ATL-DH-TR-0001 ()
https://edms.cern.ch/document/374118/

9-21 S. George et al., *PESA high level trigger selection software requirements*, ATLAS Internal Note, ATL-DAQ-2001-005 (2001)

9-22 M. Elsing et al., *Analysis and Conceptual Design of the HLT Selection Software*, ATLAS Internal Note, ATL-DAQ-2002-013 (2002)

9-23 P. Calafiura et al., *Storegate: a Data Model for the ATLAS Software Architecture*, Computing in High Energy and Nuclear Physics Conference, Beijing, 2001
http://atlas.web.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/EventDataModel/Tech.Doc/StoreGate/Chep01.pdf

9-24 M. Abolins et al., *Specification of the LVL1/LVL2 Trigger Interface*, ATL-D-ES-0003 ()
https://edms.cern.ch/document/107485/

9-25 T. Schoerner-Sadenius, *TrigConfig: The HLT configuration package*, ATLAS Internal Note, ATL-COM-DAQ-2002-019 (2002)

9-26 M. Elsing and T. Schoerner-Sadenius, *Configuration of the ATLAS Trigger System*, Computing in High Energy and Nuclear Physics Conference, San Diego, 2003

9-27 *Xerces: XML parser in C++*,
http://xml.apache.org/xerces-c/

9-28 S. George et al., *Design of the PESA core software Data Manager*, ATL-DH-EN-0004 ()
https://edms.cern.ch/document/390533/

9-29 A.G. Mello, S. Armstrong and S. Brandt, *Region-of-Interest Selection for ATLAS High Level Trigger and Offline Software Environments*, ATLAS Internal Note, ATLAS-COM-DAQ-2003-005 (2003)

9-30 S. Armstrong et al., *Algorithms for the ATLAS High Level Trigger*, ATLAS Internal Note, ATL-DAQ-2003-013 (2003)

10 Online Software

10.1 Introduction

The Online Software encompasses the software to configure, control, and monitor the TDAQ system but excludes the management, processing, and transportation of physics data. It is a customizeble framework which provides essentially the ‘glue’ that holds the various sub-systems together. It does not contain any elements that are detector specific as it is used by all the various configurations of the TDAQ and detector instrumentation. It co-operates with the other sub-systems and interfaces to the Detector Readout Crates, the Detector Control System, the LVL1 Trigger, the DataFlow, the High Level Trigger processor farms, the Offline Software, and to the human user as illustrated in Figure 10-1. It also provides the interface between the human user and the TDAQ system.

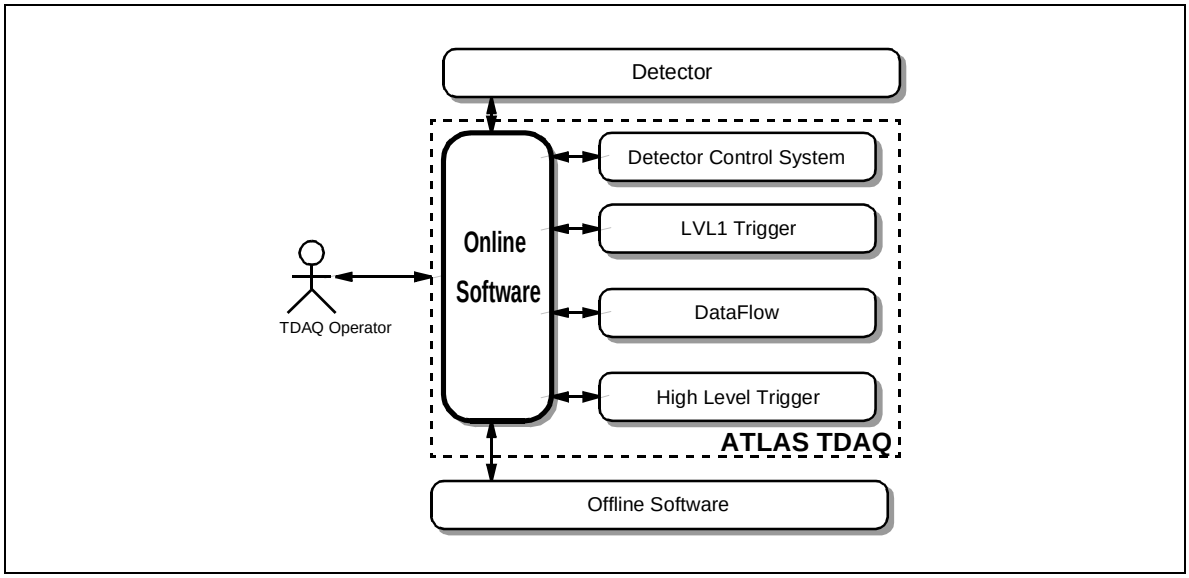

Figure 10-1 Context diagram of the Online Software

An important task of the Online Software is to provide services to marshal the TDAQ through its start-up and shutdown procedures so that they are performed in an orderly manner. It is responsible for the synchronization of the states of a run in the entire TDAQ system and for process supervision. These procedures are designed to take a minimum amount of time to execute in order to reduce the overhead, since this affects the total amount of data that can be taken during a data-taking period. Verification and diagnostic facilities help with early and efficient problem finding. Configuration database services are provided for holding the large number of parameters which describe the system topology, hardware components, software components, and running modes. During data taking, access is provided to monitoring information like statistics data, sampled data fragments to be analysed by monitoring tasks, histograms produced in the TDAQ system, and also to the errors and diagnostic messages sent by different applications. User interfaces display the status and performance of the TDAQ system and allow the user to configure and control the operation. These interfaces provide comprehensive views of the various sub-systems for different types of users and operations.

The Online Software distinguishes various types of user. The *TDAQ Operator* runs the *TDAQ System* in the control room during a data-taking period; the *TDAQ Expert* has system-internal knowledge and can perform major changes to the configuration; the *Sub-system Expert or Detector Expert* is responsible for the operation of a particular sub-system or detector; and *TDAQ and detector software applications* use services provided by the Online Software via application interfaces. These types of users are defined in detail in Appendix B. The user requirements to the Online Software are collected and described in the corresponding document [10-1].

10.2 The architectural model

The Online Software architecture is based on a component model and consists of three high-level components, called packages. Details of the architecture and a comprehensive set of use cases are described in [10-2]. Each of the packages is associated with a group of functions of the Online Software. For each package, a set of services which it has to provide is defined. The services are clearly separated one from another and have well defined boundaries. For each service a low-level component, called a sub-package, is identified.

Each package is responsible for a clearly defined functional aspect of the whole system.

1. Control — contains sub-packages for the control of the TDAQ system and detectors. Control sub-packages exist to support TDAQ system initialization and shutdown, to provide control command distribution, synchronization, error handling, and system verification.
2. Databases — contains sub-packages for configuration of the TDAQ system and detectors. Configuration sub-packages exist to support system configuration description and access to it, record operational information during a run and access to this information. There are also boundary classes to provide read / write access to the conditions storage.
3. Information Sharing — contains classes to support information sharing in the TDAQ system. Information Sharing classes exist to report error messages, to publish states and statistics, to distribute histograms built by the sub-systems of the TDAQ system and detectors, and to distribute events sampled from different parts of the experiment's data flow chain.

The interaction between the Online Software packages is shown in Figure 10-2. The Control makes use of the Information Sharing and of the Databases packages. The Databases package is used to describe the system to be controlled. This includes in particular the configuration of TDAQ Partitions, TDAQ Segments, and TDAQ Resources. The Information Sharing package provides the infrastructure to obtain and publish information on the status of the controlled system, to report and receive error messages, and to publish results for interaction with the operator.

The following sections describe these packages in more detail.

10.3 Information sharing

There are several areas where information sharing is necessary in the TDAQ system: synchronization between processes, error reporting, operational monitoring, physics event monitoring, etc. The Online Software provides a number of services which can be used to share information

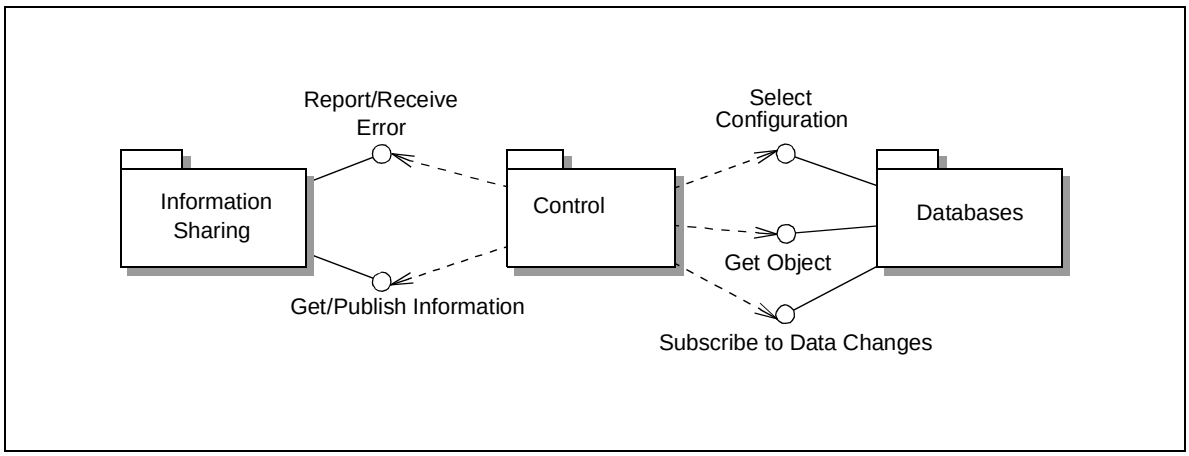

Figure 10-2 Internal interaction between the Online Software packages

between TDAQ software applications. This section will describe the architecture and prototype implementation of these services.

10.3.1 Functionality of the Information Sharing Services

Any TDAQ software application may produce information which is of interest for other TDAQ applications. In this section the application that produces information is called the Information Provider, the application that reads information data is called Information Consumer, which indicates that it is a user of the information. Any TDAQ software application may be an Information Provider and an Information Consumer at the same time. The main responsibility of the Information Sharing services is:

- transportation of the information from the Information Providers to the Information Consumers
- delivery of information requests (commands) from the Information Consumers to the Information Providers.

Figure 10-3 shows the main interactions which providers and consumers may have with the Information Sharing services.

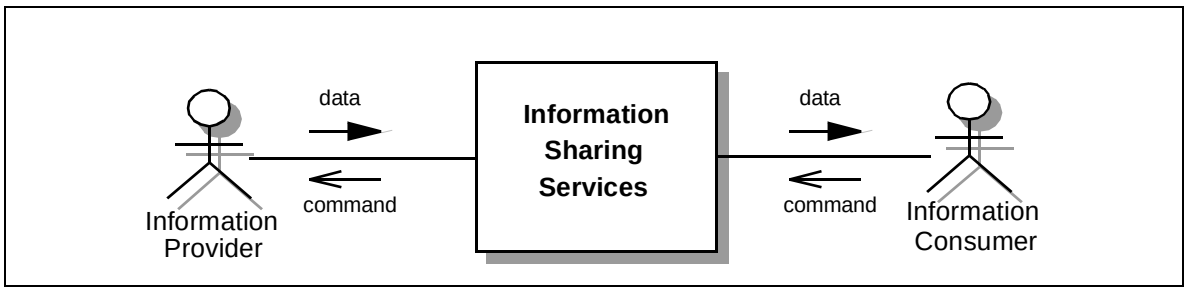

Figure 10-3 Information Sharing in the TDAQ system

10.3.1.1 Types of shared information

There are many types of information which can be shared between applications in the TDAQ system. These types are significantly different in terms of the data size, update frequency, type

of access, number of providers and consumers, etc. In order to optimize the performance of the information sharing in a large and highly distributed TDAQ system a separate service for each major class of the shared information is provided. Table 10-1 shows the main information types along with their most important characteristics.

Table 10-1 Types of shared information

Information type	Information structure	Providers produce this data...	Consumers access this data...
Physics events (or event fragments)	vector of 4-byte integers	on request	on request
Error messages	error id + description + optional qualifying attributes	when errors occur	via subscription
Histograms	several widely used histogram formats (e.g. ROOT histograms)	always	on request and via subscription
Status information	user-defined	always	on request and via subscription

10.3.2 Performance and scalability requirements on Information Sharing

The performance and scalability requirements vary according to the type of the information. Table 10-2 shows the summary of these requirements for the main types of shared information.

Table 10-2 Performance and scalability requirements for Information Sharing

Information type	Information size (kbyte)	Update frequency (Hz)	Number of providers	Number of consumers
Physics events (or event fragments)	$1.5–2.2 \times 10^3$	O(1)–O(10^3)	O(10^2)	O(10^2)
Error messages	O(1)	O(10)	O(10^3)	O(10)
Histograms	O(10)	O(1)	O(10^2)	O(10)
Status information	O(1)	O(1)	O(10^3)	O(10)

The final ATLAS TDAQ system will contain O(10^3) processes. Each of them can produce error messages and status information. The Information Consumers for the error messages are the applications which log errors, analyse them, present them to the operator, or try to do a recovery. The consumers of the status information are the applications that monitor the status of a particular component of the TDAQ system or detector, present this information to the operator, and possibly perform corrective actions when spotting problems. Therefore the Information Sharing services dealing with the errors and user-defined information have to be able to serve O(10^3) Information Producers and O(10) Information Consumers simultaneously.

Physics events (or event fragments) can be sampled from several hundred places in the data flow chain. Events can be sampled at ROD crate level, at the ROS level, and at the SFI level. In the worst case, if events are monitored at all the possible points simultaneously, there will be approximately the same number of Information Consumers for this information type.

The possible sources of histograms are ROD controllers, ROSs, SFIs, LVL2 sub-farms, and EF sub-farms. For each of these systems there are several applications which receive histograms in order to present them to the operator, analyse, or store them. Therefore the Information Sharing service dealing with histograms, must be able to handle several hundred Information Providers and several tens of Information Consumers simultaneously.

10.3.3 Architecture of Information Sharing services

The Online Software provides four services to handle different types of shared information. Each service offers the most appropriate and efficient functionality for a given information type and provides specific interfaces for both Information Providers and Consumers. Figure 10-4 shows the existing services.

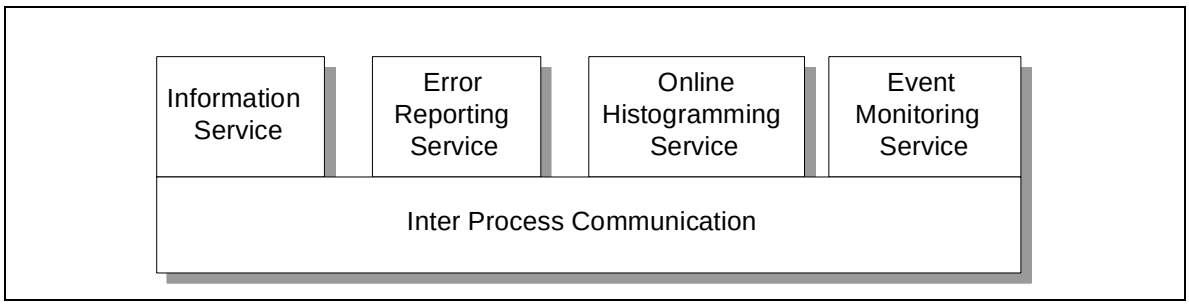

Figure 10-4 Information Sharing services

The Inter Process Communication (IPC) is a basic communication service which is common to all the other Online Software services. It defines the high-level API for the distributed object implementation and for remote object location. Any distributed object in the Online Software services has common basic methods which are implemented in the IPC. In addition the IPC implements partitioning, allowing several instances of the Online Software services to run in different TDAQ Partitions concurrently and fully independently.

10.3.3.1 Information Service

The Information Service (IS) allows software applications to exchange user-defined information. Figure 10-5 shows interfaces provided by the IS.

Any Information Provider can make his own information publicly available via the Publish interface and notify the IS about changes of the published information via the Update interface. Here there are two possibilities: the Information Provider does not implement the InfoProvider interface, in which case it has to inform the IS about all the changes of the published information; the Information Provider does implement the InfoProvider interface, in which case it notifies the IS about information changes only when it is explicitly requested by the IS via the Command interface. Some other commands might be also possible, e.g. setting time interval for the information updates.

There are also two types of Information Consumer. One can access the information on request via the Get Info interface, in which case it does not need to implement the InfoConsumer interface. The second type of Information Consumer implements the InfoConsumer interface, and is informed about changes of the information for which it subscribed, via the Subscribe interface.

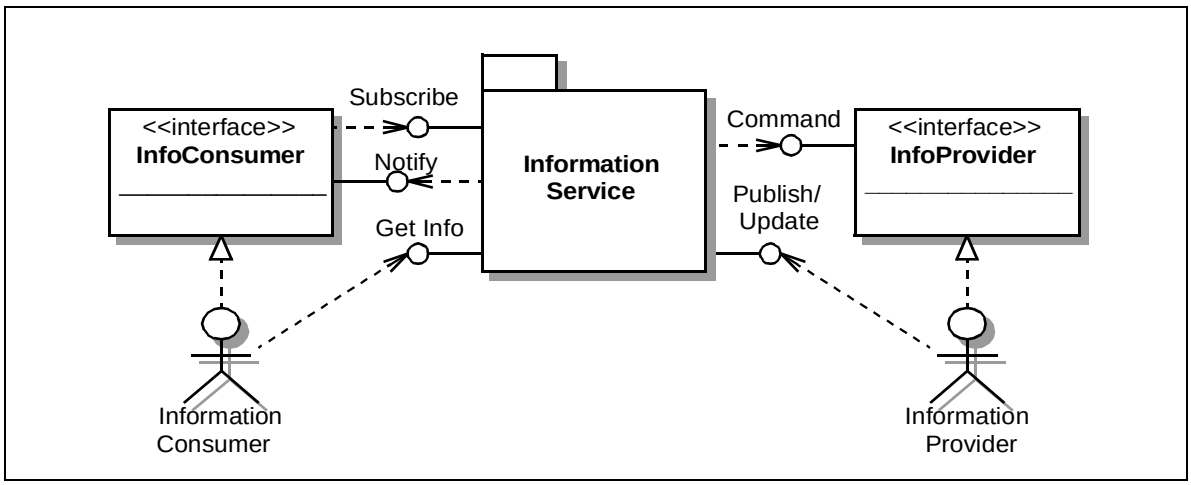

Figure 10-5 Information Service interfaces

10.3.3.2 Error Reporting Service

The Error Reporting Service (ERS) provides transportation of the error messages from the software applications which detect these errors to the applications which are responsible for their handling. Figure 10-6 shows interfaces provided by the Error Reporting Service.

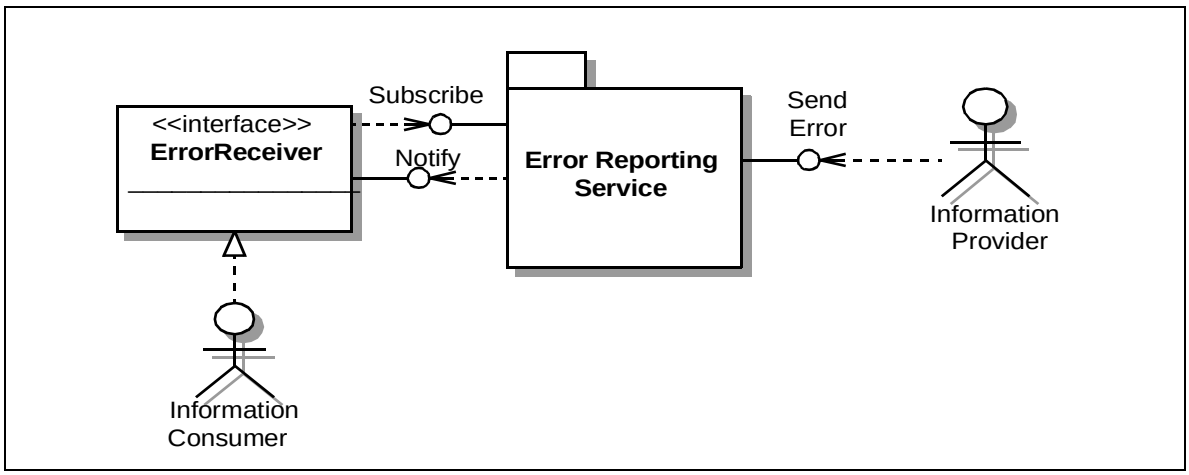

Figure 10-6 Error Reporting Service interfaces

An Information Provider can send an error message to the ERS via the Send Error interface. This interface can be used by any application which wants to report an error. In order to receive error messages an Information Consumer has to implement the ErrorReceiver interface and provide the criteria that define the kind of messages it wants to receive. These criteria have to be passed to the ERS via the Subscribe interface.

10.3.3.3 Online Histogramming Service

The Online Histogramming Service (OHS) allows applications to exchange histograms. The OHS is very similar to the Information Service. The difference is that the information which is transported from the Information Providers to the Information Receivers has a pre-defined format. Figure 10-7 shows interfaces provided by the Online Histogramming Service.

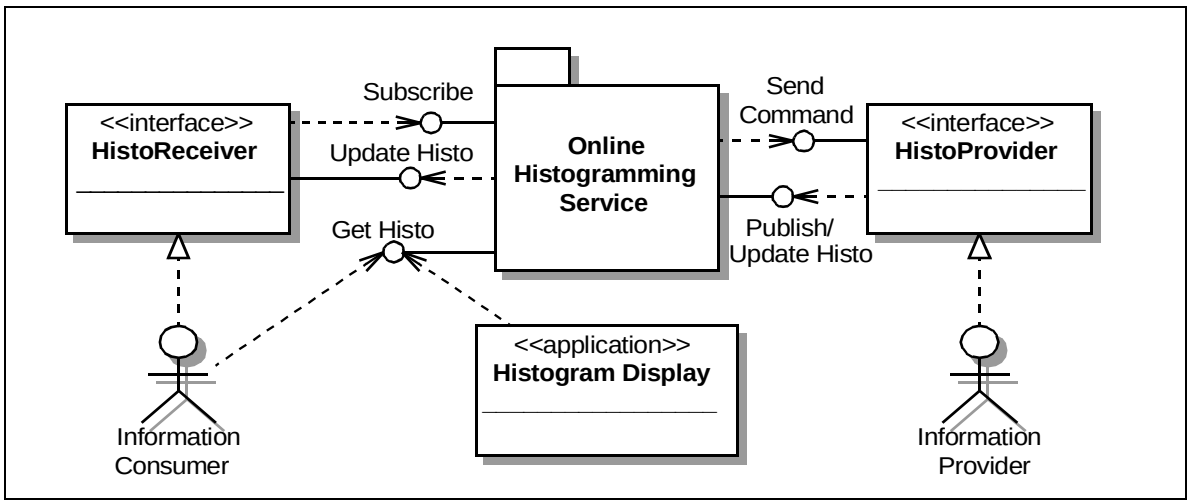

Figure 10-7 Online Histogramming Service interfaces

The OHS sub-package also provides a human user interface in the form of an application. This application is called Histogram Display and can be used by the TDAQ operator to display available histograms.

10.3.3.4 Event Monitoring Service

The Event Monitoring Service (EMS) is responsible for the transportation of physics events or event fragments sampled from well-defined points in the data flow chain and delivered to analysis applications. These will examine and analyse the data in order to monitor the state of the data acquisition and the quality of the physics data in the experiment. Figure 10-8 shows the main interfaces provided by the Event Monitoring Service.

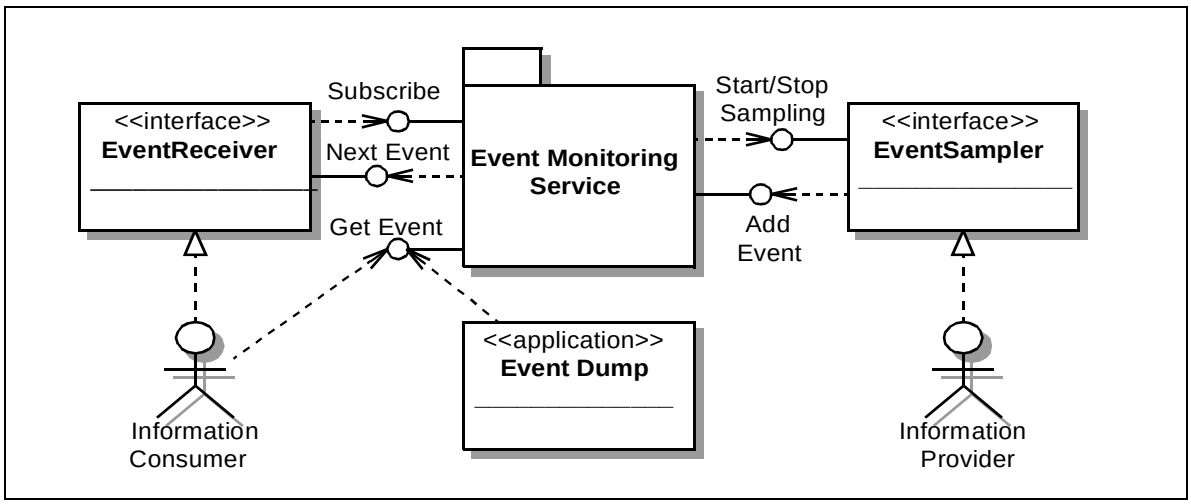

Figure 10-8 Event Monitoring Service interfaces

The application, which is able to sample events from a certain point of the data flow, has to implement the Event Sampler interface. When the Information Consumer requests samples of events from that point, the EMS system requests the Information Provider, via the Start Sampling interface, to start a sampling process. The Information Provider samples events and provides them to the EMS via the Add Event interface. When there are no more Information

Consumers interested in event samples from that point of the data flow chain, the EMS requests the Information Provider via the Stop Sampling interface to stop the sampling process.

There are also two types of interface for the Information Consumer. One is a simple Get Event interface which allows a consumer to request event samples one by one according to need. This interface will be used, for example, by the Event Dump application that implements a human user interface to the EMS system. A second interface is based on the subscription model. The Information Consumer can request the EMS system to supply the samples of events as soon as they are sampled by the Information Provider. This interface is more suitable for the monitoring tasks which need to monitor events for a long period of time in order to accumulate the necessary statistics.

10.3.3.5 Relation between Information Sharing services

All the Information Sharing services described above have essential differences because they have to deal with different types of shared information. On the other hand, similarities can be identified since they use common communication patterns, for example the subscribe/notify mechanism. The generic patterns are reused by different services whenever it is possible without loss of efficiency. For example, in the current implementation the Histogramming Service is implemented on top of the more general Information Service. The Histogramming Service defines its own specific interfaces but reuses all the communication mechanisms provided by the IS.

10.3.4 Prototype evaluation

Prototype implementations exist for all the Information Sharing services. These prototypes aim to verify the suitability of the chosen design and the choice of implementation technology for the final TDAQ system, and for use at the ATLAS test beam operations. This section contains a description of the services implementation together with their performance and scalability test results.

10.3.4.1 Description of the current implementation

Each service is implemented as a separate software package with both C++ and Java interfaces. It is possible to have several instances of each service running concurrently and fully independently in different TDAQ partitions.

The Information Sharing services implementation is based on the Common Object Request Broker Architecture (CORBA) [10-4] defined by the Object Management Group (OMG). CORBA is a vendor-independent industry standard for an architecture and infrastructure that computer applications use to work together over networks. The most important features of CORBA are object-oriented communication, inter-operability between different programming languages and different operating systems, and object location transparency.

Currently the open source implementation of CORBA provided by the Xerox company is used. It is called Inter Language Unification (ILU) [10-3]. Several other CORBA implementations are currently being evaluated. These are TAO [10-5], MICO [10-6], omniORB [10-7], and ORBacus [10-8]. They provide interesting features which are missing in ILU. ILU can be replaced by an-

other CORBA implementation without affecting the architecture and design of the Information Sharing services.

10.3.4.2 Performance and scalability of current implementation

Among the Information Sharing services, the most extensive tests have been performed for the Information Service, which provides the most general facility for information sharing. The other services are implemented on the same technology and will offer the same level of performance and scalability as the IS.

The test bed for the IS tests consisted of 216 dual-pentium PCs with processor frequencies from 600 to 1000 MHz [10-9]. The IS server was set up on one dedicated machine. From one to five Information Providers were running on the other 200 machines. Each Information Provider published one information object at the start and then updated it once per second. The remaining 15 machines were used to run 1, 5, 10 or 15 Information Consumers which subscribe for all the information in the IS. Whenever an Information Provider changed the information, this new information was transported to all the Information Consumers.

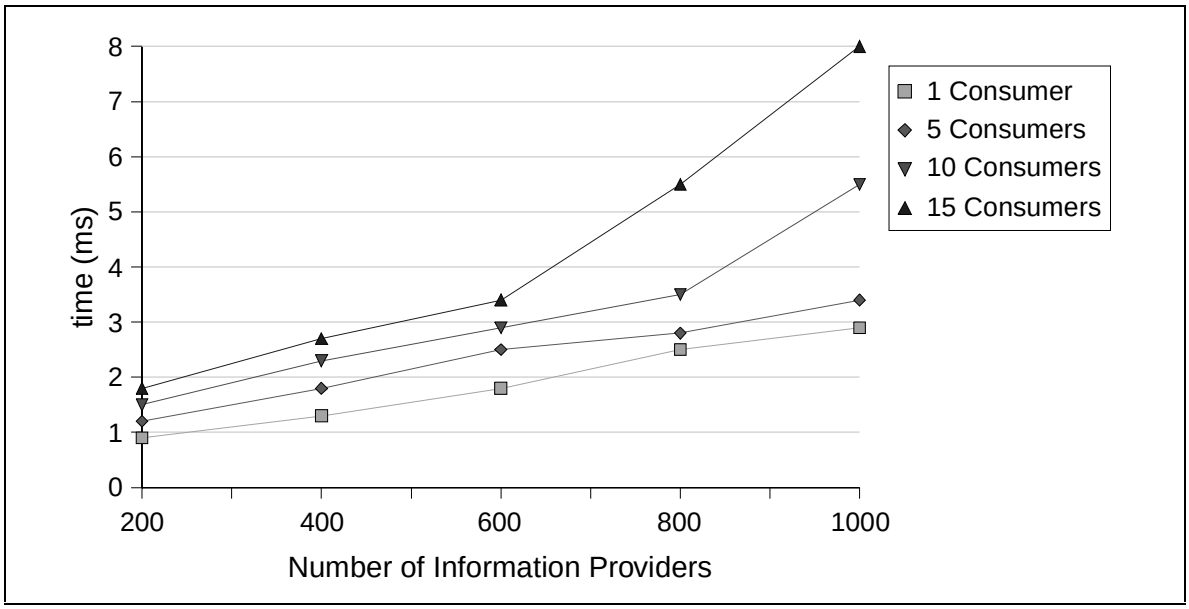

Figure 10-9 Average time spent to transport one information object from one Information Provider to a number of Information Consumers versus the number of concurrent Information Providers

Figure 10-9 shows the average time for transporting information from one Information Provider to all the subscribed Information Consumers as a function of the number of Information Providers, which were working concurrently.

The results of the tests show that a single IS server is able to handle one thousand Information Providers and about 10 Information Consumers at the same time. The design of the IS allows the total load to be distributed among a number of independent IS servers. Thus, it will be necessary to run only a few (less then 10) IS servers in order to provide the required performance for the final ATLAS TDAQ.

10.3.4.3 Usage of Information Sharing services by the TDAQ sub-systems

Usage of the Information Sharing services as they are described above will be very similar to that of the existing prototype implementations. These have been in use in various test beds and test beam activities and provide good examples for their use in the final TDAQ system. A typical test beam configuration, which was exploited by the detectors, includes the Error Reporting Service (called Message Reporting System (MRS) in the prototype implementation) and a number of Information Services. In particular, the IS server for the Run Parameters was used to distribute the run parameters such as run number, run type, beam type, etc., which previously had been defined by the operator, to system and user applications. An IS server specific for the detector information was running and passed the status of the detector hardware such as RODs, LVL1 Modules, etc. to the operator during a run. The IS server for Data Flow parameters was used to present the status of the data flow elements to the operator, and an IS server specific to DCS parameters passed important DCS parameters on. The test beam operator used the TDAQ Online Software user interface application called Integrated Graphical User Interface (IGUI) to interact with the Information Sharing services. The IGUI application contains the message display, which shows text messages sent by the TDAQ and detector applications via MRS. The IGUI also includes the Data Flow panel, which displays the status of the data flow elements to the operator, and the Run Parameters panel, from where the operator can define parameters for the next run. In addition some detector teams implemented their own custom Java panels, which were integrated into the IGUI. These panels interacted with the various IS servers to fetch the detector-specific information and displayed it to the operator. The Event Monitoring Service was extensively used during the test beams by all detector groups. They developed specific applications, which conform to the EventSampler interface defined by the EMS. These applications were running on the ROD crate controllers of each ROD crate. A variety of monitoring tasks, receiving the events, had been developed in the detector groups and were running on dedicated workstations. Monitoring tasks gathered statistics about the sampled events or displayed the events to the operator. A very interesting application was developed in the TGC team implementing an Online Event Display in Java, which receives events from the EMS and displays them in graphical form.

10.4 Databases

The TDAQ systems and detectors require several persistent services to access the description of the configuration used for the data taking as well as to store the conditions under which the data were taken. The Online Software provides common solutions for such services taking into account the requirements coming from the TDAQ systems and detectors.

10.4.1 Functionality of the databases

There are three main persistent services proposed by the Online Software:

- the **Configuration Databases** (ConfDBs) to provide the description of the system configurations,
- the **Online Bookkeeper** (OBK) to log operational information and annotation,
- the **Conditions Database interface** to store and read conditions under which data were taken.

10.4.1.1 Configuration Databases

The ConfDBs are used to provide the overall description of the TDAQ systems and detector hardware and software. They include the description of partitions defined for the system. The hardware and software are described in a flexible way and parameterized in such a way the described configuration can be used for different types of runs.

The ConfDBs are organized in accordance with the hierarchy of the TDAQ system and detectors. They allow for each TDAQ system and detector to define their specific format of the data (i.e. the database schema), to define the data themselves, and to share the schema and the data with others. They provide graphical user interfaces for the human user to browse the data and for authorized human experts to modify the data. Data access libraries hide the technology used for the databases implementation and are used by any TDAQ or detector application to read the configuration description or to be notified in case of changes. An application started by an authorized expert can use the data access libraries to generate or to modify the data.

The ConfDBs provide an efficient mechanism for fast access to the configuration data for a large number of clients during data taking. They do not store the history of the data changes which is the task of the OBK but provide archiving options. Configuration data which are important for offline analysis must be stored in the Conditions Database.

10.4.1.2 Online Bookkeeper

The OBK is the system responsible for the online storage of relevant operational information, histograms, and the configuration description provided by the TDAQ systems and detectors. It organizes the stored data on a per-run basis and provides querying facilities.

The OBK provides a graphical user interfaces to allow human users to browse contents of the recorded information or append such information. The option for appending information is limited to authorized users. Similarly, the OBK provides programming interfaces for user applications to browse the information or to append annotations.

10.4.1.3 Conditions Database interfaces

The TDAQ systems and the detectors use the Conditions Database to store conditions which are important for the offline reconstruction and analysis. The conditions data include parameters such as temperature, pressure, and voltage and vary with time. These conditions are stored with a validity range which is typically expressed in time or run number and retrieved again using the validity range as a key.

The Conditions Database is expected to come from an LHC Computing Grid [10-10] applications area project, with any ATLAS-specific implementation supported by the Offline Software group. The Online Software system provides application programming interfaces for all TDAQ systems and detector applications, and mechanisms to ensure the required performance during data-taking runs.

10.4.2 Performance and scalability requirements on the databases

The performance and scalability requirements of the database services are strongly dependent on the strategies chosen by the TDAQ systems and detectors to transport the data to their appli-

cations, to store the conditions, and to keep the operational data. For most TDAQ systems and detectors, the number of estimated clients of the Configuration and of the Conditions Databases is roughly equal to the number of local controllers (as explained in Section 10.5.3), which are estimated to be between 500 and 2000. For the Event Filter the situation is not yet defined and the number of database clients in the worst-case scenario can be $O(10^3)$, if each processing task needs to receive the configuration description and conditions.

The database information can be split into a number of independent parts according to TDAQ systems and detectors. The complete description is required only by a few applications, while most others require access to only a small fraction of it. The typical database size which completely describes all the necessary parameters of the TDAQ system or a detector for a physics run is about $O(10^2)$ Mbyte. The DCS may produce up to 1 Gbyte of measured conditions per day.

The ConfDBs data are read once in preparation for data taking and an acceptable time to get the description for the whole system is of O(10) s. During physics data taking selected parts of the databases may be changed and an acceptable time to receive the changes by registered applications is of O(1) s. During special calibration runs a considerable fraction of the data can change and the maximum rate requested in this case is 10 Mbyte per hour.

10.4.3 Architecture of databases

10.4.3.1 Configuration Databases

The ConfDBs provide user and application programming interfaces.

Via a user interface the software developer defines the data object model (database schema) describing the required configurations. The expert uses the interface to manage databases, to populate the system description, and to define configurations, which can subsequently be browsed by a user.

A TDAQ or detector application accesses the databases via data access libraries (DALs). A DAL is generated by the software developer for the part of the object model which is relevant to his sub-system. It is based on a database schema to programming language mapping and related instantiation of database objects as programming language objects. The user of a DAL never sees the low-level database management system (DBMS) and his code can be used without changes for different database systems. The DAL is used by any process required to get the configuration description or to receive a notification in case of changes. The DAL is also used by an expert process to produce the configuration data.

Figure 10-10 shows the main classes and interfaces provided by the ConfDB system and their users.

The ConfDB system contains the following classes:

- **ConfDB Repository** — defines low-level interfaces to manipulate the ConfDBs including database management by users who are granted the respective privileges, schema and data definitions and notification subscription on data changes; it defines interfaces above a DBMS used for implementation and hides any specific details, so any other ConfDB classes must never use DBMS-specific methods directly;

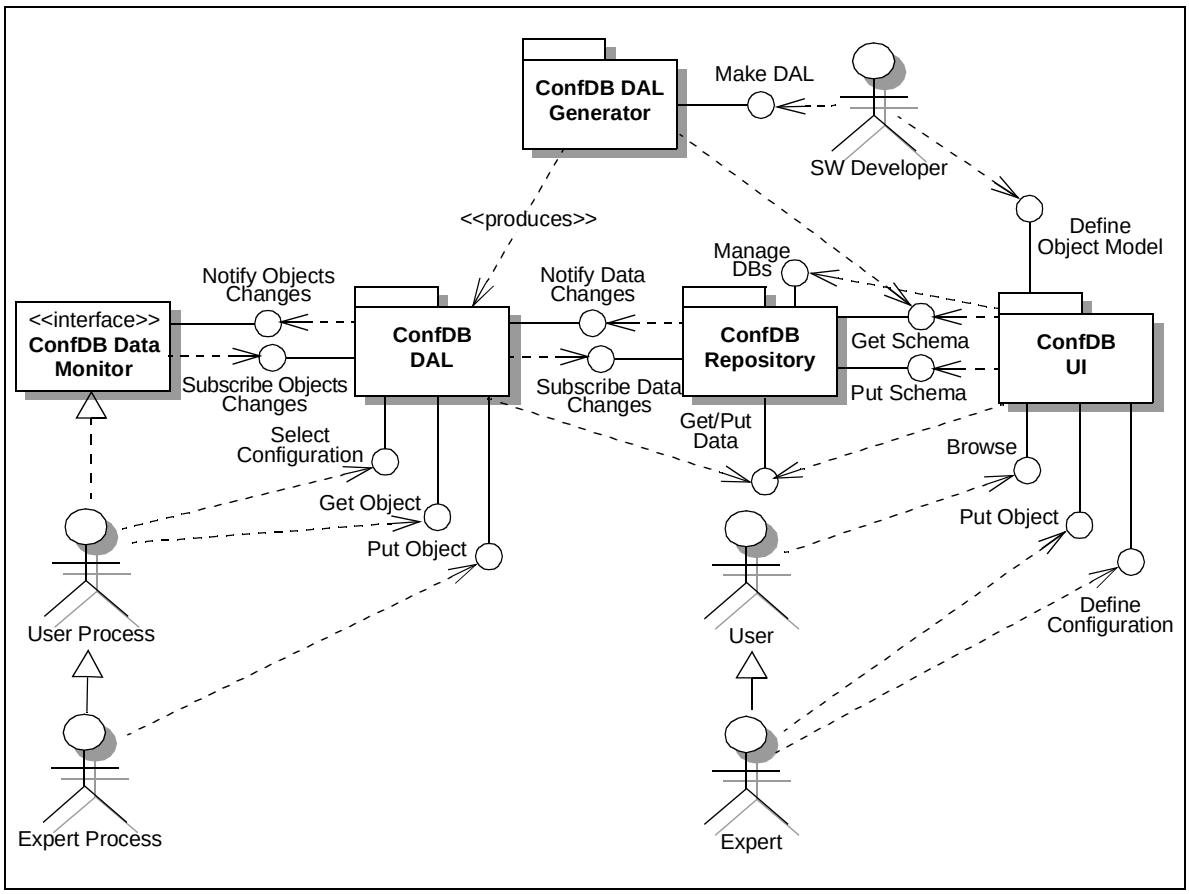

Figure 10-10 Configuration Databases users and interfaces

- **ConfDB UI** (User Interface) — defines the user interface for object model definition, configurations definition, database browsing and population by human users;
- **ConfDB DAL Generator** — defines the interface to produce a DAL;
- **ConfDB DAL** — defines interfaces for configuration selection, reading and writing of configuration objects, and subscription for notifications on data changes by user and expert processes;
- **ConfDB Data Monitor —** defines interfaces to receive notification of changes.

10.4.3.2 Online Bookkeeper

The OBK provides several interfaces to its services, some of them are to be used by the human user, while others are APIs to allow interfacing with client applications. The access to these interfaces depends on the user's (human or system actor) privileges.

The OBK uses as persistency backbone the Conditions and Time-Varying offline databases services. It counts on those services to provide the necessary means to store and retrieve coherently data that changes in time and of which there may exist several versions (e.g. configurations). Figure 10-11 shows the logical subdivision of the OBK system into abstract classes. Of the ones shown, the main ones are:

- **OBK Repository** — defines the basic methods to allow for storing, modifying, and reading of online log data, as well as the methods to set the OBK acquisition mode and also to

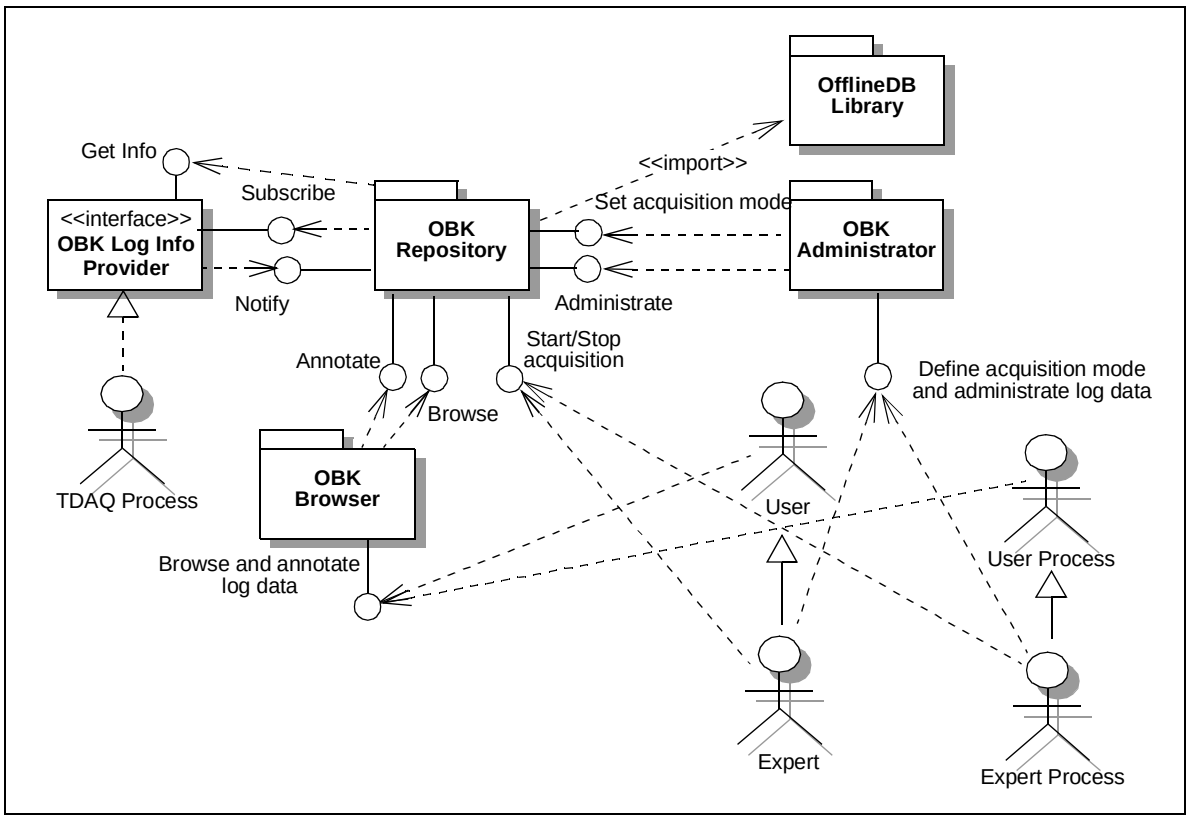

Figure 10-11 OBK users and interfaces

request log data from the several TDAQ processes providing them. It allows a human or system user to start or stop the acquisition of log data. In order to become a log data provider a TDAQ application will have to contain the **OBK Log Info Provider** interface. This interface allows a TDAQ application to accept subscriptions for log information from the OBK, as well as for the OBK to access log information in a TDAQ application;

- **OBK Browser** — this is the class responsible for providing the necessary querying functionality for the OBK database. Since the data browsing and data annotation functions are tightly coupled, the class also includes functionality to add annotations to the database;
- **OBK Administrator** — the OBK Administrator class provides to the users, which are granted the respective privileges, the functionality to alter (delete, move, rename) parts or all of the OBK database. These users are also given the possibility of changing the OBK acquisition mode (e.g. data sources, filters for the data sources).

Apart from the main classes depicted in Figure 10-11, the OBK architecture also includes four other classes (not shown in the diagram for reasons of clarity). The **OBK UI Browser** and the **OBK Browser API** both inherit from the OBK Browser class and define the human client oriented and the application client oriented versions of that class. Similarly the **OBK UI Administrator** and the **OBK Administrator API** classes define the human client and application client oriented versions of the OBK Administrator class.

10.4.3.3 Conditions Database interface

The user requirements for the ATLAS offline conditions and time-varying databases and their architecture are not yet fully specified by the ATLAS user community. The Conditions Database

interface will provide additional functionality if required by TDAQ and detector users, beyond the one provided by the Offline Software.

10.4.4 Prototype evaluation

The main functions of the ConfDB and the OBK have been implemented and evaluated in the prototype of the Online Software. The main goals were its use during test beam operations, the integration with other TDAQ sub-systems and detectors, and the evaluation of the suitability of the chosen technologies for the final system.

10.4.4.1 Scalability and performance tests of the Configuration Databases

The prototype of the ConfDBs is implemented on top of the OKS [10-11] persistent in-memory object manager. It allows the storage of the database schema and data in multiple XML files. Several subsets can be combined to get a complete description. Access to the configuration description can be made via a file system (C++ interface) and via a dedicated remote database server built on top of ILU [10-3] and tested for C++ and Java interfaces.

Figure 10-12 presents the database test results obtained during the Online Software large-scale and performance tests [10-9]. The options to read a realistic configuration via the AFS file system and from a remote database server were tested for the maximum available number of nodes. The tests with direct concurrent access to the configuration via the common AFS file system showed good scalability and performance as presented in the first graph of Figure 10-12. Such an approach can be envisaged if a common file system is available. The second graph in Figure 10-12 presents the time necessary to read different sizes of information from one database server depending on the number of concurrently reading clients. The third graph in Figure 10-12 shows the time necessary to read one information object from one database server for the case where the read request by the clients is initiated synchronously and for the case where these requests are randomly distributed over one second. Given a particular situation, one can derive from the graphs the number of necessary database servers in the system depending on the number of clients, timing requirements, data volume, and request synchronization.

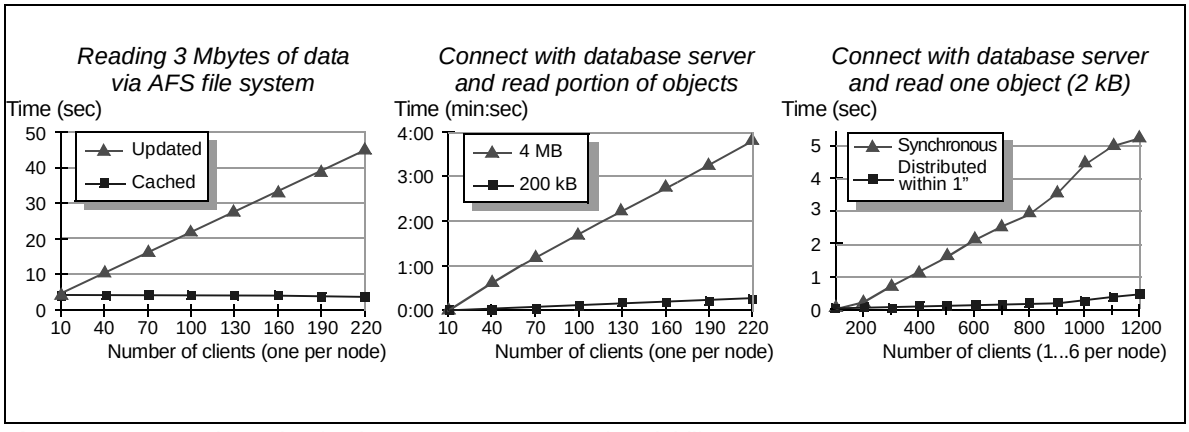

Figure 10-12 Results of the databases performance and scalability tests

The proposed architecture of the ConfDBs allows switching between implementation technologies without affecting user code. Some other database technologies are being studied for possible replacement of the ones used in the prototype, including relational databases with possible

object extensions, and the Pool Of persistent Objects for LHC (POOL) [10-12] from the LHC Computing Grid (LCG) [10-10] project.

10.4.4.2 Online Bookkeeper

The prototype of the OBK was implemented on an OKS persistent in-memory object manager and MySQL [10-13] freeware implementation of a relational database management system. The results obtained during recent performance and scalability tests [10-9] have shown that the current MySQL implementation can reach a rate of 20 kbyte/s when storing monitoring data (100 bytes per data item) produced by up to 100 providers.

10.5 Control

The main task of the control package is to provide the necessary tools to perform the TDAQ system operation as described in Chapter 3, "System Operations". It provides the functionality of the TDAQ Control as shown in the controls view in Chapter 5, "Architecture".

In addition, the package has the responsibility for the functions necessary for user interaction, process management, and access control in the computing environment.

10.5.1 Control functionality

Control encompasses software components responsible for the control and supervision of other TDAQ systems and the detectors. The functions have been derived from the user requirements and are:

- **User Interaction:** Interaction with the human user such as the operator or expert of the TDAQ system
- **Initialization and shutdown:** Operations for the initialization of TDAQ hardware and software components are foreseen. The operational status of system components must be verified and the initialization of these components in the required sequence ensured. Similar considerations are required for the shutdown of the system.
- **Run control:** System commands have to be distributed to many hundreds of clients programs. The control sub-package is responsible for the command distribution and the synchronization required between the TDAQ sub-systems and detectors.
- **Error handling:** Malfunctions can interrupt system operations or adversely affect the quality of physics data. It is the task of the control sub-package to identify such malfunctions. If required the system will then autonomously perform recovery operations and assist the operator with diagnostic information.
- **Verification of System status**: The control package is responsible for verifying the functioning of TDAQ configuration or any subset of it.
- **Process Management:** Process management functionality in a distributed environment is provided.
- **Resource Management:** Management of shared hardware and software resources in the system is provided.

- **Access Management:** The control package provides a general Online Software safety service, responsible for TDAQ user authentication and the implementation of an access policy for preventing non-authorized users corrupting TDAQ functionality.

10.5.2 Performance and scalability requirements on Control

The TDAQ system is a large and heterogeneous system composed of a large number of items to be controlled. Typically these items are clustered and range from readout modules in VME crates to workstations within HLT computer farms. Such clusters are the preferred places to interface with the Online Software Control system. The number of these clusters is estimated to be in the range of 500–1000. To control these units the TDAQ control system is built in a hierarchical and distributed manner. More detailed explanation can be found in Section 12.3.

10.5.3 Control architecture

The Control package is divided into a number of sub-packages as shown in Figure 10-13. The functionality described in Section 10.5.1 has been distributed between several distinct sub-packages:

- The User Interface (UI) for interaction with the operator
- The Supervision for the control of the data-taking session including initialization and shutdown, and for error handling
- The Verification for analysis of the system status
- The Process Management for the handling of processes in the distributed computing environment
- The Resource Management for coordination of the access to shared resources
- The Access Management for providing authentication and authorization when necessary.

10.5.3.1 User Interface

The **User Interface** (UI) provides an integrated view of the TDAQ system to the operator and should be the main interaction point. A flexible and extensible UI will be provided that can accommodate panels implemented by the detectors or TDAQ systems. Web-based technologies will be used to give access to the system for off-site users.

10.5.3.2 Supervision

The **Supervision** sub-package realizes the essential functionality of the Control package. The generic element is the controller. A system will generally contain a number of controllers organized in a hierarchical tree, one controller being in control of a number of others in the system while being controlled itself from a higher level controller. One top level controller, called the root controller, will take the function of the overall control and coordination of the system. The User Interface sub-package provides all TDAQ control facilities to the Operator. These are shown as interfaces in Figure 10-14 and discussed below in more detail.

The *Initialization and Shutdown* is responsible for:

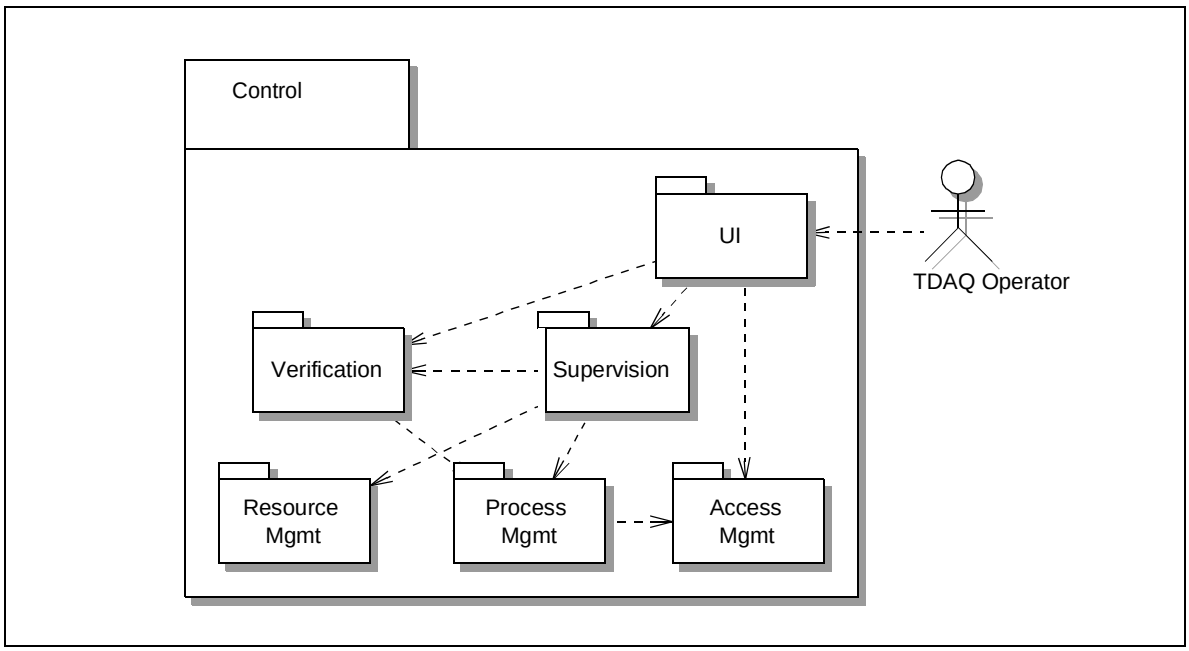

Figure 10-13 The organization of the control package

- initialization of TDAQ hardware and software components, bringing the TDAQ partition to the state in which it can accept Run commands
- re-initialization of a part of the TDAQ partition when necessary
- shutting the TDAQ partition down gracefully
- TDAQ process supervision

The *Run Control* is responsible for

- controlling the Run by accepting commands from the user and sending commands to TDAQ sub-systems
- analysing the status of controlled sub-systems and presenting the status of the whole TDAQ to the Operator

The *Error Handling* is concerned with

- analysing run-time error messages coming from TDAQ sub-systems
- diagnosing problems, proposing recovery actions to the operator, or performing automatic recovery if requested

Most of the above-defined functionality can reside in a so-called *Local Controller* and is extended by specific policies which the TDAQ sub-systems and detector expert developers implement. The interface and policies of the Local Controller can be configured by the user using a *Policy/ Knowledge* interface. In addition the supervision can profit from functionality provided by the Verification sub-system.

10.5.3.3 Verification

The **Verification** sub-package is responsible for the verification of the functioning of the TDAQ system or any subset of it. It uses developer's knowledge to organize tests in sequences, analyse

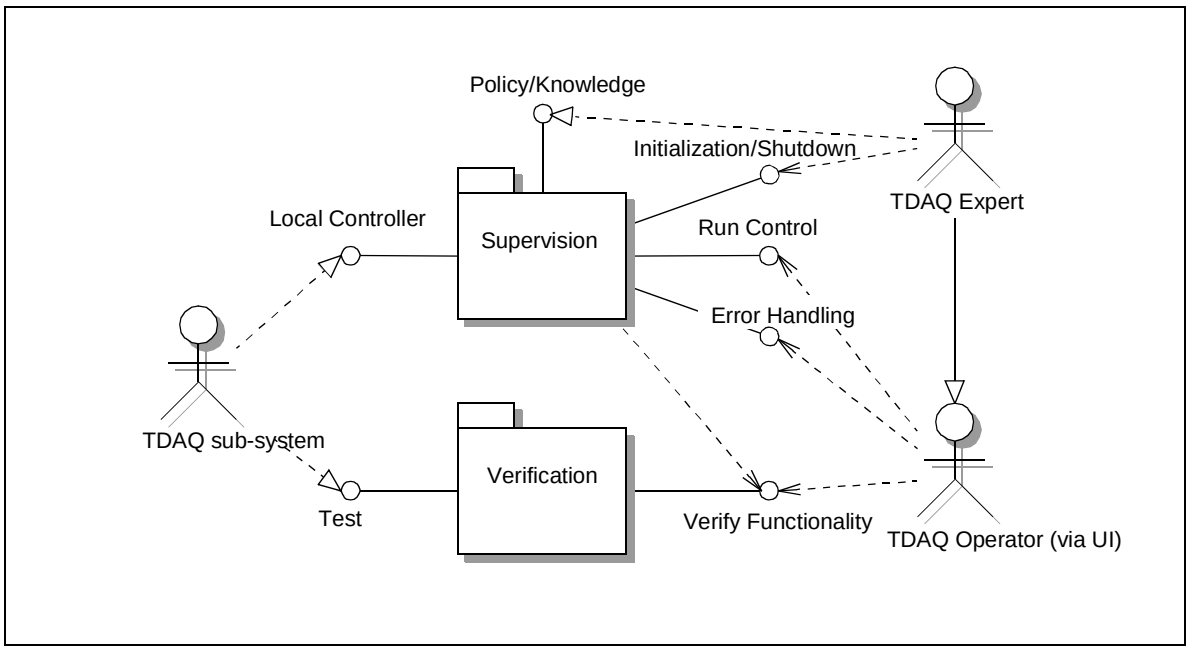

Figure 10-14 Interfaces of the Supervision and Verification sub-packages

test results, diagnose problems, and provide a conclusion about the functional state of TDAQ components.

A TDAQ sub-system developer implements and describes tests which are used to verify any software or hardware component in a configuration. This includes also complex test scenarios, where the component functionality is verified by the simultaneous execution of processes on several hosts. The sub-system uses the Process Management sub-package for the execution of tests.

The verification sub-system provides access to its functionality via the *Verify Functionality* interface. The sub-system is built on specific tests that probe the functionality of the Online Software, TDAQ sub-system or Detector functionality. These tests connect by the *Test* interface to the verification sub-system.

The Verification sub-package is used by the Supervision to verify the state of the TDAQ components during initialization or recovery operations. It can also be used directly by the Operator via the UI, as shown in Figure 10-14.

10.5.3.4 Process, Access and Resource Management systems

The Verification and Supervision sub-packages connect via interfaces to other Control sub-packages, as shown in Figure 10-15.

The **Process Management** provides basic process management functions in a distributed environment. These include starting, stopping, and monitoring processes on different TDAQ hosts. While its main users are the Supervision and Verification, it is also available for other user applications. It avoids the overhead of standard operating system tools and allows the management of the large number of processes involved in the TDAQ system.

The **Resource Management** is concerned with the allocation of software and hardware resources between running partitions. It is used by the Supervision and by the Process Management. It

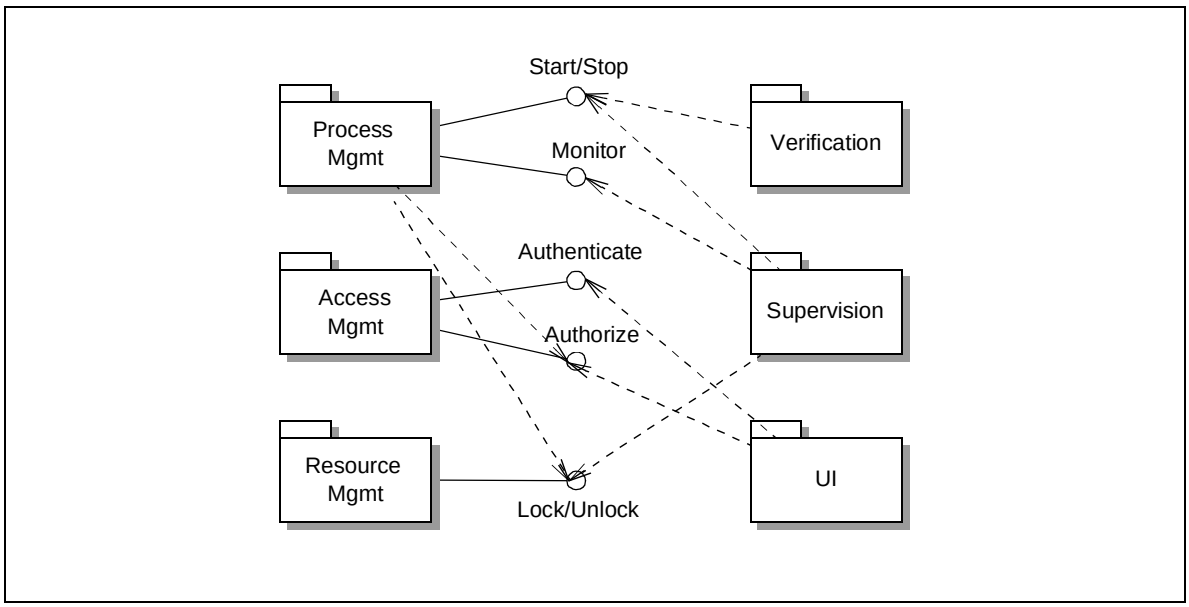

Figure 10-15 Interfaces of the Process Management, Resource Management and the Access Management

prevents the operator from performing operations on resources which are allocated to other users. This is of particular need during commissioning and testing phases, where many groups are working independently and have to share resources.

The **Access Management** is a general Online Software safety service, responsible for TDAQ user authentication. It implements an access policy, in order to stop non-authorized persons from corrupting the TDAQ system and interfering with data taking. This applies in particular to sensitive areas like the access to configuration databases, access to process management, remote access through the Web, etc. Apart from these cases, the limited access to the network at the experiment site is the main security measure.

10.5.4 Prototype evaluation

Prototype evaluations have been performed for a number of technologies. The initial choice was based on experience in previous experiments. Products were chosen that fit well in the proposed object-oriented software environment.

- A Run Control implementation is based on a State Machine model and uses the State Machine compiler, CHSM [10-14], as underlying technology.
- A Supervisor is mainly concerned with process management. It has been built using the Open Source expert system CLIPS [10-15].
- A verification system (DVS) performs tests and provides diagnosis. It is also based on CLIPS.
- A Java-based graphical User Interface (IGUI) is being used.
- Process Management and Resource Management are based on components which are part of the current implementation of the Online Software packages.

10.5.4.1 Scalability and performance tests

A series of large-scale performance tests has been performed to investigate the behaviour of the prototype on systems of the size of the final TDAQ system [10-9]. In several iterations the behaviour of the system was studied and limits were identified, the prototype was refined and tested until the successful operations of a system with a size in the required range was reached. Up to 1000 controllers and their applications were controlled reliably while meeting the performance requirements.

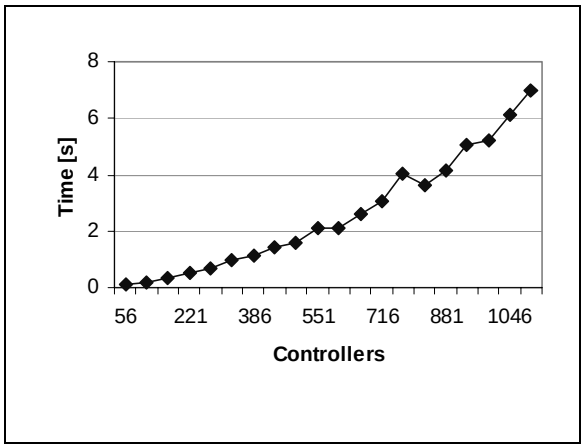

Figure 10-16 Time to perform three TDAQ state trannsitions for configurations in the range of O(10) to O(1000) controllers

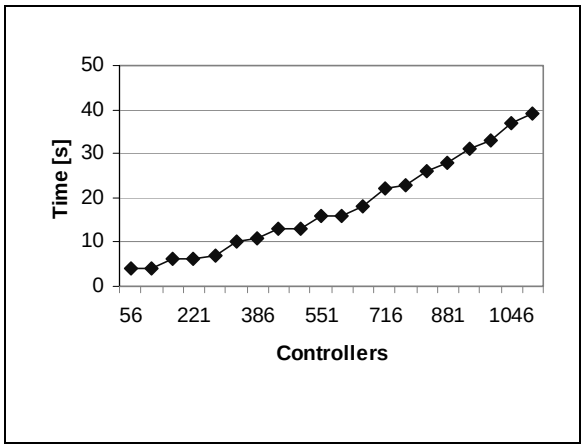

Figure 10-17 Time to start the processes for configurations in the range of O(10) to O(1000) controllers

Of note is the synchronized distribution of commands within the system. A command is sent by the root controller and propagates through the controller tree to the leaf controllers. These leaf controllers interface between the overall online control system and the specific tasks to be performed in the TDAQ system. The time to distribute a command from the top level and to obtain the confirmation from all controllers was measured. Figure 10-16 shows the time taken to perform three successive state transitions for a three level control hierarchy for configurations in the range of 10–1000 leaf controllers. A single transition takes about two seconds for 1000 controllers, a time well within expectations. In a real system each controller would perform its specific actions during such a state transition taking between a few seconds and up to tens of seconds. In relation to these values the overhead of the overall control system is small.

Also of note is the process control in the distributed system. To switch between various configurations, it will be necessary to start and stop many processes on many nodes. Although such an operation is not performed during physics data-taking, it will be of importance during development and calibration. Figure 10-17 shows the time to start the processes for a system with up to a thousand controllers. Detailed descriptions and further test results can be found in [10-9].

10.5.4.2 Technology considerations

During the evaluation of the prototype several shortcomings of the current system were identified. An important one is the lack of flexibility in the state machine implementation CHSM. In this context the expert system CLIPS [10-15] and related products were studied. The general-purpose nature of this product allows the various aspects of the supervision-like initialization, control, and error handling to be implemented. The knowledge base provides the basis for customizeble solutions, which can be specialized for different parts of the system. Another advan-

tage is the extensibility of CLIPS. It can be interfaced easily with other components of the Online Software system. Alternative products also under consideration are Jess [10-16], a similar expert system implementation written in Java and a commercial alternative, Eclipse by Haley Inc. [10-17].

Further alternatives have been investigated: SMI++ is a system that models the controlled domain as a system of interacting state machines [10-18] and is in use at several HEP experiments. Another possibility would be the use of general-purpose scripting languages, such as Python [10-19]. While each of these approaches has its particular merits, the evaluation showed that the CLIPS-based solution is better suited for our environment and is the favoured implementation choice.

10.6 Integration tests

10.6.1 Online Software integration and large-scale performance tests

Major releases of the integrated Online Software System were tested in several test series starting in the year 2000. All the major components were included. For the most recent series of tests in January 2003 [10-9], 222 dual-pentium PCs of the CERN IT LXSHARE test-bed were used. They were equipped with 600–1000 MHz Pentium III processors, 512–1024 Mbytes of memory, and were running the Linux RedHat 7.3 operating system.

The tests were aimed at verifying the overall functionality of the Online Software system on a scale similar to that of the final ATLAS installation (including up to 1000 leaf controllers) and at studying the system performance aspects in detail. A number of performance measurements for the Online Software components were also performed as a function of the system size. Details of these can be found in the component test descriptions on IS in Section 10.3.4.2, on Databases in Section 10.4.4.1, and on Control aspects in Section 10.5.4.1.

The tests followed the sequence of steps which are necessary for TDAQ start-up, running, and shut-down actions. Realistic conditions for the Online Software infrastructure were simulated by including additional traffic from monitoring activities in the detectors and sub-systems. This traffic included monitoring of statistical data and status information, error messages, histograms, and event data monitoring. Online Software specific aspects like the configuration and the control sub-systems were studied. Various special-purpose controllers and infrastructure configurations were prepared to help identify possible shortcomings.

Different types of limitations were found in consecutive test cycle phases. After first having overcome limitations due to design or implementation flaws, limits built into the underlying communication software layer ILU and the underlying TCP/IP system configuration (e.g. the maximum number of allowed connections) were reached. Their discovery led to the development of specific solutions for the Online Software situation. Other limits concerned operating system parameters and therefore required tuning of the operating system and the use of alternative system calls to realize an Online Software system of the size of the final ATLAS system. The successful results showed that the current Online Software system already scales to the size of the final ATLAS system as shown in Section 10.5.4.1. Detailed studies of the results provided important feedback on the architecture and design in the context of the iterative development process. Optimization should reduce the demand on the operating system and provide addi-

tional safety margins. Furthermore, error handling and fault tolerance aspects will have to be extended and the user interface adapted.

10.6.2 Event Filter Software tests involving the Online Software

The Event Filter is implemented as software processes running on a large processor farm which for reasons of practicality is split into a number of sub-farms. A prototype Event Filter supervision system, responsible for all aspects of software task management and control in the Event Filter farm, was successfully implemented using services from the prototype implementation of the Online Software. The prototype uses the Online Software Configuration Database, Run Control system, and Supervisor.

The run controllers are arranged in a hierarchical tree with one run controller per sub-farm which collaborates with a sub-farm Supervisor to control and monitor the Event Filter processes in the sub-farm. The top-level of the hierarchy consists of a Supervisor which is responsible for controlling and monitoring the processes in the supervision infrastructure only (sub-farm Run Controllers and sub-farm Supervisors) and a root Run Controller. Note that in the Online Software prototype only one Supervisor process would normally be used to start all processes described in the configuration database. For better scalability, the use of multiple Supervisors was tested. This has led to a much more scalable system which complies at least in part with the proposed future design of the Online Software to HLT interface [10-20].

The scalability of the prototype system was tested at the same time and using the same cluster as that used for the latest Online Software scalability tests described in Section 10.6.1. Two types of configuration emulating Event Filter farms were studied:

- The total number of processing hosts was kept constant (~200) but split into differing numbers of sub-farms (3, 8, 20).
- The number of sub-farms was kept constant (10) and the number of hosts per sub-farm was varied (5, 10, 20).

Each sub-farm processing host ran two filtering tasks. In the largest configurations studied, more than 1000 processes were under the control of the prototype Event Filter Supervision system, representing about 10–20% of the EF system which will be used for the first ATLAS run.

Detailed results are given in [10-21]. All the configurations studied could be successfully launched, marshalled through the run control sequence, and shut down. The only operation requiring a non-negligible amount of time was the initialization of the online infrastructure. However, this is done only once per data-taking period and therefore is not very significant. Nevertheless the Online Software team addressed it in the design presented above. All other run control transitions are performed in a reliable and sufficiently rapid way. Observed transition times vary from 0.1 s to about 3 s, depending on the nature of the executed transition. Note that these times are a combination of the Event Filter specific activities required at each transition and implemented in the sub-farm Run Controllers, plus the Online Software overhead. Owing to the tree architecture of the control system, the transition times do not strongly depend on the number of controlled hosts. A limitation was found in the usage of the Configuration Database which should be better adapted to the use of a large number of similar, but not identical, devices: these devices are likely to be changed during data-taking activities (added to or removed from the configuration, for example, to cope with hardware failures) and this operation should not interfere with the process of data taking. A dedicated user interface was developed to hide the complexity of the current implementation of the underlying configuration database.

In the design of the configuration databases presented above in Section 10.4.3 most of the issues raised here are already addressed. A closer integration of this user interface with that provided by the Online Software is foreseen.

Suggested modifications, as explained above, were taken into account in the design presented earlier in this chapter and the Online Software is therefore quite adequate for the control of the Event Filter.

10.6.3 Deployment in test beam

Major releases of the current Online Software have been in use in three test beam operation periods since summer 2000. This allowed it to run under realistic conditions with the detectors similar to the LHC ones and with physics triggers. Valuable feedback on the behaviour of the prototype software was gathered and has led to improvements in design and implementation. Details can be found in [10-22].

10.7 References

10-1 I. Alexandrov et al., *Online Software Requirements*, ATL-DQ-ES-0015 ()
https://edms.cern.ch/document/388874/

10-2 I. Alexandrov et al., *Online Software Architecture*, ATL-DQ-ES-0014 ()
https://edms.cern.ch/document/388865/

10-3 ILU home page,
ftp://ftp.parc.xerox.com/pub/ilu/ilu.html

10-4 CORBA home page,
http://www.omg.org/corba/

10-5 TAO home page,
http://www.cs.wustl.edu/~schmidt/TAO.html

10-6 MICO home page,
http://www.mico.org/

10-7 omniORB home page,
http://omniorb.sourceforge.net/

10-8 ORBacus home page,
http://www.iona.com/products/orbacus_home.htm

10-9 D. Burckhart-Chromek et al., *ATLAS TDAQ Online Software Large Scale Performance Tests January 2003*, ATL-DQ-TR-0012 ()
https://edms.cern.ch/document/392827/

10-10 LCG,
http://lcg.web.cern.ch/LCG

10-11 OKS User's Guide, ATLAS DAQ TN # 033,
http://atddoc.cern.ch/Atlas/DaqSoft/components/configdb/docs/oks-ug/2.0/pdf/OksDocumentation.pdf

10-12 POOL,
http://lcgapp.cern.ch/project/persist/

10-13 MySQL,
http://www.mysql.com/

10-14 P.J. Lucas, *CHSM, An Object Oriented language system for implementing concurrent hierarchical finite state machines*, Thesis, University of Illinois (1993)

10-15 CLIPS,
http://www.ghg.net/clips/CLIPS.html

10-16 Jess,
http://herzberg.ca.sandia.gov/jess/

10-17 Eclipse, Rule based programming language and inference engine, see under 'products' on
http://www.haley.com/

10-18 SMI++,
http://cern.ch/smi/

10-19 Python,
http://www.python.org/

10-20 D. Burckhart-Chromek et al., *HLT Online Software Interface*, ATL-D-ES-0009 ()
https://edms.cern.ch/document/363702/

10-21 S. Wheeler et al., *Test results for the EF Supervision*, ATL-DH-TR-0001 ()
https://edms.cern.ch/document/374118/

10-22 I. Alexandrov et al., *Systems Integration and Deployment in Test Beam and Large Scale Tests*, ATL-DQ-TR-0006 ()
https://edms.cern.ch/document/390995/

11 Detector Control System

11.1 Introduction

The principal task of the DCS is to enable the coherent and safe operation of the ATLAS detector. All actions initiated by the operator and all errors, warnings, and alarms concerning the hardware of the detector are handled by the DCS. For the operation of the experiment during data taking, a close interaction with the DAQ system is of prime importance. Safety aspects are not the responsibility of DCS. They are addressed by a dedicated Detector Safety System (DSS) and the global CERN-wide safety system, both of which DCS communicates with.

A Joint Controls Project (JCOP) [11-1] has been set up at CERN to address common points of controls for the four LHC experiments. Details of the scope and the responsibilities of JCOP are at present still being discussed.

The central part of DCS is described in this document and not the sub-detector-specific parts. The latter can be found in the relevant sub-detector TDRs. Central DCS comprises all HW and SW items needed for the global operation of the detector, i.e. starting from the equipment in the control room down to the sub-detector Controls Station and also including the Common Infrastructure Controls, see Section 11.4.1. This reflects also the attribution of responsibilities to the central DCS team and to the sub-detector DCS teams.

In this chapter, first the architecture and the logical organization of DCS are explained. Then follows a description of the different components and their interconnections. This leads to a discussion of the full read-out chain and its performance. As examples, some areas of application of DCS in the experiment are given. Finally, the communication of DCS with DAQ and with systems external to ATLAS is discussed. The utilization of the DCS, in combination with the Trigger and DAQ control, to provide the overall experiment control is described in Chapter 12.

11.2 Organization of the DCS

The functions and the scope of the DCS have been described in Chapter 1. In this section, the organization of the DCS is described. The architecture of the DCS and the technologies used for its implementation are constrained by both the functional requirements and the environmental aspects. The DCS consists of a distributed Back-End (BE) system running on PCs and of different Front-End (FE) systems. The BE will be implemented with a commercial Supervisory Control And Data Acquisition system (SCADA). The DCS FE instrumentation consists of a wide variety of equipment, ranging from simple elements like sensors and actuators up to complex Front-End Controllers (FECs) for specialized tasks. The connection between FE and BE is provided by fieldbus or LAN.

The equipment of the DCS will be geographically distributed in three areas as schematically shown in Figure 11-1, namely the main control room SCX1 at the surface of the installations, the underground electronics rooms USA15 and US15, and the cavern of the detector, UX15. USA15 and US15 are equivalent except that the first is accessible to personnel during beam operation and the second is not. The SCADA components will be distributed over the main control and the electronics rooms, while the FE equipment will be placed in USA15, US15, and UX15. The

relative independence of the operation of the sub-detectors leads to the hierarchical organization shown in Figure 11-1. In addition to the sub-detectors there is a part for Common Infrastructure Controls (CIC) that monitors services provided to the experiment as a whole, like electricity or ventilation. It also supervises equipment which is common to several sub-detectors like electronics racks.

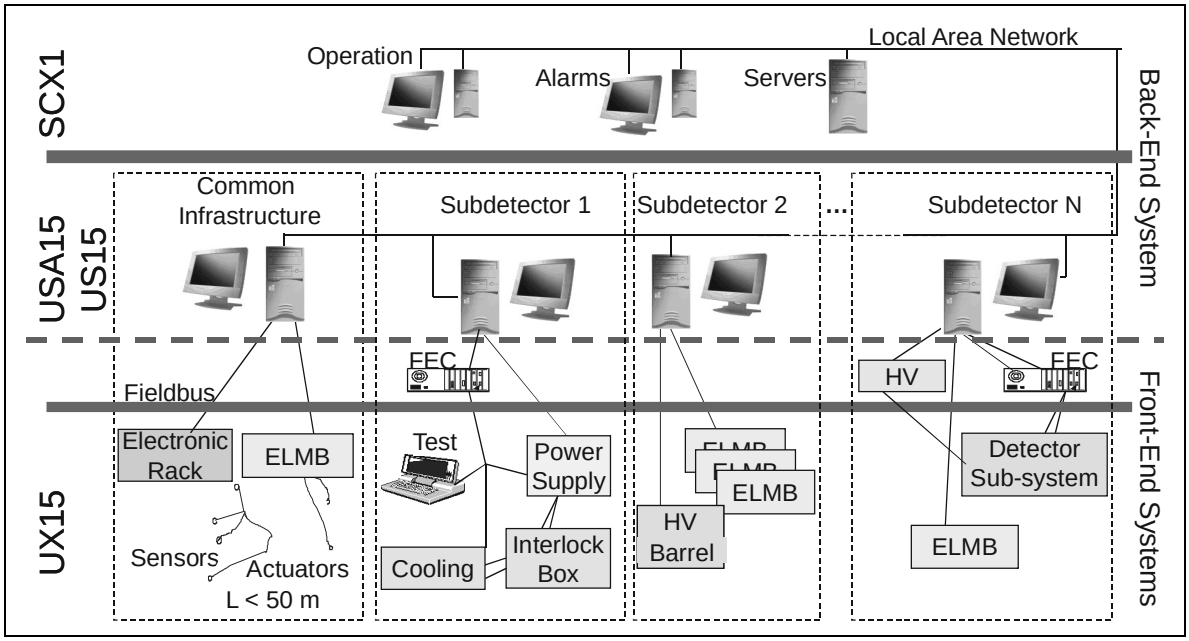

Figure 11-1 Geographical deployment of the DCS

The FE electronics in UX15 is exposed to a strong magnetic field of up to 1.5 tesla and to ionizing radiation. The DCS FE equipment will be located outside of the calorimeters, where the dose rate is of the order of 1 gray / year or less. This permits the use of selected Commercial Off The Shelf (COTS) components, which, however, have to be individually qualified for radiation following the 'ATLAS Policy on Radiation Tolerant Devices' [11-2]. FE equipment which can not operate under these conditions, like processor-based devices (e.g. FEC) or some types of power supplies, has to be accommodated in the electronics rooms.

The BE system consists essentially of PCs and is organized in a tree-like structure with several levels. It will be described in detail in Section 11.4. The highest level is situated in the control room provides all the tools needed for the integrated operation of the detector as a whole, like the operator interface, the alarm system, and various servers. The workstations in the levels below provide the online data and command handling, and full stand-alone operation capability for each sub-detector. This gives operational independence to the sub-detectors when needed, e.g. for commissioning, debugging, or calibration. The connection between all PCs of the BE over a LAN is part of the SCADA software.

11.3 Front-End system

The FE equipment connects directly to the detector hardware. It comprises sensors and actuators, digitizers, controllers and processors, commercial devices, and stand-alone computer-based systems. Concerning monitoring, the FE reads and digitizes values, processes them in some cases, and transfers the data to the BE. It also executes the commands that it receives from

the BE. The FE DCS of the sub-detectors is the responsibility of the sub-detector groups and is described in their TDRs. It consists of two categories of equipment, one being more general purpose and widely used in ATLAS, like the ELMB which will be explained in Section 11.3.1, and the other being more specialized, i.e. used for a dedicated task (like the alignment system of the Muon Spectrometer), which will be mentioned in Section 11.3.2. The first class is described in this TDR whereas for the second only the method and the point of connection to the BE is given.

The FE equipment is distributed over the whole volume of the detector with cable distances of up to 150 m. Two conflicting aspects constrain the distribution underground. Because of the radiation level, the magnetic field, the space limitations, and the inaccessibility of UX15 during beam time, it is preferable to locate the equipment in the electronics rooms. However, the complexity, cost, and technical difficulties of laying analog cabling over such distances, and the difficulties of transferring large amounts of digital data over long distances in such a harsh environment, imply the digitization and compression of the data as early as possible. This should be done on the detector in UX15 and only results of digitization, possibly with analysis of data and compression, should be transferred to USA15 or US15.

The harsh environment in the cavern limits the types of technologies that may be used. For data transmission to the BE the CAN fieldbus [11-3] has been chosen amongst the CERN-recommended fieldbuses. CAN equipment is very robust, has excellent error detection and recovery, can be used over large distributed areas, does not use components sensitive to magnetic field, and has good support from industry.

11.3.1 Embedded local monitor board

An electronics module called Embedded Local Monitor Board (ELMB) [11-4] has been developed for standard analog and digital input/output. No such commercial devices exist that can operate in the hostile environment of the cavern. Also because of cost and space limitations (several 100 000 channels are needed) an optimized design was required.

The ELMB is a single, credit-card-sized electronics board that may either be embedded into custom front-end equipment or be used in stand-alone mode. It comprises 64 high-precision analog input channels of 16-bit accuracy and it provides 8 digital inputs, 8 digital outputs and 8 configurable (either input or output) lines. Its serial port can drive additional devices like a digital-to-analog converter. As it has very low power consumption, it can be powered from the electronics rooms via the fieldbus cable.

The firmware loaded into the ELMB includes instructions for both driving the input/output functions and the communication over the CAN field bus using the higher level protocol, CANopen. Standard configuration software tools provide easy ‘plug and play’ functionality for the user. The ELMB fulfils the majority of standard I/O requirements of the ATLAS sub-detector applications in terms of functionality, accuracy, and stability. It has been tested and qualified to operate in the radiation and magnetic field environment present in UX15. Its error detection and recovery procedures have been validated in long-term operations without manual intervention.

Typical applications of the ELMB are to directly read from/write to sensors and actuators, or to control more complex devices like power supplies, gas devices, cooling systems, or whole detector elements like chambers. In the second class of applications, the ELMB is normally fully integrated in the device. In several cases, like the monitoring of the gas flow or the control of the

SCT power supply system, which require additional functionality like histogramming, specialized ELMB software has been developed by the users, which replaces its standard firmware.

11.3.2 Other FE equipment

An effort is made to standardize FE devices like high-voltage units, low-voltage power supplies, racks, PLCs etc. As all four LHC experiments have similar requirements, such front-end equipment will be supported in the framework of JCOP. This includes generic tools and libraries for configuration, data readout into SCADA, and general supervision and operation. The gas systems also fall into this category of equipment.

Specialized, non-standard FE equipment like alignment and calibration systems is usually self-contained. Often large quantities of data have to be read and processed. The results are then transmitted to DCS for further treatment, monitoring, and storage. Therefore for each such system a connection point and a protocol have to be defined. In many cases, this is TCP/IP over a LAN, but also dedicated drivers running in the BE will be used.

11.4 The Back-End system

The functionality of the BE system is two-fold. It acquires the data from the FE equipment and it offers supervisory control functions, such as data processing and analysis, display, storage, and archiving. It also provides handling of commands, messages, and alarms.

The BE system is organized hierarchically to map the natural structure of the experiment, i.e. sub-detectors, systems, and sub-systems. The BE hierarchy allows the dynamic splitting of the experiment into independent partitions as defined in Section 3.4, which can be operated in stand-alone or integrated mode. The operation of the different sub-detectors will be performed by means of Finite State Machines (FSMs), which will handle the states and transitions of the different parts of the detector, as explained in Chapter 12. The coordination of the different partitions will be performed by means of commands and messages.

11.4.1 Functional hierarchy

In order to provide the required functionality, the BE system of the DCS will be organized functionally in three levels as shown in Figure 11-2. Overall operation of the detector can be performed at the level of the Global Control Stations (GCSs), while data processing and command execution are handled at the lower levels. Archiving of data and alarms, and logging of commands and incidents will be provided at every level. Remote access to a well-defined set of actions to be performed by the two upper levels of the BE hierarchy will also be provided subject to proper access authorization.

Global Control Stations

The overall control of the detector will be performed at the top level of the BE system, which consists of the Global Control Stations (GCSs). They provide high-level monitoring and control of all sub-detectors, and of the technical infrastructure. Detailed control of the sub-detectors is provided only at the next level, i.e. the Sub-detector Control Stations (SCSs) level. The functions performed by the GCSs are the following. They assemble all status information available and

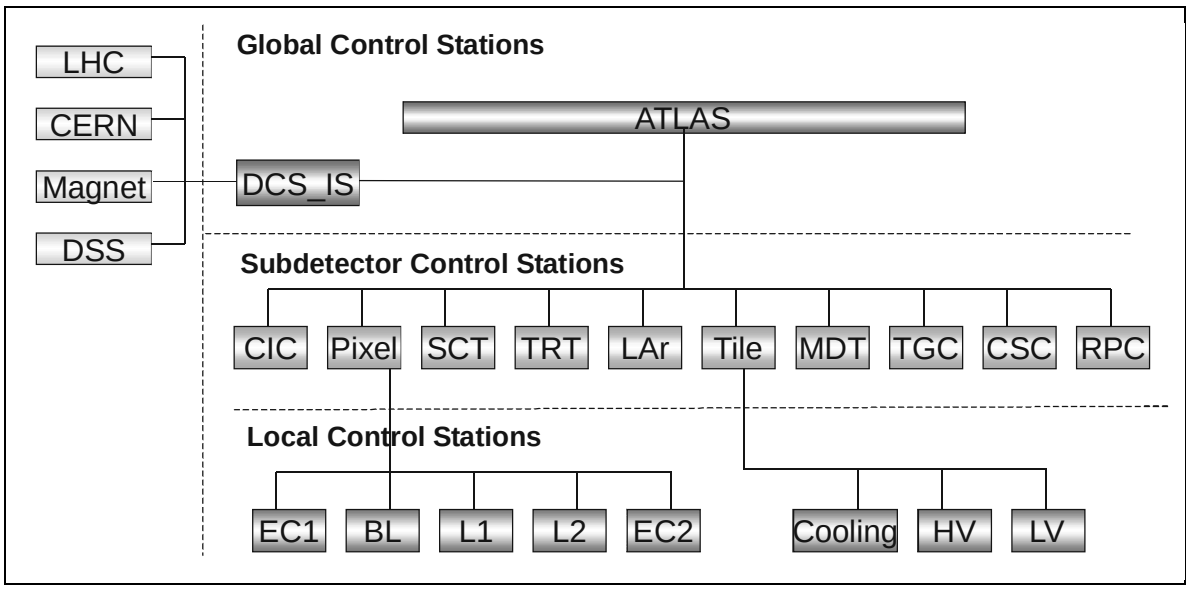

Figure 11-2 Hierarchical organization of the Back-End system of the DCS in three functional layers

present it to the operator in a hierarchical fashion. All anomalies of operation like warnings, alarms, etc. are collected, displayed, and archived. Actions from the operator like acknowledgement can be requested. The GCSs may trigger actions themselves or ask the operator to do so. Any parameter in the system can be inspected in real-time and its history can be displayed. This also includes the browsing of log files. Pre-defined commands can be given to the sub-detectors and to their sub-systems. Settings can be changed for selected parameters. The DCS Information Service (DCS_IS) handles the communication with external systems like the LHC accelerator, the CERN infrastructure, the magnets, and the DSS. Web access and database services will be handled by dedicated servers. Bi-directional data exchange between the DCS and the TDAQ system will be possible at this level but commands from TDAQ will only be sent to the level below.

Sub-detector Control Stations

The SCSs form the middle level of the BE hierarchy. There will be one SCS per sub-detector and an additional SCS to handle the monitoring of the common infrastructure of the experiment, the CIC. The latter will be directly interfaced with the DCS_IS at the GCS level. It is also foreseen to have a direct connection from the SCSs to the DCS_IS to provide the different sub-detectors with the status of the external systems, as well as with the environmental parameters. All actions on a given sub-detector that are possible from the GCSs are provided at this level as well. In addition, the SCS allows the full, local operation of the sub-detector by means of dedicated graphical interfaces. The SCSs handle the co-ordination of all sub-systems in the Local Control Station (LCS) layer, and they are responsible for the validation of all commands, which are issued either by the GCS or from the TDAQ run control. They translate global commands, such as ramping up a high-voltage system, into detailed commands, e.g. for the individual power supplies, and send these to the LCS layer below for execution. They assemble the overall status of the sub-detector and pass it on to the GCS and to TDAQ.

Local Control Stations

The bottom level of the BE hierarchy is constituted by the LCSs, which handle the low-level monitoring and control of the different systems and services of the detector. The organization of this level for a given sub-detector can be according to either geographical or functional criteria.

The former follows the topological composition of the sub-detector, i.e. barrel, end-cap, etc., whereas the latter combines functions or services of the sub-detector in one LCS, like cooling, high-voltage gas, etc. It is important to note that although the gas system will be built by the CERN Gas Group, from the DCS point of view, gas systems are regarded as a sub-detector service since each sub-detector is responsible for the monitoring and control of all parameters relevant for operation. The LCS level of the hierarchy is directly interfaced to the FE system. Besides the readout and control of the equipment, it also performs calculations and fine calibration of the raw data from the FE and comparison of the values with preconfigured thresholds for the alarm handling. The stations placed at this level will execute the commands received from the SCS in the layer above and they can also execute autonomously predefined actions if required.

11.4.2 SCADA

SCADA systems are commercial software packages normally used for the supervision of industrial installations. They gather information from the FE layer, process these data, and present them to the operator.

Besides the basic functionalities like Human–Machine Interface (HMI), alarm handling, archiving, trending, or access control, SCADA products also provide a set of interfaces to hardware, e.g. fieldbuses and PLCs, and to software, e.g. Application Program Interface (API) to communicate with external applications, or connectivity to external data bases via the Open or Java Data Base Connectivity protocols (OBDC and JDBC, respectively).

SCADA products provide a standard framework for developing applications and lead in this way to a homogeneous DCS. This also saves development and maintenance effort reducing the work for the sub-detector teams. In addition, they follow the evolution of the market, taking advantage of advances in technology, like operating system or processor platforms.

11.4.3 PVSS-II

A major evaluation of SCADA products [11-5] was performed at CERN in the framework of JCOP, which concluded with the selection of PVSS-II [11-6] from the company ETM. This product will be used for the implementation of the BE systems of the four LHC experiments.

PVSS-II is device-oriented, where variables that belong logically together, are combined in hierarchically structured data-points. For example, variables could be channels in a high-voltage crate and grouped as such in a PVSS-II data-point. Device-oriented products adopt many properties like inheritance and instantiation from object-oriented programming languages. These features facilitate the partitioning and scalability of the application. PVSS-II includes a powerful API. A driver development toolkit is also available to interface to custom applications or special equipment.

PVSS-II is designed as a distributed system. The single tasks are performed by special program modules called managers. The communication between them takes place according to the client–server model, using the TCP/IP protocol, and is entirely handled by PVSS-II. The internal communication mechanism of the product is entirely event-driven. This characteristic makes PVSS-II especially appropriate for detector control since systems that poll data values and status at fixed intervals present too big an overhead and have too long reaction times resulting in lack of performance.

The managers can be distributed over different PCs running either Microsoft Windows or Linux. The latter has been an important point in the selection of this product since the DAQ system of the ATLAS experiment is not being developed under Windows but under Linux and other Unix flavours.

PVSS-II allows the supervisory software to be split into smaller applications communicating over the network as imposed by the distribution of the DCS equipment over different locations.

The functionality provided by PVSS-II has been successfully tested in a number of test-beam applications with various sub-detectors. Aspects like the scalability and the performance of the product are currently being addressed in different tests performed in collaboration with JCOP.

11.4.4 PVSS-II framework

PVSS-II has proven to be a good basis to build the LHC experiment controls. However, it lacks some functionality required for the control of high-energy physics detectors, and some generic tools and libraries are needed to develop a homogeneous and coherent system. Therefore an engineering framework on top of PVSS-II is being developed in the context of JCOP. This comprises a set of guidelines, tools, libraries, and components needed by all four LHC experiments. This framework will lead to a significant reduction in the development and maintenance work to be done by the sub-detector teams and to a more homogeneous system. It also addresses questions of interoperability of the different components.

This framework includes programs to create the readout structure for standardized devices such as high-voltage systems and to configure them. Operational tools for displaying data or alarms are also in the scope of the framework, as well as drivers to connect commonly used hardware and software.

The ELMB has been integrated into PVSS as a component of the JCOP framework in order to facilitate the usability of the ELMB and to ensure the homogeneity of the SCADA software. The ELMB component provides all PVSS infrastructure needed to set up and to operate the ELMB. It also comprises a so-called ‘top-down’ configuration tool, which handles the configuration of the software interface of the ELMB to the BE.

Some services, tools, and even applications can be common for all PVSS-II stations in the BE. Examples are access to the conditions database, advanced data display, or the alarm system. These are also expected to be provided by this framework.

11.5 Integration of Front-end and Back-end

Several methods are possible for the connection between the BE and the FE systems:

- Dedicated driver: PVSS-II provides drivers for modbus devices, PROFIBUS, and some other hardware. It also contains a driver development toolkit that allows users to write a driver for their special hardware.
- OPC client–server connection: OPC [11-7], which is the abbreviation for ‘Object linking and embedding for Process Control’, is a widely used industrial standard. Most commercial low- and high-voltage systems are supplied with an OPC server.

- DIM software: DIM, which is the abbreviation for ‘Distributed Information Manager’ has been developed at CERN. It is a communication system for distributed and multi-platform environments and provides a network-transparent inter-process communication layer.

Because of its widespread usage and the good industrial support, OPC has been chosen as the main interface from SCADA to hardware devices. The main purpose of this standard is to provide the mechanism for communicating with numerous data sources. OPC is based on the DCOM technology of Microsoft and hence requires the Windows operating system. The specification of this standard describes the OPC Objects and their interfaces implemented by the OPC server. The architecture and specification of the interface was designed to facilitate clients interfacing to a remote server. An OPC client can connect to more than one OPC Server, in turn an OPC Server can serve several OPC clients. Consequently all OPC objects are accessed through interfaces.

11.5.1 OPC CANopen server

CANopen, which is used by the ELMB, is a high-level protocol for CANbus communication. This protocol is widely used. CANopen standardizes the types of CANbus messages and defines the meaning of them. It allows the same software to manage CAN nodes of different types and from different manufacturers. Several CANopen servers exist on the commercial market, but all of them are tailored to specific hardware interface cards and they provide only a subset of the CANopen functionality required by the ELMB. For these reasons, an OPC CANopen server has been developed to connect the ELMB to SCADA.

This OPC CANopen server works as the CANopen master of the bus, handling network management tasks, node configuration, and data transmission to the OPC client. It consists of two parts.

- The main part contains the OPC functions themselves. It implements all the OPC interfaces and main loops. Any application interacts with this part through interfaces. The CANopen OPC server transmits data to a client only on change, which results in a substantial reduction of data traffic.
- The second part is hardware dependent. It communicates with the CANbus driver of the CAN interface card chosen and controls the CANopen devices.

Several buses with up to 127 nodes each, in accordance with the CANopen protocol, can be operated by this OPC CANopen server. The system topology in terms of networks and nodes per bus is modelled at start up time in the address space of the OPC CANopen server from a configuration file created by the ELMB configuration tool mentioned in Section 11.4.4.

11.6 Read-out chain

The complete standard read-out chain [11-8] ranges from the I/O point, e.g. sensor or actuator, to the operator interface and is composed of the elements described above: ELMB, CANopen OPC Server, and PVSS-II. In addition to the data transfer, it also comprises tools to configure and to manage the bus. PVSS-II describes the system topology in terms of data-points organized in a hierarchy. The data-points could be representations of a CANbus, ELMBs, or sensors. These data-points are connected to the corresponding items in the OPC server in order to send the ap-

propriate CANopen message to the bus. In turn, when an ELMB sends a CANopen frame to the bus, the OPC server decodes it and sets the respective item in its address space, which transmits the information to a data-point in PVSS. The different elements of the read-out chain perform the functions described below.

The ELMB digitizes the analog inputs and drives the digital input and output lines. Different settings can be applied to the ADC, including choosing as data output format either raw counts or calibrated micro-volts. Data are sent either on request from the supervisor, or at predefined intervals, or when they have changed. As the ELMB is in most cases exposed to ionizing radiation, its firmware checks also for possible radiation-induced errors (e.g. memory or register changes) and tries to correct them.

The OPC server transmits data together with quality information from the CANbus to PVSS-II as soon as they have changed. It can optionally perform calculations, e.g. converting the data into physical quantities in appropriate units.

The SCADA system applies the individual calibration procedures and compares the data with pre-defined thresholds. In this way warnings, alarms, and automatic actions are established. SCADA also archives the data and allows their visualization.

11.6.1 Performance of the DCS readout chain

Safe operation of ATLAS require the complete monitoring of the detector every few seconds. In addition, the average CANbus occupancy in normal operation is required to be kept well below 50% in order to cope with higher bus loads in case of major problems with the equipment being monitored. In order to investigate the performance and the scalability of the DCS read-out chain up to the size required by ATLAS, a CANbus system consisting of six CANbuses with 32 ELMBs each was set up [11-9].

The aim was to study the behaviour of the system in order to optimize the read-out parameters and to find the performance limits of such a read-out chain. These limits define the granularity of the system in terms of number of ELMBs per CANbus and the number of buses per PVSS system. In particular, the following was investigated.

- Remote powering of the ELMB nodes via the bus. The radiation levels in the detector cavern force the power supplies to be placed in the underground electronics rooms. Therefore, the power for the nodes will have to be fed remotely via the CANbus with distances of up to 150 m.
- Bus load, which determines the number of nodes per bus under normal operation. The data traffic on the bus has to be uniformly distributed over time in order to keep the bus load low. In ATLAS the bus occupancy should be kept well below 50%, even during peak times.
- Optimization of the work distribution amongst the different processing elements in the read-out chain. The functions to be performed by the ELMB, CANopen OPC server, and PVSS have to be evenly distributed to ensure adequate load of each of these components and to avoid bottle-necks.
- Optimization of the system performance by tuning the different software settings such as the update rates for OPC and the read-out frequency.
- Determination of the overall read-out speed.

The set-up employed in the test is shown in Figure 11-3. The system of CANbuses was operated from PVSS-II using the CANopen OPC server and the ATLAS final-choice CAN interface card [11-9]. The bus lengths were 350 m in all cases, in order to fulfil the ATLAS requirements with a broad margin. Up to 32 ELMBs were connected at the end of each CANbus. The total number of channels in this system was 12 288 analog inputs, 3072 digital outputs, and 1536 digital inputs. It is important to note that the number of channels in the set-up described here is comparable to that of some large applications in ATLAS, e.g. some sub-detector's low and high voltage systems. During the test, the read-out rate was increased in order to push the system to the limit. When big bursts of data arrive at PVSS-II very rapidly, the different messages can be internally buffered. Two different situations can be distinguished:

- *'Steady run'*, where all messages sent by the ELMBs to the bus are stored to the PVSS-II database and are processed in real time, i.e. no buffering takes place at the PVSS-II or OPC level.
- *'Avalanche run'*, where the fastest possible read-out rate is maintained for a certain period of time, typically a few minutes. Under these circumstances, although all messages are archived to the PVSS database, the data flow is so high, that messages cannot be treated in real time. They are buffered at the SCADA level, leading to an increase in the memory usage. It is important to note that although long-term operation under these conditions would not be possible, such a situation can occur in the case of major problems with the equipment being monitored. This can happen, for example, with a power cut, where all the parameters change at the same time. The read-out system must be able to handle this for a short period of time and to recover from it.

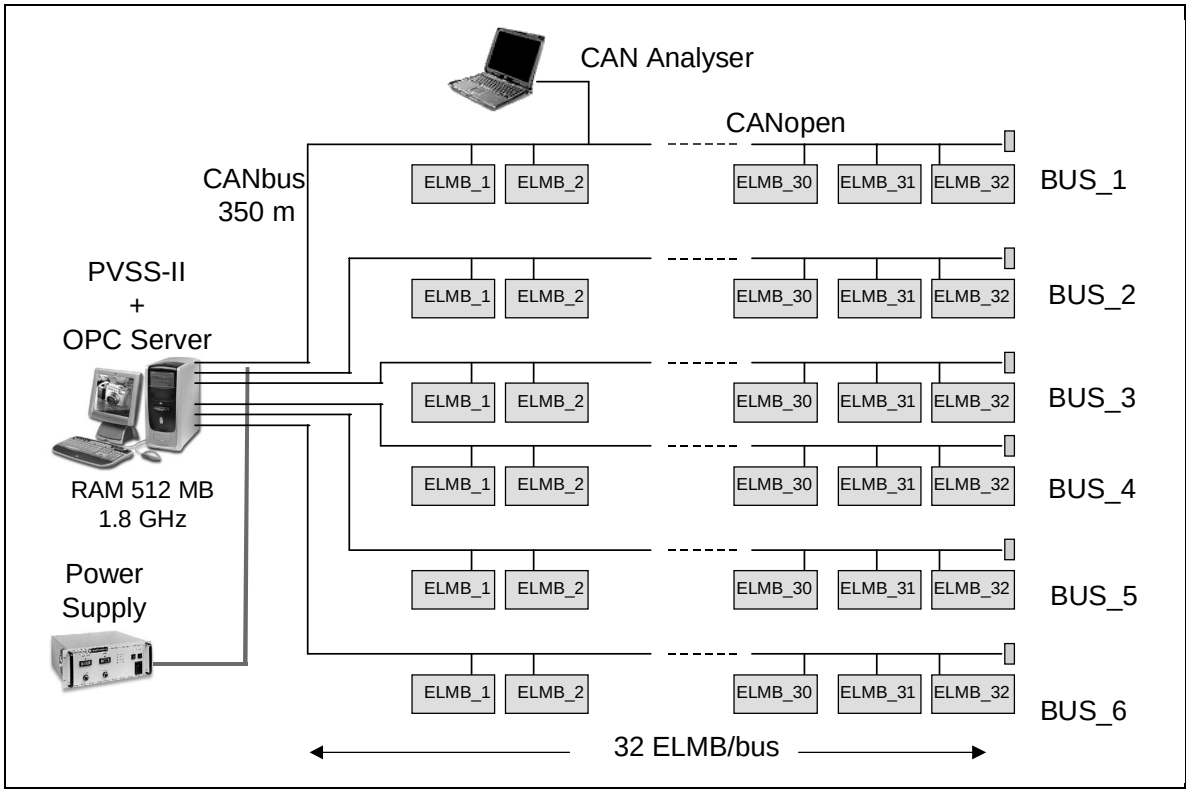

Figure 11-3 ELMB full branch test set-up

On the system of Figure 11-3, a minimum read-out cycle of 30 s is needed to stay in *'steady run'* conditions. During an *'avalanche run'*, where buffering is allowed, a read-out cycle of 8 s can be

sustained for several minutes. The examination of the CPU usage showed that in both cases the limitation is due to the work of the PVSS-II managers. These results indicate that the operation of the read-out is constrained by the performance of PVSS-II, i.e. by the PC performance. It has to be pointed out that the situation described is the worst case possible. During normal operation data will be sent only when they change, i.e. they are evenly distributed, and can be treated in real time. Moreover, the performance can be improved by using a more powerful CPU and by attributing fewer channels to a station. In this way, the required read-out interval can be achieved.

11.6.2 Long-term operation of the read-out chain

The DCS has to operate without any interruption and hence must recover automatically from any errors, including those induced by radiation. The long-term operation of the full read-out chain was tested with a number of ELMBs in a radiation environment similar in composition to the one expected in the ATLAS cavern, though at a much greater dose-rate [11-10]. This environment allows the testing of the various error recovery procedures implemented at the different levels of the read-out. A detailed description of the requirements to cope with the different effects induced by the radiation on the ELMB electronics can be found in [11-4]. A CANbus of more than 100 m was connected to a PC running the standard read-out chain. The test ran permanently for more than two months, which is equivalent to about 300 years of operation of an ELMB at the expected ATLAS dose rate. The following precautions and recovery procedures were implemented. The CAN controller in the ELMB ensures that messages are sent correctly to the bus and will take any necessary action if errors are detected. Bit flips, which were observed at the ELMB by special test software, were also handled correctly by the ELMB firmware. The OPC server ensures that all ELMBs on a bus are kept in the operational state by monitoring all messages and sending a software reset when required. At the highest level, PVSS scripts were utilized to detect an increase of the current consumption on the bus — which would be an indication of damage by radiation — and to reset the ELMBs if communication was lost. Through this script the power supply for the bus was also controlled, allowing for hardware resets by cycling the power. The system ran stably without any user intervention for the total time of the test.

11.7 Applications

The components and tools described above are used to build the applications that control and monitor the experiment equipment. The applications for the supervision of the sub-detectors are the responsibility of the sub-detector groups and are described in the relevant TDRs and in [11-11]. All equipment that does not belong directly to a sub-detector will be supervised by the SCS of the CIC, which is hierarchically situated at the level of a sub-detector. The CIC monitors the sub-systems described below.

All racks, both in the electronics rooms and in the cavern will contain a control unit based on the ELMB. It monitors the basic operational parameters like temperature, air flow, cooling parameters, etc. and also user-defined parameters. Some racks will have the option of controlling the electric power distribution. The crates housed in these racks usually have their own controls interface.

General environmental parameters like temperature, humidity, pressure, radiation level etc. will also be monitored by the CIC. Parameters of the primary cooling system also belong to this category. The individual sub-detector cooling distribution systems, however, are supervised by the corresponding sub-detector SCS.

All information collected is available to all other PVSS stations via the central DCS information server. A subset of it will also be transmitted to the DAQ.

11.8 Connection to DAQ

The communication between DAQ and DCS is very important for the coherent operation of the detector and data acquisition. This is achieved through a synchronization of the two systems on multiple levels. For this synchronization to be possible, the following types of information are exchanged between DAQ and DCS:

- bi-directional exchange of data, e.g. parameter values and status information;
- transmission of DCS messages to DAQ, e.g. alarms and error messages;
- sending of commands from DAQ to DCS and providing feedback about their execution.

The functionality of the DAQ–DCS Communication (DDC) package [11-12] has been split along the same lines as the data exchanges. The DDC software components are thus, Data Transfer (DDC-DT), Message Transfer (DDC-MT), and Command Transfer (DDC-CT). These components are described and their interfaces given in the following sub-sections.

The DDC design incorporates two fundamental concepts of major importance in ATLAS TDAQ: the concept of partitioning, which implies that the DDC functionality has to be provided for each TDAQ partition and work independently from other partitions; and the concept fault tolerance and error recovery, which for the DDC implies that if the communication with either a PVSS workstation or the DAQ is broken, it waits until they become available again and reestablishes the connection.

The DDC is developed on top of the interfaces to the Online Software (see Chapter 10) and PVSS described below. Their relationship is shown in Figure 11-4.

The DDC interface to DAQ is implemented using the following Online Software services: the Information Service, which allows the sharing of run time information; the Error Reporting Service, which distributes application messages; and the Run Control package, which uses a FSM to represent, control, and synchronize the states of TDAQ sub-systems within a partition. Also, DDC is configured using the Online Software Databases service. The DDC configuration includes the set of parameters, messages, and commands to be exchanged between DAQ and DCS.

PVSS-II has a powerful API allowing full direct network access to the run-time database that holds, for example, the DCS parameter values. This API is used as the DDC interface to DCS.

The prototype of the DDC package was used at test-beam during 2001 and 2002 and has been demonstrated to work in a satisfactory and reliable manner.

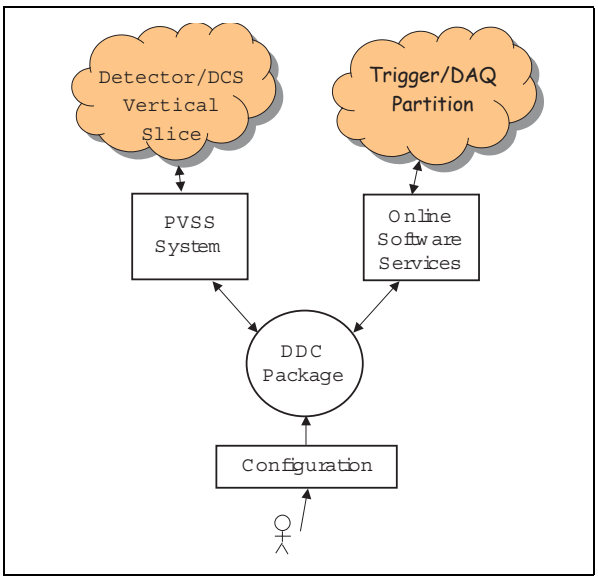

Figure 11-4 DDC package connecting DAQ and DCS

11.8.1 Data transfer facility (DDC-DT)

The data exchange in both directions is implemented via the Information Service. The application keeps the data elements, i.e. the parameters of the systems, which are declared in the DDC-DT configuration, synchronized at both sides. DDC implements this functionality by using the subscription mechanisms provided by DAQ and DCS. DDC also allows a single read of DCS data on request. Figure 11-5 shows the interfaces being used.

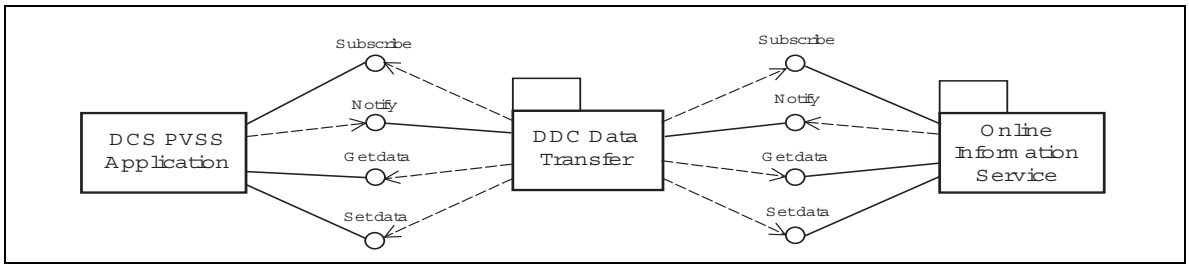

Figure 11-5 DDC data transfer interfaces

11.8.2 Message transfer facility (DDC-MT)

The DCS message transfer to DAQ is implemented via the Error Reporting Service. The interfaces used are shown in Figure 11-6. DCS messages defined in the DDC configuration, like high-voltage alarms or information messages, are sent to DAQ.

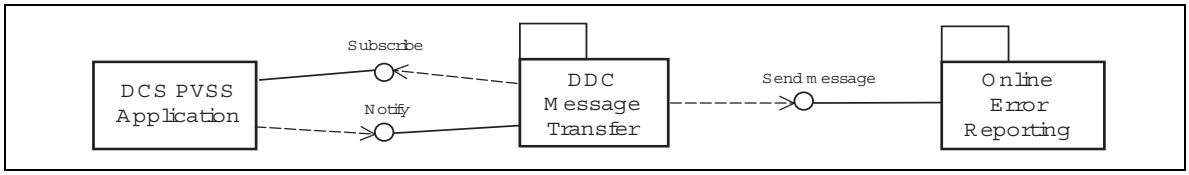

Figure 11-6 DDC message transfer interfaces

11.8.3 Command transfer facility (DDC-CT)

The DDC-CT subsystem, shown in Figure 11-7, is implemented as a dedicated run controller to be included as a leaf in a TDAQ partition run control tree. Like any other TDAQ run controller, DDC-CT implements the TDAQ FSM. When a state transition is requested, the associated command is then sent to the DCS and waits for its execution. The actions to be carried out for any command are the entire responsibility of the PVSS-II application. The DDC-CT state transition is acknowledged only after the confirmation of the successful execution of the command on the DCS side. The mapping of the run control transitions onto commands for DCS is done in the DDC-CT configuration.

In addition, the DDC-CT allows for the asynchronous transmission of commands to the DCS which are not associated to transitions in the TDAQ FSM. These are called '*non-transition*' commands and are implemented via the Information Service. Such a command may be issued at any time by any TDAQ application, including the parent run controller.

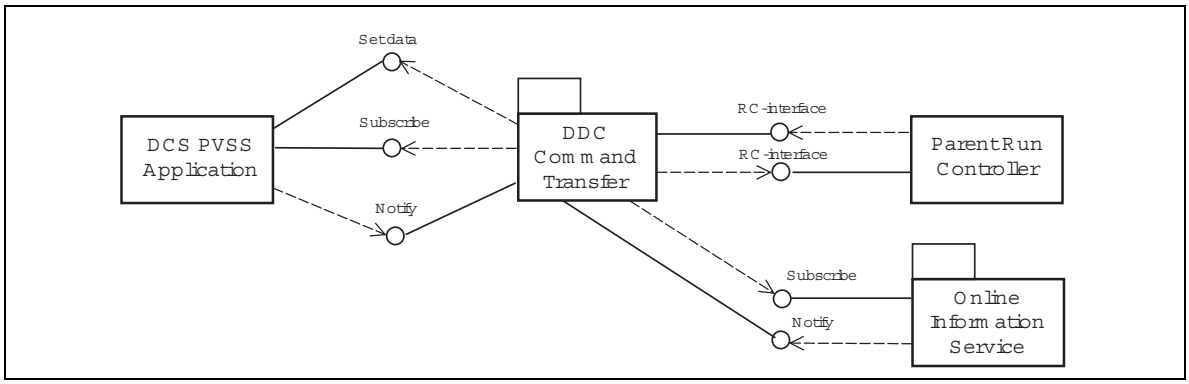

Figure 11-7 DDC command transfer interfaces

11.9 Interface to external systems

The term external system designates a system that has its own control system, and with which ATLAS has to interact. This communication will be handled by the DCS. The most notable example of such a system is the LHC accelerator.

The connection will support bi-directional information exchange and, in some cases, include sending and receiving of commands. This interface will be common for the four LHC experiments and it is expected to be developed in the framework of JCOP.

11.9.1 LHC

Data have to be transmitted in both directions between ATLAS and the LHC accelerator. The following are examples of data that the LHC has to provide to ATLAS:

- overall status of the accelerator, e.g. shutdown, filling, ramping, tuning, stable beams;
- beam parameters, e.g. energy, the different types of background, luminosities, beam positions, beam sizes, and profiles;

- ancillary parameters, e.g. collimator settings, magnet settings, and vacuum values upstream and downstream of the experiment.

This information is used in ATLAS for the validation of commands going to the sub-detectors, for interlocks, for the operation of physics data taking as described in Section 12.4.3, and eventually also for the offline physics analysis. These data will be logged by DCS.

Examples of data that ATLAS will send to the LHC are:

- various types and distribution of backgrounds as observed in the sub-detectors;
- luminosities measured and various trigger rates;
- beam positions measured by the ATLAS tracking sub-detectors;
- magnet status, since the solenoid especially may have an effect on the beams;
- status of the detector with respect to the possibility of injecting particles or dumping beam. Indeed the sub-detectors have to be in a state which is safe for these operations.

This information is needed for the general operation of the accelerator. Some of the data are particularly important for the fine-tuning and optimization of the beams.

A complete list of all parameters to be exchanged is not yet available. In particular it has not been decided whether individual bunch information will be transmitted in this way. Although the exchange of many of these parameters is needed only during data taking, a subset of this information, like the backgrounds and radiation doses is needed for periods like machine development. This information is required regardless of the state that ATLAS is in. This is one main reason why this communication will be handled by the DCS on the ATLAS side and not by the DAQ system.

11.9.2 Magnet system

It has been decided that all magnet systems of the four LHC experiments will be controlled by the tools selected for the LHC magnets and cryogenics. Therefore the ATLAS magnets are considered by DCS as an external system. Information exchange is foreseen in the same way as for the LHC accelerator, except that no actions happen. The operational data from the ATLAS magnets like the magnetic field value, currents and status, will be analysed, displayed, and stored by DCS like any sub-detector parameter.

11.9.3 CERN technical services

The technical services around the ATLAS detector include cooling, ventilation, electricity distribution, environmental radiation monitoring, the CERN safety system, etc. ATLAS needs access to some parameters of these services to ensure safe operating conditions for the sub-detectors. In particular it is essential to get early indications of problems with services like cooling. In such cases DCS can take precautions in order to avoid possible damage to the detectors. In some cases automatic feedback from ATLAS about its needs and usage of the service may be required.

11.9.4 Detector Safety System

As previously mentioned, the DCS is responsible neither for the security of the personnel nor for the ultimate safety of the equipment. The former is the responsibility of the LHC-wide hazard detection systems, whereas the latter has to be guaranteed by hardware interlocks internal to the sub-detectors and by the DSS [11-13]. One of the main purposes of DSS is to provide the correlation amongst the sub-detectors and the experimental infrastructure as far as safety is concerned. DSS consists of a front-end system with stand-alone capability, which is based on PLCs. A BE system, implemented by PVSS-II, provides supervisory functions, but it is not needed for the real-time operation. Bi-directional information exchange between DCS and DSS is provided on the PVSS-II level. Actions are triggered only in one direction, from DSS to DCS. In this way, the DCS can not disturb the operation of the safety system, but early information about minor problems that the DSS may detect enables DCS to take corrective actions or to shut down the problematic part of the detector before the problem escalates and DSS has to take a higher level and maybe more drastic action.

11.10 References

11-1 A. Daneels and W. Salter, *The LHC experiments Joint Controls Project, JCOP,* ICALEPCS, Trieste, 1999

11-2 M. Dentan, *ATLAS policy on Radiation Tolerant Electronic*s, ATC-TE-QA-0001 (2000) https://edms.cern.ch/document/113816/

11-3 CAN in Automation (CiA), http://www.can-cia.de/

11-4 B. Hallgren et al., *The Embedded Local Monitor Board (ELMB) in the LHC Front-End I/O Control System*, 7th Workshop on Electronics for LHC Experiments, Stockholm, 2001

11-5 A. Daneels and W. Salter, *Selection and evaluation of commercial SCADA systems for the controls of the CERN LHC experiments,* ICALEPCS, Trieste, 1999

11-6 PVSS-II, http://www.pvss.com/

11-7 OLE for Process Control, http://www.opcfoundation.org/

11-8 H.J. Burckhart et al., *Vertical Slice of the ATLAS Detector Control System*, 7th Workshop on Electronics for LHC Experiments, Stockholm, 2001

11-9 F. Varela Rodriguez, *Systems of ELMB Buses using the Kvaser PCI CAN card,* ATLAS DCS-IWN17 (2002)

11-10 H. Boterenbrood et al., *Results of radiation tests of the ELMB at the CERN TCC2 area,* ATLAS DCS-IWN16 (2002)

11-11 F. Varela Rodriguez, *The Detector Control System of the ATLAS experiment: An application to the calibration of the modules of the Tile Hadron Calorimeter,* PhD. Thesis, CERN-THESIS-2002-035 (2002)

11-12 H.J. Burckhart et al., *Communication between Trigger/DAQ and DCS in ATLAS*, Computing in High Energy and Nuclear Physics Conference, Beijing, 2001

11-13 S.M. Schmeling et al., *The Detector Safety System for LHC Experiments*, XIII IEEE-NPSS Real Time Conference, Montreal, 2003

12 Experiment control

12.1 Introduction

The overall control of the ATLAS experiment includes the monitoring and control of the operational parameters of the detector and of the experiment infrastructure, as well as the supervision of all processes involved in the event readout. This functionality is provided by two independent, although complementary and interacting systems: control for trigger and DAQ (TDAQ control), and the detector control (implemented by DCS). The TDAQ control is in charge of controlling the hardware and software elements in TDAQ needed for data taking. The DCS handles the control of the detector equipment and related infrastructure. The architecture of the overall system has already been discussed in Section 5.3. The DCS is based on a SCADA system PVSS-II [12-1], whereas the TDAQ control is based on the TDAQ Online Software described in Chapter 10. These systems perform different tasks and have different requirements. Whilst the TDAQ control is required only when taking data, the DCS has to operate continuously to ensure the safe operation of the detector. The operation of the detector requires a high degree of coordination between these two systems and with the LHC machine. The interaction with the LHC machine will be handled by the DCS as illustrated in Figure 5-3 and presented in detail in Chapter 11. The TDAQ system is in overall control of data-taking operations.

The general control of the experiment requires a flexible partitioning concept as described in Section 3.5, which allows for the operation of the sub-detectors in stand-alone mode, as required for calibration or debugging, as well as for the integrated operation for concurrent data taking. The overall control strategy and the control operations of the various systems are described in this chapter. Furthermore, the required coordination of the various systems involved in the scenarios for physics data taking and calibration modes, is discussed.

12.2 Detector control

The DCS system provides the flexibility to implement the partitioning concept of ATLAS. The finest granularity of the TDAQ system is given by the segmentation of the sub-detectors into TTC zones [12-2] [12-3]. For this reason, the different sections of the sub-detectors will be logically represented in the back-end software of the DCS (Section 11.4) by means of the control units, which will be operated as a Finite State Machine (FSM). A logical control unit models the behaviour of a sub-detector or a system, e.g. the Pixel sub-detector or the high voltage system. Each control unit is characterized by its state. The control units are hierarchically organized in a tree-like structure to reproduce the organization of the experiment in sub-detectors, sub-systems, etc. as illustrated in Figure 12-1. The units may control a sub-tree consisting of other control or device units. The device units are responsible for the direct monitoring and control of the equipment, i.e. they represent hardware devices like a high-voltage crate or a temperature sensor. According to this model, the DCS of the Tilecal, for example, may be described by a tree of logical control units: the root logical unit controls the sub-detector itself and four independent child units, which control the four sub-detector sections (two extended barrels, EBA and EBC, and the central barrel sections, BA and BC). The sub-detector control units are in charge of supervising the various device units. Each control unit has the capability to exchange information or pass commands to other control units in the hierarchy. The flow of commands and informa-

tion will only be vertical. Commands will flow downwards, whereas status and alarms will be transferred upwards in the hierarchy.

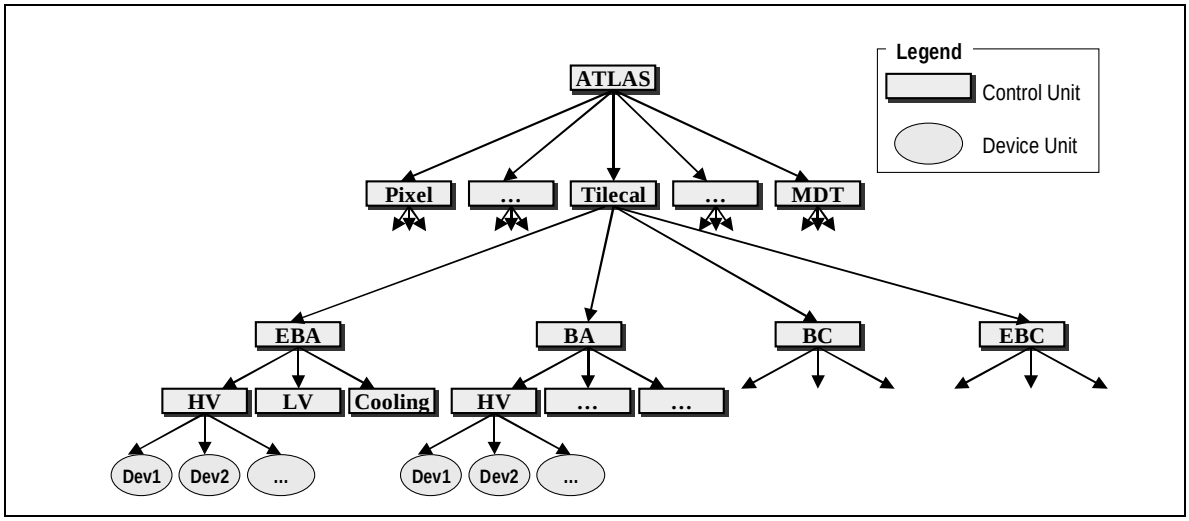

Figure 12-1 DCS logical architecture

The control units will support different partitioning modes. Any control unit and, therefore, the related sub-tree, may be excluded from the hierarchy and be operated in stand-alone mode for testing, calibrations, or debugging of part of the system. In this case the detector can be operated directly from the DCS control, whereas during physics data taking and calibration procedures, commands will be sent from the TDAQ control. Therefore a booking scheme that avoids the possibility of issuing conflicting commands must be provided. This mechanism will be developed according to the recommendations of the JCOP Architecture Working Group [12-4].

12.3 TDAQ control

The TDAQ system is composed of a large number of hardware and software components that have to operate in a coordinated fashion to provide for the data-taking functionality of the overall system. The organization of the ATLAS detector into detectors and sub-detectors leads to a hierarchical organization of the control system. The basis of the TDAQ control is provided by the Online Software, which is described in detail in Section 10.5.

The basic element for the control and supervision is a controller. The TDAQ control system is comprised of a large number of controllers distributed in a hierarchical tree following the functional composition of the TDAQ system.

This hierarchy is illustrated in Figure 12-2. Four principle levels of control are shown. Additional levels can be added at any point in the hierarchy if needed. A top level controller named the *root controller* has the overall control of the TDAQ system. It supervises the next level of controllers in the hierarchy, the *sub-detector controllers*. It is the responsibility of the sub-detector controller to supervise the hardware and software components which belong to this sub-detector.

The next control level takes the responsibility for the supervision of the sections that correspond to the TTC partitions [12-5]. The leaf controllers on the lowest level, the so-called *local controllers*, are responsible for the control of physical hardware such as ROSs. All elements of the TDAQ

system (e.g. ROSs, HLT farms etc.) employ the same type of controllers and follow a similar structure, as discussed in Section 12.3.1 and Section 12.3.2.

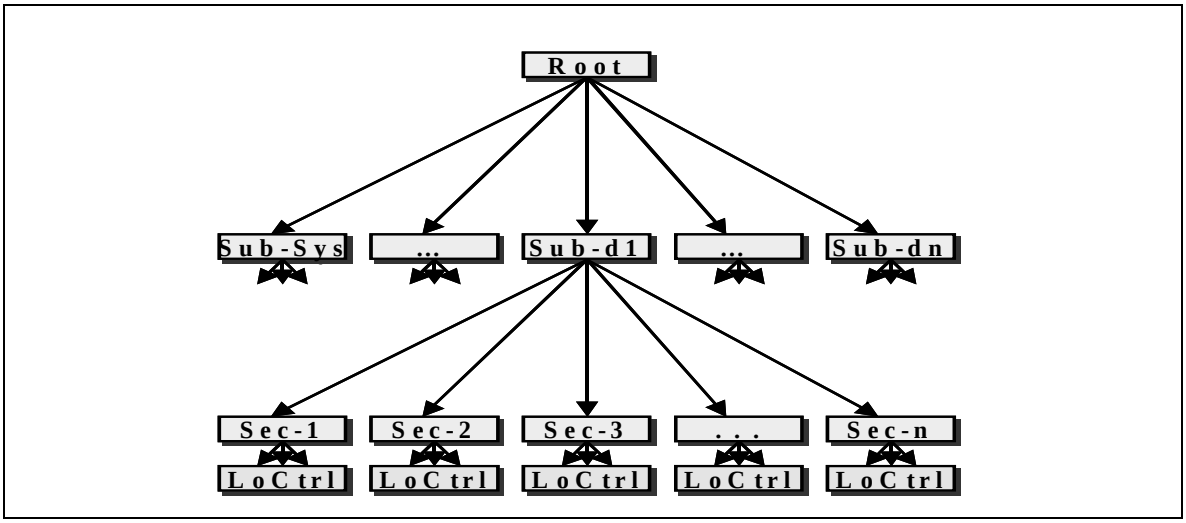

Figure 12-2 Online Software control hierarchy

A controller in the TDAQ system is characterized by its state (e.g. *Running*, *Paused*, *Stopped* etc.) given by the TDAQ state model described in Section 12.4.3. In any place of the hierarchy, a change of state may be initiated and synchronized from a given level of controller and sent down to the next lowest level. Once the requested transition has been performed, status information is returned to the requesting controller. Possible error conditions are also reported back.

A controller framework provides the facilities to handle the described operations in a coherent way on all the controllers in the system. It also gives the necessary flexibility to the detector expert to customize each controller to handle the individual tasks on the system under its control. These tasks may vary widely from initializing hardware readout to EF farm control. The information on the inter-relationship of the controllers and their responsibilities for a given partition is detailed in the configuration database (Section 10.4.3).

Each controller is responsible for the initialization and the shutdown of software and hardware components in its domain. It is also responsible for passing commands to child controllers (controllers under its control) and for signalling its overall state to its parent. Of particular importance is the synchronization necessary to start data-taking operations. This is performed by consecutive coordinated transitions through a number of intermediate states until data taking is finally started as described in Section 12.5.1. The shift operator drives the operations by issuing commands to the root controller. The inverse series of transitions is traversed when data taking is stopped.

During the operational phases, each controller is responsible for the supervision of the operation of elements under its direct control and for monitoring and handling any error conditions. In the case of a malfunction of a detector, for example, the controller can start corrective actions and/or signal the malfunction by issuing error messages. Severe malfunctions that are beyond the capabilities of a controller can be signalled by a state change to its parent (e.g. moving from a *Running* state to a *Paused* state). It is then the role of the parent controller to take further actions depending on the circumstances of the malfunction.

The design of the control, supervision, and error handling functions is based on the use of a common expert system. Specific controllers will use dedicated rules (information which tells the controllers what to do in one of many defined situations) to perform their individual functions and common rules to handle more general situations.

12.3.1 Control of the DataFlow

The DataFlow control handles all applications and hardware modules responsible for moving the event data from the detector front-end electronics and LVL1 trigger to the HLT system. It includes the control of the RoIB, the ROS, the Event Building and the SFOs. It also includes the overall control of the detector ROD crates.

There are two types of local DataFlow controllers foreseen, both making use of the Online software infrastructure in the same way, as shown in Figure 12-2. The ROS controller is tailored for the control of ROS software applications and hardware devices which do not themselves access the Online Software facilities. The Data Collection (DC) controller handles all other DataFlow applications and is optimized for the control of collection of processors. A version of the latter is also used for the control and supervision of the HLT and is described in Section 12.3.2. The other type, the ROD crate controller, takes over the communication handling between the other Online software services, Information Sharing and Configuration Databases, and the hardware that it controls.

Both controllers can be deployed at different levels of the control hierarchy. As an example, a DC controller can be used as top controller for all event-building applications, as well as a controller for a group of them. The DataFlow controllers request information on the elements they supervise from the configuration database. Their function is to initialize, configure, start, pause and stop the hardware and software data-taking elements, to monitor the correct functioning of the system, to gather operational statistics information, and to perform local error handling for those kinds of errors which cannot be handled locally, but do not need to be propagated further to higher control levels.

12.3.2 HLT farm supervision

The emphasis for HLT farm supervision is the synchronization and control of the HLT farms (LVL2 and EF) with that of the other TDAQ systems. An HLT farm is comprised of a set of sub-farms, each under supervision of a specific controller, as shown in Figure 12-3. These controllers have well-defined tasks in the control of the underlying processes running on the sub-farms.

A key overall design principle of the HLT has been to make the boundary between LVL2 and EF as flexible as possible in order to allow the system to be adapted easily to changes in the running environment (luminosity, background conditions, etc.). Therefore a joint control and supervision system for the two sub-systems has been designed and developed.

The configuration database contains data describing the software processes and hardware (processing nodes) of which the HLT is comprised as well as a wide variety of set-up and configuration data for pre-defined data-taking scenarios. The HLT supervision and control system uses these data to determine which processes need to be started on which hardware and subsequently monitored and controlled. The smallest set of HLT elements that can be configured and controlled independently is the sub-farm. This allows sub-farms to be dynamically included /

excluded from partitions during data-taking without stopping the run. Dedicated local run controllers supervise and control each HLT sub-farm, and report to an overall farm controller. This in turn condenses the information coming from each sub-farm and reports the global state of the farm (LVL2 farm or EF farm) to the run control. The controller also provides process management and monitoring facilities within the sub-farm. An expert system will guide the actions of the sub-farm controllers (e.g. spontaneously restarting crashed processes) so that it can maintain the sub-farm in the most efficient data-taking condition.

Where possible, errors are handled internally within the HLT processes. Only when they cannot be handled internally are errors sent to the supervision and control system for further treatment.

The Online Software services are used by the supervision system for monitoring purposes. For example, IS will be used to store state and statistical information that can then be displayed to the operator or made available to other control elements. For example, the malfunction of more than a pre-defined number of HLT sub-farms may precipitate the stopping of data-taking.

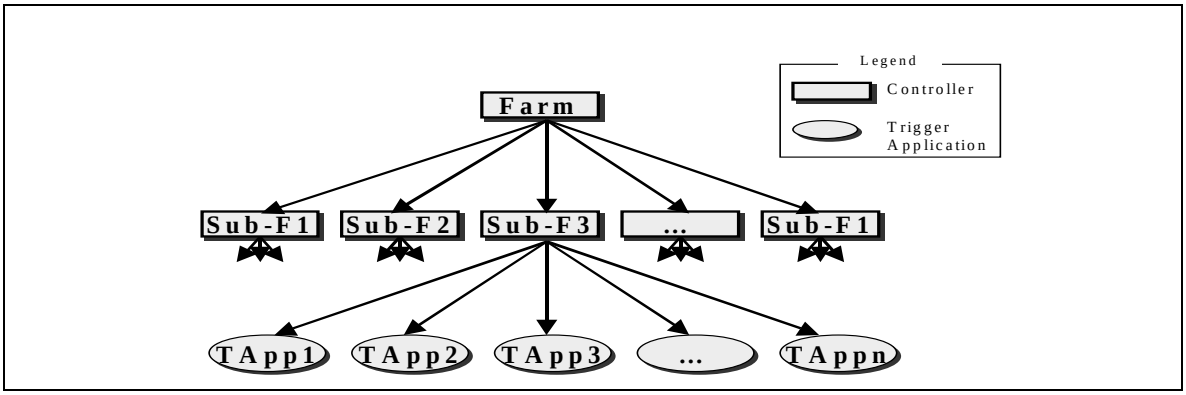

Figure 12-3 HLT farm control

12.4 Control coordination

The control of the experiment depends on the interplay between three systems: the LHC machine, the detector control, and the TDAQ control. For each of them, their instantaneous status is expressed in terms of distinct logical states (e.g. *Configured, Running, Stopped* etc.). The states of a given system are highly correlated to those of other systems and only certain combinations of states are allowed at any given time. Some examples are given here:

- the LVL1 trigger will not be started unless there are circulating beams in the LHC;
- the LVL2 trigger will not been started if some parts of its configuration process failed;
- physics data taking will not start if the detector high-voltage has failed.

The definition of these states, of their interplay, and of what actions are to be taken in what circumstances is a highly complex problem that will need the support of an advanced expert system.

12.4.1 Operation of the LHC machine

The phases of the LHC define a multitude of states [12-6] required for the internal functioning of the machine. A sub-set is of direct interest for the interaction with the experiment control, in particular those states and parameters that describe the condition of the beam with consequences for the operation of the detector. Phases with stable beam and low background may indicate that it is safe to bring the detector to the operational state for physics data-taking, whereas abnormally high background conditions may dictate that the detectors be turned off for safety reasons.

The nominal main phases to consider for the LHC are the following.

- *Filling* — the beam is being transferred from the SPS into the LHC;
- *Ramping* — the beam is being accelerated up to its nominal energy;
- *Squeezing* — the beam is being prepared for physics;
- *Collide* — physics with stable beam*;*
- *Dump, Ramp-down* and *Recover* — restart states.

It will also be necessary to know if no beam is expected for a given time since these periods will be used by the experiment to perform maintenance and test operations. The estimated duration of these periods is also of importance since the actions to be taken on the detector equipment will vary, e.g. HV reduction for short machine interventions or shutdown in case of major problems.

12.4.2 Detector states

As shown in Section 12.2, the operation of the different sub-detectors will be controlled by means of FSMs. The FSM approach allows for sequencing and automation of operations and it supports different types of operators and ownership, as well as the different partitioning modes of the detector. The FSM will handle the transition of the different parts of the detector through internal states and, together with the use of an expert system, will be vital in the coherent operation of ATLAS.

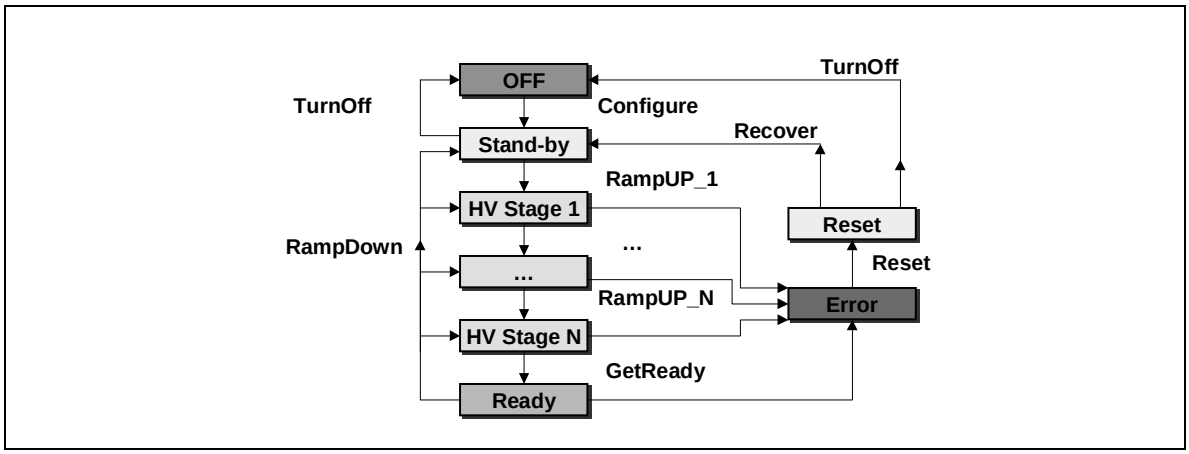

Figure 12-4 Detector states and transitions

Figure 12-4 shows an example of the internal HV states for a given sub-detector.

The sub-detector states are mainly determined by the status of the HV system. However, the status of the other systems like the cooling or low voltage systems, as well as of the external systems will also be taken into account in the state of the sub-detectors.

The starting situation for a sub-detector is the *Off* state in which the high-voltage system and, in some cases, the low-voltage system are switched off although other services like cooling may be permanently operational. On request from run control, this sub-detector may transit to the *Stand-by* state after the successful configuration of the front-end equipment. In this state the low-voltage system may be turned on and reduced HV values are applied to the sub-detectors. The transition to the *Ready* state will be performed through various intermediate states, which are mainly determined by the operational characteristics of the HV system of the sub-detector. The number of intermediate states is different depending on the sub-detector and is defined in the configuration database. In the *Ready* state the sub-detector equipment is ready for data taking. Many of these state changes, though requested by the run control, will actually pass for execution to the DCS (e.g. the run control will ask a detector's HV to be turned on by DCS). The DCS may also turn off or bring the sub-detector hardware into the *Stand-by* state in a controlled manner after a run. If an error is detected during the transition to any of these states or during data taking, the sub-detector will go to the *Error* state, where dedicated pre-defined recovery procedures will be applied depending on the type of failure.

The global operation of the DCS will be performed by a single FSM whose states will be built up from the states of the different sub-detectors. Any command issued at this level, which triggers a state transition, will be propagated to the sub-detectors. Similarly, any incident that affects the normal operation of a sub-detector will be reported and it may trigger a transition to the *Error* state.

12.4.3 Operation of the TDAQ states

The three main TDAQ states of *Initial*, *Configured* and *Running* (see Figure 12-5) were introduced in Section 3.2. These states are now further sub-divided.

Starting from booted but idle machines, the software infrastructure is initialized with the go command to arrive at the *Initial* state. Two state transitions are traversed between *Initial* and *Running*. The loading of the software and configuration data brings the system to the *Loaded* state. The system configures the hardware and software involved and enters the *Configured* state. The TDAQ system is now ready to start data taking. In the subsequent *Running* state the TDAQ system is taking data from the detector. Data taking can be paused and continued causing the LVL1 busy to be set and removed, respectively. The inverse transitions bring the TDAQ system back to the *Initial* state and it can be ended with the terminate command to arrive at the original state which has no TDAQ software infrastructure.

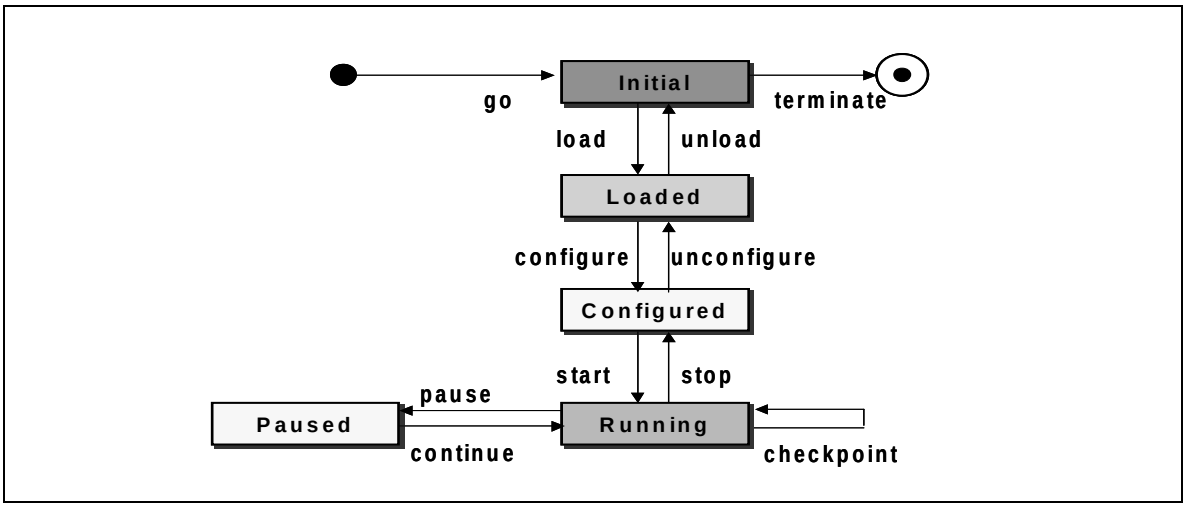

Figure 12-5 TDAQ states

The checkpoint is a transition in a running TDAQ system that is triggered by a change in conditions or by an operator. It results in events following the checkpoint to be tagged with a new run number and does not need the synchronization, via run control start/stop commands, of all TDAQ elements. Some components in the TDAQ control system require several intermediate states that are used for synchronization during certain transitions internal to the component.

12.4.4 Inter-relation of states

As discussed above, coherent synchronization between system states is vital in order to ensure the quality of the data and the safe operation of the detector. Communication with the LHC is handled by DCS as described in Chapter 11. DCS transfers both the LHC states and some of its operational parameters to the TDAQ control. On the other hand, parameters measured within the TDAQ system like luminosity, background and beam position, can be used to tune the beams and therefore, must be transferred to the LHC.

Figure 12-6 shows the overall connection for physics data taking between the TDAQ and DCS states and the LHC conditions. The actions performed by the DCS on the sub-detector hardware are coordinated with the states of the LHC machine. This is the case, for example, for the ramping up of the high voltages of some sub-detectors, like the Pixel or SCT tracker. These sub-detectors are more vulnerable to high beam background if the high voltage is on, and hence the command to get ready can only be given when the accelerator provides beam with sufficiently low background. The sub-detector states will closely follow the operation of the LHC. However, periods of particle injection or acceleration in the LHC may already be used by TDAQ to initialize and configure the different parts of the systems, such as the front-end electronics.

For physics data taking the LHC will be required to signal that stable beam conditions exist. Only when the TDAQ system is in *Configured* state, and the DCS is ready, may the operator give the command to start physics data taking. During physics data taking bi-directional communication continues to take place to assure the correct coordination and to enable the optimization of the beam.

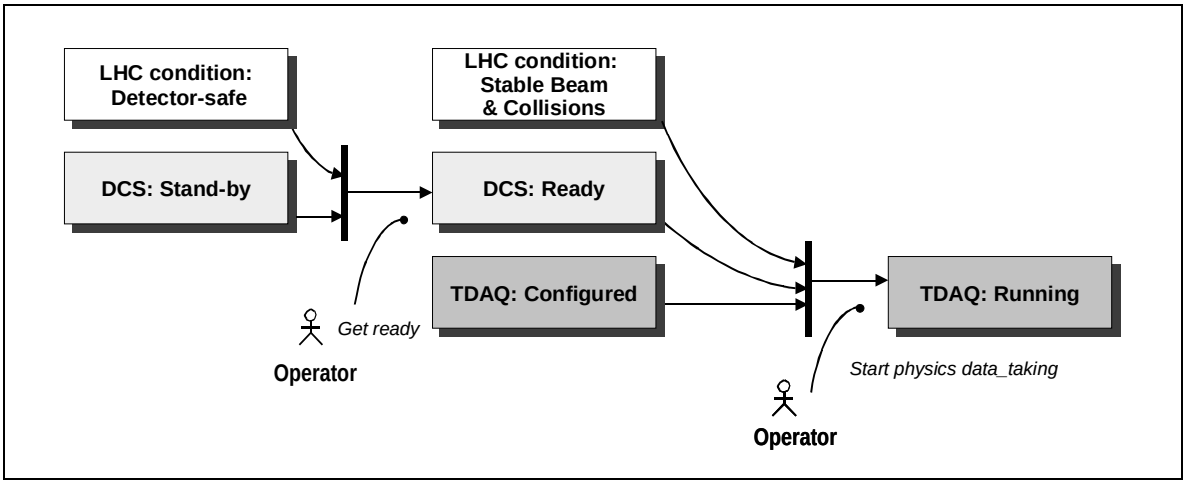

Figure 12-6 Basic example of connection between the TDAQ states, the detector states, and the LHC conditions. Note that the cross-check of LHC conditions is performed by the DCS (not shown here).

The TDAQ control is only partially coupled to the LHC and sub-detector states. State transitions of the TDAQ system are not determined by the state of the LHC. However, changes of the LHC parameters observed in the detector, like the luminosity, may require actions on the DAQ system, like the change of the trigger menus.

The TDAQ system can generally be brought from the initial to the configured state while the LHC is ramping, squeezing, and preparing for physics, or while the DCS prepares the detector for data taking. A new run can be triggered at any time regardless of the state of the LHC or of the detector. Data read out may already start under poor beam conditions using only certain sub-detectors, like the calorimeters, while the high voltage of other detectors will still be set to stand-by. As soon as the safe operation of the remaining sub-detectors is possible, the DCS will prepare them for data taking and will communicate their availability to the TDAQ system. At this moment, the physics data taking may start.

Although some calibration procedures, for example with cosmic rays or with a radioactive source, will be performed without beam, the communication and coordination with the LHC control system is still needed in order to prevent dangerous operations from being engaged that could possibly damage the detector. For most TDAQ internal system tests no co-ordination with the state of the other systems needs to take place.

12.5 Control scenarios

In this section, typical scenarios for the experiment control are presented. The first scenario describes the actions required to drive the system from its initial dormant state to the *Running* state and back again. The control of the various types of runs such as physics and calibration runs, introduced in Section 3.3, is then discussed. The procedures described rely on the ATLAS partitioning concept explained in Section 3.4 and Section 5.4.

12.5.1 Operational data-taking phases

The TDAQ states as described in Section 12.4 are traversed when the TDAQ system is run through initialization, preparation, data-taking, and shutdown phases. As the DCS is required to be always operational, in this scenario it is assumed that the DCS is in *Stand-by* state and ready to connect to the rest of the system. During the operational phases, systems and sub-systems perform a number of operations independent of each other. During the state transitions, actions specific to the sub-system like initializing software and hardware elements, loading software and parameters to hardware modules and configuring them, or starting processing tasks, are performed. Time-consuming operations are preferably performed during early state transitions. The aim is to prepare the TDAQ system to be ready for the start command such that the running state can be entered with a minimum time-overhead.

12.5.1.1 Initialization

When initiating a data-taking session, the operations of the TDAQ system start from booted but idle CPUs. The operator selects a partition that is described in the configuration database or defines a new one. The process management and information distribution infrastructure, consisting of a number of servers in the distributed system (i.e. process managers, Information Services, Error Message system), is started and initialized. The hardware and software elements of this TDAQ infrastructure are then verified to ensure that they are functioning correctly (for example the servers are tested, the process managers are contacted and therefore the status of their host PCs and the network connection is verified). The sequence and synchronization of the start-up operations follow the dependencies described in the configuration database. The Online Software–DCS communication software is started and the communication between the systems is initialized.

Once this TDAQ infrastructure is in place, the controllers and the application processes, which are part of the configuration, are started. The process management is de-centralized and therefore this can occur asynchronously. The data describing the selected partition are transmitted to DCS. The control communication with DCS is established. Having successfully finished this transition, the TDAQ system is in the *Initial* state.

12.5.1.2 Preparation

Once in the initial state, the operator can begin to configure the system. This involves synchronizing the loading and configuring of all software applications and hardware elements that participate in the data-taking process.

The *Loading* transition, is used to initialize all the processing elements in the system including, for example, the loading of the configuration data into the various controllers. Subsequently, during the *Configuring* transition, the various elements of the system are configured according to the information obtained in the previous step, for example, the establishment of communications channels between TDAQ elements such as the L2PUs and the L2SV, or the setting of hardware or software parameters.

The preparation of the sub-detector equipment for data taking involves the issuing of commands to DCS and the subsequent execution of defined control and initialization procedures. These commands can be associated to state transitions of the TDAQ controllers, or be asynchronous commands issued directly by the TDAQ operator or by stand-alone applications. The ac-

tions are detector-specific and are defined in the configuration database. They are loaded into DCS at initialization time. The different procedures to be performed on the equipment are previously validated and are cross-checked with the states of the external systems and of the common infrastructure to guarantee the integrity of the equipment. For example, the LHC must have stable beam conditions and acceptable backgrounds before certain sub-detectors' HV may be turned on. In some cases, the execution of these procedures may take several minutes, depending on the actions to be taken. These actions on the detector equipment take place in parallel to the loading and configuring of the TDAQ elements.

The operations described so far may be time-consuming and should, if possible, be performed well before the run start is required, e.g. while waiting for stable run conditions to occur. The readiness of the sub-detectors for data taking is reported to the run control by the DCS, which will then be in the *Ready* state. Generally speaking, the DCS must be in the *Ready* state for the operator to be able to start a run. However, it is possible to start a run regardless of the state of some or all of the sub-detectors if required (e.g. in commissioning phases when not all the detectors are available). In some cases the TDAQ control may decide to take data without certain sub-detectors or sections of a given subdetector. In this situation, the associated sub-detector run controllers are excluded from the DAQ partition used for data-taking and, consequently, no commands will be issued on the subdetector DCS, i.e. the sub-detector will remain in the default state (*Stand-by*) while the other subdetectors participate in the run.

The readiness of the TDAQ system to take data is defined as the moment when the top-level run controller is in the *Configured* state. This implies that all the data-acquisition and trigger systems have been booted, initialized, and configured correctly and are ready to receive event data.

12.5.1.3 Data taking

When the run is started by the operator, the LVL1 trigger is enabled and event data-taking operations commence. If necessary, a run can be paused and resumed in an orderly manner with a minimum time overhead simply by inhibiting and enabling the LVL1 trigger. In specific circumstances, such as the modification of one of a limited set of pre-defined run parameters (e.g. the beam position), the checkpoint transition, as described in Section 3.3.7, can precipitate a change in run number without having to cycle through the full run-stop and run-start procedures. The advantage of this is to minimize the time required to take account of the possible change in value of one or more of a fixed set of parameters.

Partitions comprising one or more TTC zones can be removed from the main data-taking partition, for example in case of problems with a given detector element. To do this, first the LVL1 trigger is inhibited. The detector element in question, together with its associated TDAQ resources (e.g. the ROBs associated to its RODs), is then marked as no longer forming part of the main data-taking partition. Finally the LVL1 trigger is re-enabled and data-taking can continue. The removed detector element can then be configured for stand-alone operations to allow the faulty element to be identified, tested, and repaired. Once the element is functional again, it can be re-attached to the main partition.

Depending on the system elements involved, the above actions may be able to be completed within a checkpoint transition, or they may necessitate the complete cycling through the run-stop and run-start procedures in order to re-configure the system correctly. Component failures during state transitions or while the system is in a given state that cannot be handled locally, are reported through the local controllers to the top level run control. This mechanism is introduced in Chapter 6 and described in Section 10.5.3.

12.5.1.4 Stopping

When the operator stops the run, the LVL1 trigger is inhibited and data taking is stopped. The control and application processes involved remain active. No changes on the DCS side are foreseen, the sub-detectors remain in the *Ready* state and TDAQ in *Configured* state. Systems and sub-systems may perform different kinds of actions during the *stopping* transition as compared to the *pause* transition.

12.5.1.5 Shutdown

On receipt of the shutdown command all applications and their controllers are stopped. Finally the TDAQ software infrastructure is closed down in an orderly manner, leaving the system available for a new data-taking session. If no further data taking is foreseen the Ramp Down or Turn Off commands are given to DCS in order to bring the detector to a safe state by ramping down and switching off the low and high voltage applied to the sub-detectors.

12.5.2 Control of a physics run

For physics data taking, the hierarchy of TDAQ controllers, including all sub-detectors, is arranged in a single common partition. The information on the type of run is transferred to the DCS system to prepare the different sub-detectors for physics data taking as described in the previous section. The successful execution of the appropriate DCS procedures to bring the sub-detectors to the *Ready* state, is then reported to the TDAQ system.

Figure 12-7 shows an example of the experiment control including the TDAQ control and the back-end system of the DCS. The TDAQ root controller holds the highest level of control. It connects to the detector controllers, the farm controllers for EF and LVL2, and the DF controllers as described in Section 12.3. Each sub-detector controller supervises the controllers of its sections and also the controller that provides the connection to DCS for each sub-detector. The RODs are supervised by their respective sub-detector section controller. In the following, it is assumed that the DCS is in the *Ready* state and the TDAQ control is in the *Running* state.

The control of a physics run is driven by the TDAQ system, which acts as the master of the experiment control by issuing commands to the DCS by means of specialized controllers called DDC_CT. There is one DDC_CT controller per sub-detector. Those controllers send commands directly to the sub-detector control units on the DCS side. This communication model implies that the TDAQ system interacts directly with the DCS of the various sub-detectors. However, in the case of DDC_CT connection failures the control of the sub-detectors is taken over by the DCS root control unit (called ATLAS in Figure 12-7) to ensure the safe operation of the detector equipment.

During physics data taking, only a pre-defined set of high-level commands from the TDAQ system to the DCS, like the triggering of the sub-detector state transitions at the start or end of the run, is allowed. The command is logged and feedback on its execution is reported to the TDAQ system. The TDAQ control system handles failures or time-outs from the DDC_CT in the same way as from other controllers in the system.

Global error handling and recovery is provided by the Online system control described in Section 10.5. Severe problems, e.g. in the HV system of a certain sub-detector, are reported to the TDAQ system. Depending on the problem and on the faulty system element, the TDAQ control

may decide to exclude this sub-detector from data taking and continue the run with the remaining detectors as described in the previous section. The run may continue if the readout of the detector element in question is not considered vital for the given run.

HLT sub-farms can be removed from or added to the global farm control without disturbance of data-taking activity. Breakdown and replacements of individual sub-farm nodes are handled transparently and each such operation is logged.

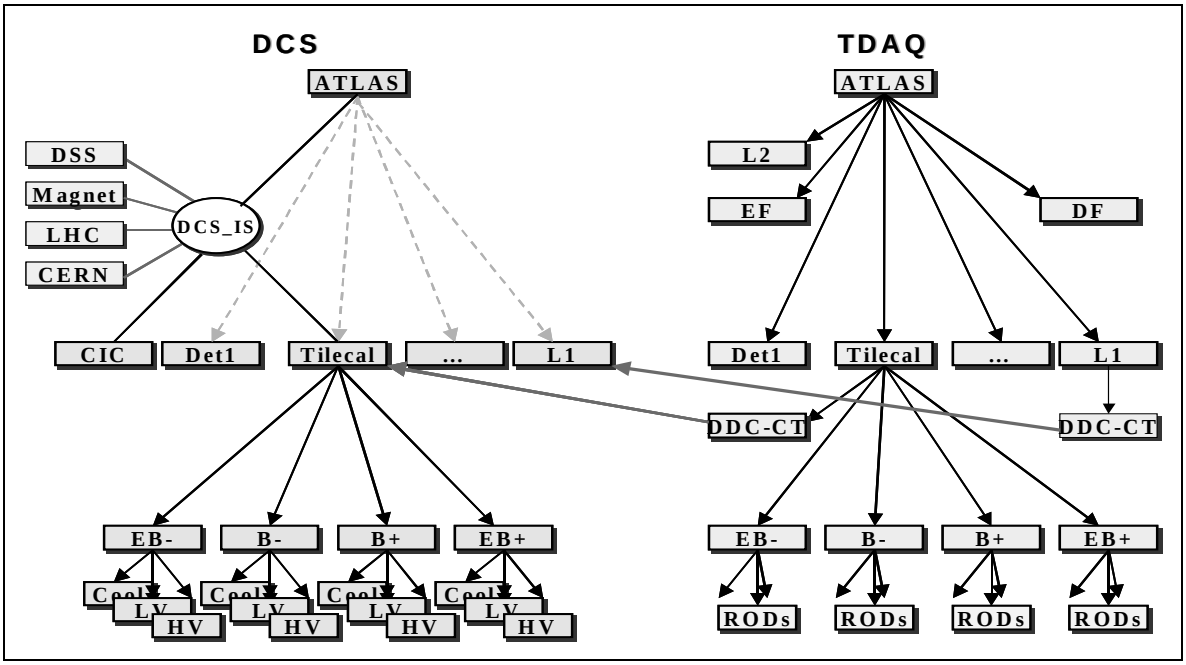

Figure 12-7 Experiment control indicating the control command flow between TDAQ and DCS that takes place at each sub-detector level, as illustrated here for the TileCal sub-detector

12.5.3 Calibration run

The calibration procedures envisaged in ATLAS take advantage of the flexible partitioning concept allowing for autonomous and parallel operations of the sub-detectors or even of different elements of a given sub-detector. From the point of view of controls, three different types of calibration can be identified:

- Procedures where only the control provided by the Online software system is required, such as the determination of the pedestals for zero suppression.
- Calibration runs entirely handled by the DCS such as the calibration of the cooling system, where the flow of the liquid is adjusted as a function of the temperature of the detector.
- Calibration procedures requiring control from both systems. This is the case, for instance, of the calibration of the Tile Hadron Calorimeter with the radioactive caesium source, where the modules of the detector are scanned with the source under control of the DCS. The signal produced is read by the DAQ system and the information is used to adjust the HV applied to the PMTs of the readout system.

In the latter case, the control needs are similar to the functionality required for physics runs as presented in the previous section. Figure 12-8 shows the example of the interplay between the

TDAQ control and the DCS for calibration of the Tilecal detector. As for physics data taking, these calibration procedures are driven by the TDAQ control and commands follow the same path. The main difference with respect to physics data taking, is the arrangement of the partitions. In the example presented in the figure, a TDAQ partition is defined for the stand-alone operation of the Tilecal sub-detector. It is important to note that although some calibration procedures are executed without beam or even without the control provided by the DCS, the communication with the LHC machine and other external systems must always be guaranteed since this information is of crucial importance for the integrity of the sub-detectors and for the preparation of the next run.

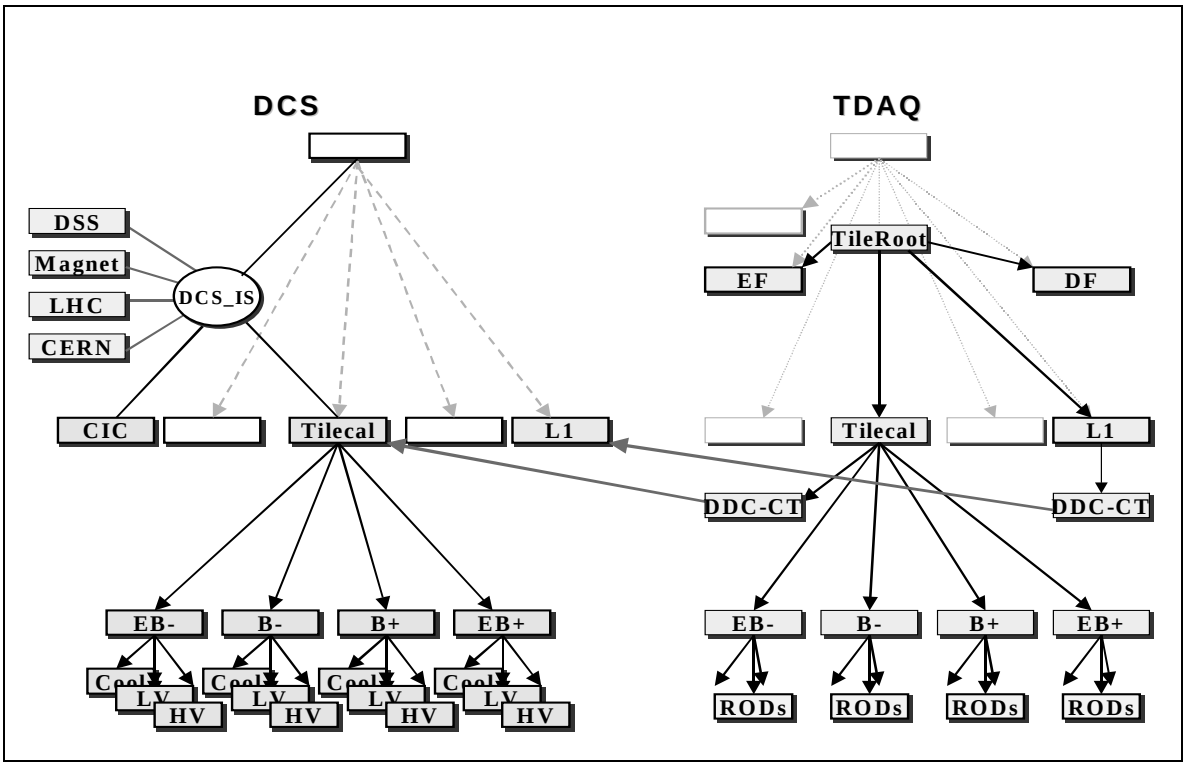

Figure 12-8 Detector Stand-alone control mode, indicating the control command flow between TDAQ and DCS at the sub-detector level, as illustrated here for the TileCal sub-detector

12.5.4 Operation outside a run

The states of the DCS and TDAQ outside a run are determined by the duration of the periods without data taking. As described in Chapter 3, during transitions between runs and short interruptions, the TDAQ system can be set to one of its intermediate states or be unavailable, while the DCS remains completely operational. If longer interruptions are foreseen, such as periods of machine development, the HV applied to the sub-detectors is reduced and the detector states are set to *Stand-by* or *Off*.

During shut down periods and long intervals without data-taking, the TDAQ system is not necessarily operational although it may be available on demand for calibration and debugging purposes. In these periods a number of sub-detector services like the LAr cryogenics or ID cooling stay operational. The monitoring and control of the humidity and temperature of the electronics racks, the supervision of the un-interruptible power supply system and of other detector specific equipment are also performed. The ATLAS magnets are permanently switched on and there-

fore the interface with the DCS must be continuously available. The radiation levels monitored by the LHC machine must be accessible by the DCS at all times. For these reasons, the DCS is required to be fully operational allowing, if necessary, the sub-detectors to operate without the support of the rest of the TDAQ system. Similarly, the interfaces between the DCS and the fire alarm systems, the access security system and the DSS must be continuously operational.

12.6 References

12-1 PVSS-II,
http://www.pvss.com

12-2 *ATLAS High-Level Triggers, DAQ and DCS Technical Proposal*, CERN/LHCC/2000-17 (2000)

12-3 *ATLAS Level-1 Trigger Technical Design Report*, CERN/LHCC/98-14 (1998)

12-4 JCOP Architecture Working Group, *Framework Design Proposal*, CERN-JCOP-2000-06 (2000),
http://cern.ch/itco/Projects-Services/JCOP/SubProjects/Architecture/pdf/HC7.0.pdf

12-5 *Partitioning - Global issues working group document*,
http://cern.ch/atlas-project-tdaq-giwg/Documents/Documents.htm

12-6 LHC Operations project,
http://lhcop.web.cern.ch/lhcop/

Part 3

System Performance

13 Physics selection and HLT performance

13.1 Introduction

A preliminary view of the on-line event-selection scheme and the corresponding physics coverage was presented in the HLT, DAQ and DCS Technical Proposal (TP) [13-1]. Since then the studies have evolved to cope with a new scenario for the start-up of the LHC machine and in response to funding constraints. The LHC start-up scenario has a target to deliver 10 fb^{-1} in one year, now assuming a peak luminosity per fill of 2×10^{33} cm^{-2} s^{-1}, which is a factor of two higher than assumed in the TP. At the same time as having to address this higher initial luminosity, financial resources from the HLT/DAQ project had to be re-assigned to meet urgent needs elsewhere in the experiment. As a consequence, the construction of the HLT/DAQ system will have to be staged to match the available funding, so that only a reduced system will be available for the initial running. These changes required a major revision of the Physics and Event Selection Architecture of the experiment, including new ideas for reducing event rates and sizes while retaining as many as possible of the ATLAS physics goals. Needless to say, only the availability of real data will allow this proposal to find a final implementation and a fine tuning of the relative weights of the selection signatures. However, it is important to be able to face this phase with the most complete set of tools and a versatile selection architecture, in order to cope with the obvious unknowns that will likely show up at the time of LHC start-up.

As it has been described in Chapter 9, the High Level Trigger (HLT) system of the experiment is composed of two separate event-selection steps, LVL2 and the Event Filter (EF), each with distinctive and complementary features. A common denominator is that they will operate using software algorithms running on commercial computers to test hypotheses of particle identification and apply event-selection criteria. LVL2 will do this with special-purpose algorithms that need to operate in about 10 ms and use only part of the detector information at full granularity, while the EF will have the fully-built events at its disposal and work with an execution time of the order of a second. It is important to maintain a flexible scheme able to adapt easily to changes in machine conditions (e.g. luminosity or background rates). The modularity of the HLT will allow the implementation of different reduction steps at different stages.

A essential input to the HLT process is the seeding of the selection with the results from LVL1. When making performance studies for the HLT, a detailed simulation of the LVL1 result is therefore needed. This identifies the regions of the detector (Regions-of-Interest) where potential candidates for interesting physics objects are found. This simulation, described in Section 13.2, allows for a realistic use of the information coming from LVL1, using the same algorithms and procedures that will be implemented in custom hardware in the experiment.

Given the commonalities between LVL2 and the EF, it was recognized that a coherent and organized approach to the software components of the trigger was needed. The work presented in Section 13.3, which represents an important step forward with respect to the TP and is the result of a large effort, has concentrated on this issue. Common tools have been developed for the event selection, and common data-model components and methods have been identified that can be shared by different algorithms, in particular at LVL2. This approach will ease the implementation of different selection schemes and also simplify the migration of algorithms between trigger levels.

Another important point for new developments has been compliance with the updated detector geometry and with the format of the raw data as it will come from the ReadOut System. This implies that the algorithms must operate starting from streams of bytes organized according to the readout structure of each detector, in exactly the same way as in the real experiment. This has allowed one to study and understand the implications of converting these byte-streams to the objects needed by algorithms in order to perform trigger selections, including making preliminary measurements of the computing time needed for these conversions.

In Section 13.4 the outcome of the present studies is presented. It should be noted that the detailed results on trigger rates and efficiencies presented in the TP and in related papers and presentations are still considered a valid baseline approach. Present work has concentrated on implementing a realistic approach to data handling and preparation for the algorithms and on assembling a proper set of software selection tools. Emphasis has been put on the selection of electrons and photons, both of which stem from an electromagnetic cluster identified at LVL1, and muons. For each of the electron/photon and muon event-selection "vertical slices", a thorough implementation of the approach described above has been attempted. For events that passed the LVL1 selection, bytestream raw data organized according to the actual detector readout format are used by LVL2 algorithms operating within the PESA Steering and Control framework (see Section 9.5). Simulated RoI information from LVL1 is used to initiate the LVL2 processing. Trigger elements are built using detector information and used to test particle identification hypotheses. For events retained by LVL2, the EF reconstruction and analysis are performed (the EF may or may not be seeded using the LVL2 result) and the final-selection result is made available for off-line use. Results on rejection against the dominant backgrounds and on the efficiencies for typical signals are reported, as well as the rates deriving from each of the selections.

In order to span fully the ATLAS physics coverage, signatures involving jets, hadronic decays of tau leptons, large missing transverse energy and also jets with b-quark content are needed, in addition to the electron, photon and muon ones discussed above; these are also discussed in Section 13.4. As described in Chapter 4, spare on-line resources will be used for B-physics studies, e.g. for luminosities below the peak one. Results are based on preliminary analyses of samples of the order of 10^6 events and they will be extended in the near future to the larger samples of several 10^7 events produced thanks to the effort of the ATLAS Data Challenge Group [13-2].

The global assessment of the event rates to be recorded for off-line analysis, based on the present knowledge for each signature, is made in Section 13.5. A sketch of issues related to the initial phase of the experiment, seen from the selection architecture point of view, is given in Section 13.6.

13.2 The LVL1 trigger simulation

An important ingredient to many HLT tests and studies is the simulation of the LVL1 trigger, the results of which serve as input to the HLT trigger processing. The ATLAS LVL1 trigger [13-3] is itself a complex system consisting of the calorimeter trigger, the muon trigger and the Central Trigger Processor (CTP) that makes the final LVL1 event decision. Figure 13-1 gives an overview of the LVL1 trigger; the various components mentioned in the figure will be explained later in this section except for the TTC system (trigger, timing and control) which has no equivalent in the simulation.

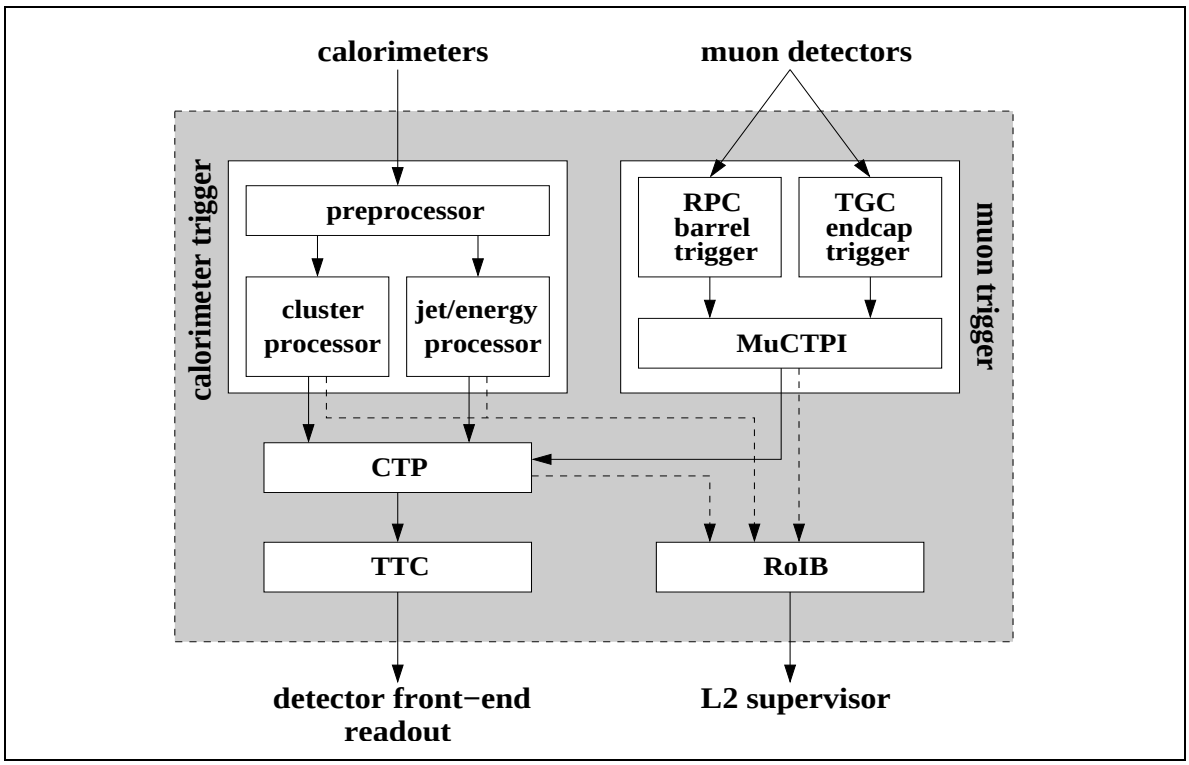

Figure 13-1 An overview of the ATLAS LVL1 trigger system. The Region-of-Interest Builder (RoIB) formally is not a part of the LVL1 trigger. However, it is simulated together with the LVL1 trigger.

The LVL1 trigger simulation is implemented in C++ in the ATLAS offline computing framework Athena. It relies heavily on the ATLAS offline data storage implementation, the so-called Transient Event Store (TES). The structure of the simulation follows closely the structure of the LVL1 trigger hardware. Figure 13-2 shows a package view of the LVL1 simulation. The simulation consists of packages simulating various hardware blocks: the resistive plate chamber (RPC) muon trigger (indicated by the package TrigT1RPC in Figure 13-2), the Muon-to-CTP Interface (MuCTPI, package TrigT1Muctpi), the calorimeter trigger (package TrigT1Calo) and the Central Trigger Processor (package TrigT1CTP). The LVL1 configuration (package TrigT1Config) and the simulation of the Region-of-Interest Builder (package TrigT1RoIB) are provided as additional packages. There are also packages for the definition of the LVL1 result raw data object (package TrigT1Result), for classes used by more than one package (package TrigT1Interfaces), and for the conversion of the LVL1 result into the hardware format, the so-called *bytestream* conversion (package TrigT1Result-Bytestream). The various parts of the simulation shown in Figure 13-2 will be explained in the next sections. The simulation of the muon trigger in the endcaps, the signals for which are provided by the thin-gap chambers (package TrigT1TGC), so far exists only as a stand-alone program and will not be discussed in detail in this document.

The interfaces and data formats to be used in the simulation [13-4] were designed to follow as closely as was practical the data formats used in the LVL1 trigger hardware which are documented in [13-5]. Additional information on the LVL1 simulation can be found in [13-6].

13.2.1 Configuration of the LVL1 trigger

The first task of the LVL1 trigger configuration package is to translate the trigger menu, i.e. the collection of event signatures LVL1 is supposed to trigger on, into something that the simula-

tion of the CTP can understand and use in making the event decision based on logical combinations of the inputs delivered by the calorimeter and muon triggers. The LVL1 signatures, or *trigger items*, are combinations of requirements (or *trigger conditions*) on the multiplicities of various kinds of candidate objects found by the calorimeter and muon triggers in the event. (See later subsections for details about the calorimeter and muon triggers and their simulation.)

A simple example of a trigger item is 'one (or more) electron / photon candidate(s) with transverse energy above 10 GeV and one (or more) muon candidate(s) with transverse momentum above 15 GeV'. In a frequently-used and obvious notation this reduces to '1EM10+1MU15', where the string 'EM' ('MU') represents the electron / photon (muon) candidate, and the numbers in front of and behind the string symbolize the required multiplicity and the required transverse momentum, respectively. The combination of a string and a threshold value (like 'EM10') is called a *trigger threshold*.

The second task of the LVL1 configuration package is to configure the simulation of the calorimeter and muon triggers to deliver the information required to make the event decision using the trigger menu, i.e. to deliver the multiplicities for the required trigger thresholds. For the example mentioned above, the calorimeter trigger has to be configured to deliver the multiplicity for the threshold 'EM10', i.e. the number of electron / photon candidate objects with transverse momentum above 10 GeV, to the CTP. It is obvious that the trigger menu and the trigger thresholds for the calorimeter and muon triggers have to be defined consistently. In particular, all thresholds used in the definition of any trigger condition in any trigger item must be delivered to the CTP by the calorimeter and muon triggers.

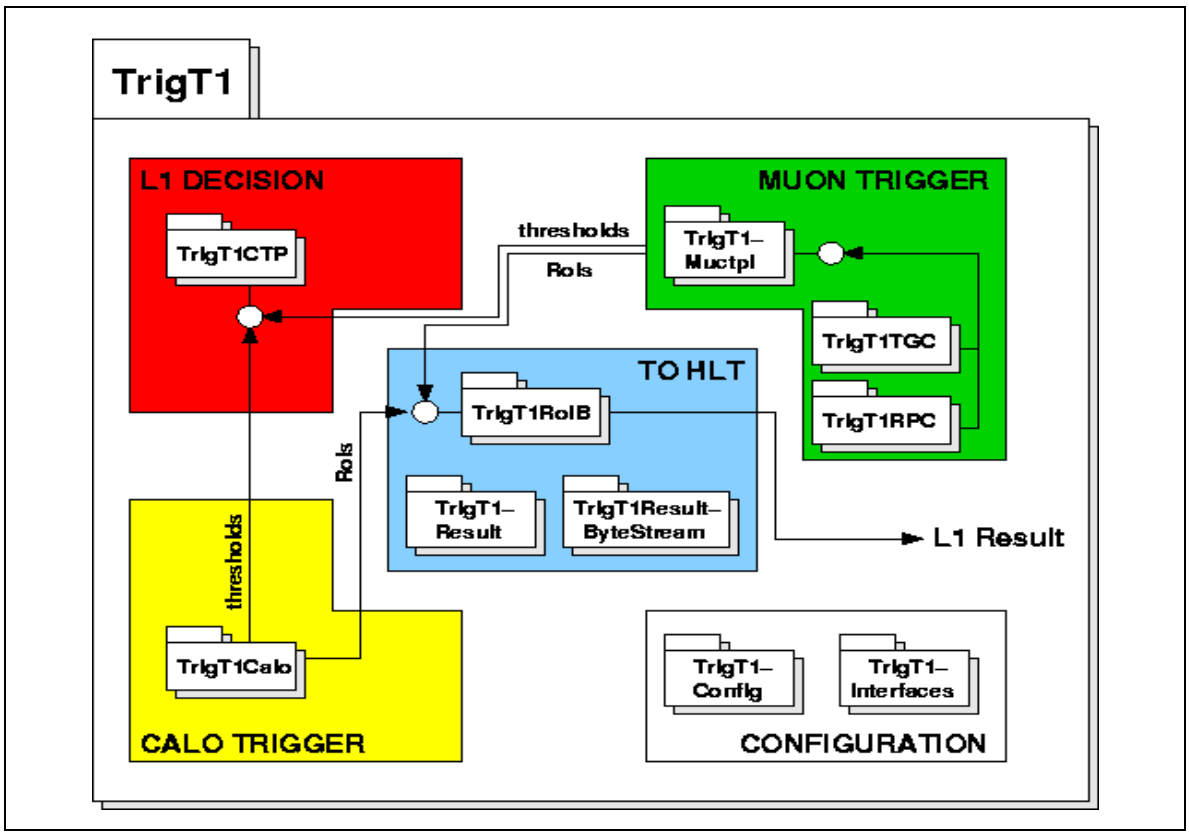

Figure 13-2 A package view of the LVL1 trigger simulation.

Both the trigger menu and the list of required trigger thresholds are defined using XML and are parsed into instances of C++ classes using the Xerces DOM API [13-7]. The parsing of the trigger menu creates an object which contains the information on how the CTP simulation has to discriminate the calorimeter and muon trigger inputs (trigger conditions) and what items have to be built from these conditions. In addition, configuration objects for the calorimeter and muon triggers are created in the configuration process and are stored in the TES for later retrieval by the calorimeter and muon trigger simulations. These objects contain the list of thresholds for which the subsystems have to provide multiplicity information to the CTP simulation.

The LVL1 trigger configuration software is currently being adapted to also be able to configure the LVL1 trigger hardware by deriving the necessary look-up table files and FPGA configuration files from the XML trigger menu and trigger threshold list. Such a common configuration scheme will allow for cross-checks between hardware and software.

13.2.2 The calorimeter trigger and its simulation

The LVL1 calorimeter trigger [13-8] searches for localized energy depositions in the calorimeters due to electrons and photons (electromagnetic clusters), single hadrons and hadronic decays of tau leptons (narrow hadronic clusters) or jets (broader hadronic clusters). For each type of cluster, a number of programmable transverse-energy (E_T) thresholds are provided. With the exception of the jet clusters, isolation requirements can be imposed — these are implemented by applying thresholds on isolation variables associated with the cluster. The multiplicities of the candidate objects of each type are counted for each threshold set, and the multiplicity values are passed on to the CTP to be used in making the LVL1 event decision.

In addition to the local-energy triggers discussed above, the calorimeter trigger calculates global energy sums (total transverse energy and missing transverse energy) which are discriminated against a number of programmable thresholds; the results of the threshold comparisons are also passed to the CTP to be used in making the LVL1 event decision.

The calorimeter trigger uses signals from ~7200 trigger towers which are analogue sums over calorimeter cells in the liquid-argon or scintillator-tile calorimeters. The trigger tower signals are digitized in the preprocessor electronics, which also performs digital signal processing to calculate transverse energy and make bunch-crossing identification. The resulting data are passed on to two processor subsystems. The Cluster Processor searches for electron/photon and tau/hadron candidates within 3200 trigger towers of granularity $\Delta\eta\times\Delta\phi = 0.1\times0.1$ in each of the electromagnetic and hadronic calorimeters in the pseudorapidity range $|\eta| < 2.5$. The Jet/Energy Processor searches for jets and makes the global energy sums using coarser (jet) elements of granularity $\Delta\eta\times\Delta\phi = 0.2\times0.2$ over a larger rapidity range ($|\eta| < 3.2$ in case of the jet trigger; $|\eta| < 4.9$ for forward jets and the energy-sum triggers). Note that it is the preprocessor that sums the trigger towers, independently for the electromagnetic and hadronic calorimeters, to form the larger elements used in the Jet/Energy Processor.

In the electron/photon trigger a candidate object is defined by a local E_T maximum in a region of 2×2 trigger towers (electromagnetic plus hadronic), corresponding to a 0.2×0.2 region in η–ϕ space. Vetos may be applied on the hadronic transverse energy in this region and/or on the transverse energies in the rings of towers surrounding the 2×2 region in the electromagnetic and hadronic calorimeters. The cluster thresholds are applied on the maximum transverse energy in any edgewise-adjacent pair of electromagnetic-calorimeter trigger towers within the central 2×2 region. (See Refs. [13-3] and [13-9] for a more detailed description of the various calorimeter trigger algorithms.)

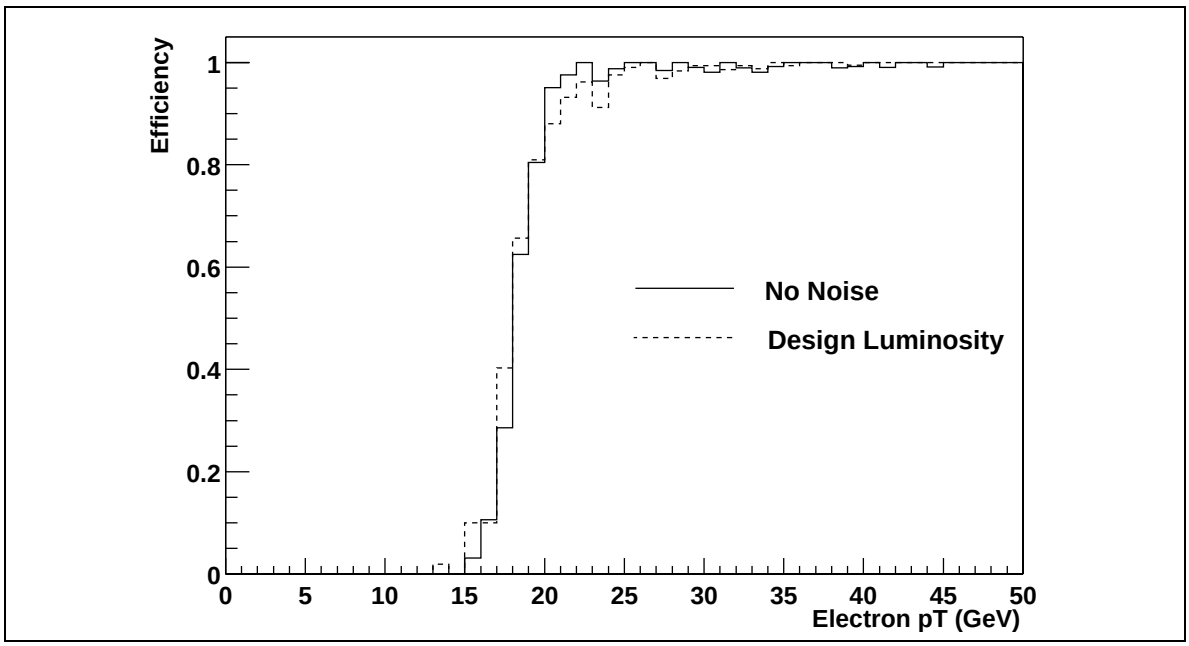

Figure 13-3 Single electron trigger efficiency as a function of the electron transverse energy E_T, using a 17 GeV threshold for different scenarios. See text for details.

In addition to counting multiplicities of candidate objects, Regions-of-Interest (RoIs) are identified and passed, via the Region-of-Interest Builder, to the LVL2 trigger for events that are retained by the LVL1 trigger. The RoIs, which contain information on the geographical location of the object in the calorimeters and on the E_T thresholds passed, are used to seed the HLT event-selection process. More detailed information from the LVL1 calorimeter trigger is sent to the DAQ using standard readout links.

The LVL1 calorimeter trigger simulation is designed to reproduce in detail the functionality of the hardware, but it does not entirely duplicate the dataflow. The primary reason for this is efficiency — the hardware trigger will process large amounts of data in parallel, which does not translate well to offline simulation software.

Currently the simulation of the LVL1 calorimeter trigger starts from calorimeter cell energies which are summed to form tower energies. (A more complete simulation which is in preparation will include details of the signal-processing chain from the calorimeters to the output of the preprocessor, but this is not yet available.) The cell energies, which can be taken from the ATLAS fast simulation or from the detailed GEANT simulation of the calorimeters, are used to build, in a simplified geometrical approach, trigger tower signals to which calibration and a gaussian noise can be applied. The tower data are passed to the Cluster Processor simulation (electron / photon and tau / hadron finding), and are summed into the coarser jet elements used in the simulation of the Jet / Energy Processor (jet finding and calculation of energy sums).The results from the simulation of the calorimeter trigger are stored in the TES for later retrieval. These include the multiplicity outputs that are used by the CTP in making the event decision, and the details of candidate objects that are used to guide the LVL2 trigger processing (the simulation of the output from the calorimeter trigger simulation to the DAQ has not yet been implemented.). Note that the RoI data from the simulation of the calorimeter trigger are stored in the same bytestream format as will be used in the hardware.

The HLT steering software requires the RoIs to be given with coordinates in η–ϕ space and a value (in GeV) of the E_T threshold that was passed, not in terms of the LVL1 internal data for-

mat (which reflects the details of the electronic implementation of the trigger). Software converters are provided to translate the raw 32-bit RoI data words [13-5] into objects, complete with methods to return the required data.

An effort has been made to validate the performance of the calorimeter trigger simulation [13-10]. Here we present some examples of efficiency and rate results for the electron / photon trigger.

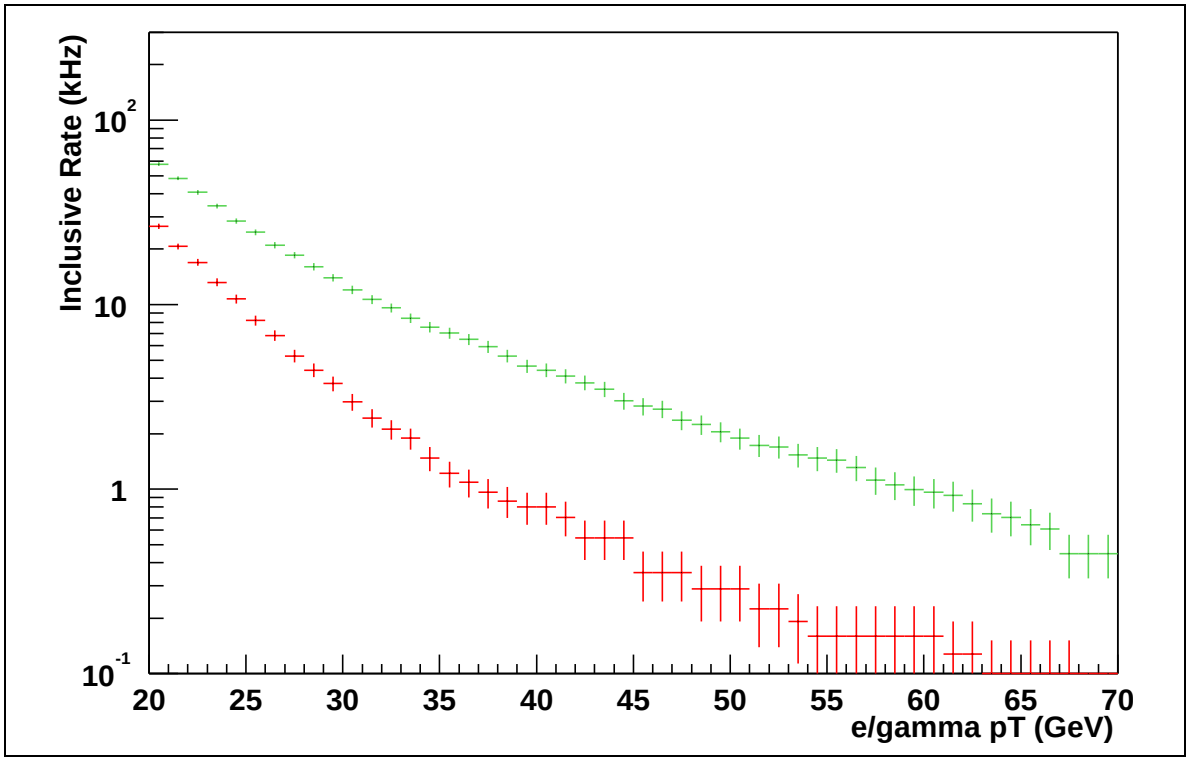

Figure 13-4 Electron/photon trigger rate versus E_T threshold without (top) and with (bottom) isolation requirements at low luminosity 2×10^{33} cm^{-2} s^{-1}. See the text for details.

Figure 13-3 shows the electron trigger efficiency as a function of p_T when requiring that the trigger cluster E_T be greater than 17 GeV. The electrons used in this study came from simulated $Z \rightarrow e^+e^-$ decays, and were required to be well isolated from other high-p_T particles (to ensure that the electron shower was responsible for the trigger) and restricted to a fiducial range $|\eta| < 2.45$. Results are shown without electronic noise or pileup, and also at design luminosity (1×10^{34} cm^{-2} s^{-1}) with electronic noise added. A sharp rise in the efficiency is observed around the threshold value in both cases, with little degradation due to noise and pileup

The rate of the LVL1 electron / photon triggers is dominated by misidentified jets. Figure 13-4 shows the estimated trigger rate for the single electron / photon trigger as a function of transverse-momentum threshold for a luminosity of 2×10^{33} cm^{-2} s^{-1}. The upper and lower bands give the rate before and after applying the isolation requirements. The E_T-threshold scale in the plot is defined so that the efficiency for selecting genuine electrons with transverse-energy equal to the quoted value is 95%. Figure 13-5 presents the rate for the electron / photon-pair trigger for the high-luminosity scenario of 1×10^{34} cm^{-2} s^{-1}, with and without isolation.

We have compared the results obtained using the new object-orientated software against those presented in the LVL1 TDR [13-2] which were obtained using the previous FORTRAN-based

software. The results are not expected to be identical since the studies used different version of Pythia for the event generation, there have been significant changes to the detector model, and some changes have been made to the RoI-selection and isolation algorithms. Note also that new simulation does not yet include the details of the signal-processing chain for the trigger-tower summation, digitization and bunch-crossing identification. In general there is fair agreement between the old and new results. For example the estimated rates for the single electron / photon trigger for a luminosity of 2×10^{33} cm^{-2} s^{-1} agree with the earlier simulations to within 20%. In view of the potential sensitivity of the isolation cuts to details of the trigger-tower simulation and to be conservative, we have based our estimates of event and data-rate requirements for the HLT / DAQ system on an extrapolation of the LVL1 TDR results.

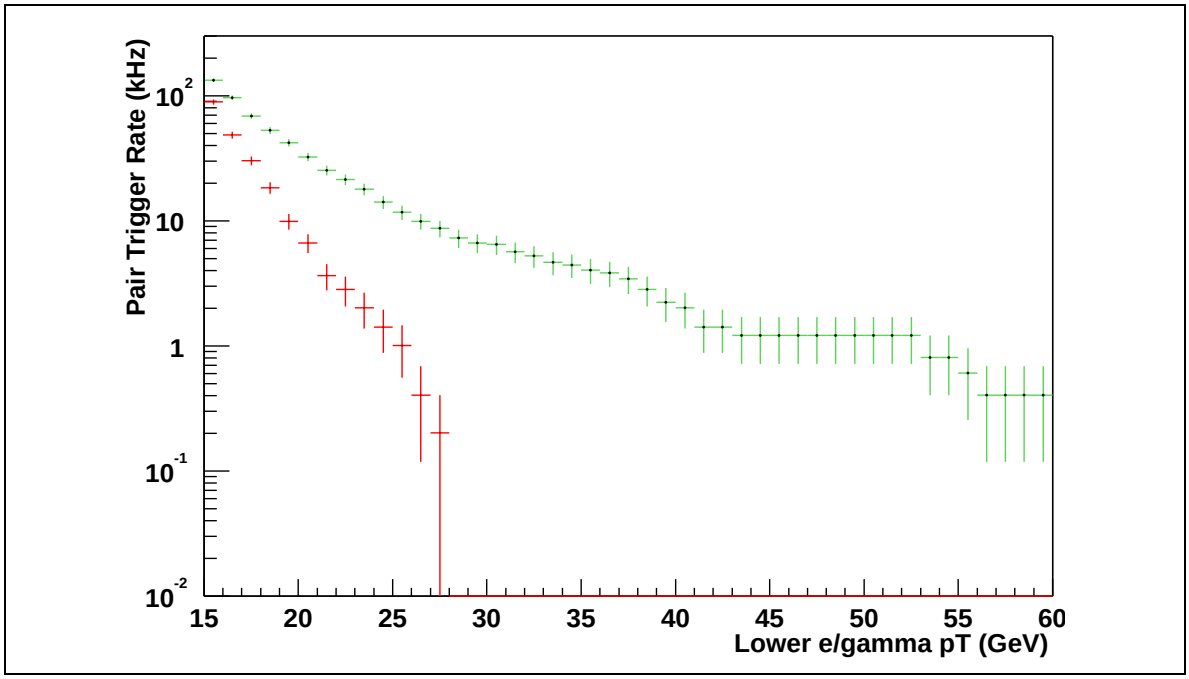

Figure 13-5 Predicted trigger rates for electron pairs without (top) and with (bottom) isolation at high luminosity 1×10^{34} cm^{-2} s^{-1}. See the text for details.

13.2.3 The RPC muon trigger and its simulation

The barrel muon trigger ($|\eta| < 1.05$) uses information from six layers of Resisitive Plate Chamber (RPC) detectors that are organised in three stations [13-11]. The middle RPC station (RPC2) is called the *pivot plane*. The principle of the algorithm that finds muon candidates is as follows [13-12]: each hit found in the pivot plane is extrapolated to the innermost RPC station (RPC1) along a straight line through the interaction point, and a *coincidence window* is defined around the point of intersection. Since muons are deflected in the magnetic field, the size of the coincidence window defines the p_T threshold of the trigger. A low-p_T muon candidate is found if there is at least one hit in the coincidence window and if in at least one of the stations RPC1 and RPC2 there are hits in both planes of the doublet. This condition must be satisfied for the measurements in the bending plane and also for those in the non-bending plane. If, in addition, there is a hit within a coincidence window in at least one of the two planes of the outermost station RPC3, a high-p_T candidate is found. Again the coincidence condition must be satisfied for the measurements in the bending plane and also for those in the non-bending plane.

In the hardware implementation of the trigger, the algorithm is implemented using logic that is programmed to recognize valid patterns of hits in the RPCs. All pivot-plane hits are processed in parallel, allowing a trigger with a very short fixed latency. There are three independently programmable thresholds for each of the low-p_T and high-p_T triggers. For each of the 64 sectors of the RPC trigger, up to two muon candidates can be selected and sent to the MuCTPI. If there are more than two candidates in a sector, the two highest-p_T candidates are retained and a flag indicates that additional candidates were suppressed.

The input to the simulation of the muon trigger logic was provided by a package that simulates the RPC detector system; this was done with the ATLSIM program. The muon detector layout used for this simulation was version "P03" [13-12]. The geometry of the RPC detectors and the stations built from them, as well as the positioning within the muon spectrometer, were reproduced in great detail according to the latest engineering drawings. The material composition and geometry of the single-RPC units were also correctly simulated. The simulation of the RPC detector response was based on results obtained in test-beam experiments. The hits produced in the simulation of charged particles crossing the RPC detectors were collected and stored in output files for use in downstream packages for the simulation of the trigger logic and also for the event reconstruction.

The detector simulation stage is followed by a number of packages which, using Athena algorithms and services, simulate in detail the hardware core and the overall architecture of the LVL1 muon barrel trigger. The hardware simulation is built from a set of classes which reproduce, up to the level of the internal data registers, the behaviour of the basic hardware components: the Coincidence Matrix ASIC (CMA), the Pad Board, the Sector Logic and the Read-Out Driver. The detector data are first accessed by the CMA, in which the RPC digits are translated into bits indicating which input channels fired. Channel masking, time alignment and the introduction of an artificial dead time for the fired channels are possible, although not used yet in the present implementation. The outputs of the eight CMAs belonging to each Pad Board are collected and are searched for valid trigger coincidences in the r–z and r–ϕ views. The Sector Logic then identifies the two highest-p_T muon candidates among all the candidates from all the Pad Boards in the corresponding sector. The output of the Sector Logic, including the addresses and p_T-threshold information for the RoIs, is finally stored in the transient event store of the Athena framework from where it can be retrieved by the MuCTPI simulation package. The CMAs supply information also to the read-out system; this data path is also simulated. The resulting data are organized in a structure that follows exactly the one implemented in the hardware (i.e. bytestream format). This, together with software converters of the interpretation of the bytestream data, allows the use of the RPC data in LVL2 selection algorithms such as muFast ([13-12] and [13-13]).

The architecture model takes care of connecting together the different simulated hardware components and of simulating the data flow in the RPC muon trigger. It is built using a data-driven approach: the event data are arranged in a hierarchical structure going from the RPC digits up to the Sector Logic output. Each level in the hierarchy corresponds to a complete processing step of one hardware component, and the result of each step is available on demand. The "on-demand" approach is very flexible and allows one to save processing time since simulation is only requested for the sectors that contain data.

The architecture simulation makes use of two services that describe the setup of the trigger system: the geometry of the trigger chambers and the cabling of the hardware components. The geometry service accesses the RPC engineering parameters via the AMDB database and builds a model in which the chambers are organized in a suitable way for the LVL1 trigger (i.e. in trigger stations, building a continuous detector surface at fixed radius, and not as appendices of the

MDT chambers as it is in the database). The cabling service provides the mapping between the trigger components (RPC readout strips to CMA channels, CMAs to Pad Boards, and Pad Boards to Sector Logic) and holds the configuration of the CMAs. This configuration depends on the muon p_T thresholds which are required. The cabling data as well as the CMA configuration data are read from ASCII files.

The packages that implement the architecture model also provide a fast simulation of the trigger logic that is completely disentangled from the hardware simulation and does not take the timing of signals into account. The main application of this fast simulation is inside the LVL2 selection algorithm muFast, where it is used to emulate the functionality of the LVL1 trigger. Some information on the performance of the RPC muon trigger can be found in [13-13] and [13-14].

13.2.4 The Muon-to-CTP interface and its simulation

The Muon-to-CTP Interface (MuCTPI, see Ref. [13-15]) receives up to two muon candidates from each of the 208 sectors of the barrel (RPC) and endcap (TGC) muon triggers. From these candidates, the multiplicities of muons are calculated for up to six different muon p_T thresholds and sent to the CTP for use in making the trigger decision. If the event is retained by the LVL1 trigger, RoI information on the muon candidates is sent via the Region-of-Interest Builder to the LVL2 trigger (if there are more than 16 candidates, the 16 highest-p_T candidates are retained and a flag is set). More detailed information is sent on a separate link to the DAQ. The data sent to LVL2 and the DAQ conform to the standard ROD bytestream data format. The MuCTPI avoids double-counting of muons which pass through more than one pivot plane, e.g. in the barrel–endcap transition region.

The MuCTPI simulation follows the hardware scheme as closely as possible, down to the data formats used in the hardware. The data flow is emulated using the same stages of processing as in the hardware, including the propagation of error and status bits. Access functions are provided for every type of information available in the hardware. The simulation was originally developed as a stand-alone program for testing the prototype MuCTPI hardware. It has recently been ported to the Athena framework and integrated with the simulation of the RPC trigger on the input side, and with the simulations of the CTP and the RoIB on the output side. The output to the readout is also simulated although this is not yet used within the LVL1 simulation efforts.

13.2.5 The LVL1 CTP and its simulation

The LVL1 trigger event decision is made in the Central Trigger Processor (CTP, see Ref. [13-16] and [13-17]) in the two-step procedure that was discussed above:

The multiplicities of candidate objects from the calorimeter and muon triggers for various p_T thresholds are compared with the *trigger conditions* introduced in Section 13.2.1, checking against simple multiplicity requirements. Depending on the inputs from the calorimeter and muon trigger, each trigger condition takes a logical value TRUE or FALSE.

The trigger conditions (or rather their logical values) are combined using AND, OR and NOT operations to give complex *trigger items*. Each trigger item corresponds to a signature to be triggered on by LVL1 as defined in the trigger menu; gates and prescales can be applied to each individual item. The LVL1 trigger result is the logical OR of all trigger items.

The logical relations between the conditions and the items on one side, and the conditions and the input multiplicities on the other side, are provided by the LVL1 trigger configuration (Section 13.2.1). The CTP provides outputs to the RoIB and to the readout; the information that is sent includes bit patterns for the input signals and for the trigger items before and after prescales and vetos, as well as the final event-decision result.

In the currently-existing prototype hardware implementation of the CTP, the CTP-D ('D' for demonstrator, see Ref. [13-18] and [13-19]), the selection procedure is implemented using two look-up tables (LUT) to compute the trigger conditions and two programmable devices (CPLDs) for the combination of conditions to trigger items. The design of the final CTP is in progress based on the use of large FPGAs.

The current CTP simulation follows closely the CTP-D design — conversion to the final CTP design will be done in due course. First, the input multiplicities from the calorimeter trigger and MuCTPI simulations are collected. The multiplicities that are used in the trigger menu are checked against the respective conditions (the conditions are taken from the C++ object representing the trigger menu that is provided by the configuration step). The conditions are then combined to trigger items in a recursive algorithm. All trigger items are passed through a prescale algorithm, and the logical OR of all items is formed, resulting in the LVL1 event decision (the LVL1 accept or L1A signal which can be expressed as 0 or 1, FALSE or TRUE). The deadtime algorithms that exist in the hardware have not yet been implemented in the simulation. Finally, the CTP result object, which in content and format corresponds precisely to the one provided by the hardware, is formed and stored in the TES for later use by the RoIB.

13.2.6 Interface to the HLT

The interface between the LVL1 trigger and the HLT is the Region-of-Interest Builder (RoIB, see Ref. [13-20]). This device, which is formally part of the DataFlow system, collects the information relevant for LVL2 from the calorimeter and muon triggers and from the CTP. It combines all the data into a single block which it then passes to the LVL2 supervisor assigned to the event in question. The LVL1 data are received in S-LINK format (four links from the calorimeter trigger Cluster Processor, two from the calorimeter Jet/Energy Processor, and one each from the MuCTPI and the CTP). The RoIB has to operate at the highest foreseen LVL1 output rates without introducing additional deadtime.

The RoIB simulation picks up the information stored in the TES by the calorimeter trigger, MuCTPI and CTP simulations, and constructs a LVL1 result raw-data object (RDO). Converters are provided for translating the bytestream format into objects which serve to seed the LVL2 trigger. These contain the value (in GeV) of the threshold that has been passed and the location of the RoI in η–ϕ space.

13.3 Common tools for on-line HLT selection

The HLT algorithms are the basic software components that provide results from which the trigger decision is derived. These algorithms operate within the context and environment of the Event Selection Software (ESS) which was already discussed in Chapter 9. Section 13.3.1 provides an overview and description of the ESS from the viewpoint of the HLT algorithms. The objects of a common Event Data Model, i.e. the objects that the algorithms exchange and manipulate, are described in Section 13.3.2. A description of the HLT algorithms intended to operate

in the LVL2 environment is given in Section 13.3.3, while the algorithms for the EF are described in Section 13.3.4.

13.3.1 Algorithmic view of the Event Selection Software

Unlike their counterparts from the offline reconstruction, HLT algorithms must allow themselves to be guided by the Steering of the Event Selection Software. The Steering is configured with sets of Sequence and Menu Tables and guides the HLT algorithm processing using so-called Trigger Elements. Trigger Elements characterizing abstract physical objects with a label (e.g., 'electron', 'jet', or 'EMRoI') and effectively decouple the Steering of the trigger selection from details of the reconstruction Event Data Model. The ESS uses the seeding mechanism (see Section 9.5) to guide the HLT algorithm processing to the those subsets of event data that correspond to the LVL1 RoIs.

The HLT trigger processing is data driven, because the set of LVL1 RoIs determines which of the predefined Sequences are to be executed for this event. Each of the LVL1 RoI objects are associated to a Trigger Element on which the Steering acts upon. For each Trigger Element, the Steering executes the required HLT algorithms as defined in the Sequence Table. Hence, it is possible that a given algorithm may be executed several times per event. This is fundamentally different to the 'Event Loop' approach of the offline reconstruction paradigm where a given offline algorithm acts only once upon each event.

During the LVL2 trigger latency event data remain within ROBs unless and until they are actively requested. The LVL2 algorithms request and process only a small fraction of the event data, leading to a very substantial reduction in the network and computation resources that would otherwise be required.

The first step in the process of obtaining data for a RoI is the conversion of a geometrical region (e.g. a cone with an extent η and ϕ) into Identifiers; this is accomplished with the HLT Region Selector [13-21]. Presently these Identifiers are the `IdentifierHash`es used in the offline software. Each Identifier corresponds to a Detector Element in a sub-detector, e.g., a Pixel Module or a sampling of a LAr Trigger Tower. The Region Selector uses information from the detector description during its initialization phase to build an so-called `EtaPhiMap` for each layer (or disk) of a sub-detector. This map is essentially a two-dimensional matrix in η and ϕ. Each element of the matrix consists of a list of Identifiers. An HLT Algorithm uses the Region Selector interface, which is the name of the sub-detector to request data from (*i.e.*, Pixel, SCT, TRT, LAr, Tile, MDT, RPC, CSC, or TGC) and the physical extent of the geometrical region, to request the list of identifiers. Given the vastly different designs of the sub-detectors, different procedures are used dependent on sub-detector.

Interactions with the Data Collection system are hidden from the HLT algorithm behind a call to Storegate. Within Storegate, sub-detector data are aggregated into collections within a container. Each collection is labelled with the Identifier corresponding to the Detector Element. Algorithms request event data from Storegate using the set of Identifiers obtained by the HLT Region Selector. If the collections are already within Storegate, it returns them to the HLT algorithm. If not, Storegate uses the Raw Data Access scheme (Chapter 9) to request the corresponding ROB data from the Data Collection system. A converter translates the Raw Data into either Raw Data Objects (RDOs) or, by invoking an Data Preparation algorithm, into Reconstruction Input Objects (RIOs). The obtained RDOs or RIOs are stored within the collections within the Container within Storegate.

13.3.2 Event Data Model Components

During 2002 and 2003, there has been a substantial effort within the HLT, offline, and sub-detector communities to establish a common Event Data Model (EDM). In Section 13.3.2.1 the concept of Detector Elements is discussed as an organizing and identifying principle for most EDM objects. The raw-data model, that forms the present agreed organization of trigger objects and their relationship, is described in Section 13.3.2.2 and reconstruction data-model classes specific to LVL2 and EF algorithms are described in Section 13.3.2.3.

It is worth noting that a large amount of effort has been invested by ATLAS as a whole (TDAQ, offline and detector communities) in making as realistic as possible a simulation of the data that the HLT/DAQ will have to deal with.The detector response is simulated using a detailed GEANT3-based simulation of the ATLAS detectors. The simulation of the raw data includes the effects of pile-up in the bunch crossing that gave rise to the LVL1 trigger, and also in other bunch crossings nearby in time, with an average number of interactions per bunch crossing chosen according to the relevant luminosity scenario. Electronic noise is also included in the simulation. The “time-frame” of digitized data (i.e. measurements in successive bunch crossings) are then processed to reproduce in as much detail as possible the raw data as they will appear at the output of the Readout System. For example, in the LAr calorimeter digitizations from five successive bunch crossings are combined to extract measurements of energy, time and quality. In the experiment, this signal processing will be performed at the level of the Readout Driver.

13.3.2.1 Event Data Organization

Event Data (e.g., Raw Data Objects (RDOs) and Reconstruction Input Objects (RIOs)) are aggregated into collections corresponding to adjacent readout channels within the physical detector. These collections reside in an Containers with collection Identifier labels corresponding to the unit of aggregation. For most sub-detectors, the organizing principle is that of the Detector Element.

In the Pixel detector a Detector Element is a module, equivalent to a single Silicon wafer; there are 1744 such elements. For the SCT, a Detector Element is one side of a module, equivalent to a bonded pair of wafers whose strips are oriented in a single direction (*i.e.*, axial or stereo); there are 8176 SCT Detector Elements. For the TRT no such direct correspondence to detector module exists. There are 19008 TRT Detector Elements [13-22].

For the calorimeters, the concept of Detector Element is difficult to define. Instead, the organizing principle for event data is that of the LVL1 Trigger Tower and sampling.

Within the barrel muon spectrometer an MDT Detector Element corresponds to one of the two multi layer chambers in a station. For the LVL2, the RPC data is organised in Detector Elements according to the LVL1 coincidence logic. A Detector Element is identified with the RPC pad associated with the second station. For the EF the RPC data is reorganised into a Detector Element for each station. A TGC Detector Element is one TGC η division, or chamber, in a TGC station; there are 24 forward stations in a ring and 48 endcap stations in a ring, and there are four rings at each end of the ATLAS detector. Finally, the CSC Detector Element is a single CSC chamber in a station. A CSC station typically has two multilayers.

13.3.2.2 Raw Data Model Components

Byte Stream Raw Data are ROB-formatted data produced by the ATLAS detector or its simulation [13-22]. The format supports a set of hierarchical fragments, where only the bottom level, the ROD fragment, is defined by the sub-detector group. Although the format of the Byte Stream has not been completely finalized, the detector groups have implemented the Raw Data simulation according to the best available information.

A Raw Data Object (RDO) is uncalibrated Raw Data converted into an object representing a set of readout channels. Historically these data have been referred to as *Digits*. It is the representation of Raw Data which is put into the Storegate and is potentially persistifiable.

The purpose of the RDO converters is dual: first a Byte Stream file can be created by taking the information from already created RDOs (in the transient store, from ZEBRA); second, a Byte Stream file can be read by the converters to fill the RDOs (or the RIOs for LVL2). The same converters are used for decoding the Raw Data from the online Data Collection. Since the RDOs are a representation of the specific detector output, their content can change during the lifetime of the sub-detectors.

A detailed description of the Raw Data Model components is available elsewhere [13-23].

13.3.2.3 Reconstruction Data Model Components

Algorithms interact with Reconstruction Input Objects (RIOs) as opposed to RDOs. For each sub-detector system, classes of RIOs have been defined and are described in the following subsections.

13.3.2.3.1 Inner Detector

For the Inner Detector the RIOs are organised into Detector Element collections.

The Pixel and SCT RIOs are Clusters. A Cluster in the Pixel detector is a two-dimensional group of neighbouring readout channels in a Detector Element. A Cluster in the SCT is a one-dimensional group of neighbouring readout channels in a Detector Element. For Pixel and SCT, there are currently two implementations of the Cluster class: one used for the EF and offline, and one used for LVL2. The one used in the EF has Pixel and SCT sharing the same class. For LVL2 there is a common structure for Pixel, SCT and TRT, but each subdetector has its own concrete class. Both the LVL2 and EF sets of cluster classes contain a list of RDO identifiers from which the cluster is built. The number of member functions is limited in both set of classes and the member functions follow the Inner Detector Requirements [13-22].

At LVL2, Pixel and SCT RIOs are converted to 3-dimensional coordinates in the ATLAS global coordinate system using an HLT algorithm to create Space Points. These algorithms accept as input a Container of Cluster Collections of the appropriate type and return an Container of Space Points.

For the Pixels, the creation of Space Points consists of combining the local coordinates of Clusters with information on the position and orientation of the Detector Element to give the global coordinates. The process for the SCT is more complicated since a single SCT detector provides only a one-dimensional measurement. However, an SCT module, consisting of two detectors in a stereo-pair, provide two-dimensional information. One species of SCT Detector Element, phi-

layer, has strips orientated parallel to the beam axis, the other, *u* or *v* layer, is rotated by ± 40 mrad with respect to those of the phi-layer Detector Elements. The formation of Space Points consists of the following steps:

- Associate each phi-layer Cluster Collection[1] with the corresponding stereo-layer Cluster Collection.
- For each pair of Collections (phi + stereo), take each phi-layer Cluster and search for associated stereo-layer Clusters. If there is more than one associated stereo layer Cluster, a Space Point is formed for each one (in this case, at most one will be a correct measurement, the others will form 'ghost' points). If no associated stereo-layer hit is found, a point is created from the phi-layer information alone.
- Calculate the second coordinate (*z* for the barrel, or *R* for the end-caps).
- Using information on the position and orientation of the Detector Element transform to global coordinates.

Note that for the LVL2 Space Points a simplifications is made in the interest of speed. No attempt is made to form Space Points from Tracks passing close to the edge of a module, where the corresponding stereo-layer Cluster is in a different module.

The TRT RIO gives the drift circle of a straw. Here, the same classes are used for LVL2, EF and offline. The granularity of the TRT RIO is the same as for the RDO: that of a straw, thus the RIO contains an identifier which is the offline identifier for a straw. In the case of the RDO the straw information is uncalibrated, while in the case of the RIO the straw information is calibrated, i.e. the drift radius is calculated from the drift time. For the current prototype, the time–radius function is the same for all straws, but in the future the parameters of the function will come from the Interval of Validity Service [13-24].

13.3.2.3.2 Calorimeters

For the Calorimeters, the RIOs are calibrated calorimeter cells (`LAr` and Tiles), the same as in the offline reconstruction.

Both LAr and Tile have a common `CaloCell` base class which represents the measurement in the calorimeters of energy, position, time and quality. A `CaloCell` has been calibrated so that it returns the physical energy deposit *in the cell* with units of GeV, but without any kind of leakage corrections. Time is given in nanoseconds and refers to when the deposit occurred, relative to the trigger; it should be consistent with zero for good hits. Quality reflects how well the input to the system matched the signal model on which the algorithm is based.

13.3.2.3.3 Muon Spectrometer

For the barrel Muon Spectrometer it was found expedient to use in part RDOs instead of RIOs as input to the HLT muon-selection algorithms. The RDOs are organized inside Storegate in identifiable collections and can be accessed in the same way as RIOs. For the MDTs, the RDOs and RIOs are ordered in collections that correspond each to an MDT Detector Element, i.e. an MDT chamber. Each RDO [13-25],[13-26],[13-27] contains the information of one MDT read-out channel, *i.e.* one tube. The information is the time of the leading edge of the MDT pulse and the

1. There is a Cluster Collection per Detector Element.

charge integrated over the first 20 ns. From this, the uncalibrated drift time can be calculated. The RIOs contain the calibrated drift time.

The definition of RPC RDOs is complicated by the fact that the RPCs are trigger chambers. Their read-out is optimised for the LVl1 trigger task and does not reflect an easily identifiable geometrical structure such as an RPC chamber with its strips. Consequently, RPC RDOs are ordered following the read-out structure, by PADs and CMAs. Each RDO corresponds to a fired CMA channel [13-28], [13-29]. The RDOs are organized in CMA objects, i.e. in collections corresponding to one CMA each. The CMA objects in turn are organized per PAD, i.e. in collections corresponding to one PAD each.

One RPC RIO corresponds to an RPC Detector Element. An RPC RIO contains the information of a collection of RPC strips that fired. There is no simple correspondence between RPC strips and RPC read-out channels. In order to translate a fired RPC read-out channel, which is a CMA read-out channel, into a RPC strip, a cable map and processing of the information of the CMAs for the opposite view is required.

Corresponding class definitions are being developed for the TGC and CSC detectors.

13.3.3 HLT Algorithms for LVL2

LVL2 is the most demanding step, in terms of required system performances (latency, data transfer, etc.), of the HLT trigger selection. In the following the present view of the algorithms needed to implement LVL2 selection is given. It is worthwhile noticing that several options for using in the best possible way detector information are taken into account, hence more than one algorithm is available to accomplish a defined task. This will allow to implement a robust, flexible and redundant selection scheme, which will be studied with present and future simulations.

13.3.3.1 IDSCAN

IDSCAN (see Refs. [13-30] and [13-31]) is a track-reconstruction package for LVL2. It takes as input Space Points found in the Pixel and SCT Detectors. A series of sub-algorithms (Z-Finder, Hit Filter, Group Cleaner, Track Fitter) then processes these inputs and output Tracks and the Space Points associated with them.

The Z-Finder determines the *z*-position of the primary interaction vertex. The algorithm assigns the input hits to narrow ϕ-bins. For each bin it then extrapolates lines joining pairs of hits back to the beam-line, entering in a histogram the *z* values obtained for all hit-pairs. After integrating over all ϕ-bins, it takes as the *z*-position of the primary vertex the histogram region with the most entries.

The Hit Filter finds groups of hits compatible with Tracks coming from the *z* position found by Z-Finder. It puts all hits into a histogram binned in ϕ and η. It then finds clusters of hits within this histogram. It creates a *group* of hits if such a cluster has hits in more than a given number of layers.

The groups of hits found by the Hit Filter are used by the Group Cleaner which splits them into Tracks and removes noise hits. Each triplet of hits forms a potential track for which p_T, ϕ_0, and d_0 are calculated. It forms groups from these triplets with similar parameters, applying certain quality cuts. It accepts a track candidate if a group contains enough hits.

Finally, the Track Fitter verifies track candidates and calculates the track parameters by using a fitting algorithm adapted from SCTKalman [13-32][13-33][13-34]. It returns a list of Space Points on the Track, the Track parameters, and an error matrix.

13.3.3.2 SiTrack

SiTrack is a track-reconstruction package for LVL2 which extends and upgrades a previous algorithm called PixTrig. SiTrack takes Pixel and SCT Space Points as input, and outputs fitted reconstructed Tracks each storing pointers to the Space Points used to build it. SiTrack is implemented as a single main algorithm SiTrack which executes a user-defined list of sub-algorithms (chosen among ST-Space Point Sorting, ST-Muon Vertex, ST-Track Seeding, and ST-Three Point Fit).

The ST-Space Point Sorting collects the Space Points coming from the Pixel and SCT detectors and sorts them by module address, storing the result in a Standard Template Library (STL) map. This processing step is performed in order to speed-up data access for the other reconstruction sub-algorithms.

The ST-Muon Vertex is a primary-vertex identification algorithm mostly suitable for low-luminosity events with a high-p_T muon signature. It is based on track reconstruction inside a LVL1 muon RoI — the highest-p_T track is assumed to be the muon candidate or, failing that, to come from the same primary interaction as the muon. The primary-vertex position along z is taken from the point of closest approach of the track to the beam line.

The ST-Track Seeding uses the sorted Space Point map and a Monte Carlo look-up table (MC-LUT) linking each B-layer Pixel module to modules belonging to other logical layers. It builds track seeds formed by two Space Points and fits them with a straight line; one or more logical layers can be linked to the B-layer, the latter option being particularly useful if robustness to detector inefficiencies must be ensured. If the primary vertex has already been reconstructed by ST-Muon Vertex, a fraction of fake track seeds can be rejected during their formation, applying a cut on their z distance from the primary vertex.

If no vertex information is available, a histogram, whose resolution depends on the number of seeds found, is filled with the z impact parameter of each seed. The z position for the primary vertex is then determined from the position of the maximum in the histogram. This vertexing algorithm, which can be operated in both RoI and full-scan modes, is best suited for high-luminosity events containing many high-p_T tracks (e.g., b-tagging). Independent cuts on r-ϕ and z impact parameters are eventually applied to the reconstructed seeds to further reduce the fake fraction.

ST-Three Point Fit extends track seeds with a third Space Point; it uses a map (which is built using Monte Carlo data) associating to each seed a set of module roads[1] the track could have hit passing through the Pixel or SCT detectors. A subset of modules is extracted from each road according to a user-defined parameter relating to their ‘depth’ inside it (e.g., the user can decide to use modules at the beginning or in the middle of each road, etc.). Space Points from the selected modules are then used to extend the seed, and candidate tracks are fitted with a circle. Ambiguities (e.g., tracks sharing at least one Space Point) can be resolved on the basis of the track quali-

1. A road is a list of modules ordered according to the radius at which they are placed starting from the innermost one.

ty, to obtain an independent set of tracks that can be used for trigger selection or as a seed for further extrapolation.

13.3.3.3 TRTLUT

TRT-LUT is a LVL2 algorithm for track reconstruction in the TRT. It is described in detail elsewhere [13-35][13-36]. The algorithm takes as input Hits in the TRT. The algorithmic processing consists of Initial Track Finding, Local-Maximum Finding, Track Splitting, Track Fitting and Final Selection. It outputs the Hits used and Tracks with their parameters.

During the Initial Track Finding a histogramming approach is employed. A two-dimensional histogram is constructed with bins in ϕ and Q/p_T for the track parameters at the primary vertex (Q indicates the electric-charge sign). For each hit in the event, the contents for all ϕ and Q/p_T bins consistent with the hit position are incremented. The bin corresponding to the true parameters of a track will receive entries for all hits on that track. A very fast implementation of the algorithm is achieved by using a Look-Up Table (LUT) to obtain the list of bins associated with given hit. Having built the histogram, tracks can be found by Local Maximum Finding.

The Track Splitting stage of the algorithm analyzes the pattern of hits associated to a track candidate. By rejecting fake candidates composed of hits from several low-p_T tracks, the track splitting step results in an overall reduction by a factor of roughly two in the number of track candidates. For roads containing a good track candidate, it identifies and rejects any additional hits from one or more other tracks. The result of the overall Track Splitting step is a candidate that consists of a sub-set of the straws within a road.

The final step of TRT-LUT, Track Fitting and Final Selection, performs a fit in the r-ϕ (z-ϕ) plane for the barrel (end-caps) using a third-order polynomial to improve the measurement of ϕ and p_T. Only the straw position is used (*i.e.*, the drift-time information is not used). The track is assumed to come from the nominal origin. After the fit, a reconstructed p_T threshold of 0.5GeV is applied.

13.3.3.4 TRTKalman

TRT-Kalman [13-37][13-38] is a package based on xKalman++ (see Section 13.3.4.1). The name is in fact a misnomer since the Kalman-filter component of xKalman++ is not used for the TRT; a histogram search and least-squares fit is used instead.

13.3.3.5 T2Calo

T2Calo (see Refs. [13-39], [13-40], [13-41] and [13-42]) is a clustering algorithm for electromagnetic (EM) showers, seeded by the LVL1 EM trigger RoI positions [13-43]. This algorithm can select isolated EM objects from jets using the cluster E_T and shower-shape quantities.

The RIOs from which it starts are calibrated calorimeter cells (LAr or Tiles), as in the offline reconstruction. The output is a LVL2-specific class containing the cluster energy and position, and the shower-shape variables useful for the selection of EM showers.

The first step in T2Calo is to refine the LVL1 position by finding the cell with highest energy in the second sampling of the EM calorimeter. This position (η_1, ϕ_1) is later refined by calculating the energy-weighted-average position (η_c, ϕ_c) in a window of 3×7 cells (in $\eta \times \phi$) centred

around (η_1, ϕ_1), using as weights the energy in the second sampling. As described in Ref. [13-40], a number of parameters are calculated for use selecting EM clusters and rejecting background due to jets:

- In sampling two, $R^{shape}_{\eta} = E_{37} / E_{77}$ is calculated. The expression E_{nm} stands for the energy deposited in a window of $n \times m$ around (η_1, ϕ_1).
- In sampling one, $R^{strip}_{\eta} = (E_{1st} - E_{2nd}) / (E_{1st} + E_{2nd})$ is obtained in a window of $\Delta\eta \times \Delta\phi = 0.125 \times 0.2$ around (η_c, ϕ_c). Here E_{1st} and E_{2nd} are the energies of the highest and second-highest local maxima found after summing over the two strips in ϕ for each position in η within the window. A local maximum is defined as a single strip with energy greater than each of its two adjacent strips.
- The total energy, E, deposited in the EM calorimeter is calculated in a window of 3×7 cells around (η_1, ϕ_1), summing over all three samplings.
- Finally, the energy that leaks into the hadron calorimeter E^{had} is calculated in a window of size $\Delta\eta \times \Delta\phi = 0.2 \times 0.2$ around (η_c, ϕ_c).

13.3.3.6 muFast

The muFast algorithm is a stand-alone LVL2 tracking algorithm for the Muon Spectrometer. Previous versions are described in detail elsewhere [13-13].

The algorithm is steered by the RoI given by the LVL1 Muon Trigger and uses both RPC and MDT measurements. At present this algorithm is limited to the barrel region and is based on four sequential steps:

1. LVL1 emulation: pattern recognition in the MDT system is initiated by the RPC hits that formed the LVL1 track candidate. The RoI information obtained directly from LVL1 (via the RoI Builder) gives only the position of the track at the pivot plane. Details of the RPC hits that made up the LVL1 track therefore have to be obtained by running a fast algorithm that repeats the basic logic of the LVL1selection.
2. Pattern recognition is performed using the RPC hits that made up the LVL1 track candidate to define a road in the MDT chambers around the muon trajectory. MDT tubes lying within the road are selected and a contiguity algorithm is applied to remove background hits not associated with the muon track.
3. A straight-line track fit is made to the selected tubes (one per tube monolayer) within each MDT station. For this procedure the drift-time measurements are used to exploit fully the high measurement accuracy of the muon-tracking system. The track sagitta is then evaluated by combining the measurements from the different stations.
4. A fast p_T estimate is made using LUTs. The LUT encodes, as a function of η and ϕ, the linear relationship between the measured sagitta and Q/p_T.

The output of this algorithm is a measurement of the muon p_T at the production point, together with η and ϕ. The detailed description of the implementation of muFast in the trigger framework is given in Ref. [13-44].

13.3.3.7 muComb

The combination of the features of tracks measured at LVL2 in the muon spectrometer and the Inner Detector (ID) provides rejection power against decays in flight of charged pions and kaons (i.e. $\pi^+ \rightarrow \mu\nu$ and $K^+ \rightarrow \mu\nu$), and of fake muon-spectrometer tracks composed of hits induced by the cavern background. In addition, the combination of the two measurements improves the momentum resolution of reconstructed muons over a large momentum range.

The matching of tracks between the muon spectrometer and the ID can be performed extrapolating the ID track to the muon detectors. The procedure needs to take into account the detector geometry, the material distribution and the inhomogeneity of the magnetic field. An accurate extrapolation would require the use of detailed geometry and magnetic-field databases, together with a fine tracking. All this would be expensive in terms of CPU time and therefore not acceptable for the LVL2 trigger.

To provide a fast tracking procedure, the effects of the geometry, material and magnetic field have been parameterized using simple analytic functions of η and ϕ. The extrapolation of ID tracks to the entrance of the muon spectrometer is performed using straight-line extrapolation in two independent projections, the transverse and the longitudinal views, corrected for bending in the magnetic field and energy-loss effects. In the transverse projection the ID track extrapolation in ϕ is corrected as follows:

$$\Delta\varphi = \frac{Q\alpha}{p_T - p_T^0} \qquad 13\text{-}1$$

where α is related to the field integral, and p_T^0 accounts for the transverse-energy loss in the material of the calorimeter that is approximately independent of the track transverse momentum, p_T. Both α and p_T^0 have been determined by fitting $\Delta\varphi$ for simulated muons as a function of p_T. It is found that $p_T^0 \sim 1.5$ GeV, *i.e.* about half of the transverse-energy loss of low-momentum muons, as naively expected. A similar approach has been followed for the extrapolation of the *z*-coordinate in the longitudinal view.

The matching is done using cuts on the residuals on the positions in the orthogonal coordinates *z* and $R\phi$. For matching tracks the transverse momentum of the muon is estimated using a weighted average of the independent p_T measurements in the muon spectrometer and the ID. For each combined track, a χ^2 function is used to evaluate the quality of the p_T matching. Reconstructed muons from π and K decays in flight often give high χ^2 values (e.g. because the ID measures the hadron momentum while the muon-spectrometer measures that of the muon which is lower) and thus can be rejected thanks to the excellent resolution of both the ID and muon-spectrometer measurements.

13.3.4 HLT Algorithms for EF

The baseline option for the selection at the Event Filter stage is to adopt algorithms from the offline software suite. This will allow the online selection to easily benefit from improvements stemming from offline analyses and to avoid duplication of efforts. In the following, a preliminary list of the presently available algorithms and their characteristics can be found, which have been used in the EF selections described in Section 13.4.

13.3.4.1 xKalman++

xKalman++ is a package for global pattern recognition and Track fitting in the Inner Detector for charged tracks with transverse momentum above 0.5GeV. A more detailed description of this algorithm is available elsewhere [13-45].

The algorithm starts the track reconstruction in the TRT using a histogramming method or in the Pixel and SCT detector layers using segment search.

The first reconstruction method outputs a set of possible track-candidate trajectories defined as an initial helix with a set of parameters and a covariance matrix. As a second step the helix is then used to define a track road through the precision layers, where all the measured clusters are collected. xKalman++ attempts to find all possible helix trajectories within the initial road and with a sufficient number of clusters.

The other method, where track finding starts in the Pixels or SCT, outputs a set of Space Points as an initial trajectory estimation. In the next step the Space Points serve as an input for the Kalman filter-smoother formalism that will add the information from the remaining precision layers. Each reconstructed Track is then extrapolated into the TRT, where a narrow road can be defined around the extrapolation result. All TRT Clusters together with the drift-time hits found within this road are then included for the final track-finding and track-fitting steps.

There are three seeding mechanisms available in the offline environment: the reconstruction of the full event; the reconstruction of a region-of-interest and (available soon) EM calorimeter seeding. In the HLT environment, used as an EF algorithm, xKalman++ will be seeded by the LVL2 result.

After the pattern-recognition and Track-fitting steps, xKalman++ stores the final Track candidates in a collection in Storegate. The Track candidate contains the following information:

- fit procedure used;
- helix parameters and their covariance matrix at the end-points of the filter procedure in the precision layers (point on the trajectory closest to the vertex) and in the TRT (point on the trajectory closest to calorimeter);
- total χ^2 resulting from final fit procedure;
- list of all hits on track from all subdetectors;
- total number of precision hits N_p.
- total number of straw hits N_s, empty straws crossed N_e, and of drift-time hits N_t.

Furthermore, a track candidate is stored in the final output bank if it passes the following cuts:

- the number of precision hits is larger than 5 to 7;
- the ratio $N_s/(N_s+N_e)$ is larger than 0.7 to 0.8;
- the ratio N_t/N_s is larger than 0.5 to 0.7;
- no track accepted previously has the same set of hits as the current one — this last cut removes full *ghost tracks*.

13.3.4.2 iPatRec

iPatRec [13-46] is a pattern-recognition algorithm used in the Event Filter that searches for tracks starting from Space Point combinations. Pixel and SCT Space Points are used to form Track candidates. Candidates are extrapolated to the TRT and drift-time hits are added. At the initialization phase, iPatRec creates a geometry data-base describing the properties of each detector module in the precision tracker plus the module's relationship to a simplified material model. This model consists material “layers” assumed to be either concentric cylinders in the barrel region or planes normal to the beam-axis in the end-cap regions. Additional “layers” represent the TRT detector, beam-pipe and inert support / service material. Track finding, following and fitting procedures make extensive use of this data-base. Another initialization task is to parameterize the magnetic field to enable a fast propagation of track parameters between layers.

In the first step of event reconstruction, adjacent Raw Data channels are clustered, and Space Points are produced from these Clusters. Each Space Point is assigned to one of seven partitions according to its distance from the beam intersection region. Within each partition the points are ordered according to their azimuthal coordinate. The general procedure is to form Track candidates using Space Point combinations from three different partitions, subject to criteria on maximum curvature and crude vertex region projectivity. Candidates then undergo a Track Fit procedure to give Track parameters with covariance at the point of closest approach to the beam-line (perigee parameters). The Track Follower algorithm propagates these parameters to form an intersect with error ellipse at each “layer” in turn. Clusters from the traversed detectors are associated to the Track. Final allocation of Clusters are taken after a further track-fit. During this fit, energy loss and Coulomb scattering are taken into account by allocating a scattering centre (with associated extra fit parameters) to each “layer” traversed. An active detector region traversed without an associated cluster is classified as a “hole” and retained for material and quality information. Tracks with a fit probability greater than 0.001 and a maximum of three holes are extrapolated to the TRT, where a histogramming technique is used to select the TRT hits to be added to the Track. Tight cuts are made on the straw residual and on the ratio of found to expected straws, in order to limit high-luminosity occupancy effects.

Tracks with clusters in the two innermost Pixel layers plus TRT association are called primary Tracks. Otherwise at most one hole is allowed: truncated Tracks start in the innermost layers but cannot be followed to the outermost layers or TRT; secondary Tracks start further out and are required to have TRT association. Various partition combinations are taken to maintain track-finding efficiency for the three types of Track even in the event of a higher than expected detector inefficiency. To avoid track duplication, only candidates with two unallocated Space Points are initiated, and tracks sharing more than 50% of their Clusters are deemed ambiguous whence only the one with higher quality is retained.

To speed up the execution, a preliminary track-finding pass looks only for high-quality primary Tracks and finishes as soon as one is found. The vertex from this Track is then used to subdivide the Space Point partitions into projective slices, greatly reducing the combinatorial load. Impact-parameter criteria are adjusted according to the distance to the first cluster to ensure there is no bias against b-, c- or s-particle decays. The code iterates to allow for several interaction vertices, very necessary at high luminosity, and reverts to a slower algorithm when no high quality tracks are found. A special fit procedure is available to better handle electron bremsstrahlung. This is invoked from the subsequent combined reconstruction for tracks associated to an EM-calorimeter cluster.

13.3.4.3 LArClusterRec

LArClusterRec is the reconstruction package for electromagnetic clusters in the calorimeter; it is organized in two steps as described below.

In the first step, towers are created by summing the cells of the electromagnetic calorimeter and the pre-sampler in depth using a granularity of $\Delta\eta \times \Delta\phi = 0.025 \times 0.025$ corresponding to the granularity in the second sampling of the EM calorimeter. The inputs to the tower building are the calibrated calorimeter cells which are produced by the package LArCellRec.

In the next step, a sliding-window algorithm is used. If a local maximum is found with total E_T in the window above a given threshold, a cluster is created which is subsequently stored in the cluster container. To reconstruct the cluster energy and position is calculated in a given window.[1] The cluster energy is corrected for η and ϕ modulations of the calorimeter response and for leakage outside the cluster in a given window. In the region between the barrel and end-cap calorimeters, the cluster energy is in addition corrected for energy losses using the energy deposit in the crack scintillators. The η position in the first and second sampling is corrected for s-shapes, the ϕ position is corrected for an offset: both of them are geometry effects.

13.3.4.4 egammaRec

EgammaRec is an algorithm designed to calculate quantities that can be used to separate electrons and photons from jets. To do so, electromagnetic-cluster and tracking information are used.

In the electromagnetic calorimeter electrons are narrow objects, while jets tend to have a broader profile. Hence, shower shapes can be used to reject jets. This is handled by the EM Shower Builder which calls a number of algorithms which calculate diverse quantities using information from the first and second samplings of the electromagnetic calorimeter, as well the first sampling of the hadronic calorimeter.

Cluster and track information are combined in the Track Match Builder. For a given cluster all tracks are examined in a window around the cluster position. In case more than one track is found, the one with the highest p_T is retained. If the *E/p* ratio is $0.5 < E/p < 1.5$, the track match is successful. In the subsequent particle-identification step the information provided by egammaRec can be used. In the case of an electron hypothesis, jets can be rejected by analysis of the EM shower shape, tight track-quality cuts, *E/p* matching, and the position match in η and ϕ directions between the Cluster and the Track. Photons can be selected by analysing the EM shower shapes, using reconstruction of photon conversions in the Inner Detector, and, possibly, using a Track veto for non-converted photons.

13.3.4.5 Moore

Moore is an offline track reconstruction package for the Muon Spectrometer. A detailed description of Moore is available elsewhere [13-47]. Moore reconstructs tracks in the full η range (barrel+endcaps) of the muon spectrometer. However, we will restrict the description to the barrel since at present only this region is included in the trigger chain (Muon vertical slice).

1. This window can be different from the one used for the sliding window algorithm.

In the standard offline configuration Moore takes as input collections of digits or clusters inside the muon spectrometer and outputs fitted reconstructed tracks whose parameters are expressed at the first measured point inside the muon spectrometer. The reconstruction flow is desegregated into sequential steps, and each step is driven by an algorithm module that builds partial or final reconstruction objects. Each algorithm retrieves objects created by the previous modules from Storegate and builds transient objects that are subsequently recorded in Storegate from where they are available for other algorithms. Data and algorithms are strictly separated: the algorithms should know the structure of the data objects that they are accessing or producing, but the objects do not depend on the algorithms. The existence of only a dependence from algorithms to data and the flowing sequence of the reconstruction steps allows to establish which algorithm will produce an object at run-time. At present the overall reconstruction starts from the searches of regions of activity performed in two consecutive steps: one searches for activity regions in the ϕ trigger hits from RPC and another performs the search on the *r-z* view considering the precision hits of MDTs.

Inside the MDTs the drift distance is calculated from the drift time by applying various corrections to the simplest linear formula. These corrections concern the time of flight, the second coordinate and the propagation along the wire, and the Lorenz effect. The best Track segment is chosen form the possible four lines tangential to the drift circles. All the MDT segments of the outer station are combined with those of the Middle layer. The MDT hits in the segments are combined with the RPC ϕ information, forming outer Track candidates. All the successfully fitted candidates are kept for further processing.

Each outer Track that is successfully reconstructed is subsequently used to associate inner-station MDT hits. A final track is defined as a collection of RPC hits and MDT hits successfully fitted from at least two layers. The parameters of the fitted Tracks are specified at the first measured point inside the Muon Spectrometer. In order to be used for physics studies an extrapolation of the Track parameters to the production point is needed. To accomplish this task a different offline package, Muon Identification (MuId), is used. In MuID the multiple scattering is parameterized as scattering planes in the calorimeters and the energy loss is evaluated from the calorimeter measurements or from a parameterization as a function of η and the muon momentum. After a refit at the vertex the Track parameters are used for further analysis.

When dealing with data already selected by the trigger the first two steps can be substituted by *ad hoc* makers that seed the track search in the regions selected by the trigger, by using the HLT Region Selector. We refer to this way of executing as “seeded mode”. More details regarding the implementation of the Moore package in HLT are given in Ref. [13-48].

13.4 Signatures, rates and efficiencies

In the following subsections, we present a preliminary assessment of the physics performance of the trigger using algorithms for LVL2 and EF applied to representative classes of final-states: electrons and photons; muons; jets, taus and missing transverse energy; b-jets; and B-physics. This broad classification stems from the physics goals of the ATLAS experiment, as explained in Chapter 4 and in Ref. [13-49].

Whenever possible, realistic raw data formats have been used as a starting point for the HLT processing, using a very detailed simulation of the raw data as they will appear when accessed from the Read Out System. This is deemed to be extremely important since, as described in the following, the steps that prepare the data for the algorithmic processing represent a sizeable

fraction of the total computing time, even when using optimized data-preparation software. We would like to underline that this has been made possible thanks to the close collaboration between the TDAQ and the Detector groups.

Steering control (as described in Section 9.5) has also been employed, highlighting the flexible boundary between LVL2 and EF. Selection schemes are then derived, which contain the signatures used to decide whether or not to reject events.

In order to maximize the discovery potential, the selection schemes generally only use inclusive signatures. With the exception of B-physics, exclusive reconstruction of particle decays is not required and topological variables[1] are not used in the selection, although this is technically feasible both at LVL2 and in the EF (e.g. to select $Z \rightarrow l^+l^-$ decays exclusively).

It must be emphasised that system performance (e.g. minimizing execution time and volume of data needed) is a major requirement in implementing the HLT selection, to comply with the constraints imposed by the on-line environment and available resources. In this chapter also some indication of the compliance with the requirements will be given for representative selections. Since, obviously, system and physics performance are correlated, all results have been obtained after trying to optimise both simultaneously.

13.4.1 e/gamma

In the present view of the ATLAS trigger menus, the inclusive electron and photon triggers are expected to contribute an important fraction of the total high-p_T trigger rate. After the selection in LVL2 and the EF, the remaining rate will contain a significant contribution due to events from Standard Model physics processes containing real isolated electrons or photons (b, c $\rightarrow$eX, W $\rightarrow$eν, Z $\rightarrow$ee, direct-photon production, etc.). According to the general guidelines indicated in Chapter 4, the identification of these trigger objects is made using only inclusive selection criteria.

The electron and photon triggers can be viewed as a series of selection steps of increasing complexity. At LVL1, electrons and photons are selected using calorimeter information on a reduced granularity. After receiving the LVL1 electromagnetic (e.m.) trigger RoI positions, the LVL2 trigger performs a selection of isolated e.m. clusters using the full calorimeter granularity within the RoI of size $\Delta\eta \times \Delta\phi$=0.2 $\times$0.2. To calculate the cluster E_T, the most up-to-date available calibration data are used (see Section 13.3.3.5). Electrons and photons are selected based on the cluster E_T and shower-shape quantities that distinguish isolated e.m. objects from jets. A further, more refined calorimeter-based selection may classify the e.m. cluster as a candidate photon trigger object. In the next step, electrons are identified by associating the e.m. cluster with a track in the Inner Detector. In general, track candidates are found by independent searches in the TRT and SCT/Pixel ('Precision') detectors in the region identified by the LVL1 RoI. Details of the different LVL2 tracking algorithms that are currently being studied are described in Section 13.3.3.1, Section 13.3.3.2 and Section 13.3.3.4. For the results presented in the next section, however, only IDscan (Section 13.3.3.1), has been used; work is in progress to evaluate the other algorithms. For electron candidates, matching in position and momentum between the track and the cluster is required. Electron candidates passing the LVL2 selection criteria are retained to be examined by the EF.

1. E.g., variables, such as invariant masses, constructed by combining of several high-p_T objects.

In the EF, electrons are selected with a very similar strategy to LVL2 using information from the calorimeters and the Inner Detector. For track reconstruction, results obtained with xKalman++ are presented below; work is in progress to evaluate the alternative track-reconstruction program iPatRec. First results show very similar behaviour between these two algorithms. The main differences with respect to LVL2 arise from the availability of calibration data and the possibility to use more sophisticated reconstruction algorithms with access to the detector data for the full event. This results in sharper thresholds and better background rejection. In order to avoid biases introduced by using different reconstruction algorithms for online and offline selection, the EF will select events using as far as possible the offline reconstruction algorithms. However, using the offline reconstruction in the EF implies that the offline algorithms must comply with the stricter EF requirements in terms of robustness and system performance. This has not yet been achieved, but work is in progress to change the algorithms accordingly.

The present study uses the currently-available ATLAS offline reconstruction software (see Section 13.3.4) as a prototype of the future EF code. The criteria used to identify electrons and photons need to be softer in the EF than in the offline in order not to lose events prematurely. In previous EF studies [13-50], [13-51], the electron and photon selections used the same cuts as in the offline selection, but did not use some “critical” criteria. For example a track veto for non-converted photons is not applied at the EF level because it requires good control of the fake tracks in the Inner Detector and thus a very good understanding of the tracking performance, especially in the presence of pile-up. A more realistic EF electron selection has been used for the studies presented here.

In the following, the physics performance of the selection of electrons by the HLT is reviewed. Only the main performance issues are discussed — a more complete report on this work can be found in Ref. [13-52]. The results were obtained using the new framework as described in Section 9.5. Work is in progress to assess more fully the physics performance, including the photon selection which will be reported on at a later stage. The system-performance aspects of the electron selection are discussed briefly in Section 13.4.1.2. Note that the electron and photon selections presented here build on earlier studies [13-50],[13-51],[13-53].

13.4.1.1 HLT Electron Selection Performance

The performance of the electron and photon triggers has been estimated for single electrons and photons, and for some standard physics channels (e.g. $Z \rightarrow ee$, $W \rightarrow e\nu$, $H \rightarrow 4e$). The physics performance is characterized in terms of efficiency for the signal channel, and the rate expected for the selection. Here only the single-electron trigger is considered; for the two-object triggers sufficiently high-statistics datasets could not be reconstructed and analysed in time for the TDR. The efficiencies were obtained using fully-simulated events with single electrons, and the rates were estimated with $\sim 2 \times 10^6$ fully-simulated di-jet events; pile-up was included corresponding to low and design luminosity (see Ref. [13-2]). A higher-statistics sample of 10^7 di-jets is available, but this still needs to be analysed.

The events were generated using PYTHIA 6.203 [13-54][13-55]. It should be noted that compared to older studies the overall cross section for QCD di-jets with E_T>17GeV has increased by 50% [13-56]. However, a comparison made using fast simulation [13-56] has shown that, at the same time, the rate of isolated electrons has gone down by 50% due to a decrease in the rate of electrons from b and c-decays.

Compared to previous studies a more up-to-date detector geometry has been used. An important change is the amount of material in the Inner Detector that has increased significantly, in

particular due to the support of the first b-layer. Therefore, bremsstrahlung effects for electrons have become more important, especially in the end-caps. In addition, a realistic magnetic field map is now used in the simulation instead of a constant solenoidal field of 2T. This change affects the tracking performance in the end-caps. More details on the datasets that have been produced can be found in Ref. [13-2].

In general, events with electrons and photons are selected on the basis of single high-p_T objects or of pairs of somewhat lower-p_T objects. The physics performance of the single-electron triggers is summarized below and documented in more detail in Ref. [13-10] and Ref. [13-52].

The currently-achieved performance for the single isolated-electron HLT algorithms is summarized in Table 13-1 as a function of the main steps in the LVL2–EF trigger chain for the 2×10^{33} cm^{-2} s^{-1} luminosity scenario. The trigger steps have been factorized by detector in order to show the rejection that each stage contributes to the trigger. It should be noted that there are strong correlations between the different selection criteria.

Table 13-1 Performance of the single-electron HLT at 2×10^{33} cm^{-2} s^{-1}, obtained in a preliminary study with the new HLT software chain. The results are presented in a single sequence. 'Matching' refers to position and energy–momentum matching between calorimeter clusters and reconstructed tracks (at LVL2 only precision-detector tracks are used in the present study). The efficiencies are given for single electrons of p_T = 25 GeV already selected by LVL1, averaged over the full pseudorapidity range $|\eta| < 2.5$. For reference, the efficiency of the LVL1 selection is 95%. The rates are normalized to a LVL1 rate for e.m. clusters of ~12 kHz. Note that the quoted errors are statistical — as discussed in the text, there are very large systematic uncertainties on the rate estimates, e.g. from cross-section uncertainties.

Trigger Step	Rate [Hz]	Efficiency [%]
LVL2 Calo	2114 ± 48	95.9 ± 0.3
LVL2 Tracking	529 ± 24	88.0 ± 0.5
LVL2 Matching	137 ± 12	86.6 ± 0.6
EF Global	30 ± 5	76.2 ± 0.7

The overall reduction in rate achieved by LVL2 is ~90 for a loss of efficiency of 14% with respect to LVL1. The additional rate reduction provided by the EF is a factor of about five for a further efficiency loss of ~10%. This shows the power of the two-tier HLT selection.

Comparing the new results with previous studies [13-1], the rate reduction at LVL2 was found to be almost identical for the same electron p_T. This is a useful cross-check and a significant achievement given the completely different selection-software architectures. It also gives us confidence that the new software that became available only shortly before the submission of this document, is working correctly. Moreover, comparing this to previous studies [13-3], we find the rate reduction at LVL2 is almost identical: at the EF, the rate reduction is currently 4.6, compared to 5.2 as reported in [13-3], These numbers still need to be understood better, and further optimization and cross-checks are needed to ensure that the selection scheme does not reject too many 'real' electrons prematurely.

As a cross-check of the above results, the electron efficiencies have also been extracted using fully-simulated $W \rightarrow e\nu$ events. With respect to LVL1, efficiencies of $(90.6 \pm 0.8)\%$ after LVL2 and $(83.2 \pm 1.0)\%$ after EF have been obtained. The simulation included pile-up for a luminosity of 2×10^{33} cm^{-2} s^{-1} and used W events in which the electron had p_T>25 GeV at the generator level. These efficiencies are slightly higher than the ones given in Table 13-1 for single electrons of p_T=25 GeV. This is expected since the efficiency is better for electrons with p_T well above the

threshold value than for those at the threshold value. Preliminary studies of the full offline electron / jet separation analysis, that uses even more powerful selections than the ones we can exploit at the EF, also confirm the rates and efficiencies found here.

The analysis of the performance of the single-electron trigger e30i at high luminosity is in progress and the results are still very preliminary. First results using only the EF selection give a rate of 165 Hz for an efficiency of ~72% with respect to LVL1. Further work is needed to optimize the signal efficiency and rate reduction, including also the LVL2 trigger.

Based on our previous experience, a reasonable target at the EF is to accept electrons with an overall efficiency of about 80% w.r.t. LVL1 in order not to cut too hard on physics. For the rate obtained in the previous studies (40 Hz), the electron efficiency with the new software would be ~78% w.r.t LVL1, as can be seen in the table extrapolating the results given after the EF calorimeter selection. From this we conclude that, with an improved tuning of the different selection steps, a final rate of ~40 Hz at 2×10^{33} cm^{-2} s^{-1} luminosity for an 80% electron efficiency, as obtained in previous studies [13-57], is still valid using the new selection algorithms and updated detector geometry, and allowing for the changes at generator level. The results are preliminary and work is in progress to optimise and cross-check the current selection cuts. For example, the cuts will be tuned in a more optimised way as a function of pseudorapidity, and additional criteria such as isolation in the Inner Detector around the electron track are being studied.

13.4.1.2 HLT Strategy, Algorithm Optimisation and the LVL2–EF Boundary

A first assessment of the system performance has been made for LVL2 in view of its highly demanding environment in terms of execution time and latencies for data transfers. The algorithm execution times per RoI measured on a 2 GHz PC are ~2 ms for the calorimeter reconstruction using T2Calo and ~ 3 ms for the track reconstruction using IDscan — note that the latter is only called for RoIs retained after the calorimeter-cluster selection. The data-preparation code for the calorimeters is currently being optimized to minimize the computing time per RoI. Timings of less than 10 ms per RoI have been achieved on a 2 GHz PC, and efforts will continue to reduce this further.

It is worth noting that the evaluation mentioned above of the data-preparation time was only possible thanks to the realistic simulation of the raw data that the HLT will receive from the Readout System. It appears that the data-preparation time could be a very significant part of the total LVL2 processing time, even though it is restricted to the RoIs.

For the EF, we do not yet have results at the same level of detail as for LVL2, although it is planned to make measurements for both data preparation and algorithmic processing. As an example of the preliminary results obtained so far, data preparation in the EF for the whole LAr calorimeter takes about 0.5 s on a 2 GHz PC. Results from the studies that are under way will be documented in future ATLAS notes.

The use of system resources in the electron HLT can be minimized by exploiting the modularity of the trigger. By ordering the trigger steps so that events are rejected as early as possible, both the overall processing times and data transfer rates may be reduced. Factorizing the trigger-algorithm components also provides flexibility to move the rejection power from LVL2 to the EF or vice versa, to optimize efficiency of the physics selection, rejection of high-rate backgrounds and use of system resources. These issues have been extensively studied in the past and are reported in Ref. [13-53].

If the rate for a trigger item from any level in the selection is too high, one can reduce it either by raising the E_T threshold of the item or by tightening the selection criteria. However, this results in a loss in efficiency for physics signals. The loss in physics may partly be recovered by adding more exclusive trigger selections for the channels of interest, but the contribution to the overall rate of these extra items must be taken into account. There are long-term ongoing studies, performed together with the ATLAS physics community, to assess the impact of such changes in order to prepare alternative scenarios.

As an example, Figure 13-6 illustrates the impact on the selection of $W \rightarrow e\nu$ events of raising the threshold in the single-electron HLT selection at 2×10^{33} cm^{-2} s^{-1} (nominal threshold of 25 GeV). The impact on other physics signal such as $Z \rightarrow ee$, and $H \rightarrow 4e$ is discussed in Ref. [13-52].

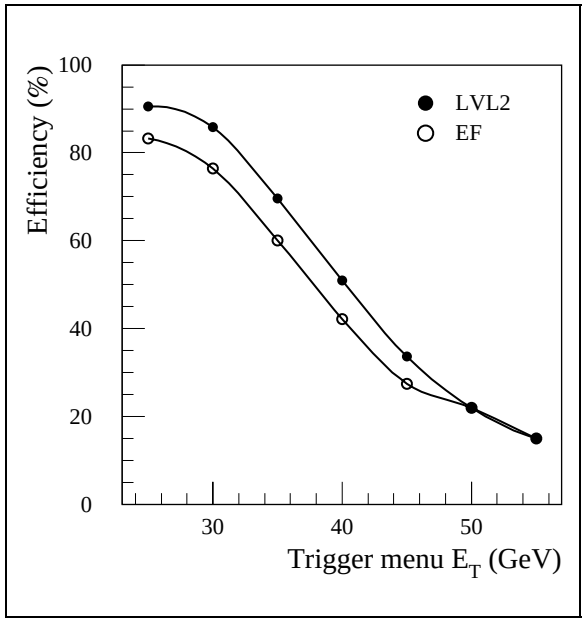

Figure 13-6 Efficiency to select $W \rightarrow e\nu$ events at LVL2 and in the EF as a function of the E_T threshold in the trigger menu at 2×10^{33} cm^{-2} s^{-1} luminosity. Only events that passed the LVL1 selection and for which the electron has p_T>25 GeV at the generator level have been considered.

As illustrated above, the HLT strategy contains considerable flexibility. Various possibilities exist to adjust the balance between minimizing the required computing resources and maximizing the physics performance. For many channels of interest, the selection scheme also provides considerable redundancy. More details on the trigger selection strategy can be found in [13-1] and in Chapter 4.

13.4.2 Muon selection

The main purpose of the high-level muon trigger is the accurate reconstruction of muon tracks within RoIs indicated by the LVL1 muon trigger. LVL2 and the EF must reject low-p_T muons (i.e. muons with p_T below the threshold that are initially selected due to the limited resolution in the first trigger level), secondary muons produced in decays in flight of charged pions and kaons, and fake muon tracks composed of hits from the cavern background. The EF must be able to reconstruct additional muons present in the event that were not reconstructed or selected by the LVL1 and LVL2 triggers.

Whilst the LVL1 trigger system uses only hits from the dedicated trigger detectors (RPCs in the barrel and TGCs in the endcap), LVL2 and the EF have access to the full measurements of the Muon Spectrometer, in particular the data from the Monitored Drift Tubes (MDTs). This allows very good track reconstruction in the muon spectrometer. The high-background environment in the muon spectrometer demands algorithms with robust and fast pattern recognition capable of rejecting hits induced by the cavern background.

The tracks found in the LVL2 muon trigger are extrapolated for combination with measurements in the Inner Detector and the calorimeters. Matching between muon tracks measured independently in the muon system and the Inner Detector selects prompt muons and rejects many fake and secondary muons. This is especially important for the B-physics trigger in low-

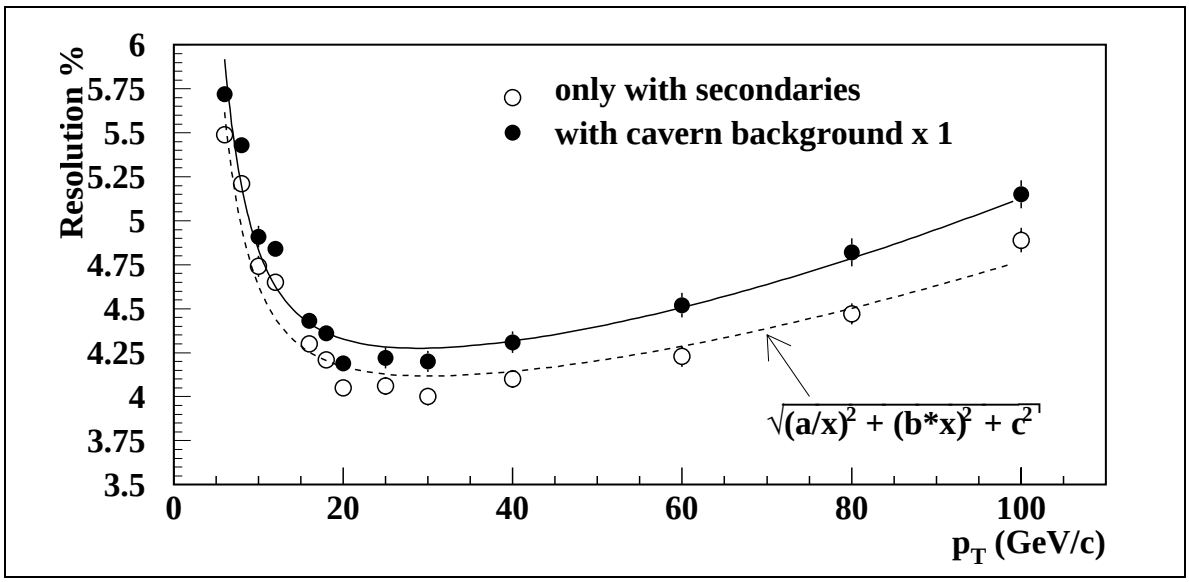

Figure 13-7 The p_T resolution of the muFast algorithm as a function of muon p_T.

luminosity running, for which the selection of relatively low-p_T prompt muons events is the first stage in B-physics trigger selections. The studies presented in this section are limited to the barrel region ($|\eta| < 1$) — future work will extend them for the full pseudorapidity range.

13.4.2.1 The Physics Performance of LVL2 Muon algorithms

The physics performance of the LVL2 muon trigger algorithms was presented and discussed in detail in [13-1]. The algorithm muFast has been implemented in the new framework with no important changes in the code used to reconstruct the muon tracks. The main difference of the present version of the code, compared to the one studied previously, is the use of "LVL1 Emulation" to identify among the RPC hits the ones used by the LVL1 trigger to select muon candidates. We expect this to have very little impact on the overall muon-reconstruction efficiencies at LVL2, and no effect at all on the muon p_T resolution. The following summarizes the most relevant results obtained in the previous studies [13-13].

Table 13-2 Total output rates [kHz] of the stand-alone LVL2 muon trigger after application of the muFast algorithm for the 6 GeV low-p_T threshold at 1×10^{33} cm^{-2} s^{-1} and 20 GeV threshold at the design luminosity.

Physics Process	low-p_T	high-p_T
π/K decays	3.00	0.07
b decays	0.90	0.09
c decays	0.50	0.04
$W \rightarrow \mu\nu$	0.003	0.02
cavern background	negligible	negligible
Total	**4.40**	**0.22**

Given the steeply-falling muon-p_T spectrum, the rate of muons measured with transverse momenta above a given threshold depends strongly on the p_T resolution of the reconstructed muons that is therefore a critical parameter in the LVL2 trigger. The resolution of the muFast algorithm is shown as a function of p_T in Figure 13-7. The resolution ranges between 4.0% and 5.5% for muons in the p_T interval 6–20 GeV. These results are comparable with the p_T resolution obtained by the offline muon reconstruction program MUONBOX [13-58].

The total rates after this algorithm for 6 GeV and 20 GeV thresholds have been evaluated by convolving the efficiency as a function of p_T with the muon differential cross-sections of the dominant muon-production processes. Account was taken of the rejection provided by the LVL1 selection. The rates after LVL2 are shown in Table 13-2[1].

The rates from π/K decays were calculated using the predicted cross-sections from the DMPJET program; the rates would be lower by about 50% if the PYTHIA prediction were used instead. Preliminary studies of the trigger rate arising from the cavern background have been made using predictions of the cavern background from the FLUKA package. The probability that a fake LVL1 muon trigger is accepted by the LVL2 is below 10^{-2}. With this upper limit it is safe to neglect the contribution from fake muons.

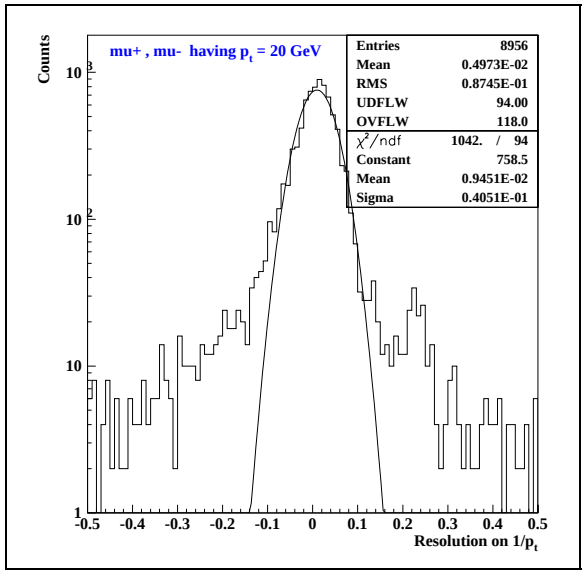

Figure 13-8 Transverse-momentum resolution $(1/p_T\text{-}1/p_T^{true})/(1/p_T^{true})$ for 20 GeV muons reconstructed with the muFast in the new selection framework (no cavern background). Tails arise from events with large-angle Coulomb scattering or from poor momentum reconstruction where delta-rays were emitted by the muon.

The implementation of muFast in the new software framework has been used to reconstruct events with single muons of p_T = 20 GeV. Figure 13-8 shows the distribution of $(1/p_T\text{-}1/p_T^{true})/(1/p_T^{true})$. The resolution is 4.0% which is identical to the result obtained with muFast in the old implementation. This result supports the expectation that the physics performance does not change with respect to the previous results [13-13].The implementation of the muComb algorithm in the new framework is still ongoing. Also in this case the algorithm under implementation is the same one discussed in [13-1]. We can therefore rely on the results already obtained. Figure 13-9 shows as a function of the muon p_T the efficiency for the combined (muon spectrometer plus inner detector) reconstruction of prompt muons and of secondary muons from π/K decays in flight.

The requirement of good matching (i.e. z/ϕ and p_T matching) between the muon-spectrometer and inner-detector tracks reduces contribution from π/K decays in flight to the low-p_T trigger rate to 1.0 kHz: a factor three reduction compared to the rate after the muFast algorithm. Taking into account the reduction in rate from the improved p_T resolution for prompt muons, the total rate after the muComb algorithm is 2.1 kHz for muons with p_T>6 GeV and $|\eta|$<1. Note that these numbers are calculated for a luminosity of 1.0×10^{33} cm^{-2}s^{-1}.

13.4.2.2 The Physics Performance of the Muon Event Filter Algorithm

The physics performance of the MOORE EF package has been evaluated with simulated single-muon samples, with no cavern background, in the p_T range 3–1000 GeV. Here we have considered a *fully-wrapped* version of the offline code that is completely equivalent, from the recon-

1. $W \rightarrow \mu\nu$ cross section found with PYTHIA for p_T^{μ}>3 GeV and $|\eta_{\mu}|$<2.7 : 9.56 pb.This corresponds to 100 Hz at 1×10^{34} cm^{-2} s^{-1}

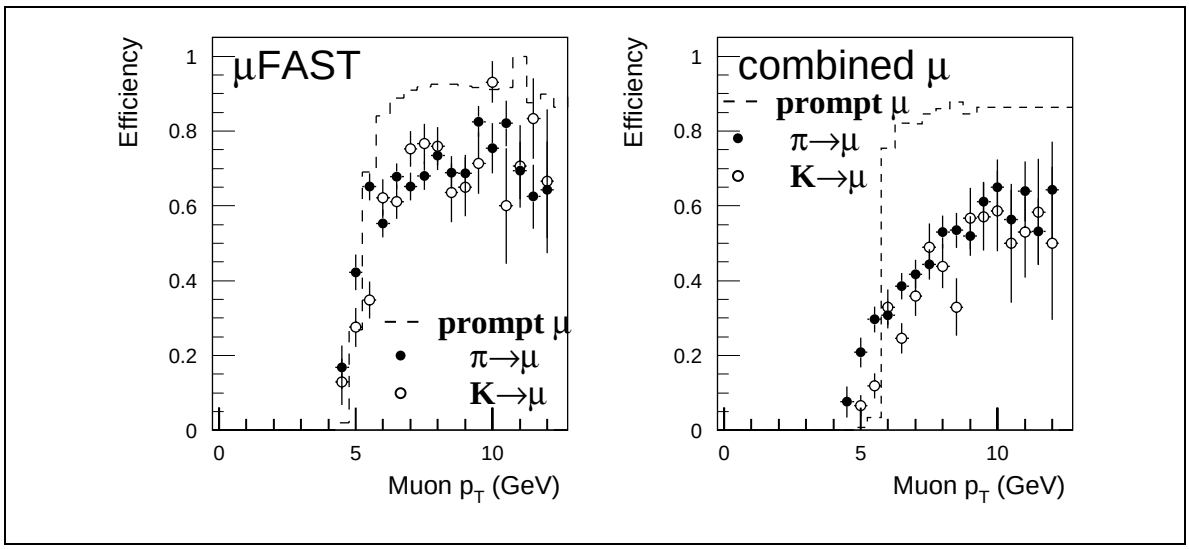

Figure 13-9 The efficiency with respect to LVL1 of the combined (muon spectrometer plus inner detector) reconstruction at LVL2 for prompt muons and for muons from π/K decays in flight. The left-hand plot shows the efficiency of the stand-alone algorithm muFast and the right-hand plot shows the efficiency of the combined muon algorithm muComb.

struction point of view, to the offline version. This version does not support seeding of the track search, e.g. by the results of the LVL2 trigger (see below).

Figure 13-10 shows the track-reconstruction efficiency as a function of muon p_T in the barrel region. The triangle-up symbols show the efficiency of the reconstruction in the muon spectrometer ("Moore"), while the triangle-down symbols show the efficiency of the reconstruction after the track has been extrapolated to the interaction region ("Muid"). The energy loss in the calorimeter has been parameterized as a function of η and muon momentum. The loss of efficiency of "Muid" at low p_T is due to the failure of the extrapolation of muons that exhibit in the spectrometer a transverse momentum of a few GeV. Muons with p_T larger than 8 GeV are reconstructed with efficiencies above 95%, which is equivalent to the results from MUONBOX shown in [13-49], [13-58].

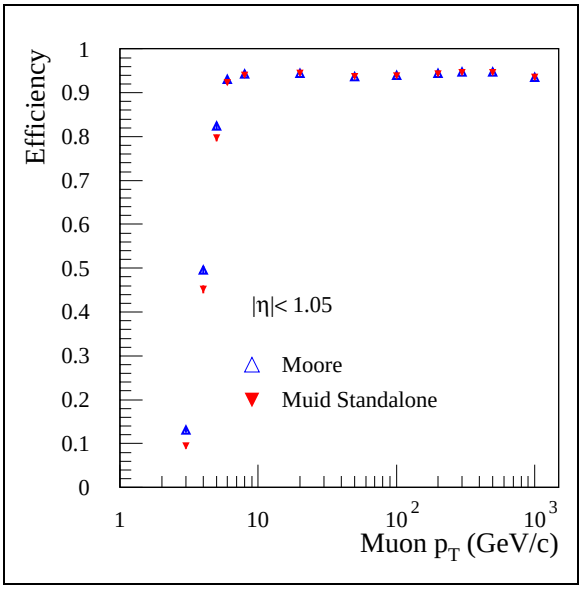

Figure 13-10 Single-muon reconstruction efficiency as a function of muon p_T in the barrel region; triangle-up: stand-alone reconstruction (Moore); triangle-down: reconstruction at the nominal interaction vertex point.

Figure 13-11 shows the muon p_T resolution of the Moore package in the stand-alone muon spectrometer and after extrapolation to the interaction vertex. Again, the results are consistent with earlier studies. The physics performance has been checked for a *seeded* [13-48] version of the algorithm. We don't expect significant differences with respect the full version, since the reconstruction and fitting methods are the same as the ones used by the offline package. This is supported by Figure 13-12 where the transverse-momentum resolution of the seeded version is

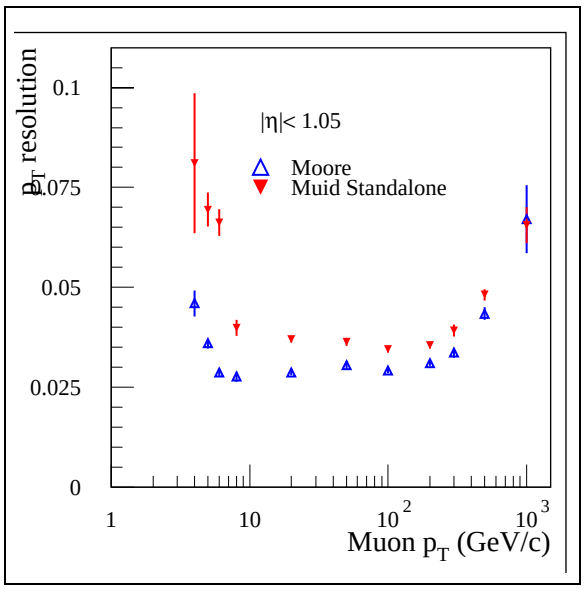

Figure 13-11 *Fully-wrapped* version: transverse-momentum resolution as a function of the generated muon p_T.

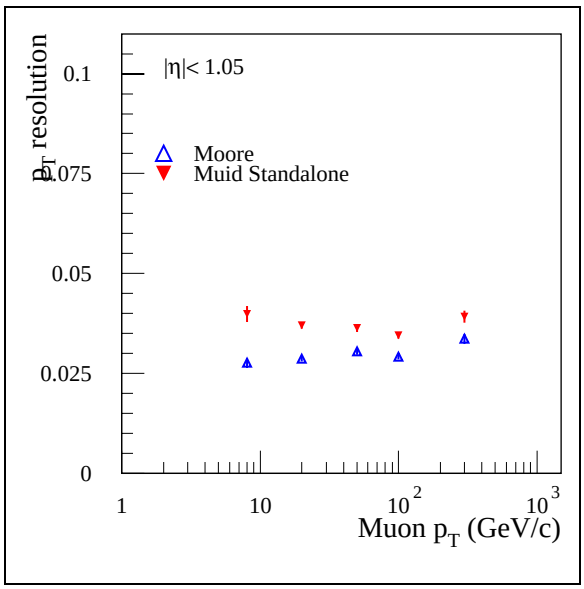

Figure 13-12 *Seeded* version: transverse momentum resolution as a function of the generated p_T.

shown for p_T above 8 GeV. A more complete presentation of the Moore physics performance is given in Refs. [13-47] and [13-48].

13.4.2.3 The Timing Performance of the Muon Algorithms

The muFast trigger algorithm has been benchmarked on a 2 GHz machine. The event sample consisted of about 430 events with single muons in the barrel region of p_T=100 GeV, with and without simulation of the cavern background. As shown in Figure 13-13 the average processing time of muFast is about 1.0 ms per RoI with an r.m.s. of 0.2 ms, and with no event taking more than 4.1 ms.

Figure 13-14 shows the strong correlation of the processing time with the amount of RPC activity. The composition of this overall timing is shown in Figure 13-15 — as can be seen, the processing time is dominated by the "Level-1 Emulation" that contributes about 0.8 ms.

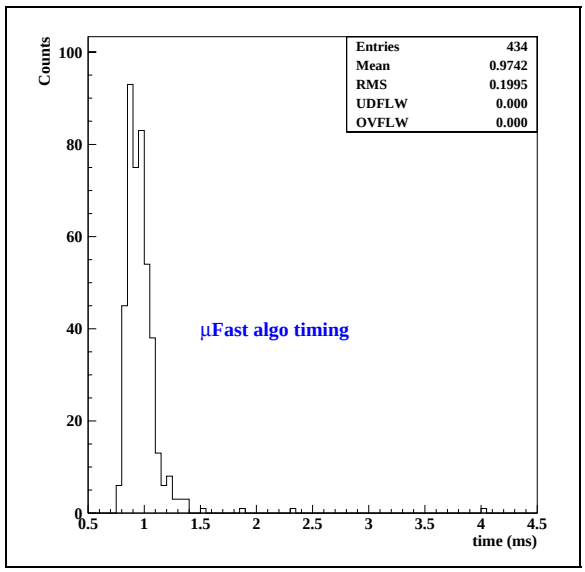

Figure 13-13 Processing time of muFast.

A very preliminary simulation of the cavern background has been made to check the robustness and the latency of the selection algorithms. This background has been simulated assuming the nominal luminosity 1×10^{34} cm^{-2} s^{-1} and boosting by a factor two the predictions provided by the GCALOR package [13-59]. In presence of the cavern background the muFast timing increases by only about 100 μs. The data-preparation time has been evaluated on the same sample of events. Without cavern background the total time is 3.4 ms; this increases to 8.9 ms if the cavern background is added. The time has been found to depend linearly on the

muon-hit multiplicity. This allows one to extrapolate the data-preparation time to more severe background conditions. Considering a safety factor of five, the data-preparation time is expected to increase to 22.5 ms, while the muFast algorithm processing time remains below 1.5 ms. Therefore, on a 2 GHz processor, the overall CPU time is expected to be below 25 ms.

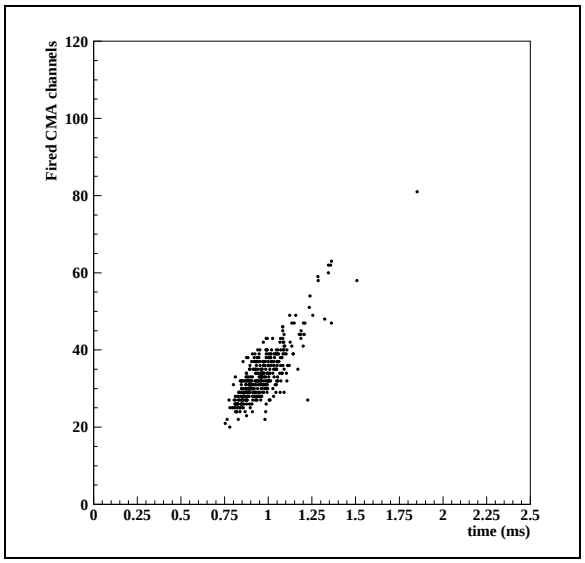

Figure 13-14 Correlation between the muFast processing time and the number of the fired channels in the RPC readout.

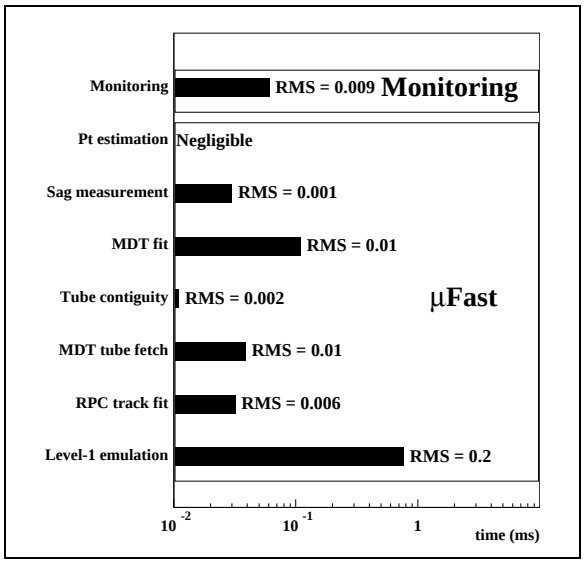

Figure 13-15 Contributions to the muFast processing time. The bar length shows the average processing time; the RMS is given in ms.

The timing performance of the Moore algorithm, for both seeded and fully-wrapped modes, has also been evaluated on a 2 GHz machine. The code was built in optimized mode. The event samples consisted of around 500 events each, fully simulated. Table 13-3 shows the average execution times per event both for seeded and wrapped modes. The timing includes the procedure of extrapolation to the vertex of the reconstructed track. In these results a conservative approach has been adopted where the timings include also the accesses to the data, the data preparation, and the region-selector accesses.

Table 13-3 Summary of Moore timing tests for single-muon events, without and with cavern background at 1×10^{34} cm^{-2} s^{-1}: (a) nominal background intensity, (b) twice the nominal background. intensity

Sample	Time (seeded mode) ms		Time (wrapped mode barrel) ms	
	< >	RMS	< >	RMS
8 GeV	86	69	80	63
20 GeV	65	36	66	43
50 GeV	68	68	68	51
100 GeV	71	60	76	69
300	75	60	81	88
100 GeV (a)	775	70	2800	600
100 GeV (b)	1500	600	3600	2800

The general behaviour of the average execution time for the different samples is represented in Figure 13-16 for both seeded and wrapped mode. In order to show the impact of the extrapolation to the vertex we have plotted the execution times for the track reconstruction inside the MuonSpectrometer (Moore), the execution times for extrapolating the track to the vertex (MuID), and the sum of the two (total). The execution times are rather flat over the analysed p_T range.The time for the whole reconstruction of single muon, including data access and data preparation, is on average below 100 ms. For those events we do not expect a large difference running in wrapped and seeded mode, since the data set that is accessed is quite similar.

The points in Figure 13-16 were obtained by averaging only on events for each sample that register execution times below one second. In order to show the impact of events that register longer execution times, we define a time efficiency as the ratio between the number of reconstructed tracks in one second and the number of Regions of Interest. The plot of time efficiency is shown in Figure 13-17 for both seeded and wrapped mode.

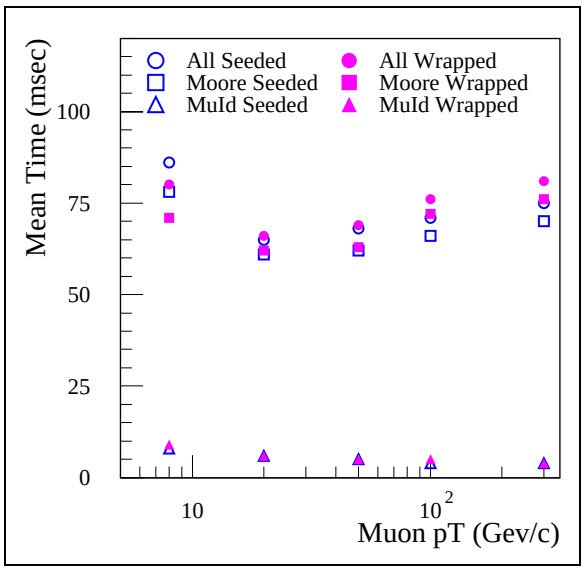

Figure 13-16 Average execution time (in ms) for different p_T values (in GeV), obtained with Moore, with MuId stand-alone and with both of them, in seeded and wrapped mode.

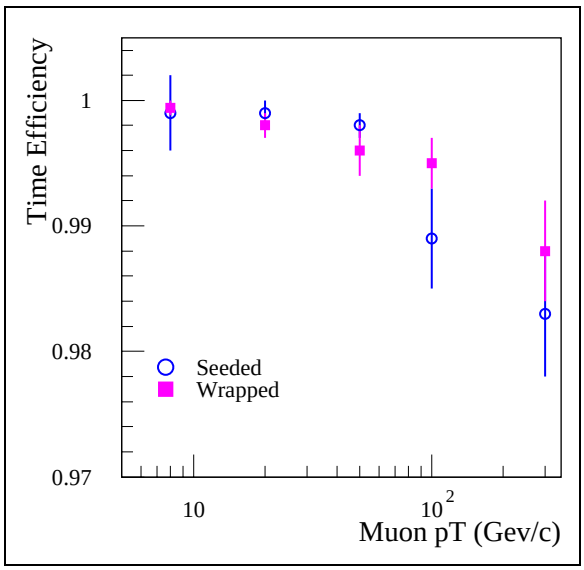

Figure 13-17 Time efficiency for seeded and wrapped mode for different p_T values.

13.4.3 Tau/Jets/$E_T$miss selection

13.4.3.1 The Tau Trigger

A major Standard Model source of tau leptons in ATLAS will be $W \rightarrow \tau\nu$ and $Z \rightarrow \tau\tau$. The tau lepton will also play a key role in the search for new physics. In the MSSM the heavy-scalar (H) and pseudo-scalar (A) Higgs-boson decays to tau-pairs are strongly enhanced with respect to Standard Model Higgs boson case. Also, a key decay channel for the charged Higgs boson is $H^{\pm} \rightarrow \tau\nu$.

The identification of the hadronic decays of tau leptons is based on the selection of narrow isolated jets with low multiplicity in the tracking system. The shower isolation and shape are calculated for both the e.m. and hadronic calorimeters separately. The fraction of energy deposited

by the tau-jet in the e.m. calorimeter has a mean value around 60%. The hadronic shower is broader in the hadronic calorimeter than in the e.m. calorimeter. Thus the jet information obtained from the e.m. calorimeter is more selective than that from the hadronic calorimeter. A detailed description of the tau trigger studies presented below is given elsewhere [13-60].

13.4.3.1.1 The LVL1 Tau Trigger

The motivation for a LVL1 tau calorimeter trigger is manifold, both in conjunction with electron, muon or missing-E_T signatures to enhance Z, A or W coverage, and for calibration purposes. Narrow tau jets containing one (three) charged particles give rise to narrow isolated energy depositions in the calorimeters. It is envisaged that an isolation requirement will be a valuable part of the tau trigger at all levels.

The e/γ and τ/h LVL1 algorithms are described in detail elsewhere [13-1],[13-3]. The LVL1 τ/h calorimeter trigger is based on a 4×4 array of "trigger towers" in the electromagnetic and hadronic calorimeters (within the region $|\eta| < 2.5$) where the tower granularity is (0.1×0.1) $\Delta\eta \times \Delta\phi$. A core E_T is defined in the trigger algorithm as the sum of the electromagnetic and hadronic E_T in a 2×2 trigger-tower region. The trigger algorithm is based on four elements — the trigger cluster(s), an e.m. isolation region, a hadronic isolation region and an "RoI cluster".

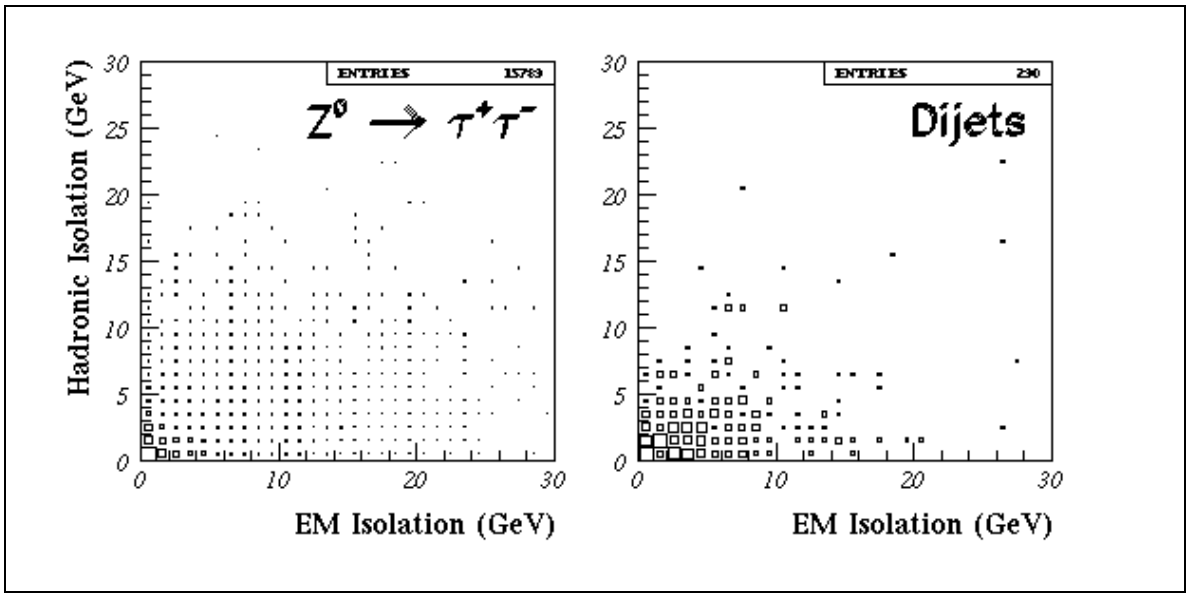

Figure 13-18 The hadronic isolation E_T vs. the e.m. isolation E_T (rings of 12 towers) for tau and QCD di-jets using the LVL1 trigger algorithm.

The distributions of the two isolation variables of the LVL1 τ/h trigger algorithm (in which the cluster E_T is measured in a region of 2×1 e.m. + 2×2 hadronic towers) are shown in the scatter plots of Figure 13-18 for simulated $Z \rightarrow \tau^+\tau^-$ events and QCD di-jet events. Looking at the corresponding projections, it is found that the EM isolation is more powerful that the hadronic one. In both plots a LVL1 algorithm has been employed with a core-E_T threshold of 20 GeV. The correlation between the isolation E_T values indicates that the hadronic isolation is of limited use in rejecting QCD jets. For example, the stand-alone tau trigger rate from QCD di-jets, at a luminosity of 1×10^{33} cm^{-2} s^{-1} and neglecting the effect of pile-up, for a core-E_T of 20 GeV, a jet threshold of 20 GeV and an isolation cut at 10 GeV would be about 19 kHz.

13.4.3.1.2 The High-Level Tau Trigger

The tau trigger utilizes the e / gamma slice tools described in Section 13.4.1. The signal selection is tuned using events of the type $Z \rightarrow \tau^{+}\tau^{-}$. Background evaluation is performed using fully-simulated di-jet events. The LVL2 studies involve the verification of the LVL1 decision and, subsequently, tau identification using parameters that describe the shower shape in layers one and two of the e.m. calorimeters. The LVL2 variables used in a previous analysis [13-61], [13-62] could not be used in the present due to changes in the e / gamma software implementation. Additional rejection of background jets can be achieved by using the information from tracks associated to the tau RoI.

The LVL2 algorithm is applied to LVL1 tau RoIs. Loose LVL1 cuts are chosen for the study presented here: a cluster E_{T} in excess of 20 GeV is required, and the e.m. and hadronic isolation thresholds are set to 10 GeV. LVL2 jet calibration is applied to the cells within the LVL1 RoI window. The energy-weighted position ($\Delta\eta_{\tau} \times \Delta\phi_{\tau}$) of the tau-jet candidate is computed from all calorimeter cells within the LVL1 window. The first part of the LVL2 algorithm is the confirmation of the LVL1 decision. In order to do this the LVL1 algorithm described above is repeated using fine-grained cell information instead of the coarse-grain information available at LVL1.

As a first step, a check was made to ensure that LVL1 RoI coordinates are good approximations of the LVL2 tau coordinates, by measuring the distance between LVL1 RoI and the associated LVL2 cluster. After this successful check, for the LVL2 clusters associated with LVL1 RoIs, the next step is to look at the LVL2 e / gamma calorimetric variables that have some power to select taus over QCD jets.

Three variables were identified to discriminate between τ-jets and the di-jet background. The performance of the algorithm as a function of the shower-shape variable R_{37} was examined first. This variable is defined as the ratio of the E_{T} contained in a 3×7-cell cluster to the E_{T} in a 7×7-cell cluster centred on the same seed cell, calculated for the second layer of the e.m. calorimeter (Section 13.3.3.5). The second variable studied, $F_{12,}$ is defined in the first layer of the e.m. calorimeter and functions as an isolation variable: $F_{12} = (E_{max1} - E_{max2})/(E_{max1} + E_{max2})$ where E_{max1} and E_{max2} are the highest and second highest strip energies, respectively, in the first layer of the e.m. calorimeter: this quantity is defined for each LVL2 cluster. Is should be noted that the variables R_{37} and F_{12} are highly correlated. The third variable was, $F_{\mathrm{e.m.}}$, the e.m fraction of the total energy.

Table 13-4 Trigger efficiencies for taus and for QCD jets, for different R_{37} cuts and the LVL1 trigger condition defined in the text. The right-hand column shows the corresponding stand-alone tau trigger rate based on QCD di-jets with a Pythia generation threshold E_{T} of 35 GeV.

Cut	Tau Efficiency % (wrt LVL1 accepts)	QCD jet efficiency% (wrt LVL1 accepts)	LVL1/2 tau trigger rate using calorimetry (kHz)
$R_{37} > 0.75$	88 ± 1	71 ± 1	11.1 ± 0.2
$R_{37} > 0.80$	83 ± 1	59 ± 1	9.4 ± 0.2
$R_{37} > 0.85$	75 ± 1	44 ± 1	7.0 ± 0.2
$R_{37} > 0.90$	62 ± 1	26 ± 1	2.3 ± 0.2

The variable with the most power to select taus and reject QCD jets was found to be R_{37}. Since the F_{12} variable is highly correlated with R_{37}, it was not found possible to obtain useful further

improvements in tau efficiency and jet rejection using F_{12}. Likewise, it was not possible to employ $F_{\text{e.m.}}$ to gain further QCD jet rejection whilst maintaining a good tau efficiency.

Three possible cuts in the R_{37} variable were studied: $R_{37} > 0.75, 0.80, 0.85$. These cuts were chosen to maintain a tau efficiency greater than 75%. Table 13-4 shows the efficiency for triggering on $Z \rightarrow \tau^+\tau^-$ and QCD di-jets, along with the corresponding stand-alone tau trigger rate at LVL2 estimated using QCD di-jets, for the three R_{37} cuts mentioned. The values are also given for reference for $R_{37} > 0.90$, although this gives an unacceptably low efficiency for selecting taus. The chosen set of LVL1 conditions, denoted 20-10-10, are the loose LVL1 cuts defined for the LVL1 study: in this case the core E_T is 20 GeV and the e.m. and hadronic isolation thresholds are both 10 GeV.

Additional rejection of background QCD jets can be achieved by using the information from tracks associated with the tau LVL1 RoI. The track information available at LVL2 was used to associate Inner Detector tracks, found using the "IDscan" algorithm, with the tau RoI by requiring that ΔR between the track and the LVL2 tau cluster direction, associated with the tau LVL1 RoI, obeyed the relation, $\Delta R < 0.3$. The inner-detector track-multiplicity distributions obtained for $Z \rightarrow \tau^+\tau^-$ and QCD di-jets are shown in Figure 13-19. The resulting LVL2 tau and QCD di-jet efficiencies are shown in Table 13-5 for a few useful combinations of calorimeter and track-based cuts.

Table 13-5 The tau and QCD-jet efficiencies for various useful combinations of R_{37} and LVL2 track-multiplicity cuts, for the specified LVL1 trigger condition. The right-hand column shows the corresponding stand-alone tau trigger rate based on QCD di-jets with a Pythia generation threshold E_T of 35 GeV.

Cuts	Tau efficiency % (wrt LVL1 accepts)	QCD di-jet efficiency % (wrt LVL1 accepts)	Stand-alone tau/hadron trigger rate (kHz)
LVL1: 20-10-10			15.6 ± 0.2
R_{37}>0.8, #LVL2 tracks<5	74 ± 1	34 ± 1	5.3 ± 0.2
R_{37}>0.8, #LVL2 tracks<4	70 ± 1	30 ± 1	4.7 ± 0.2
R_{37}>0.9, #LVL2 tracks<5	57 ± 1	18 ± 1	2.8 ± 0.2
R_{37}>0.9, #LVL2 tracks<4	54 ± 1	15 ± 1	2.3 ± 0.2

13.4.3.1.3 Tau Selection in the Event Filter

At the Event Filter stage, access to the complete, calibrated, event is possible for the first time. In addition, the tracking information has been further refined to reduce the number spurious track segments and minimize tracking inefficiency. Thus, it is possible to refine the LVL2 decision. Existing off-line studies of tau / hadron identification and jet rejection [13-63] provide the basis for the EF trigger decision. Typical trigger criteria for tau / hadron jets with $E_T > 30$GeV and $|\eta| < 2.5$ are as follows:

- The jet radius computed using only the e.m. cells contained in the jet, R_{em}, must obey the inequality: $R_{\text{em}} < 0.07$.
- The difference between the E_T contained in cones of sizes $\Delta R = 0.2$ and $\Delta R = 0.1$, normalized to the total jet E_T, ΔE_T must obey the inequality: $\Delta E_T < 0.1$ (isolation fraction).
- The number of reconstructed charge tracks pointing to the cluster (within a ΔR of 0.3), N_{tr}, is equal to one or three.

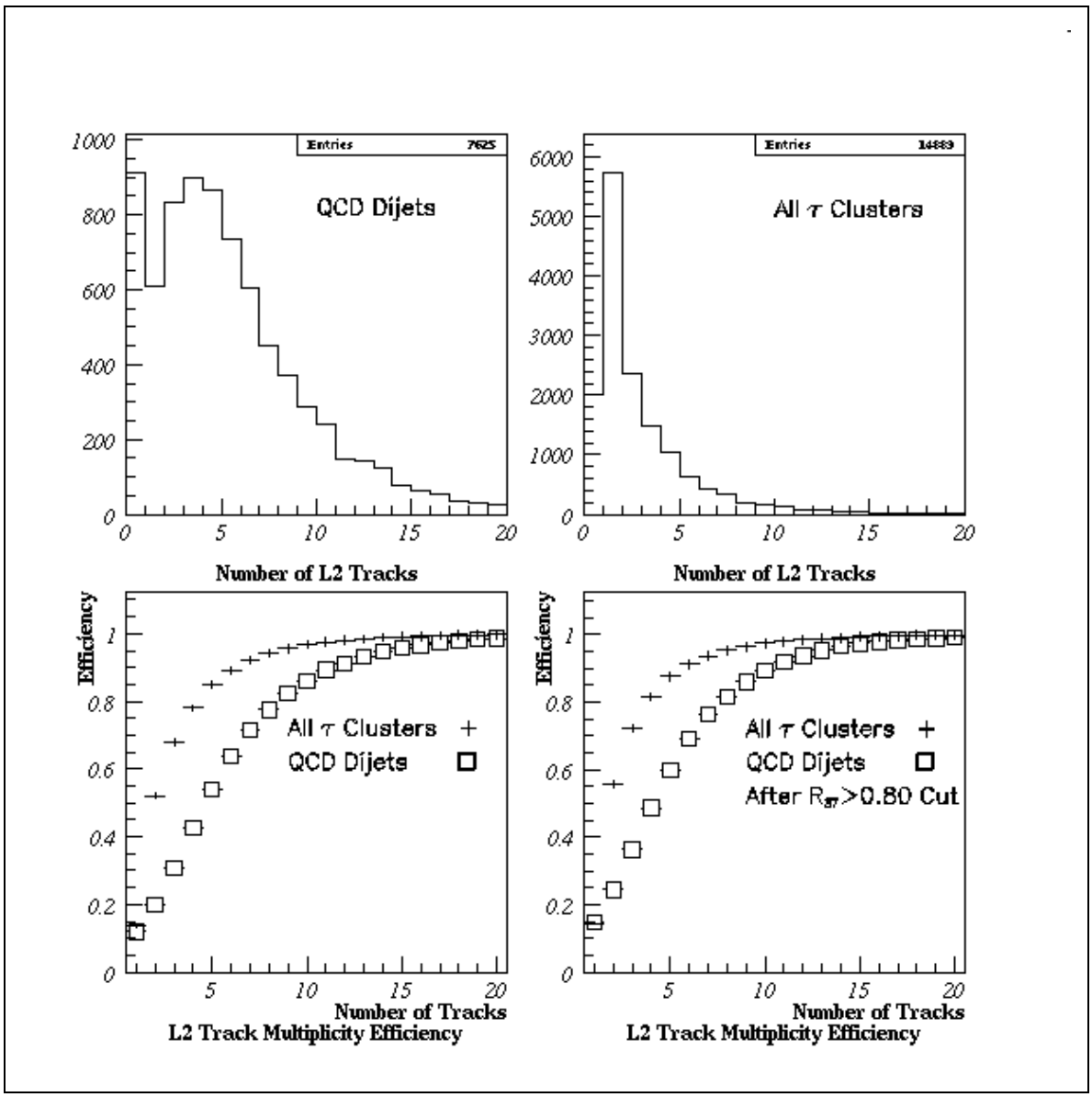

Figure 13-19 The top plots show the inner detector track multiplicities determined at LVL2 for $Z \rightarrow \tau^+\tau^-$ &QCD di-jets The bottom plots show the variation of tau & jet finding efficiency with upper cut on track multiplicity.

Other tau-identification variables that have been considered are the tau hadronic E_T and a tau-likelihood variable included in the "Taurec" reconstruction package in the ATHENA framework. The likelihood function uses simple analytical fits to the distributions of the variables mentioned above, as well as the p_T of the highest p_T track. The variables mentioned above still have power at the EF level to reject QCD jets whilst retaining an adequate efficiency for selecting taus. An analysis of the rejection of QCD jets at the EF level is currently under way [13-60].

13.4.3.1.4 Jet Rejection at LVL2 Following a Tau +E_T-miss trigger

A method of improving the signal acceptance for final states involving taus, as well as retaining an acceptable trigger rate, is to combine the stand-alone tau trigger with an E_T^{miss} trigger. The effect of adding a LVL1 E_T^{miss} requirement on stand-alone tau HLT rates is shown in Table 13-6 for one set of LVL2 trigger criteria and four trigger-threshold configurations: τ20 + xE25; τ20 +

xE35; τ30 + xE25; and, τ30 + xE35. As can be seen from the table the LVL1 E_T^{miss} requirement substantially enhances the jet rejection and thus reduces considerably the tau trigger rate coming after LVL2.

Table 13-6 Tau and QCD di-jet efficiencies for various LVL1 & LVL2 trigger criteria. In the right-hand column the corresponding tau stand-alone LVL2 trigger rates are given including the LVL1 E_T^{miss} requirement.

LVL1 Tau + E_T-miss trigger (GeV)	LVL2 Cuts	LVL2 tau eff. wrt LVL1 (%)	LVL2 jet eff. wrt LVL1 (%)	LVL2 stand-alone tau trigger rate (kHz)
τ(20-10-10) + xE25	-	29 ± 1	10 ± 1	1.56 ± 0.16
τ(20-10-10) + xE35	-	11 ± 1	1.9 ± 0.2	0.30 ± 0.03
τ(20-10-10) + xE25	R_{37}>0.9 #LVL2tracks<5	15 ± 1	1.6 ± 0.2	0.25 ± 0.03
τ(20-10-10) + xE35	R_{37}>0.9 #LVL2tracks<5	5.1 ± 0.2	0.5 ± 0.1	0.07 ± 0.01
τ(30-10-10) + xE25	-	43 ± 1	18 ± 1	0.85 ± 0.05
τ(30-10-10) + xE35	-	20 ± 1	4.3 ± 0.4	0.20 ± 0.02
τ(30-10-10) + xE25	R_{37}>0.9 #LVL2tracks<5	23 ± 1	3.6 ± 0.4	0.17 ± 0.02
τ(30-10-10) + xE35	R_{37}>0.9 #LVL2tracks<5	9.6 ± 0.5	1.2 ± 0.2	0.06 ± 0.02

13.4.3.2 E_T-miss Trigger

Missing transverse energy will provide a distinct and important signature for new physics at the LHC. A precise and reliable measurement of E_T^{miss} requires good calorimeter performance and energy resolution, good linearity of response and hermetic coverage over a wide range of pseudorapidity.

The E_T^{miss} + jet trigger is an example of a trigger based on the combination of a global variable (E_T^{miss}) and localized RoIs in the detector. The bulk of the trigger rate will result from fluctuations in the energy measurements of QCD jets, partly as the result of the presence of material in front of the calorimeters and in the regions between the various calorimetry sections. The main instrumental effects arise from the difference in response between the various calorimeter technologies and from the fact that the calorimetry is highly non-compensating.

The contribution of the EF to reducing the LVL1 E_T^{miss} trigger rate should be important for three main reasons. Firstly, accurate calorimeter calibration and inter-calibration are available [13-49]. Secondly, a separate calibration for cells between clusters can be utilized. Thirdly, the cell E_T cutoff applied to suppress noise can be tuned accurately [13-66]. Initial studies of the E_T^{miss} + jet trigger rate used samples of QCD di-jets (from Pythia) made with both fast and full simulation. Excellent agreement was obtained between these simulation methods. Details of the simulation can be found elsewhere [13-67].

The E_T^{miss} + jet trigger-rate calculation has been revisited using the latest ATHENA object-oriented reconstruction software framework. In addition, the following improvements were included in the analysis: an updated version of the Pythia event generator; an updated model for minimum-bias interactions; a new algorithm for E_T^{miss} calculation utilizing the H1 calibration approach as well as tuned cell cuts. Initially the low-luminosity scenario has been considered without pile-up but with electronic noise added to the simulation. The preliminary results at 2×10^{33} cm^{-2} s^{-1} give an E_T^{miss} + jet trigger rate of 50±12 Hz for (E_T^{jet}, E_T^{miss})>(60, 60) GeV, and

40±12 Hz for (E_T^{jet}, E_T^{miss})>(70, 70) GeV. These rates are consistent with earlier results [13-1] and can be reduced further by applying topological cuts as in the past. A plot of the trigger rate versus the E_T^{miss} threshold is shown in Figure 13-20 for both jet threshold settings.

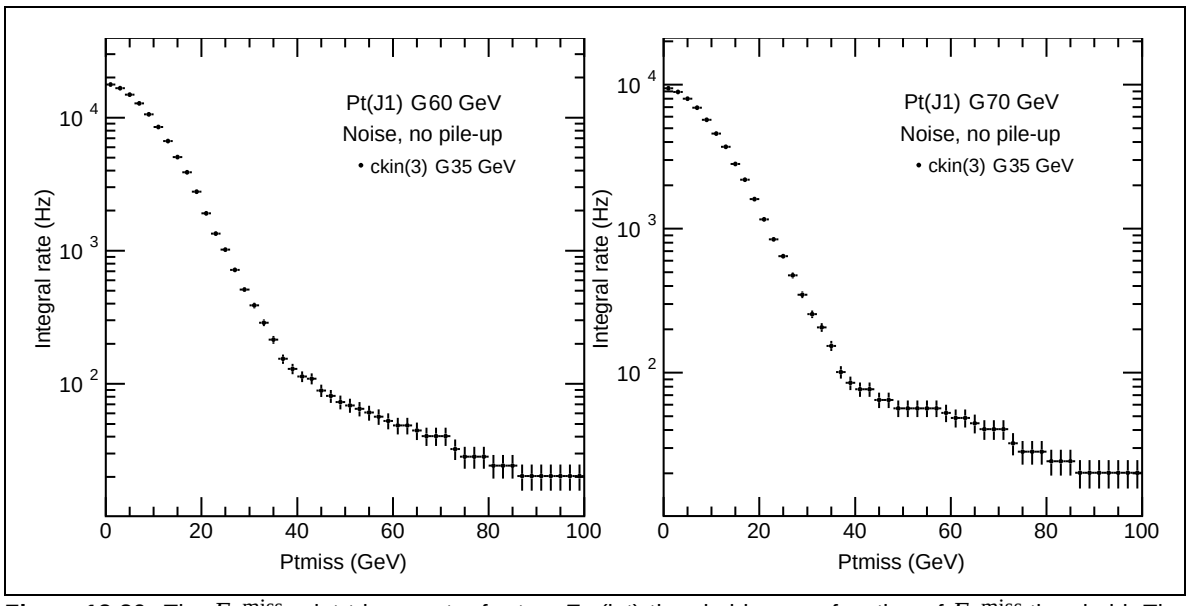

Figure 13-20 The E_T^{miss} + jet trigger rate, for two E_T (jet) thresholds, as a function of E_T^{miss} threshold. The plots are for a luminosity of 2×10^{33} cm^{-2} s^{-1} but without pile-up included. Electronic noise has been added.

Recently estimates of the tau + E_T^{miss} HLT rate have been made using the ATHENA reconstruction software. In this case the HLT rate was estimated by assuming that the existing offline tau and E_T^{miss} code to simulate the performance of the tau + E_T^{miss} HLT. The effects of electronic noise in the calorimetry were included in the simulation. This preliminary analysis of the low-luminosity trigger rates did not include the effects of pileup. The same offline tau-identification criteria bulleted in Section 13.4.3.1.3 were used. Also, a cut of ~2σ in the electronic noise was applied to all calorimeter cells immediately after reconstruction and before any further analysis. In this case an improvement was made to the standard offline analysis by making the tau-identification criteria dependent on the E_T range [13-68]. In this way the cuts could be optimised for the highest rejection for a given efficiency. It was seen that the jet rejection came mainly from the E_T^{miss} cuts and after that from the tau-identification criteria. This preliminary analysis had only limited statistics and thus only a rough estimate of the trigger rate is possible. However, fixing our cuts to give a tau efficiency of 55%, and taking the tau E_T >100 GeV, the estimated tau + E_T^{miss} rate is ~5Hz.

13.4.3.3 Triggering on Jets

The purpose of the high-level jet trigger is to reduce the rate of events containing jets, compared to the LVL1 one, by exploiting the improved E_T measurement. This improvement is achieved by applying a more refined energy calibration and jet definition, and using the fine-granularity and high-precision readout data. As jets are the dominant high-p_T process rate reduction cannot be expected by removing fake jet objects. In contrast to e.m. clusters, jets cannot be flagged as background (unless they are due to instrumental effects). Jets are searched for in the region of $|\eta| < 3.2$ in the e.m. and hadronic calorimeters. Hereafter, preliminary results will be presented using the following threshold values and luminosity scenarios: j180, j75x3, and j55x4 at

1×10^{33} cm^{-2} s^{-1} and j290, j130x3, and j90x4 at 1×10^{34} cm^{-2} s^{-1}. Work is in progress to better adapt the studies to the present start-up scenario.

The LVL2 jet finding is guided by the LVL1 jet regions of interest (RoI) — jets are sought in a region around these RoIs. In the current view the EF performs jet finding across the whole pseudorapidity region with the improved jet calibration and inter-jet calibration allowed at this level. The performance of the high-level jet trigger is described in detail elsewhere [13-69]. The jet rates in this report were obtained using fully-simulated QCD di-jet events.

Table 13-7 The top portion of the table shows HLT rates, obtained using fully-simulated events, for different trigger-menu items [13-69]. The middle portion of the table shows the estimated rates using a fast simulation [13-70]. The bottom portion of the table shows the jet-trigger rates for the two menu values indicated.

item	1×10^{33} cm^{-2} s^{-1} LVL2+EF (Hz)	Item	1×10^{34} cm^{-2} s^{-1} LVL2+EF (Hz)
j180	253 ± 26	j290	275 ± 87
3j75	286 ± 28	3j130	440 ± 110
4j55	127 ± 15	4j90	165 ± 67
j180	228 ± 8	j290	270 ± 33
3j75	314 ± 10	3j130	270 ± 33
4j55	153 ± 7	4j90	149 ± 25
2j180	101 ± 6	2j290	109 ± 21
2j130	377 ± 11	2j230	358 ± 38

The ATLAS fast-simulation program ATLFAST (version 00-02-22) was used to generate 13 million di-jet events that were utilized to provide an estimate of the HLT (LVL2+EF) jet-trigger rates for one, two, three and four-jet final states [13-70]. In this case the jet-finding algorithm required a jet-initiator E_T of 5 GeV, a jet-cone radius R=0.4 and a jet E_T threshold of 10 GeV. The trigger rates were normalized, using a single scaling factor, to the rates predicted, for low and high-luminosity running, in a previous analysis based on fully-simulated QCD di-jets [13-69]. The one, two, three, and four-jet trigger rates are reported in Table 13-7 for the thresholds given in reference [13-69] and obtained using the ATLFAST simulation.

13.4.4 b-tagging

The selection of b-jets at trigger level can improve the flexibility of the HLT scheme and possibly extend its physics performance. In particular, for topologies containing several b-jets, the ability to separate b-jets from light-quark and gluon jets could increase the acceptance for signal events (if the use of jet-E_T thresholds lower than those discussed in Section 13.5 is feasible) or reduce the background (and hence the rate) for events containing b-jets that have already been selected by other triggers.

The study presented in this section defines and characterizes, in the low-luminosity case, a b-jet selection for LVL2 [13-71] based on Inner Detector information. The extension of the selection to the EF and the definition of an overall strategy for the b-tagging will be addressed in future studies.

13.4.4.1 LVL2 track reconstruction for b-tagging selection

The track reconstruction and the precise determination of the track parameters (in particular the impact parameter (d_0) in the transverse plane) are crucial ingredients of the b-jet trigger.

Several tracking algorithms based on the silicon detectors have been presented in Section 13.3. This study is based on SiTrack [13-72], a LVL2 algorithm that selects and builds track candidates using triplets of space points, because of its good impact-parameter resolution. An early version of SiTrack, PixTrig (based solely on the pixel detector), was used to perform a similar study for the Technical Proposal [13-1].

The recent change in the pixel detector geometry (increased B-layer radius) caused a significant degradation in the impact-parameter resolution for tracks built using the pixel detector alone. The impact-parameter resolution has been recovered, and slightly improved at high p_T (see Figure 13-21), using the capability of SiTrack to use any layer of the inner detector — the comparatively large lever arm granted by an SCT layer ensures a better resolution of the track parameters in transverse plane.

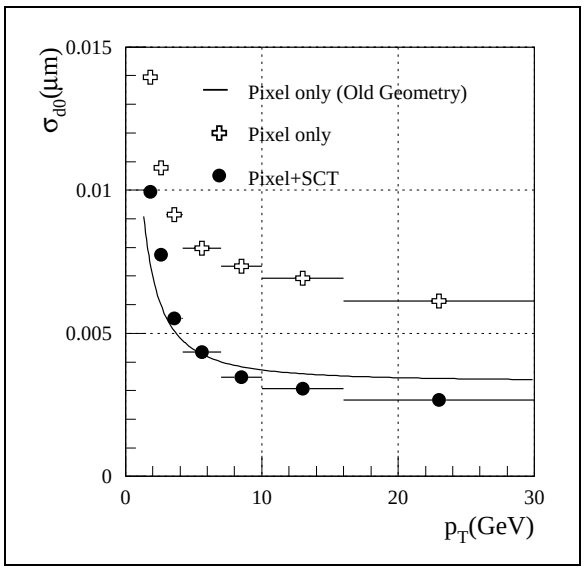

Figure 13-21 Comparison of d_0 resolution as a function of p_T for different configuration of the LVL2 tracking algorithm SiTrack. The label "Old Geometry" refers to the results obtained in Ref. [13-1] with an earlier geometry of the pixel detector (the so-called Physics-TDR layout).

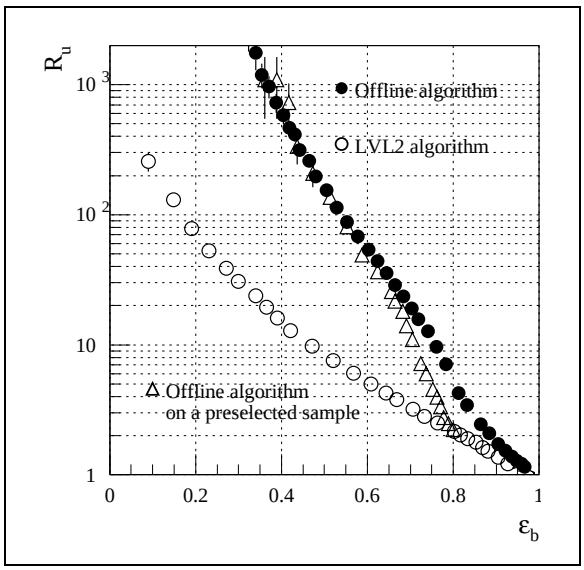

Figure 13-22 he u-jet rejection as a function of the b-jet efficiency for jets from the decay of Higgs bosons with m_H=120 GeV for trigger and offline algorithms. The bias of the LVL2 selection on the offline selection is shown.

13.4.4.2 b-tagging algorithm

The b-tagging selection starts with track reconstruction performed by SiTrack within the LVL1 jet RoI. For each reconstructed track the significance of the transverse impact parameter $S = d_0 / \sigma(d_0)$ is computed; the error on the impact parameter $\sigma(d_0)$ is parameterized, using simulated events, as a function of p_T.

The b-jet discriminator variable is then built using the likelihood-ratio method; for each track (i) in the jet, the ratio of the probability densities for the track to come from a b-jet or a u-jet, $f_b(S_i) / f_u(S_i)$, is calculated; the product $W = \Pi\, f_b(S_i) / f_u(S_i)$ of these ratios over all tracks in the jet is com-

puted; finally the tagging variable $X = W/(1+W)$ is defined. Jets are tagged as b-jets if $X \sim 1$ and u-jets if $X \sim 0$. The selection efficiency for b-jets and the rejection of light-flavour jets can be tuned varying the cut on the X variable.

The processing time of the b-tagging selection algorithm is dominated by the track-reconstruction phase and today is less than 8 ms per jet on a 2 GHz processor for dijet events. Note that the figure of 8 ms does not include the data-preparation time that remains to be assessed in the new HLT software framework.

13.4.4.3 Results on single b-jet tagging

The b-tagging algorithm has been characterized on single b-jets coming from $H \rightarrow b\bar{b}$ decays (m_H = 120 GeV) produced in association with a W boson at low luminosity, and corresponding u-jets (taken as representative of the light-flavour jets) obtained by artificially replacing the b-quarks from the Higgs decay with u-quarks; the LVL1 jet RoI was simulated by selecting a region $\Delta\phi \times \Delta\eta = 0.4 \times 0.4$ centred around the direction of the quarks coming from the decay of the Higgs boson.

The efficiencies for b-jets (ε_b) and rejection factors (R_u) against u-jets (defined as the inverse of the efficiency for u-jets) are given in Table 13-8. The modest rejections obtained do not spoil the

Table 13-8 Rejection of the LVL2 b-tagging algorithm against u-jets for three different values of the b-jet efficiency: 60% (top), 70 % (middle) and 80 % (bottom) at low luminosity. The results are shown for different intervals of E_T and η of the jet.

	40 GeV < E_T < 70 GeV	**70 GeV < E_T < 100 GeV**	**E_T > 100 GeV**
$\|\eta\| < 1.5$	5.9 ± 0.4	5.1 ± 0.6	4.8 ± 0.7
	3.6 ± 0.2	3.9 ± 0.4	3.1 ± 0.4
	2.3 ± 0.1	2.8 ± 0.3	2.4 ± 0.3
$\|\eta\| > 1.5$	3.7 ± 0.4	4.9 ± 1.0	3.0 ± 0.8
	2.6 ± 0.2	2.9 ± 0.5	2.6 ± 0.6
	1.5 ± 0.1	1.6 ± 0.2	1.7 ± 0.4

interest of applying a b-tagging selection at LVL2: in events with multiple b-jets, loose selections (necessary to minimize the bias on the selected sample) applied to several jets can still produce significant rejections

13.4.4.4 Comparison with Offline b-tagging

The performance of the LVL2 b-jet tagging algorithm has been compared to that of an offline algorithm based on impact parameter measurements in the transverse plane. This choice provides a coherent comparison of online and offline algorithms; more exhaustive comparison studies will be performed on specific physics selections in due course.

Figure 13-22 demonstrates that the LVL2 and offline selection are well correlated and that, with an appropriate tuning of the LVL2 selection, it is possible to provide subsequent analyses with an unbiased sample.

Different combinations of working points for the LVL2 trigger selection and offline analysis could be chosen depending on the topology of the events and on the required offline b-tagging efficiency. Additional flexibility can be added to the b-tagging selection by integrating the LVL2 selection with a selection at EF, where tracking with offline quality will be available.

13.4.5 B-physics

About one collision in every hundred will produce a $b\bar{b}$ quark pair. Therefore, in addition to rejecting non-$b\bar{b}$ events, the B-trigger must have the ability to identify and select those events containing B-decay channels of specific interest. Important areas of study include: CP-violation measurements with the channels $B_d \to J/\psi K_s$ (with both $J/\psi \to e^+e^-$ and $J/\psi \to \mu^+\mu^-$) and $B_d \to \pi^+\pi^-$ (and more generally $B_{d,s} \to h^+h^-$, where h denotes a pion or kaon); measurements of B_s oscillations using $B_s \to D_s\pi$ and $B_s \to D_s a_1$ decays with $D_s \to \phi\pi$; analysis of $B_s \to J/\psi\phi$ and $B \to J/\psi\eta$ final states, including the search for CP-violation in $B_s \to J/\psi\phi$ decays where new physics could enhance the effect significantly beyond Standard Model expectations; rare decays of the type $B_{d,s} \to \mu^+\mu^-(X)$; and *B*-hadron production measurements. High statistics are required for the precision measurements. As documented in Ref. [13-49], ATLAS is well placed to make significant and competitive contributions in many of these areas.

Since the HLT/DAQ/DCS Technical Proposal [13-49] the B-trigger has been re-assessed in the light of a number of developments, including the likelihood of a reduced inner-detector layout at the start of running, a factor of two increase in the target start-up luminosity and various trigger scenarios for deferring spending on the HLT/DAQ system. The aim is to retain the maximum possible coverage of key B-physics channels within the available resources.

It is important to study a range of scenarios since the actual LHC machine start-up conditions are uncertain, the luminosity is expected to vary from fill-to-fill, and there are large uncertainties in the physics cross-sections and in the calculation of required resources. A flexible trigger strategy has, therefore, been developed based on a di-muon trigger (with p_T thresholds, which may be η dependent, of about 3–6 GeV) at luminosities of $\sim 2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ or above, and introducing other triggers at lower luminosities. Lower luminosities will occur since some LHC fills will have lower luminosities than others, and the luminosity will fall during each beam-coast (over the period of a beam-coast the luminosity is expected to fall by about a factor of two).

Two strategies have been investigated for these additional triggers, as follows:

- The preferred strategy is to require at least one LVL1 jet or EM RoI in addition to a single-muon trigger (e.g. $p_T > 8$ GeV). After validating the LVL1 muon at LVL2, and in the EF, track reconstruction is performed within the RoIs using pixel, SCT and, optionally, TRT information. The reconstructed tracks form the basis of selections for, e.g., $J/\psi(e^+e^-)$, $B(h^+h^-)$ and $D_s(\phi\pi)$. Since track reconstruction is performed only inside the RoIs, the resources required are modest. More details are given in Ref. [13-73].
- If the LVL1 RoI multiplicity proves too high (or the efficiency too low) for the above approach, a fall-back solution is to perform track reconstruction within the full acceptance of the SCT and pixel detectors (so-called full-scan) for events with a LVL1 single-muon trigger (e.g. $p_T > 8$ GeV), after confirmation of the muon at LVL2. In order to minimize execution time, the TRT is not presently included and so the $J/\psi(e^+e^-)$ trigger is not possible. The reconstructed tracks form the basis of selections for, e.g., $B(h^+h^-)$ and $D_s(\phi\pi)$. This requires somewhat greater resources than the method described above, in order to perform

the full-scan, but promises better efficiency. This strategy is described in detail in Ref. [13-74].

In all cases, at least one LVL1 trigger muon is required to initiate the B-trigger. Since the inclusive cross-section for muon production from decays in flight of pions and kaons falls more rapidly with p_T than that for prompt muon production from B-hadron decays, see Figure 13-23, an appropriate choice of p_T threshold gives a powerful reduction of the trigger rate due to background processes. For example, a p_T threshold of 8 GeV gives a single-muon trigger rate of 10 kHz at LVL1 for a luminosity of 1×10^{33} cm^{-2} s^{-1}. Most of this rate is due to muons with true p_T below threshold, originating from pion and kaon decays in flight, a large proportion of which can be rejected at LVL2 on the basis of more precise measurements in the muon spectrometer and inner detector. After the LVL2 selection the single-muon trigger rate is about 2 kHz at a luminosity of 1×10^{33} cm^{-2} s^{-1}; about one third of this rate is due to b→μ decays. It is important not to set the muon p_T threshold too high as this would significantly reduce the statistics in the signal channels of interest and render the measurements un-competitive. The rate is further reduced at LVL2 and in the EF by requiring other triggers in addition to the muon, as described in the following sections.

13.4.5.1 Di-muon triggers

A di-muon trigger provides a very effective selection for several important B-physics channels, e.g. $B_d \rightarrow J/\psi(\mu^+\mu^-)K_s$ and rare decays of the type $B \rightarrow \mu^+\mu^-(X)$. The LVL1 muon trigger is efficient down to a p_T of about 5 GeV in the barrel region and about 3 GeV in the end-caps. However the actual thresholds used for the di-muon trigger will be determined by rate limitations. For example, a p_T threshold of 6 GeV would give a di-muon trigger rate of about 200 Hz after LVL1 at a luminosity of 2×10^{33} cm^{-2} s^{-1}. These triggers are mostly due to muons from heavy-flavour decays, see Figure 13-23, plus some single muons which are doubly counted when they traverse more than one pivot plane in the muon trigger chambers (in most cases such double counting is already resolved at LVL1). Doubly-counted single muons are removed at LVL2 which also sharpens the p_T threshold, reducing the rate to about 200 Hz. In the EF, the tracks are reconstructed using offline-quality algorithms and specific selections are made on the basis of mass and decay-length cuts. These consist of semi-inclusive selections, for example to select $J/\psi(\mu^+\mu^-)$ decays with a displaced vertex, and in some cases exclusive selections such as for $B_{d,s} \rightarrow \mu^+\mu^-$. Estimated trigger rates are shown in Table 13-9..

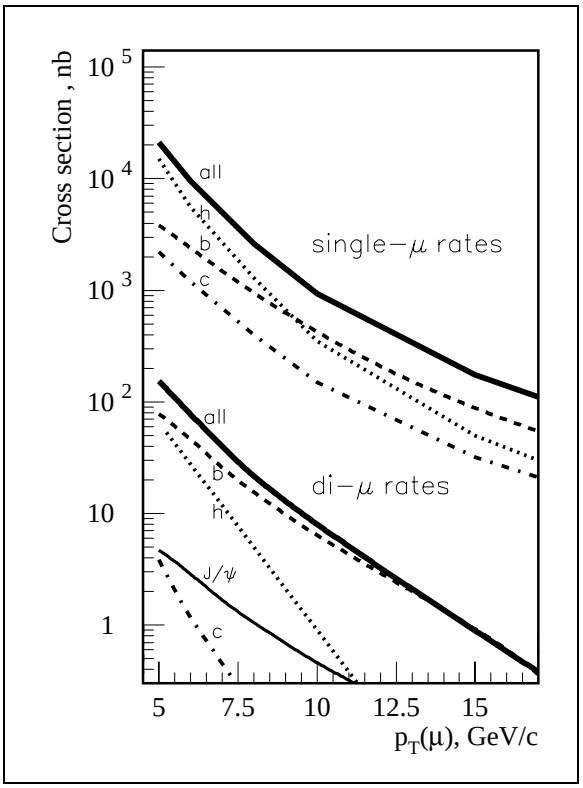

Figure 13-23 Single-muon and di-muon cross-sections. Curves are shown for muons from K and π decays in flight (labelled “h”), b and c decays, and for the sum of these sources (“all”). Muons are considered for $|\eta| < 2.5$. For di-muons, the horizontal axis shows the p_T of the ;ower p_T muon. At least one muon must have $p_T > 6$ GeV and $|\eta| < 2.4$ corresponding to LVL1 trigger conditions..

13.4.5.2 Hadronic final states

For events with a muon trigger, two strategies have been studied for selecting hadronic final states based on either an ID full-scan or an RoI-based method. The ID full-scan consists of track reconstruction at LVL2 within the entire volume of the SCT and pixel detectors and, optionally, the TRT. The alternative, preferred, strategy uses low-E_T LVL1 jet clusters to define RoIs for track reconstruction in the ID. By limiting track reconstruction to the part of the ID lying within the RoI, only about 10% of the detector on average, there is potential for a significant saving in data movement and processing time compared to the full-scan (up to a factor of ten, depending on the RoI multiplicity per event).

Preliminary studies of efficiency and jet-cluster multiplicity at LVL1 have been made using a fast simulation program. This uses a rather detailed model of the calorimeter which includes a parametrization of longitudinal and transverse shower profiles, and takes into account the effects of pulse history, digitization and Bunch Crossing IDentification (BCID). These studies indicate that a cut $E_T > 5$ GeV on the transverse energy reconstructed in the LVL1 jet window gives a reasonable mean jet-cluster multiplicity of about two RoIs per event for events containing a muon with $p_T > 6$ GeV[1], see Figure 13-24. .

Table 13-9 Estimated trigger rates for examples of B-physics trigger selections at luminosities of 2×10^{33} cm^{-2} s^{-1} and 1×10^{33} cm^{-2} s^{-1}. In these examples the di-muon selection at LVL2 consists only of confirming the muons. The division of selections between LVL2 and the EF remains to be optimised.

	2×10^{33} cm^{-2} s^{-1}		1×10^{33} cm^{-2} s^{-1}	
Trigger	**LVL2**	**EF**	**LVL2**	**EF**
$B_{d,s} \rightarrow \mu^+\mu^-(X)$	200 Hz	small	100 Hz	small
$J/\psi(\mu^+\mu^-)$		10 Hz		5 Hz
$D_s(\phi\pi)$	–	–	60 Hz	9 Hz
$B(\pi\pi)$	–	–	20 Hz	3 Hz
$J/\psi(ee)$	–	–	10 Hz	2 Hz
Total	200 Hz	10 Hz	190 Hz	20 Hz

These studies have been repeated using a full GEANT-based detector simulation followed by a simulation of the LVL1 calorimeter trigger. All sources of electronic noise are simulated, both for the LAr and Tile calorimeters, as well as for the trigger towers. The effect of the BCID system was not simulated, however, and as a result there is a significant contribution to the RoI multiplicity from small signals from other bunch crossings. The mean number of jet RoIs found per event is shown in Figure 13-24 as a function of the LVL1 E_T threshold. The mean multiplicities are significantly higher than those obtained with the fast simulation — this is most probably explained by the lack of BCID in the simulation. Once BCID is included, the measured multiplicities are expected to be much closer to the results of the fast simulation, although this remains to be verified. The full-simulation results, therefore, represent an upper limit, with the anticipated multiplicity lying closer to the fast-simulation results. The full simulation shows that a LVL1 jet

1. These studies used a muon-p_T threshold of 6 GeV. The LVL1 jet and EM RoI multiplicities can be assumed to be similar for events selected with an 8 GeV muon threshold, within the accuracy of these estimates.

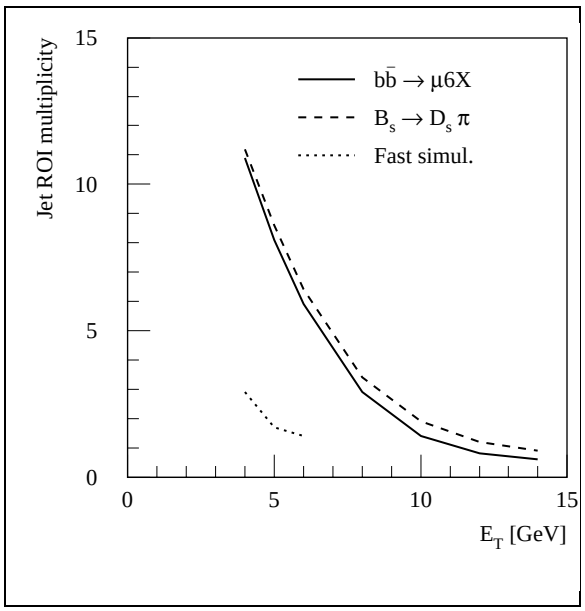

Figure 13-24 The mean number of jet RoIs per event shown as a function of the E_T cut. Results are shown for bb→μX events with a muon of p_T > 6 GeV. Results are shown for both the fast simulation (dotted curve) and full simulation (solid line). The mean multiplicity is also shown for $B_s \rightarrow D_s \pi^+$ signal events with full simulation.

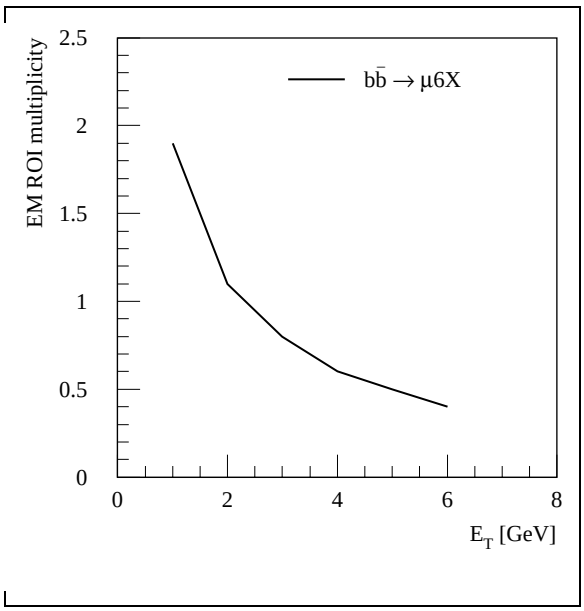

Figure 13-25 The mean number of EM RoIs per event shown as a function of LVL1 E_T cut. Results are shown from the fast simulation for bb→μX events with muon p_T > 6 GeV.

trigger requiring E_T > 6 GeV would give 80% efficiency for finding the RoI for $B_s \rightarrow D_s \pi^+$ events containing a B_s meson with p_T > 16 GeV

Based on the ID tracks reconstructed by one of the above methods (either full-scan or RoI-based) further selections are made for specific channels of interest. These are kept as inclusive as possible at LVL2, with some more exclusive selections at the EF. For example, samples of $B_s \rightarrow D_s \pi$ and $B_s \rightarrow D_s a_1$ events can both be triggered by selecting events containing a $D_s(\phi\pi)$ candidate.

Tracks are reconstructed in the EF inside RoIs defined from the results of LVL2. Using LVL2 to guide the EF reconstruction reduces the amount of data to be processed. For example, a region encompassing all LVL2 tracks forming $D_s(\phi\pi)$ or $B(\pi^+\pi^-)$ candidates corresponds to about 10% of the ID acceptance, on average. Tighter mass cuts can be applied in the EF than at LVL2 since there is time available for a more refined track reconstruction which yields better track-parameter resolutions. In addition, EF selections may include precise decay-vertex reconstruction, allowing further cuts on vertex-fit quality and decay length.

Studies using a full-detector simulation have shown that an overall HLT efficiency of about 60%[1] can be obtained for $B_s \rightarrow D_s \pi$ signal events where all final-state particles have p_T > 1.5 GeV. The corresponding trigger rates are shown in Table 13-9; further details are given in Ref. [13-75]. It has been shown that, with appropriate tuning of selection cuts, there is very little degradation of trigger performance if the pixel layout is changed from three barrel layers to the two layers expected at the start of LHC running. For example, studies based on the LVL2 IDSCAN algo-

1. These studies were based on a full-scan of the SCT and pixels at LVL2 with a p_T threshold for reconstructed tracks of 1.5 GeV. The efficiency for an RoI-based trigger has yet to be measured.

rithm show that reducing the cut on the number of SCT and pixel space-points to a minimum of four for the initial layout (compared to a minimum of five for the full layout) yields the same efficiency as for the full layout with only a 10% increase in trigger rate. Other studies have shown that the trigger is insensitive to the anticipated levels of misalignment in the ID [13-76].

13.4.5.3 Muon–electron final states

A muon–electron trigger is used to select channels such as $B_d \rightarrow J/\psi(e^+e^-)K_s$ with an opposite-side muon tag, or $B_d \rightarrow J/\psi(\mu^+\mu^-)K_s$ with an opposite-side electron tag. The LVL1 trigger is used to find low-E_T electron / photon clusters which define RoIs to be investigated at LVL2. Preliminary studies, using a fast simulation, show that a reasonable compromise between RoI multiplicity and electron efficiency might be obtained requiring E_T>2 GeV for the LVL1 cluster. This gives a mean RoI multiplicity of about one for events containing a muon with $p_T > 6$ GeV, see Figure 13-25. These studies give an efficiency of about 80% to find an RoI for both the e^+ and the e^- from $J/\psi(e^+e^-)$ in events where both final-state particles have $p_T > 3$ GeV. At LVL2, the electron / photon RoIs are confirmed in the calorimeter, using full-granularity information and including the pre-sampler. Inside the RoI, a search is then made for tracks in the SCT, pixels and, optionally, the TRT. The RoI about each electron candidate can be quite small, about $\Delta\eta \times \Delta\phi = 0.2 \times 0.2$. This gives a large saving in reconstruction time, compared to a full-scan, but requires a higher p_T threshold than is possible with the ID full-scan. The tracks are reconstructed again in the EF, including a vertex fit, and cuts are applied to the decay length and fit quality. The estimated trigger rates are shown in Table 13-9.

13.4.5.4 Resource estimates

In order to estimate the computing resources required for the B-trigger, measurements of algorithm execution times have been combined with estimates of trigger rates at each step of the selection. Various reconstruction algorithms have been timed on several different platforms in order to determine the mean execution time at a given luminosity, the scaling of execution time with the number of hits in an event, and hence the scaling with luminosity. These timing measurements have been combined with the estimates of trigger rates and RoI multiplicities to give an estimate of the computing resources required for the B-physics trigger [13-77].

It is estimated that ~15 additional 2 GHz processors are required for the B-physics LVL2 trigger, using the preferred RoI-guided strategy. The corresponding number is about 50 processors for the fall-back full-scan strategy. These estimates are based on algorithm execution times measured in a previous software framework, CTrig. Preliminary measurements in the new Athena framework give algorithm execution times (excluding data access overheads) that are a factor 2–3 higher than the CTrig measurements. It is anticipated that with further optimization, algorithm speeds will approach those obtained in the CTrig framework.

In summary, the use of low-E_T LVL1 RoIs to guide reconstruction at LVL2 promises a full programme of B-physics for very modest resources. However, RoI multiplicities and efficiencies need to be verified in studies using a full-detector simulation and including a detailed simulation of the LVL1 trigger, with BCID. This will be followed up in further studies.

13.5 HLT output rates to off-line

After the submission of the HLT/DAQ Technical Proposal [13-1], the baseline scenario for the start-up of the LHC was changed to a target initial peak luminosity of 2×10^{33} cm^{-2} s^{-1}. The information already available from the detailed HLT performance studies, as documented in the TP, was used to derive the ATLAS HLT trigger menu for this new scenario; the work concentrated on the parts of the menu contributing most of the rate, consisting mainly of inclusive high-p_T object selections. In addition, the impact of constraints coming from delayed availability of financial resources was taken into account by restricting the B-physics trigger selection at peak luminosity to di-muon signatures only (see Section 13.4.5). The resulting trigger menus for LVL1 and for the HLT are presented in Table 13-10 and Table 13-11 respectively. The notation used in these tables is explained in Chapter 4.

The signatures shown in the trigger menus are needed to guarantee complete coverage of the ATLAS physics programme for observations of new physics and precision measurements. As discussed in Chapter 4, the aim is: to be as open as possible to (perhaps unpredicted) signatures of new physics; to avoid as much as possible any bias introduced in the trigger selection; and to allow refined selection criteria to be used in the offline analyses. It should be noted that these menus assume stable operation of the LHC machine and the ATLAS detector, and thus do not apply to the initial start-up phase of ATLAS and the LHC, which is addressed in more details in the following section.,

Table 13-10 LVL1 trigger menu with rates for a luminosity of 2×10^{33} cm^{-2} s^{-1}

LVL1 signature	rate (kHz)
EM25I	12.0
2EM15I	4.0
MU20	0.8
2MU6	0.2
J200	0.2
3J90	0.2
4J65	0.2
J60+XE60	0.4
TAU25I+XE30	2.0
MU10+EM15I	0.1
others (pre-scaled, exclusive, monitor, calibration)	5.0
Total	~25.0

Table 13-10 lists the main LVL1 selection signatures together with the rates expected for an initial peak luminosity of 2×10^{33} cm^{-2} s^{-1}. For most of the signatures, the nominal threshold given is defined to give an efficiency of about 95% for a physics object (μ, e/γ, etc.) with E_T equal to the threshold value. This is achieved by an appropriate tuning of the selection cuts at LVL1. The selection signatures at LVL1 include single- and di-object isolated electromagnetic-calorimeter clusters (which provide candidates for electron and photon object reconstruction at the HLT), single- and di-muon candidates, and the combination of an isolated electromagnetic cluster and

a low-p_T muon. In addition, hadronic jets are selected for various multiplicity requirements and E_T thresholds. It is expected to also include a di-jet trigger (with a threshold lower by several tens of GeV than the one of the inclusive single-jet trigger), which is not shown in the table. Finally, signatures requiring large missing transverse energy, in association with either a jet or a tau[1] candidate are present. About 20% of the total rate of 25 kHz is allocated for pre-scaled, exclusive, calibration, monitor and other triggers, examples of which are described in Chapter 4**.**

It should be noted that the LVL1 trigger rate estimates are obtained using a physics event Monte Carlo generator and a detailed simulation of the ATLAS detector response. They do not contain any safety factor against uncertainties on the rates, for which possible sources are discussed detail later. For the design value of the maximum LVL1 accept rate of 100 kHz, this trigger menu thus gives an effective safety factor of four against possible uncertainties affecting the LVL1 trigger rate**.**

Table 13-11 HLT trigger menu with rates for a luminosity of 2×10^{33} cm^{-2} s^{-1}

HLT signature	rate (Hz)
e25i	40
2e15i	<1
γ60i	25
2γ20i	2
μ20i	40
2μ10	10
j400	10
3j165	10
4j110	10
j70+xE70	20
τ35i+xE45	5
2μ6 with vertex, decay-length and mass cuts (J/ψ, ψ', B)	10
others (pre-scaled, exclusive, monitor, calibration)	20
Total	~200

Table 13-11 shows the HLT output rate corresponding to the above LVL1 trigger menu; as for before, it does not include any safety factor against uncertainties on the rates. The rates shown were obtained using the selection algorithm steps at LVL2 and in the EF for the various objects, as described in detail in the previous sections of this chapter. The LVL1 electromagnetic selections separate at the HLT into single-/di-electron and single-/di-photon signatures, which together account for about 1/3 of the HLT output rate. About 25% of the HLT output rate is from the single- and di-muon triggers, whereas single- and multi-jet triggers constitute about 15% of

1. In past documents [13-49], [13-3], [13-1] the LVL1 tau trigger always implicitly required isolation criteria to identify narrow hadronic jets from tau decays, which was not signalled by the letter 'i' in the physics object shown in trigger menus. In this document, the notation has been updated to indicate more correctly the selection conditions.

the total rate. Selections involving missing transverse energy contribute about 15% of the rate. Only 5% of the HLT rate at peak luminosity is allocated to B-physics related triggers (the low-p_T di-muon signature with additional mass cuts to select J/ψ, ψ' and rare B-meson decays). About 10% of the total rate is allocated for pre-scaled and other triggers.

As already mentioned for the LVL1 trigger menu, it is important to note that the rate estimates are the result of simulations, which start with a physics event generator (mostly PYTHIA) and then involve a detailed GEANT-based simulation of the ATLAS detector response. The rate estimates are thus subject to several sources of uncertainties:

- **Knowledge of cross-sections:** some of the cross-sections (e.g. the ones for multi-jet production) have big uncertainties (factors of two or more), which directly affect the corresponding contributions to the trigger rates.
- **Realistic detector behaviour and performance:** the results presented in this document are obtained using the simulated detector behaviour, thus they apply only for stable running conditions with a commissioned and well-understood detector.
- **Background conditions:** unforeseen beam-related background conditions could have an impact on the trigger rates; such backgrounds could lead to increased rates (possibly only from certain regions of the detector).
- **Resources constraints and software performance:** the timing performance of the selection software when faced with real data might limit the rate capabilities of the HLT (or intermediate stages of it) given constraints from the availability of computing resources — the choice of LVL1 thresholds for the physics objects used in the on-line selection might have to be adjusted as a result, with implications also for the HLT thresholds.

As mentioned already, the menus discussed above are shown for the target initial peak luminosity of 2×10^{33} cm^{-2} s^{-1}. It is expected that during a beam coast, as the luminosity drops by a factor of about two or more, or for machine fills which do not reach the target peak luminosity, the selection will be enlarged by assigning a larger fraction of the available rate to pre-scaled triggers (and / or to include additional exclusive or topological selections, which are not activated at the target luminosity). This would result in the writing of events at a constant rate to mass storage, making full use of the online data-movement and computing resources.

13.6 Start-up scenario

The TDAQ system will have to cope with an evolving data-taking environment, in particular during the start-up phase of the detector operation. It will have to accommodate:

- quickly-evolving luminosity;
- variable beam-related background conditions;
- variable electronic-noise conditions;
- changing configuration of sub-detector electronics, subject to final adjustment iterations;
- limited data-handling capacity of the initial HLT / DAQ system imposed by financial limitations and the need to add computing and data-movement capacity as new resources become available;
- learning stages of the detector operation.

The TDAQ system commissioning thus has to be considered as a self-adjusting process, optimised to use efficiently the data available at each stage of the detector operation — at first data collected in sub-detector stand-alone runs, then data collected in the cosmic-ray muon runs, subsequently data collected in single-beam operation of LHC, and eventually beam–beam collision data.

13.6.1 LVL2 trigger and EF commissioning

The commissioning phase of the HLT/DAQ system will follow that of the subdetectors and of the LVL1 trigger. Many issues will need to be addressed at that stage, to achieve proper synchronization of LVL1 signals from trigger detectors (calorimeters, RPC, TGC) and of the CTP. This relates as well to the operations of Data Collection, Data Flow and Event Builder components, that feeds the algorithms at the LVL2 and EF levels with the proper data fragments or fully-assembled events. Also data consistency from all ATLAS sub-detectors will need to be established, and pre-processing and formatting will need to be studied since they directly impact the behaviour of the data-preparation step of the HLT algorithms.

Once the sub-detector data have been understood sufficiently (e.g. in terms of calibration and alignment), it will be necessary to debug and tune the selection of events at LVL2 and in the EF. The commissioning of LVL2 and the EF can be facilitated by splitting it into several phases. Initially the HLT systems will probably be used only to flag events, allowing offline debugging and testing of the selection before it is used actively in the online environment.

Needless to say, as many preparations as possible will made before the first beam–beam collisions occur, starting with test data, followed by an extended period of cosmic-ray running. It may be possible to carry out further commissioning during running with a single proton beam, e.g. using beam–gas interactions in the ATLAS detector.

Once the HLT is operational, there will be a period in which the selection criteria at all trigger levels are adapted, perhaps radically, in the light of experience from the first running. Account will have to be taken of any limitations in the performance of the initial HLT/DAQ system (e.g. any bottlenecks in the Dataflow or HLT systems), and also any problems with the detectors or the LVL1 system (e.g. ‘hot channels’). The trigger menus will have to be tuned, taking into account the measured rates for different LVL1 signatures, to optimize the physics coverage within the available HLT/DAQ capacity. Account will also have to be taken of triggers due to beam-related backgrounds that might be particularly significant in the early running.

13.7 References

13-1 *ATLAS High-Level Triggers, DAQ and DCS Technical Proposal*, CERN/LHCC/2000-17 (2000)

13-2 ATLAS Data Challenge 1,
http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/DC/DC1/DC1-Report-V1.0-211002.pdf

13-3 *ATLAS Level-1 Trigger Technical Design Report*, CERN/LHCC/98-14 (1998)

13-4 T. Schörner-Sadenius and T. Wengler, *Formats and Interfaces in the Simulation of the ATLAS First Level Trigger*, ATL-DA-ES-0029 ()
https://edms.cern.ch/document/345463/

13-5 M. Abolins et al., *Specification of the LVL1 / LVL2 trigger interface*, ATL-D-ES-0003 () https://edms.cern.ch/document/107485/

13-6 T. Schoerner-Sadenius, *The ATLAS Level-1 Trigger offline Simulation: Overview, Core Parts, and Use*, ATLAS Internal Note, ATL-COM-DAQ-2003-008 (2003)

13-7 *Xerces: XML parser in C++*, http://xml.apache.org/xerces-c/

13-8 E. Eisenhandler, *Level-1 Calorimeter Trigger URD*, ATL-DA-ES-0001 (1998) https://edms.cern.ch/document/ATL-DA-ES-0001/

13-9 A.T. Watson, *Updates to the Level-1 e/gamma and tau/hadron Algorithms*, ATLAS Internal Note, ATL-DAQ-2000-046 (2000)

13-10 E.J.W. Moyse and A.T. Watson, *Performance and Validation of TrigT1Calo, the offline Level-1 Calorimeter Trigger Simulation*, ATLAS Internal Note, ATL-COM-DAQ-2003-010 (2003)

13-11 ATLAS Collaboration, *Muon Spectrometer TDR*, CERN/LHCC/97-22 (1997)

13-12 A. Di Mattia and L. Luminari, *Performances of the Level-1 Trigger System in the ATLAS Muon Spectrometer Barrel*, ATLAS Internal Note, ATL-DAQ-2002-008 (2002) http://atlas.web.cern.ch/Atlas/GROUPS/MUON/layout/muon_layout_P.html

13-13 A. Di Mattia et al., *A muon trigger algorithm for Level-2 feature extraction*, ATLAS Internal Note, ATL-DAQ-2000-036 (2000)

13-14 A. Di Mattia et al., *RPC Trigger Robustness: Status Report*, ATLAS Internal Note, ATL-DAQ-2002-015 (2002)

13-15 S. Veneziano, *Preliminary Design Report of the MUCTPI Electronics*, ATC-RD-ER-0001 () https://edms.cern.ch/document/310152/

13-16 N. Ellis, *Central Trigger Processor URD*, ATL-DA-ES-0003 (1999) https://edms.cern.ch/document/ATL-DA-ES-0003/

13-17 P. Farthouat, CTP Technical Specification, ATL-DA-ES-0006 (1999) https://edms.cern.ch/document/107447

13-18 G. Schuler et al., Central Trigger Processor Demonstrator (CTPD), ATL-DA-ER-0005 (2000) https://edms.cern.ch/document/115660/

13-19 I. Brawn et al., A Demonstrator for the ATLAS Level-1 Central Trigger Processor, ATL-DA-ER-0008 (2000) https://edms.cern.ch/document/249113/

13-20 R. Blair et al., *ATLAS Second Level Trigger Prototype RoI Builder*, ATL-D-ES-0011 (2003) https://edms.cern.ch/document/367638/

13-21 A.G. Mello, S. Armstrong, and S. Brandt, *Region-of-Interest Selection for ATLAS High Level Trigger and offline Software Environments*, ATLAS Internal Note, ATL-COM-DAQ-2003-005, ATL-SOFT-2003-005 (2003)

13-22 S. Armstrong et al., *Requirements for an Inner Detector Event Data Model*, ATLAS Internal Note, ATL-DAQ-2002-011 (2002)

13-23 PESA Core Algorithms Group, S. Armstrong editor, *Algorithms for the ATLAS High Level Trigger*, ATLAS Internal Note, ATL-COM-DAQ-2003-013 (2003)

13-24 C. Leggett and A. Schaffer, ATLAS EDM-DD Workshop, CERN, 2003

13-25 K. Assamagan et al., *Definition of Raw Data Objects for the MDT Chambers of the Muon Spectrometer*, ATLAS Internal Note, ATL-COM-MUON-2003-020 (2003)

13-26 T.A.M. Wijnen, *The MROD Data Format and the towering partioning of the MDT chambers*, ATLAS Internal Note, ATL-COM-MUON-2002-011 (2002)

13-27 P. Bagnaia et al., *Online versus offline identifiers for the MDT chambers of the Muon Spectrometer*, ATLAS Internal Note, ATL-COM-MUON-2003-017 (2003)

13-28 K. Assamagan et al., *Raw Data Object Definition for the RPC chambers of the ATLAS Muon Spectrometer*, ATLAS Internal Note, ATL-COM-MUON-2003-19 (2003)

13-29 G. Aielli et al., *Data Format of the RPC Detector of the Muon System*, ATLAS Internal Note, ATLAS-COM-MUON-2003-018 (2003)

13-30 H. Drevermann and N. Konstantinidis, *Determination of the z position of primary interactions in ATLAS*, ATLAS Internal Note, ATL-DAQ-2002-014 (2002)

13-31 H. Drevermann and N. Konstantinidis, *Hit Filtering and Grouping for Track Reconstruction*, ATLAS Internal Note, ATL-DAQ-2003-XXX (2003), document in preparation

13-32 I. Gaines, S. Gonzalez and S. Qian, *Implementation of an Object Oriented Track Reconstruction Model into Multiple LHC Experiments*, Computing in High Energy and Nuclear Physics Conference, Padova, 2000

13-33 D. Candlin, R. Candlin and S. Qian, *Update of an Object Oriented Track Reconstruction Model for LHC Experiments*, Computing in High Energy and Nuclear Physics Conference, Beijing, 2001

13-34 J. Baines et al., *B-Physics Event Selection for the ATLAS High Level Trigger,* ATLAS Internal Note, ATL-DAQ-2000-031 (2000)

13-35 J. Baines et al., *Global Pattern Recognition in the TRT for B-Physics in the ATLAS Trigger*, ATLAS Internal Note, ATL-DAQ-99-012 (1999)

13-36 M. Sessler and M. Smizanska, *Global Pattern Recognition in the TRT for the ATLAS LVL2 Trigger*, ATLAS Internal Note, ATL-DAQ-98-120 (1998)

13-37 S. Sivoklokov, presentations made in PESA Core Algorithms Group meetings, December 2002 and January 2003

13-38 S. Sivoklokov, *High pT Level-2 Trigger Algortihm for the TRT Detector in ATRIG*, ATLAS Internal Note, ATL-DAQ-2000-043 (2000)

13-39 M.P. Casado, S. González, and T. Shears, *TrigT2Calo package*, http://atlas-sw.cern.ch/cgi-bin/cvsweb.cgi/offline/Trigger/TrigAlgorithms/TrigT2Calo/

13-40 S. González, T. Hansl-Kozanecka and M. Wielers, *Selection of high-pT electromagnetic clusters by the level-2 trigger of ATLAS*, ATLAS Internal Note, ATL-DAQ-2000-002 (2000)

13-41 S. González, B. González Pineiro and T. Shears, *First implementation of calorimeter FEX algorithms in the LVL2 reference software*, ATLAS Internal Note, ATL-DAQ-2000-020 (2000)

13-42 S. González and T. Shears, *Further studies and optimization of the level-2 trigger electron/photon FEX algorithm*, ATLAS Internal Note, ATL-DAQ-2000-042 (2000)

13-43 *ATLAS Level-1 Trigger Technical Design Report*, CERN/LHCC/98-14 (1998)

13-44 A. Di Mattia, S. Falciano and A. Nisati, *The implementation of the muFast algorithm in the new PESA framework*, ATLAS Internal Note, ATL-COM-DAQ-2003-024 (2003)

13-45 I. Gavrilenko, *Description of Global Pattern Recognition Program (xKalman)*, ATLAS Internal Note, ATLAS-INDET-97-165 (1997)
http://maupiti.lbl.gov/atlas/xkal/xkalmanpp/index.en.html.

13-46 R. Clifft and A. Poppleton, *IPATREC: Inner Detector Pattern Recognition and Track Fitting*, ATLAS Internal Note, ATLAS-SOFT-94-009 (1994)

13-47 D. Adams et al., *Track Reconstruction in the ATLAS Muon Spectrometer with MOORE*, ATLAS Internal Note, ATL-COM-MUON-2003-012, ATL-COM-SOFT-2003-008 (2003)

13-48 D. Adams et al., *MOORE as Event Filter in the ATLAS High Level Trigger*, ATLAS Internal Note, ATL-COM-MUON-2003-013, ATL-COM-SOFT-2003-009, ATL-COM-DAQ-2003-012 (2003)

13-49 *ATLAS Detector and Physics Performance Technical Design Report*, CERN/LHCC/99-14 and CERN/LHCC/99-15 (1999)

13-50 R. Mommsen et al., *Performance studies for electron and photon selection at the event filter*, ATLAS Internal Note, ATL-DAQ-2000-007 (2000)

13-51 R. Mommsen, *Electron trigger performance studies for the event filter*, ATLAS Internal Note, ATL-DAQ-2001-004 (2001)

13-52 J. Baines et al., *Performance Studies of the High Level Electron Trigger*, ATLAS Internal Note, ATL-COM-DAQ-2003-0020 (2003)

13-53 J. Baines et al., *First study of the LVL2-EF boundary in the high-p_T e/gamma high-level-trigger*, ATLAS Internal Note, ATL-DAQ-2000-045 (2000)

13-54 T. Sjostrand, Comput. Phys. commun. 82 (1994) 74

13-55 T. Sjostrand et al., Comput. Phys. Commun. 135 (2001) 238

13-56 E. Richter-Was, *Samples from the DC1 production: dataset 2000, dataset 2001, dataset 2003; few items from physics content evaluation*, ATLAS Internal Note, ATL-COM-PHYS-2003-026 (2003)

13-57 S. Tapprogge, ATLAS Rates, Presentation at LHCC Meeting, May 2002,
http://agenda.cern.ch/askArchive.php?base=agenda&categ=a02443&id=a02443s1t2/transparencies

13-58 A. Virchaux et al., *MUONBOX: a full 3D Tracking Programme for Muon Reconstruction in the ATLAS Spectrometer*, ATLAS Internal Note, ATL-MUON-1997-198 (1997)

13-59 GCALOR Package,
http://wswww.physik.uni-mainz.de/zeitnitz/gcalor/gcalor.html

13-60 B. Caron, J. L. Pinfold and R. Soluk, *A Study of the ATLAS Tau/hadron Trigger*, ATLAS Internal Note, ATL-COM-DAQ-2003-030 (2003)

13-61 *ATLAS Trigger Performance Status Report*, CERN/LHCC/98-15 (1998)

13-62 B. G. Piniero, *Tau Identification in the Second Level Trigger*, ATLAS Internal Note, ATL-DAQ-1998-127 (1998)

13-63 D. Cavalli and S. Resconi, *Combined Analysis of $A/H \rightarrow \tau\tau$ Events from Direct and Associated bbA Production*, ATLAS Internal Note, ATL-PHYS-2000-005 (2000)

13-64 D. Cavalli et al., *Search for $A/H \rightarrow \tau\tau$ Decays*, ATLAS Internal Note, ATL-PHYS-94-051 (1994)

13-65 D. Cavalli and S. Resconi, *Tau Jet Separation in the ATLAS Detector*, ATLAS Internal Note, ATL-PHYS-98-118 ()

13-66 D. Cavalli, *Missing Transverse Momentum Reconstruction in ATLAS*, ATLAS Internal Note, ATL-PHYS-96-080 (1996)

13-67 B. Caron et al., *Event Filter Rates for the E_T-miss + jet Trigger at Low Luminosity*, ATLAS Internal Note, ATL-DAQ-2000-016 (2000)

13-68 D. Cavalli et al., *Status Report on the τ-Identification Performance*, ATLAS Physics Workshop, Athens, 2003

13-69 M. Wielers, *Performance Studies of Jets in the High Level Trigger*, ATLAS Internal Note, ATL-DAQ-2000-015 (2000)

13-70 J. de Jong and J. L. Pinfold, *Estimating the ATLAS Multi-jet Trigger Rate Using ATLFAST*, ATLAS Internal Note, ATL-COM-DAQ-2003-029 (2003)

13-71 M. Cervetto et al., *b-tagging Event Selection for the ATLAS High Level Trigger: an update*, ATLAS Internal Note, ATL-COM-DAQ-2003-034 (2003)

13-72 M. Cervetto et al., *SiTrack: a LVL2 track reconstruction algorithm based on Silicon detectors*, ATLAS Internal Note, ATL-COM-DAQ-2003-25 (2003)

13-73 A. Watson and V. Ghete, *A study of the use of Low E_T calorimeter RoI in the ATLAS B-Trigger*, document in preparation

13-74 J. Baines et al., *B-Physics Event Selection for the ATLAS High Level Trigger*, ATLAS Internal Note, ATL-DAQ-2000-031 (2000)

13-75 B. Epp, V.M. Ghete and A. Nairz., *Event Filter Rate for the Ds Trigger*, ATLAS Internal Note, ATL-DAQ-2001-003 (2001)

13-76 J. Baines et al., *Effects of Inner Detector Misalignment and Inefficiency on the ATLAS B-physics Trigger*, ATLAS Internal Note, ATL-DAQ-2001-006 (2001)

13-77 J. Baines et al., *Resource Estimates for the ATLAS B-physics Trigger*, ATLAS Internal Note, ATL-COM-DAQ-2002-001 (2001)

14 Overall system performance and validation

14.1 Introduction

In this chapter the system performance of the design presented in this TDR in terms of rate capability and functionality is considered. Results of tests and of modelling, aimed at validating the various aspects of the design, are presented and discussed. The tests concern the performance of event selection, tests of the rate capability of the Data Flow system in an environment representing about 10% of the final system (the '10% testbed'). Specialized functionality tests of the whole DAQ chain as well as the experience gained from using it for real data taking in the H8 testbeam[1] are also presented. Computer models have been used for analysing measurement results from the 10% testbed, and then to validate the models of components which were calibrated using measurement results from small test setups. Full system models have provided insight into and strategies for avoiding potential problem areas with respect to rate capability of the full system. The chapter is concluded with an outlook with respect to anticipated technology developments relevant for the HLT/DAQ system in the near future.

14.2 High-Level Trigger Prototypes

The High-Level Trigger will select and classify events based on software largely developed in the offline environment. This approach minimizes duplication of development effort, eases software maintenance, and ensures consistency between the offline and the online event selections. However, given the strict performance requirements of a real-time online environment, it is essential to evaluate the performance of the HLT event selection software (ESS) in a realistic trigger prototype.

The resource utilization characteristics of the HLT software are an important input to the models that predict overall system size and cost. For this reason, a prototyping program was developed to perform dedicated system performance measurements of the event selection software in a testbed environment.

14.2.1 Scope of measurement and validation studies

The scope of the work reported here is limited to a system with full event selection and minimal Data Flow capability, providing full trigger functionality with limited performance. Such a dedicated 'vertical slice test' is sufficient to test the performance of the HLT event selection in a realistic environment. Nevertheless, even in such a limited system, tests and measurements of the data flow aspects relevant to event selection can be performed.

An important aspect of this prototyping work is component integration. Although single components may perform very well in isolated tests, only integration with other system elements

1. Large scale and performance tests of the Online Software are discussed in Chapter 10. The aim of these tests was to verify the overall functionality of the Online Software system on a scale similar to that of the final ATLAS installation and to study the system performance aspects in detail.

may reveal weakness not foreseen in the original design. The integration and testing work described here followed the steps outlined below:

1. Individual component testing and validation (addressed in Chapter 8 for RoI collection and event-building and Chapter 13 for the ESS)
2. Functional integration of relevant components (Online Software, Data Flow Software, ESS) in a small testbed, providing feedback to developers.
3. Measurement program, including event throughput and network latencies.

The last two steps were carried out for a LVL2 testbed, an EF testbed, and a combined HLT testbed in the context of validating the HLT/DAQ architecture.

In addition to the testbed measurements, a complementary set of validation tests and measurements can be performed in the offline environment. Although these offline measurements cannot address the full system aspects of the trigger, they help in isolating and understanding the pure algorithmic processing times of the ESS. This is especially relevant for the Event Filter, where events are processed only after all fragments have been assembled, and thus the data flow latencies are completely de-coupled from the ESS latencies.

The following sections summarize the outcome of this integration and measurement program.

14.2.2 Event selection in a LVL2 prototype

14.2.2.1 Prototype and software configuration

All elements necessary to transport and process event data inside the L2PU were assembled in a LVL2 vertical slice prototype. As shown in Figure 14-1(left), the following components were in-

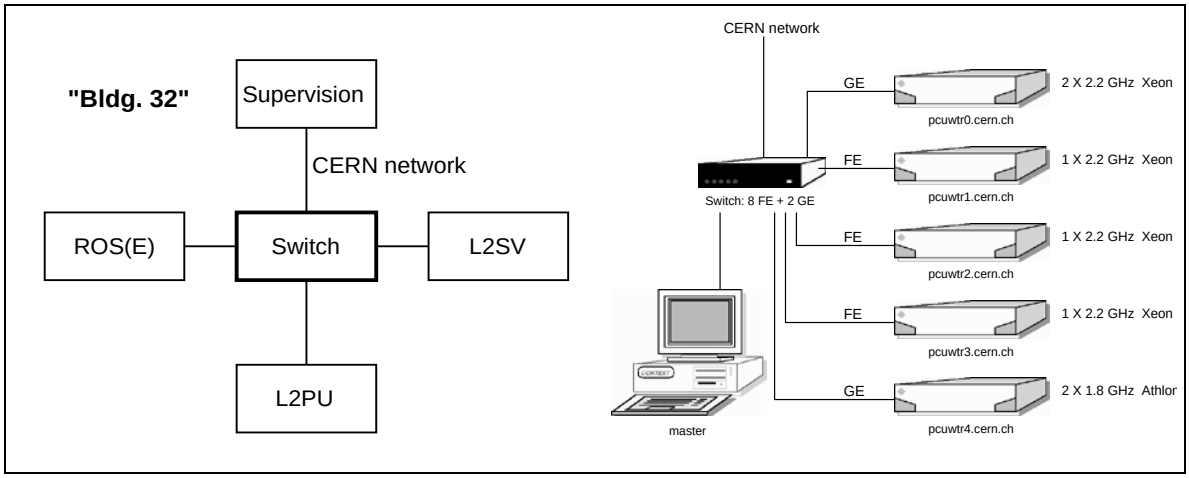

Figure 14-1 The setup for the LVL2 vertical slice testbed. The left figure shows the components in a typical three node configuration. The figure on the right shows the hardware configuration of the five-node LVL2 testbed.

cluded in the prototype:

- L2PU (Described in Section 9.2.4)

- ROS or ROS emulator (ROSE, described in Section 8.1.3)
- LVL2 Supervisor (L2SV, described in Section 9.2.3)

The above applications, all controlled by the Online Software (see Chapter 10), ran on a five-node testbed at CERN. Figure 14-1(right) shows the configuration of the testbed, which was connected to the CERN network through a Fast / Gigabit Ethernet switch. The host machines for the applications were typically Dual-processor Xeon (2.2 GHz), Athlon machines (1.8 GHz) or single processor Xeons (2.4 GHz). A detailed description of the set-up can be found in [14-1].

The L2PU application hosts both the Data Flow and the HLT software. In building the vertical slice prototype, the major challenge was achieving the integration of both software frameworks, including the offline components that form part of the ESS. As described in Section 9.2.4.2, the PSC interfaces the control aspect of the Data Flow and the ESS. The selection software used in the testbed comprised most elements described in Chapter 9, including the detector software necessary to assemble and decode the raw data fragments delivered by the ROS. A detailed description of the software integration within the L2PU, including difficulties and unresolved issues, can also be found in [14-2].

The prototype ran on fully simulated event data. The input data was generated in the offline environment and written in byte-stream format, a serialized version of the raw data format that includes the LVL1 electromagnetic trigger simulation (see Chapter 13). Before starting a run, the detector data fragments were pre-loaded into the ROS (or ROSE) while the LVL1 fragments, corresponding to the RoIs assembled by the RoI Builder, were pre-loaded into the L2SV. Files containing di-jet events at a luminosity of $2 \times 10^{33}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$ were used (see Chapter 13) for the measurements. The events in the file were pre-selected to contain events that pass the LVL1 trigger. Each event contains an average of 1.06 electromagnetic RoIs. A suite of trigger algorithms designed to select electromagnetic clusters ran within the L2PU, together with the appropriate detector software to decode the raw data.

The LVL2 calorimeter trigger is the first step after a LVL1 accept of an electromagnetic cluster. Since the calorimeter algorithm executes at the LVL1 accept rate, it is the most critical component when selecting photons or electrons in the LVL2 trigger. For this reason, the LVL2 electromagnetic selection algorithm (T2Calo, described in Chapter 13) was the first algorithm to be integrated in the LVL2 testbed. Unless otherwise noted, all LVL2 trigger prototype measurements shown here are limited to the LAr calorimeter trigger. However, since all data converters and algorithms share the same offline infrastructure, any problems identified in the testbed for the calorimeter (e.g., issues related to the common data flow aspects of the testbed), would most likely arise with other detectors. In addition, by using the calorimeter as a test case for data flow issues, many performance measurements for the other detectors can be first carried out in the offline environment.

14.2.2.2 Measurements

After a first cycle of design (see Chapter 9), the HLT event selection software was implemented using mostly offline software components. These components, many of which were in the early stages of development at the time of these tests, had not yet been optimized for performance. Nevertheless, after achieving the integration, it was important to obtain performance measurements with this early software so that any incompatibilities with the LVL2 trigger could be identified. By quantifying the behaviour of the ESS in a realistic trigger environment and identifying any bottlenecks [14-3], a feedback cycle could be established with the original developers of the software.

14.2.2.2.1 Measurement methodology

In order to measure the performance of the LVL2 prototype, various key components of the ESS and Data Flow were instrumented with time stamps [14-1]. The time stamps allowed detailed event-by-event profiling of the system behaviour of the prototype. The event throughput, in terms of the mean rate at the L2PU, provided another measure of global performance. All ESS performance measurements quoted here were carried out on a L2PU application running in a dual-CPU 2.2 GHz Xeon machine with 1 GB of memory. Unless otherwise noted, the L2PU was configured to run with one worker thread (see Section 9.2.4.1).

14.2.2.2.2 Initial performance of the ESS in the LVL2 prototype

Initial measurements revealed that the LVL2 Calorimeter ESS alone required considerably more processing time than the ~10 ms per event average budget for the LVL2 trigger. Almost all of the processing time was consumed in the data conversion and preparation steps. This software converts the byte-stream data into objects that the downstream algorithms can process and applies calibration constants. It had only recently been made available and had not yet been optimized.

These first measurements also used the initial form of the so called “London scheme” for data access (described in Chapter 9), where ROB data fragments are requested on demand across the network in a sequential fashion. In this case the total network latencies incurred by the data requests were 3.7 ms. Given that a typical electromagnetic RoI spans 13 to 18 ROBs, this latency measurement agrees with the measurements presented in Chapter 8, where each ROB request with comparable payload introduces a latency of ~220 μs.

The LVL2 calorimeter algorithm mean execution time is 1.5 ms per event, with an additional ~1 ms consumed by framework and service overheads. The physics performance of the LVL2 calorimeter algorithm itself has already been documented elsewhere [14-4].

14.2.2.2.3 First performance improvements

After the above measurements were completed, an initial optimization of the ESS was made for a few critical components. Because the above measurements identified that data transfer and conversion dominated the processing time, work was performed to reduce these contributions. These studies are the first to investigate in detail data access performance.

The initial network latency was reduced by implementing the optimization described in Section 9.5.5.2 whereby all of the data fragments of each RoI are pre-fetched across the network in a single operation. After this change, the network access latency was reduced from 3.7 ms to 650 μs per event for the system configuration shown in Figure 14-1, where one ROS delivers all ROB fragments.

In order to provide a baseline measurement of the minimum time the data converter function would require using no offline-inherited code, a new LAr converter prototype, was developed. This converter satisfies the time-critical needs of LVL2 but avoids off-line dependencies, and was developed based on a lookup table method for region selection and an optimized raw data conversion scheme. In this prototype converter, the event data is passed directly to the requesting algorithm instead of publishing in a Transient Event Store. In addition, a scheme with calibration after zero-suppression was introduced (details can be found in [14-1]). This optimized

converter prototype demonstrated that a data conversion processing time of 1.3 ms per event for an RoI size of $\Delta\eta \times \Delta\phi = 0.3 \times 0.3$ and no zero-suppression was possible.

Applying all of the above improvements gives a total execution time per RoI in the prototype of 3.4 ms and 5.9 ms for RoI sizes of $\Delta\eta \times \Delta\phi = 0.3 \times 0.3$ and 0.5×0.5, respectively.

Figure 14-2 (left) shows the total latency distribution for di-jet events at low luminosity for a

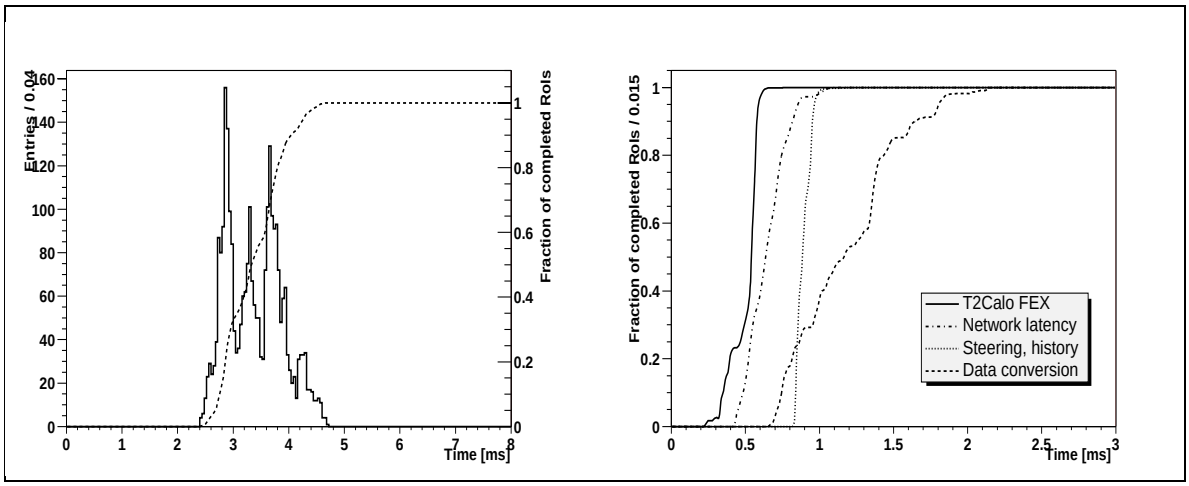

Figure 14-2 Total latency for RoI processing (shown at left) in the LVL2 LAr trigger for di-jet events at low luminosity. The dotted curve is the integral of the distribution, showing that 95% of the events are processed within 5 ms. The four main contributions to the latency are shown (right) as curves of integrals. The contributions are (in order of decreasing importance): data access and conversion, framework overheads, network access time, and algorithmic processing.

$\Delta\eta \times \Delta\phi = 0.3 \times 0.3$ RoI. As can be seen from the figure, over 95% of the events are processed within 5 ms. Figure 14-2 (right) shows the main contributions to the total latency. Pure feature extraction in the calorimeter is completed within 500 μs, while data access is typically completed within 1.8 ms for 95% of the events.

The use of zero-suppression in the calorimeter [14-4] has been shown to be very effective in reducing algorithm execution time. Applying a 20 MeV energy threshold cut in the L2PU during the data conversion step reduced the mean algorithm execution time by 310 μs and 1.4 ms per event for RoI sizes of $\Delta\eta \times \Delta\phi = 0.3 \times 0.3$ and 0.5×0.5, respectively. The bulk of this reduction is due to the reduced number of calorimeter cells that the feature extraction algorithm has to process. The data conversion time could also be reduced if zero-suppression could be applied upstream of the L2PU (*e.g.*, in the RODs). More details on these measurements can be found in [14-1] and [14-5].

These results demonstrate that the system performance required can be achieved with current software, but that some of the offline components need optimization to provide a better match to the LVL2 trigger. The optimizations described above are now being studied for implementation in the LAr data conversion software (for use in both the trigger ESS and the LAr offline itself), and substantial improvements have already been achieved there. Thus, feedback from the re-use of offline software in the LVL2 trigger is mutually beneficial. These LVL2-motivated optimizations will benefit not only the LVL2, but the Event Filter and the offline software itself, reducing the overall computing resource needs of ATLAS.

14.2.2.2.4 Other measurements

Since the L2PU will run multiple worker threads hosting the ESS, it is important to test the LVL2 event selection software in a multi-threaded configuration. After applying the changes necessary to make the ESS thread-safe [14-2], the prototype ran with two worker threads in the L2PU. The event throughput increased from 266 Hz to 323 Hz, although the CPU utilization was only ~50% per CPU. The non-scaling effect is due to an inefficient use of memory allocation locks [14-1] in the Standard Template Library (STL). By partially repairing the inefficient parts of the software, an increase in the event throughput has been observed. This will be improved in the next round of software optimization.

The LVL2 trigger can be configured to run multiple L2PU applications in a single host machine. In this configuration, each L2PU runs different instances of the ESS, each processing events in parallel. This scheme increases the resource requirements on the host machine since memory is not shared between the L2PUs and since the number of external connections increases. The three-node testbed was configured to run with two L2PUs in a dual-CPU host. The event throughput rate was measured to be 470 Hz. In order to draw any conclusions from this study, a careful analysis of the resource implications of using multi-process versus multi-threaded applications must be made. This study will be performed when the multi-threading issues outlined above are resolved.

All LVL2 testbed measurements quoted above have been done for the LAr calorimeter. At the time of writing, the initial implementation of the SCT/Pixel converters based on the Chapter 9 design was only just becoming available and had not been optimized for LVL2. In previous reference measurements for the SCT/Pixel data conversion process, an early SCT and pixel prototype of the trigger software, implemented before the design described in Chapter 9 was available, was developed and tested in the LVL2 testbed [14-5]. This prototype included all software components necessary to map ROBs to an (η,ϕ) region, request the data [14-6], and decode the SCT/Pixel byte-stream data and the LVL1 information. In addition, a LVL2 tracking algorithm, SiTree [14-7], reconstructed high-p_T tracks within each RoI. The prototype yielded a mean total processing time of 2.5 ms per event for track reconstruction of single electrons with no pile-up. This result is significantly less than the ~30 ms required by the first implementation of the new SCT and Pixel converters. However, by implementing improvements similar to those applied to the LAr converter, the SCT and Pixel decoding software can be expected to achieve a similar level of performance to that of the early prototype.

Because it is executed at the LVL1 muon rate, the processing time of the LVL2 muon trigger, like the LVL2 calorimeter trigger, is also critical. Measurements [14-8] of LVL2 muon trigger performance were carried out in the HLT offline environment (i.e. running the ESS in the offline environment rather than in the testbed). Running a full selection chain for $p_T = 100$ GeV muons in the barrel at high luminosity yielded a mean total processing time of ~10 ms on a 2.4 GHz machine. This processing time been calculated taking into account the cavern background simulation in the muon spectrometer for high luminosity ($L = 1 \times 10^{34}$ cm^{-2} s^{-1}) with a safety factor of two (see Section 13.4.2). It is dominated by data preparation – 1 ms per RoI for the RPCs [14-9] and 7.9 ms per RoI for the MDTs [14-10]. The cavern background increases the MDT occupancy and the data preparation time scales linearly with the occupancy. The remaining 1.1 ms is consumed by the muFast pattern recognition and tracking algorithm itself (algorithm description can be found in Chapter 13). Although at this time the LVL2 muon trigger software has not been integrated in a testbed environment, these measurements indicate that the processing times, including data access, are well understood. Assuming that data flow and network overheads for the muon trigger are similar to those for the LAr trigger, the overall latency for the LVL2 muon trigger is well under control.

14.2.2.3 Conclusions and outlook

As seen in the previous section, detailed studies and first measurements with the HLT event selection software show that there is great potential for optimization of the present initial implementation. Some components of the ESS already perform well, e.g., the selection algorithms. Other components, in particular the detector data converters, need further optimization. Dedicated test prototypes have shown that this optimization is possible. In addition, the muon trigger software already performs in the offline HLT environment at a level that is compatible with the LVL2 processing budget. Extrapolating the present performance figures of these prototypes to 8 GHz CPUs which are expected to be available at LHC turn-on, gives total processing times that are compatible with the 10 ms average event processing time budget of LVL2. In all cases, the data preparation is proving to be the major fraction [14-3] of the LVL2 processing time. Thus detailed studies have been made, and will continue, of the data preparation process.

There are a few open issues that need to be addressed for the LVL2 trigger. The multi-threading inefficiencies encountered with the STL need to be resolved, and the implications on the current working model need to be fully understood.

The data access mechanism of the ESS needs to be optimized for the LVL2 trigger. First, the implementation of the internal data access including conversion, calibration and data organization need to be optimally designed for execution speed. Secondly, the ‘on-demand’ nature of many of the offline configuration and data access services, particularly at initialization time, should be adapted to a deterministic model, more suitable for a trigger environment. Thirdly, fragments required for a given RoI must be requested in parallel. Implementing these optimizations brings very substantial performance improvements as was demonstrated above.

The ESS will be processing events in the LVL2 environment at the LVL1 output rate. Consequently, any offline components used in the trigger will be executed nearly 10^3 times more frequently in LVL2 than in offline, imposing severe constraints on stability and robustness of the software. Increasing the modularity of the software so that the components needed to build the LVL2 form a restricted highly-robust software ‘core’ would help to address these constraints. This core would still form the basis for the offline reconstruction.

In conclusion, running the ESS in a LVL2 vertical slice prototype serves two purposes: it provides performance measurements for estimating resource requirements, and it provides a platform for validating the ESS and the trigger architectural choices. Building the event selection code with offline components, not only reduces the development and maintenance costs in the LVL2, but it helps in optimizing the performance of these offline components. However, in order for this model to work, the LVL2 constraints must be taken into account in the core offline software. The LVL2 tests shown here demonstrate that this is not only possible, but desirable and mutually beneficial for both the trigger and offline software systems.

14.2.3 Event selection in an Event Filter prototype

In the Event Filter, the event selection process is carried out by the Processing Task (PT), described in Section 9.3**.** The PT hosts the HLT selection software, which is fully implemented using ATHENA (the ATLAS offline software framework [14-11]) and offline reconstruction components. In order to validate the event selection in the Event Filter, a prototype was developed that brings together the DAQ Data Flow sub-system (SFI and SFO), the EF data flow (EFD) and the PT running the event selection software (ESS) in a single system (see section 9.3.2.3 and [14-12]).

The performance of the EFD running with dummy PTs is described in Section 9.3.2.4. Here we concentrate on the performance involving real events processed by the ESS.

14.2.3.1 Integration of EFD with ATHENA data access services

The PTs are independent processes running on each EF node. In order to access event data, the event selection software running in the PTs are interfaced to the EFD which supplies them with complete events from the Event Builder. This interface has been implemented by adapting the ATHENA conversion services that carry out the conversion between different data types.

The integration of the ESS in the Event Filter consisted then of developing dedicated implementations of some of these data conversion services. These EF-specific implementations read, handle, and exchange event data with the EFD using a shared memory mechanism. When a PT requests an event from the EFD, a pointer to the event in this shared memory is returned to the PT (see Section 9.3.2). After the processing is completed, a trigger decision is sent to the EFD. If the event is accepted, the EFD appends to the original event, data generated during the EF processing.

14.2.3.2 Prototype and software configuration

Tests were performed by running the Event Filter with simulated events. The data set used was a sample of di-jet events that was pre-selected to pass the LVL1 electromagnetic trigger. The data sample is the same as that used for the LVL2 tests and is described in Section 14.2.2. The event size in this sample was ~3 Mbytes. The data were pre-loaded in an SFI emulator. At the time these tests were made, the EF algorithms were not yet available and so a LVL2 calorimeter algorithm was used. Tests using more sophisticated algorithms are planned in the next three months.

For the purposes of these tests, the EF result (data generated during processing) contained a single data fragment produced by a converter dedicated to serialising the results of EF reconstruction, which are in object form. No additional reconstructed objects were serialised in the prototype, however, the same mechanism will be used for multiple object serialization in the future when more complete EF algorithms are available for use in the testbed. For accepted events, this serialized fragment was written into the shared memory from where it was appended to the original event by the EFD and then sent to the SFO.

Three different hardware configurations were used to carry out the measurements:

1. A single-host dual-processor: one Intel Xeon 2.2 GHz processor with 1 Gbyte of RAM
2. A multiple host set-up: two Intel Xeon 2.2 GHz processors with 1 Gbyte of RAM interconnected by Fast Ethernet
3. A multiple host set-up: two Intel Xeon 2.2 GHz processors with 1 Gbyte of RAM interconnected by Gigabit Ethernet

14.2.3.3 Measurements

A series of tests was performed on each of the testbed configurations described above. The tests were:

1. Validation of the exchange of data between the EFD and the ATHENA PT hosting the ESS
2. Throughput measurements.

Validating the event input procedure consisted of checking the integrity of the data after passing it from the EFD to ATHENA, by checking that the ESS produced the same results as when running offline. In order to validate the output procedure, accepted events were sent by the EFD to the SFO and from there to a local disk file. The integrity of these data was then verified by reading, unpacking, and re-processing them offline. The throughput was measured as the average processing time per event (i.e., the inverse of the event rate).

Measurements were conducted with a dual-processor machine hosting all processes, SFI, SFO, EFD, and PTs, (configuration 1 above), and also on a multiple-host setup with the SFI and SFO running on one host and the EFD and PTs running on the other (configurations 2 and 3 above).

The ATHENA PT ran the calorimeter algorithm. The EF selection was configured so that all events were accepted. The total user time per event was on average 180 ms for the di-jet event sample described above, and the virtual memory size was typically 260 Mbytes. Table 14-1 summarises the results.

Table 14-1 Results of throughput measurements obtained with the various test configurations (see text for details)

Configuration	Number of ATHENA PTs	Time per event (1/Rate)	Data volume throughput	Remark
1	1	190 ms	16 Mbytes/s	
	2	110 ms	27 Mbytes/s	
	3	timeout		limited by memory swap
2	1	380 ms	8 Mbytes/s	limited by network
3	1	190 ms	16 Mbytes/s	
	2	120 ms	25 Mbytes/s	

In the first configuration, adding a second PT profits from the dual-CPU and increases the throughput significantly, though not by a factor of two as the other resident processes (SFI, SFO, and EFD) require some fraction of the CPU. Adding a third PT saturates the memory; consequently, swapping slows down the process considerably. Under these conditions, the PTs were blocked by the timeout mechanism of the EFD. In the second configuration, with multiple hosts interconnected by Fast Ethernet, the throughput was limited by the network bandwidth even with only one PT (in normal EF running conditions, with an event size of the order of 1 Mbyte and a latency of the order of 1s, Fast Ethernet bandwidth should be adequate). In the third configuration (multiple hosts interconnected by Gigabit Ethernet), there was no bandwidth limitation and a single PT used close to 100% of a CPU. Adding a second PT increased the throughput. Here, each PT used about 80% of a CPU, the remainder being used by the EFD.

14.2.3.4 Conclusions

The ATHENA-based event selection software was successfully integrated with the EFD, demonstrating that the EF selection can be built using offline components. As in the case of the LVL2, this approach minimizes development and maintenance costs, while providing a unified event selection chain.

Although the tests described here were not conducted with a full EF selection suite, the measurements already highlight some of the key performance issues. The behaviour of the ATHENA PT running the selection software in terms of processing time, memory usage and event sizes will be further evaluated in order to define an optimal Event Filter hardware configuration. These tests demonstrate the correlation of these parameters.

14.2.4 The HLT vertical slice

The LVL2 trigger and the EF were integrated in a single testbed as shown in Figure 14-3. The

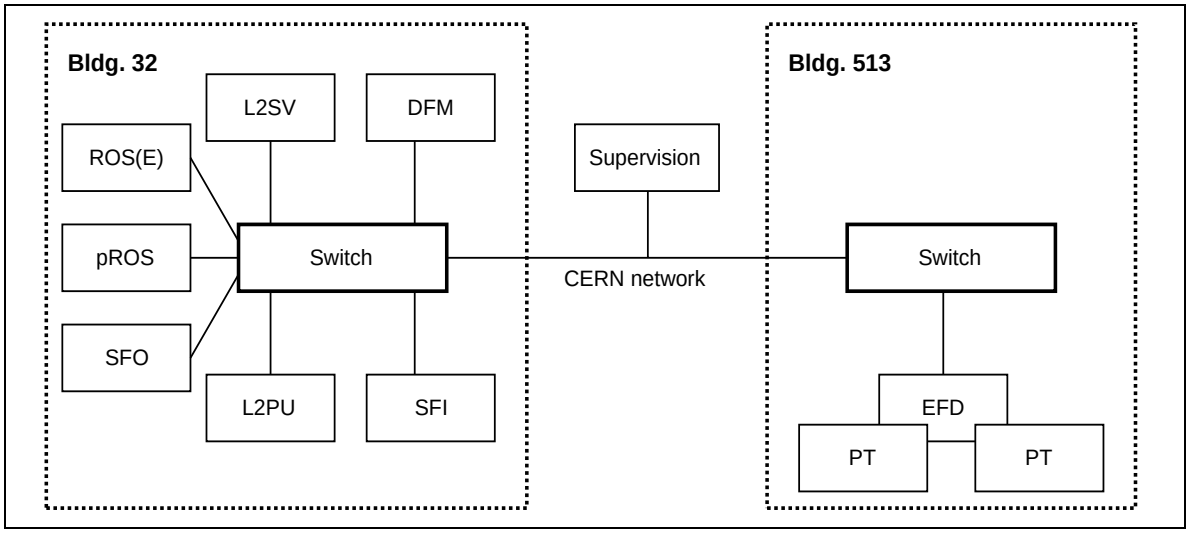

Figure 14-3 Setup for the combined LVL2 and EF vertical slice testbeds.

LVL2 slice (described in Section 14.2.2) and the EF slice (described in Section 14.2.3) were connected to form an 'HLT vertical slice' using the CERN network infrastructure. The DFM and the Event Builder were located geographically close to the event fragment sources. In order to pass the LVL2 result to the EF, one of the ROSs was configured as a pROS (described in Section 8.1.3.4). The entire system, consisting of 11 nodes, was configured and controlled by the Online Software through the CERN network.

During the test, one ROSE was pre-loaded with 120 LVL1-preselected di-jet events with electronic noise as used in the LVL2 slice tests. The energy threshold cut for the LVL2 calorimeter algorithm was set low (20 GeV) in order to accept approximately two thirds of the events. The LVL2 used the T2calo algorithm with optimised data conversion for the LAr Calorimeter (as described in Section 14.2.2). The LVL2 system was configured to execute the ESS in two L2PUs running in parallel. The EF used a non-optimised calorimeter algorithm (which included Tile Calorimeter handling) running in a sub-farm with three nodes, each with one EFD and two PTs. Events accepted by the LVL2 were assembled by the Event Builder (2-3 MBytes/event) and

passed on to the ATHENA-based ESS running in the EF. Finally, accepted events were recorded in byte-stream format by the SFO.

This functional integration was validated by verifying that the event recorded by the SFO was the same as the simulated event selected by the LVL2 trigger. Now that this HLT slice has been functionally demonstrated, it is planned to integrate the HLT slice with the 10% Data Flow testbed (see Section 14.4) in order to study performance aspects of the complete HLT/DAQ system.

14.3 HLT CPU Requirements

The HLT/DAQ architecture has been designed to handle a maximum LVL1 trigger rate of 100 kHz. The estimated CPU requirement of the HLT system to handle this LVL1 rate is summarized in this section.

For LVL2, there are several ingredients to this estimate:

- Examples of the feature extraction and reconstruction algorithm performance are presented in Chapter 13 and its associated references, and in Section 14.2.2. Typical timing numbers on a 2 GHz CPU are:
    - Calorimeter: ~2 ms
    - Muon: ~1 ms
    - SCT/Pixel: ~3 ms
    - TRT: ~9 ms
- The frequency of use of each of the algorithms — calculated from the trigger rates and acceptance factors (see Chapter 13 and Appendix A) and the number of RoIs per event for each RoI type
- The CPU requirement for the preparation of the data to be used by the algorithms

  Initial measurements have indicated that this will be a significant fraction of the total required CPU time. However, as discussed in Section 14.2.2.2, the measured data-preparation times are preliminary and significant improvements can be expected in most cases.
- The time to access data from the ROBs using the RoI mechanism in the LVL2 trigger

  This has been studied in detail and results are presented in Section 8.3.2.2. The data-access time is very small compared both to the algorithm and data-preparation times.
- The CPU overhead of the overall software framework in which the selection algorithms run

  This is estimated to be a few percent of the overall processing time per event.
- Additional, currently ill-defined CPU requirements, such as access to calibration data residing in the conditions databases and monitoring

  It is not foreseen for the LVL2 trigger to access the conditions databases during a run. However, the EF will need this ability. As the conditions database is currently in its design phase, we do not attempt any estimate of data-access times at this stage. This will be addressed further in the Computing TDR. System and local trigger monitoring procedures will certainly increase the overall CPU requirements of the HLT.

The above considerations and the sequential data processing steps (see Chapter 2 and Appendix A) have been used as input parameters to detailed modelling of the LVL2 system. In the model, typical timing numbers have been scaled to 8 GHz CPUs, the speed of CPUs expected to be available in 2007. The result of the model is that for a LVL1 rate of 25 kHz, ~250 CPUs at 8 GHz will be required for LVL2. Scaling this number to the maximum LVL1 rate of 100 kHz gives a total LVL2 CPU requirement of 500 dual-processor machines.

For the EF, performance and timing studies are still in progress. We therefore use a target figure of 1s / event for the average global EF processing time, assuming 8 GHz CPUs. Assuming an event-building rate of 3.2 kHz (corresponding to a LVL1 rate of 100 kHz) this gives an estimate for the EF of 1600 (8 GHz) dual-CPUs. These estimates for LVL2 and the EF are used in the overall HLT / DAQ-system costing presented in Chapter 16.

The resource estimates clearly are subject to large uncertainties coming from various sources:

- At the start-up luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, for the physics selections presented in Chapters 4 and 13, simulations give a LVL1 trigger rate of ~25 kHz without any safety factor and with a limited physics menu. Clearly, many possible selection channels have not been simulated which would add to the LVL1 rate. There are also very large uncertainties in the underlying cross-sections and the background conditions which will affect this estimate. Designing the system for 100 kHz gives an effective ‘safety factor’ of four compared to the above LVL1 rate.
- In extrapolating the LVL1 rate to 100 kHz, we have made the assumption that the mixture of trigger types remains constant.
- We have made extensive studies of trigger-algorithm data preparation as discussed in Chapter 13 and in Section 14.2.2. These results are still preliminary for several reasons since the detector data formats themselves are preliminary, and studies of data converters in the ESS have been done only with initial prototype-software not yet fully optimized for the trigger

  However we can already conclude that the data-preparation CPU requirement will be significant, and in some cases possibly dominant, compared to the algorithm execution time.
- The ESS is a first implementation of the design presented in this TDR. Work both on the trigger testbeds and in offline studies of the trigger algorithms shows that there are many improvements and optimizations which can be made.
- The ATLAS offline software, upon which much of the ESS is based, has undergone a complete re-design and implementation in the last three years and this process will continue, culminating in the Computing TDR in two years time. Many improvements and optimizations are expected in the offline software which will be directly transferred to the associated HLT components. The estimates presented above reflect the present performance of the ESS and the offline.

14.4 The 10% testbed

In order to study the combined performance of the components and subsystems of the ATLAS Data Flow system, a testbed with full Data Flow functionality and a size of approximately 10% of the final system has been assembled. Although the testbed is necessarily a scaled down ver-

sion of the final system, individual components are operated at rates similar to those expected in the final system.

The primary aim of the 10% testbed is to demonstrate the full and concurrent execution of the RoI data collection and Event Building functionality to check for possible interference between them, for instance, in the form of reduced performance. Measurements will also be performed on the test bed to study outstanding design and implementation issues, for example bus-based and switch-based readout and of the number of central switches. In addition measurements performed on the testbed are being used to validate and calibrate computer models of the system. The subsequent reproduction of the performance and scaling behaviour of the 10% testbed by modelling will strengthen conclusions drawn from modelling studies of full-size HLT/DAQ system.

As the 10% testbed has only been assembled and commissioned in the weeks preceding the submission of this report, only the results of preliminary measurements are reported here.

14.4.1 Description of the 10% testbed

The 10% testbed presently consists of a set of PCs and custom hardware devices, used to emulate ROSs, inter-connected via a Gigabit Ethernet network. The testbed, shown in Figure 14-4, reflects the architecture of the final system. It implements two, separate, central switches for RoI data collection and event building and allows for additional central switches to be added for the studies of scalability, for example two RoI collection switches and two event building switches. In addition, the testbed is such that two methods of accessing data buffered in ROBs may be studied the: bus-based ROS and switched-based ROS, see Section 5.5.4.

14.4.1.1 Readout subsystems in the 10% testbed

In the testbed the bus-based ROS, as described in Section 8.1.3.3, has been implemented on PCs (numbers 108, 114-116 in Figure 14-4). Each PC is equipped with two Gigabit Ethernet NICs connecting the ROS to each of the central switches. As described in Section 8.1.3.3, this version of the bus-based ROS emulates the interactions with the ROBins as the prototype ROBin (see Section 8.1.3.2) is as yet not installed. Additional bus-based ROSs are emulated in the testbed by sixteen programmable Alteon Gigabit Ethernet NICs (see labels ALTx in Figure 14-4). The Alteon NIC has only a single Gigabit Ethernet port, i.e. they cannot be simultaneously connected to both central switches. To overcome this limitation, in the testbed, a bus-based ROS is emulated by two Alteons, one connected to the LVL2 central switch and the other connected to the event building central switch, allowing the emulation of eight bus-based ROSs. Together the PCs and the Alteons provide twelve bus-based ROSs which is 10% of the number foreseen in the final system.

In the switched-based ROS scenario, each ROBin has its own connection to the central switches via a concentrating switch. Within the testbed two different types of emulators of ROBin are deployed, the FPGA ROBin emulators and the Alteon NICs. The FPGA ROBin emulators (FPGA #1 – FPGA #4 in Figure 14-4) has a single Fast Ethernet link to a concentrating switch (labelled T5C-GF in Figure 14-4). Thirty-two FPGA emulators are connected to a concentrating switch and each concentrating switch has two Gigabit Ethernet links, one to each of the central switches. There are 128 FPGA ROBin emulators. The Alteon NICs are as described above.

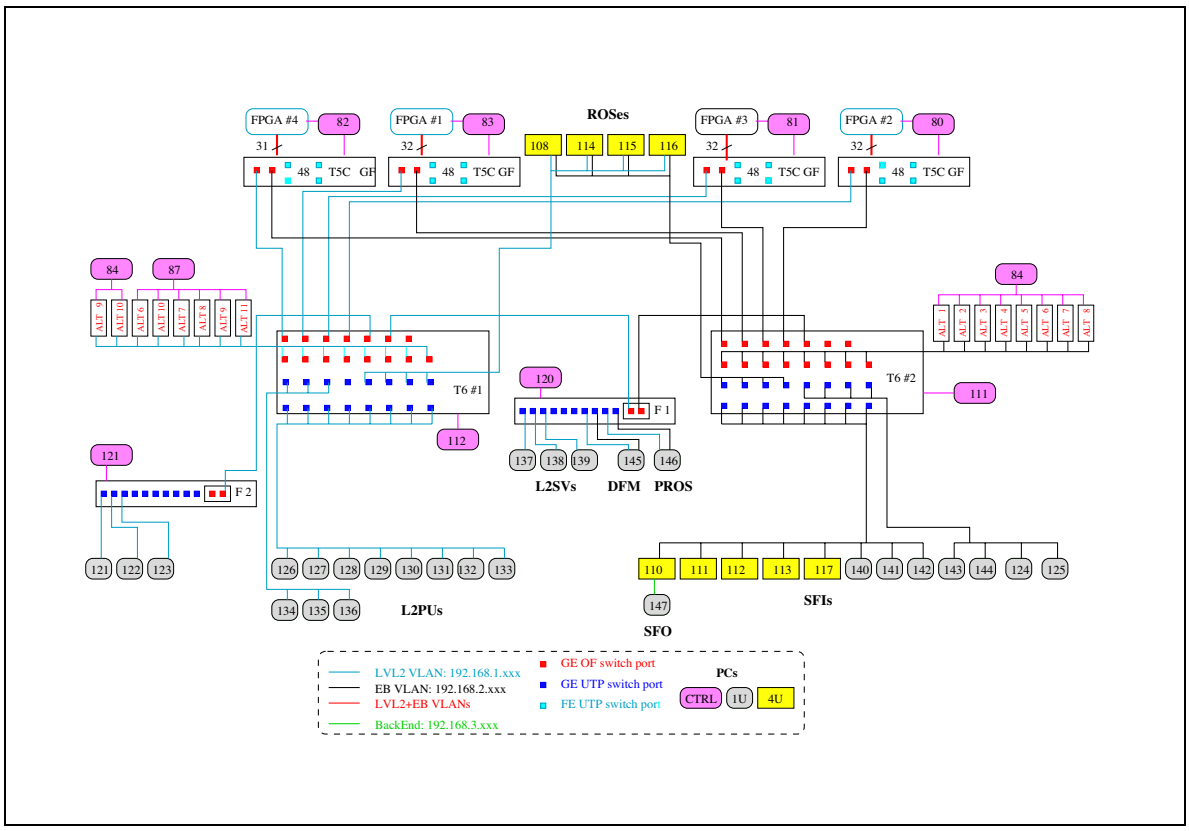

Figure 14-4 Organization of the 10% testbed

Each FPGA ROBin emulator is limited to sending a maximum message size equal to a single Ethernet frame, i.e. they can only emulate a ROBin with a single ROB. However, they will respond to any number of requests for data and used in this way in the testbed they can emulate 1600 ROBins. As the Alteon NIC is programmable, they too can respond to any number of requests for data.

For some system performance measurements, the performance of the Alteon NICs limited the performance of the system, particular, when these devices were used to emulate a ROBin with more than one ROB or a ROS with more than one ROBin. In the case of the FPGA ROBin emulators, the output bandwidth of the concentrating switch (~105 Mbyte/s) limits the obtainable rates particularly when they are used to emulate 1600 ROBins.

14.4.1.2 RoI data collection in the 10% testbed

Referring to Figure 14-4, the LVL2 trigger consists of a central switch (T6#1), fourteen L2Ps, three L2SVs and a pROS. The central switch is a 32-port Gigabit Ethernet switch consisting of sixteen optical ports and sixteen electrical (UTP) ports. The fibre ports are used to connect the ROBin emulators and for connecting to a grouping switch (described below). The UTP ports are used to connect the: PCs used for the bus-based ROS and eleven L2Ps. The L2Ps, of which there are presently eighteen, are 1U rack-mounted 2.2 GHz and 2.4 GHz dual-processor Xeon machines and are connected within the testbed in two ways. Some nodes (PCs 126-136 in Figure 14-4) are connected directly to the LVL2 central switch, while other nodes (PCs 121-123) connect to the LVL2 central switch via a grouping switch (F2 in Figure 14-4). The grouping switch is a Gigabit Ethernet switch with ten UTP ports and two optical ports. The final system has all L2Ps connected to the LVL2 central switch via grouping switches. In the testbed a mix-

ture has been used to understand possible effects of having grouping switches. In the testbed there are fourteen L2Ps this is less than the 10% of the final system but as the L2PUs will not be executing algorithms (at least in initial measurements) each L2PU will collect RoI data at rates beyond what is required of them in the final system, thus the fourteen L2PUs effectively emulate more than 10% of the final L2PUs foreseen in the final system.

14.4.1.3 Event building in the 10% testbed

The event building consists of a central switch, twelve SFIs and a DFM. The central switch (T6 #2 in Figure 14-4) is a 32-port Gigabit Ethernet switch identical to the LVL2 central switch. Twelve of its sixteen optical ports are used to connect the ROBin emulators. Twelve of its sixteen UTP ports are used to connect SFIs (PCs 110-113, 117, 140-144 and 124-125 in Figure 14-4). Other UTP ports are used to connect the four PCs used for the bus-based ROS and to connect the DFM and pROS via a grouping switch (F1 in Figure 14-4). The nine SFIs in the testbed correspond to 10% of the number foreseen in the final system.

14.4.1.4 Simultaneous RoI collection and event building in the 10% testbed

For the measurements consisting of concurrent RoI data collection and event building on operation of the two subsystems is coordinated via back pressure from the DFM to the L2SVs. The L2SVs on the testbed emulates the LVL1 trigger by generating event processing requests to be sent to the L2PU applications on the L2Ps at rates as high as allowed by the back pressure. The DFM maintains a queue of events accepted by LVL2 and awaiting assignment to an SFI. When this queue is full, a message is sent to the L2SV to signal that further sending of requests to L2PUs should be discontinued. When the occupancy of the queue drops below 80%, the DFM sends another message to the L2SVs signalling that sending additional requests can be resumed. If the queues for event processing requests in the L2PUs fill, they also can assert back pressure to throttle the rate with which the L2SV generates events.

The network topology of the final system has Ethernet loops between the two central switches and the ROBin concentrating switches (see Section 8.3.1.2.2) and the Spanning Tree algorithm (STP) or VLANs will be used to ensure a loop free topology. However, the switches used in the testbed do not allow STP to be applied on a VLAN basis and the Message Passing layer does not support VLAN tags. Hence network loops have been avoided by avoid network loops the DFM and pROS (whose communicate across VLANs) were each equipped with two network interface cards, one each to connect to the LVL2 VLAN and the other to the EB VLAN.

14.4.2 Preliminary results of the 10% testbed

This section presents the preliminary results of initial measurements performed on the fully functional testbed, i.e. concurrent RoI data collection and event building. The results of the system performance using a bus-based readout are shown in Figure 14-5 and Figure 14-6, while the system results using a switch-based readout are shown Figure 14-7.

In Figure 14-5, the sustained event building rate is plotted versus the number of L2PUs used in the testbed for different LVL2 accept fractions. In this set of measurements there were eight SFIs and four bus-based ROSs each emulating the support of 12 ROLs. Each L2PU had two worker threads requesting RoI data, for different events, of size 1.5 kbyte from a single ROL per event. The size of the event be built is 48 kbyte. the event building rate is higher than it would be in the

full system because of the smaller number of ROSs (four instead of ~130), even considering the smaller number of SFIs (eight instead of ~90).

For a LVL2 accept fraction of 1% the sustained event building rate increases linearly with the number of L2PUs and an event building rate of 0.5 kHz is achieved. For this accept fraction the rate is limited by the number of the L2PUs in the testbed. As the LVL2 accept fraction increases the event building rate also increases but with increasingly non-linearly with the number of L2PUs. The sustained event building rate reaches a maximum of 5.7 kHz corresponding to each ROS driving its link to the event building at ~68 Mbyte/s. Note that in the final system, LVL2 accept rates of approximately 3% are expected.

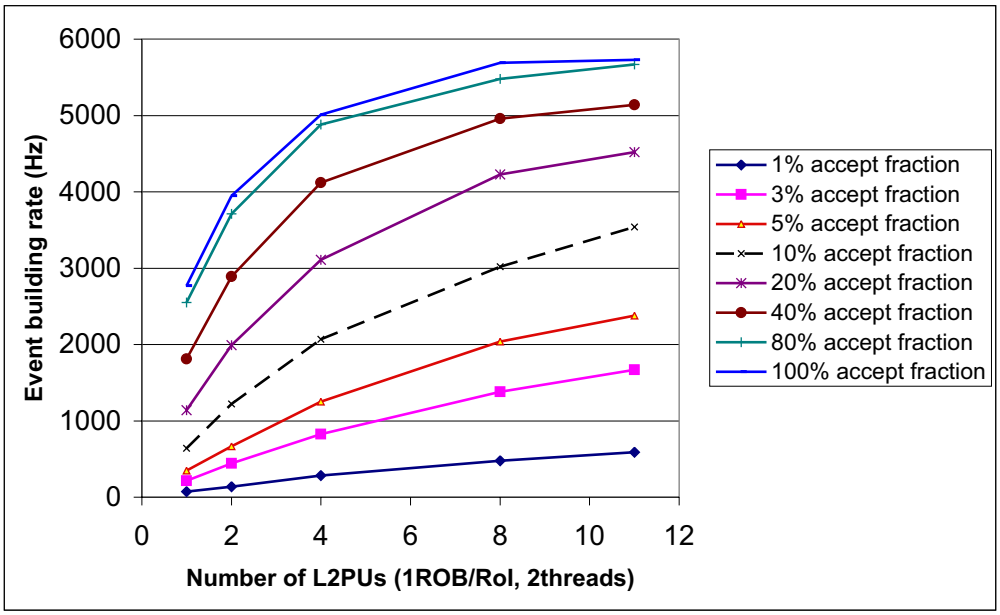

Figure 14-5 Event building rate for simultaneous RoI handling and event building in the 10% testbed. 4 ROS units, each servicing 12 ROLs, were emulated with 4 PCs. Per event the L2PUs requested data as would be received via one of the ROLs associated with one of the emulated ROS units. 8 SFIS were requesting the data from all 12 emulated ROLs of each ROS unit for the accept fractions indicated

The results of similar measurements are shown in Figure 14-6, however, for this set of measurements twelve bus based ROSs were used, four based on the PCs and eight based on the Alteon emulators. The data show that for LVL2 accept fractions of at least 10% the sustained event building rate reaches a 3 kHz limit with only four L2PUs. Similar measurements performed on the testbed but only for event building show that the limit of 3 kHz is due to the limited number of SFIs used and that with ten SFIs a limit of 4 kHz would be reached which is imposed by the limitations of the Alteon emulators. It should be noted however that the 3 kHz event building rate achieved is ~10% lower compared to event building alone. At the time of writing, it is still to be established whether this is due to extra load on the ROS due to the process of RoI data collection. Note that for these measurements, the number of ROSs and SFIs is 10% of those expect-

ed in the final system for a LVL2 accept rate of ~3% and the sustained event building rate achieves the design value of 3 kHz for this accept rate.

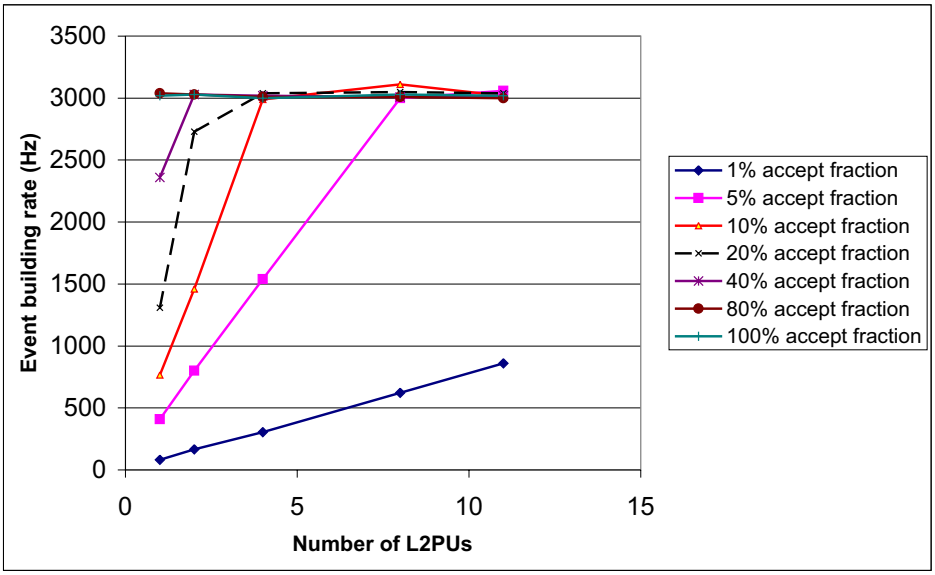

Figure 14-6 Event building rate for simultaneous RoI handling and event building in the 10% testbed. 12 ROS units, each servicing 12 ROLs, were emulated with 4 PCs and 16 Alteon NICs (two per ROS unit, one supplying LVL2 data, the other EB data). Per event the L2PUs requested data as would be received via one of the ROLs associated with one of the emulated ROS units. 8 SFIs were requesting the data from all 12 emulated ROLs of each ROS unit. for the accept fractions indicated

In Figure 14-7 the preliminary results of initial measurements performed on the testbed with a switch-based ROS are shown. The measurements were performed with both types of ROBin emulators (FPGA based and the Alteon). In one set of measurements the 125 FPGA ROBin emulators were used to emulate 1600 ROBs, in a second set of measurement the Alteon NICs were used to emulate 1600 ROBs. In the third set of measurements the 1000 ROBins are emulated by the 125 FPGA ROBin emulators and the Alteon NICs emulate 600 ROBins.

The figure shows the sustained event building rate versus the number of L2PUs, for a LVL2 accept fraction of 3% and eight SFIs performing event building. Ethernet flow control was on. In the measurements where a single type of ROBin emulator was used the event building rate is limited, in the case of the FPGA emulator, the link bandwidth connecting the ROS concentrating switches to the event building central switch. In the case of the measurements performed with the Alteons, the achieved event building rate is limited by the performance of the Alteon NIC. The set of measurements obtained by the joint use of the FPGA and Alteon ROBin emulators reaches the limit imposed by the use of only eight SFIs. Similarly to the results obtained with the bus-based ROS, the sustained event building performance is ~14% lower than in the case of event building alone. This cause of this reduction remains to be understood.

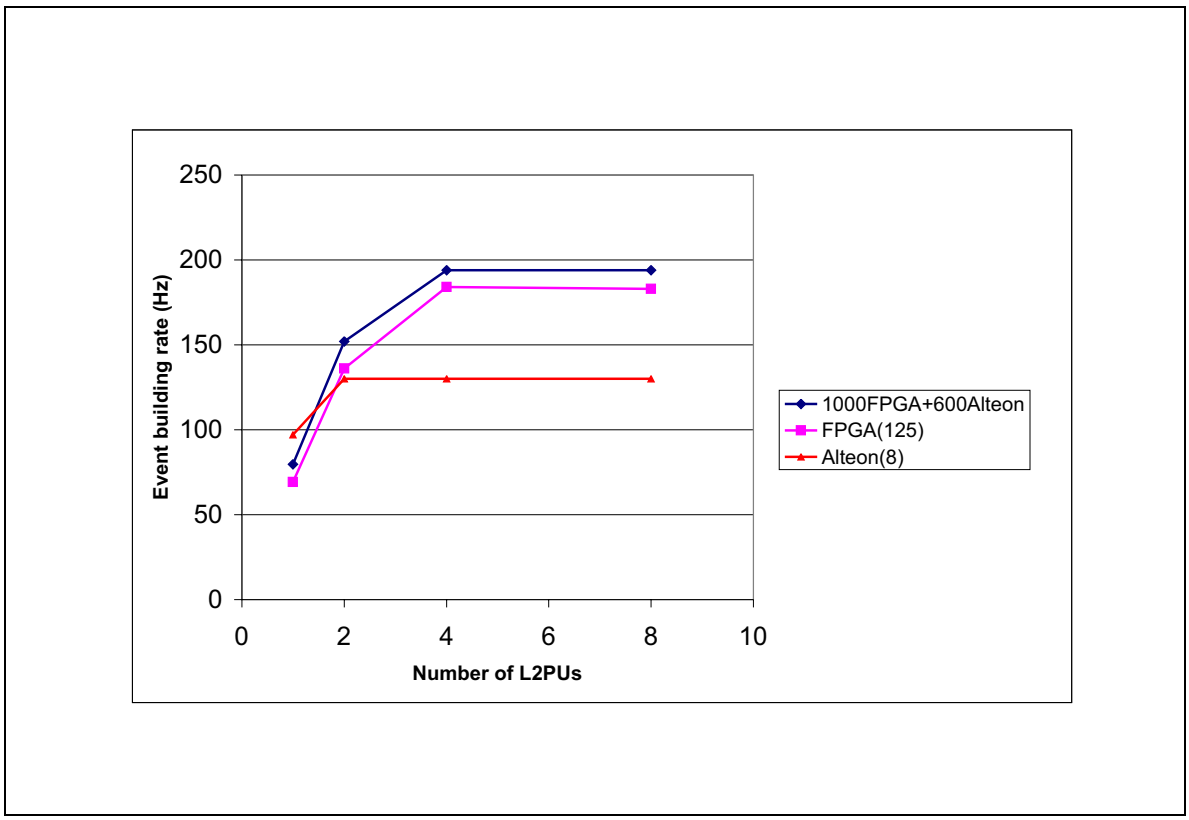

Figure 14-7 Event building rate for simultaneous RoI handling and event building in the 10% testbed for three different ways of emulating 1600 ROBs, as explained in the text. 8 SFIs were requesting the data from all ROBs for an accept fraction of 3%. Ethernet flow control was switched on

The measurements presented here represent only the results from an initial set of studies. Analysis of the measurements presented above, along with modelling of the testbed configurations used for the measurements, are still in progress at the time of submission of this report. Further Studies of the 10% testbed will continue and it is expected to replace the emulators used for the initial set of measurements by the prototype ROBin thus eliminating some of the limitations of the current testbed.

14.5 Functional tests and test beam

Often, during prototyping, more weight is put on the performance of a system than on its stability and maintainability. Functional user requirements tend to have a lower priority than the achievement of the performance requirements during this phase of development. This is to some extent true also for the ATLAS HLT/DAQ system. Nevertheless, by carrying out a series of functional tests and exposing the system to non-expert users at the ATLAS test beam sites, these issues have been addressed.

Four different aspects of the global functionality have been covered: system configuration, stability in cycling through the control states of the system, monitoring and fault tolerance. These aspects have first been tested in dedicated laboratory setups and then verified in a ‘real’ environment, during test beam data taking.

14.5.1 System configuration

A data acquisition system must be easily reconfigurable in order to accommodate the substitution of hardware, the change of trigger conditions, etc. Tools for storing/retrieving the configuration parameters into/from a database must be available, and Run Control, Data Flow, and Trigger software must be designed to be dynamically reconfigurable.

In test beams, data taking configurations tend to change very often, because of the addition of new components and sub-systems that need stand-alone as well as integrated debugging in the read-out chain. In order to ease integration of detector read-out chains in the Data Flow system, nested partitions were introduced. Nested partitions allow different groups to develop their databases independently and then to link them together under a common top partition. The success of this technique was recently proven, especially during beam tests of the Muon detectors. For these tests, five different ROD or ROD emulator crates were independently setup and tested, before being connected for further debugging to their corresponding ROS units and finally also to the event building system.

Although the configuration of the HLT/DAQ hardware and of the software applications are specified at the beginning of a data taking session, a number of configuration parameters for various applications can be changed dynamically, e.g. amount of memory to be reserved for the data, length of queues, etc. Some very dynamic parameters which change for every run, e.g. the run number and calibration variables, are not kept in the configuration database and are distributed via the Information Service provided by the Online Software (see Chapter 10) at the start of the run. Mechanisms for dynamic database changes, with corresponding mechanisms to notify the affected applications are being studied.

14.5.2 Finite state machine transitions

When performing a series of measurements with different configuration options, the HLT/DAQ system must be capable of cycling through a finite set of states in a stable fashion. This capability has been checked with the help of automated scripts cycling repeatedly through the finite state machine associated with the states. The absence of problems has been verified during all testbeam periods.

14.5.3 Monitoring

In a distributed system such as the HLT/DAQ system it is important to monitor the operation of the system continuously. All applications regularly publish statistics on their performance, as well as on the occurrence of errors, via the Information Service. Furthermore, the ROS and the SFI are capable of providing event data for physics monitoring. Operational monitoring has been intensively used to conduct all the performance measurements described in the previous chapters. All aspects of the monitoring facilities have been regularly used by the people on shift during testbeam data taking.

14.5.4 Fault tolerance

The HLT/DAQ system needs to be fault tolerant. Additional work is needed in this area, therefore it was not considered to be appropriate yet to conduct a series of systematic tests in order to

assess the performance of the system in case of errors. In general, an error that is classified as *WARNING* does not cause any disruption in the system, except for the possible loss of some event data. Examples of such errors have been observed in the testbeam setup, for instance when a ROD does not send consecutive LVL1 event identifiers (L1IDs) to the ROS, or when an SFI does not receive the requested data of an event in a time-out period. However, a *FATAL* error in one application, which prevents it from continuing to take data has a potentially serious effect on the overall HLT/DAQ system. For instance, if the system is unable recover from failure of a component that is unique and necessary for data taking, such as the DFM or the RoI Builder. By design, the system can recover from the failure of one or more L2PUs or SFIs, because the L2SV or DFM, respectively, can dynamically mask the failing component. The failure of a ROS, in contrast, requires dynamic re-configuration of the downstream data-taking chain which is not possible at the time of submission of this report. Similarly the mechanism to dynamically mask a single ROL in a ROS is not in place.

14.5.5 Conclusions and outlook

The functional performance of the HLT/DAQ system has been tested during the development of the system and during its exploitation in a testbeds and testbeams. The functionality today is already adequate for successful application of the present prototype implementation in testbeam setups and for carrying out performance measurements. Further development is necessary with respect to the dynamic re-configuration of the system during data taking, in particular in the case where re-configuration is required because of changes in the run conditions on the occurrence of errors. It is essential that components that are not unique can be dynamically excluded from the running system if required. Re-insertion of such components without stopping data taking, may be difficult due to synchronization issues, but the possibility will be studied further.

14.6 Modelling results

14.6.1 Paper model

Estimates of average message frequencies, data volumes and the amount of processing power required for the HLT/DAQ system have been made with the help of a ‘paper model’. The most important results have been presented in Chapter 2. A description of this model and its results can be found in Appendix A.

14.6.2 Computer model

The availability of network connections and switches with sufficient bandwidth and of a sufficient amount of computing resources in the DAQ and HLT systems is not sufficient to guarantee that the performance requirements are met. Also necessary are:

1. an even distribution of the computing load over the available computing resources
2. minimal congestion and large enough data buffers in switches
3. sufficient spare processor capacity and network bandwidth to cope with fluctuations

To verify that these conditions are met, the dynamic behaviour of the full system needs to be studied with the help of simulation with a 'computer model', as the construction of a full scale testbed is not feasible. Computer models therefore have been developed to obtain information on basic and fundamental properties such as the achievable throughput, distributions of the LVL2 decision time and of the event building time, queue development in various places in the system (switches and end-nodes), and to study the impact of various traffic shaping and load balancing schemes.

The type of simulation used for the computer models is known as 'discrete event simulation'. The simulation program maintains a time-ordered list of 'events', i.e. points in time at which the simulated system changes state in a way implied by the type of 'event' which has occurred. Only at the time of occurrence of an event is the modelled system allowed to change its state. In most cases only a small part of the state of the simulated system needs to be updated. The state change can result in the generation of new events at a later time, which are entered at the correct position in the time-ordered list. The simulation program executes a loop in which the earliest event is fetched from the event list and subsequently handled.

The model of the HLT/DAQ system implemented in the simulation programs is an object-oriented model, in which most objects represent hardware (e.g. switches, computer links, processing nodes), software (e.g. the operating system, Data Flow applications), or data items. Two simulation programs have been used, the at2sim program [14-15] and the Simdaq program [14-16]. The at2sim program makes use of the general purpose simulation environment of the Ptolemy system[14-17]. Ptolemy offers support for discrete event simulation and allows the implementation of object-oriented models. The Simdaq program is a dedicated C++ program, with the discrete event simulation mechanism being a part of the program.

The component models used in the at2sim program are models of the testbed components described in Section 14.4. They were kept as simple as possible, but sufficiently detailed to reproduce the aspects of their behaviour relevant for the issues studied. Parameterized models of all Data Collection applications [14-18] and Ethernet switches [14-19] have been developed. Computer models of small test set-ups have been developed and have been used for characterizing the behaviour of system components. Also models of testbeds and of the full system have been developed. For the calibration of the models of the Data Collection applications, time stamps were obtained with the help of code added to the Data Collection software (for this purpose a library based on access to the CPU registers was developed). The time stamps provided estimates on the time spent in various parts of the applications. The calibration obtained in this way was cross-checked with results from measurements performed in specialized setups with the application tested running at maximum rate. Parameterized models of the switches were obtained with the help of dedicated setups. In these setups use was made of hardware traffic generators. The aim was to find possible limitations in the switches which may affect the performance required for the full HLT/DAQ system. The process of identification of appropriate models and a corresponding set of parameters and of collection of the parameter values with the help of dedicated setups was iterative and interleaved with validation phases. In the validation phases larger setups were modelled. Discrepancies between results from modelling and from measurements usually gave rise to a modification in the model(s) and associated parameters and another calibration phase.

The component models in the Simdaq program are less specific than the models used in at2sim. The current version of the program can be seen as a dynamic version of the paper model. The program makes use of the same information concerning LVL1 trigger menus, mapping of the detector onto the ROLs, sizes of fragments associated with ROLs, execution times of LVL2 processing steps and associated reduction factors, etc. as the paper model. The eta and phi coor-

dinates of the RoIs generated according to the LVL1 trigger menu are chosen at random from the possible RoI positions as defined by the LVL1 trigger (the probability of choosing a certain position is determined by the size of the area in eta-phi space associated with the position). Results for average rates should be the same (within the statistical errors) for both models, while computer model results for the computing resources actually used, taking into account their utilization, should be equal to the paper model results for the computing resource requirements. Good agreement between results from the paper model and Simdaq has been achieved, and the results of both models have been checked for consistency.

14.6.2.1 Results of testbed models

The results of the measurements focusing on the scalability of the Event Builder have been compared to computer model predictions. Three different setups with homo- or heterogeneous sets of ROB emulators have been modelled. In Figure 14-8 a comparison between results from measurements and model predictions for the event building rates as a function of the number of SFIs collecting data is presented. Three setups were investigated: a setup with 125 FPGA ROB emulators (13 ROBs per FPGA), a setup with 8 Alteon ROB emulators (200 ROBs per Alteon) and a setup with a mixture of the two types of emulators: 1000 ROBs were emulated by 125 FPGA emulators (8 ROBs per FPGA) and the remaining 600 ROBs were emulated by 8 Alteons (75 ROBs per Alteon). A set of modelling results is associated to each set of testbed measurement results. The three setups showed different saturation rates due to either the performance of the emulators or due to the arrangement of the setup. For a small number of SFIs, the event building rate increases by 30 Hz each time a new SFI is added to the system (this is the maximal building rate of a single SFI). The lowest maximum event building rate is observed for the setup with 8 Alteons emulating 75 ROBs each. The internal processing time of the Alteon NICs for the incoming messages is 40 µs and as each emulator has to process 200 requests for an event, the upper limit for the rate is 125 Hz. In the setup with the FPGA emulators, the rate limitation is caused by the throughput of the Gigabit Ethernet up-links connecting the concentrating switches to the EB central switch. With 2.2 Mbyte of event data spread uniformly over the FPGA emulators connected via four concentrating switches, each up-link has to deliver 2.2 Mbyte / 4 = 0.55 Mbyte of data. Assuming the maximum payload which can be transferred in 1300 byte packets over the Gigabit Ethernet to be 105 Mbyte/s, the event building rate is limited to 191 Hz. The highest rate can be observed in the setup using a mixture of FPGA and Alteon emulators. The rate limit is determined again by the throughput of the Gigabit up-links between the concentrating switches to which the FPGA emulators connect and the EB central switch. In this setup 1000 out of 1600 ROBs are emulated by the FPGA devices producing 1000/1600 * 2.2 Mbyte = 1.375 Mbyte of event data. This data is spread over the FPGA emulators attached to four concentrating switches and requires that 0.344 Mbyte will be sent per event over the up-link. This limits the rate to 305 Hz. In this setup the limit due to the Alteon NICs is higher, as each emulates 75 ROBs and with 40 µs of processing time for each request the limit would be 333 Hz. In the setup with a mixture of ROB emulators, the rate does not scale linearly with the number of SFIs if this number is larger than four and below saturation of the event building rate. This is due to queuing of packets heading for the up-link in the concentrating switch, which is very sensitive to the traffic shaping (or lack of it). Figure 14-8 shows very good agreement between the results from measurements and predictions from modelling. These results validate the calibration of the model components (switches, SFIs, emulators).

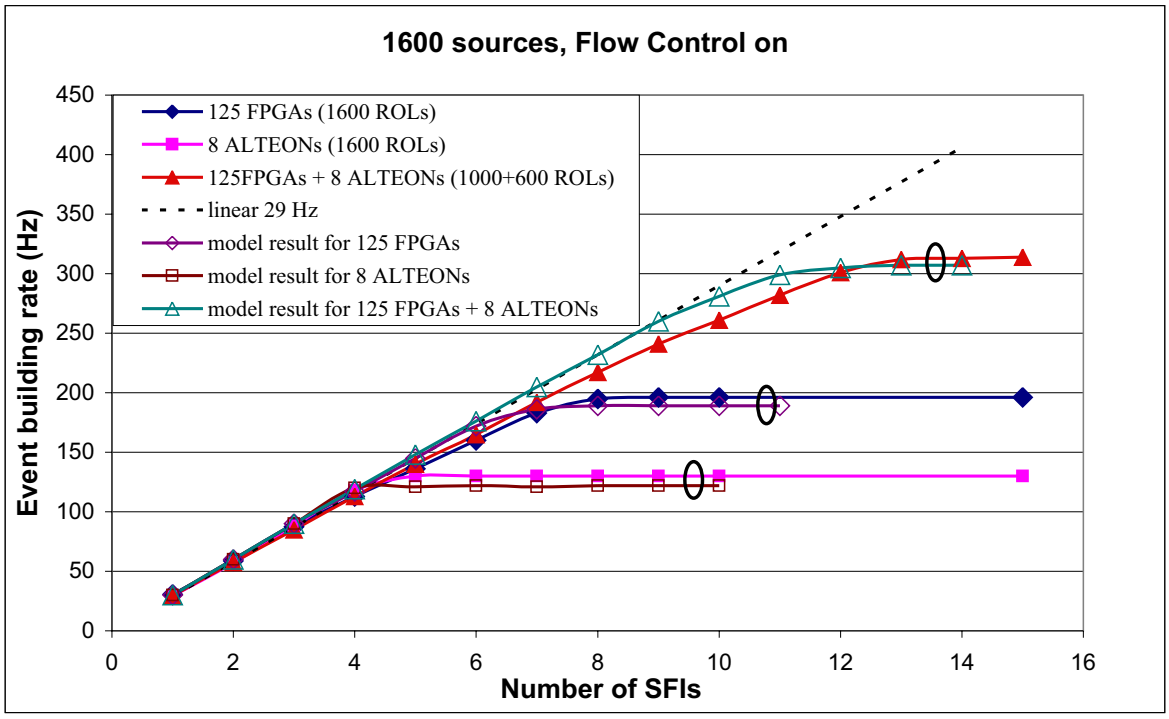

Figure 14-8 Comparison of measurement results and computer model results for event building only, without LVL2 traffic in the testbed. Ethernet flow control was switched on.

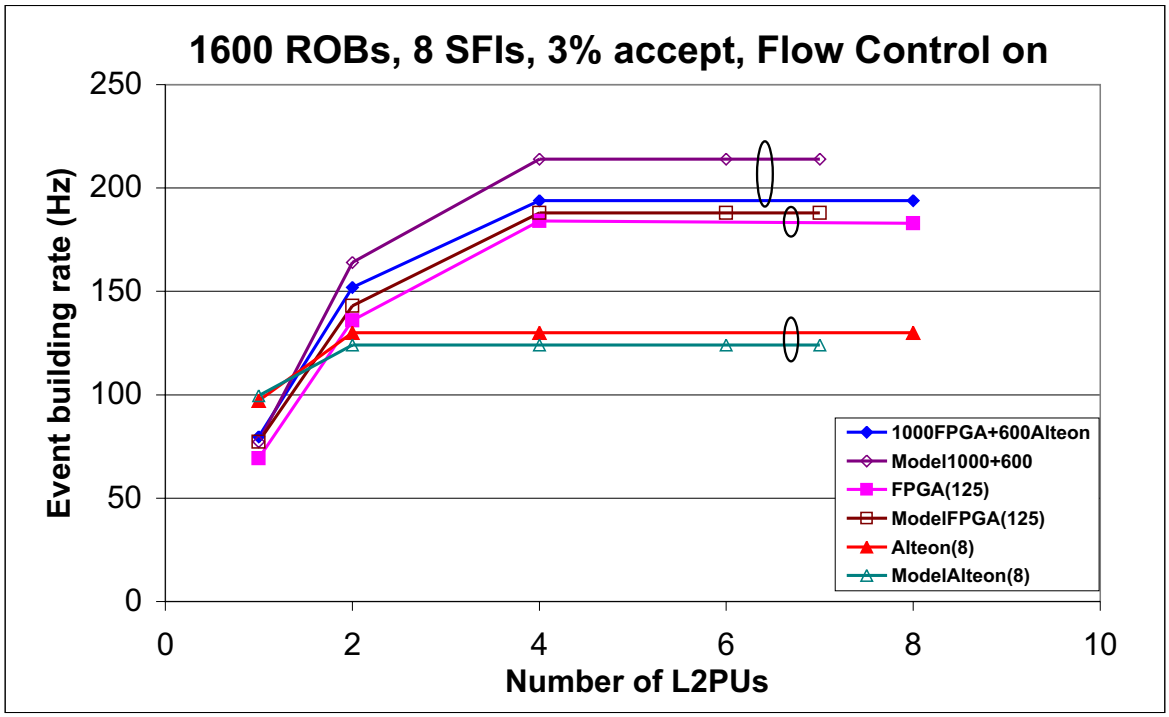

Figure 14-9 Comparison of measurement and modelling results for the event building rate for simultaneous RoI handling and event building in the 10% testbed for three different ways of emulating 1600 ROBs. Ethernet flow control was switched on.

In Figure 14-9 a comparison is made between measurement results obtained with the testbed for simultaneous RoI handling and event building and modelling results. Again the three set-ups were used for obtaining the results presented in Figure 14-8. The EB subsystem was receiving 3% of the processed events accepted by the L2PUs and for those events 8 SFIs were used to collect data from the ROBs and to build events. Very good agreement has been obtained for the setups with either Alteons or FPGAs ROB emulators. In both cases, the event building rate saturates at the same level as for the EB scalability tests. Less good agreement, however still within 10% tolerance, has been reached for the setup with a heterogeneous set of ROB emulators. For this setup the value at which the event building rate saturates is determined by the performance of the 8 SFIs.

14.6.2.2 Results of extrapolation of the at2sim testbed model

The full system model chosen to be implemented in at2sim has 1654 ROBins, each servicing a single ROL and each having an individual network connection [14-13]. The LVL2 subsystem has been assumed to be composed of 180 L2PUs connected to two Gigabit Ethernet LVL2 central switches in two groups of 90. The L2PUs are connected to the central switches via Gigabit Ethernet L2PU grouping switches with 7 L2PUs attached to the same switch. The EB subsystem is composed of 80 SFIs connected in two groups of 40 to two Gigabit Ethernet EB central switches. The SFIs are connected directly to the EB central switches. The L2SV DFM, and pROS are connected via a dedicated small Gigabit Ethernet switch to the four central switches. The ROBins are connected in groups to concentrating switches, each having four up-links to the central switches. The number of concentrating switches per sub-detector depends on the average size of the event fragments from a given sub-detector (for results of calculations see Ref. [14-13]). In total there were 46 concentrating switches. Results were obtained for the following configurations:

1. ***Individual ROBin Fast Ethernet***: each ROBin has a single Fast Ethernet connection to a concentrating switch, this reflects the configuration in the 10% testbed with the BATM T5Compact switch,
2. ***Individual ROBin Gigabit Ethernet***: each ROBin has a single Gigabit Ethernet connection to a concentrating switch (thus the concentrating switch becomes an all-Gigabit switch),
3. ***Two, four or six ROBins aggregate***: two, four or six ROBins are assumed to share a single network connection, a single request produces a response with a size two, four or six times larger than the response of a single ROBin, the number of ROBins connected to a concentrating switch is a factor of two, four or six smaller than for individually connected ROBins.

The traffic generated in the model resembles the traffic in the 75 kHz system: the LVL2 subsystem was running at event rate of 75 kHz and the EB subsystem at a rate of ~3 kHz. The L2PUs were making only one processing step — for each event, data from ten randomly chosen ECAL ROBins were requested and a decision was produced and sent to the L2SV. The L2PUs were not calibrated — they were used only to generate the LVL2 traffic in order to obtain a more realistic environment. The SFIs were requesting data in random order from all ROBins or ‘ROBin aggregates’.

The effect of a credit based event building traffic shaping on the event building latency and queue build-up has been investigated with the model. The left plot in Figure 14-10 shows that increasing the number of credits per SFI above ten does not improve the latency for event building except for the ‘individual ROBin Fast Ethernet’ configuration. In the latter configuration, the

latency is still related to the transfer speed of the Fast Ethernet links (12.5 Mbyte/s). Therefore queuing of fragments during their transfer to the SFIs is unlikely and the time needed for building a complete event will depend on the time needed to transfer an event fragment via a Fast Ethernet link. The shorter latency for setups with 'ROBin aggregates' with respect to those with individually accessed ROBins is due to the smaller number of requests to be generated per event. The CPU time for receiving replies scales with the number of frames received. The latter scales approximately with the number of ROBins. The CPU time spent on generating requests however scales with the number of ROBins aggregated. As the EB rate is limited by the SFI performance (the network provides sufficient bandwidth), the smaller amount of time needed for generating requests allows an SFI to process events faster. More quantitatively: the SFI rate when sending requests to the individual ROBins is 30 Hz, i.e. per event 33 ms is spent (the SFIs run at 99% CPU utilization). The 33 ms is spent on the generation of 1600 requests and the reception of 1600 replies. It has been measured that for the reception of a packet 14 µs of CPU time is needed. The reception of 1600 packets will take more than 22 ms. The remaining 11 ms is used to produce 1600 requests. In case of an aggregation factor of 2, 5 ms will be saved, for an aggregation factor of 4 or 6 this will be 8 or 9 ms. Interrupt coalescence, for which the default time-out is 65 µs, also plays a role in the speed-up. During 65 µs only a few requests can be generated and consequently, only a few replies will arrive. Thus the gain in processing time due to the use of interrupt coalescence is limited - one interrupt is generated for a few packets received. In the case of aggregation a single request will produce 2 - 6 replies, depending on the aggregation factor. Therefore considerably more packets can be handled per interrupt and the time spent per packet received is smaller. This in turn leads to a reduction of the time needed for event building.

A possible consequence of the aggregation of ROBins consists of overflow of the queues in the switches. The right plot in Figure 14-10 shows the maximum length of the queues associated with the ports in the EB central switches connecting to the SFIs. In case of aggregation, up to six packets of replies (in the six ROBin aggregate scenario) can be returned for a single request, so the maximum queue length (measured in number of packets queued) can be expected to be equal to 6 times the number of credits. Therefore the number of credits may have to be reduced with respect to the scenarios where each ROBin has an individual network connection, otherwise the queue length may reach a switch buffer limit and give rise to packet loss.

In Figure 14-11 a prediction is shown for the event building capability of the full HLT/DAQ system as a function of the number of SFIs of the type used in the testbed. The maximum number of credits was set to 30. Flow control was switched on. The buffer size for the output ports in the central switches was set to 160 packet slots, the flow control was not (and should not have been) activated in the central switches. It can be seen that the maximum event building rate scales linearly with the number of SFIs to a rate of 3 kHz for 110 SFIs.

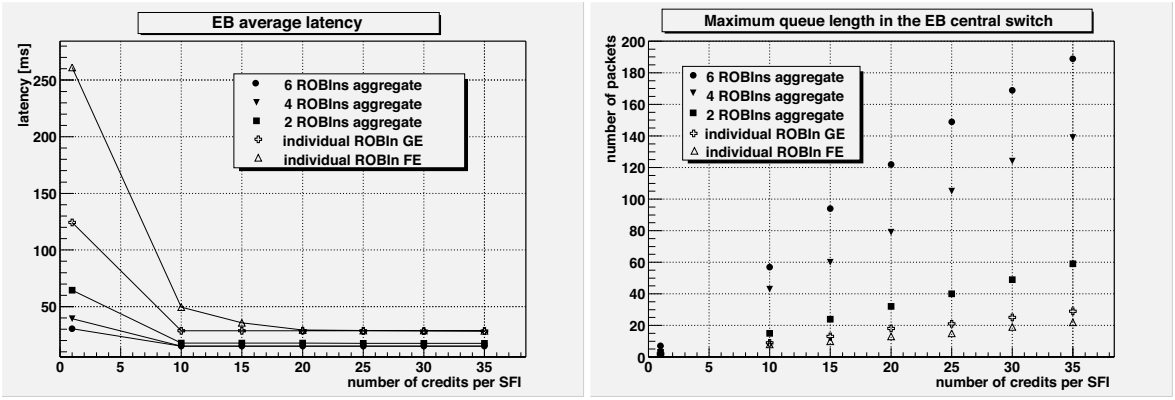

Figure 14-10 Average event building latency and maximum queue length in the EB central switches for different ROBin configurations obtained with the at2sim full system model. Flow control was switched off.

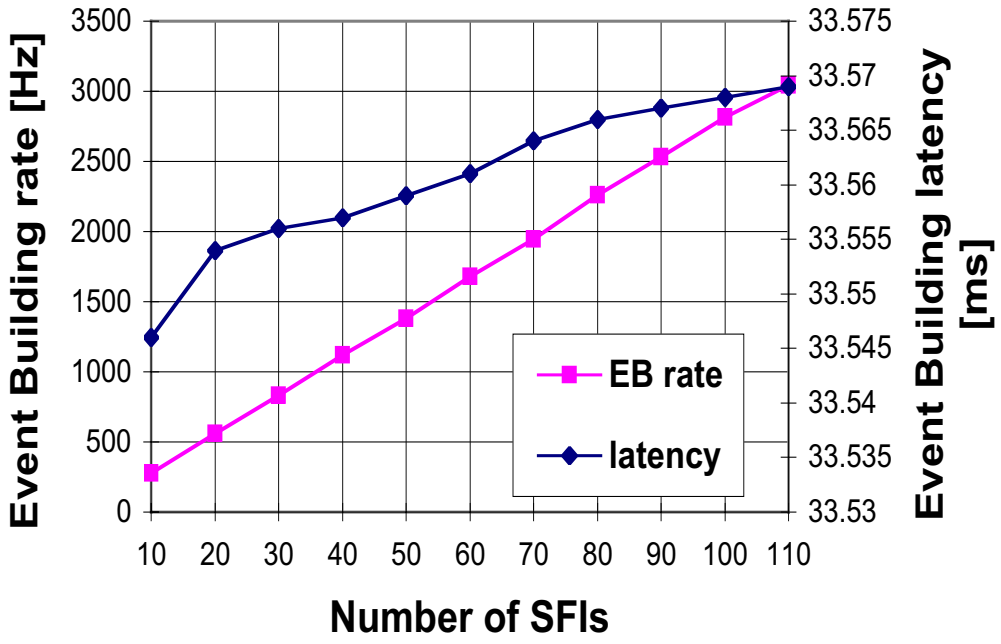

Figure 14-11 Model prediction for the Event Building capability using the calibrated models of the testbed components. The LVL2 accept fraction is 3%.

14.6.2.3 Results of the full system Simdaq model

The results presented in this report for the full system model obtained with Simdaq are all for design luminosity and a LVL1 accept rate of 100 kHz. Although the distributions for decision times and queue sizes are sensitive for the details of the models the general trends are not. The effect of various strategies for minimization of decision times and of queue lengths therefore can be studied with the model implemented in Simdaq, and also apply for low luminosity and lower LVL1 accept rates.

Two different configurations (see Section 5.5.4) have been studied, a ‘bus-based system’, i.e. a system with a bus-based ROS, see Figure 14-12 and a ‘switch-based system’, i.e. a system with a switch-based ROS, see Figure 14-13. In the bus-based system, ROBins are grouped together in groups of three (four ROLs each – 12 ROBs in total) in ROS units, each with two Gigabit Ethernet connections, one to a central LVL2 switch and one to a central EB switch. In the switch-based system ROBins are assumed each to service four ROLs and to be connected via a Gigabit Ethernet connection to a ‘concentrating switch’. In the bus-based system the L2PUs and SFIs can request data from several or all ROLs of a ROS unit, respectively with a single request. The response consists of a single, usually multi-frame, message. The addition of a header by the ROS has not been taken into account in the results presented. A ROB is associated with each ROL in the model of the ROS unit, the maximum numbers of fragments that need to be buffered are output by the simulation program. A single processor in the ROS unit takes care of distributing requests to the ROBins and of collecting and concatenating the responses. In the switch-based system event fragment data associated with different ROLs have to be requested separately. For example, the SFIs have to send four request messages per event to each ROB, and each will respond with four separate response messages (one per request). Again the maximum numbers of fragments that need to be buffered are output by the simulation program.

The L2PUs are dual-CPU machines, each running four processing threads.The model for the L2PU is built around an object managing the scheduling and de-scheduling of the threads on the two CPUs and objects representing the threads. The model allows an arbitrary choice for the number of threads and the number of CPUs. Four L2SV processors each manage a group of 125 L2PUs and four DFM processor each manage a group of 16 SFIs. Each L2PU receives RoI information from and sends decisions to the L2SV controlling the group to which the L2PU belongs. The L2PUs also send LVL2 trigger result data to the pROS (not shown in the figures). One of the DFM processors is associated with each L2SV, the L2SV sends blocks of decisions to it. The DFM processors collect LVL2 rejects and multi-cast them in blocks of 300 clears to the ROS units or ROBins via the central Event Builder switches (and concentrating switches in case of the switch-based system). The DFM translates LVL2 accepts into build requests for the SFIs, in the current model these requests are sent to the SFIs according to a round-robin scheme. Each SFI sends an ‘End-of-Event” message to the DFM controlling the group to which the SFI belongs, after building an event. This is converted to a clear and sent as part of a block of 300 clears to the ROBins or ROS units. The SFIs in this model are single processor machines. Transfer of complete events to the processors of the Event Filter has not been modelled, the events built are discarded by the SFIs.

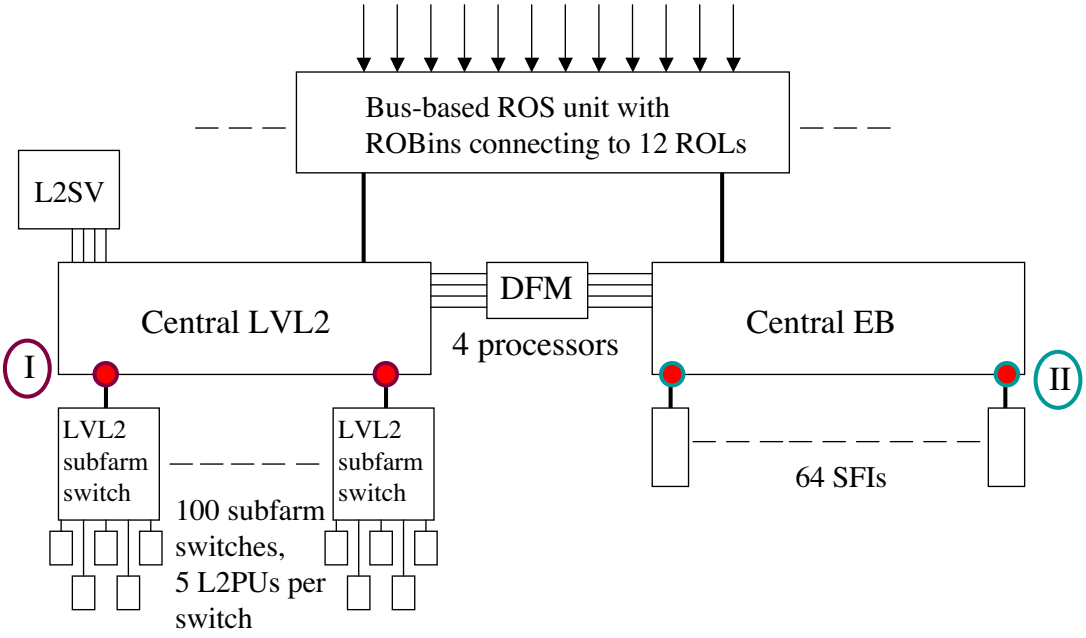

Figure 14-12 Schematic representation of the bus-based system. Possible 'hot spots' are indicated.

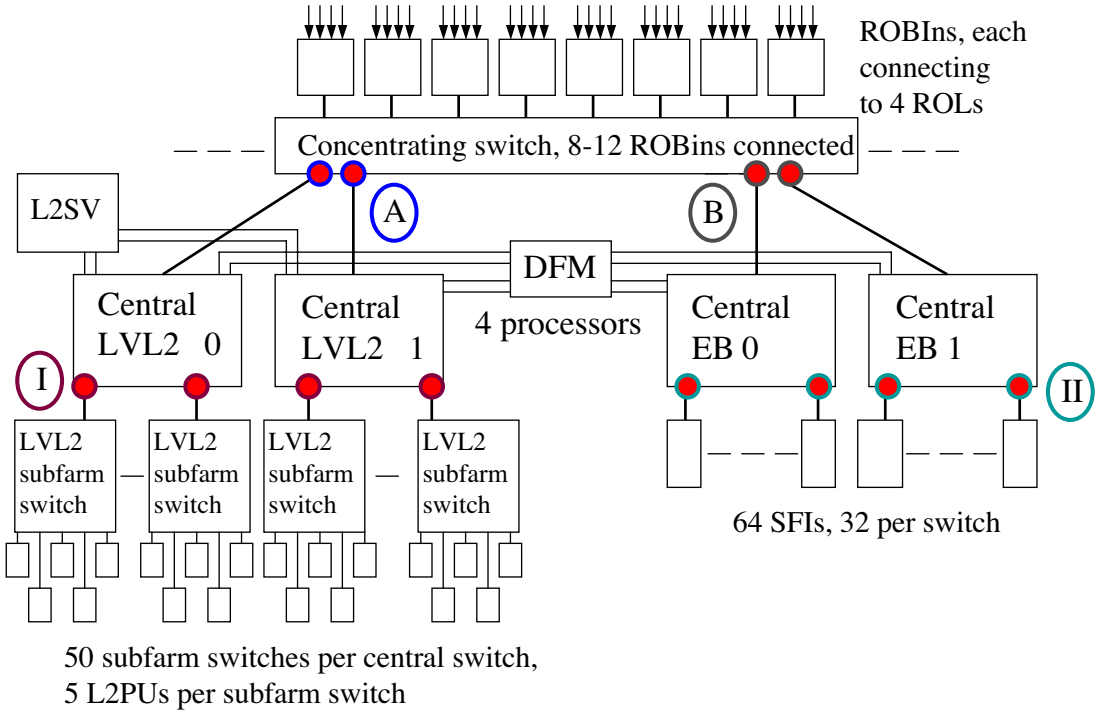

Figure 14-13 Schematic representation of the switch-based system in which ROBins are directly connected to the network. Due to the bandwidth requirements the number of ROBins connected to a single concentrating switch has been set to 8 for the ROBins receiving data from the Pixels and SCT sub-detectors and to 10 for the ROBins receiving data from the TRT and from the LVL1 RODs, for all other sub-detectors 12 ROBins are connected to a single concentrating switch. Possible "hot spots" are indicated.

An even distribution of the computing load can be achieved by means of a suitable strategy for assigning events to the L2PUs or SFIs. For example a simple and effective strategy can be implemented with the help of a record of how many events are being handled by each L2PU or SFI. As the supervisor and DFM are notified when processing is finished such a record can be maintained without an additional exchange of messages. A new event can then be assigned to the L2PU or SFI with the smallest number of events to process ('least-queued assignment'). This is an effective strategy, which makes high average loads of the L2PUs possible. In Figure 14-14, results for the LVL2 decision time (the interval between the time of occurrence of a LVL1 accept and the arrival of the LVL2 decision in the L2SV) are presented for this type of event assignment. Results are also presented for round-robin assignments in combination with assignment of a maximum of four events to the same processor, as well as for a round-robin-only assignment scheme. The peaks in the distribution for least-queued assignment are caused by the various processing steps. For each step a fixed processing time has been assumed, in reality algorithm processing times will depend on the properties of the input data, this will result in less pronounced peaks than found with the model. The average utilization of the L2PUs is here 77%. The tail of the distribution for round-robin assignment does become much longer for a smaller number of L2PUs, i.e. for higher average utilization. Above a utilization of 85 - 90% stable operation of the system is no longer possible, as the amount of processing resources requested is not evenly distributed over the L2PUs. With least-queued assignment stable operation is still possible at this level of utilization.

After each new assignment by the L2SV of an event to one of the L2PUs, the number of events assigned to that L2PU is entered in a histogram. Distributions obtained in this way are presented in Figure 14-15, again for least-queued assignment, round-robin assignment with a maximum of four events assigned to the same L2PU and round-robin assignment only. The least-queued strategy clearly results in a minimum number of events assigned simultaneously to the same L2PU. From the distributions it can be inferred that the choice of four threads per L2PU is appropriate for the system modelled, two threads probably would not be enough, in particular for round-robin assignment.

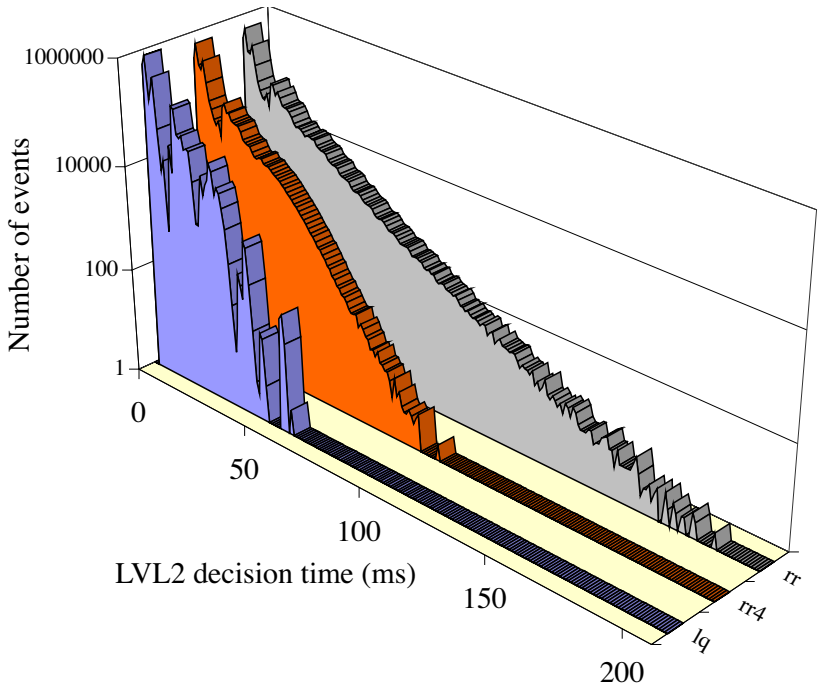

Figure 14-14 LVL2 decision time for the bus-based system, for round-robin assignment (rr), round-robin assignment of at maximum 4 events to the same L2PU (rr4) and least-queued assignment (lq). The results for switch-based read-out are almost identical. The average decision times are 11.9 ms (rr), 10.7 ms (rr4) and 8.8 ms (lq). The maximum number of fragments to be buffered per ROL is about 3100 (rr), 3000(rr4) and 2600(lq), taking into account that deletion of events accepted by the LV2 trigger requires clears from the event building system.

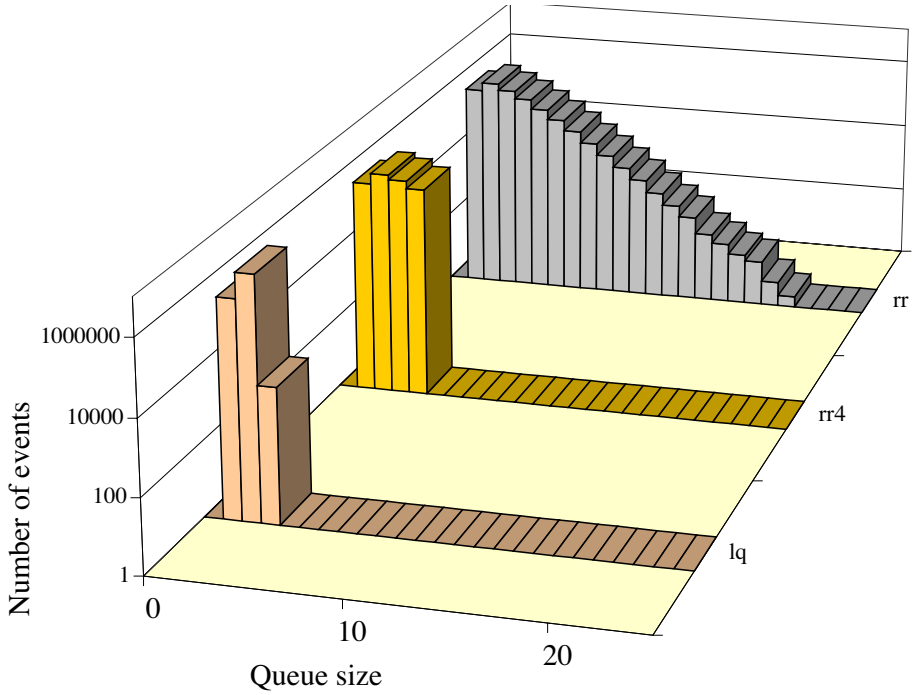

Figure 14-15 L2SV queues for round-robin assignment (rr), round-robin assignment of at maximum 4 events to the same L2PU (rr4) and least-queued assignment. The results for switch-based read-out are almost identical. The averages of the distributions are: 1.8 (lq), 2.4 (rr4) and 2.8 (rr)

With the model the building up of queues can be studied. It is important that the predicted queue lengths do not exceed the available buffer capacity in the switches, as otherwise in reality either packet loss will occur or flow control will be activated. The latter prevents packet loss, but may also cause temporarily blocking of other data transfers. Therefore it is to be preferred to keep the queues short and to prevent long tails in the queue length distributions by using simple, but effective measures. These can consist of requiring the number of outstanding requests in the L2PUs and the SFIs to be smaller than a certain maximum and, for the switch-based system, of choosing a suitable pattern for requesting data by the SFIs, as will be illustrated with a few model results. Less important is the assignment pattern (not to be confused with the assignment strategy) of events to the L2PUs. It should be noted that the results presented in this section are for the case of flow control switched off.

In both the bus-based and switch-based systems, queues tend to form in the output ports of the central switches connecting to the LVL2 subfarm switches (“point I”) and to the SFIs (“point II”). Figure 14-16 shows distributions for the sizes of queues in point I for four different scenarios for assigning events to the L2PUs for the bus-based system. In Figure 14-17 the same results are presented for the switch-based system. The size of a queue is equal to the number of Ethernet frames stored in the queue. For each message the number of frames in the queue is entered in a histogram at the time of arrival of the last frame of that message in the queue. These histograms are displayed in the figures. The distributions have shorter tails for the switch-based system. This is due to the fact that one concentrating switch deals with the data from 32 - 48 ROLs, while in the bus-based system one ROS unit deals with the data from 12. ROLs. On average therefore somewhat more data are flowing for the switch-based system via a single port into the central LVL2 switch than for the bus-based system. This leads to less queuing in point I as the incoming frames arrive one after the other with the same speed as with which they can also be output again. The figures also show that for the bus-based system the tail of the distribution for least-queued assignment becomes somewhat smaller if subsequent events are assigned as much as possible to L2PUs connected to different LVL2 subfarm switches. However, the tail can only effectively be suppressed by limiting the number of outstanding requests in the L2PUs. Again the distribution for the bus-based system has a longer tail than for the switch-based system, this is now mainly due to the fact that in the first case there is a maximum to the outstanding number of (in most cases) multi-frame messages requested, while for the bus-based system in most cases single frame messages are requested. The distribution for the LVL2 decision time is for both cases almost identical to the distribution for least-queued assignment presented in Figure 14-14.

The lengths of the queues associated with point II (output ports of the central EB switches connecting to the SFIs) can be controlled with the maximum number of credits, i.e. outstanding requests, in the SFIs, as already discussed in Section 14.6.2.2. For bus-based read-out again multi-frame messages are requested, so for the same maximum number of credits the queues will become longer, see also Section 14.6.2.2.

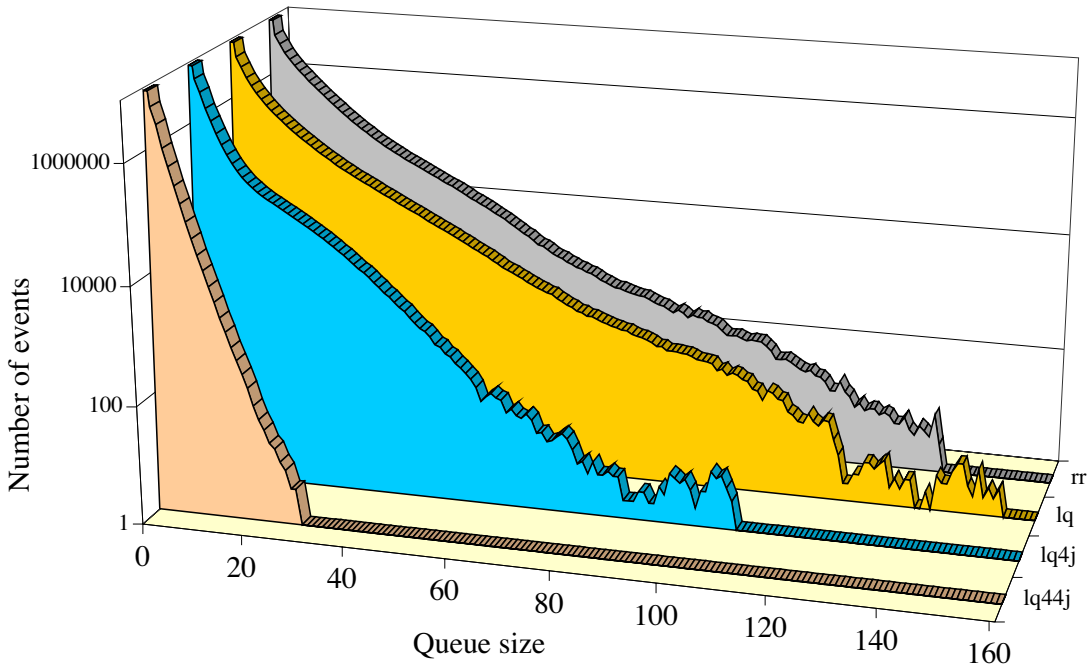

Figure 14-16 Queue sizes at point I, bus-based read-out for round-robin assignment of events to L2PUs (rr), least-queued assignment (lq), least-queued assignment of subsequent events to L2PUs connected to different subfarm switches (lq4j) and for the same strategy, with as additional requirement that the number of outstanding requests is smaller than 4 (lq44j).

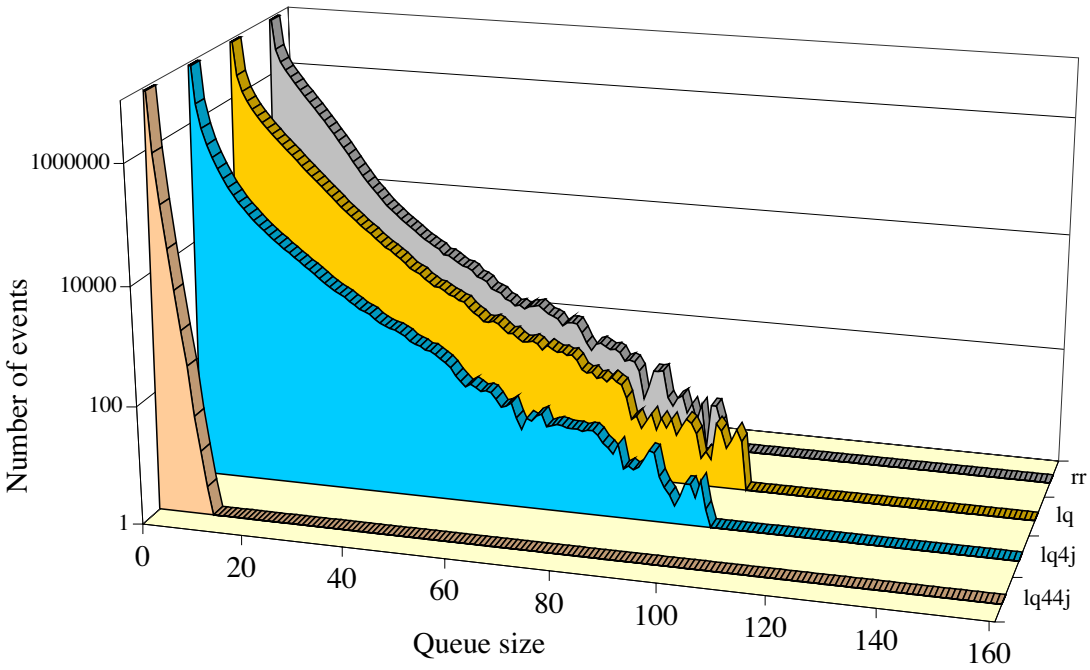

Figure 14-17 Queue sizes at point I, switch-based read-out for round-robin assignment of events to L2PUs (rr), least-queued assignment (lq), least-queued assignment of subsequent events to L2PUs connected to different subfarm switches (lq4j) and for the same strategy, with as additional requirement that the number of outstanding requests is smaller than 4 (lq44j).

Queues associated with point B (output ports of concentrating switches connecting to the central EB switches) in the switch-based system are sensitive for how the SFIs send event fragment requests. For round-robin requesting the requests arrive in all ROBins connected to a single concentrating switch almost at the same time, the responses may then cause contention for access to one of the up-links. It is not excluded that requests from different SFIs follow closely after each other with as consequence more contention. This contention will be alleviated if each SFI requests data from the ROBins in a way which avoids sending subsequent requests to the same concentrating switch, e.g. by selecting the ROBins at random, as done in the testbed measurements and in the at2sim computer model. In the Simdaq model first data are requested from only one of the ROBins associated with each concentrating switch. Only one request message is sent to each ROBin, so the data associated with only a single ROL is requested. This procedure is repeated for the remaining ROLs until all data is requested. In Figure 14-18 the effect of this procedure is shown for the concentrating switches connected to the ROBins receiving data from the Pixels detector. The procedure has also a favourable effect on the sizes of the queues in point A (output ports of the concentrating switches connecting to the central LVL2 switches) for the same detector (see Figure 14-19). However, these queues tend to be short and are less likely to give rise to problems. The maximum number of credits per SFI in the model has been set to 80. This number directly controls the maximum queue size at point II, but queuing at Points B and A is less sensitive to this number.

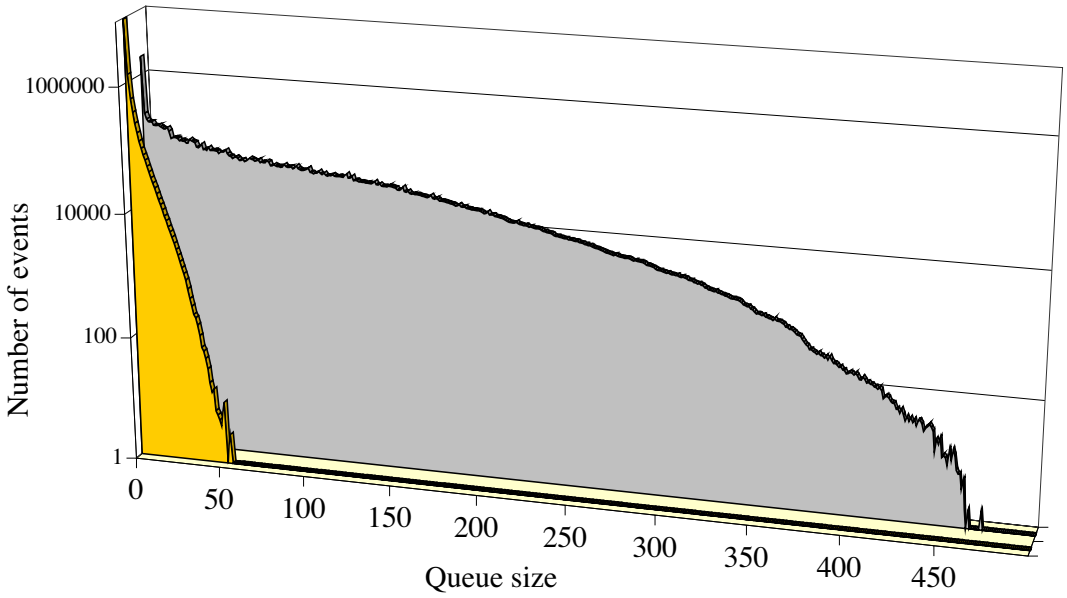

Figure 14-18 Queue sizes at point B for the Pixels. The distribution with the long tail occurs for round-robin requesting of data by the SFIs, the tail disappears if nearly simultaneous requesting via the same switch is avoided with the help of a suitable request pattern, see the text for further explanation.

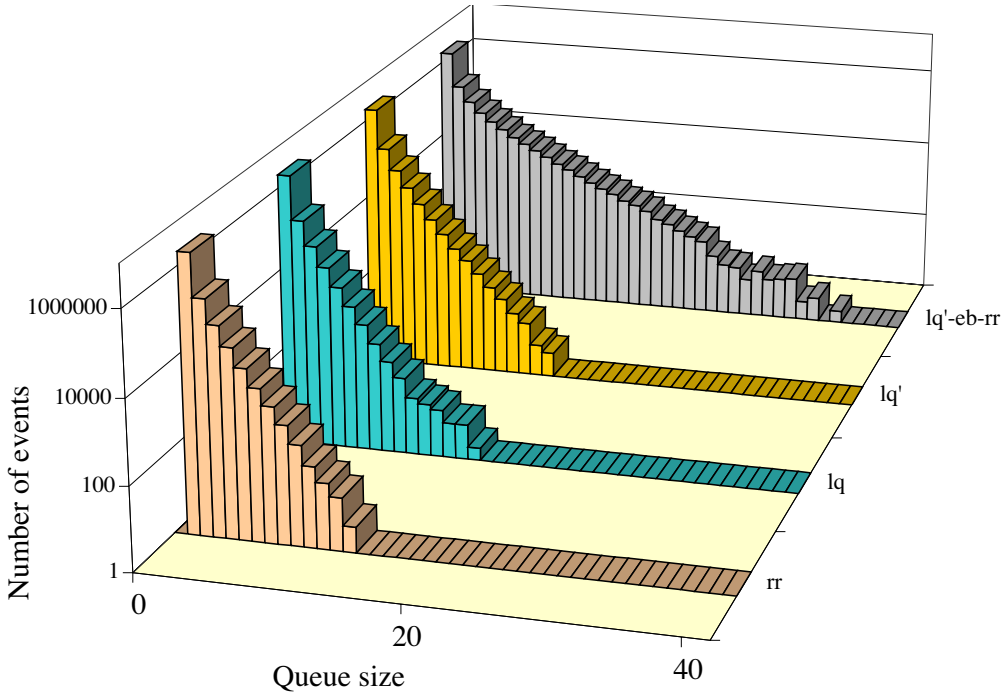

Figure 14-19 Queues sizes at point A for the Pixel detector, for round-robin requesting of event fragments by the SFIs in combination with least-queued assignment of events to L2PUs and a maximum number of outstanding requests of 4 per L2PU (lq'-eb-rr), and for a request pattern avoiding nearly simultaneous requesting by the SFIs via the same switch for the same type of assignment of events to the L2PUs as for lq'-eb-rr, for least-queued assignment of events to the L2PUs without further conditions (lq) and for round-robin assignment (rr).

14.6.2.4 Conclusion

A good understanding of the behaviour of current technology (hardware and software) is available in the form of calibrated component models. An understanding has been developed of how hot spots can be avoided in the full system and how an even distribution of the computing load over the L2PUs and SFIs can be obtained. The modelling results indicate that it is justified to assume that a system based on Gigabit Ethernet and with the current type of components can be operated at the performance level required. A model based on calibrated components for the full system (including the LVL2 system) will make it possible to find good choices for connectivity and operating conditions of the system, but already from the present results it is clear how potential problems can be avoided.

14.7 Technology tracking

14.7.1 Status and prospects

The ATLAS HLT/DAQ system is almost exclusively comprised of off-the-shelf commodity equipment; PCs, and Ethernet links and switches; the exceptions being the RoI Builder and ROBins where custom equipment has had to be developed. The technical evolution of commod-

ity computing, communications equipment, as well as pricing, is therefore an important consideration in the performance, costing and life cycle of the HLT/DAQ system.

Impressive price and performance improvements have occurred over the last two decades. In this section the prospects over the next decade, a period which covers the run up to the commissioning of ATLAS and the first years of running, are considered.

14.7.1.1 The personal computer market

Moore's Law, the doubling of the number of transistors on a chip every 1.5 years, has been respected over the last three decades, and the trend is expected to continue at least through to the end of this decade. In practice, Moore's law has resulted in a doubling of PC performance about every two years, where performance can be quantified in terms of the clock speed of high-end microprocessor chips. The computer industry has offered increasing performance at a more or less constant unit price.

For the future it seems that technically, on the time scale of ATLAS, Moore's law will continue to hold. The turndown in the world economy and a reduced willingness to invest the large sums of money required to deliver new generations of microprocessors may however change this expectation.

The current performance of PC based components and systems in the ATLAS TDAQ are based on ~2 GHz PCs. In estimating the performance of the system we have assumed the use of 8 GHz PCs. This is a conservative estimate. In practice the processing power needed for the LVL2 and Event Filter farms will be purchased in stages and will therefore be able to profit from still higher processor clock speeds. This will be particularly true for the Event Filter farms where the processing time will be long compared to the I/O time. Components in the system with high I/O requirements will be more bounded by link speed, but will also benefit from improvements in processor performance, as illustrated by the performance of the DFM as a function of processor clock speed shown in Figure 14-20.

14.7.1.2 Operating systems

The Linux operating system has evolved rapidly in the last years. Many commercial companies have invested heavily in improving the operating system (IBM, HP, Sun). Currently the main developments are in the areas of user interfaces and high-performance computing. ATLAS can clearly benefit from the Linux developments in the high-performance computing area. Such improvements include better support for multiple processors, better multi-threading and improved networking support. Practically all the new developments toward high-throughput transmission protocols over long-haul links were first implemented under Linux. In the long-term, the optimization of the operating system will continue, fuelled by strong support from the academic and commercial worlds. The wide-spread usage in universities means that ATLAS will have access to qualified Linux professionals throughout the life of the experiment.

14.7.1.3 Networking

The ATLAS HLT/DAQ system uses Ethernet network technology for RoI collection and event building, as well as data distribution to the EF. It is also used in other networks associated with control and monitoring. Ethernet is, throughout the world, the dominant local area network

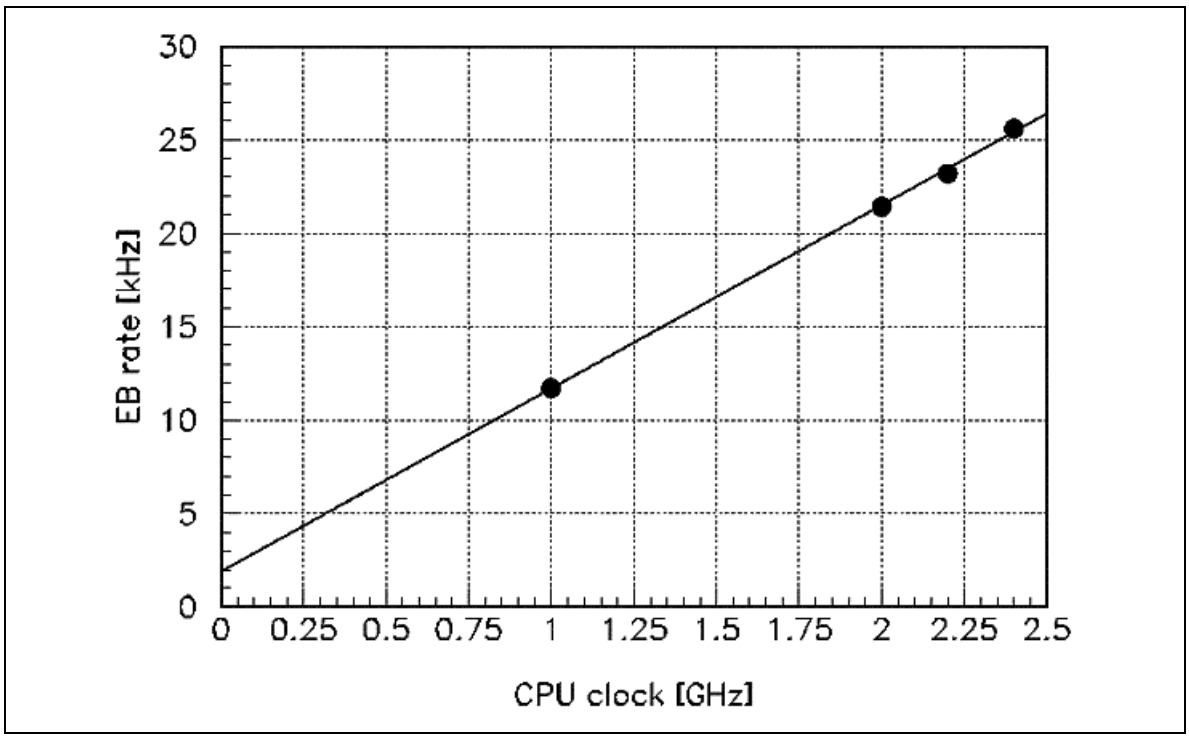

Figure 14-20 The performance of the DFM as a function of processor clock speed.

(LAN) technology. It has evolved from the original IEEE standard, based on a 10 Mbit/s shared medium, to today's point to point links running at speeds of up to 10 Gbit/s [14-20].

The price of Ethernet technology has followed a strong downward trend driven by high levels of competition in a mass market. Figure 14-21 shows the price of 100 Mbit/s (FE) and 1 Gbit/s Ethernet (GE) network interface cards and switch ports as a function of time. Most PCs are now delivered with a GE controller integrated on the motherboard, making the connection essentially free. Further price drops will certainly occur for GE switch ports, in line with what has happened earlier with FE. This trend is coupled to the increasing provision of a GE connection in all PCs.

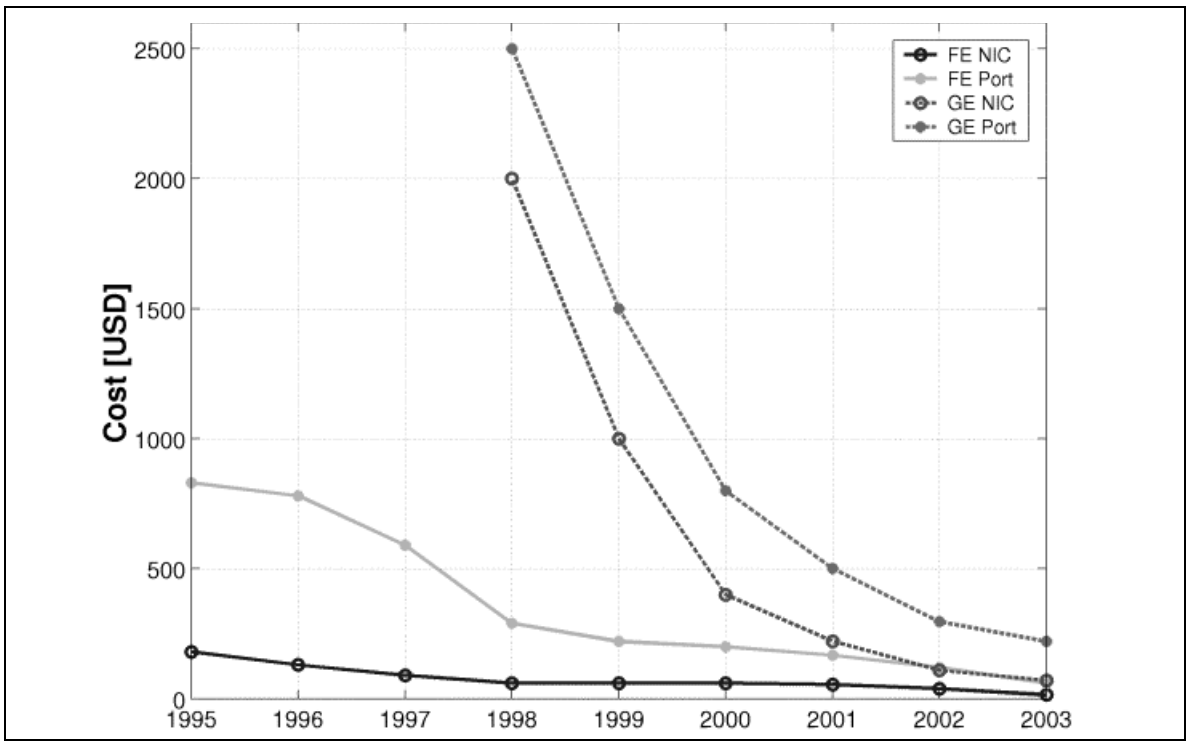

Figure 14-21 The evolution of the cost for Fast and Gigabit Ethernet switch ports and network interface cards.

The HLT/DAQ system described in this TDR can be built using Ethernet switches available today. Even the most demanding components in the system, the large central switches, are comfortably within today's norm. The prognosis for using Ethernet is therefore excellent. It is a very widely supported international standard, which meets and even exceeds our foreseeable need and will certainly have a lifetime surpassing that of ATLAS.

Consideration is being given to the use of off-site computing capacity to process ATLAS events in real time. Tests made recently have shown the technical feasibility of error free Gb/s transmission between CERN and NBI, Copenhagen over the GEANT pan European backbone network [14-21]. Figures 14-22 and 14-23 show the setup used and some of the results obtained.

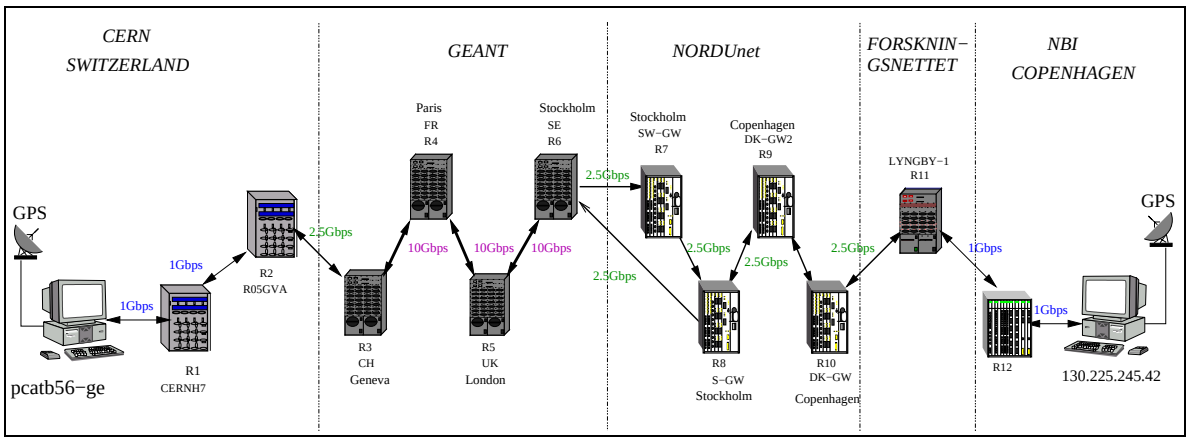

Figure 14-22 The network infrastructure between CERN and NBI (Copenhagen) over which Gb/s tests have been carried out.

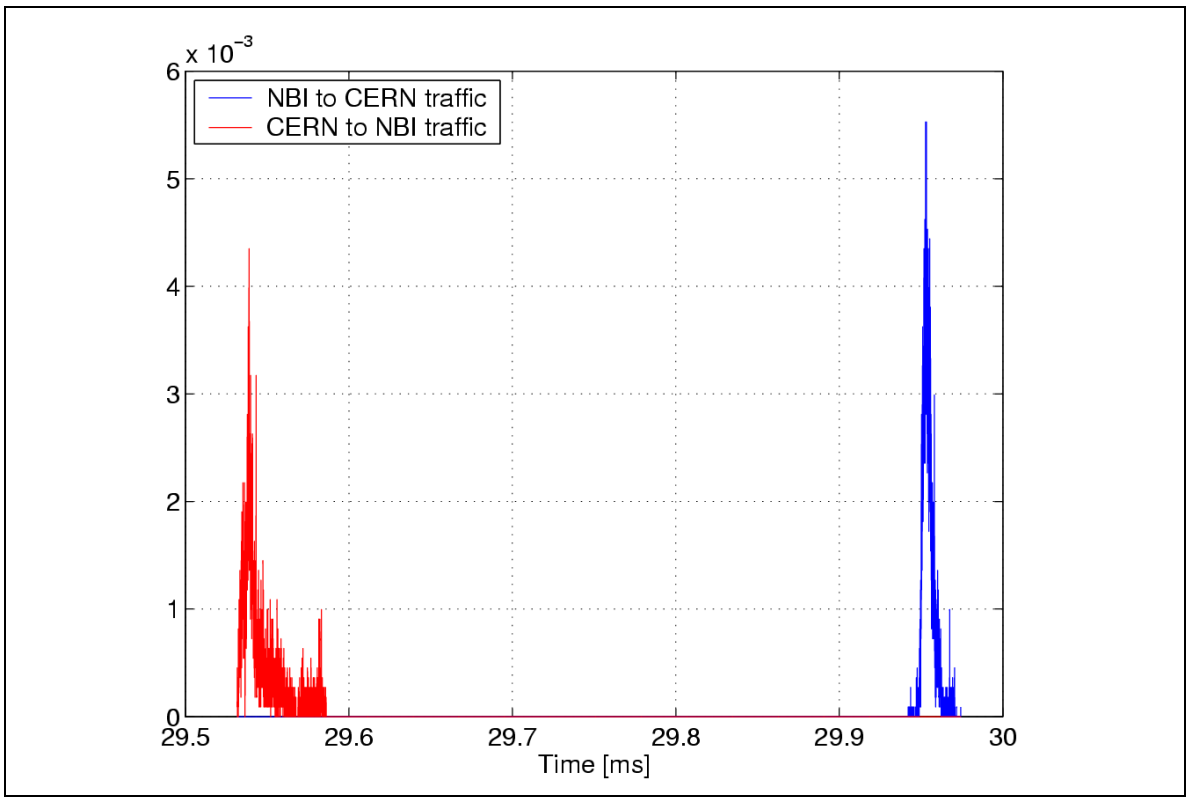

Figure 14-23 Packet latency measured between CERN and NBI (Copenhagen) at a 1 Gbps transmission rate.

For the future, it appears technically feasible to export events at high rates from CERN to centres in member states, for processing in real time. It is within this context that 10-Gigabit Ethernet may have an important role to play. However, the use of such a scheme will ultimately depend on the economics of long haul telecommunications. This factor, as well as technical considerations and practical testing, are part of our on going program of work.

14.8 References

14-1 Tests of the ATLAS LVL2 Trigger with PESA Selection Algorithms", Andre dos Anjos *et al.*, ATL-DH-TR-0002 (June 2003)

14-2 "The use of Gaudi in the LVL2 trigger", S.Gonzalez, A.Radu, W.Wiedenmann, ATL-DAQ-2002-012 (January 2002)

14-3 HLT Validation of ATHENA", M.Bosman, K.Karr, C.Bee, S.Gonzalez, W.Wiedenmann, ATL-DAQ-2002-005 (Jan 2002)

14-4 "Further studies and optimization of the LVL2 electron / photon FEX algorithm", S.Gonzalez, T.Shears ATL-DAQ-2000-42 (May 2000)

14-5 "Initial LVL2 tests with the SiTree algorithm", J.T.M.Baines *et al.*, ATL-DH-TN-001 (March 2003)

14-6 "A data manager for the ATLAS HLT", J.T.M.Baines, W.Li, ATL-DAQ-COM-2003-021 (June 2003)

14-7 "Initial LVL2 Tests with the Si Tree Algorithm", Andre dos Anjos *et al.*, ATL-DH-TN-0001 (March 2003)

14-8 “The implementation of the muFast algorithm in the new PESA framework”, A.Di Mattia, ATL-COM-DAQ-2003-024

14-9 “Definition of Raw Data Objects for the MDT chambers of the muon spectrometer”, K.Assamagan *et.al.*, ATL-COM-MUON-020

14-10 “Raw Data Object definition for the RPC chambers of the ATLAS muon spectrometer”, K.Assamagan *et.al.*, ATL-COM-MUON-019

14-11 ATLAS ATHENA web site, http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/index.html

14-12 Event Filter infrastructure validation tests ”, C.Bee *et al.*, ATL-DH-TR-0004 (June 2003)

14-13 *ATLAS TDAQ: A Network-based Architecture,* H.P. Beck et al.,ATLAS EDMS Note, ATL-DQ-EN-0014, https://edms.cern.ch/document/391592/

14-14 Tigon/PCI Ethernet Controller, Revision 1.04, August 1997, Alteon Networks, F. Barnes et al, "Ethernet Networks for the ATLAS Data Collection System: Emulation and Testing", Paper presented at RT2001, 12th IEEE-NPSS Real Time Conference, Valencia, June 2001, http://atlas-tdaqtalks.web.cern.ch/ATLAS-TDAQtalks/RT01/meirosu.pdf

14-15 R.Cranfield et al., "Computer modelling the ATLAS Trigger/DAQ system performance", submitted to IEEE Trans. on Nuclear Science

14-16 J.C. Vermeulen et al., "Discrete Event Simulation of the ATLAS Second Level Trigger", IEEE Trans.Nucl.Sci.45:1989-1993,1998

14-17 Ptolemy project, http://ptolemy.eecs.berkeley.edu

14-18 P.Golonka, K.Korcyl, ”Calibration of the ATLAS Data Collection component models.”; ATL-DQ-TP-0001, https://edms.cern.ch/document/391590

14-19 P. Golonka et al., “Modelling large Ethernet networks for the ATLAS High-Level Trigger system using parameterized models of switches and nodes.”, CERN-OPEN-2001-061, poster presented on RT 2001 Conference, Valencia

14-20 Dobinson et al. RT2001 Lyon

14-21 Santiago di Compostella, RT2003

Part 4

Organisation and Plan

15 Quality assurance and development process

15.1 Quality assurance in TDAQ

Quality assurance during the production of hardware and software systems is provided by the adoption of a development framework for TDAQ components. The development framework consists of distinct development phases. At the end of each phase a set of deliverables is provided. For hardware, the design and specification of each element will be subject to reviews which will cover all aspects of the system design including integration issues, logistics, quality, and safety. These reviews will also provide an opportunity for QA aspects of the design to be thoroughly examined. For software, the framework is complemented by guidelines, checklists and standards, internal reviews, templates, development and testing tools, and coding standards. These are being adopted as common working practices and help with error removal and error prevention in the system.

The system is divided into many, essentially independent, subsystems. Proper functional definition of each subsystem and specifications for interconnections between them are essential to build, commission, and maintain a quality system. These definitions and specifications are described in User Requirements Documents and Interface Definitions that also specify procedures of control, testing, and maintenance as well as outlining in detail the physics performance goals of the system.

A TDAQ-wide body, the Connect Forum [15-1] assists in coordinating development process activities and quality assurance methodologies across ATLAS TDAQ/DCS. It also provides advice, especially via the recommendations and information made available through Web pages which reflect the dynamic nature of the activity. A common approach to the development via the use of rules, in-house standards, and document templates helps in building a project culture. Those rules as well as the development phases themselves are not enforced but rather aim to help developers. Emphasis on the various phases will vary and evolve with the life of the project. During event production for example, the emphasis will be put on maintenance and regular automated validation testing.

15.2 Hardware development and procurement

15.2.1 Reviews

As noted above, hardware will be subject to a review process: the Preliminary Design Review (PDR), the Final Design Review (FDR), and the Production Readiness Review (PRR) [15-2]. The PDR will be used to examine and assess the full requirements and specifications for the subsystem element, to identify any missing functionality, to ensure full compatibility with all connecting subsystems, and to determine overall feasibility. For custom hardware this is a very important part of the review procedure, as it will determine the direction of subsequent engineering effort.

For custom hardware the FDR will be held before the design is sent for manufacture, and is intended to catch any design or engineering errors before budgetary commitment. This review

will necessarily be of a more technical nature than the initial review, but there should be few problems to detect by this stage. It will be merged with the PRR which aims to address production organization, version control, and the organization of quality control.

For components with commercial hardware to be bought in significant numbers the FDR will be held after a prototype of the component has been assembled and tested. Examples of such components are HLT sub-farm racks with a mixture of commercial items, and ROS Units with a combination of commercial and custom hardware. The role of the FDR is to check that the component fully meets the required specification.

Subsequent purchases of the component, which are likely to be in several batches, will then be preceded by a tendering exercise. This will start with a market survey to check current performances and prices, then a proposal for the precise configuration for the purchase which is reviewed by a PRR, followed by formal tendering. This process should ensure uniform configurations of PCs and local network switches within each round of purchasing, even if constraints of the different funding agencies require them to be purchased from different sources. Note, however, that in the case of HLT sub-farms where batches are purchased at intervals of typically one year, identical configurations would not be expected between batches.

15.2.2 Testing

The following tests and procedures on the TDAQ hardware components will be performed to guarantee their performance to the standards required:

- Evaluate the probable frequency of repair to meet the physics goal on each subsystem.
- Evaluate the effect of failure on system components for various types of failure and design the system so that the more probable failures will cause least problems to the overall system.
- Select components for the resulting level of required reliability.
- Burn-in all essential custom boards and units to reduce initial failures to a minimum.
- Ensure that the communities possess sufficient spares for repair purposes for the entire 10 year running cycle. This will include a study of alternative components to replace those that may not be produced throughout the experiment's lifetime due to a termination of present production processes.
- Establish repair routines for the more routine failures; commercial items should have standard warranties with conditions checked at the PRR.
- Give each element a serial number; a record will be kept to store the information relative to each board: origin of the components which populate the board, date of production, results of tests, date of installation, name of the sub-detector using the board, history of failure, and actions taken.
- Test commercial items; initial test by the supplier, acceptance test by the responsible TDAQ institute.

The networking components are one type of commercial item which will be given special attention. These play a critical role within the ATLAS TDAQ system and the Data Flow network has very high performance demands specific to this application. A network tester system is being developed for the evaluation and testing of network switches. For the concentrating switches, to be purchased in large numbers, the tests of prototypes will include the use of this system. For

the central switches only very few will be purchased and a modified procedure will be required. Experience obtained from extensive tests on switches allows precise specifications to be given for these switches. These will be used to select one or more candidates which will then be evaluated with the tester to ensure that they fully meet the requirements. Once a switch is integrated into the TDAQ system, it will be managed using a commercial monitoring tool (see Chapter 7). Tests will also be made on the connecting cables to be used within the networks, in particular for possible problems with electromagnetic interference when large numbers of Unshielded Twisted Pair (UTP) cables are used.

15.3 The Software Development Process

The Software Development Process (SDP) in ATLAS TDAQ [15-3] provides the structure and the sequence of activities required for development. A basic framework is provided to guide developers through the steps needed during the development of a component or a system. Continual review and modification of the SDP provides it with the flexibility to adapt to the evolution of the components and systems.

Many of the recommended approaches in the SDP are also applicable to the development of hardware components or sub-systems involving both software and hardware. The SDP consists of the following phases as shown in Figure 15-1: Brainstorming, Requirements, Architecture and Design, Implementation, Testing, Maintenance, complemented by reviews. Emphasis on the different phases will evolve with time.

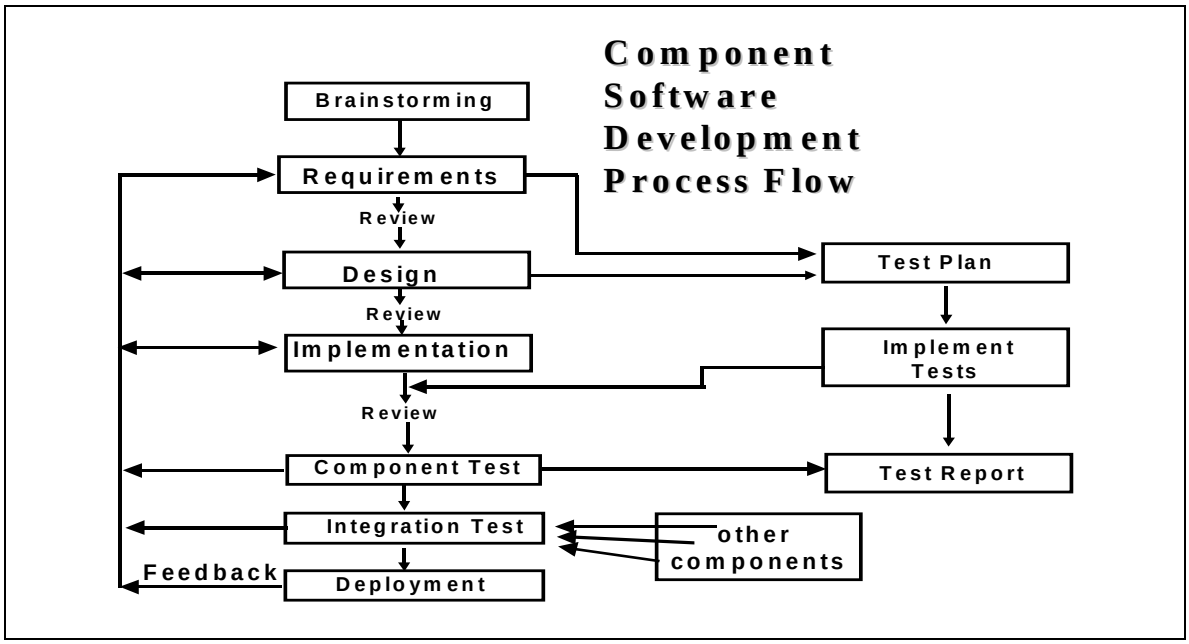

Figure 15-1 Phases and flow of the Software Development Process

15.3.1 Inspection and Review

Written material including documents and code is subjected to a process of inspection and review at each step from Requirements to Implementation, in the SDP. Inspection is essentially a quality improvement process used to detect defects. The inspection process in the ATLAS

TDAQ project is based on Gilb and Graham's Software Inspection method [15-4]. An important feature of the inspection procedure is its flexibility, allowing it to evolve as needs change during the lifetime of the project [15-5].

Overall responsibility for an inspection is taken by an inspection leader who appoints an inspection team consisting of the document author and three to five inspectors. The core of the inspection process is the checking phase where the inspectors read the document in detail, comparing it against source documents and lists of rules and standards. Defects are logged in a table, where a defect is defined as a violation of any of the standards. Emphasis is placed on finding major defects which could seriously compromise the final product. The defects are discussed at a logging meeting and their acceptance or rejection is recorded in an inspection issue log. The document author edits the document according to the log making an explanatory note if an issue is rejected. Feedback is also obtained on how the inspection procedure itself may be improved.

The principal aim of inspection is to detect and correct major defects in a product. An additional benefit is the possibility to prevent defects in future products by learning from the defects found during inspection procedures. Inspection also provides on-the-job education to people new to a project and generally improves the project's working culture.

A number of Web pages have been produced which provide supporting material for inspections such as instructions for inspectors and log file templates [15-1].

15.3.2 Experience

The SDP provides a disciplined approach to producing, testing, and maintaining the various systems required by the ATLAS TDAQ project. It helps to ensure the production of high-quality software and hardware which meet the requirements within a predictable schedule.

However, one of the key differences in adopting the SDP in an HEP — as opposed to an industrial — environment is that its application cannot be enforced. Furthermore, the use of such a process may appear too rigid to physicists not accustomed to working in a strong management framework. Nonetheless, the working culture can be changed by increasing awareness of the benefits of the SDP through training, for example involving new group members in inspections, and ensuring that the SDP itself is sufficiently flexible to evolve with the changing needs of an HEP experiment. This approach is working. The SDP as outlined in this section has already been adopted by a number of the sub-systems in the ATLAS TDAQ project with positive outcomes.

15.3.3 The development phases

15.3.3.1 Requirements

The Requirements phase [15-6] for a particular sub-system or component consists of gathering the requirements and then documenting them. Several documents have been produced to aid and control these activities, based on the early experience of some of the sub-systems. The whole process of working group setup, requirements collection, feedback and review is described in [15-6]. Another document sets out the principles governing the requirements gathering and documentation processes, stressing the importance of, for example, documentation,

evolutionary development, communication, and collective ownership of the requirements specification.

The actual process of establishing the requirements for a sub-system or component is aided by a collection of ‘hints’, and reinforced by a set of rules for the requirements document itself, for which a template has been provided in each of the supported documentation formats.

15.3.3.2 Architecture and Design

The Architectural Analysis and Design Phase of the SDP [15-7] follows the Requirements phase and takes as its starting points the User Requirements and Use Cases together with accompanying documents. This phase has sometimes been referred to as ‘high-level system design’. A system’s architecture is the highest level concept of that system in its environment. It refers to the organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces. A design presents a model which is an abstraction of the system to be designed. The step from a real world system to abstraction is analysis. A ‘How to’ note has been produced describing the overall process.

For this phase, the approach of the Rational Unified Process (RUP) is largely followed. This approach contains descriptions of concepts, artefacts, guidelines, examples, and templates. Items of the RUP have been highlighted, in particular, the descriptions of architectural analysis and design concepts, and the guidelines for producing software architecture and design documents.

The RUP template for architecture and design documents has been adapted by including explanations and making it available in supported formats. The recommended notation is the Unified Modelling Language (UML), and the design is presented in the template as a set of UML-style views. Recipes for producing appropriate diagrams and incorporating them into documents have also been prepared.

15.3.3.3 Implementation

The Implementation Phase of the SDP is largely concerned with writing and checking code, and producing and debugging hardware components. At the end of the implementation phase an Inspection is performed.

ATLAS C++ coding conventions [15-8] are being applied to newly written code and being introduced for existing code still in evolution. In the case of Java, the outcome of the ATLAS investigation of coding conventions is awaited. DCS will follow the coding standards provided by the JCOP Framework for PVSS [15-9].

Guidelines [15-10] have been provided for multi-user, multi-platform scripting, as well as many explanations and examples in UNIX-scripting.

Experience has been gathered with a number of software tools and recommendations have been made in the areas of design and documentation [15-11], code checking, and source code management.

15.3.3.4 Component and integration testing

Testing occurs during the entire life-time of a component, group of components, or entire system. Referring to Figure 15-1, the initial test plan is written during the requirements and design phases of the component, so as not to be biased by the implementation. Since testing is likely to be an iterative process the test plan is written with re-use in mind. Once implementation is complete and passes relevant checking tools the component undergoes unit testing to verify its functionality. Compatibility with other components is verified with integration tests. Several types of tests can be envisaged for both individual components and groups of components. These include functionality, scalability, performance, fault tolerance, and regression tests.

A test report is written once each test is complete. To aid the testing procedure, templates [15-13] are provided for both the test plan and test report in each of the supported documentation formats. More detailed descriptions of the types of test, hints on testing and recommended testing tools are also provided. Testing is repeated at many points during the lifetime of a component, for example at each new release of the component software or after a period of inactivity (system shutdown). Automatic testing and diagnostic procedures to verify the component before use greatly improve efficiency.

15.3.3.5 Maintenance

As with testing, maintenance occurs during the entire lifetime of a component. Several types of maintenance can be envisaged. Corrective maintenance involves the fixing of bugs. Adaptive maintenance involves alterations to support changes in the technical environment. Preventive maintenance entails the restructuring and rewriting of code, or modification of hardware for future ease of maintenance. Maintenance is closely coupled to regression testing which should occur each time a maintenance action has been completed to verify that the detected problems have been solved and new defects have not been introduced. Significant changes to the functionality of the component such as the addition of large numbers of new requirements should involve a full re-iteration of the SDP cycle.

15.3.4 The Development Environment

Regular releases of the sub-system software to be used in test-beam operation, system integration, and large-scale tests is being complemented by nightly builds and automated tests to ensure early problem finding of newly developed or enhanced products. The use of a source code management system and of the standard release building tool CMT [15-14] allows for the building of common releases of the TDAQ system. These releases are available for the platforms used in ATLAS TDAQ which are currently a number of Linux versions and for some sub-systems LynxOS and SunOS. Build policies of different sub-systems like the use of compiler versions and platforms are co-ordinated.

Development tools like design tools, memory leak checking tools, automatic document production tools, and code checking tools are vital elements of the development environment.

15.4 Quality Assurance during deployment

15.4.1 Quality Assurance from early deployment

Deployment of the TDAQ system in test beam and during commissioning gives early feedback on the suitability of the design and the implementation of the system. It constitutes an integration test under realistic conditions in terms of functionality, reliability, and usability. Experience and findings are directed to the developers so that improvements can be made to the software and hardware concerned. A bug tracking system is in use for the reporting of software problems, their recording, and their follow-up. Complex integration aspects can be better understood and lead to adjustments in the design of components, interfaces, and strategies. The early use of the system allows the TDAQ user to become familiar with its operations and usage.

15.4.2 Quality Assurance of operations during data taking

The quality of the TDAQ system must be assured when it is in use during the setup and installation phase of the ATLAS data acquisition together with the detectors. Correct and smooth data taking will be aimed for during calibration and physics event production.

Quality assurance is achieved by prevention, monitoring, and fault tolerance.

- Prevention — this includes training, appropriate documentation, a well-defined development process, proper management of computing infrastructure (computer farms, readout electronics, and networks), tracing of hardware and software changes, and regular testing of components.
- Monitoring — special tasks to monitor proper functioning of equipment and data integrity. These may run as special processes or be part of the TDAQ applications. Anomalies are reported, analysed by human / artificial intelligence, and appropriate recovery action is initiated. This may include running special diagnostic code, replacement of faulty equipment, rebooting of processors, restarting of applications, re-establishing network connections, and reconfiguration to continue with a possibly reduced system. Incomplete or corrupted data should be marked in the event data stream and possibly recorded in the conditions database. Physics monitoring may lead to a change of run with different trigger conditions and event selection algorithms.
- Fault tolerance — built into the system from the start using an efficient error reporting, analysis, and recovery system as explained in Chapter 6, "Fault tolerance and error handling". Some redundancy to reduce possible single points of failure is foreseen where affordable.

During the life of the experiment small or major pieces of hardware or software will need to be replaced with more modern items, possibly using new technologies. The component structure with the well-defined functionality of each component and well-defined interfaces allowing for black-box testing according to those functionality specifications will allow the smooth incorporation of new parts into a running system, and in particular when staging of the system is required.

15.5 References

15-1 ATLAS Connect Forum,
http://cern.ch/Atlas-connect-forum/

15-2 P. Farthouat, *Guidelines for Electronics Design and Production Readiness Reviews*,
http://cern.ch/Atlas/GROUPS/FRONTEND/WWW/designreview.pdf

15-3 Software Development Process,
http://cern.ch/Atlas-connect-forum/ - SDP

15-4 T. Gilb and D. Graham, *Software Inspection*, Addison-Wesley (1993)

15-5 I. Alexandrov et al., *Impact of Software Review and Inspection*, Computing in High Energy and Nuclear Physics Conference, Padova, 2000

15-6 SDP Requirements documents,
http://cern.ch/Atlas-connect-forum/ - SDP - Requirements:
Practical Steps towards an Atlas TDAQ Requirements document
Requirements gathering and documentation 'principles'
Hints on how to establish requirements
Requirements Document Rules ATLAS DAQ 'in-house' rules for Requirements documents
Requirements Document Template for Systems and Components, Software and Hardware

15-7 Architecture and Design documents,
http://cern.ch/Atlas-connect-forum/ - SDP - Architecture & Design:
How-to for Design
RUP URL for concepts
RUP URL for guidelines
Template for Software Architecture Document

15-8 ATLAS Quality Assurance Group, *ATLAS C++ Coding Standard Specification*, ATLAS Internal Note, ATL-SOFT-2002-001 (2002)
http://cern.ch/Atlas/GROUPS/SOFTWARE/OO/Development/qa/General/index.html

15-9 Joint Controls Project,
http://cern.ch/itco/Projects-Services/JCOP/

15-10 SDP Implementation,
http://cern.ch/Atlas-connect-forum/ - SDP - Implementation:
Multi-user multi-platform scripting guidelines

15-11 SDP Tools,
http://cern.ch/Atlas-connect-forum/ - Tools

15-12 SDP Templates,
http://cern.ch/Atlas-connect-forum/ - Templates

15-13 SDP Testing phase,
http://cern.ch/Atlas-connect-forum/ - Testing

15-14 Configuration Management Tool,
http://www.cmtsite.org/

16 Costing

The cost estimate of the ATLAS HLT/DAQ system is based on a detailed model of its size (i.e. number of components) as a function of the input Level-1 trigger rate. The model is parametrized by the assumptions described in Tables 5-3 and 5-4; conservatism and safety factors are applied in particular for the performance (processing time per event and rejection power) of the HLT and the estimated unit cost of both custom (e.g. the ROBin) and commercial components (e.g. processors). In the following are described the HLT/DAQ system evolution from commissioning to its expected final performance, the rationale behind the costing of individual components, the subdivision of the system cost in terms of functional categories, and a summary of the expenditure profile.

16.1 System evolution and staging

The ATLAS HLT/DAQ system has been designed to be staged, with the size and performance of the system evolving as resources become available. The final performance corresponds to a Level-1 rate of 100 kHz. Table 16-1 indicates, for the period 2004 to 2009, the performance capability (second column) and the functional capability (third column) of the installed system. The dates of 2008 for the nominal 75 kHz system, and 2009 for the final HLT/DAQ performance, are indicative, in so far as they depend on the availability of resources, the luminosity performance of the LHC and experience from initial running.

Table 16-1 HLT/DAQ performance profile

Year	Sustained Level-1 rate (kHz)	Notes
2004	N/A	Pre-series only
2005	N/A	Detector & HLT/DAQ commissioning. 75% of detector readout Use pre-series HLT farms
2006	N/A	ATLAS cosmic-ray run Use pre-series HLT farms
2007	37.5	LHC startup 37% HLT farms
2008[a]	75	Nominal LHC luminosity 100% of detector readout 75% HLT farms
2009[a]	100	Final HLT/DAQ performance 100% HLT farms

a. Indicative date.

Chapter 5 discusses how the baseline architecture supports the staging of the HLT/DAQ system. The detector readout procured and installed in 2005–2006 is just for the non-staged part of the detector (which represents ~75% of the ROLs). For the purpose of uniformity, custom components, in particular the ROBin modules, are fully procured in 2005, although some of them will be physically installed later (probably in the year 2008) with the staged parts of the ATLAS

detector. The other parts of the ROS system are, however, procured following the staging of the detector (i.e. 75% in 2005–2006 and the remaining 25% in 2008). The strategy for staging of the HLT/DAQ system is based on the staging of the LVL2 and EF farms and, as a consequence, of the Event Builder (in particular the SFIs) and the central networks.

16.2 Costing of components

The majority of the ATLAS HLT/DAQ system is based on Commercial Off The Shelf (COTS) components: commodity computer and communications hardware (e.g. PCs and Gigabit Ethernet) and related equipment. The estimated unit cost of these components is based on a conservative extrapolation of today's costs. In addition, for the HLT farms, a 30% overall safety factor is added to allow for the large uncertainties discussed in Section 14.3.

The unit cost of custom components, i.e. the ROBin and RoIB, has been established on the basis of R&D prototypes.

16.3 Categories of expenditures

For the purpose of describing the cost and its evolution in time, the HLT/DAQ system has been organized in terms of categories of expenditure, closely related to the HLT/DAQ baseline architecture:

- DCS: includes the components needed for the operation of the detector and for the supervision of the infrastructure of the experiment: operator workstations, servers, detector master stations, custom modules, DSS, sniffer, and UPS systems.
- Detector readout: this category includes the ROBins, the ROSs, and the related infrastructure.
- Level-2 RoI: includes the RoIB, the LVL2 Supervisor, the Level-2 central network, and the infrastructure.
- Level-2 processors: includes the Level-2 processor farm and the infrastructure.
- Event Builder: includes the DFMs, the SFIs, the Event Builder network, the SFO (with the local data storage), and the related infrastructure.
- Event Filter: includes the Event Filter processor farm, the network internal to the Event Filter, and the infrastructure.
- Online: includes processor farms for running the online software, HLT/DAQ operations and monitoring, the central online network, and the related infrastructure.
- Other: includes items such as contributions to high speed data links in and out of the experimental area, and the acquisition of software products such as farm and network management tools.
- Pre-Series: includes a small-scale version of the system for the purpose of validating the HLT/DAQ implementation.

16.4 Expenditure profile and system cost

The expenditure profile is summarized in Table 16-2 and Figure 16-1. The former indicates the cost for the ATLAS HLT/DAQ system in each year. Figure 16-1 shows the expenditure profile including its split on the basis of the categories of expenditure.

Table 16-2 HLT/DAQ system cost profile (unit is kCHF)

	Up to 2003	2004	2005	2006	2007	2008	2009	Total
DCS	296	1624	489	294	422	0	0	3125
Pre-series	0	1048	0	0	0	0	0	1048
Detector R/O	0	0	3049	606	0	405	0	4060
Level-2 RoI	0		400	230	238	868	16	1752
Level-2 Proc	0	0	0	450	899	1348	899	3596
Event Builder	0	0	100	526	274	910	274	2084
Event Filter	0	0	0	1375	2863	4351	2862	11 451
Online	0	0	208	622	0	0	0	830
Infrastructure	0	0	0	508	508	508	508	2032
Total	296	2672	4246	4611	5204	8390	4559	29 978

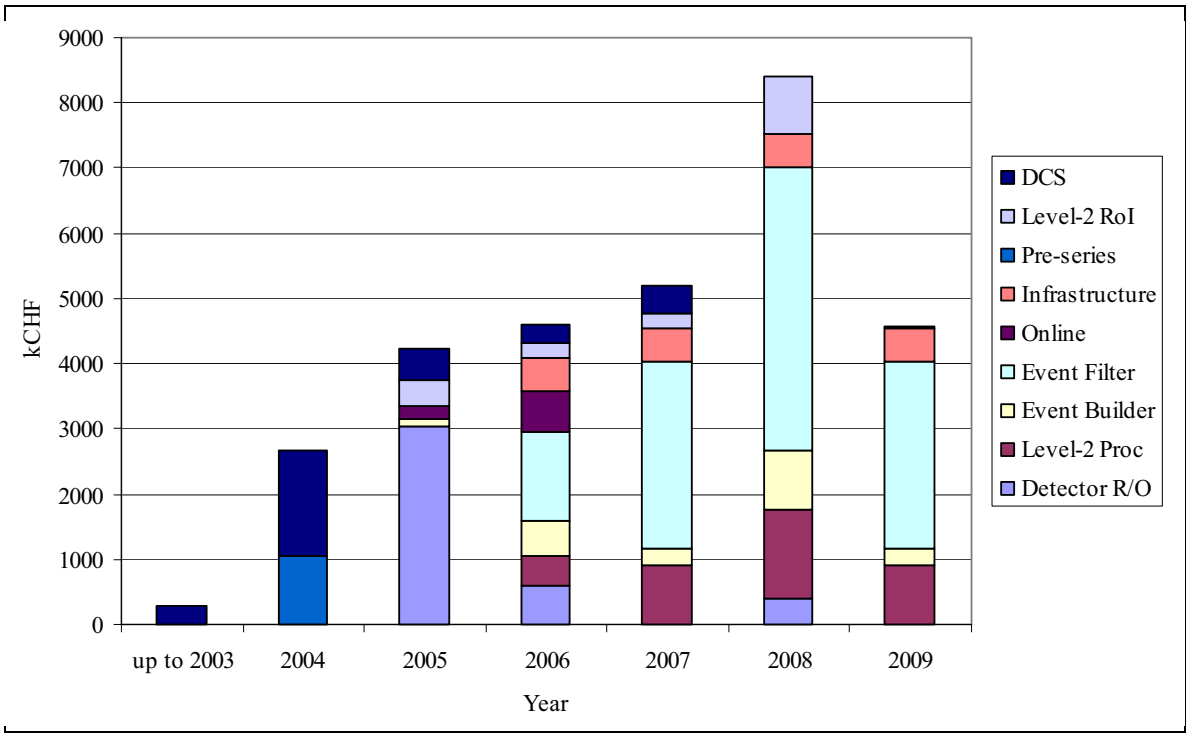

Figure 16-1 Expenditure profile

The sharing of the cost (for the final system capable of sustaining a 100 kHz Level-1 rate) between the categories of expenditure, is shown in Figure 16-2.

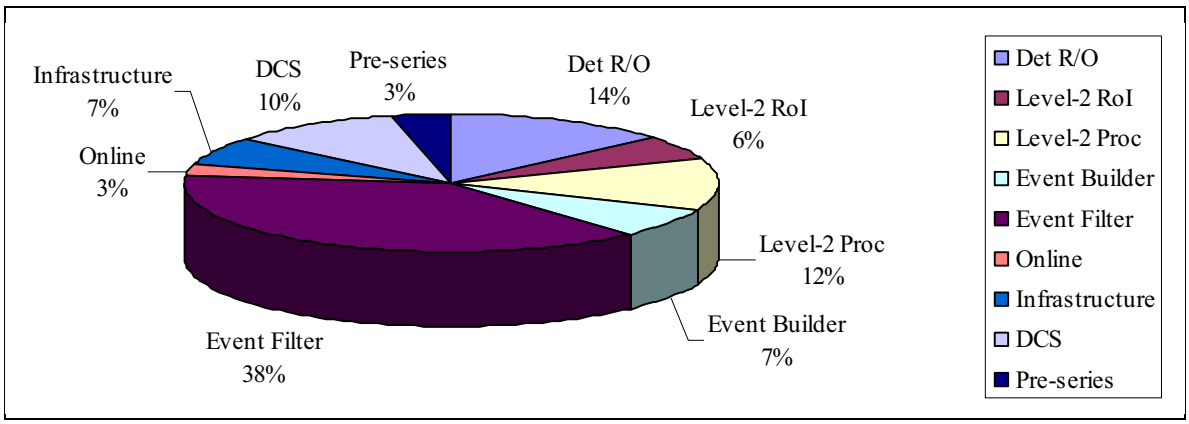

Figure 16-2 Relative cost per category of expenditure (for the final system capable of sustaining a 100 kHz Level-1 rate)

The total HLT/DAQ system cost, as a function of the Level-1 rate and not including pre-series, is shown in Figure 16-3.

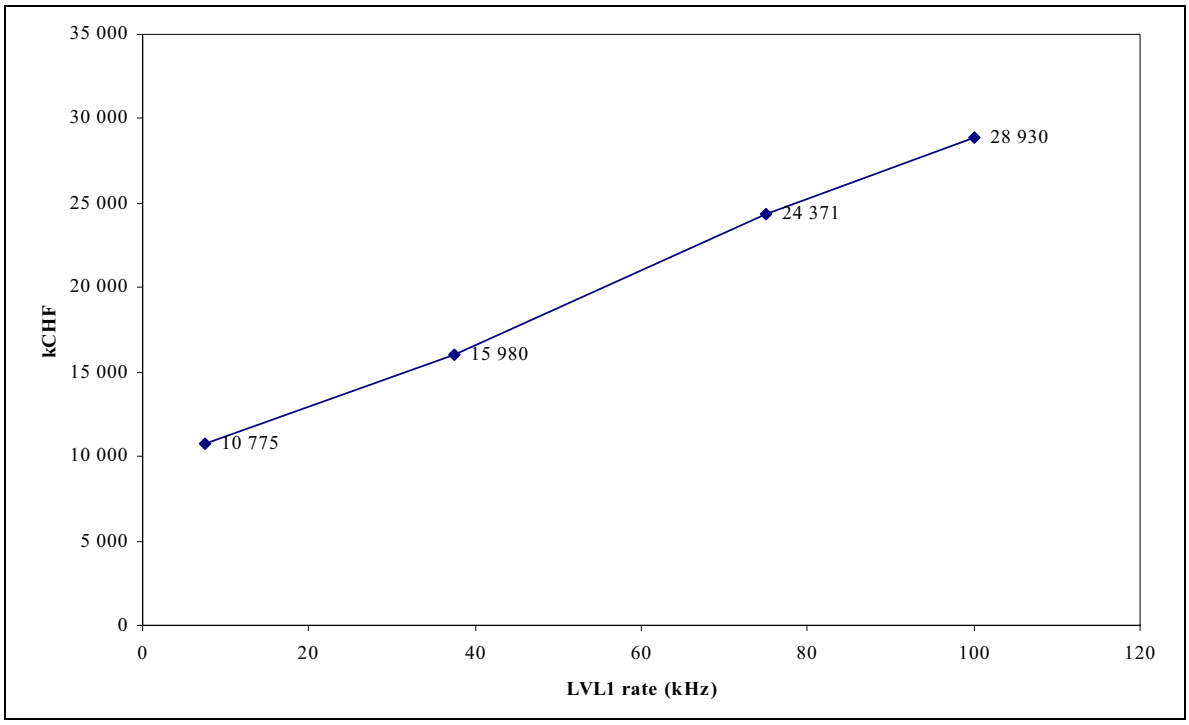

Figure 16-3 HLT/DAQ cost (excluding pre-series) versus Level-1 rate achievable

17 Organization and resources

17.1 Project organization

The ATLAS Trigger/DAQ Project is organised in three main sub-projects: the Level-1 Trigger (N. Ellis), the High Level Trigger (C. Bee) and the Data Acquisition (L. Mapelli). The management of the overall project is organised in three levels:

1. The TDAQ Management Team (TDMT), whose members are the leaders of the three sub-projects with the close involvement of the chairperson of the Trigger/DAQ Institute Board (TDIB, see Section 17.2). One of the three sub-project leaders acts as overall Project Leader on a yearly rotational basis, with the particular role of representing the project towards outside bodies.
2. The Trigger/DAQ Steering Group (TDSG), composed of the coordinators of the main sub-systems of each sub-project, the coordinators of cross-system activities and a number of ex-officio members, including the ATLAS Management and the TDIB Chair (see Figure 17-1).

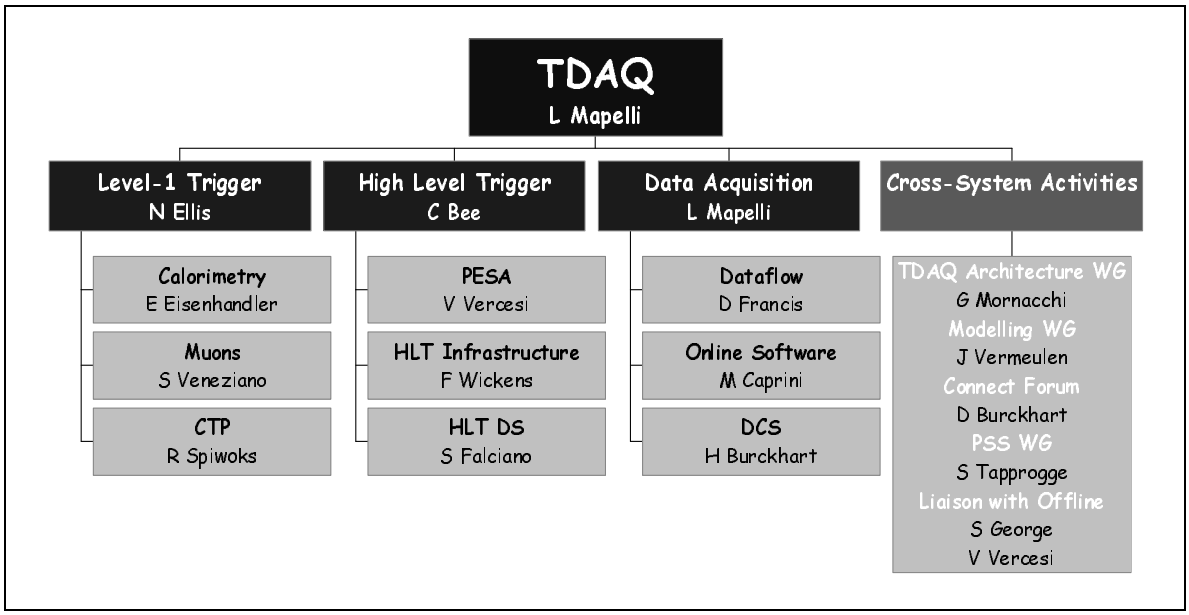

Figure 17-1 The Trigger/DAQ Steering Group (HLT DS stands for HLT Detector Slices, and PSS stands for Physics Selection Strategy)

3. Each sub-project has its own internal organization and management, tailored to the specific features and needs of the system. The organisation of the HLT and DAQ sub-projects is illustrated in Figure 17-2 and Figure 17-3. Both the HLT and the DAQ are organised in three sub-systems each and a number of cross-sub-system activities. The organization of the LVL1 trigger sub-project is described elsewhere [17-1].

The progress of the TDAQ project also requires significant interactions with other parts of ATLAS, and for this formal links have been established in several areas. First the Detector Interface Group (DIG) provides a forum to co-ordinate the work of TDAQ and the detector communities, particularly for test beam work and detector commissioning. The DIG coordinator is an

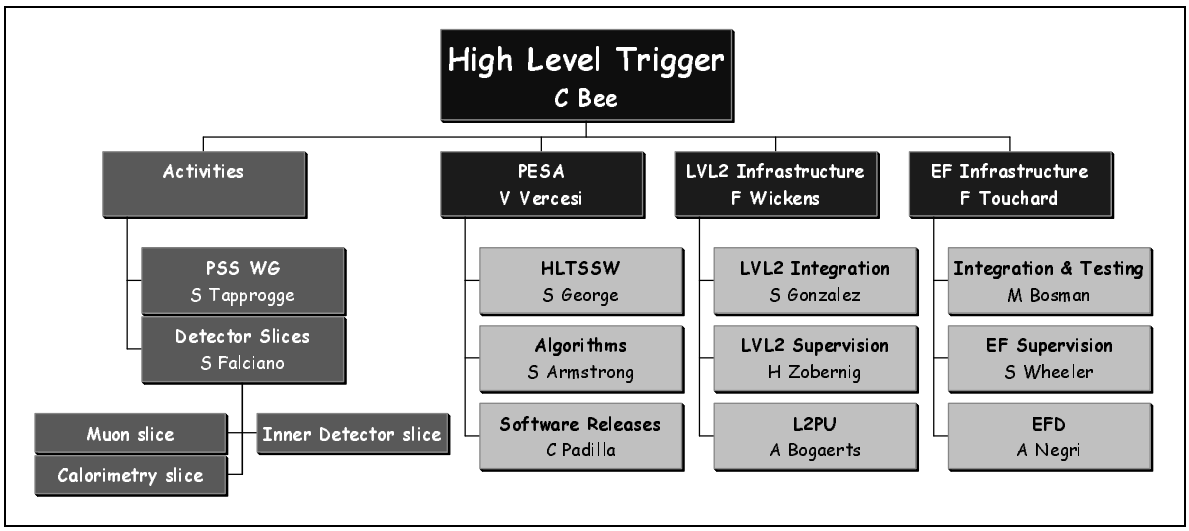

Figure 17-2 The High Level Trigger Steering Group ('Activities' are working groups cutting across the sub-systems)

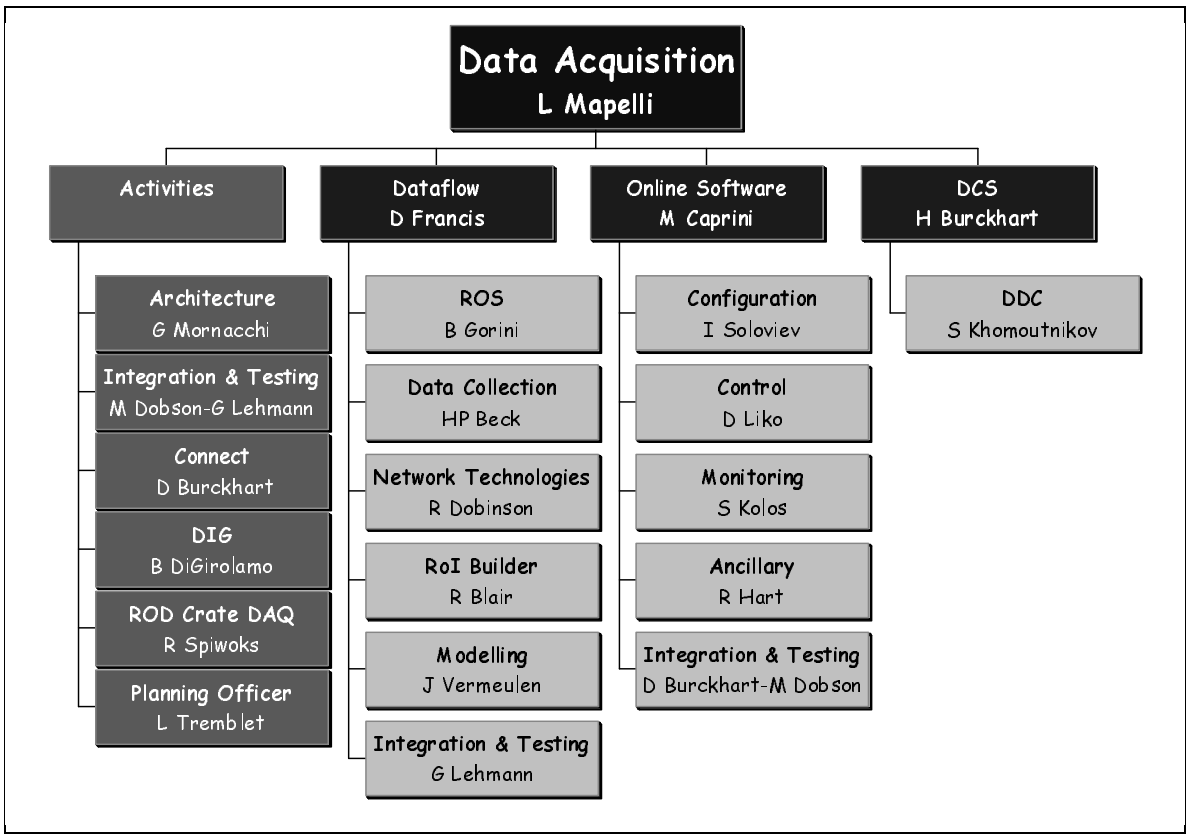

Figure 17-3 The DAQ Steering Group ('Activities' are working groups across the sub-systems, extendable to HLT and LVL1 as well)

ex-officio member of the TDSG and several members of TDAQ are members of the DIG Coordination Team. For interactions with ATLAS Technical Coordination, particularly in issues relating to the infra-structure requirements of TDAQ, the TDMT has been assisted by members of the community undertaking specific responsibilities. For the future it is foreseen that there will be a TDAQ Planning Officer who would take responsibilities in this area. There are particularly strong links with the ATLAS off-line group and in addition to frequent meetings at the level of

the management teams two members of the TDSG are also members of the Computing management. In resource matters the TDMT is assisted by the Chair of the TDIB (see below) and the TDAQ Resource Coordinator (F. Wickens), the latter also acts as the contact between TDAQ and the ATLAS Resource Coordinator.

The submission and approval process of this TDR marks the completion of most elements of the R&D phase of the TDAQ project. The project organization will be adjusted in late 2003 in order to reflect the change of phase of the project, as it moves from design and prototyping to production.

17.2 Resources

The overall ATLAS Trigger / DAQ Collaboration consists of 51 Institutes from 22 countries. Each institute is represented in the TDAQ Institute Board (TDIB) [17-2], the policy and decision making body of the TDAQ project, presently chaired by A. Nisati. Typical tasks of the TDIB include discussion and decisions on financial and human resource as well as policy making.

The TDIB is assisted by two committees, the Resource Committee and the Speakers Committee.

- ***TDAQ Resource Committee (TDRC):*** The TDIB is advised on finance and resource matters by the TDRC, comprising the TDMT, one member per major Funding Agency and two additional members representing collectively the other Funding Agencies. The TDRC is chaired by the TDIB chair, assisted by the TDAQ Resource Coordinator.
- ***TDAQ Speakers Committee:*** This is a three member body, presently chaired by L. Levinson, mandated to recommend policy regarding conference speakers, to maintain a complete archive of conference presentations and to ensure a fair distribution of conference talks across the TDAQ Collaboration.

The HLT / DAQ part of the Collaboration comprises 41 Institutes from 21 countries for a total of ~230 members.

17.3 References

17-1 *ATLAS Level-1 Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

17-2 TDIB,
http: / / cern.ch / Atlas / GROUPS / DAQTRIG / IB / ib.html

18 Workplan and schedule

This chapter outlines the short-term (6–12 months) post-TDR workplan for the major activities in the HLT and DAQ systems (including DCS). The global HLT/DAQ development schedule is presented in Section 18.1 as the basis for the definition and context of the workplan. The detailed workplan, which is still in preparation, is introduced and a number of issues that will have to be addressed are indicated (Section 18.2). The strategy that has been developed for the detector integration and commissioning is described in Section 18.3.

18.1 Schedule

This section presents the overall schedule for the HLT/DAQ system up to LHC turn-on in 2007. The development and deployment of the HLT/DAQ system, both hardware and software, has been mapped onto the ATLAS detectors installation plan [18-1].

18.1.1 System hardware

For the system hardware, the planning is driven by the production schedule for the two custom readout components:

- the S-LINK Link Source Card (LSC), to be installed on the RODs;
- the ROBin, the receiver part of the ROS (S_LINK receiver and ROL multiplexer).

An analysis of the detector installation schedules points to the first quarter of 2004 as the moment for the Final Design Review of the LSC and the ROBin. The review of the ROBin requires the I/O optimization studies (see Section 18.2.1) to have been completed beforehand. (The third custom component of the HLT/DAQ, the RoI Builder, is not on the critical path for the detector installation schedule.)

The global HLT/DAQ production schedule is shown in Figure 18-1. It identifies the principal milestones concerning the detector needs for TDAQ installation, the TDAQ component production (in the case of custom components), the component purchasing and associated tendering (in the case of commercial components), as well as testing and commissioning.

18.1.2 System software

Six major releases of the system software have been identified (dates are indicative at this stage and will be adapted if necessary). The release strategy is driven by detector operations (e.g. test beams and cosmic-ray run) and commissioning (see Section 18.3).

1. Current DAQ release, integrating Online Software and DataFlow. It is targeted at the needs of the ATLAS H8 test-beam operations in 2003 and to the I/O optimization and system-performance measurements. This release has been operational since May 2003.
2. ‘Combined test-beam’ release for the combined run in 2004.
3. ‘Sub-detector readout’ release for initial detector commissioning with single ROD Crate DAQ (RCD) for single crates.

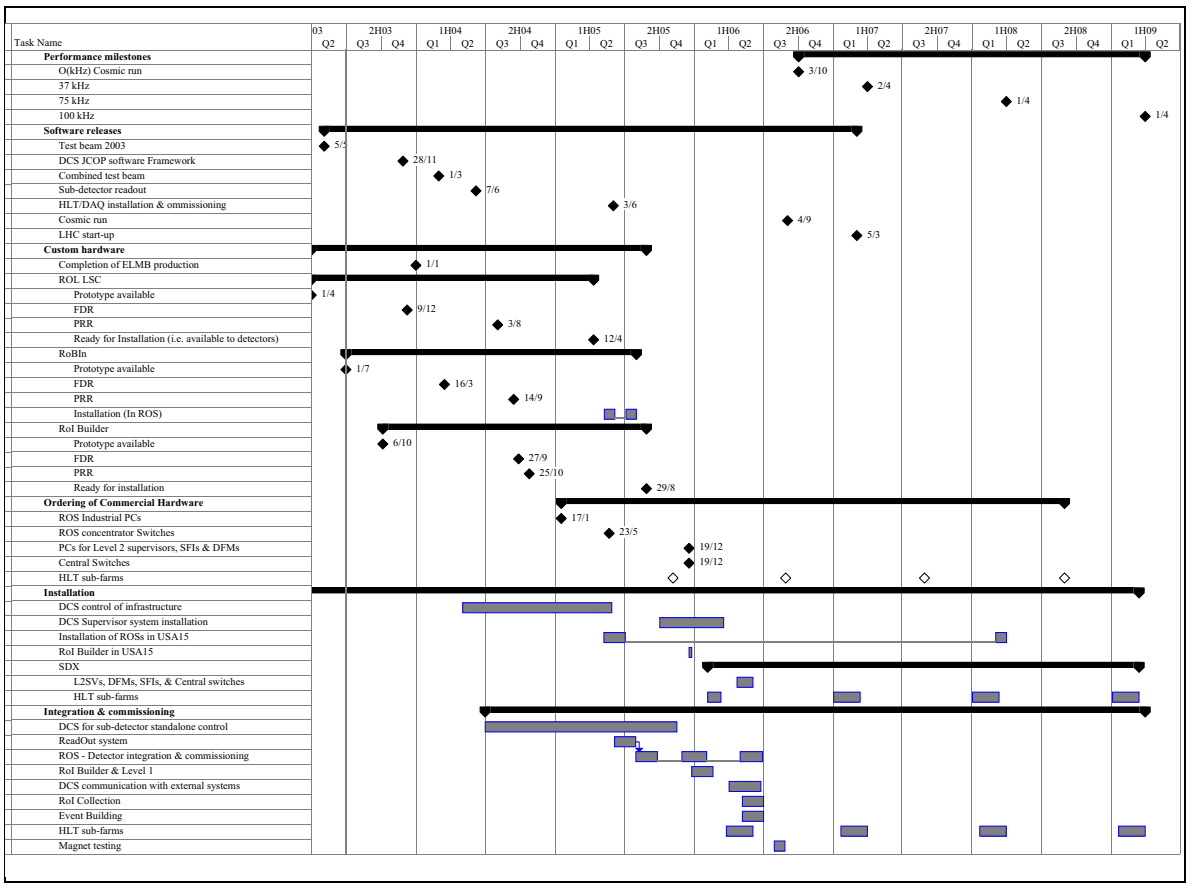

Figure 18-1 Schedule of the overall HLT/DAQ project

4. 'HLT/DAQ installation' release for commissioning of the HLT/DAQ components and global detector commissioning in autumn 2005.
5. 'Cosmic-ray run' release for global HLT/DAQ and global detector commissioning, as well as for the ATLAS cosmic-ray run in autumn 2006.
6. 'LHC start-up' release for the start of LHC operation in April 2007.

An overview of the software-development schedule is shown in Table 18-1 for the software of the three sub-systems, Online Software, DataFlow, and High-Level Trigger. For the HLT, only the LVL2 and EF infrastructure part of the software (the latter is required for detector monitoring during detector commissioning phases) is considered here. The development schedule of the event selection software and the selection algorithms, is not linked to the DAQ releases and is not reported in the table. This schedule will be defined later this year.

18.2 Post-TDR workplan

The detailed workplan to meet the objectives described in the schedule of the previous section is in preparation and will be finalized soon after the TDR publication. Its description goes beyond the scope of this document, which specifically addresses the workplan of the short-term post-TDR phase, characterized by the completion of the studies for the optimization of the HLT/DAQ baseline, both in the DataFlow (Section 18.2.1) and in the HLT (Section 18.2.2). The

Table 18-1 Software development schedule for the software of Online Software, DF and HLT

RELEASE	Online Software	DataFlow	High Level Trigger
TDR release (Jun 03)	As described in TDR - Mar 03	As described in TDR, first full DataFlow - May 03	Current EF release - Summer 03
Combined test beam (Summer 04)	Prototype versions of control, configuration, and monitoring services - Feb 04	Consolidation of TDR release, evolution of ROD Crate DAQ - Apr 04	EF infrastructure (version from PDR) - May 04 (LVL2 infrastructure - late summer 04)
Sub-detector readout (Autumn 04)	Full support of final ROD Crate DAQ - Jun 04	First release of final Dataflow, including ROD Crate DAQ - Dec 04	
HLT/DAQ installation (Autumn 05)	Full functionality of control, configuration, and monitoring services, including I/F to conditions DB - Apr 05	Consolidation of previous release, completion of DataFlow functionality - Jun 05	Completion of HLT functionality for cosmic-ray run - Summer 05
Cosmic-ray run (Autumn 06)	Final large-scale performance and support for partitioning - Mar 06	Final large-scale release - Jun 06	Ready for cosmic-ray run - Summer 06
LHC start-up (Apr 07)	Final implementation ready for tuning - Dec 06	Consolidation of previous release - Dec 06	Ready for initial physics run - Dec 06

Detector Control System workplan is outlined in Section 18.2.3, while Section 18.2.4 lists some of the other issues that will be addressed.

18.2.1 DataFlow workplan

In the baseline design, the flow of data and of control messages between the ROS and the HLT can be implemented using bus-based or switch-based data collection, for the aggregation of data from a number of Read Out Links into each port of the main DataFlow network. Work up to the Final Design Review (FDR) of the custom hardware in the DataFlow (first quarter of 2004) will address, with high priority, the optimization of the data flow at the ROS level.

As described in Chapter 8, a full-system prototype has been developed supporting simultaneously the two data-aggregation techniques, by means of a ROBin prototype with two Read Out Links at the input, and output to both PCI-bus and GEth (see Section 8.1.3.2). Performance measurements done on a prototype of the full HLT/DAQ architecture with a size of order 10% of the final system, as reported in Chapter 8 and Chapter 14, demonstrate the overall viability of the baseline architecture. The measurement programme is continuing with the deployment of a number of ROBin modules in the 10% prototype system, allowing a direct comparison of functional and performance aspects. This will lead to the choice of the optimal Input/Output path before the FDR of the ROBin, currently planned for the first quarter of 2004.

The results of the measurements will continue to be used in the discrete event modelling aimed at providing a description of the behaviour of the system scaled to the final size.

18.2.2 High-Level Trigger workplan

The work on the HLT system up to the end of 2003 will be focused on several principal areas which are discussed below.

The LVL2 and EF testbeds, discussed in Section 14.2, will be exploitated during the summer period in order to arrive at a complete set of measurements concerning the functionality and performance of the HLT event-selection software. This information, together with the initial results of physics-performance studies using the selection algorithms running in the selection-software framework, will be analysed in an internal review currently planned for late October 2003. The review will identify areas in the software which may require optimization, partial re-implementation, or re-design. The review will define the end of the initial cycle — requirements analysis, design, implementation, testing, and exploitation — which characterizes modern software projects. The second, shorter, cycle will be launched immediately following the review with a view to completing it a year later. During this time, the system-exploitation phase will include its use in the combined ATLAS test-beam during the summer of 2004.

In parallel with the testbed exploitation, further optimization studies of the selection-software design and implementation will be done, as discussed in Section 14.2.1. The results of these studies will provide important input to the review.

Data are available from the production of large samples of simulated physics channels with noise and pile-up included. Exploitation of these data using the LVL2 and EF algorithms is now under way in order to optimize the implementation and tune the performance of the selection algorithms in terms of signal efficiency, background-rate reduction, and execution time. This will lead to an update of the numbers presented in Refs. [18-2], [18-3] and [18-4]. Initial results are presented in this TDR. This work will continue for the rest of this year and beyond.

The HLT system software (used to control the flow of data to and from the HLT algorithms, and for the control and supervision of the HLT) is being tested in the testbeds and also, currently (in the case of the EF), in testbeam. The design and implementation of these software systems will also be reviewed towards the end of 2003 with a view to having an updated design and implementation by the middle of 2004.

The purchase of the bulk of the HLT computing and network hardware will take place at as late a date as possible compatible with ATLAS installation and commissioning requirements as well as the LHC start-up schedule. This is done to benefit from the best possible price / performance ratio of the equipment. The standard sub-farm racks, discussed in Section 5.5.11, will need to be prototyped. This work will begin later this year with a view to having a first rack prototype in 2004.

As mentioned notably in Sections 1.5 and 5.2.3.1.4, aspects of access to and use of the conditions database, and connection of the TDAQ system to the offline prompt reconstruction and to CERN mass storage system are not yet sufficiently defined. These issues have already begun to be addressed with the offline computing group, and this will continue into next year.

18.2.3 Detector Control System workplan

The DCS schedule is summarized in Figure 18-1.

18.2.3.1 Front-End

The process of producing the ELMBs has started. At the beginning of the third quarter of 2003 the radiation qualification of the components will be done, and fully-tested modules should be ready at the end of the last quarter of 2003. The software for the ELMB is being finalized and will be tested in the third quarter of 2003 before being loaded into the ELMB as part of the production process.

18.2.3.2 Back-End

The Framework Software that is built on top of the SCADA software PVSS is being reviewed and the implementation of a re-designed version will start at the beginning of the third quarter of 2003 in the framework of the JCOP project. The core libraries and tools are expected by the end of November 2003. This final software will be used for the test-beam operation in 2004, at the ATLAS pit for controlling the sub-detectors, and the experimental infrastructure from mid-2004 onwards. In the beginning, each system will operate in stand-alone mode. Integrated operation will be needed from the end of 2005 onwards. On this timescale, the overall supervisory system for coherent operation of the whole ATLAS experiment from the main control room will be set up. Communication with external systems will be needed in 2006 in order to allow global commissioning of ATLAS, and for the cosmic-ray run in the third quarter of 2006.

18.2.4 Other issues to be addressed

During the deployment of the full-system HLT/DAQ prototypes for measurements and optimization of performance, a number of other issues will be addressed that are important for the system functionality and the definition of certain services. Amongst these issues, the following have so far been identified:

- processor and process management in the HLT farms;
- flow of data in databases (for production, conditions, configuration);
- fault tolerance;
- TDAQ output and computing model viz., offline prompt reconstruction and mass storage;
- use of remote EF sub-farms;
- system scalability and robustness against variation of parameters, eg LVL2 rejection power.

As an illustration, the last of the these points is developed briefly here. The feasibility of the architecture depends on a number of assumptions regarding both external conditions (such as the data volume of a region of interest) and extrapolations of measurements performed today. Further study of the robustness of the baseline architecture with respect to reasonable, and maybe foreseeable, variations of these assumptions will be an item for the short-term post-TDR workplan.

Table 18-2 lists some of the key parameters for the architecture. For each parameter, the currently assumed value is given, and consequences of the variations are discussed.

Table 18-2 Variations of baseline parameters

Parameter	Current value	Remarks
RoI data volume	~2%	Implications on ROL–ROB aggregation factor, ROB–ROS aggregation factor, and LVL2 switch size.
RoI request rate/ ROB	Uniform distribution	This will not be the case. Non-uniform distributions have been studied extensively in Paper Models. Their implications on the traffic pattern through the LVL2 network need further studies.
LVL2 acceptance	30:1	A more pessimistic figure implies a higher rate into the EB or a reduction in the maximum LVL1 rate that can be accepted, hence an effect on the ROL and ROB aggregation factors and the number of SFIs. Thus a large EB network, as well as a related impact (higher EB rate) on the EF farm size.
HLT Decision time/event	10 ms @ LVL2 1 s @ EF	Variations of O(10%) have a corresponding effect on the size of the farms and the related central (LVL2 or EB) networks.
SFI Input/Output capability	70 Mbyte/s In 70 Mbyte/s Out	Impact the size of the EB network (because of additional SFIs) Impact the organization of the EF farm. Fewer events output by the SFI will mean smaller sub-farms (or more SFIs per sub-farm)

18.3 Commissioning

During detector installation, the corresponding detector readout elements will need to be tested and commissioned. This section presents the strategy developed so that detector commissioning requirements can be met in parallel with the commissioning of the HLT/DAQ elements themselves.

A more detailed description of the commissioning plan can be found in the relevant documentation of the ATLAS Technical Coordination [18-5].

18.3.1 Detector commissioning

Detector commissioning will be performed in three phases, namely the readout of a single ROD crate, the readout of multiple ROD crates, and the readout of multiple sub-detectors. The necessary tools for the implementation of such a strategy are briefly described here. Some of these tools are already available today and used at the test beam, in test beds, and at test sites.

18.3.1.1 Tools for detector commissioning

18.3.1.1.1 ROD-Crate DAQ

The first need of a sub-detector will be to check the functionality of the front-end electronics via the readout FELs into ROD modules. For this first functionality check, the ROD-Crate DAQ (RCD) will be used. The RCD software is being completed now and its exploitation on test beam operations is starting. For the initial phase of detector commissioning, it will be a mature and stable product.

The RCD is the minimal infrastructure needed to read out the sub-detectors' RODs. The readout will be done initially via VMEbus, by the ROD Crate Controller (RCC), as shown in Figure 18-2. Data processing can be done at the level of the RCC or in an external workstation (connected via Ethernet to the RCC) running the ROS application. Data in the workstation can then be stored to disk. The necessary infrastructure will be in place in the underground counting room, including the DCS, the TTC, the Local Trigger Processors, and the Conditions Database for the storage of calibration data.

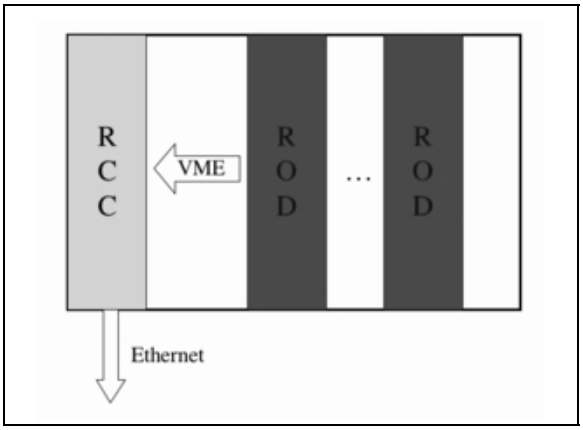

Figure 18-2 Readout of ROD modules via VMEbus by the ROD Crate Controller

The second step will implement the ROD readout via the standard Read Out Link (ROL) into a stand-alone ROS, as in Figure 18-3. This setup will enable the data integrity over the ROLs to be checked simultaneously for a number of RODs and is the first major step in the detector–DAQ integration. This setup is very similar to the one already in use and well tested at the H8 ATLAS test beam.

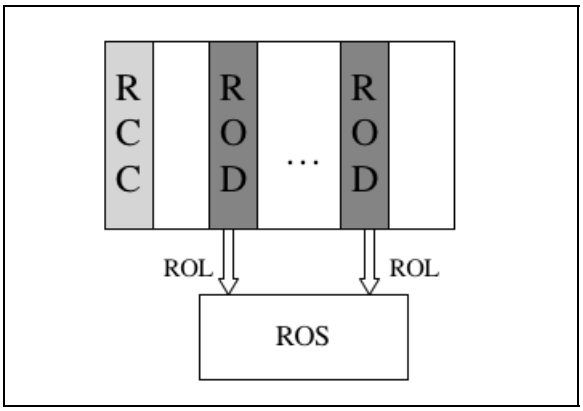

Figure 18-3 Readout of ROD Modules via ROLs connected to a minimal ROS system

18.3.1.1.2 Readout of multiple ROD crates

In the second phase, data taking from multiple ROD crates will be implemented. This will allow the complete readout of one or more TTC partitions, requiring multiple ROS units and a minimal Event Builder. In all cases the storage of the data to disk or to the Conditions Database will be allowed (Figure 18-4).

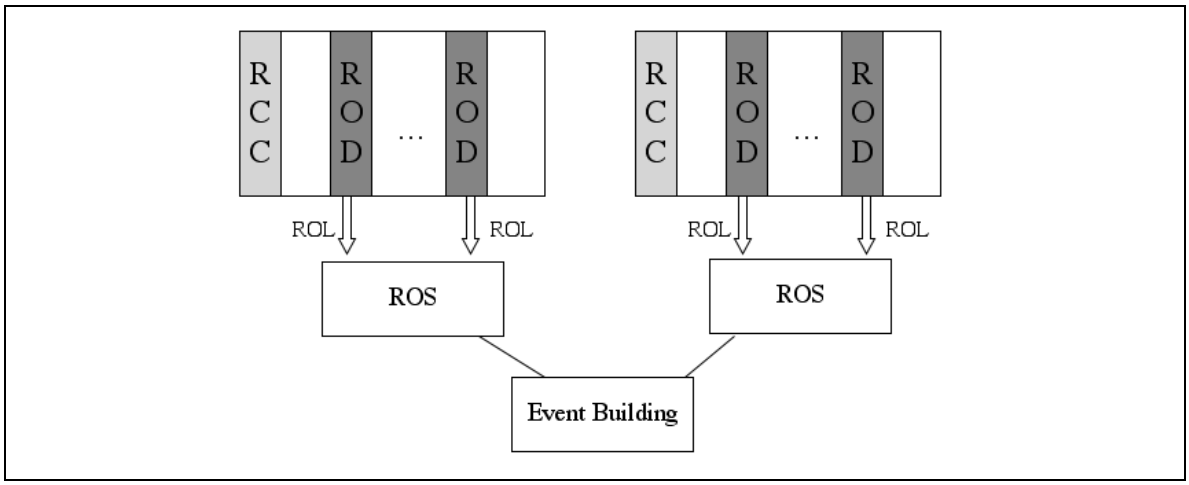

Figure 18-4 Readout of multiple ROD crates via ROLs with a minimal Event Building infrastructure

18.3.1.1.3 Readout of multiple sub-detectors

In preparation for the cosmic-ray run and during the final phase of the ATLAS Commissioning, simultaneous readout from several sub-detectors will be necessary. The time scale for these operations matches the time scale for the completion of installation of the final TDAQ elements. A possible configuration for the readout of more than one sub-detector is very similar to the one presented in Figure 18-4. The number of hardware elements involved may vary significantly; however, the major change will be the addition of Event Filter sub-farms (only needing minimal processing power) to complete the dataflow chain. In the case of multiple sub-detector readout, the CTP infrastructure and the DCS supervisor will also be needed.

18.3.2 HLT/DAQ Commissioning

HLT/DAQ commissioning will be performed in parallel with detector commissioning. As portions of the HLT/DAQ systems are commissioned, they will provide resources for detector commissioning.

The HLT/DAQ commissioning will start with the installation and commissioning of a vertical slice of the system, followed by the expansion of the parallelism of the system as needed to support detector installation, commissioning, and HLT processing requirements for commissioning.

The strategy for commissioning the HLT selection software is being developed as part of the development of the ATLAS plan for detector commissioning with physics data. Development of this plan has recently started in concert with the ATLAS detector and physics groups.

18.3.2.1 Tools for HLT/DAQ commissioning

HLT/DAQ commissioning will require an appropriate set of diagnostic tools. The diagnostic tools used for commissioning will generally be the same as those used routinely to test and to troubleshoot the system during future operations. They will include end-to-end tests of the readout chain, as well as the diagnostics necessary to localize failures of the replaceable component. The diagnostic tests fall into three categories: connectivity or communications tests, functionality tests, and performance or stress tests. In order to provide test data for diagnostic and commissioning tests, components will implement test buffers at their inputs and at their outputs. The suites of diagnostic tests that will be required for commissioning are being developed now for testing and evaluation of HLT/DAQ test beds, for data acquisition, and for detector beam tests.

18.3.2.2 Preparations for HLT/DAQ commissioning

Smooth commissioning of the ATLAS HLT/DAQ systems will be greatly facilitated by proper preparation. In addition to development of an appropriate set of diagnostic tools, as described above, preparations for commissioning will include rigorous testing of all hardware and software before installation in the system. All hardware and software designs will be integrated into sub-systems and into complete systems, and will be validated on test beds prior to installation. Every hardware component will undergo rigorous production acceptance testing in the laboratory before installation. These preparations will minimize the number of failed components and unforeseen problems experienced during the commissioning process.

18.3.2.3 Sequence of HLT/DAQ commissioning

The first phase of HLT/DAQ commissioning will consist of sequentially installing subsystems necessary to build up a vertical slice of the complete system. Following completion of the vertical slice, the HLT/DAQ system will be expanded to provide the online software and dataflow services necessary to support the growing needs of detector commissioning. The HLT processing capabilities will also be expanded in preparation for ATLAS cosmic-ray running in autumn 2006 and LHC start-up in spring 2007. The HLT/DAQ commissioning sequence ties in with the requirements for detector commissioning at several points.

18.3.2.3.1 Establishing an HLT/DAQ vertical slice

The first step in building up an HLT/DAQ vertical slice will be to install and commission the minimal complete set of Online Software functionality. This step will provide the configuration, control, and monitoring functionality that will be used for subsequent steps of HLT/DAQ commissioning. It will also provide the functionality required for detector commissioning with ROD-Crate DAQ (see Section 18.3.1.1), including connection to the Conditions Database and the DAQ–DCS Connection (DDC).

The second step in HLT/DAQ commissioning will be to install and commission a minimal dataflow system. This system will first be commissioned without connection to detector front-end electronics. Subsequently, ROSs for the first sub-detector system will be connected (as depicted in Figure 18-3), and the detector–DAQ interface commissioned. Detector commissioning with readout of multiple ROD crates (as depicted in Figure 18-4) can then proceed, including the ability to log data to disk via an SFI. Detectors can also commission independently of, and in parallel with, other detectors using the partitioning feature of HLT/DAQ.

After commissioning the detector–DAQ interface, components of the LVL1 Trigger will be connected to a ROS and to the RoI Builder, the CTP will be connected to the TTC, and the LVL1–HLT/DAQ interface will be commissioned. Following this step, readout of multiple sub-detectors, coordinated by the LVL1 trigger, can proceed (as described in Section 18.3.1.1.3). Detectors can also continue to commission independently of one another using the partitioning feature of HLT/DAQ. Integration of an Event Filter subfarm will complete the commissioning of the HLT/DAQ vertical slice.

18.3.2.3.2 Building out the HLT/DAQ system

During the second phase of HLT/DAQ commissioning, the vertical slice will be expanded by commissioning additional parallelism, that is, incorporating additional components, e.g. ROSs, SFIs, etc. The additional parallelism will be added as and when sub-detectors are installed and ready for testing. By Summer 2006, the HLT/DAQ system will have reached the scale required for testing ATLAS with cosmic rays. At each step in the expansion of the system, additional components will be commissioned, and the functionality of the expanded system tested.

Once the vertical slice is complete, performance and stress testing will be initiated. Performance tests will confirm that the required system performance is achieved as the size of the system is increased. Stress tests will be conducted by operating the system at high levels of performance for sustained periods of time. Such tests will have previously been performed on test beds, but on a limited scale. Behaviour on scales larger than the largest test bed will be encountered for

the first time during the commissioning phase. Any unforeseen performance behaviour will need to be rapidly understood and remedied.

18.3.2.3.3 Final steps in HLT/DAQ commissioning

Commissioning of the HLT/DAQ system will include validation of the timing of all detector electronics systems with respect to the DataFlow system. The procedure for establishing the timing of individual detector electronics systems via the TTC system is described in the TDR for the LVL1 Trigger. During commissioning, it must be established that event fragments from a triggered event are coherently labelled, i.e. with the same bunch crossing and LVL1 identifiers, by all parts of all detectors. This validation will use built events triggered by cosmic rays that traverse a pair or group of sub-detectors. The data contained in event fragments from each detector will be matched with the data in fragments from other detectors. Final validation of coherent timing will occur by reconstructing events triggered during colliding beams.

A recent ATLAS workshop was dedicated to analysing the detector performance and physics commissioning needs of the experiment. More info can be found at [18-6].

18.4 References

18-1 ATLAS Installation,
http://cern.ch/Atlas/TCOORD/Activities/TcOffice/Scheduling/Installation/ATLASinstallation.html

18-2 *ATLAS High-Level Triggers, DAQ and DCS Technical Proposal*, CERN/LHCC/2000-17 (2000)

18-3 S. Tapprogge, *Physics Requirements for the ATLAS Trigger,* Presentation to LHCC Meeting, March 2002,
http://agenda.cern.ch/askArchive.php?base=agenda&categ=a02381&id=a02381s1t2/transparencies

18-4 S. Tapprogge, *ATLAS Rates*, Presentation to LHCC Meeting, May 2002,
http://agenda.cern.ch/askArchive.php?base=agenda&categ=a02443&id=a02443s1t2/transparencies

18-5 ATLAS Technical Co-ordination, *ATLAS Commissioning - Sub-Detectors needs for TDAQ readout*, ATC-TD-IN-0002 (2003)
https://edms.cern.ch/file/375183/

18-6 Detector performance and physics commissioning with physics data,
http://cern.ch/gianotti/commissioning.html

A Paper model results

The ‘paper model’ is a static model of the HLT/DAQ system used for estimating rates and processing requirements. On account of the RoI-driven nature of the LVL2 trigger and the different processing sequences and associated data request patterns, a simple calculation of the quantities of interest on the basis of the LVL1 and LVL2 accept rates and average event fragment sizes is not feasible. In previous studies a spreadsheet was used. Because it was difficult to modify parameters such as trigger menus and processing sequences, to extract the results, and to check the correctness of the computing procedures implemented, the spreadsheet has been replaced by a program written in C++. All parameters are now specified in separate input files and are therefore easy to modify. The results to be output by the program (an array of tables in ASCII format) are selected by configuration parameters in an input file, and the program source provides a clear overview of the computing procedures involved in the modelling process itself.

A.1 Input

The LVL1 trigger defines a finite number of possible RoI locations. A small region in eta-phi space corresponds to each location. For each location the rate of hits satisfying appropriate LVL1 trigger criteria is assumed to be proportional to the area of the region associated with the location and not to depend on its coordinates in eta-phi space. The sum of the rates for all possible locations is equal to the LVL1 menu RoI rate, so that the RoI rate per location can be determined. Along with information on the sizes of the regions-of-interest (see Table A-1), and the mapping of the detector on the ROLs, the RoI rate for each ROL can be obtained.

Table A-1 LVL2 RoI sizes

Type of RoI	Size in η	Size in ϕ
EM	0.2	0.2
Jet	0.8	0.8
Tau	0.2	0.2
Muon	~0.3–0.4 (depends on detector)	~0.1–0.4 (smallest in muon and in inner detector)

The detailed breakdown of the exclusive ‘nominal’ rates for the different LVL1 trigger menu items for start-up luminosity are presented in detail in Section 13.5. The total rates are ~25 kHz at start-up luminosity and ~40 kHz at design luminosity [A-1]. These rates include a contribution of ~5 kHz for pre-scaled and calibration triggers which are not considered further in the model[1]. Estimates of the fragment sizes are presented in Table A-2. To calculate the bandwidth requirements, the fragment sizes have been increased to include headers and trailers.

The LVL2 processing consists of several steps, and after each step a decision is taken on whether data from other sub-detectors within the RoI should be requested for further analysis. For the

1. The number of 5 kHz is an estimate. The effect of these triggers on the ROB access rates and the event-building rates will be studied when a better understanding of their composition is available from the detectors. Only a small percentage will be passed through the event-building stage.

Table A-2 Estimate of data fragment sizes per ROL for the various sub-detectors. These are given with the number of ROLs for each of the sub-detectors, for low and design luminosity. These numbers are used as input for modelling.

		Fragments size per ROL (byte)	
Sub-detector	**Number of ROLs**	**Low luminosity (2×10^{33} cm^{-2} s^{-1})**	**Design luminosity (1×10^{34} cm^{-2} s^{-1})**
Pixels	120	200	500
SCT	92	300	1100
TRT	232	300	1200
E.m. calorimeter (LAr Barrel and EMEC)	724	752	752
FCAL	16	752	752
Hadron calorimeter	64 (Tilecal)	752	752
	24 (HEC)	752	752
Muon precision	192	800	800
Muon trigger (RPCs and TGCs)	48	380	380
CSC	32	200	200
LVL1	56	1200 (average)	1200 (average)
Total event size, raw (Mbyte)		1.0	1.3
Total event size, with headers (Mbyte)		1.2	1.5

four different types of RoIs, the sub-detectors from which data are needed as input for the different processing steps are specified in Table A-3. The acceptance factors associated with the processing steps are also specified in the table. In combination with the RoI rates per ROL, the rates of requests for RoI data sent by the LVL2 trigger to the buffers receiving data from the ROLs, can be calculated. The trigger acceptance factors in Table A-3 determine the overall LVL2 accept rate and therefore the event building rate of ~600 Hz at the nominal LVL1 accept rate for start-up luminosity. A reliable estimate for the final LVL2 accept rate for design luminosity is not yet available, but for modelling purposes, 1.2 kHz at nominal LVL1 rate (and therefore 3.5 kHz at a 100 kHz LVL1 accept rate) has been chosen as a working hypothesis for these studies. However, what is more relevant here are the modelling estimates for processing resources required for the initial ATLAS data-taking at start-up luminosity, where the modelling input is better understood.

In order to estimate the processing resources needed for the LVL2 trigger, the algorithm execution times and the overheads for sending requests to, and receiving data from the ROS, are needed. The following typical numbers, assuming execution on 8 GHz dual-CPU machines, have been used in the model for each of the detectors; Muons — 6.5 ms, Calorimeters — 3.0 ms, TRT — 7.2 ms (15.3 ms for design luminosity), and SCT / Pixels — 6.7 ms (7.9 ms for design luminosity). The numbers specified include an initial estimate for data preparation for each sub-detector and RoI (see Section 14.2.2). Furthermore, for each message sent or received an overhead of 10 µs is included. The processing step resulting in a decision is assumed to take 50 µs.

Table A-3 Sub-detector data requested by different processing steps of the LVL2 trigger for the different types of RoIs and associated acceptance factors. The acceptance factors are relative to the LVL1 RoI rate.

Type of RoI	First step	Acceptance factor	Second step	Acceptance factor	Third step	Acceptance factor
EM	E.m. calorimeter	0.19 (design lum.: 0.16)	Hadron calorimeter	0.11 (design lum.: 0.16)	TRT /SCT/ Pixels Higher & sharper threshold	0.012 (electron) 0.006 (photons)
Jet	E.m. and hadron calorimeters	1.0			Higher & sharper threshold	0.05
Tau	E.m. and hadron calorimeters	0.20	TRT /SCT/ Pixels	0.10	Higher & sharper threshold	0.033
Muon	Muon precision and trigger detectors	0.39	SCT/Pixels	0.18	E.m. and hadron calorimeters	0.10

Merging of event fragments into a larger fragment suitable for input to the algorithms is assumed to proceed at 160 Mbyte/s. TRT data is assumed not to be analysed for muon RoIs.

A.2 Results

The LVL2 system and the Event Builder both send requests for data to the ROS. The rate of requests from the Event Builder for data from a single ROL, buffered in the ROS, is equal to the event-building rate. The LVL2 request rate for data from a single ROL for a given sub-detector is not the same for all ROLs, as the rate depends on the mapping of the detector onto the ROLs, the possible RoI locations defined by LVL1, and on the RoI sizes. The average request rate for data from a single ROL and the largest individual request rate, calculated as outlined in the previous section, are presented in Table A-4 (for 100 kHz LVL1 accept rate). The average and largest individual volume of data requested per ROL by the LVL2 trigger and by the Event Builder for a given sub-detector are presented in Table A-5. Similar results are given in Table A-6 and Table A-7 for bus-based ROS units (see Section 5.5.4) handling data from 12 ROLs[1]. Fragments of the same event input via different ROLs can be requested by a single message from a bus-based unit (see Chapter 5), and are concatenated and output as a single message. Data Collection and Raw Ethernet wrappers have been taken into account in the data volumes output by the ROS units.

Results for the processing resources required for the LVL2 trigger, for the number of SFIs, and for the rates per L2PU and SFI are presented in Table A-8 for a 100 kHz LVL1 rate. The number of SFIs has been obtained by requiring that the input data volume be no larger than 60 Mbyte/s. As shown in Chapter 8, this data volume can be accommodated with Gigabit Ethernet. The processing resources required for the LVL2 trigger have been calculated using the rates,

1. For each sub-detector, groups of 12 ROLs have been formed, without regard for the partitioning of the sub-detector; for cases where the number of ROLs is not a multiple of 12, the ROS unit with less than 12 ROLs connected has not been included in the calculation of the averages.

processing times, and acceptance factors presented in the previous section. The number of LVL2 processors obtained also assumes a 77% CPU utilization. It should be emphasized that the reliability of these numbers is determined by the quality of the input parameters used. The modelling gives similar results, based on current hypotheses for input parameters, at design luminosity, for the LVL2 processing requirements and rates per L2PU. At design luminosity the number of SFIs increases to 87 due to the increase in event size.

Table A-4 LVL2 request rate (in kHz) for data from a single ROL for a 100 kHz LVL1 accept rate

Luminosity / rate	Muon precision	Muon trigger	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low / average	0.10	0.22	2.03	1.36	0.19	0.53	0.67
Low / maximum	0.20	0.30	7.42	1.99	0.22	0.74	0.98
Design / average	0.29	0.62	1.74	0.89	0.04	0.79	0.99
Design / maximum	0.57	0.86	6.34	1.30	0.05	1.08	1.42

Table A-5 Data volume (in Mbyte/s) requested per ROL (LVL2 data and data sent to the Event Builder) for a 100 kHz LVL1 accept rate

Luminosity / rate	Muon precision	Muon trigger	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low / average	2.79	1.55	4.29	3.72	1.28	1.42	1.11
Low / maximum	2.88	1.59	8.90	4.26	1.29	1.50	1.20
Design / average	3.40	1.99	4.46	3.74	4.59	5.13	2.70
Design / maximum	3.66	2.10	8.40	4.09	4.60	5.49	2.96

Table A-6 LVL2 request rate (in kHz) per ROS unit (12 ROLs) for a 100 kHz LVL1 accept rate

Luminosity / rate	Muon precision	Muon trigger	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low / average	0.9	1.9	12.0	9.4	1.0	3.9	5.1
Low / maximum	1.4	2.5	21.5	10.5	1.3	4.3	7.6
Design / average	2.5	5.5	10.4	6.2	0.2	5.7	7.5
Design / maximum	4.1	7.1	18.6	6.9	0.3	6.3	11.2

Table A-7 Output data volume (in Mbyte/s) per ROS unit (12 ROLs) (LVL2 data and data sent to the Event Builder) for a 100 kHz LVL1 accept rate

Luminosity / rate	Muon precision	Muon trigger	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low / average	34.6	19.4	54.0	46.9	16.1	18.2	14.5
Low / maximum	35.2	19.8	70.1	50.3	16.2	18.9	15.5
Design / average	42.3	25.2	56.0	46.8	56.5	64.3	34.1
Design / maximum	44.1	26.1	69.7	49.0	56.6	66.9	36.7

Table A-8 Overview of paper model results concerning L2PUs and SFIs for a 100 kHz LVL1 trigger rate at low luminosity

LVL2 farm size (8 GHz dual-CPUs)	495
Fragment rate in = request rate out per L2PU (kHz)	1.8
Event data volume in per L2PU (Mbyte/s)	3.2
Decision rate per L2PU (kHz)	0.20
Number of SFIs required for 60 Mbyte/s input per SFI	59
Event building rate per SFI (Hz)	51

A.3 References

A-1 S. Tapprogge, *Physics Requirements for the ATLAS Trigger,* Presentation to LHCC Meeting, March 2002, http://agenda.cern.ch/askArchive.php?base=agenda&categ=a02381&id=a02381s1t2/transparencies

B Glossary

The glossary has been split into two sections, one with acronyms and their meaning, and another with actual definitions of some terms.

B.1 Acronyms

API	Application Program Interface
ASIC	Application Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
BC	Bunch Crossing
BCID	Bunch Crossing Identifier
BCR	Bunch Counter Reset
BE	Back-End
CAN	Controller Area Network
CBR	Constant Bit Rate
CBQ	Class Base Queuing
CERN	European Laboratory for Particle Physics
CF	Connect Forum
CFS	Complex Front-end Systems
CIC	Common Infrastructure Controls
CMA	Coincidence Matrix
CMC	Common Mezzanine Card
CMT	Configuration Management Tool
CondDB	Conditions Database
ConfDB	Configuration Database
CORBA	Common Object Request Broker Architecture
COTS	Commodity / Commercial Off-The-Shelf
CP	Cluster Processor
CSC	Cathode Strip Chamber
CTP	Central Trigger Processor
DAL	Data Access Library
DAQ	Data Acquisition System
DBMS	Database Management System
DC	Data Collection
DCS	Detector Control System

DDC	DAQ-DCS Communication
DDC-CT	DDC Control Transfer
DDC-DT	DDC Data Transfer
DDC-MT	DDC Message Transfer
DF	DataFlow System
DFM	Data Flow Manager
DID	Destination Identifier
DIG	Detector Interface Group
DMA	Direct Memory Access
DSA	Diagnostics Supervision Agent
DSP	Digital Signal Processor
DSS	Detector Safety System
DVS	Diagnostics and Verification System
EB	Event Builder
EBN	EB Network
ECAL	Electromagnetic Calorimeter
ECR	Event Counter Reset
ED	Event Dump
EDM	Event Data Model
EF	Event Filter
EFD	Event Filter Data flow
EFL	Event Format Library
EFN	EF Network
EFPU	EF Processing Unit (processor)
EH	Event Handler
EL1ID	Extended Level-1 ID
ELMB	Embedded Local Monitor Board
EMB	Electromagnetic Barrel
EMEC	Electromagnetic Endcap
EMS	Event Monitoring Service
ERS	Error Reporting Service
ESA	European Space Agency
ESS	Event Selection Software
EVS	Event Viewing System
FC	Flow Control

FCAL	Forward Calorimeter
FDR	Final Design Review
FE	Front-End
FEC	Front-End Controller
FEL	Front-End Link
FILAR	Four Input Links for ATLAS Readout
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GCS	Global Control Station
GID	Global event Identifier
HEC	Hadronic Endcap Calorimeter
HLT	High Level Trigger
HMI	Human Machine Interface
HOL	Head Of Line
HOLA	High-speed Optical Link for ATLAS
ID	Inner Detector
IDC	Identifiable Container
IGUI	Integrated Graphical User Interface
ILU	Inter-Language Unification system
IOV	Interval Of Validity
IP	Interaction Point
IP	Internet Protocol
IPC	Inter-Process Communication
IPC_REF_FILE	IPC Reference File
IS	Information System
JCOP	Joint Controls Project
JDBC	Java Database Connectivity
JEP	Jet Energy Processor
L1A	LVL1 accept
L1ID	LVL1 Trigger Accept Identifier
L2N	LVL2 Network
L2P	LVL2 Processor
L2PU	Level-2 Processing Unit (application)
L2SV	Level-2 Supervisor
LAr	Liquid Argon

LAN	Local Area Network
LBSF	Local Buffering and Storage Facility
LCG	LHC Computing Grid
LCS	Local Control Station
LDC	Link Destination Card
LHC	Large Hadron Collider
LVL1	Level-1 trigger system
LVL2	Level-2 trigger system
LSC	Link Source Card
LTP	Local Trigger Processor
LUT	Look-Up Table
MAC	Media Access Control
MDT	Monitored Drift Tube
MRS	Message Reporting System
MSSM	Minimal SuperSymetric Model
MTTF	Mean Time To Failure
NIC	Network Interface Card
OBK	Online Book Keeper
ODBC	Open Database Connectivity
OHS	Online Histogramming Service
OKS	Object Kernel Support
OLE	Object Linking and Embedding
OMG	Object Management Group
OPC	OLE for Process Control
OSF	Online Software Farm
OSN	Online Software Network
PCI	Peripheral Component Interconnect
PDR	Preliminary Design Review
PESA	Physics and Event Selection Architecture
PLC	Programmable Logic Controller
PMG	Process Manager
PP	Pre-Processor
pROS	pseudo-ROS
PRR	Production Readiness Review
PSC	PESA Steering Controller

PT	Processing Task
QoS	Quality of Service
RC	Run Control System
RCC	ROD Crate Controller
RCM	ROD Crate Module
RCP	ROD Crate Processor
RCW	ROD Crate Workstation
RDB	Remote Database
RDO	Raw Data Object
RIO	Reconstruction Input Object
RM	Resource Manager
ROB	Read-Out Buffer
ROBin	Read-Out Buffer input
ROC	Read-Out Crate (Specific implementation of a ROS)
ROD	Read-Out Driver
RoI	Region of Interest
RoIB	Region of Interest Builder
ROL	Read-Out Link
ROS	Read-Out Sub-system
RPC	Resistive Plate Chamber
RRC	ROD to ROB Connection
RRM	ROB to ROS Multiplexer
RT	Real Time
RUP	Rational Unified Process
SCADA	Supervisory Control and DAQ
SCS	Sub-system Control Station
SCT	Silicon Tracker
SCX	Surface control room
SDP	Software Development Process
SDX	Surface counting room
SERDES	SERial DESerialiser
SFC	Sub-Farm Crate
SFI	Sub-Farm Input
SFO	Sub-Farm Output
SID	Source Identifier

S-LINK	Simple Link Interface
STL	Standard Template Library
STP	Spanning Tree Protocol
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TDAQ	ATLAS Trigger / DAQ / DCS
TDR	Technical Design Report
TES	Transient Event Store
TGC	Thin Gap Chamber
TM	Test Manager (also TMGR)
TMGR	Test Manager
TOF	Time Of Flight
TRG	Trigger Module (function inside present implementation of ROS)
TRT	Transition Radiation Tracker
TTC	Timing, Trigger and Control (TTC)
TTCrx	TTC Receiver
TTCvi	TTC VME Interface
UDP	User Datagram Protocol
URD	User Requirements Document
US15	Underground service area
USA15	Underground counting room
UX15	Experimental cavern
VLAN	Virtual Local Area Network
WRR	Weighted Round Robin
XML	Extensible Markup Language

B.2 Definitions

Bunch Crossing Identifier (BCID)

Number that defines the bunch crossing at which an event occurred. Potential bunch crossings are numbered 0 to 3563 per LHC orbit, starting with the first following the LHC extractor gap.

Bunch Counter Reset (BCR)

Signal Broadcast by the TTC system once per LHC orbit to control the phase of local bunch counters.

Back-End (BE)

Part of the DCS system furthest from the detector (as opposed to Front-End)

Central Trigger Processor (CTP)

The place where the LVL1 trigger is generated.

Conditions database (CondDB)

The conditions database contains the record of the detector conditions required for data analysis, e.g. calibration and geometry constants.

Configuration databases (ConfDB)

The configuration databases store the parameters necessary to configure the TDAQ system's architecture, hardware and software components, and running modes.

TDAQ Run or Run

A continuous period in time of data taking using a given hardware and software configuration and a defined set of run parameters. It is identified by a unique run number. The run begins when the TDAQ, detectors and other sub-systems are correctly configured and the machine conditions are acceptable. A run terminates either cleanly when the pre-defined goals of the run are met (e.g. a certain number of events have been taken) or aborts when a serious unexpected problem occurs (e.g. loose the beam or the machine conditions are unacceptable etc.) or when the configuration of the partition changes.

DataCollection (DC)

DataCollection is a subsystem of the Atlas TDAQ DataFlow system responsible for the movement of event data from the ROS to the High Level Triggers. This includes data from Regions of Interest (RoIs) for LVL2 Processing, building complete events for the Event Filter and finally transferring accepted events to Mass Storage. It also handles passing the LVL1 RoI pointers and the allocation of LVL2 processors and load balancing of Event Building.

DataCollection Framework

A set of services used by all LVL2 and EB applications, which provides a unified program structure and common interfaces to Configuration Database, Run Control and other Online Software services.

Data Flow Manager (DFM)

The DFM orchestrates the correct flow of data fragments between ROSs and SFIs. It is triggered by the L2SV, load balances the event building tasks on the SFIs and ensures that the ROSs do not overflow their internal memory buffers.

Data Flow system (DF)

System comprising the ROS and DC HLT subsystems.

Detector Control System (DCS)

It comprises the control of the subdetectors and of the common infrastructure of the experiment and the communication with the services of CERN (cooling, ventilation, electricity distribution, safety etc.) and the LHC accelerator.

Diagnostic package (DVS)
This element uses the test manager to diagnose problems with the TDAQ system and confirm its functionality.

Event
All ROB fragments from the same beam crossing. Identified by run number and GID after event building.

Event Builder (EB)
Part of the DF system, it merges all the fragments belonging to a unique EL1ID into a full event at a single destination and assigns a GID.

Event Counter Reset (ECR)
Signal broadcast by the TTC system to reset the local event counters.

Event Filter (EF)
The hardware and software required for the final stage of the on-line event selection, data monitoring and calibration using offline style algorithms operating on complete events accepted by LVL2.

Event Filter Dataflow (EFD)
Part of the EF system responsible for the flow of event data within the EF.

Event Filter Farm
The farm of processors in which the Event Filter runs. The same farm may also be used for different purposes, e.g. calibration, by running different software on the farm.

Event Filter Sub-Farm
A sub-set of the Event Filter Farm. Input and output are provided, respectively, by the Sub Farm Input and Output elements.

Event filter supervisor
The hardware and software required to globally control the Event Filter. It is also responsible for the configuration, initialisation, and overall error handling of the Event Filter.

Event fragment
A generic term for a sub-set of event data. Specific instances of an event fragment are ROD, ROB, ROS, and sub-detector fragments.

Event Handler (EH)
The logical object within the EF consisting of an event distributor, an event collector, one or more processing elements, an event handler supervisor and an appropriate communication layer.

Extended Level-1 ID (EL1ID)
The L1ID extended to 32 bits by concatenating an 8 bit ECR counter in the high end bits.

Global event Identifier (Gid)
For a given run, the unique TDAQ wide identifier of an event added to the event during event building.

High Level Triggers (HLT)
Comprised of both the LVL2 and EF, the two ATLAS trigger levels that are implemented primarily in software.

Level-1 Trigger (LVL1)

The ATLAS First Level Trigger system provides a hardware based first trigger decision using only a sub-set of an event's data (Calorimeter and Muon only). Normally only the events accepted by LVL1 are transferred from the detectors into the HLT/DAQ system.

Level-1 Trigger Type

An 8-bit word transmitted with the L1A and giving information about the type of event.

Level-2 Farm

The farm of processors (L2Ps) in which the LVL2 software runs.

Level-2 Sub-Farm

A sub-set of the LVL2 farm.

Level-2 Trigger (LVL2)

The ATLAS Second Level Trigger system is a sub-system of the HLT which provides a software based second stage trigger decision, to reduce the rate of triggers from LVL1 by about a factor of 100. It uses 'Regions of Interest' (RoIs) as given by the LVL1 trigger to selectively read out only certain parts of the ATLAS detector hardware and computes a LVL2 trigger decision.

Local Buffering and Strorage Facility (LBSF)

The short term disc buffering between the SFO and the offline mass storage.

LVL1 Accept (L1A)

LVL1 trigger Accept signal produced by the Central Trigger Processor (CTP) when an event has met the LVL1 trigger criteria. Also in case of partitioned running the local triggers generate L1As.

LVL1 Trigger Accept Identifier (L1ID)

A L1ID is built at different levels of the read-out system. The TTCrx provides a 24-bit L1ID with each L1A signal. In conjunction with the BCID, it defines uniquely an event within a small time frame, of order 1 s.

LVL2 Processing Unit (L2PU)

The L2PU is the application running on one of the Level-2 processors, hosting the HLT processing algorithms and also incorporating the calculations from which the LVL2 trigger decision is derived.

LVL2 Processor (L2P)

A processor within the LVL2 farm.

LVL2 Supervisor (L2SV)

The L2SV is the interface to the LVL1 system via the RoI Builder. It is responsible for distributing events to the LVL2 farm/subfarms and manages the computing resources by means of load balancing algorithms. The L2SV receives the final LVL2 decision from the L2PU. The decision results are communicated to the DFM so that accepted events can be further analyzed, and rejected events can be flushed from the ROS memory.

LVL2 Trigger Type

An 8-bit word transmitted with the LVL2 decision.

Message Reporting System (MRS)

A facility which allows all software components in the ATLAS TDAQ system to report error messages to other components.

Object Kernel Support (OKS)

This is a package which provides a simple active persistent in-memory object manager which is used to implement run-time configuration databases.

Online Software System

All the software for configuring, controlling and monitoring of the TDAQ system. It excludes the management, processing and transportation of physics data.

Partitions (many defs)

See chapter 3.

Process Manager (PMG)

This element performs basic job control of software components of the TDAQ (starting, stopping, and monitoring basic status).

Processing Task (PT)

The software running on an EF processing node in charge of transforming the event data. The transformation may be filtering, reformatting, monitoring or calibration. Several processing tasks can be running concurrently on the same processing node.

pseudo-ROS (pROS)

The pROS is the unit which receives the detailed LVL2 result from the LVL2 Processing Units for each event to be passed to event building. At event building time the pROS is treated just like a ROS, thus including the detailed LVL2 result into the full event.

Read-Out Buffer (ROB)

The buffer which receives data from one ROL.

Read-Out Driver (ROD)

The detector specific Front-End Functional element which gathers data from the derandomizers over one or more data streams and builds ROD fragments of events to be sent to the ROS or RoIB. Gathers data from FEBs; Sends fragments to ROB or RoIB.

Read-Out Link (ROL)

The physical link between ROD and ROS through which the data are sent at the event rate of the LVL1 trigger accept.

Read-Out Sub-system (ROS)

Unit which holds a number of ROBins.

Read-Out System (ROS)

A system of the ATLAS TDAQ comprising the ROS sub-systems and responsible for receiving data from the RODs and for supplying event fragments to the LVL2 and EB sub-systems.

Region of Interest (RoI)

A region limited in eta and phi, indicated by the LVL1 trigger to contain candidates for objects requiring further computation. Some RoI's may be defined internally within the level-2 trigger system.

Region of Interest Builder (RoIB)

The element which combines RoI information from different parts of LVL1 and forwards it to a LVL2 supervisor.

ROD fragment

Data fragment provided by one ROD to the Read-Out System (ROS).

ROS fragment

The set of ROB fragments, within a ROS (in this case sub-system), with the same Gid and to be sent to HLT.

C Participating Institutes

C.1 Authors from participating institutes

The following is the list of institutes which are participating in the ATLAS High-Level Trigger, Data Acquisition, and Detector Control System Collaboration, together with the names of the physicists and senior engineers concerned. Short names are given for each institute; for the official name and location see the opening pages of this document.

Alberta
B. Caron, J. de Jong, J.L. Pinfold, R. Soluk, S. Wheeler

Argonne
R.E. Blair, J. Dawson, J. Schlereth

Barcelona - IFAE
M. Bosman, M. Dosil, K. Karr[1], G. Merino, A. Pacheco, E. Segura

Bern
H.P. Beck, G. Comune, S. Gadomski[2], C. Haeberli, S. Kabana, V. Perez-Reales, K. Pretzl, A. Radu

Bucharest
E. Badescu, M. Caprini

CERN
S. Armstrong, M. Barczyk, J.A.C. Bogaerts, V. Boisvert, D. Burckhart-Chromek, H. Burckhart, M.P. Casado Lechuga, M. Ciobotaru, J. Da Silva Conceicao, J. Cook, P.J. de Matos Lopes Pinto, B. Di Girolamo, R. Dobinson, M. Dobson, N. Ellis, M. Elsing, J. Flammer, D. Francis, S. Gameiro, P. Golonka, B. Gorini, M. Grothe, M. Gruwe, S. Haas, B.I. Hallgren, R. Jones, M. Joos, E. Knezo, G. Lehmann, D. Liko, R. McLaren, T. Maeno, L. Mapelli, B. Martin, C. Meirosu, G. Mornacchi, E. Moyse, C. Padilla, E. Palencia Cortezon, I. Papadopoulos, J. Petersen, A. Poppleton, D. Prigent, S. Rosati, T. Schoerner-Sadenius, S. Stancu, R. Spiwoks, L. Tremblet, E. van der Bij, F. Varela, T. Wengler, P. Werner

Copenhagen
H. Bertelsen, M. Dam, J.R. Hansen, A. Wäänänen

Cracow
Z. Hajduk, W. Iwanski, A. Kaczmarska, K. Korcyl, J. Olszowska, M. Zurek

Geneva
A. Clark, M. Diaz Gomez[3], L. Moneta[4], I. Riu, A. Straessner

Genoa
M. Cervetto, P. Morettini, F. Parodi, C. Schiavi

1. now at Argonne
2. on leave from H. Niewodniczanski Inst. of Nuclear Physics, Cracow
3. now at CERN
4. now at CERN

Hiroshima Institute of Technology
Y. Nagasaka

Innsbruck
B. Epp, V.M. Ghete, E. Kneringer, D. Kuhn, A. Nairz[1], A. Salzburger, A. Wildauer

JINR
I. Alexandrov, S. Korobov, V. Kotov, M. Mineev, V. Tryanin, D. Yanovich

KEK
H. Fujii, A. Manabe, K. Nakayoshi, Y. Watase, Y. Yasu

Lancaster University
M. Smizanska

Lecce
G. Cataldi, G. Chiodini, E. Gorini, M. Primavera, S. Spagnolo

Lisbon FCUL
A. Amorim, N. Barros, D. Klose, J. Lima, C. Oliveira, L. Pedro, J. Villate, H. Wolters

LNF, Frascati
M. Beretta, M.L. Ferrer, W. Liu, I. Sfiligoi

London RHUL
S. George, B. Green, A. Lowe, A. Misiejuk, J. Strong, P. Teixeira-Dias

London UCL
P. Clarke, R. Cranfield, G. Crone, J. Drohan, N. Konstantinidis, M. Muller

Mainz
L. Koepke

Manchester
R.E. Hughes-Jones

Mannheim
C. Hinkelbein, A. Khomich, A. Kugel, R. Männer, M. Müller, M. Yu

Marseille
C. Bee, F. Etienne, C. Meessen, Z. Qian, C. Serfon, F. Touchard

Michigan State
M. Abolins, R. Brock, Y. Ermoline, R. Hauser, B.G. Pope

Moscow SU
N. Nikitin, S. Sivoklokov, L. Smirnova

Nagasaki
M. Shimojima, Y. Tanaka

1. now at CERN

Naples
M. Biglietti, G. Carlino, F. Conventi, L. Merola

NIKHEF
M. Barisonzi[1], H. Boterenbrood, R.G.K. Hart, W.P.J. Heubers, P.P.M. Jansweijer, G.N.M. Kieft, R. Scholte[2], J.C. Vermeulen

Pavia
R. Ferrari, A. Negri, G. Polesello, D.A. Scannicchio, V. Vercesi

Petersburg NPI
V. Filimonov, S. Katunin, A. Kazarov, V. Khomutnikov, S. Kolos, Y. Ryabov, I. Soloviev

Prague AS & CU
J. Chyla, J. Dolejsi

RAL
J.T.M. Baines, D.R. Botterill, W. Li, F.J. Wickens

Rio de Janeiro
A. dos Anjos, M. Losada Maia, A.G. Mello, J.M. de Seixas

Rome I
A. Di Mattia, S. Falciano, C. Luci, L. Luminari, F. Marzano, A. Nisati, E. Pasqualucci

Rome III
P. Branchini, A. Farilla, C. Stanescu

Shinshu
Y. Hasegawa, T. Takeshita

Tel Aviv
H. Abramwicz, Y. Benhammou, E. Etzion

Univ. California Irvine
A.J. Lankford, R. Mommsen, S. Pier, M. Schernau

Weizmann Institute of Science
D. Lellouch, L. Levinson

Wisconsin
S. Gonzalez, W. Wiedenmann, S.L. Wu, N. Xu, H. Zobernig

1. also University of Twente
2. also University of Twente

C.2 Authors from non-participating institutes

The following is the list of authors belonging to institutes which are not participating in the ATLAS High-Level Trigger, Data Acquisition, and Detector Control System Collaboration.

K. Assamagan Brookhaven
D. Cavalli INFN Milano
W. Krasny LPNHE, Université Pierre et Marie Curie, Paris
M. Le Vine Brookhaven
H. Ma Brookhaven
S. Rajagopalan Brookhaven
G. Stavropoulos Berkeley LBNL & UC
S. Tapprogge Helsinki
A. Watson Birmingham
M. Wielers Grenoble

This document has been prepared with Release 6.0 of the Adobe FrameMaker® Technical Publishing System using the Technical Design Report template prepared by Mario Ruggier of the Information and Programming Techniques Group, ECP Division, CERN, according to requirements from the ATLAS collaboration.

To facilitate multiple author editing and electronic distribution of documents, only widely available fonts have been used. The principal ones are:

Running text:	Palatino 10.5 point on 13 point line spacing
Chapter headings:	Helvetica Bold 18 point
2nd, 3rd and 4th level headings:	Helvetica Bold 14, 12 and 10 point respectively
Figure and table captions:	Helvetica 9 point