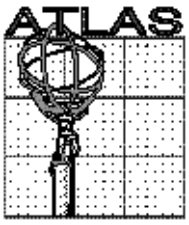

ATLAS
High-Level Trigger, Data Acquisition and Controls

Technical Design Report

Issue:	Final Draft
Revision:	0
Reference:	ATLAS TDR-xx
Created:	12 November 2002
Last modified:	6 May 2003
Prepared By:	ATLAS HLT/DAQ/DCS Group

ATLAS Collaboration

CERN
European Laboratory for Particle Physics (CERN), Geneva

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 Document overview

This Technical Design Report (TDR) for the High Level Trigger (HLT), Data Acquisition (DAQ) and Detector Control Systems (DCS) of the ATLAS experiment builds on the preliminary documents already published on these systems: the Trigger Performance Status Report [1-1], the Trigger DAQ Status Report [1-2] and the HLT/DAQ/DCS Technical Proposal [1-3]. Much background and preparatory work relevant to this TDR can be found referenced in the above documents.

This chapter introduces the overall organisation of the document and gives an overview of the principal system requirements and functions as well as listing the principal data types used in the system.

The document has been organised 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 TDAQ system, the global system operations, and the physics requirements and event selection strategy respectively. Chapter 5 defines the overall architecture of the HLT/DAQ 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 design and performance of the data-flow component which is responsible for the transport of event data from the output of the detector readout links (ROLs) to the HLT system and selected events to mass storage as well as serving data to the HLT system. Chapter 9 explains the decomposition of the HLT into a LVL2 trigger and an event filter component, and 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 framework which runs in the HLT. Chapter 10 addresses the online software which is responsible for the run control and supervision of the entire TDAQ and detector systems during data-taking. It is also responsible for many miscellaneous services such as error reporting, run parameter accessibility, and histogramming and monitoring support. Chapter 11 describes the DCS, responsible for the control and supervision of all the detector services such as gas and high voltage as well as monitoring many critical parameters of the detectors' operation. Chapter 12 draws together the various aspects of experimental 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 and performance. The common tools used for physics selection are described along with the physics 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 event selection assuming a specific machine start-up scenario is presented. Chapter 14 discusses the overall performance of the HLT/DAQ system from various points of view, namely: the HLT performance as analysed in dedicated testbeds, the overall performance of the system in a testbed of ~ 10% ATLAS size, and functional tests of the system in the detector test beam environ-

ment. Data from these various testbeds are also used as input to detailed modelling of a full-scale ATLAS 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 scenarios. Chapter 17 presents the overall organisation of the project and general system resource issues. Chapter 18 presents the HLT/DAQ workplan for the next phase of the project up to LHC turn-on in 2007.

1.1 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.

1.1.1 Requirements from physics

The LHC proton beams will have a crossing frequency of 40 MHz. At the machine's design luminosity of $1 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, some 20 inelastic proton–proton collisions will be produced at each beam crossing. The LVL1 trigger [1-4] will make the first level of event selection in ATLAS, reducing the initial p–p interaction rate down to 75 kHz. Possible future upgrades of the detector electronics will permit this maximum LVL1 output rate to rise to 100 kHz. The High Level Trigger system (HLT) is then required to further reduce the event rate from 75 (100) kHz down to O(100) Hz. Each selected event will have a total size of ~ 1 Mbyte giving a required storage capability of O(100) Mbyte/s.

ATLAS has taken an inclusive approach to its physics selection strategy in order to maximise its physics coverage and to gain flexibility to adapt to new and possibly un-foreseen signatures and data taking scenarios. The configuration of the entire trigger system, including LVL1, must be sufficiently flexible that one can easily change both algorithms and their thresholds in between data-taking runs. The system also needs to be able to respond rapidly to the consequences of changes in the LHC's performance and stability.

1.1.2 Requirements from detector readout

The system is required to handle in parallel data coming from the detectors in some 1,600 readout links, each running at a possible maximum rate of 160 Mbyte/s. The total readout rate to be dealt with after the LVL1 trigger is ~ 150 Gbyte/s. This number 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 currently under study in the individual detector systems. A more detailed description of the parameters of the detectors' readout systems, their readout data bandwidths and channel occupancies can be found in Chapter 2.

1.1.3 Requirements from functional and operational aspects

From its early stages of development, elements of the ATLAS TDAQ system (in particular those concerned with data acquisition) have been used and tested in a test beam environment, providing the necessary data acquisition functionality and performance for the detector test beam data taking. A major effort has been made to minimise the functional divergence between the system used in the test beam and that being developed for the final experiment. Apart from providing a real-life, albeit scaled down, testing facility for the TDAQ system, this policy also has the advantage of familiarising the detector communities in ATLAS with the TDAQ system at an early stage. Some of the elements of the TDAQ 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 the detectors to be able to use and give feedback on the TDAQ system well in advance of this, therefore offers considerable advantages both to the TDAQ and to the detectors in terms of easing the installation and commissioning phase of the experiment.

One important requirement on the TDAQ system which will be particularly necessary in the commissioning and installation phase is that of 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 in order to facilitate and render as parallel as possible the detector debugging and commissioning operations. During running, it will be necessary to have the capability to run a partition of a part of a given detector in test mode to help track down a fault while the rest of the ATLAS detector is taking physics data.

The DCS forms 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 being highly integrated with other parts of the TDAQ system, the DCS has the particular requirements of being operational 24 hours a day, 7 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.

Constraints of floor space and cooling capacity, in particular in the experiment's underground cavern and adjoining service rooms limit the number of racks available to the TDAQ system in these rooms. The consequences of this limitation are discussed later in this document **XXXX ref.?**

1.1.4 Requirements from the long timescale operation of ATLAS

The installation and commissioning phase of ATLAS 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 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 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.

1.2 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 in-depth descriptions are given. The HLT/DAQ can be broken down into four principal systems, namely:

- The Data-flow system - responsible for the readout of detector data, the serving of data to the HLT system, and the transport of the selected data from the readout buffers to mass storage
- The High Level Trigger system - responsible for the post-LVL1 selection involving a rate reduction of a factor of several hundred, and for the filtering and classification of all events accepted by the LVL1 trigger
- The Online system - responsible for all aspects of experiment and TDAQ operation and control during data taking, test and calibration periods
- The Detector Control system - responsible for the coherent and safe operation of the ATLAS detector as well as the interface with external systems and services as well as with the LHC itself.

A schematic diagram containing the principal components of the data-flow and HLT systems is presented in Figure 1-1. The online system is implicitly understood to be connected to all elements in this figure, and the DCS to all hardware elements which are required to be monitored and controlled.

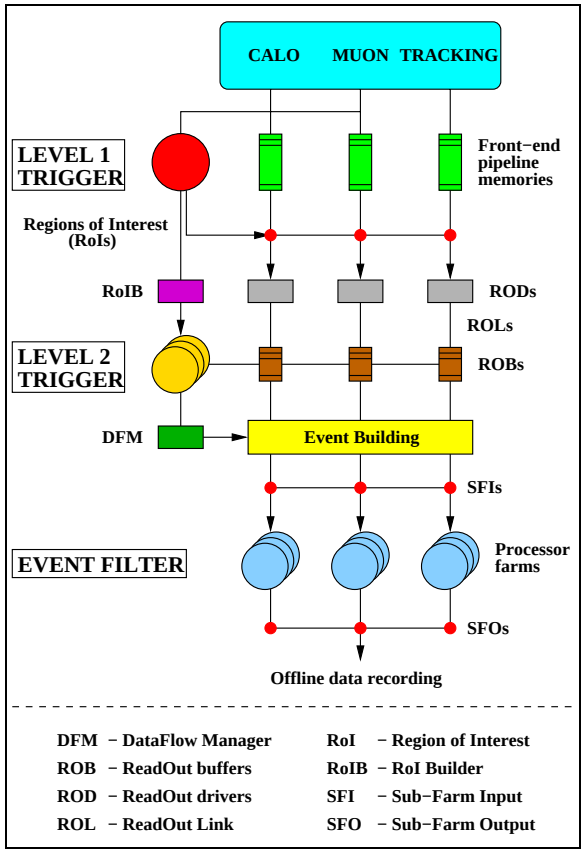

Figure 1-1 Principal components of the data-flow and HLT systems.

1.2.1 The Data-flow system

ATLAS has decided early on to define the boundary of responsibility between the detector readout and the data acquisition to be at the input of the ROL [1-5]. The ROLs transport data fragments of LVL1 accepted events from the detectors' readout drivers (ROD) to the ~ 1600 readout buffers (ROB). From here, requested data fragments from selected ROBs are served to the LVL2 trigger element of the HLT system. All the readout buffers are subsequently informed of the LVL2 trigger decision for the event and mark the data fragments for deletion or for event build-

ing accordingly. These marked data fragments for LVL2 accepted events are then built from the readout buffers across a switched ethernet network, under supervision of the data-flow manager (DFM) into a coherent ATLAS formatted event residing in one of the ~ 100 sub-farm interfaces (SFI). 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 ethernet network and switching technology. The final event building rate which is required by ATLAS is significantly reduced because of the event rate reduction of the LVL2 trigger. The required event building capability is expected therefore to be ~ 3 Gbyte/s with a similar amount being transported to the LVL2 system.

1.2.2 The High Level Trigger system

The HLT system comprises the LVL2 trigger and the event filter. Although they will both be comprised of farms of standard PCs interconnected by ethernet networks they differ in several important respects.

The LVL2 trigger is required to work at the LVL1 accept rate of 75 kHz within an average event treatment time of ~ 10 ms. In order to achieve this degree of rejection in the required time, the LVL2 will use a sequence of highly optimised trigger selection algorithms each operating on only a fraction of the event data. The LVL1 trigger identifies regions in the detector where it has found interesting features, regions-of-interest (RoIs) [1-4]. The region-of-interest builder (RoIB) builds the RoI information from the various parts of the LVL1 trigger. These RoIs are then used to seed the LVL2 algorithms. This enables them to precisely select the region of the detector in which the interesting features reside and therefore from which ROBs to request the data for analysis. Data requests may be done several times per event by different feature extraction algorithms and at each stage in the processing, an event may be rejected. Each processor is used to treat several events in parallel. The final trigger decisions are communicated to the DFM and ROBs for event deletion or building. It should be noted that the entire data for a given event is stored in the ROBs during the LVL2 processing and until the event building process is completed. The LVL2 trigger accept rate is O(1–3) kHz depending heavily on the LHC luminosity and the trigger selection required.

The Event Filter is required to work at the LVL2 accept rate with an average event treatment time of ~ 1 s. More sophisticated reconstruction and trigger algorithms, tools adapted from those of the offline, and more complete and detailed calibration information are used here to effect the selection. The EF receives fully built events from the SFI and so the entirety of the data is available locally for trigger selection. 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 on the fly and those accepted are passed to the SFO for transmission to mass storage.

The scope, complexity, degree of generality, and resolution of the LVL2 and EF algorithms is different. However the overall HLT software selection framework 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 enormous advantage of having a high degree of development commonality across the spectrum of the HLT and the offline, as well as facilitating performance comparisons.

1.2.3 The Online system

The Online Software system is responsible for configuring, controlling, and monitoring the DAQ and HLT systems, but excludes any management, processing, or transportation of physics 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.

An important part of the online software is to provide the services to enable the DAQ and HLT to start up and shutdown. It is also responsible for the synchronisation of the states in 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 the storing of 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.

1.2.4 The Detector Control system

The DCS supervises all hardware of the experimental set-up, not only the different subdetectors of ATLAS, but also 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 conditions before executing commands. Also tools for interlocks both in hardware and in software are provided by DCS. Monitoring and prevention of situations which could cause major damage to the detector or even endanger people's lives are not in the realm of DCS. The former is the responsibility of a dedicated Detector Safety System (DSS), with which DCS interacts, and the latter is addressed by the CERN-wide safety and alarm system.

It is mandatory that all actions initiated by the operator and all errors, warnings and alarms concerning the hardware of the detector are handled by DCS. It has to provide online status information to the level of detail required for global operation. Also the interaction of equipment experts with their subdetector should normally also go via DCS. DCS has to continuously monitor all operational parameters, give guidance to the operator, and signal any abnormal behaviour. It must also have the capability automatically to take appropriate action 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 synchronisation between the DAQ system and DCS. Both systems are complementary in as far that the DAQ deals with the data describing a physics event and DCS treats all data connected with the hardware of the detector. The former are organised by event number and the latter are normally categorised by a time interval. The correlation between them is established in 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 its supervision by DCS is needed continuously. DAQ 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. Different modes of operation are foreseen such as taking data with colliding beams, detector calibration, and stand-alone operation of a subdetector or even of an individual detector element.

1.3 Data types

The 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 values

  The DCS system will produce large amounts of system status and channel value data at a rather high frequency. 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 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 format of these data is defined by the characteristics of each detector's readout electronics, however the overall format of the 'raw event data' is common across the experiment [1-6]. These data are represented physically as a single stream of bytes, which is referred to subsequently as 'bytestream'. These bytestream data are reformatted into more physical objects in preparation for their analysis by the HLT (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

  Configuration data are those data used to configure the TDAQ and detector systems in preparation for data-taking and represents the set of data which is 'active' or selected during a specific data taking run. Examples of this data type include: electronics parameters and thresholds, module mapping, software executables, network parameters, trigger parameters and thresholds, etc. A given data-taking run is defined by a 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. 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 but will also include such data as: 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 online control network.

- 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 and the online control network. This subject is addressed in detail in Chapter 7.

1.4 Long Term Perspectives

A more detailed view of the future planning is described in Chapter 18, but it is useful to set out the principal elements here. In the current installation schedule, 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 LHC is expected to attain a luminosity of $2 \times 10^{33}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$. The installation schedule of the TDAQ 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 TDAQ services during that time should be fulfilled.

The first elements of the TDAQ system will be required by the detectors in early-2005. These 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 trigger farms, their interconnections, and software will be required later in time. The description of those elements in this TDR, documents the current status of development of the system. The desirability of purchasing computing and network hardware at a date which is as late as possible while being consistent with the ATLAS schedule so as to benefit from the latest technological developments is clear. It is vital that maximum flexibility is 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. It will have to be maintained and supported over periods of 15 years or more by bodies of 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 the HLT/DAQ system 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 designs and 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 (a few VME crates and their associated support services). Therefore, it is desirable to test the system 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 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 trigger 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.

1.5 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 inner detector and the liquid argon calorimeter
- 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 end-cap or the muon RPCs
- System - the component systems of the TDAQ are the LVL1 trigger together with those defined in Section 1.2
- Sub-system - the component parts of any of the above TDAQ systems, e.g. the LVL2 trigger, the LVL1 muon trigger, the data collection sub-system
- TDAQ - comprises the LVL1 trigger plus the HLT / DAQ (including the DCS).

Anything else to go here???

1.6 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 First-Level Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

1-5 *Trigger & DAQ interfaces with front-end systems: requirement document version 2.0,* ATLAS Internal Note, ATL-DAQ-98-103 (1998)

1-6 *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 / DCS system. These include the detector readout parameters and the trigger selection for the correct dimensioning of the data-flow system and for understanding the data volumes that will need to be stored. These are the subject of the first two sections.

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 subdivision of the system in detector parts and the amount of configuration data traffic in the case of startup configuration and re-configuration of possible faulty elements.

2.1 Detector readout parameters

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

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

The Calorimeter System consists of several sub-detectors based on different technologies. The barrel electromagnetic, the endcap electromagnetic, the endcap hadron and the forward calorimeters use liquid argon as the active medium [2-3]. The barrel and two extended barrel hadron calorimeters at larger radii (with respect to the other calorimeters) in the range $|\eta| < 1.7$, together forming the Tilecal calorimeter, are instead based on scintillator–iron technology [2-4].

The Muon Spectrometer is divided into a barrel part and two endcaps extending up to $|\eta| \leq 2.4$. 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 chambers: Thin Gap Chambers (TGCs). Furthermore at large pseudo rapidities and close to the interaction point, Cathode Strip Chambers (CSCs) are used, which can handle the higher rate and the severe background conditions [2-5].

The LVL1 Trigger is another source of data for the Data Acquisition (DAQ) system, and dedicated ReadOut Drivers (RODs) are used [2-6].

The organization of the ATLAS detector readout is specified in Table 2-1 in terms of the partitioning, of data sources (the RODs), of ReadOut Links (ROLs), and of ReadOut System (ROS) subsystems, assuming that a maximum of 12 ROLs can be connected to a single ROS subsystem. The information in the table is for a large part based on information provided by the subdetector groups during the third ROD Workshop held in Annecy in November 2002 [2-7]. The parti-

tions used coincide with the TTC partitions described in [2-6]. Each ROD module, ROD crate and ROS subsystems is 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].

A and C are the labels used to identify the two sides of any ATLAS component respect to the pseudorapidity $\eta = 0$. They correspond to the convention of the two sides of the ATLAS cavern. If we define the Z-axis as the one along the beam direction, when looking from inside the LHC ring, the positive Z is the left direction. The positive Z is identified as side A. The negative Z is the right direction and is identified as side C. The beam is going from right to left along Z. The side B is kept for elements on the Z = 0 plane.

The X-axis is horizontal and points from the IP to the LHC ring center. The Y-axis is perpendicular to X and to Z, and it is inclined by 1.236% with respect to the local perpendicular (with respect to the cavern floor). This small angle is needed to follow the beam slope.

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

Table 2-1 The distribution of the RODs per detector per partition.

Detector	Partition	RODs	ROD crates	ROLs	ROS subsystems	ROLS per ROS subsystem
Inner Detector	B Layer	44	3	44	4	3*12+8
	Disks	12	1	12	1	1*12
	Layer 1 + 2	38+26	4	38+26	6	5*12+4
SCT	Barrel A	22	2	22	2	1*12+10
	Barrel C	22	2	22	2	1*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*12+8
	Barrel C	32	3	32	3	2*12+8
	Endcap A	96	8	96	8	8*12
	Endcap C	96	8	96	8	8*12
LAr	EMB A	56	4	224	19	18*12+8
	EMB C	56	4	224	19	18*12+8
	EMEC A	35	3	140	12	11*12+8
	EMEC C	35	3	140	12	11*12+8
	FCAL	4	1	16	2	1*12+4
	HEC	6	1	24	2	2*12

Table 2-1 The distribution of the RODs per detector per partition.

Tilecal	Barrel A	8	1	16	2	1*8+4
	Barrel C	8	1	16	2	1*8+4
	Ext Barrel A	8	1	16	2	1*8+4
	Ext Barrel C	8	1	16	2	1*8+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*8+4
	Endcap C	8+8	1	16	2	1*8+4
RPC	Barrel A	16	1	16	2	1*8+4
	Barrel C	16	1	16	2	1*8+4
TGC	Endcap A	8	1	8	1	1*8+4
	Endcap C	8	1	8	1	1*8+4
LVL1 (RoI, CP, JEP and PP RODs belong to the same partition)	MIROD	1	1	1	5	4*12+8
	RoI	6	1 or 2	6		
	CP	4		16		
	JEP	4		16		
	PP	4	8	16		
	CTP	1	1	1		
Total	33	984	92	1628	146	

Each subdetector will contribute to the ATLAS event with a data fragment. The maximum expected data fragment sizes, as specified by the subdetector groups, and estimates of the data fragment sizes are presented in Table 2-2. To each data fragment is added a ROD header and trailer (48 Bytes total) and a ROB Input (ROBin) header (56 Bytes). In general, a ROS subsystem will concatenate several data fragments and then add a ROS header (52 Bytes) and a wrapper for the Data Collection software (36 Bytes), the latter is removed on receipt of the data. The total event size without and with headers is also specified in the table and falls in the range 1.2–2.2 Mbyte.

Table 2-2 Estimates and maximum data fragment sizes in bytes.

Subdetector	Number of ROLs	Low luminosity (2×10^{33} cm^{-2} s^{-1})	Design luminosity (1×10^{34} cm^{-2} s^{-1})	Max. as specified in ROD workshop of November 2002
Pixels	120	200	500	1300
SCT	92	300	1100	1600
TRT	256	300	1200	1200
E.m. calorimeter (LAr Barrel and EMEC)	728	752	752	1400
Hadron calorimeter	64 (Tilecal)	752	752	1100 (Tilecal)
	24 (HEC)	752	752	1400 (LAr)
Muon precision	192	800	800	1000
Muon trigger (RPCs and TGCs)	48	380	380	1000
CSC	32	200	200	200
FCAL	16	1400	1400	1400
LVL1	56	1200 (average)	1200 (average)	1200 (average)
Total event size, raw		1006864	1346864	2064000
Total event size, with headers		1183352	1523352	2240488

2.2 Trigger and DataFlow parameters

Two baseline scenarios are used in this report:

1. 'low luminosity': for a luminosity of 2×10^{33} cm^{-2} s^{-1} with a LVL1 accept rate of about 20 kHz. The LVL2 accept rate is expected to be about 600 Hz,
2. 'design luminosity': for a luminosity of 1×10^{34} cm^{-2} s^{-1} with a LVL1 accept rate of about 35 kHz. The LVL2 accept rate is expected to be about 1.5 kHz.

The 'LVL1 trigger menus' associated with these scenarios are specified in Chapter 4 and in Appendix A. In view of the uncertainties in the trigger rates the TDAQ system is required to cope with the rates implied by a LVL1 rate of 75 kHz. For low luminosity the LVL2 accept rate is in that case 2.2 kHz, while for high luminosity it is 3.3 kHz.

The data needed for the LVL2 trigger and the type of processing performed by it, depend on the regions-of-interest supplied by the first level trigger. Each of the four different types of RoIs ('muon', 'electron/gamma', 'jet', and '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 subdetectors within the region-of-interest should be requested for further analysis. The data rates can be estimated with the help of information on the type, rate, sizes, locations of the regions-of-interest, and on the mapping of the detector onto the ROB inputs (ROBins). More

details are provided in Appendix A. In Table 2-3, an overview is presented of typical values of rates and data volumes for a 75 kHz LVL1 trigger rate.

Table 2-3 Overview of typical values and rates for 75 kHz LVL1 trigger rate.

	Low luminosity	Design luminosity
Average number of ROLs per RoI request, per LVL1 trigger	17.9	16.2
Average number of groups of 12 ROLs receiving a RoI request, per LVL1 trigger	9.0	8.5
Maximum average output bandwidth per ROBIn (Mbyte/s)	6.7	6.9
Maximum average RoI request rate per ROBIn (kHz)	5.6	4.8
Maximum average output bandwidth per 12 ROBIns (Mbyte/s)	53	59
Maximum average RoI request rate per 12 ROBIns (kHz)	16	14
Total bandwidth LVL2 traffic (Gbyte/s)	1.2	1.1
Event Building rate (kHz)	2.2	3.3
Total bandwidth traffic to Event Builder (Gbyte/s)	2.7	5.0

The size of the LVL2 farms and the number of Sub-Farm Inputs (SFIs) has been estimated assuming the use of dual CPU, respectively single CPU computer server (4 GHz PCs). In Appendix A details are provided on the acceptance factors of the various steps of the LVL2 processing and on processing times assumed. It is expected that at 75 kHz LVL1 rate about 100–150 dual CPU machines will be needed for the LVL2 farm, and about 50–100 SFIs, taking an input bandwidth per SFI of ~ 70 Mbyte/s. The SFIs send the complete events for further analysis to the Event Filter where a factor of ten rate reduction is expected (see Chapter 13). For a typical processing time of one second per event a farm of at least 1000 dual-processor machines will be needed.

2.3 Monitoring requirements

Monitoring will require the transfer of event fragments and/or monitoring results such as histograms from the sources to the destinations. Investigations are under way to identify the sources and the traffic they generate. The following sources of data can be identified:

- ROD and/or ROB
- ROS
- SFI
- Trigger processors (LVL1, LVL2, EF)

Likewise some destinations and their associated networks can be specified:

- private workstations + Control or Data Collection network
- Online monitoring farm (possibly the EF Farm, or a sub-set) + Data Collection network

The following table summarises the present knowledge of the relations between the sources and destinations for monitoring. Figures for the expected traffic generated by monitoring are still missing. The table will be updated when feedback from the concerned groups is provided.

Table 2-4 Monitoring matrix

Source Destination	ROD/ROB	ROS	SFI	LVL1	LVL2	EF
private WS in Control Room	event fragments via Data Collection Network	event fragments via Data Collection Network	events via Data Collection Network			
private WS in Control Room	histograms via Control Network			histograms via Control Network	histograms via Control Network	histograms via Control Network
Online/EF Farm			events via Data Collection Network	events via Data Collection Network	events via Data Collection Network	events via Data Collection Network

2.4 Calibration requirements

The subdetectors calibrations are another important source of data for the experiment. Depending on each subdetector and on 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 Dataflow elements used during the normal data taking, i.e. to the ROS and up to the Sub-Farm Output.

This section presents the current understanding on the calibrations of the electronics, or of the reference systems of the sub-detectors (e.g. the Tilecal LASER system). The in situ calibration of the detector with dedicated physics channels are instead treated in Chapter 4.

The aim of this section is not to describe in details each sub-detector calibration, but rather to indicate when the data will flow through the Data Acquisition infrastructure, how frequently and what will be the amount of data produced. Only a partial view can be given due to the current status of the calibration strategy elaborated by each sub-detector.

Most of the calibrations are dedicated to the caracterization of the sub-detector modules via comparator threshold scans (as for SCT, Pixel and TRT), and of the readout electronics with injection of reference charges to check the amplitude and the timing of the output signals (as for SCT, Pixel, TRT, LAr and Tilecal).

In other situations real physics data are needed as for the timing adjustment of the on-chamber electronics, as for the Muon MDTs.

Most of the calibrations require dedicated runs, but there are calibrations that are performed while taking data in physics mode, during the assigned LHC empty bunches (e.g. the Tilecal LASER system). Some other calibrations do not interact with DAQ and will make use of dedi-

cated data-acquisition systems: the Tilecal Cs source system, the SCT and the MDT alignement systems. The dedicated systems may be more integrated with the DCS infrastructure in future.

The use of the Dataflow infrastructure (high bandwidth ans substantial computing power) for some of the calibrations has not yet been decided. However given the amount of data produced by some calibration tasks and the computing power required to process it, there does not seem to be any other choice. This is for example the case of the LAr 'technical calibrations' in which all the samples are sent out from the RODs, producing about 50 Gbyte of data to be treated. In this case, a clear solution comes from sending the data to a dedicated Event Filter processing task [2-7]. Table 2-5 summarizes the current view on the sub-detector calibrations, elaborated initialy from the various talks held during the DIG Forums [2-9]

2.5 DCS parameters

The DCS controlled systems will be configured using the configuration database (ConfDB) and will output the data to the condition database (CondDB). Therefore we can categorize the DCS data into: configuration data and conditions. The configuration data can be subdivided in: static data, variable data, 'recipes'.

The static data concern the 'system' level configuration that will be done only infrequently, i.e. only at times of hardware modification (new equipment added, possibly equipment changed). The associated data volume is large due to the high number of separate systems to be set up. A high data transfer rate is not required as the operations are not time-critical and will normally be performed during shut-down periods.

During operation of the LHC, each sub-system has various operating modes, each with its own hardware settings. A 'recipe' is defined as a collection of settings associated with a given operating mode. For a given run, more than one recipe may be required at different stages of the run (e.g. beam starting, beam on, beam stopping).

The variable data concern the 'operating mode' configuration that needs to be done more frequently than the system level configuration. Before a run starts, any recipes used will be downloaded from ConfDB and stored locally with each system (to avoid data consistency problems it may be necessary to lock ConfDB before downloading starts). The data volume is lower than that for system configuration, though still quite large as there are many systems, each requiring all settings needed for a number of 'stages'. Downloading is not a time-critical operation, as it will be done outside runs. During a run the downloaded settings are applied when required.

The CondDB will store data collected in the DCS system, that is the conditions. Data will flow from the TDAQ system to the DCS system to allow data from both DCS and TDAQ to be correlated and displayed by the DCS supervisory control system (PVSS). These data are then exported to the CondDB directly from the DCS. It is desirable to store all values into the CondDB (i.e. not only data required for the off-line data-analysis). The data volume will be high (~ 1,000,000 channels plus other information) but the data rate required will be low as data need not be sent in real-time (e.g. only every 15 minutes, one hour, several hours).

Table 2-5 Summary table of some of the sub-detectors calibrations.

Detector	Calibration	May require Dataflow	Dedicated run	During Physics	Comments
SCT	Module charact.	No	Yes	No	Typical
	Module charact.	Yes	Yes	No	sophisticated
	ROD settings	No	No	Yes	monitoring
	Det. Alignment	No	Yes	No	separate DAQ
	Det. Alignment	Yes	No	Yes	real data
TRT	Setup Calib Timing	Yes	Yes	No	
	Setup Calib Noise	No	Yes	No	
	X-check Calib TP time delay	Yes	Yes	No	
	X-check Calib TP ampl	No	Yes	No	
Pixel	Module charact.	No	Yes	No	
	Timing	Yes	No	Yes	real data
LAr	Standard calib	No	Yes	?	calculations in RCC: 50 Mbyte
	Technical calib	Yes	Yes	?	50 Gbyte
Tilecal	Cs source	No	Yes	No	dedicated DAQ
	LASER	Yes	Yes	Yes	dedicated runs + during empty bunches
	Charge injection	Yes	Yes	Yes	dedicated runs + during empty bunches

2.6 References

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

2-2 *ATLAS Pixel Detector Technical Design Report*, CERN / LHCC 98-13 (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 96-42 (1996)

2-5 *ATLAS Muon Spectrometer Technical Design Report*, CERN / LHCC 97-22 (1997)

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

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

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

2-9 *ATLAS Technical Co-ordination, Detector Interface Group, DIG Forums*, http: / / atlas.web.cern.ch / Atlas / GROUPS / DAQTRIG / DIG

3 System Operations

3.1 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.
- Standby: 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 standby 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.

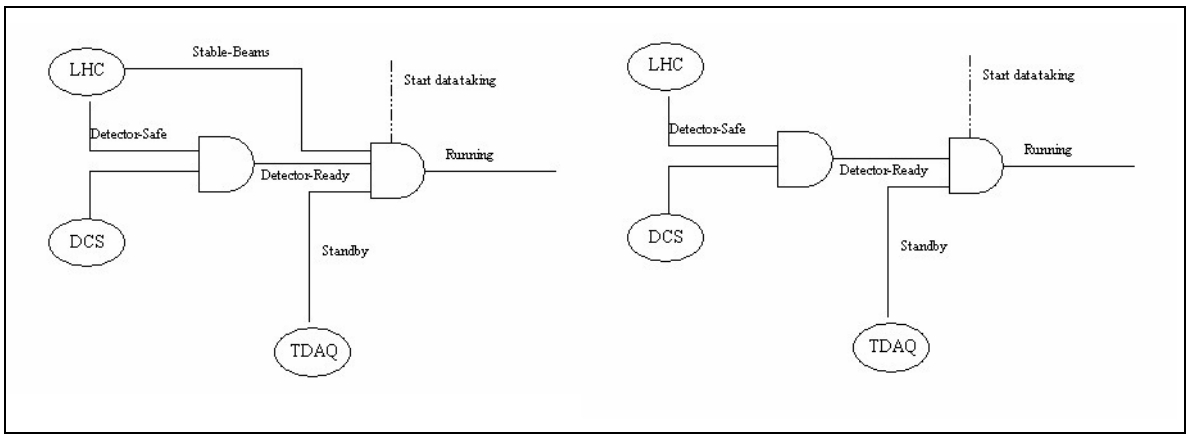

Figure 3-1 Inter-relationship of the detector, machine and TDAQ states for stable and non-stable bean conditions

3.2 The run

3.2.1 Run and Run Number

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.

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.

3.2.2 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.

3.2.3 Requirements

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:

- 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.

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.

3.2.4 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.

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

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 3.3).

Detector calibration without beams: this type of run needs the same actors as above, with the difference that the LHC machine has to be in a 'detector safe' state, i.e. such that the detector may be safely switched on.

(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. The fact that, for instance, the system is set for a 'physics run' does not prevent the system to deal 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 calibration 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.

3.2.5 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: all the conditions described in Section 3.1 are met, 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 enters the 'stopped' 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).

When the pause 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.

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.

3.2.6 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 3.2.5 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.

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). 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.

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. Note 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

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.

processing units, programs sampling events after LVL2 might have to handle events belonging to more than one run).

3.3 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 the smallest 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 the smallest set of TDAQ system elements that can be configured and controlled[1] independently from the rest of the TDAQ system. A TDAQ segment may 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 readout 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. The word ‘smallest’ is the key to the clear understanding of these concepts or definitions. There 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. Indeed a segment cannot be a resource insofar as it is not the ‘smallest’ element that can be disabled (e.g. a sub-farm can be broken up into individual processors, which are the smallest elements that can be individually removed); and a resource cannot be a segment insofar as it cannot be operated independently. With the exception of the ROD crate, which may be seen both as a segment and a partition, a partition cannot be a segment since it is not the ‘smallest’ set of TDAQ elements that can be controlled independently and a segment cannot be a partition since, being the ‘smallest’ set, it cannot take data.

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[2]), 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

1. That is the segment is capable of receiving commands from the TDAQ control system.
2. That is the complete functionality of the DAQ system is available to run a subset of the detector.

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 production data taking at the experiment. A TDAQ Partition always includes some FE elements of one or more sub-detectors in order to provide data. In one exceptional case, that of the DAQ partition, the TDAQ partition may include simulated input data instead of FE elements. A TDAQ Partition may stop at the level of one (or more) ROD crate(s) (ROD Crate DAQ), otherwise it will always include parts of the Event Building and parts of the Event Filter farm (i.e. it will always include a vertical slice of the DAQ system).

TDAQ partitions are properly (i.e. abiding to the rules needed to have a runnable system) 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 readout 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 readout 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 (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.

3.4 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.

3.5 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 chaotic 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).

3.6 Databases

The issue of databases 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 1.3, 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. 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. TDAQ components access this information in a read-only mode.
- 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. 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

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

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).

The amount of potential data producers and consumers, that is to say the total number of TDAQ components who are users of databases, 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.

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

Component	Number (order of magnitude)	Configuration data volume at initialisation per unitary component	Conditions data volume at initialization per unitary 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 access to the configuration and conditions databases, in particular at the time of initialisation and configuration, is a problem: for instance O(1000) Event Filter processors will access simultaneously of O(100) Mbyte of data each. This calls for a structured organisation of the database servers, so as to allow operating the system on reasonable time scales (e.g. seconds/minutes versus hours). To this end, it is proposed to organise the database infrastructure in terms of:

- Local data servers: database servers that hold a copy of the conditions and configurations databases. It is local to a group (order of 30 components) of homogeneous TDAQ components which access databases through the local server. A copy of downloadable software and other miscellaneous data that could be needed by the related TDAQ components are also held on the local server.
- Central server: a fault tolerant, redundant database server which holds the TDAQ master copy of the TDAQ software and the configuration and conditions databases. Local servers will get updates from the central server. The central server gets/provides updates to the offline conditions database. There will be at least one physical central server.

3.7 References

3-1 ATLAS TDAQ / DCS Global Issues Working Group, *Run and States*, ATLAS Internal Note, ATL-COM-DAQ-2003-004 (2003)

3-2 ATLAS TDAQ / DCS Global Issues Working Group, *Partitioning*, ATLAS Internal Note, ATL-COM-DAQ-2003-005 (2003)

3-3 *ATLAS First-Level 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 for the online event selection to reduce the interaction rate of about 1 GHz at the design luminosity of 1×10^{34} cm^{-2} s^{-1} by about seven orders of magnitude to a 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 has to involve LVL1 as well.

The first section describes the requirements defined by the physics programme intended to be studied by ATLAS, followed by a discussion on the approach taken for the selection at LVL1 and the HLT. Next, a brief overview of the major selection signatures and their relation to the various components of ATLAS is given. Then, an overview of the various parts of the trigger menu for running at an initial luminosity of 2×10^{33} cm^{-2} s^{-1} is presented, together with a discussion of the expected physics coverage. This is followed by a description of how changes in the running conditions are going to be addressed, and finally ideas are presented for the strategy of determining trigger efficiencies from data alone.

4.1 Requirements

The ATLAS experiment has been designed to cover the physics in proton-proton collisions with a center-of-mass energy of 14 TeV at the LHC. Amongst the primary goals are the understanding of the origin of the 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 will not have been modelled. The observation of new heavy objects with masses of O(1) 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 coverage of lighter objects with masses of O(100) GeV. In addition, precision measurements of processes within (and outside if found) 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 simple cuts, requiring mostly several 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.

The online event selection strategy has to define the proper criteria to efficiently cover the physics program 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.

The event selection at the 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 value 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, leads to a rate of about 2 kHz 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.

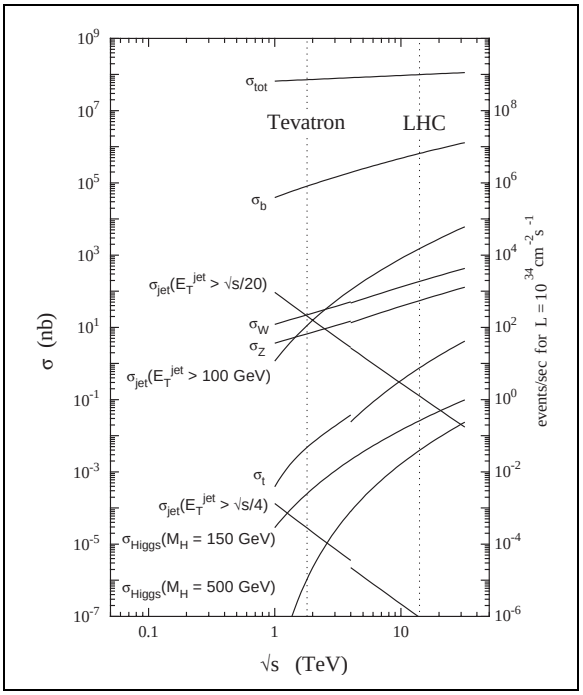

Figure 4-1 Cross-section and rates (for a luminosity of 1×10^{34} cm^{-2} s^{-1}) for various processes in proton-(anti)proton collisions, as a function of the center-of-mass energy

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 to operate in the challenging environment of the LHC, with up to about 20 inelastic events per bunch crossing at design luminosity. Furthermore, it has to also 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. While presently the selection is based on rather simple criteria, however making use of the ATLAS capabilities to reject most of the fake signatures for a given selection, it is mandatory to have additional tools such as exclusive criteria or more elaborate object definitions at disposal for the online selection.

4.2 The approach

In order to guarantee optimal acceptance of new physics within the current paradigm of particle physics, an approach based on emphasising the use of inclusive criteria for the online selection has been taken, 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 of O(10) GeV. The choice of the thresholds has to be made such that there is a good overlap with the reach of the Tevatron and previous as well as other existing colliders, and the sensitivity to new light objects, e.g. Higgs bosons, is guaranteed. To enlarge this high p_T selection to complement the ATLAS physics potential requires access to signatures involving more exclusive selections, using e.g. charged particles with transverse momenta of O(1) GeV for the study of b-hadron physics.

The selection at the HLT will be in most cases seeded by the information already obtained at LVL1 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 fulfil the latency con-

straints. These will mostly require only a few per cent of the event data, relying on the guidance of the regions-of-interest (RoIs) from LVL1. The selection signatures will be refined at the EF, where the full event is available for analysis, using more elaborate calibration and alignment parameters and having less constraints on the latency. Furthermore, the EF will provide classification of the accepted events, in order to facilitate offline analyses.

Although the primary part of the selection will be predominantly based on simple and inclusive signatures to allow for future optimization, more refined selection tools have to be available. These include the usage of more elaborate algorithms, e.g. b-tagging of jets, and the application of more exclusive criteria, e.g. in order to enrich the available samples for certain physics processes. 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 Selection objects

The selection objects defined for the HLT can be based on information from all sub-detectors of ATLAS, at full granularity. As mentioned above, the difference in definition of these objects between LVL2 and the EF will mostly refer to the complexity of the algorithm interpreting the raw data and the detail and level of accuracy for the alignment and calibration information used. In addition, the EF has the full event at its disposal for the search for these objects.

ATLAS, as a multi-purpose detector, will have charged particle tracking in the Inner Detector covering the pseudo-rapidity region of $|\eta| < 2.5$ inside a solenoidal field of 2 T and fine-grained calorimeter coverage for $|\eta| < 2.4$, esp. in the electromagnetic compartments. The calorimeter coverage extends 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 selection objects foreseen to be used at 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, a search for high p_T tracks, and a match between the cluster and tracks. Further refinement is possible via the identification and recovery of Bremsstrahlung events and the application of isolation criteria.
- Photon (within $|\eta| < 2.5$): the selection of photons will be also 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 conversions have been identified).
- Muon (within $|\eta| < 2.4$): the muon selection will make use of the stand-alone muon system to determine the muon momentum and in some cases the charge. A refinement of this information can be obtained by searching for tracks in the Inner Detector and matching these candidates with the stand-alone muon track segment. Also for muons, isolation criteria can be used, e.g. using information from the calorimeters.
- Tau (within $|\eta| < 2.5$): the selection of taus in the hadronic decay mode will use the calorimeter shower shapes to identify narrow hadronic jets. These can be matched to one or more tracks found in the Inner Detector.

- Jet (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 as well. Furthermore jets can be searched for in the forward calorimeter between $3.2 < |\eta| < 4.9$.
- b-tagged jet (within $|\eta| < 2.5$): the selection will be based on jets already selected, where the associated tracks found in the Inner Detector are used to search for e.g. large values of the impact parameter or for the presence of secondary vertices, as well as 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. This requires the reconstruction and identification of low p_T charged hadrons and leptons (electrons and muons), with guidance from LVL1 not always available. At LVL1 the trigger will demand the presence of at least one low p_T muon.

4.4 Trigger menus

In this section, the present understanding of the trigger menu for running at an initial peak luminosity of 2×10^{33} cm^{-2} s^{-1} is presented. Several parts of this trigger menu are distinguished and 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 program,
- prescaled physics triggers, which will extend the physics coverage for ATLAS, e.g. by having inclusive selections with lower threshold to enlarge the kinematic reach, and provide samples for understanding background processes and detector performance,
- (semi-)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 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 parts of the current trigger menu is followed by a discussion of the physics coverage achieved, including indications of the dependence of the acceptance for several physics processes on the threshold values.

The derivation of the trigger menus and the threshold values starts from the analysis of the physics requirements, followed by an assessment of the rejection capabilities at the various selection stages, and finally taking into account the estimates for the total HLT output bandwidth.

This procedure is iterative in order to include existing information, e.g. from studies of the LVL1 trigger (see [4-3]), or from past studies of the HLT performance, e.g. as documented in [4-2].

Not discussed in this document are possible trigger selections dedicated to the study of forward physics or heavy ion interactions, as these additional aspects of the ATLAS physics potential are only presently under investigation. The flexibility in the online selection will allow to address these physics processes. 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, the selection strategy in this document should be extremely useful for these environments as well. Furthermore 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 from future discoveries prior to the start of the LHC might influence the contents of the trigger menu.

In the subsequent sections, the use of labels with the form 'NoXXi' to identify specific trigger items will be used. The meaning of the labels is the following: '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); 'N' the number of objects; and 'i' indicates an isolation requirement. As an example, 2μ20i refers to the requirement of two muons, with a p_T threshold of 20 GeV each, fulfilling isolation criteria. The thresholds indicate the true value above which the selection has good efficiency. 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 bandwidth of the accepted events to mass storage.

4.4.1 Physics triggers

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

A large part of the physics program will rely heavily on the inclusive single and di-lepton triggers, involving electrons and muons. Besides selecting Standard Model events, such as the production of W and Z bosons, gauge boson pair production, $t\bar{t}$ production (except the fully hadronic decay) and several decay modes of the Standard Model (and MSSM) Higgs boson(s), they also provide sensitivity to new heavy gauge bosons (W', Z'), to supersymmetric particles, large extra dimensions (via the Drell-Yan di-lepton spectrum), to particle decays involving τ's (via their leptonic decay), etc.

The inclusive and di-photon triggers will select a light Higgs boson via its decay $H \rightarrow \gamma\gamma$ as well as 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, where the latter are especially relevant in case of R-parity violation. The inclusive and the di-jet trigger will be used for instance 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.

Selection signature	Examples of physics coverage
e25i	$W \rightarrow l\nu$, $Z \rightarrow ll$, top production, $H \rightarrow WW^{(*)}/ZZ^{(*)}$, W',Z'
2e15i	$Z \rightarrow ll$, $H \rightarrow WW^{(*)}/ZZ^{(*)}$
μ20i	$W \rightarrow l\nu$, $Z \rightarrow ll$, top production, $H \rightarrow WW^{(*)}/ZZ^{(*)}$, W',Z'
2μ10	$Z \rightarrow ll$, $H \rightarrow WW^{(*)}/ZZ^{(*)}$
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
τ60	charged Higgs
μ10 + e15i	$H \rightarrow WW^{(*)}/ZZ^{(*)}$, SUSY
τ35 + 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-decays ($B \rightarrow \mu\mu X$) and $B \rightarrow J/\psi\,(\psi')X$

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.

Rare b-hadron decays and b-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

In Table 4-2 a prototype for additional contributions to the trigger menu in the form of prescaled physics triggers is given. These triggers are needed to 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.

A typical example of 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 minimum bias selection will be crucial in determining trigger efficiencies from 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 required statistical accuracy for the given application, taking into account the total available bandwidth. Furthermore, the ranges for the thresholds should be

Selection signature	Physics motivation
single jets (8 thresholds between 20 and 400 GeV)	inclusive jet cross-section
di-jets (7 thresholds between 20 and 350 GeV)	di-jet cross-section
three jets (6 thresholds between 20 and 165 GeV)	multi-jet cross-section
four jets (5 thresholds between 20 and 110 GeV)	multi-jet cross-section
single electrons (4 thresholds between 5–25 GeV)	inclusive electron cross-section
di-electrons (2 thresholds between 5–15 GeV)	
single muons (5 thresholds between 5–20 GeV)	inclusive muon cross-section
di-muons (2 thresholds between 5-10 GeV)	
single photons (6 thresholds between 10-60 GeV)	inclusive photon cross-section
di-photons (2 thresholds between 10-20 GeV)	
taus (3 thresholds between 25 and 60 GeV)	
di-tau (thresholds TBD)	$Z \rightarrow \tau\tau$ 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

Table 4-2 Examples for additional prescaled physics triggers

seen as indicative only: the aim will be to cover the widest possible range, i.e. extending the coverage from the values of the nominal un-prescaled selection down to the lowest possible one for a given signature. The values of the prescale factors will evolve with time, these triggers will be of very high importance in the early phases of the data taking after the start-up of the LHC.

4.4.3 (Semi-)exclusive physics triggers

In Table 4-3 a list of further selections using exclusive criteria is presented. An example is given

Selection signature	Physics motivation
e20i + xE	$W \rightarrow e\nu$
$\mu + \gamma$ (thresholds TBD)	lepton flavour violation
μ8 + B-decays	b-hadron physics
b-jet (multiplicity / thresholdsTBD)	

Table 4-3 Examples for additional (semi-)exclusive physics triggers

by the extension of the selection for b-hadron physics to involve more decay modes. As discussed in more detail in [4-1] and [4-2], 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 on semi-exclusively reconstructing selected decays, which involves the possibly unguided search for rather low p_T charged particles.

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

4.4.4 Monitor and calibration triggers

The selection signatures presented in Tables 4-1, 4-2, and 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 decay of the copiously produced Z bosons will be of great help, e.g. to determine the absolute energy scale for electrons, and to intercalibrate the electro-magnetic parts of the calorimeter, using $Z \rightarrow ee$ events. The muonic decay will set the absolute momentum scale both in the Inner Detector and in the muon system. In addition, the inclusive electron and muon triggers will select $t\bar{t}$ production, via the semi-leptonic decay of one of the top quarks. This sample sets the jet energy scale, via the hadronic decay to two jets of a W boson from the t decay, and determine the b-tagging efficiency.

Selection signature	Example for application
random trigger	zero bias trigger
unpaired / empty bunch crossing random trigger	background monitoring
e25 loose cuts	trigger monitoring
e25 no isolation	trigger monitoring
μ20 loose cuts	trigger monitoring
μ20 no isolation	trigger monitoring
γ60 loose cuts	trigger monitoring
γ60 no isolation	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

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

Samples of inclusive muons are the starting point for the Inner Detector alignment, and will be used to study their energy loss in the calorimeters and to understand and align the muon detectors. Furthermore, samples of inclusive electrons will be used to understand the electromagnetic energy scale of the calorimeter, to perform alignment of 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 the use of inclusive (di-)jet samples.

However, there will be further requirements from the sub-detectors of ATLAS to collect samples of events for calibration, alignment and monitoring purposes. Some examples for such applications are:

- LAr recording all 32 time samples
- online determination of the vertex position in x, y and z
- FILL IN HERE AND IN THE TABLE
- ADD something on luminosity monitoring (Z, rates, ...)

These 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 store only the results of processing the event in the EF and not to store the raw data. This would only be possible after stable operation of the detector and the machine had been reached.

4.4.5 Physics coverage

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

As an example, the search for a Standard Model Higgs boson H 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 involves the requirement of a lepton from the semi-leptonic decay of one of the top quarks. In [4-1], the study was based on the assumption of a p_T threshold for both the electron and the muon of 20 GeV. The example in Table 4-5 shows the impact of raising one or both of these thresholds on the expected significance for signal observation.

$p_T(e) >$	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**

Table 4-5 Example of loss in significance for the associated production of $t\bar{t}H$ for an integrated luminosity of 30 fb^{-1}

A specific 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 [4-1], for a mass of $M(\omega_T)$ = 500 GeV, the offline analysis would demand 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 of Tevatron for an integrated luminosity of 15 fb^{-1} should cover E6 di-quarks with masses up to 700 GeV, heavy W' / Z' bosons up to 850 GeV, techni-rho mesons ρ_T up to 900 GeV, excited

quarks q* up to masses of 950 GeV and axigluons up to 1250 GeV. The transverse energy spectrum of the jets from the decay of such a resonance would lead to a Jacobian peak of 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 are targeted at enlarging 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 tag jets. Requiring that two such jets are present in an event and are separated significantly in rapidity could allow to lower cuts on e.g. lepton thresholds and thus increase the significance for Higgs observation. A second related example is the search for an invisibly decaying Higgs, where at the trigger the requirement of two tag jets and missing transverse energy would be used.

The overall optimization of the online selection will be refined until the first collisions are recorded. The confrontation with real data might necessitate the revisiting of this optimization, which will have to be continuously refined throughout the lifetime of the experiment. It has to take into account at least the following aspects:

- the present understanding of physics,
- the effect of the foreseen thresholds on the acceptance for known (and unknown) physics processes,
- the rate for the selection (as the rejection cannot be infinite due to e.g. the copious production of W bosons at LHC, which contribute via the leptonic decay to true electron and muon triggers),
- the inclusiveness of the selection (where one should note that a more exclusive selection does not necessarily imply a significantly reduced output rate, because many final states will have to be considered in order to achieve good coverage),
- and the available resources for the online selection.

Various handles 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 selection, and where possible, tightening of selection cuts in the trigger object definition. Some of these handles can also be applied at LVL1, in order to reduce the input rate to the HLT, in case the resources for the treatment of the design LVL1 accept rate were to be insufficient.

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. Furthermore the introduction of additional biases at the trigger level due to less inclusive selection should be avoided, as it might influence the accuracy of various precision measurements.

4.5 Adaptation 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 will include a decrease in the luminosity during a machine fill, 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 effec-

tively and thus maintain a robust selection process. It is important to keep a proper history of all changes to the trigger configuration. These changes might be flagged by choosing a new run number.

4.5.1 Luminosity changes

During the life time of a machine fill of about 14 hours, the luminosity will drop by a factor of about four. To take advantage of the available bandwidth in the HLT system, the trigger should be adjusted such that a constant output rate of the HLT to mass storage is maintained. This could be achieved e.g. by including more trigger selections in the trigger menu, by adjusting prescale factors to fill up the available bandwidth with more selections, using lower thresholds, with more exclusive selections, or even by changing (lowering) the thresholds of the inclusive physics triggers. In this way, the physics coverage of ATLAS will be extended during a machine fill. One example is the case of B-hadron physics, where the inclusive trigger menu as stated above is restricted to select only final states involving B-decays leading to at least two oppositely charged muons. As the luminosity drops below the peak value, other decay modes (e.g. fully hadronic ones or decays involving two low p_T electrons) might be added.

In addition one has to take into account that the sizeable change in the luminosity during a machine fill 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.

4.5.2 Background conditions

One will have to foresee adjustments to 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 acceptable. Furthermore, machine related background 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.

4.5.3 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-initialisation procedures. Changes in the prescale factors could be done dynamically in an automated fashion (e.g. updating these number every N minutes), however it might be preferable to perform these changes less frequently in order to simplify the calculation of integrated luminosities for cross-section measurements.

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 alone. In this section, only a few basic ideas will be described, more quantitative details need to be worked out. In addition, no explicit distinction between LVL1 and the HLT selections will be made. The efficiency determination will have to be done separately for each trigger (sub-)level. Furthermore, it is mandatory that the efficiency of the selection be monitored carefully throughout the lifetime of ATLAS, which implies that the low level triggers, from which the efficiencies are calculated, have to be kept running with pre-scale factors set to provide appropriate 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.

4.6.1 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 e.g. electron triggers with higher E_T thresholds are studied using samples of events selected by an electron trigger with a lower (and thus already understood) threshold. The same holds for all other object types.

4.6.2 Orthogonal selections

For the HLT, it will also be possible to foresee orthogonal (i.e. completely independent) selections in order to study the trigger efficiency of a particular step in the selection sequence, e.g. by using a sample which requires the presence of high p_T 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. It could be however envisaged to use e.g. events selected by a muon trigger to study the calorimeter trigger response at LVL1, using events where a jet is balancing a Z-boson decaying to two muons.

4.6.3 Di-object selections

A further possibility is to use di-object selections (e.g. the plentiful produced $Z \rightarrow ll$ decays) which have been selected online by a single object requirement and to study the trigger response for the second object, which is not required for the online selection. Here also other samples of resonances such as the J/ψ or the Υ might prove very useful for the region of lower transverse momenta.

4.6.4 Required statistics

It cannot be emphasized enough that the amount of data needed to perform a proper understanding of the online selection is not going to be negligible in the start-up and throughout the lifetime of ATLAS, and will play an important role in assuring the potential of the experiment for discoveries and precision physics measurements. A more detailed assessment of the expected needs of ATLAS should be done in the next couple of years. For the moment only 10% of the

total rate of the HLT is attributed to these triggers, which might not be sufficient, and which is clearly smaller than the fraction currently used by running experiments. During run I of CDF, as much as 30% of the accepted triggers at the second stage 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. As D0 kept the thresholds fixed during a machine fill, the fraction of pre-scaled triggers increased towards the end of fill, where it could make up 90% of the total rate.

4.7 Outlook

Details of the implementation of this strategy in terms of the software framework to perform the selection can be found in Chapter 9.4. In Chapter 13, more information on selection algorithm implementations and their performance in terms of signal efficiency and background rejection can be found. Finally, Chapter 14 addresses the issue of system performance of the online selection, presenting the current understanding of the resources (e.g. CPU time, network bandwidth) needed to implement the selection strategy presented in this chapter.

4.8 References

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

4-2 *ATLAS HLT, DAQ and DCS Technical Proposal*, CERN / LHCC / 2000-017 (2000)

4-3 *ATLAS First-Level Trigger Technical Design Report*, CERN / LHCC / 98-14 (1998)

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

5 Architecture

This chapter defines the architecture of the ATLAS TDAQ system. First, it describes how the ATLAS TDAQ system is positioned with respect to both the rest of the experiment and external systems such as the LHC machine or CERN technical services. This is followed by the description of the organisation of the TDAQ system: in terms of the functions it provides and how these latter are organised in terms of sub-systems.

The system architecture is defined next. It is represented in terms of abstract (implementation independent) components, each associated to a specific function, and their relationships. The way different TDAQ sub-systems map onto the architecture follows.

The concrete implementation (baseline) of the ATLAS TDAQ architecture is then presented, followed by a discussion of how it will scale with respect to the performance profile required by the experiment.

5.1 TDAQ context

5.1.1 Context Diagram

The ATLAS TDAQ context diagram is shown in Figure 5-1. The LVL1 trigger provides LVL2 with Regions-of-Interest (RoI) and other data needed to guide the LVL2-trigger data selection and processing; this interface is discussed in detail in Part 2. The Timing, Trigger and Control (TTC) system provides signals associated with events that are selected by the LVL1 trigger. ReadOutDrivers(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 which provide LVL1 trigger decision data to be read out for the selected bunch crossings. The LVL1 trigger system, the TTC system and the ROD systems of the detectors all need to be configured by the DAQ system, for example at the start of each run. These components are shown in the top part of the diagram.

Interfaces to other external systems are also illustrated in Figure 5-1. These connect to the LHC machine (e.g. to exchange information on beam parameters), to the detectors (e.g. to control voltages), to the experimental infrastructure (e.g. to monitor temperatures of racks), and to the CERN technical infrastructure.

The remaining interfaces relate to long-term storage of data that must also be accessed for offline analysis of the event data. For events that are retained by the high level triggers, the event data have to be stored for offline analysis. In addition, a large amount of non-event data has to be stored: alignment and calibration constants, configuration parameters, etc. Not shown in the figure is the importation of programs from the offline software for use by the HLT.

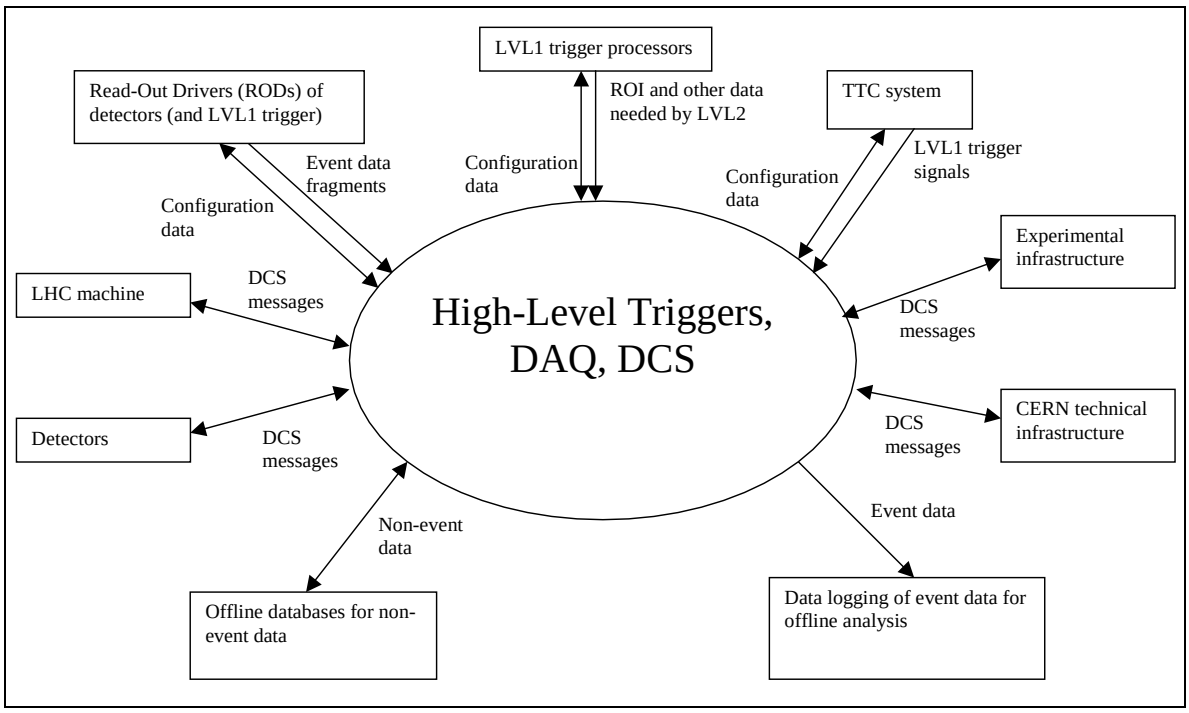

Figure 5-1 Context diagram.

5.1.2 TDAQ Interfaces

The ATLAS TDAQ system interfaces to a variety of other subsystems inside ATLAS as well as external systems which are not under the experiments control. The following describes these interfaces in terms of:

- Partners involved (TDAQ subsystem and non-TDAQ system).
- Responsibilities for the interface on both sides.
- Data exchanged via the interface.
- Pointers to documentation on interfaces including data formats.

The interfaces can be split into two classes, those to other parts of ATLAS and those to external systems. Table 5-1 summarises the requirements for external interfaces.

5.1.2.1 TDAQ interfaces to ATLAS

5.1.2.1.1 LVL1 Trigger

Although technically part of the ATLAS TDAQ, the LVL1 trigger is here seen as an external subsystem from the point of view of the DAQ and the HLT components.

The direct interface to the LVL1 trigger is provided through the RoI Builder. It connects to both the CTP and the Calorimeter and Muon LVL1 triggers, and receives direct input from them. This information is combined on a per event basis inside the RoI Builder and passed to the LVL2 system. Access to the LVL1 readout buffers is possible at the level of the Event Filter.

The physical interface is provided by an S-LINK. There are inputs from nine different sources into the RoI Builder. The interface has to run at full LVL1 speed, i.e. 75 kHz. Data flows from the LVL1 system to the RoI Builder with the S-LINK interface providing only flow control information in the reverse direction. Asserting XOFF is the only way for the RoI Builder to stop the trigger. The specification of the interface is described in [5-3].

Only one partition at a time can include the LVL1 and LVL2 trigger. This will be the default trigger during a physics run.

5.1.2.1.2 Detector specific triggers

For test beams, installation and commissioning as well as calibration runs it will be necessary to trigger the data acquisition for a subset of the full system (i.e. a partition) independent from LVL1 and LVL2 and in parallel with other ongoing activities. To this end detectors provide a Local Trigger processor (LTP): a functionally equivalent, yet detector specific, version of the LVL1 CTP. These triggers are referred to as detector specific triggers. For event building a Data Flow Manager (DFM) per partition is assumed.

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. The DFM must be able to throttle the detector specific trigger via the ROD-busy tree.

5.1.2.1.3 Detector Front-ends

The detector front-ends provide the raw data for each event that LVL1 triggers on. The detector side of the interface is the ROD while the TDAQ side is the ReadOut Buffer (ROB). The connection between the two is the ReadOut Link (ROL).

From the point of view of the detector side the interface follows the S-LINK specification. Implementation details can change as long as the specification is followed and all RODs provide room for a mezzanine card to hold the actual interface.

Data flows from the ROD to the ROBs, while only the S-LINK flow control is available in the reverse direction. This interface has to work at the full LVL1 accept rate, i.e. 75 kHz.

5.1.2.1.4 DCS

There is a two-way exchange of information foreseen between DCS and the rest of the TDAQ system. DCS will report information to TDAQ about the status and readiness of various components and TDAQ will both provide configuration information and issue commands related to runs. DCS is also the only interface to all information regarding the LHC machine.

All this communication happens via mechanisms defined and provided by the Online Software, [5-4].

5.1.2.1.5 Detector Monitoring (ROD Crate to Online SW)

TBD

5.1.2.1.6 Conditions Database

The conditions database will store all time-dependent status information of the system that is important for reconstructing events. Components of the HLT/DAQ system will mostly write information into the database, but some like the Event Filter will also read from it.

The concrete interface to the conditions database is not yet defined. Assuming that the implementation is making use of a relational database a variety of communication mechanisms will be available. It remains to be studied how accessing the conditions database will affect the HLT performance itself and how frequently this will occur.

5.1.2.2 External interfaces

5.1.2.2.1 Mass Storage

Events that have passed the Event Filter will eventually be written to mass storage. This service will be centrally provided at CERN. However, in the current design, the Sub Farm Output (SFO) component produces a series of raw data files which are stored on disk to provide local buffering. The local buffer, for example, will be large enough to accommodate for failures in the network connecting to the CERN computer centre. A separate process will take these files up and transfer them to the computing division.

The data files are stored in a well-defined format and libraries are provided to read these files in offline applications.

Table 5-1 Overview of interfaces between TDAQ and other ATLAS or external systems

Interface	Data Rate	Data Volume	Data Type
LVL1 Trigger	75 kHz		Trigger and RoI data
Detector specific trigger	xx kHz		Trigger
Detector Front-ends	75 kHz	135 Gbyte/s	Raw data
Detector Monitoring	few Hz	few Mbyte/s	Raw data
DCS	few Hz	?	Control information
Conditions Database	?	?	System status
Mass Storage Interface	200 Hz	360 Mbyte/s	Raw data + LVL2 and EF results

5.1.2.2.2 LHC machine

All communication between the LHC machine and TDAQ is done via DCS. Therefore there is no direct interface between any other TDAQ component and the LHC. The communication mechanisms are described in Chapter 11.

5.2 TDAQ organisation

The purpose of this section is to show how TDAQ is organised internally. The internal organisation is looked at from three perspectives: what functions are performed by TDAQ, how functions are associated to TDAQ building blocks, and an abstract categorisation of internal elements. Generality (as opposed to implementation) and complementarity of views is stressed.

5.2.1 Functional decomposition

The TDAQ system provides the ATLAS experiment with the capability of: **moving** the detector data (physics events) from the detector to mass storage; **selecting**, between detector and mass storage, those events which are considered of physical interest; and **controlling** and **monitoring** the whole experiment.

The following functions are identified:

- ***Detector readout***: the data produced by one bunch crossing are stored in detector memories (RODs), an event is therefore split in a number of fragments: there are ~ 1600 of such memories which have to be read out at a rate of 75 kHz into a set of TDAQ buffers (the TDAQ event memory).
- ***Movement of event data***: once buffered event fragments have to be moved to the high level triggers and, for selected events, to mass storage. This is a complex process which involves both moving small amounts of data at the LVL1 trigger rate (the region-of-interest data for the LVL2 trigger at 75 kHz) and the full event (i.e. ~ 1 Mbyte) at the rate of the LVL2 trigger (few kHz).
- ***Event selection***: TDAQ is responsible for reducing the rate and the data volume to the manageable amount of ~ 100 Mbyte / s; this is achieved by a sophisticated, 2-level, trigger system.
- ***Event storage***: events selected by the HLT system are written onto permanent storage for further offline analysis.
- ***Controls and monitoring***: this refers to the capability of i) operating and controlling the experiment (detector, infrastructure, TDAQ) and ii) monitoring the state and behaviour of the whole of ATLAS.

5.2.2 TDAQ building blocks and sub-systems

The ATLAS TDAQ system is designed to provide the above functions in terms of the following building blocks:

- ***ReadOut System (ROS)***: event data is buffered, by the ATLAS detectors, in the RODs; each ROD holding a fragment of the whole ATLAS event. The ROD fragments are read by TDAQ into its own buffers, the ReadOut Buffers (ROBs). Logically, but not necessarily implementation-wise, the number of ROB buffers is the same as the number of ROD fragments (indeed, see below, the LVL2 trigger needs to access data at the level of the individual ROD fragments). Event fragments are kept in the ROB buffers until they are either moved downstream (accepted by LVL2) or they are removed from the system (rejected by LVL2). The depth of the ROB buffers is determined by the time needed by LVL2 to select events, plus the additional overhead to clear (in case of a LVL2 reject) or transfer the frag-

ment to the Event Builder and clear it. The ROS provides individual event fragments, out of the ROBs, to the LVL2 trigger and to the event builder: in this latter case a further level of buffering, multiplexing several individual ROBs into a single event builder input, may be provided by the ROS.

- ***Level-2 trigger***: as described in Chapter 1, the LVL2 trigger uses the RoI mechanism to selectively read out only a fraction of an event. Using information from the LVL1 trigger, the RoI defines what fragments the LVL2 trigger will need for a particular event. Appropriate fragments are requested from the ROBs and used to decide on the acceptance or rejection of that event. The LVL2 trigger requests fragments on the basis of i) the LVL1 identifier and ii) the ROL number (as opposed to the ROB number).
- ***Event Builder***: the event is kept in the form of many (~ 1600) parallel streams up to the decision by the LVL2 trigger. Any further reduction in the event rate needs working on the complete event, hence the requirement for a component which merges all the fragments of an event into a single place. This is the event builder, which also includes a number of buffers where full events are collected.
- ***Event Filter*** (EF): another level of event rate reduction is provided by the event filter which requests complete events from the Event Builder buffers and performs on them complex selection algorithms.
- ***TDAQ controls***: the function in charge of the control and supervision of the whole TDAQ system; this includes the initialisation and configuration of the TDAQ components, the supervision of (potentially multiple) data taking sessions (runs).
- ***Detector controls***: it represents the function in charge of controlling and monitoring all aspects of the ATLAS detector; it also includes the function of initialisation and configuration of the ATLAS detector.
- ***Monitoring***: it is the part of TDAQ in charge of both the (event data based) monitoring of the experiment and the operational monitoring of TDAQ.
- ***Online services***: information management, databases, and services to operate the experiment.

For the purpose of the organisation of the development, the work has been organised in terms of broad, function oriented, sub-systems:

- ***The dataflow sub-system***: it is responsible for the development of the detector readout and data transport functions.
- ***The high-level trigger (HLT) sub-system:*** it is responsible for the development of the LVL2 and event filter components. It is in turn organised in terms of
  - HLT infrastructure: responsible for the development of the infrastructure necessary to support the trigger and filter algorithms, and
  - physics and event selection architecture (PESA): to achieve a coherent description of the physics needs which will drive the strategy for the selection of events in the ATLAS Trigger/DAQ system, with the emphasis on the HLT.
- ***The online software sub-system***: it is responsible for the development of all the supporting software, such as the one related to controls, databases, monitoring, etc.
- ***Detector Control System (DCS)***: it comprises the control of the subdetectors and of the common infrastructure of the experiment and the communication with the CERN services (cooling, ventilation, electricity distribution, safety, etc.) and the LHC accelerator.

5.2.3 Categories of components

The functions required by the ATLAS TDAQ system call for the use of a small number of categories of components. This is presented in the following.

- ***Buffers***: they are used to decouple the different parts of the system: detector readout, LVL2, event builder and event filter. Because of the parallelism designed into the system, buffers fulfilling the same function (e.g. ROBs) operate concurrently and independently.
- ***Processors***: to run event selection algorithms, to monitor and control the system. They are organised in farms, groups of processors performing the same function.
- ***Supervisors***: these elements coordinate concurrent activities, in terms of assigning events to processors and buffers, at the different levels: the LVL2 trigger (RoIB and L2SV), the event builder (DFM), and event filter.
- ***Communication systems***: they 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. In particular some of them provide a multiplexing function: 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).

5.3 TDAQ generic architecture

5.3.1 Architectural components

This section presents the global architecture of the ATLAS TDAQ system. This is the result of several years of studies, design, development of prototypes and their exploitation in test beams. In particular the ATLAS TDAQ architecture builds upon the work presented in previous LHCC documents: the ATLAS technical proposal [5-1] and the DAQ/DCS/HLT technical proposal [5-2]. Additional design decisions have taken place since [5-2]: for example a request/response protocol has been adopted for the movement of event data between the ROS and, respectively, the LVL2 and Event Builder.

Table 5-2 Performance requirements

Function	Input requirements	Output requirements	Comments
Detector readout	O(1600) * O(1 kbyte) * 75 kHz	Few % of input fragments to LVL2 at 75 kHz O(1600) fragments at a few % of 75 kHz	
Level-2	Few % of event fragments at 75 kHz	75 kHz decision rate	
Event Builder	O(1600) fragments at a few % of 75 kHz		
Event Filter			

The architecture is presented following the functional breakdown described in the previous sections. First the components visible at the architectural level are defined from the functional point of view, without any references to a possible implementation. Then some detail is provided for sub-systems, in order to describe how a sub-system maps onto the general architecture.

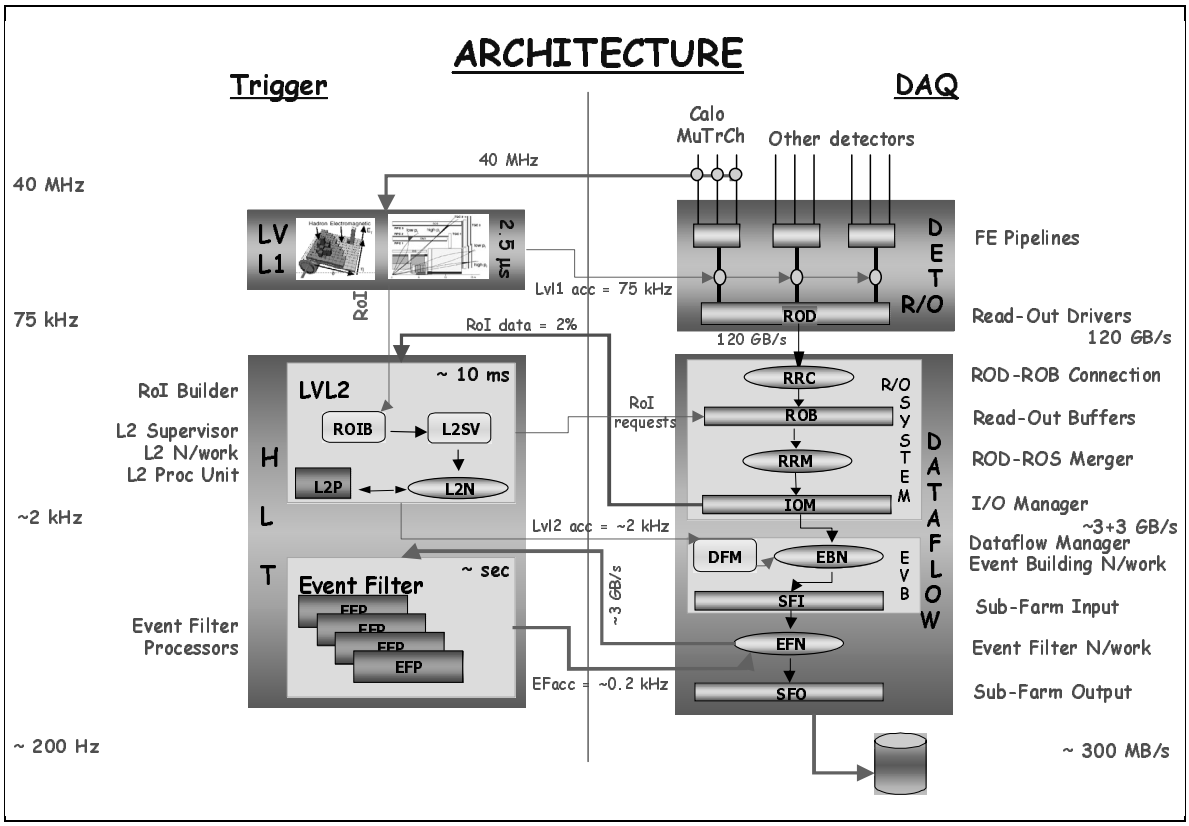

Figure 5-2 General architectural components, their relations and performances

Figure 5-3 depicts the architectural components (defined in the following text), and indicates where buffers, processors, supervisors and communication systems are located and how they relate. Table 5-2 summarizes the performance required from the different TDAQ system functions.

5.3.1.1 Detector readout

ROL (ReadOut Link): the communication link from the detector buffers (RODs), to the TDAQ buffers (ROBs). Each ROD may have one or more ROLs; each ROL corresponds to one event fragment. The ROL is expected to transport data at a rate equal to the maximum event fragment size times the maximum LVL1 rate (i.e. up to possibly 160 Mbyte/s).

ROD to ROB connection (RRC): the connection between the ROL and the ROB may be multiplexed, that is to say one or more ROLs may be connected to a single ROB. Therefore the RRC functional element represents the multiplexing of ROLs into ROBs.

ReadOut Buffer (ROB): all the TDAQ buffers are identical in design[1] and are used by all the ATLAS sub-detectors. The detector fragments are read out of the RODs and stored into TDAQ buffers; depending on the level of multiplexing provided by the RRC component, one or more fragments may be stored into a single ROB for the same event.

ROB to ROS Multiplexer (RRM): in order to reduce the number of connections into the LVL2 and event builder networks it is envisaged to funnel a number of ROBs into a single component. The RRM represents the ROB multiplexing capability in TDAQ.

ReadOut System (ROS): the component for serving, out of the ROB buffers, data to the LVL2 and event builder. This component could also be used to introduce a further level of buffering before the event builder: for example the ROS could present the Event builder with an aggregation of ROB contents for the same event, thus reducing the size of the Event Builder network and the number of messages traversing this latter. As for LVL2, the ROS will provide, upon request, individual ROL fragments.

5.3.1.2 LVL2

region-of-interest Builder (RoIB): the component which determines which fragments ought to be analysed by LVL2 for a particular event, based on information received from the LVL1 trigger. This component takes input from the LVL1 RODs and provide region-of-interest information to the LVL2 supervisors (see below). The RoIB will run at the rate of the LVL1 trigger.

LVL2 Supervisor (L2SV): the component which, for a given event accepted by LVL1, receives the information produced by the RoIB, assigns a L2PU to process the event and sends the L2PU the information provided by the RoIB (that is to say the list of ROL fragments constituting the complete RoI computed by LVL1). It receives, from the L2PU, the accept/reject decision for the event. If an event is rejected, the decision is passed to the ROS in order for this latter to remove (from the ROS buffers) the event. If an event is accepted, the decision is forwarded to the DFM in order for it to supervise the transfer of the corresponding event to the event filter.

LVL2 processing Unit (L2PU): the component which, using the information provided by the L2SV, requests event fragments from the ROS, processes the RoI (i.e. runs trigger algorithms in the event data belonging to the RoI) and produces a decision (accept/reject) for the event. The decision is passed back to the L2SV.

LVL2 Network (L2N): the networking system used to connect all the ROSes, LVL2 processors and supervisors for the purpose of moving RoI data and LVL2 decisions between the TDAQ buffers, LVL2 processors, and supervisory components. Data transport and its control share the same network.

5.3.1.3 Event Builder

Data Flow Manager (DFM): the element that receives the information about which events have been accepted by LVL2 and supervises the transfer of their data to the event filter. In particular it assigns an Event Builder buffer (the SFI described below) to an event and supervises the correct assembly of the event fragments in the SFI. It also provides a LVL2 bypass mechanism for when the TDAQ system, or one of its partitions, runs without the LVL2 sub-system. That is to say it also has a connection to the CTP and the sub-detector Local Trigger Processors.

1. Contrary to the ROD, which is a specialised detector buffer, RODs for different detectors may implement additional, detector dependent, functionality. For example digital signal processing in the case of the LAr detector.

Event Builder Network (EBN): the event builder will handle the assembly of full events at a rate of a few kHz. To achieve this performance several events will be built concurrently into many buffers (SFI's) by means of a switching network[1], which connects ROSes, SFIs, and DFM. Data and control (event builder) share the same network.

S*ub-Farm Interface (SFI)*: the buffer where a full event is built prior to being transferred to the event filter for further selection.

5.3.1.4 Event Filter

Event Filter Processors (EFP): a farm of computers (each called an Event Filter processing Unit, EFPU), to run the Event Filter algorithm. It is logically organised in terms of sub-farms, each associated to one or more SFI buffers. It may include a supervisory component to assign events available in the SFIs, to event filter processors.

Event Filter Network (EFN): the communication system connecting SFIs, event filter processing unit and SFOs.

Sub-Farm Output (SFO): the Event Filter output buffers, intended for the events accepted by the event filter prior to writing the events to permanent mass storage.

5.3.1.5 Detector Control System (DCS)

At the level of detail suitable of this chapter, the detector control system is a component on its own, without internal structure. It is interfaced to the data acquisition control.

5.3.1.6 Online system

Online Software Farm (OSF): the farm of computers on which the TDAQ software services, such as the run control and the monitoring facilities, run. Single partitions as well as the whole experiment are operated out of this farm. In addition to the computers used to operate the experiment, the OSF contains also farms of computers specialised in particular 'Online Software' functions such as:

- Database servers, holding the software (or firmware) to be run (loaded) in all the TDAQ computers: there will be servers local to clusters of homogeneous (i.e. performing the same function) computers as well as central, backup servers.
- Farms of computers on which 'online software' processes (control, information services, operational monitoring, etc.) run.
- Farms dedicated to event monitoring.

Online Software Network (OSN): the network connecting the Online software farm, the detector control system as well as the controllers and supervisors local to the TDAQ components. A more detailed organisation of this network, showing which TDAQ elements have a controller etc., is provided below in the detailed component views.

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

5.4 TDAQ DataFlow architectural view

Specialise generic architecture for the purpose of Data flow.

Shall contain: functional decomposition into DF packages and sub-packages; interfaces and boundaries between DF packages and sub-packages; main use-cases realisation; 'Event control and event flow' view which will include the rates and data volumes between DF packages and sub-packages (including type of communication).

5.5 TDAQ controls and supervision view

Specialised generic architecture for the purpose of control and supervision (eg show local controllers).

Includes both DCS and Online controls

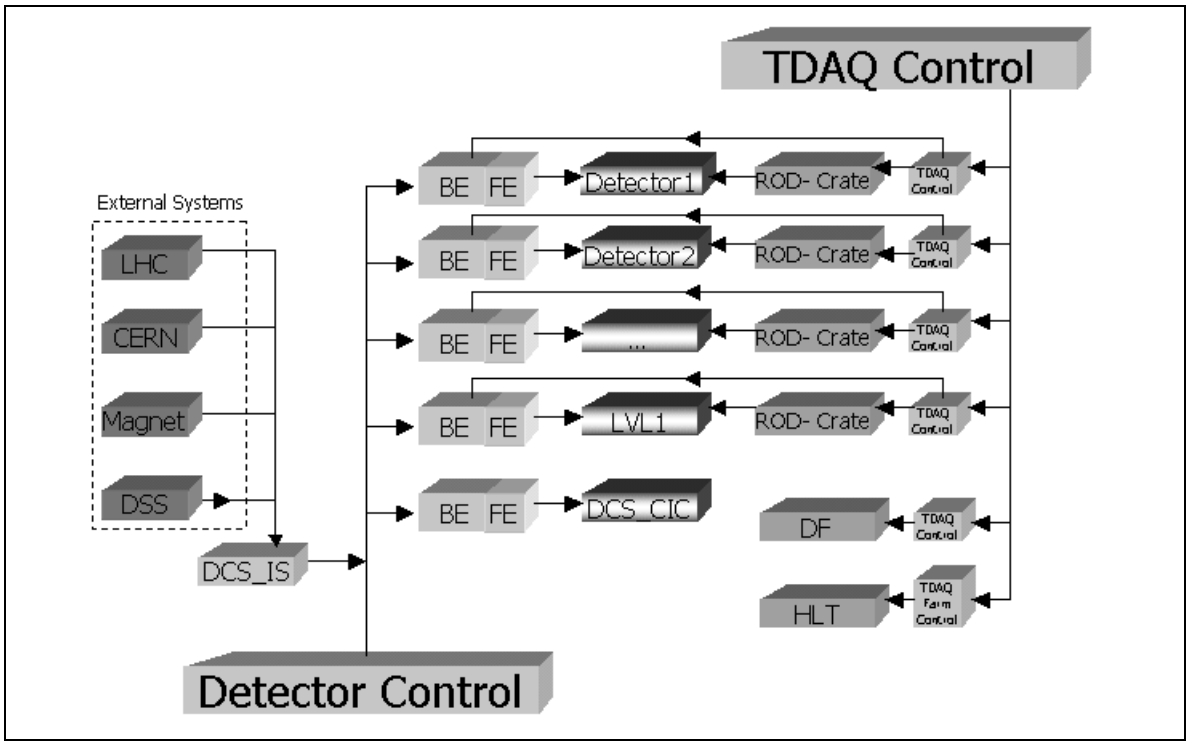

Figure 5-3 Controls View.

Remarks to this view:

This view has been chosen to illustrate the relation between the TDAQ Control and the DCS Control and explanations and some reasons for choosing this view are listed below.

- *Arrows represent the direction of command flow.*
- *Lines without arrows represent information exchange which is vital for correct decision making of the controlling master. Component boxes represent logical entities.*

- *Systems external to Atlas are shown where they have a vital importance to the experiment control. The controlling master must have knowledge of the machine status in order to take correct decisions.*
- *LHC: LHC machine status; CERN: Cern infrastructure; Magnet: Magnet status; DSS: Detector Safety System*
- *During data taking periods when TDAQ Control is active it has master control over the TDAQ system and the Detector control system.*
- *Outside data taking periods, when TDAQ Control is not active, the Detector Control system stays fully operational and controls all its connected units.*
- *Each detector can be controlled independently both from the TDAQ Control including the Detector Control during data taking periods, for example during installation and test phases, or outside data taking periods via Detector Control.*
- *The Command flow from TDAQ Control to Detector Control is performed from the TDAQ control on the level of detectors to the Detector Control on the same level.*
- *The presented components will be expanded and explained in more detail in the chapter on Experiment Control, when necessary details have been explained in the chapters on components and interfaces.*

5.6 Information sharing services view

Specialise the generic architecture for the purpose of information sharing services provided by the online software.

There are several areas where information sharing is used in the TDAQ system: synchronisation between processes, error reporting, operational monitoring, physics event monitoring, etc. There are different types of information which TDAQ applications may share in different cases. The Online Software provides a number of services to support all the possible types of information exchange between TDAQ software applications.

As it is shown on Figure 5-4, each of those services acts as a common communication bus for all the TDAQ systems and detectors. Information can be shared between applications belonging to the same TDAQ system, among several TDAQ systems, and to each of the TDAQ systems and detectors.

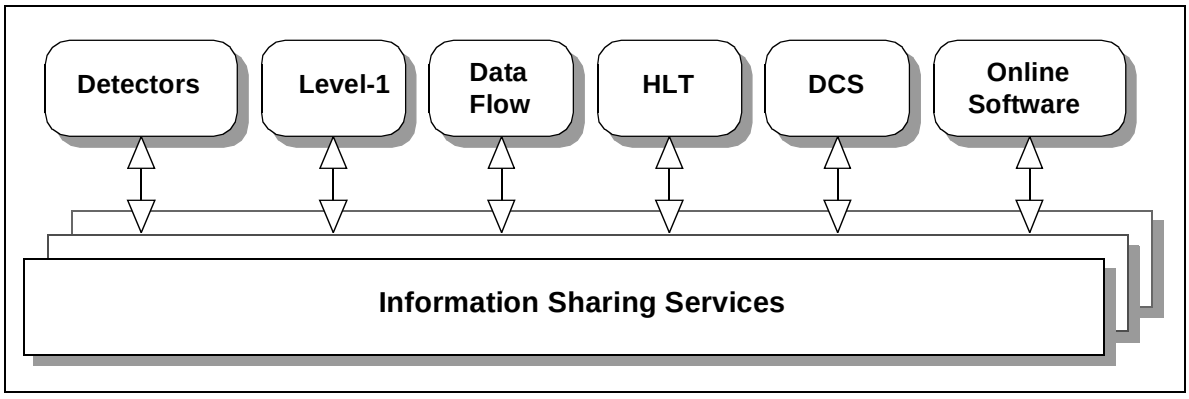

Figure 5-4 Information Sharing context diagram

All the Information Sharing services are partitionable in a sense that different instances of the same service are able to work in different TDAQ Partitions concurrently and fully independently.

5.7 TDAQ database view

database architecture: including where access to (in and out of) databases is done.

Remark: This is a very basic view of the databases in TDAQ. It is expected that more details can be presented when a common understanding is reached on the sharing of non-event data across DCS, DAQ, HLT and offline systems.

TDAQ and detectors are using the configuration databases to describe their system topology and the parameters which are used for data-taking. A variety of configurations can describe and combine different combinations of existing partitions which are prepared for different types of runs (physics, calibration, debug, shutdown, etc.).

The TDAQ and detectors are using the offline conditions databases to read and to store conditions under which the event data were taken. Such databases are used by the offline group for analysis and reconstruction of physics data, and by the TDAQ experts to analyse logs of the operational monitoring information stored during data taking by the online bookkeeper.

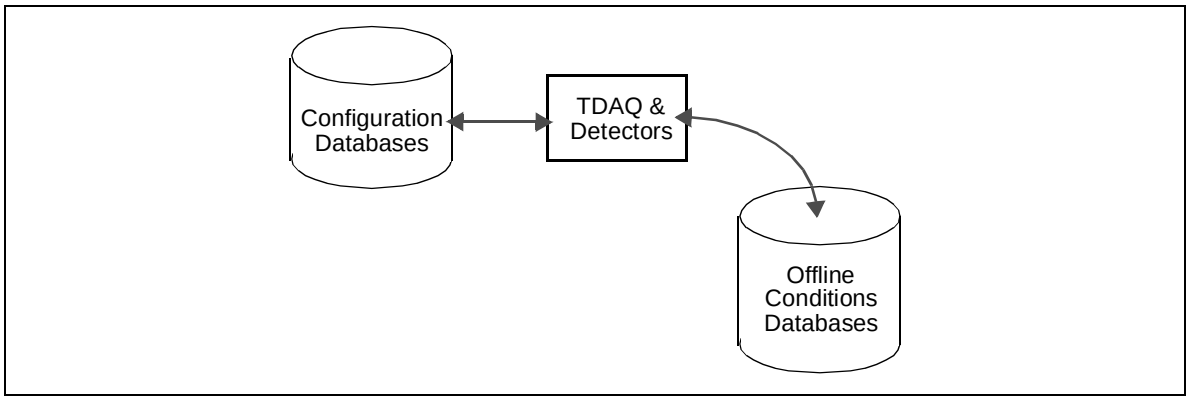

Figure 5-5 TDAQ database view

5.8 HLT view

The HLT issues relative to the generic architecture. It should probably include the organisation of the EF and LVL2 blobs, how HLT gets at the data.

5.9 Partitioning

The different partitions, and their related operations, are defined in detail in Chapter 3, "System Operations". Partitioning refers to the capability of providing the functionality of the complete TDAQ to a subset of the ATLAS detector.

The definition of the detector subset defines, because of the connectivity between RODs and ROBs, which ROBs belong to the partition. Downstream of the ROBs a partition is realised by assigning part of TDAQ (EBN, SFI, EF, online farm, and network) to the partition: it is a resource management issue. Particular examples of TDAQ components which may have to be assigned to a TDAQ partition are: a subset of the ROBs, as mentioned above, and a subset of the SFIs.

As regards the transport of the data to the allocated resources, the DFM plays the key role of routing subsets of ROBs to the associated subsets of the SFI's. In order for this to happen, in the case of partitions associated to non physics runs (i.e. when there is no LVL2), the DFM must receive, via the TTC, the triggering information for the active partitions. Hence the need for the full connectivity between the DFM and the O(30) TTC partitions. It is noted that this connectivity has not been defined yet, and it could be based e.g. on a network technology.

5.10 Baseline architecture implementation

5.10.1 Overview

The baseline architecture outlined in this section defines a concrete implementation for each of the components in the previous sections. The choices for the baseline are guided by the following criteria:

- the existence of working prototypes.
- performance measurements which either fulfil the final ATLAS specifications today or can be safely extrapolated (e.g. CPU speed of commodity PCs).
- the availability of a clear evolution and staging path from an initial small system for use in test beams, to the commissioning of the full ATLAS system.
- the overall cost-effectiveness and a cost-effective implementation of a staging scenario.
- the possibility to take advantage of future technological changes over the lifetime of the ATLAS experiment.

The proposed baseline architecture is a system that can be built with today's technology and can achieve the desired performance. It is expected that changes in the area of networking and computing will continue with the current pace over the next few years and will probably simplify various aspects of the proposed architecture. In addition optimization, in particular in the area of the ROB I/O, will still be studied prior to beginning their final implementation.

By making use of commercial off-the-shelf (COTS) components wherever possible the architecture will be able to take advantage of any future improvements in industry in a straightforward way. Only two custom components are foreseen in the final system[1]: the RoI Builder, of which only a single instance is needed, and the ROBins. The prototype for the latter is currently implemented on a single PCI board and about 400 will be needed for the full system.

The component performance figures which help to justify the proposed architecture can be found in Part 3.

1. S-LINK is another custom component which is under the responsibility of the detectors.

The baseline architecture is defined as a specialisation of the general architecture described in Section 5.3. The elements of Section 5.3 are assigned an implementation. Figure 5-6 depicts the baseline architecture including the few most relevant numbers related to its size and performance. Table 5-3 defines the assumptions under which the baseline architecture has been defined.

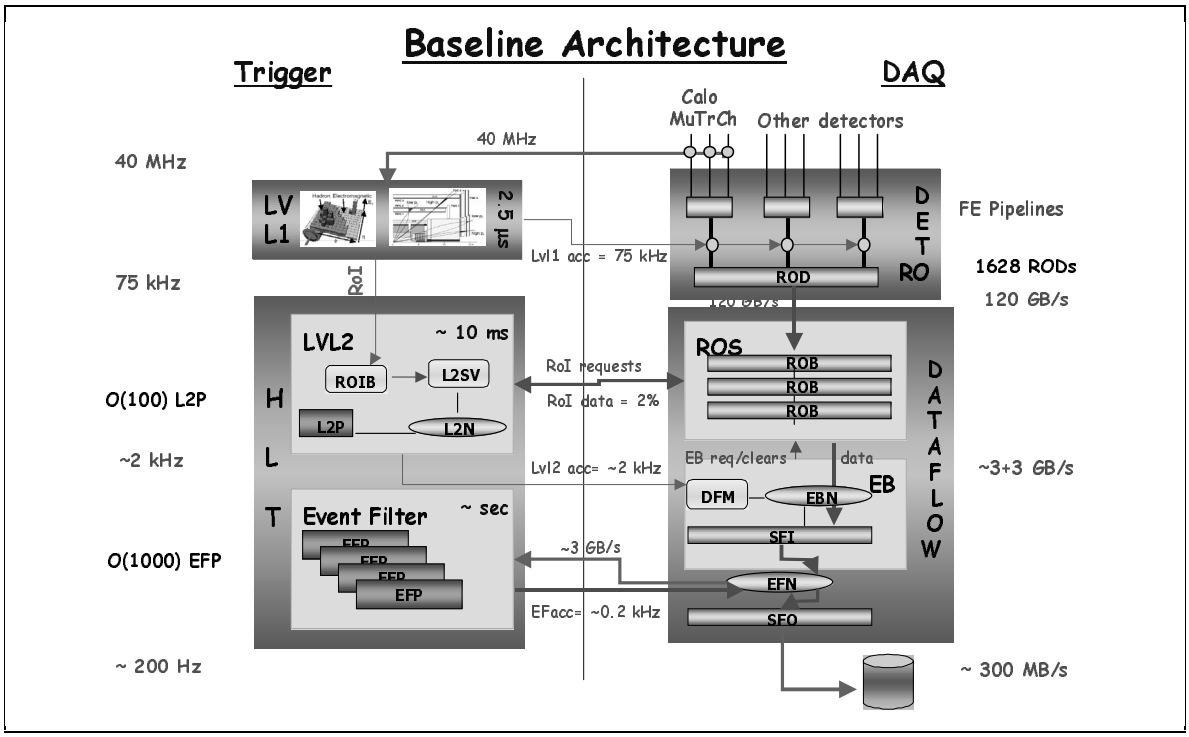

Figure 5-6 Baseline architecture

Table 5-3 Assumptions

Parameter	assumed value	Comments
LVL1 rate	75 kHz	TDAQ design LVL1 rate
Event Size (Maximum)	2 Mbyte	Assumes maximum fragment size for all sub-detectors
LVL2 rejection	30	Conservative assumption. Implies an EB rate of 2.5 kHz
RoI data volume	2% of total event size	Average value
L2PU events / second	~ 500	LVL2 decisions / sec / L2PU A L2PU is assumed to be a 2-CPU (4 GHz) machine
EFPU events / second	2	EF decisions / sec / EFPU a EFPU is assumed to be a 2-CPU (4 GHz) machine

A discussion of its robustness with respect to a variation of these assumptions is the subject of Section 5.12. Table 5-4 presents more detail. It summarizes the components that make up the

Table 5-4 Baseline implementation and size

Component	Type	Number	Technology	Comments
ROL	Custom	~ 1628	S-LINK 160 Mbyte / s	
ROB	Custom	~ 400	S-LINK@160 M byte / s (In) 1 Gb / sec out	Multiplex and buffer 4 ROLs Output channel sees only a fraction of the input data
ROS	Commercial	~ 65 GE switches 65 industrial PCs	10 x 4 Gigabit Ethernet	PC houses ~ 10 ROLs Individual ROBs are addressed via the concentrator switch
ROS	Commercial	~ 150	PCI	High end PC 3 ROBs are multiplexed over the PC's PCI bus
LVL2 Network	Commercial	~ 650	Gigabit Ethernet	Connects ROS and LVL2 processors
EB Network	Commercial	280	Gigabit Ethernet	Connects ROS and SFI
RoIB	Custom	1		
L2SV	Commercial	~ 10		
DFM	Commercial	~ 30	High end PC	Potentially one per TTC partition
L2PU	Commercial	~ 500	High end PC	Runs LVL2 selection algorithms
SFI	Commercial	~ 90	High end PC	Build (and buffer) complete events
EFPU	Commercial	~ 1600	High end PC	Runs EF selection algorithms
EFN	Commercial	~ 1700	Gigabit Ethernet	Connects SFI, EFPU and SFO
SFO	Commercial	~ 30	High end PC + ~ 1 TB disk storage	Buffers events accepted by EF and stores them on local permanent storage
File Servers	Commercial	~ 100	High end PC + 1 TB disk space	Holds copy of databases and software. Local to a group of functionally homogeneous elements (e.g. an EF rack)
DB servers	Commercial	~ 2	Large file servers	Hold master copy of databases, initialisation data, software.
OSF	Commercial	~ 50	PCs	To support the online sub-system, operate and monitor the experiment
OSN	Commercial	250	Gigabit Ethernet	Connects online farm to groups of functionally homogeneous elements (e.g. an EF sub-farm)

baseline architecture, its size (e.g. in terms of the number of components), and the performance. required by each component.

5.10.2 ReadOut Link

The ROL will be implemented using S-LINK technology [5-5]. Since the ROBs will be located near the RODs in USA15 these links will be short and will possibly be implemented in copper. About 1600 links will be needed.

5.10.3 ReadOut Buffer

The ROB is implemented as a PCI board taking a number of input S-LINKs, a PCI interface, and a gigabit Ethernet output interface [5-7]. The high-speed input data path from the RODs is handled by an FPGA. The buffers (with possibly a logical buffer space associated to each input ROL) will be enough to deal with the system latency (LVL2 decision time, time to receive the clear command after a LVL2 reject and time to move the fragment to the Event Filter). An additional PowerPC CPU is available on each ROB.

The ROB is located in USA15 near the ROD crates. The final ROB design is expected to support four ROL channels. The ROB realises the multiplexing (by a factor 4 x 1) indicated by the RRM function in the generic architecture.

5.10.4 ReadOut System

The ROS is a rack-mounted PC with multiple PCI buses, multiple ROBs per ROS, and outputs towards the LVL2 and event builder networks (see Figure 5-7).

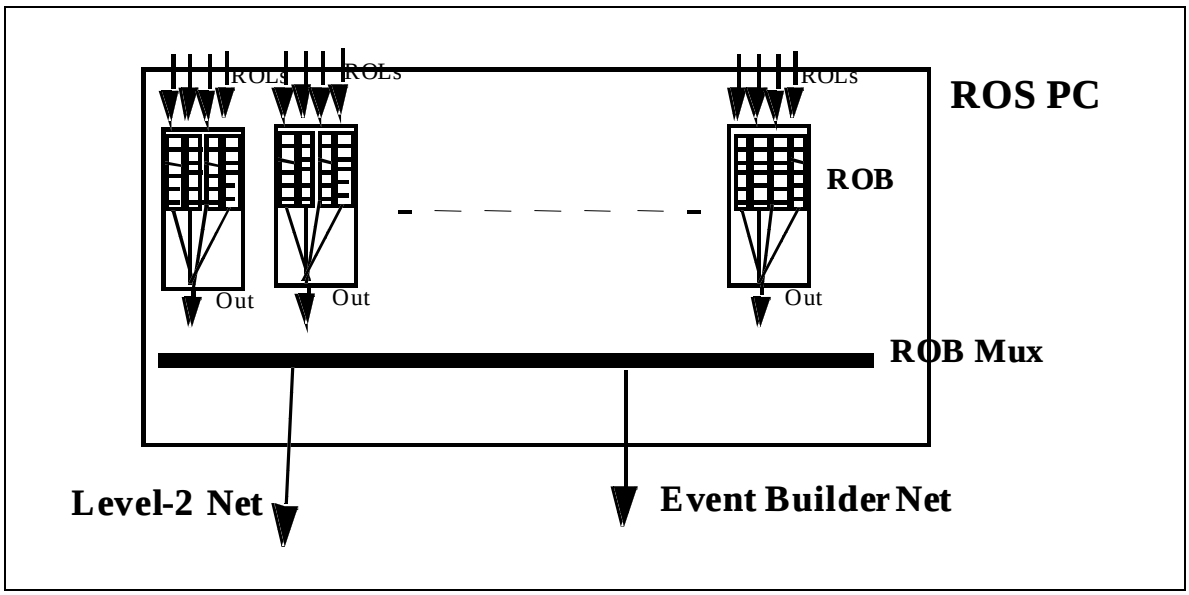

Figure 5-7 ROS architecture

The ROS houses a number of ROB modules, each multiplexing four ROLs into a single output channel (the RRC component in the generic architecture), and multiplexes the ROB outputs (the RRM component of the generic architecture) into the central networks. The RRM multiplexing factor, i.e. how many ROBs may be concentrated into the central networks, depends on the physical number of links between the ROS and each of the central networks. Once this is fixed, it is limited by the gigabit Ethernet bandwidth and therefore depends on external parameters: in particular the average RoI size, the peak RoI fragment request rate per ROB, and the LVL2 rejection power.

The RRM component may be optimised in two different ways, depending on implementation:

- firstly one could include three ROBs per PC, each ROB with four ROLs and one PCI output. In this case the RRM component is implemented by the PC PCI bus. Requests for fragments coming from LVL2 and requests for super-fragments (sequential merging of up to 12 fragments) from the event builder are handled by the ROS, i.e. by the PC. Two Gigabit Ethernet interfaces connect the ROS to, respectively, the LVL2 and event builder networks.

  This option is called the bus-based ROS.

- secondly the ROS could house 10 ROBs in a PC, each ROB with four ROLs and one Gigabit Ethernet output. The RRM component is implemented using a 10 x 4 gigabit ethernet switch, which concentrates the 10 ROB outputs directly into four gigabit Ethernet outputs: two for the LVL2 network and two for the event builder network. The ROS PC, in this case, does not play any role in the process of transferring data between the ROBs, and the LVL2 and event filter. The ROS PC is responsible for housing the ROBs (power etc.), their initialisation and control, and possibly for monitoring.

  This option is called the switch-based ROS.

Bus-based and switch-based ROSes are depicted in Figure 5-8. The optimization of the ROS architecture, as indicated previously, will be the subject of further post TDR studies, with the objective of making a choice on a time scale compatible with the production of the ROB element. The current ROB prototype indeed allows access to the ROB data via both the PCI and the Gigabit ethernet interfaces.

The ROSes are expected to be housed underground, so that the number of long fibre connections is much smaller than the potential O(1600).

5.10.5 LVL2 and event building network

All data transferred between the ROSes and, respectively, the LVL2 and event building functional blocks use Gigabit Ethernet[1] as the data link layer. The system includes a set of concentrating switches around the LVL2 units and a small set of central switches, for the LVL2 and the event builder networks. The concentrating switches are used to reduce the number of ports on the central switches whenever the total bandwidth requirements of a component allows to group it with others without exceeding the capacity of a single Gigabit Ethernet link.

1. The baseline architecture assumes the use of Gigabit Ethernet interfaces throughout the system.

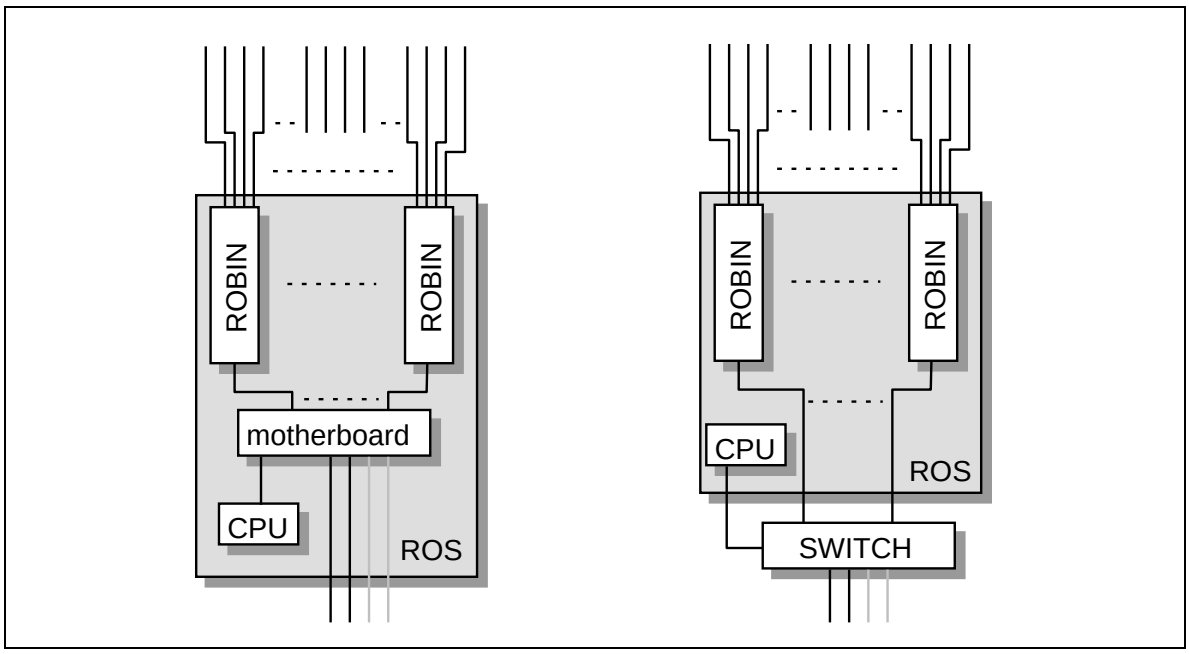

Figure 5-8 Bus and switched based ROS

The LVL2 and event builder central networks will, logically, be monolithic: in the sense that each network will connect to all sources and destinations. However, they may be physically organised either in terms of large monolithic switches or in terms of combined small switches. The detailed network topology, which is a function of the number of links downstream the RRM component, will be fixed at the time of implementation.

Software-wise the event builder and LVL2 networks are supported by the data collection component of the dataflow.

5.10.6 RoI Builder and LVL2 Supervisor

The RoI Builder is custom designed and built, and is described in [5-6]. It receives input from the LVL1 system via S-LINK. The output of the RoI Builder is again sent via S-LINK to one of multiple LVL2 Supervisor processors. The latter are normal PCs connected to the DataFlow network.

The RoI Builder design is modular and scalable: additional LVL2 Supervisor nodes can be easily added and will increase the performance linearly. Measurements done so far show that only a small number (< 5) of them will be needed.

The LVL2 supervisor is a PC running Linux. The only custom component is the S-LINK receiver card. Supervisor nodes can therefore be easily added and replaced in case of failure.

5.10.7 LVL2 Processing Units

LVL2 processing units will be normal rack-mounted PCs running Linux. Dual CPU systems are the most cost-effective solution at the moment, although that may change in the future. There will be O(250) LVL2 nodes in the final system. Initial setups during commissioning will be

much smaller. LVL2 processing units are COTS components and can be replaced and added at any time.

5.10.8 DataFlow Manager

The DFM will be physically implemented by rack-mounted PCs running Linux. Again dual CPU systems are the most cost-effective solution at the moment.

Although there will be only one (logical) Data Flow Manager per partition, the need for multiple partitions may require a set of DFM nodes at any given time, all capable of performing the same task. All DFMs except for the one running the real trigger partition will need a TTC receiver board to receive detector specific triggers. DFMs receive input from the LVL2 Supervisor communicate with the SFIs and the ROBs, and provide back pressure (to LVL2 or to the appropriate local trigger processor when taking data in a sub-detector TDAQ partition).

5.10.9 SFI

The SFI components need no special hardware except for a second Gigabit Ethernet interface that connects them to the Eventfilter network.

The SFI's are rack-mounted PCs running Linux. The actual event building requires a lot of 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.

5.10.10 Event filter Network

The current baseline for the EF network is a set of small networks, each connecting a set of Event filter nodes with a small set of SFI's (possibly even one) and SFOs. Such a modular combination of SFIs, SFOs and EFPU is called an event filter sub-farm. This allows maximum flexibility in choosing the number of Event filter nodes for each cluster. The input rate can be increased by simply adding more event building nodes to a given sub-farm. The sub-farm network will be implemented by networked clusters of EFPU and a sub-farm switch connecting clusters, SFIs and SFOs.

The clustering of EFPUs is justified by the fact that the Event Filter processes are CPU bound. Although each EFPU node may not require the full Gigabit bandwidth, gigabit Ethernet is nevertheless assumed throughout the Event Filter.

5.10.11 Event filter Nodes

Eventfilter nodes are rack-mounted PCs, again most likely dual CPU machines. Computing performance is more important than I/O capacity for these nodes. Due to the large number of CPUs required the use of blade servers which house hundreds of processors in a rack together with local switches might be a more attractive solution than 1U rack-mounted PCs.

5.10.12 SFO

The Sub Farm Output (SFO) nodes take the accepted events from the Eventfilter nodes and pass them to mass storage. In the current implementation these are normal PCs with an attached hard disk. They write events to the disk in a series of files. It is assumed that a different process picks up those files and sends them to their final destination in the CERN computing division.

5.10.13 Mass storage

The SFOs also provide the necessary buffering if the network connection to the CERN main site is down. Assuming that the SFOs have to buffer up to 24 hours of event data, of an average size of 1.8 Mbyte per event, at 200 Hz, they will need a total of 32 TB of disk storage. Today PC servers can be bought with > 3 TB of cheap IDE disk storage for less than $10,000. The SFOs will therefore consist of normal PCs but with a housing that allows the addition of large disk arrays.

5.10.14 Online farm

The online farm provides a diversity of 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 Event Filter sub-farm) and clusters of processors providing homogeneous functions (experiment control, general information services, central database servers).

5.10.15 Online network

The organisation of the online network, potentially the largest network in TDAQ, reflects the organization of the online farm. It is hierarchical, with networks local to specific functional elements (e.g. an event filter sub-farm or online clusters) and up-links to a central network providing the full connectivity. Gigabit Ethernet will be used throughout.

5.11 Scalability and Staging

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

Table 5-5 TDAQ required performance profile

Phase	Date (TDAQ ready by)	Performance (LVL1 rate)
ATLAS Commissioning	2005	12.5 kHz
Cosmic run	2006	25 kHz
ATLAS start-up	2007	50 kHz
Full performance	2008	75 kHz

The scalability of the TDAQ system is discussed on the basis of the above performance profile.

The baseline architecture implies that the detector readout system is fully available at the time of the detector commissioning (i.e. in 2005) irrespective of the required performance.

The implication of the performance profile is strongest on the HLT system. Indeed a reduced LVL1 rate requires smaller LVL2 and Event Filter farms as well as less SFI's (and SFOs). As a consequence, a time profile in LVL1 rate implies a time profile also for the size of the LVL2, Event Builder and Event Filter networks (a constant number of ROSes but less HLT nodes, less SFI's and SFOs).

The staging strategy chosen for the baseline architecture is therefore the staging of the farms (that is installing the required number of nodes at the required time) and networks. In the latter case, additional central switches (if a topology based on multiple central switches will be 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 evolution of the system size, as regards the HLT and the central networks is shown in Figure 5-9.

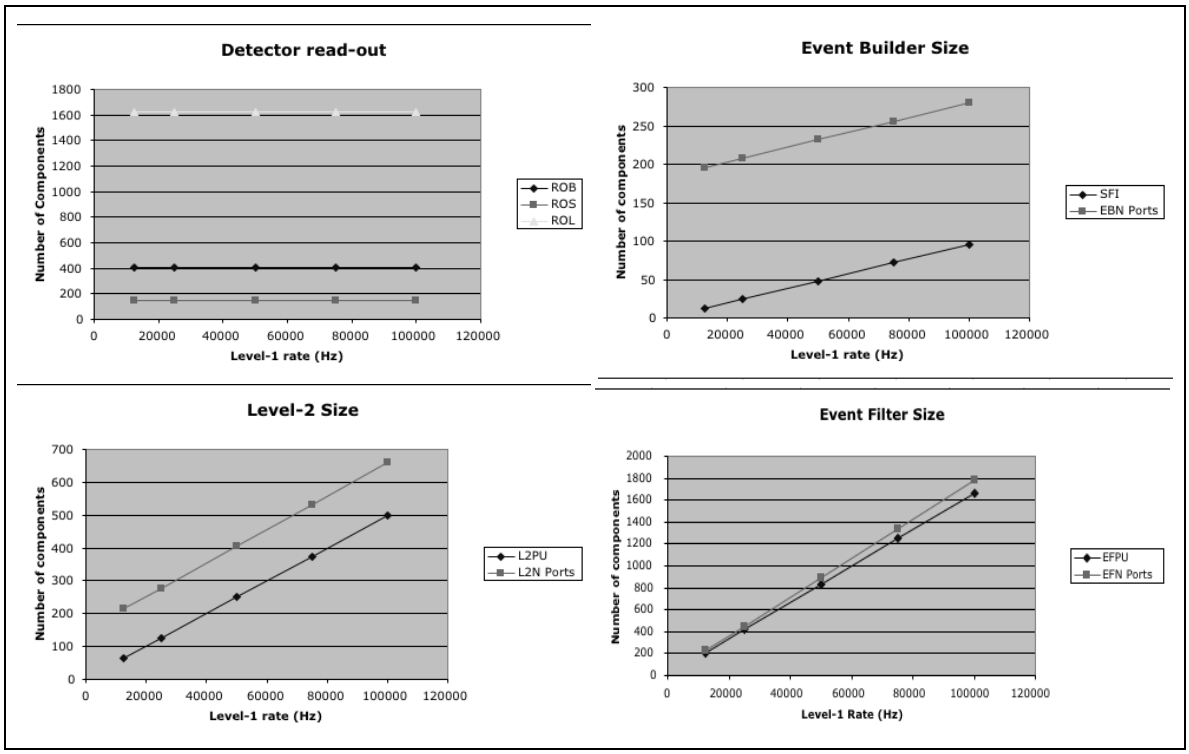

Figure 5-9 System scaling with staging

5.12 Robustness

Discuss: change in size of system when varying assumptions:

- LVL2 rejection
- LVL1 rate to 100 kHz

- performance of components (L2PU, EFPU)
- RoI data volume (average)
- RoI data volume: peak request.

5.13 References

5-1 ATLAS : technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN - ATLAS Collaboration. CERN-LHCC-94-43 ; LHCC-P-2. - Geneva : CERN , 1994.

5-2 ATLAS High-Level Triggers, DAQ and DCS : Technical Proposal - ATLAS Collaboration. CERN-LHCC-2000-017. - Geneva : CERN , 31 Mar 2000.

5-3 L1L2 interface.

5-4 DCS-OSW communications package.

5-5 Recommendations of the Detector Interface Group - ROD Working Group http: / / edms.cern.ch / document / 332389 / 1

5-6 R. Blair et al.) . A Prototype RoI Builder for the Second Level Trigger of ATLAS Implemented in FPGA´s. ATL-DAQ-99-016 (1999)

5-7 ROBin summary document

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. By failure we mean the inability of a component, 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 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:

- Minimize the number of single points of failure in the design itself. 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 first local measures are taken to correct it. 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 be available for later analysis if necessary. Where the error affects data quality the necessary information will be stored in the condition 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 for the error disappears.

Error response describes the immediate action taken by the component once it detects an error. This action will typically allow the component to keep working but 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 necessary action about 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 the failure condition is reported to receiving 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 shifter 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 shifter, 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 replacing 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. 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 judgement; 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 the message transport, message filtering and message distribution. Optional and mandatory attributes can be passed with the message. The facility allows receiving applications to request messages according to the severity or other qualifiers independent of its origin. A set of commonly used qualifiers will be recommended. These can for example include the detector name, the failure type (e.g. hardware), network, software failures, or finer granularity indicators like ‘Gas’, ‘HV’ etc. Along with man-

datory 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 case of major system failures. A count on the suppressed messages will be kept. Message suppression at the message reporting system level will also be possible.

An error message database may be used to help with the standardization of messages including their qualifiers. A help facility could be attached which would allow the operator to get detailed advice for the action on a given failure.

6.4 Error diagnostic and 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, which are specific for 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 actual component where it 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 a lower level. For example it might be able to pause the run and draw the attention of the shifter to the problem, or it might take a subfarm out of the running system and proceed without it.

The actual reaction to the failure will strongly depend 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. 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 decide when 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 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 any source and its destination in the system.

Given the size of the Trigger and DAQ systems, there is a fair possibility that at any given moment one of the sources of data may stop responding. Each element in the DataFlow can be generally seen as both source and destination.

Due to electronics malfunctioning, e.g. a fault in the power 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 is 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 where there is no need for a timeout 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.

Conversely there are situations where the timeout 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. A switched network is 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 timeouts can only be made once the system is fully assembled with the final number of nodes connected to the switches. Only at that moment, the normal operation timing can 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 calibrations. In the calibration mode the time for getting a data fragment may be higher due to the bigger amount of data to be transfered from a source to a destination.

6.8 Use cases

A short list of possible reactions on different levels (from inside an application to system wide) and their impact on data taking follows:

- Symptom: readout link not working properly.
    - Action: Reset of local hardware
    - Impact: Some parts of the event might be missing. If successful only an informational message would be send to the higher level. If not successful an error message would be issued.
- 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 mostly 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 higher level (who might decide to stop the run and take other measures like calling an expert).
    - Impact: Data missing in every event.
- Symptom: a LVL2 supervisor event request to a LVL2 node times out.
    - Action: retry a configurable number of times. Then take node out of scheduler and report to higher level.
    - Impact: Available LVL2 processing power reduced (as well as achievable LVL2 rate)
- Symptom: a LVL2 Supervisor reports that a LVL2 node repeatedly times out.
    - Action: Remotely reset the offending node. If successful, the node should come back into the run. If not successful then take node out of scheduler and report to higher level.
    - Impact: LVL2 rate is reduced while node is reset.
- Symptom: None of the nodes in an EF subfarm can be reached via the network (e.g. in 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 is reduced.

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.

6.9 References

6-1 ATLAS TDAQ Error Handling Policy and Recommendations

7 Monitoring

7.1 Overview

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

The monitoring can be done at different places in the DataFlow part of the TDAQ system: ROD, ROS, and SFI. Additional monitoring information can be provided by the LVL2 trigger and by the Event Filter due to the fact that these programs will decode data, compute tracks and clusters, count relevant quantities for simple event statistics, or for monitoring the functioning of the various trigger levels and their selection power. The collected monitoring information may be processed locally, e.g. in the ROD module or in dedicated processes running in the Event Filter. Additional processing may be produced by a dedicated Online Monitoring Farm possibly also in charge of calibration activities. Results will be sent to shift crew in SCX1 Control Room.

It is clear that monitoring cannot yet 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 just started at the level of the whole ATLAS community [7-1] [7-2]. They 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
- operational monitoring (throughput and similar, scaler histograms)
- hardware monitoring.

7.2.1.2 Data Collection monitoring

Here, it is DAQ specific monitoring which is addressed :

- 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 simulate the decision of the trigger stages on 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 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 offline 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 which records which signatures in the trigger menu were passed. Histograms will be filled locally on the processors where the selection is performed. With an accept rate of 1 kHz for LVL2 and 200 Hz 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 therefore 1.2 kbyte / s.

7.2.2.1.5 Physics monitoring

In addition to classification monitoring, the simplest approach to monitoring the quality of the physics which is sent to permanent storage is to measure the rates for some physics channel. 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 trigger 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 e.g. 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 will be monitored at both the hardware level 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 real beam conditions can be found. For example, the calorimeter trigger fills histograms, in hardware, of the ‘level 0’ rates and spectra of every trigger tower and can quickly identify, and if necessary suppress, hot channels. Hardware monitoring is also used to check the integrity of links between the successive steps in the trigger processor pipeline. The Central Trigger Processor (CTP) will be monitored mainly at the ROD level, using internal scalers and histograms. It will include beam monitoring, i.e. 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 at a lower rate than hardware monitoring, but with greater diagnostic power. 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 as part of 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 DC, will be monitored:

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

Other valuable pieces of information for monitoring are :

- LVL2 selectivity per LVL1 trigger type
- RoI sizes
- RoI occupancies per sub-detector
- RoI specific hit-maps per sub-detector

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, which could 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 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 finalised 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, the physics channel).

The results of the data flow monitoring will be sent to the operator via standard Online SW 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 purpose 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 having been transferred).

7.2.3 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 due to the fact that these programs will decode data, compute tracks and clusters, count relevant quantities for simple event statistics, and to 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 by e.g. DSPs installed directly on the ROD board allows sophisticated calculations to be performed and histograms to be filled. Sending these histograms to analysis workstations will then be performed by the ROD crate CPU (using the Online SW 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 having this first check 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.

An extended part of the detector is available at the ROS level, and monitoring at this level is therefore considered as a potentially interesting facility. A correlation between ROS crates is not

seen as needed because such a correlation may be obtained at the SFI level. Event fragments sampled at the level of the ROS could then 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 then the place where the calorimeter, muons and LVL1 want to have the first cross check of consistency between the LVL1 information and the raw values of the trigger towers. 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 in first priority for selection, it seems more efficient to have a dedicated monitoring farm running besides 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 online the alignment of the tracking system. This requires some thousands of selected events, either stored on a local disk of fed directly to the processing task. Then CPU intensive calculations are required to invert matrices which may be as large as 30000 x 30000. With a cluster consisting of 16 PCs (as available in 2007, i.e. 5 GHz CPU clock, 1 Gbyte of fast memory and 64-bit floating point arithmetic unit), this can be made 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 Monitoring destinations and means

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 which is 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 the monitoring information:

- **Event Monitoring Service** - is responsible for the transportation of physical events or event fragments sampled from well-defined points in the data flow chain to the software

applications which can analyse them in order to monitor the state of the data acquisition and the quality of physics data in the experiment.

- **Information Service** - is 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** - is a specialisation of the Information Service with the aim of transporting histograms. It accept several commonly used histogram formats (like 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 which are 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 Monitoring computing resources

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 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 will be a dedicated one (e.g. a VLAN) or the general purpose network 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 (clusters ?) 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 there some monitoring activities, in addition to the ones related directly to the selection operation. EF is indeed the first place in the data taking chain where the full information about the events is available. Decisions from the previous levels of the trigger system can be checked from both accepted and rejected (on a pre-scaled basis) events. 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),

- in dedicated monitoring tasks running in the context of the Event Filter (then, one should think of passing the information gathered in the filtering task to take profit of the CPU power already used).

As already stated, the first priority of the EF must be the selection procedure which should not be jeopardised by introducing some CPU intensive applications in the processing host. Monitoring in the EF should therefore be reserved to lightweight 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 7.2.3, 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 was 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, since intensive monitoring is likely to be more required during commissioning or debugging phases while physics quality is not the first priority, and conversely. The size of this Online Farm is still to be evaluated.

7.4 Archiving monitoring data

Data which are produced by monitoring activities should be archived by some bookkeeping service so that it can be cross-checked offline with more detailed analysis. One should also store (in a dedicated channel?) 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.

7.5 References

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

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

Part 2

System Components

8 Data-flow

8.1 (Possible introduction)

8.2 Detector readout and event fragment buffering

8.2.1 ReadOut link

Sub-detectors transmit event data accepted by the LVL1 over Front-End links 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 out in the Trigger & DAQ Interfaces with Front-End Systems: Requirement Document [8-1]. 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 [8-2]:

- 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 for the fibre version, 25 m for the electrical version.

To ensure homogeneity, the output of the ROD is defined by the S-LINK specification [8-3]. In addition, the raw event format [8-4] 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 used in COMPASS and in other LHC experiments, e.g. CMS. S-LINK is an interface definition; it only defines a protocol and recommends connector pin-out. As shown in Figure 8-1, the ROD end of the ROL is called the Link Source Card (LSC) and the ROS end the Link Destination Card (LDC). They are connected by optical fibres or copper cables. 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.

The DIG - ROD Working Group have also recommended that the LSC be placed on a mezzanine card to facilitate support and upgradeability [8-5]. The form factor of these mezzanine cards is based on the CMC [8-6] 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 / deserialiser (SERDES) chip which performs parallel-to-serial data con-

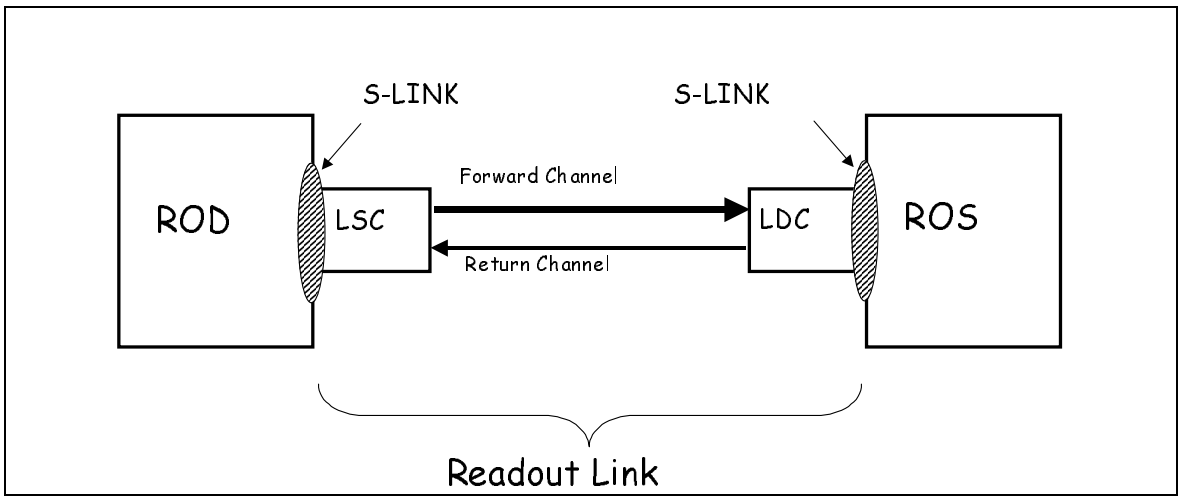

Figure 8-1 The relationship between the S-LINK and the ROL.

version 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. In fact the current LSC and LDC are physically the same card with different programs in the FPGA.

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 at 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) [8-7]. 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 [8-8] 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 demonstrat-

ed a bandwidth of 450 Mbyte/s into a PC. The next version of the interface, the FILAR, will have four ROLs on-board and should be ready for the April 2003 test-beam.

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 who 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 probably be handled by EP/ESS.

8.2.2 ReadOut subsystem

8.2.2.1 High Level Design

The ROS has three major components: the RoBIn, the IOManager and the LocalController. Figure 8-2 shows the relationship between the three ROS components and other relevant TDAQ components. A complete high level design cover both software and hardware aspects of the prototype ROS can be found in [8-9], only a summary is presented here.

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 full LVL1 accept rate, i.e. 75 kHz. All incoming ROD event fragments are subsequently buffered for the duration of the LVL2 trigger decision and the event building time for approximately 2% of the events. In addition, it must also provide ROD event fragments to the LVL2 trigger, and, for events accepted by the LVL2 trigger event fragments to the Event Building.

Due to these very demanding requirements the baseline RoBIn is designed and implemented as a custom hardware module. Section 8.2.2.2 describes the high level design of the prototype RoBIn component and the results of performance measurements.

The IOManager is deployed in the bus-based ROS scenario. Its main function is the servicing of data requests by the HLTs. According to the criteria specified in the data request, the IOManager collects ROB fragments from one or more RoBIns and builds a ROS Fragment. This fragment is then sent to a destination specified in the original request. The IOManager also receives from the HLTs the request to release buffer space occupied by ROD event fragments. These event fragments have either been rejected by the LVL2 trigger or successfully built by the Event Building. So as to maximise the overall performance of the ROS, the design of the IOManager allows a number of data requests and releases to be handled concurrently, that is to say, its design and implementation overlaps I/O operations with the processing of requests. In the baseline TDAQ implementation the IOManager is implemented as multithreaded software process.

In the deployment of a switch-based ROS the IOManager is redundant as all the RoBIns are connected to the DataFlow network. In such a scenario all the control, monitoring, error handling functionalities provided by the IOManager are provided by the LocalController.

The LocalController also provides a single interface point between the ROS and the Online software (configuration database, run control, message reporting, monitoring and process control). In particular the LocalController is responsible for retrieving the relevant information from the configuration database and sending it to the IOManager component. The IOManager also provides for the configuration, control and operational monitoring of its associated RoBIns.

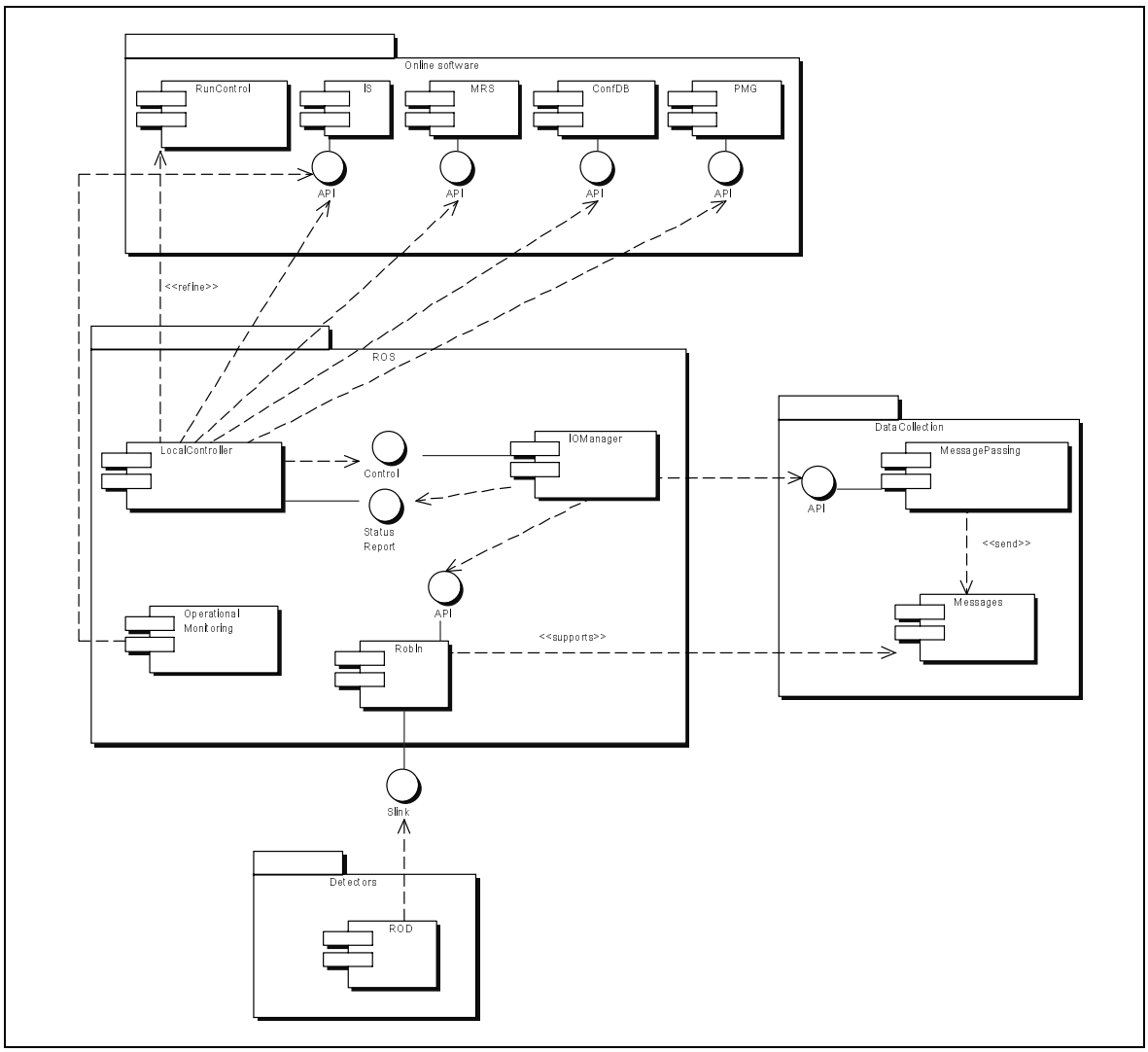

Figure 8-2 Main ROS components and their relationship with the other TDAQ components.

The LocalController is organized in specific units, each of which is connected to one or more IOManager units. In the TDAQ baseline implementation the LocalController is implemented as a multithreaded software process.

8.2.2.2 Design of the RoBIn

The RoBIn is located at the boundary between the detectors and the ROS. It receives ROD event fragments from a number of ROLs. The context of the RoBIn is shown in Figure 8-2 and its basic functionality can be described by the following four tasks:

- Receive ROD event fragments from the ROL
- Buffer ROD event fragments
- Send ROD event fragments, on request, to the High Level Triggers
- Delete ROD event fragments, on request, from the buffer.

The final prototype RoBIn takes into account the experience and results of studies from previous prototyping studies [8-10], [8-11] and the requirements on it are documented in the ROS-URD [8-12]. Its complete design and implementation are described in [8-13], [8-14] and [8-15], only a summary description is given.

Referring to Figure 8-3, the primary functions of the prototype RoBIn (receive, buffer, send and release) are mapped onto a small number of specialised building blocks. The data from two ROLs are stored in a separate buffers. All functions related to the receiving of ROD fragments from the ROLs, i.e. operations occurring at up to 75 kHz, are realised in an FPGA. A CPU is used to implement the functions of memory management, servicing of data requests and operational monitoring.

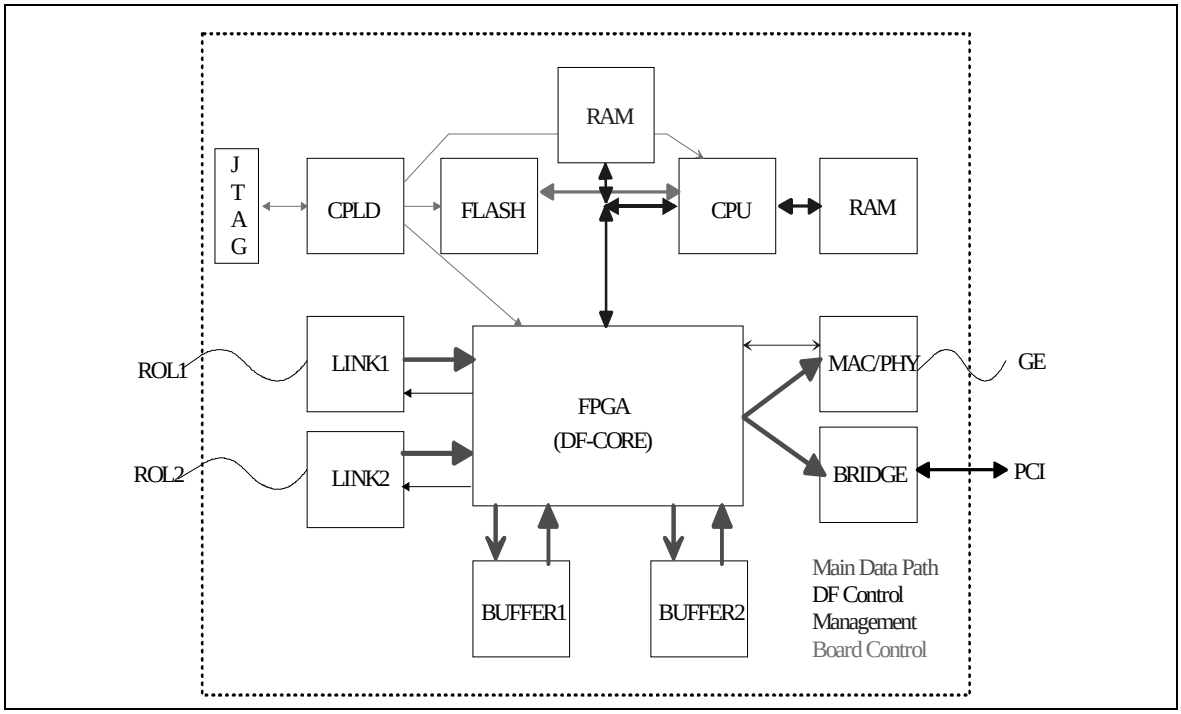

Figure 8-3 Schematic diagram of the final protoype RoBIn.

Input and output with other DataFlow components are provided by a PCI bus and a network interface. The provision of two interfaces allows I/O optimization studies to be performed. The design of the final RoBIn can be realised by removing (and not by adding) functionality, i.e. the PCI bus or network interface.

A software interface [8-15] defines the set of services provided by the prototype RoBIn and the messages to request these services and to transfer the responses. Using this software interface all remaining DataFlow architectural optimization studies can be performed with the same software. That is to say the software protocol via which I/O is performed is invariant with respect to which I/O interface is used. Specific operations related to a specific I/O interface have been encapsulated in separate modules.

8.2.2.3 Implementation and performance

The baseline architecture permits 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 de-

scribed in Section 5.10.4, one of these scenarios is termed the bus-based ROS and the other is termed the switch-based ROS. These terms reflect that in each case data from a number of RoB-Ins is collected exclusively via PCI bus or an Ethernet switch.

The deployment of the bus-based ROS is shown in Figure 8-4. It consists of two nodes: a ROS PC and the prototype RoBIn. The former is a desktop PC running the Linux operating system and has at least one Ethernet connection for the purpose of communication with the Online system. In addition, it has at least four 64 bit / 33 MHz and 3.3 V PCI bus slots. These slots are used to host the prototype RoBIn. The IOManager via the Message Passing interface (see Section 8.4.1.3) receives data requests and release messages, and returns ROS event fragments to the High Level Trigger components.

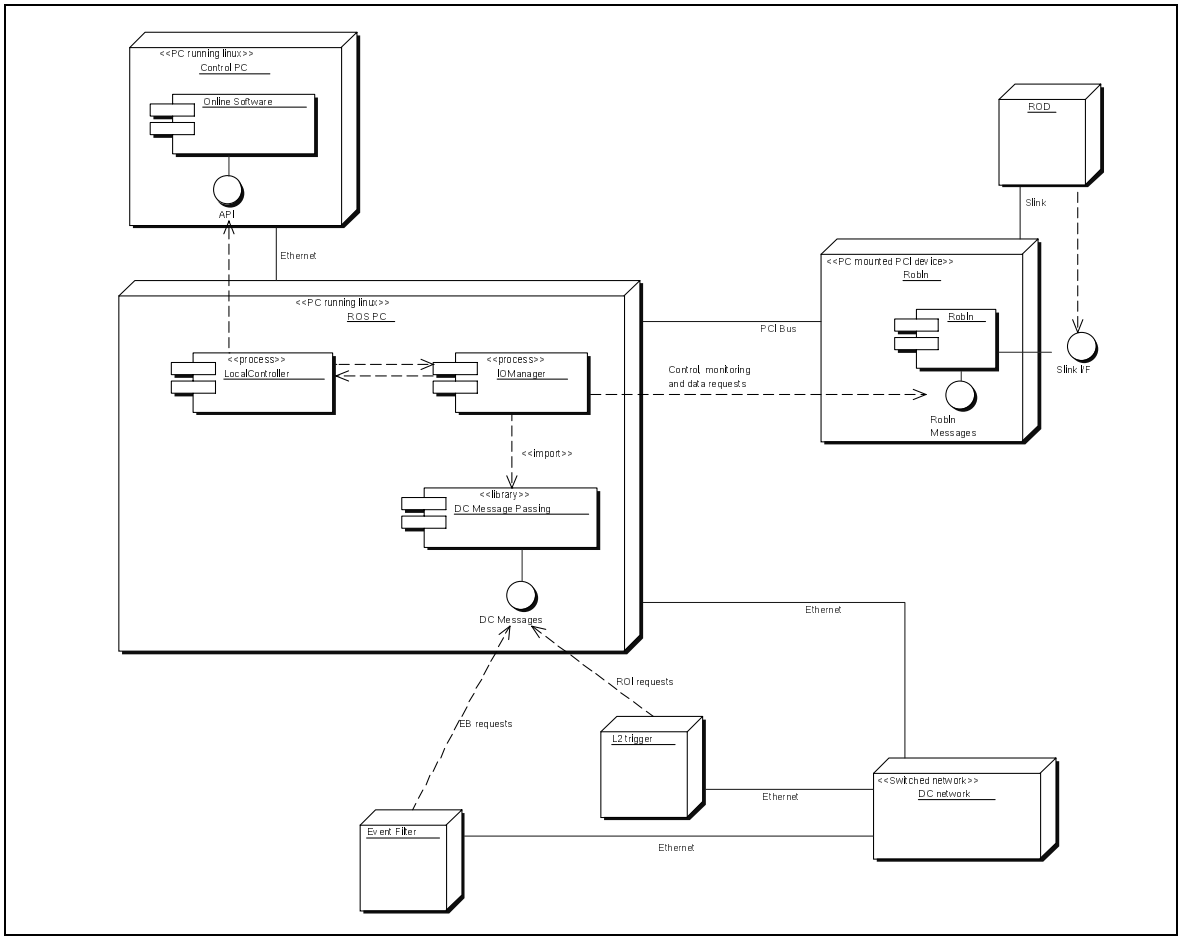

Figure 8-4 Deployment of the bus-based ROS.

Figure 8-5 shows the deployment of the switched-based ROS. In this case the RoBIns are now implemented on dedicated boards and in this scenario the LVL2 trigger requests event data fragments directly from the RoBIn without passing via the IOManager process.

Extensive measurements have been made on the performance of these two scenarios. In this section the results of the bus-based scenario are presented and the results on the switch-based ROS are presented in Section 8.4.2.3. The complete set of results are documented in [8-16].

Figure 8-6 shows the setup for the bus-based testbed. In this testbed an IOManager and a Local-Controller process were deployed on a standard 2 GHz Xeon PC with a single processor and a

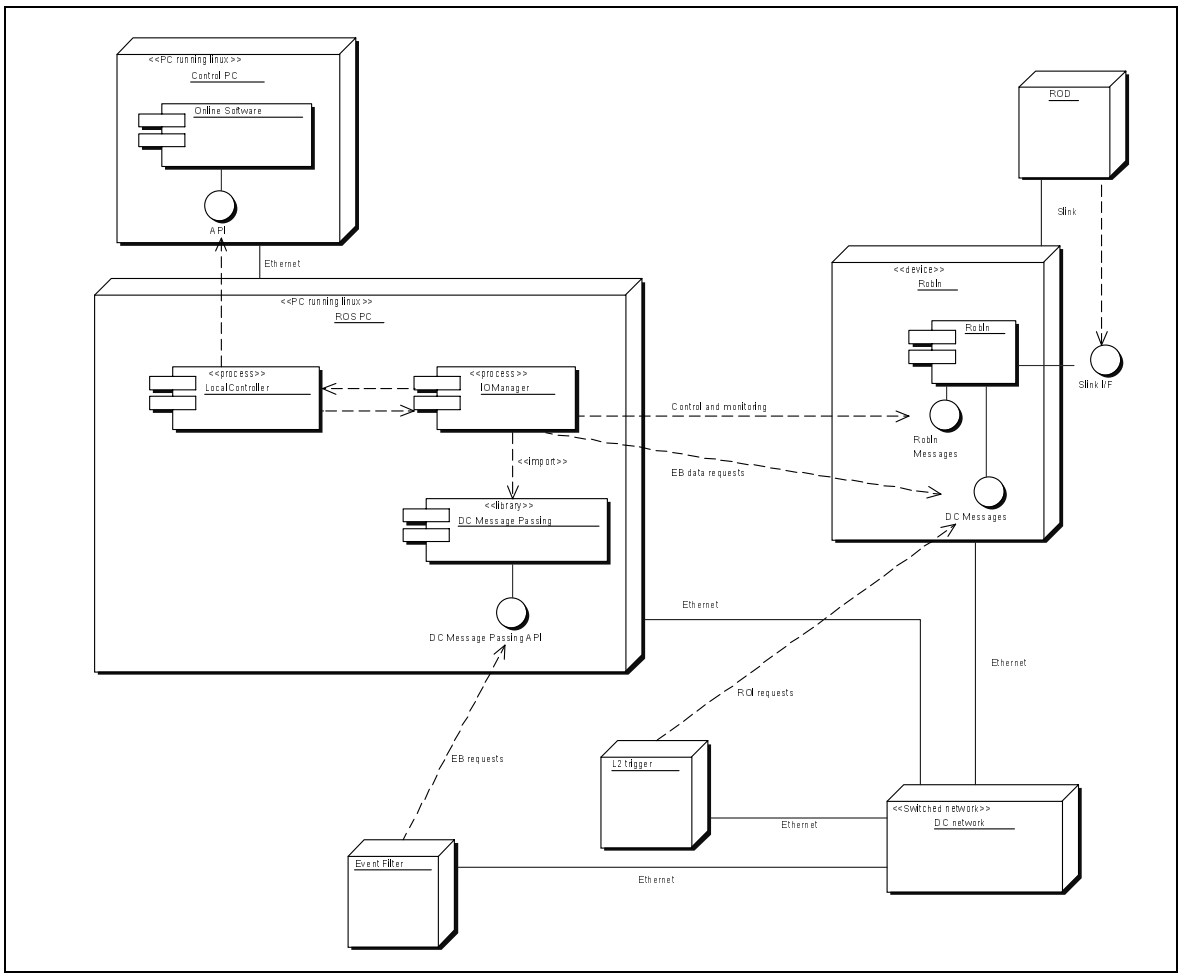

Figure 8-5 Alternative ROS deployment scenario.

66 MHz / 64 byte PCI bus, running RedHat Linux 7.3. As the final prototype RoBIns were unavailable at the time of these measurements, I/O with a RoBIn was emulated using a number of MPRACE boards [8-17]. These boards have the same physical PCI bus interface as the prototype RoBIn, and thus provide a very accurate emulation of the final devices. The MPRACE boards were not connected to a ROL and were programmed to generate ROB Fragments on demand.

The testbed has been operated in a standalone configuration, where the IOManager generated triggers internally and the ROS Fragments produced sent no where, and in a configuration where the IOManager was receiving real data request messages from the network and sending back the ROS Fragments to the requesting process. These two configurations allowed the aspects of the ROS performance associated to non-networking to be measured independently from those associated to networking.

Figures 8-7 and 8-9 show the maximum LVL1 rate that an IOManager was able to sustain for different percentages of LVL2 and Event Building requests and for different number of MPRACE boards connected to it. As only six MPRACE boards were available at the time of measurement, a software simulation of the RoBIn was developed to allow the testing of the IOManager performances with larger numbers of connected RoBIn modules.

Figure 8-9 shows how the simulation reproduces the measured results for up to six RoBIns and the results obtained for a larger number of RoBIns.

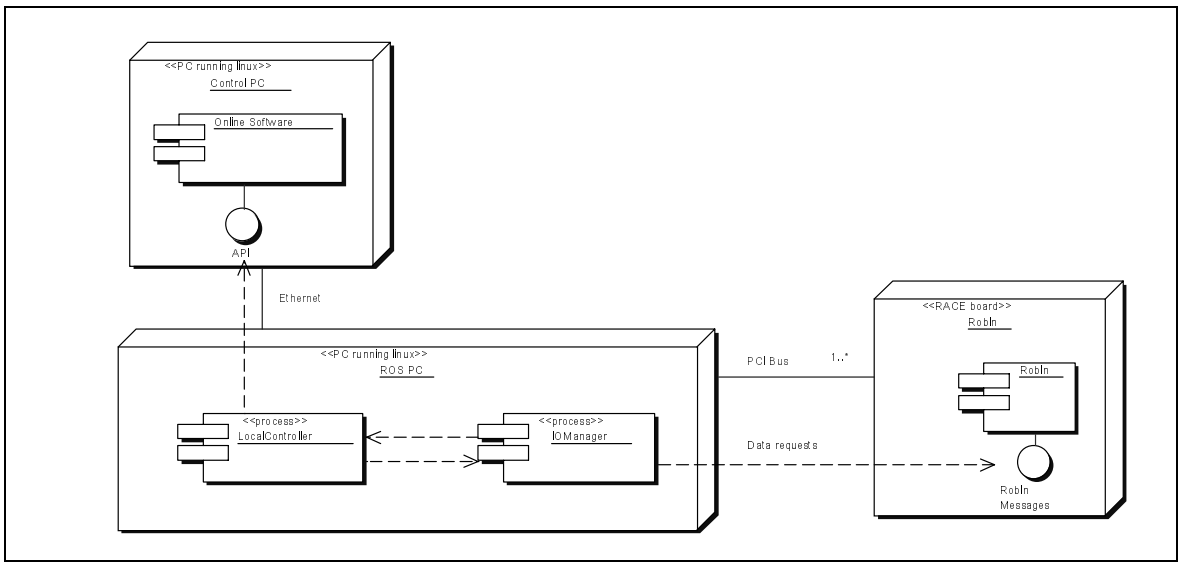

Figure 8-6 Setup of the testbed for studying the bus-based ROS system.

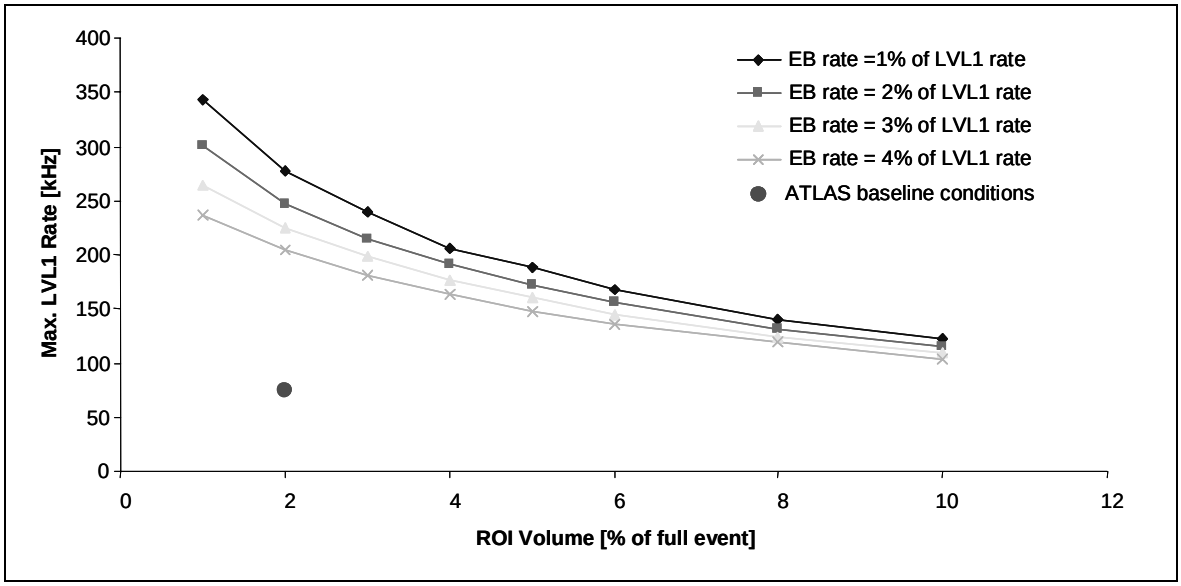

Figure 8-7 For a standalone bus-based ROS, the maximum sustainable LVL1 rate as a function of the requested RoI data volume for different percentages of Event Building requests.

The system fulfils the requirements for the final system with at least three PCI RoBIn modules, each equiped with four ROLs, connected to a same IOManager.

8.2.2.4 pROS

The Pseudo-ROS receives the detailed result records of the L2PUs for accepted events and participates to the event building process, such that the LVL2 detailed result appears within the full event record. 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(2 kHz) and the estimated size of the LVL2 detailed result is O(1 kbyte), it is purely a software process receiving event fragments via an Ethernet connection. That is to say that un-like 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

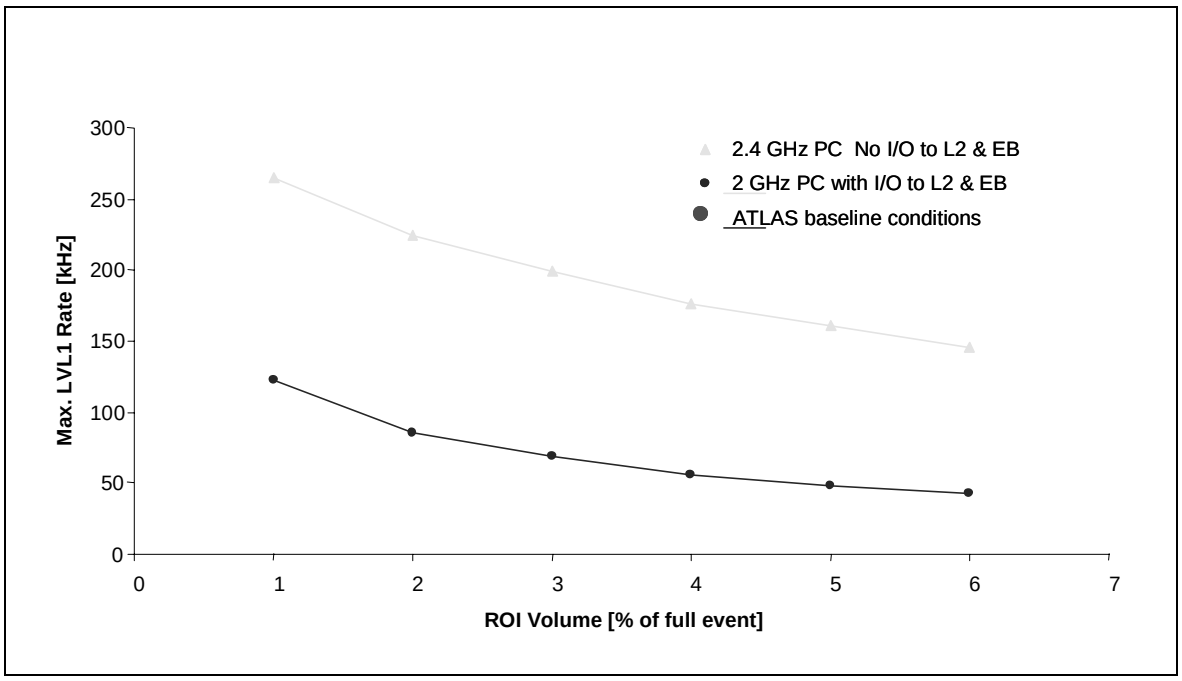

Figure 8-8 Maximum sustainable LVL1 rate for different fractions of LVL2 and Event Building rate of 3%, for a standalone bus-based ROS with real I/O to other DataFlow components.

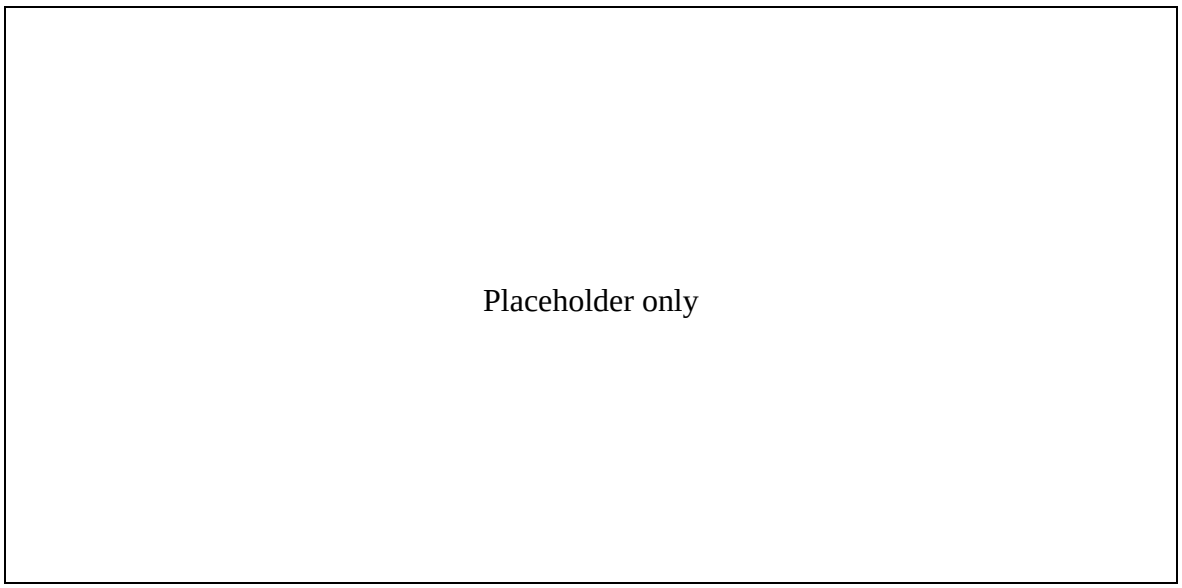

Figure 8-9 Maximum sustainable LVL1 rate for different fractions of LVL2 and Event Building requests for a standalone bus-based ROS with simulated PCI RoBIn input.

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 [8-18] and [8-19].

8.2.3 ROD crate data acquisition

The ROD is a sub-detector specific front-end element. It is located, in the event data flow, after the first level of on-line event selection, between the Front-end Electronics (FE) and the ROS. The ROD receives data from one or more Front-end Links (FELs) and sends data over the ROL

to the RoBIn. The ROD System covers all RODs and other functional elements at the same hierarchical level in the event data flow between the FE and the ROS. Those elements are grouped in crates. The crates contain ROD Crate Modules (RCMs) which can be: RODs, modules other than RODs (e.g. for control of the FE, for processing event data upstream of the RODs or for driving a TTC partition, as well as not fully functional ROD prototypes in laboratory setups or at test beam) and one or more ROD Crate Processors (RCPs). Each ROD Crate is connected to one or more ROD Crate Workstations (RCWs). This description is shown graphically in Figure 8-10.

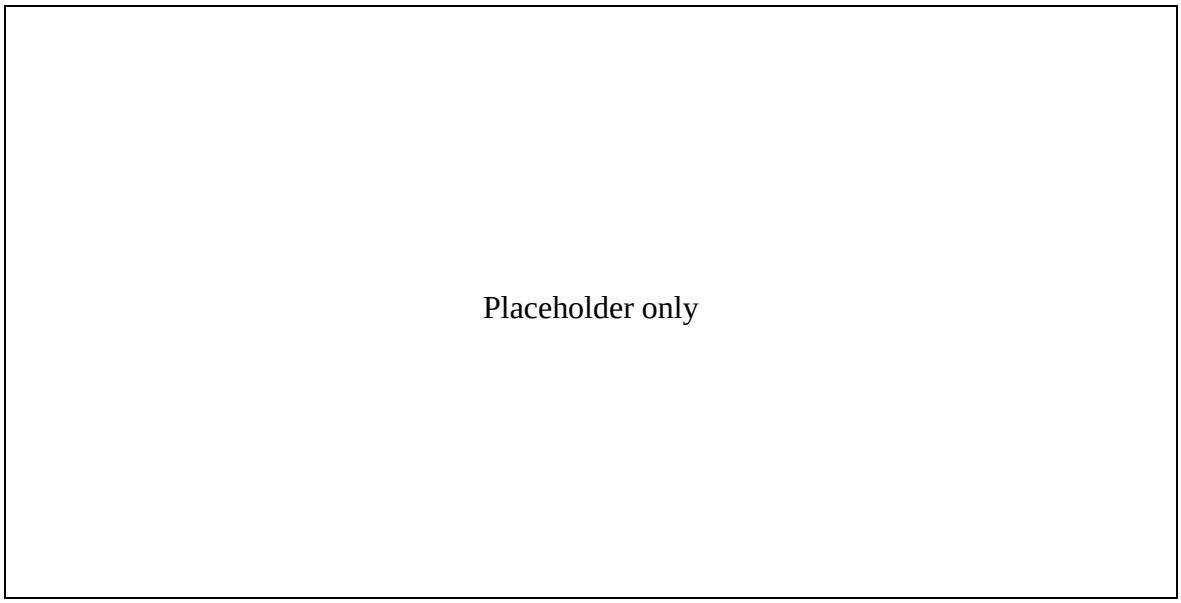

Figure 8-10 Block diagram of the context and contents of ROD Crate DAQ.

The sub-detectors need common DAQ functionality at the level of the ROD Crate for single or multiple ROD Crates in laboratory setups, at assembly of detectors, at test beam, and at the experiment during commissioning and production. ROD Crate DAQ [8-20] is part of the TDAQ system. It comprises all software to operate one or more ROD Crates and runs inside the ROD Crate 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.

8.2.3.1 High Level design

The ROD Crate configuration describes all necessary data required to fully configure all modules of the ROD Crate and the RCW(s). All ROD Crate configuration data are stored in the online configuration database implemented by the Online Software system. Several different ROD Crate configurations are stored in the database(s) concurrently. At initialisation of a run, one configuration is selected and loaded.

The ROD Crate control takes all necessary actions required to control fully all modules of the ROD Crate and the RCW(s). It is based on the run control of the Online system and implemented as a tree of run controllers, one per ROD Crate and others on the RCW(s) as necessary. The ROD Crate controller (RCC) drives all RCMs of the ROD Crate into well-known states and investigates their status. It may require interaction with the TTC system and / or DCS.

The primary event data flow of the ROD Crate transports event data from the FE over the FEL, the ROD and the ROL to the ROS. With respect to this flow of event data the ROD Crate DAQ provides for the collection of sampled event data from multiple RODs in the same ROD Crate and, at a rate O(Hz), the building of sampled event data from multiple ROD Crates. The event data is sample in the RODs via the VMEbus and allows for optional: ROD Crate data collection; ROD Crate event building; recording event data to mass storage.

Some ROD prototypes are not fully functional RODs and some are non-ROD modules, in particular at test beam. They have to be read out at the same hierarchical level in the event data flow as a ROD. ROD emulation provides the missing functionality. ROD emulation may be based on the primary or on the secondary event data flow. In both cases, an RCP is required.

Monitoring is another basic function of the ROD Crate DAQ. Different types of monitoring have to be distinguished depending on the different types of monitoring data they are collecting. Event data monitoring comes from the secondary data flow. Scaler and histogram monitoring is derived from event data. Operational monitoring reads values not directly derived from event data.

ROD Crate calibration provides sub-detector calibration at the level of the ROD Crate. It reads all event data from the secondary event data flow, processes them and calculates calibration data. The calibration data are written to data storage or to the calibration database.

ROD Crate event building is achieved by event building sources, one for each ROD Crate which participates in the event building, and one event building destination. An event building source is an output of a ROD emulation, monitoring or calibration activity. It sends all event data of the secondary event data flow over a Local Area Network (LAN) to the event building destination. The event building destination is the input of a dedicated ROD emulation, monitoring or calibration activity and usually runs on the RCW.

The basic functions of ROD Crate DAQ can be combined to provide high-level functionality for physics and calibration runs. They can also be used in different setups for physics data taking, ROD emulation, event building from multiple ROD crates, and small laboratory setups.

The framework of ROD Crate DAQ is organized into four layers: hardware, operating system, low-level services, and high-level tasks. A call for tender for the hardware of the RCP is under way. It is assumed that PCs will be used for the hardware of the RCWs. Linux is the first choice of operating system. LynxOS will be used for the RCPs in case real-time performance is required. The low-level services, like libraries and drivers, are organized into three different layers for hardware access, high-level task support and support for the Online Software.

A ROD Crate DAQ task is a high-level task for the ROD Crate controller, ROD emulation, monitoring, calibration or event building. ROD Crate DAQ tasks are provided as skeletons made of generic functions which may be extended by the sub-detector groups. Some standard functions are provided, e.g. for event building, which probably do not require extension.

The generic functions of the ROD Crate dataflow task are: the ‘input function’ which reads data from FEL, RCM or LAN, the ‘processing function’ which processes and selects data, the ‘output function’ which sends data over the ROL or LAN, or to data storage, the ‘control function’ which communicates with the ROD Crate controller, and the ‘monitoring / histogramming function’ which communicates with the monitoring / histogramming of the Online Software. The specific tasks for ROD emulation, monitoring, calibration or event building are distinguished by the implementation of their individual functions.

8.2.3.2 Implementation

ROD Crate DAQ re-uses existing software where possible. The Online Software is used as is, with some adaptation to sub-detector specific needs, in particular, for configuration. The ROD Crate controller is adapted from the controller developed for the ROS. The ROS software is also used to provide skeletons for the different tasks of ROD emulation, monitoring, calibration and event building.

The initial software is being developed in conjunction with detector specific developers. The workplan [8-21] allows for a first distribution of ROD Crate DAQ to be available by June 2003.

8.3 Boundary and interface to the LVL1 trigger

The LVL2 trigger is seeded by the RoIs found by the LVL1 trigger. The information includes both the triggers which were passed and the details of where, in η and φ, the trigger primitives that caused the accept came from. This requires information internal to the LVL1 system to be passed on to the HLT. The interface between the LVL1 and LV2L trigger has been design and documented [8-22] and the collecting of the information and passing it on to the LVL2 trigger is the responsibility of the RoIB.

Figure 8-11 shows the RoIB and its connections to the LVL1 system. As the LVL1 accept rate can reach 75 kHz, the RoIB is designed to spread the LVL1 events over a small farm of processors which are referred to as supervisor processors. Each supervisor process receives a single data record containing the summary information for each event that it supervises. Thus, no single processor has to deal with the full LVL1 rate. The supervisors are responsible for a rudimentary form of load levelling. They are aware of the disposition of events that they send to LVL2 processors and need to make sure that events are dealt with in a timely way. They also need to be assured that no single processor is overloaded with pending events. The operation of the supervisors is described in Section 9.2.3.

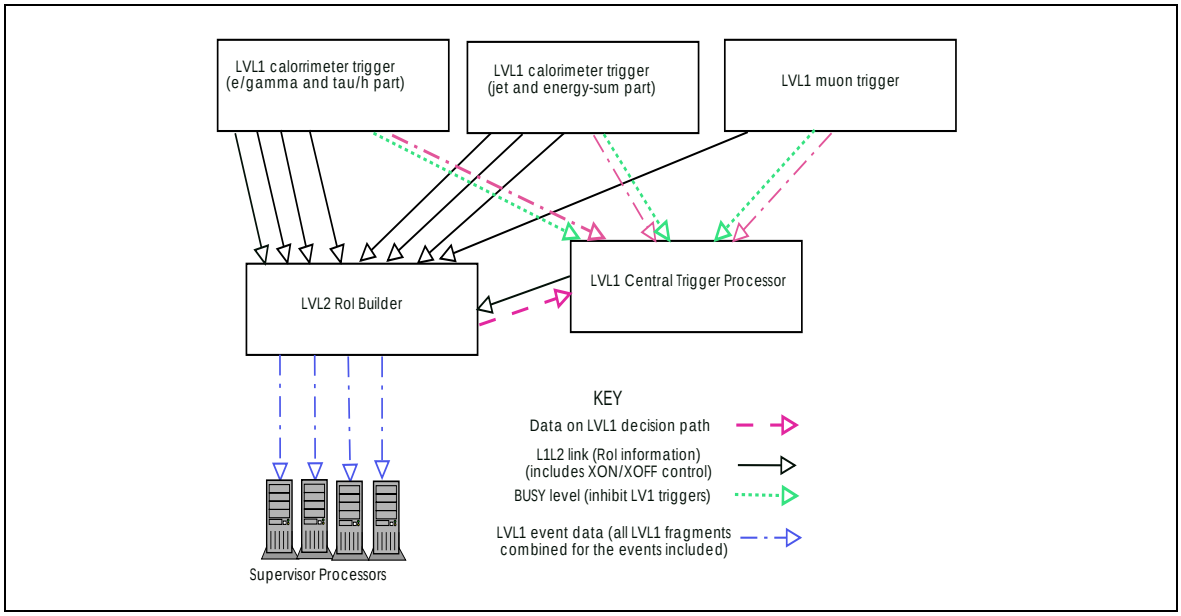

Figure 8-11 RoIB and its connections to the LVL1 system.

8.3.1 Description

A block diagram showing the LVL1 system and its interconnection with the RoIB is shown in Figure 8-12. Each link from the LVL1 system is an independent S-LINK input that sends a compact description of the event for that component. Each link is limited to sixty three 32-bit words or less per event. The various pieces of an event (referred to as fragments) arrive at the RoIB within one millisecond of each other. The RoIB assembles the event data and sends them to a supervisor processor which then initiates the LVL2 processing by passing a record to a LVL2 processor which includes the pertinent LVL1 RoI data.

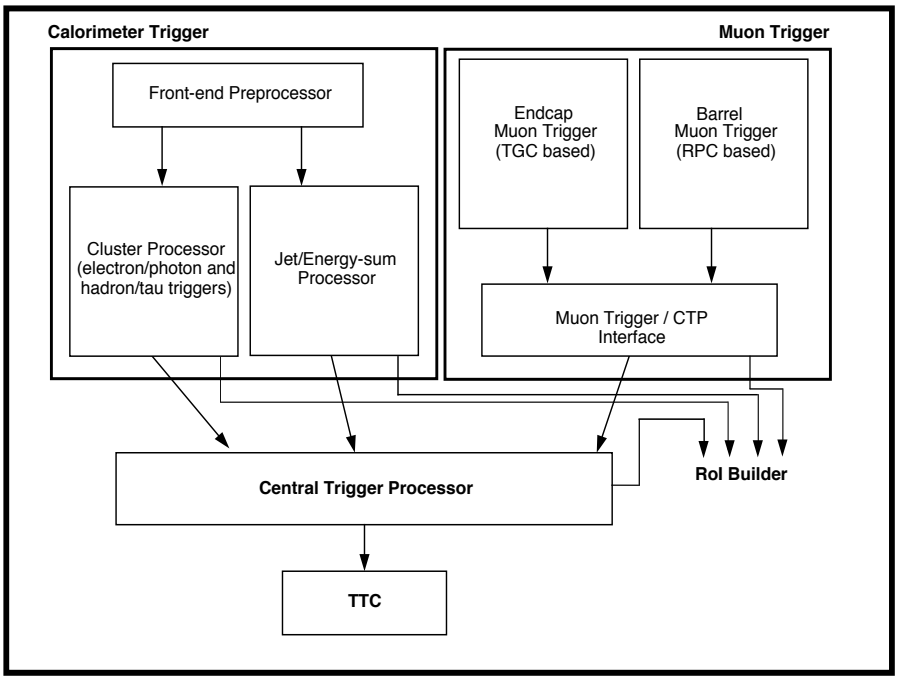

Figure 8-12 LVL1 system and its interconnection with the RoIB.

8.3.2 region-of-interest builder

The RoIB is a VME based system which uses FPGAs to combine the LVL1 fragments into a single record. It is composed of two parts. A pair of cards buffer the LVL1 input and direct fragments to cards which assemble individual events. Twelve inputs are adequate. This will include both the LVL1 fragments and an independent TTC input to assure consistency between LVL2 and the ReadOut system. The input cards communicate over a dedicated backplane connection to one or more ‘builder’ cards that provide four outputs for assembled events. Figure 8-13 shows the system organization. The system can service four supervisors with a single ‘builder’ card and can be expanded in units of four by adding ‘builder’ cards.

8.3.2.1 Performance

A prototype of the RoIB was fabricated and tested in 1999. This version was built using a pair of

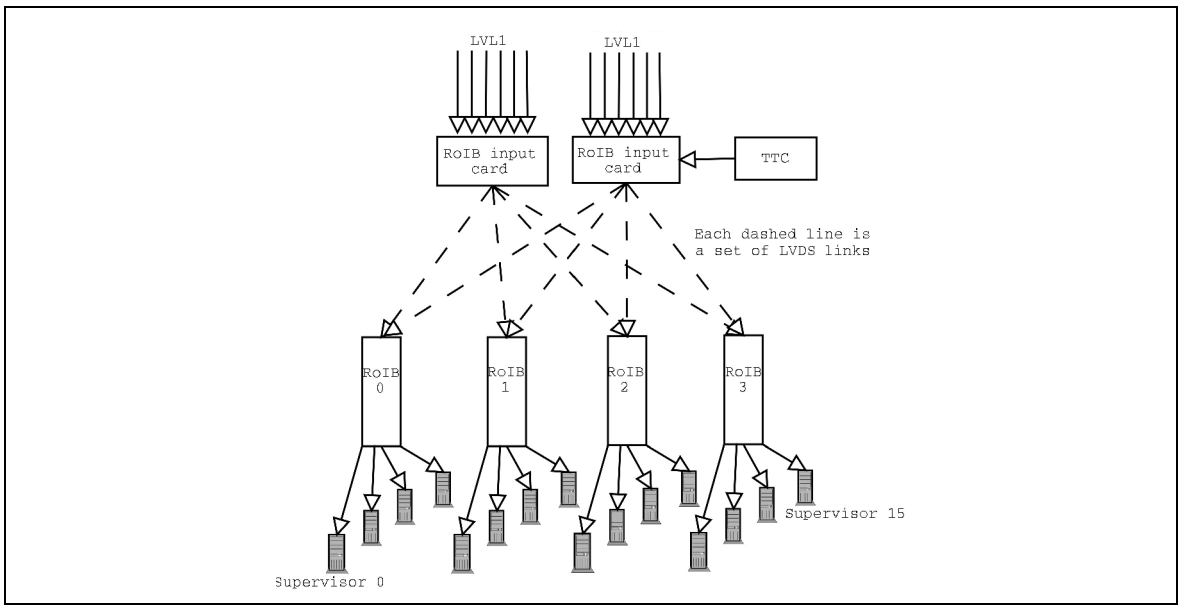

Figure 8-13 RoIB System organisation.

12U VME cards, an input card capable of handling six S-LINK inputs and a pair of builder cards able to output to a pair of processors. This system utilized 76 Altera 10K40 FPGA's and 8 10K50's. The system and early performance measurements are documented in [8-23].

This system was adequate to demonstrate a number of critical points. It showed that the idea of combining records from several sources using an FPGA-based device is feasible. It showed that four 300 MHz PentiumII machines were adequate to handle the 75 kHz rate. Subsequent tests with several prototype pieces of the LVL1 system (the muon-CTP interface and the calorimeter CPROD) made a start on debugging the component interfaces and further demonstrated that external inputs could be handled at the expected rates [8-24].

8.4 Control and flow of event data to high level triggers

8.4.1 Message passing

8.4.1.1 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 [8-25] and [8-26], here only its major features are summarized. Figure 8-14 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 to a L2PU for analyse.

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,

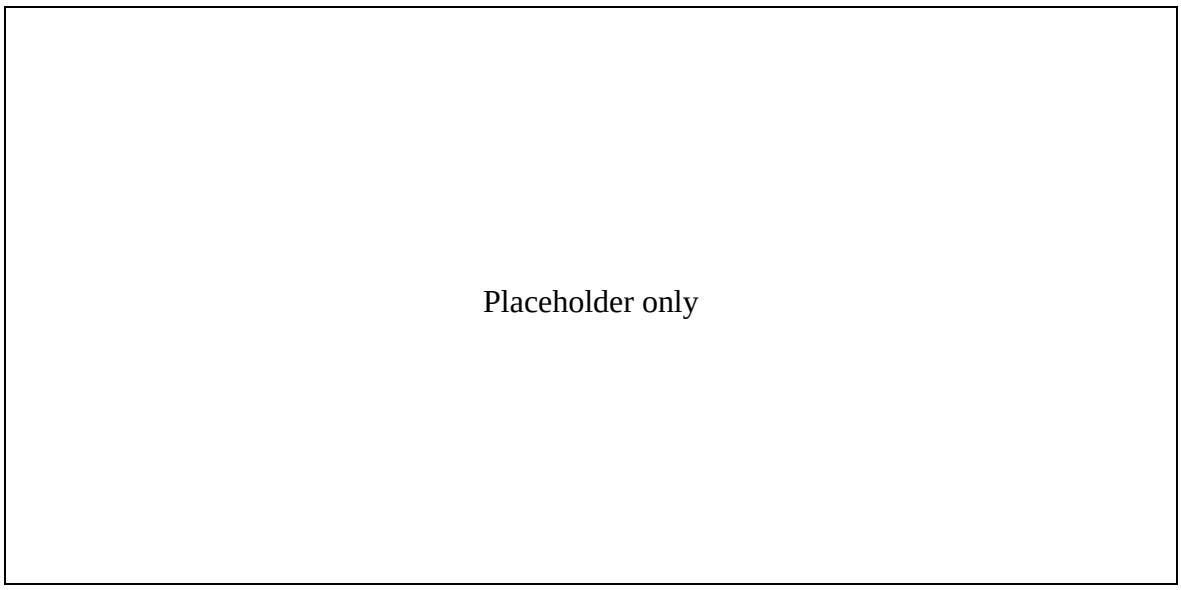

Figure 8-14 Sequence diagram showing the interactions between DataFlow components.

on average, in 1.6 RoI data requests messages being sent to a sub-set of the ROS units 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 to 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.

On reception of a group of LVL2 Decisions the DFM analyses 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. In the case of rejected events, the DFM multicasts a Clear message to all ROSs. This message contains the identifiers of events which should be cleared, i.e. those which have been rejected by the LVL2 trigger.

The SFI builds the event by sequentially requesting event date from all or some of the ROSs. The built event is subsequently sent to the EF subfarm for further processing.

Table 8-1 summarises the control and data message rates exchanged between the DataFlow components.

Table 8-1 Average message rates and bandwidth per DataFlow components.

Communicating components	Message type	Rate	Bandwidth
RoIB to a supervisor process	Data	7.5 kHz	4.8 Mbyte/s
A supervisor process to a L2PU	Data	100 Hz	50 kbyte/s
A L2PU to ROSs	Control	160 Hz	
ROS to a L2PU	Data		
A L2PU to a supervisor process	Control	100 Hz	
A supervisor process to DFM	Control	750 Hz	
DFM to a SFI	Control	35 Hz	3.5 kbyte/s
A SFI to ROSs	Control	56 kHz	5.6 kbyte/s
ROS to a SFI	Data	57 kHz	56 Mbytes
A SFI to DFM	Control	35 Hz	3.5 kbyte/s
DFM to ROSs	Control	250 Hz	25 kbyte/s

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.

8.4.1.2 Ethernet

8.4.1.2.1 Introduction

For the correct functioning of the baseline architecture, validation studies have been performed on those Ethernet features which ensure that it can meet the performance and functional needs of the DataFlow. These features are:

- Performance of switches in terms of: throughput, packet loss, latency and Media Access Control (MAC) address table size;
- Flow Control at different levels, i.e. across switch and between Ethernet nodes;
- Virtual Local Area Network (VLAN) implementation;
- Quality of Service (QoS);
- Broadcast and multicast handling.

To study the features listed two traffic generators have been developed. One tester implements Fast Ethernet (FE) and the other Gigabit Ethernet (GE). The FE tester implements 32 full duplex FE ports using Altera Flex FPGAs. The GE tester is based on the Alteon Gigabit Ethernet Net-

work Interface Card (NIC) which has a Tignon II PCI Ethernet Controller. In turn the controller has two programmable MIPS processors which enables the NIC to be system specific.

Both types of tester are capable of generating traffic with different packet sizes and an inter-packet gap according to a specified Constant Bit Rate (CBR) or a Poisson (negative exponential) distribution. In addition they allow measurements of packet loss and of latency to a precision level of 300 ns. More detailed information on the testers may be found in [8-27].

8.4.1.2.2 Basic measurements on switches

Switches must meet the throughput requirements of the architecture with a minimum latency and packet loss. The latter results in a degradation of the system's performance as it implies the use of timeouts and retries at the application level. In switches, packet loss occurs when their buffers overflow. The Ethernet Flow Control helps prevent buffer overflow, but it does not solve the packet loss problem completely, see Section 8.4.1.2.10.

Packet loss and latency have been studied for different frame sizes, loads (from 10% to 100% of the line speed), with CBR or with Poisson inter-packet gap, using unicast, multicast and broadcast traffic.

Figure 8-15 shows the results of a test performed on two different switches, with 1518 byte 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.

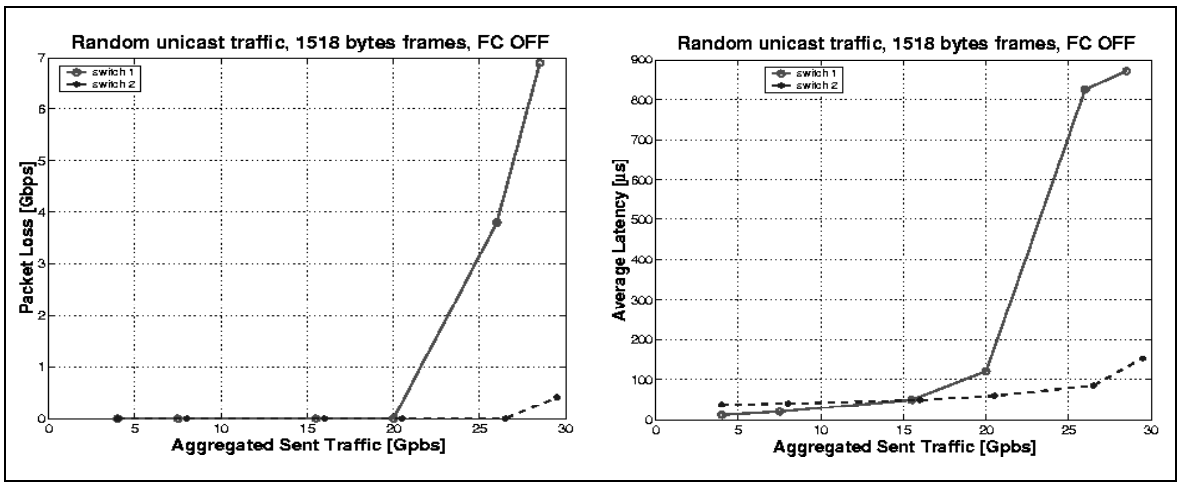

Figure 8-15 Switch measurements for unicast traffic, Poisson inter-packet gap, 1518 byte frames: (a) Packet loss (b) Average latency.

Consequently, any switch deployed must operate below the saturation point, to be determined by measurement, to avoid packet loss and the subsequent increase in latency.

Similar measurements using multicast and broadcast traffic have also been performed and again the results have proven to be vendor specific. In some cases the maximum throughput is surprisingly low, i.e. less than 10% of the line speed. This vendor dependency is one of the reasons for choosing unicast traffic (a request-response scenario) in the baseline architecture.

8.4.1.2.3 MAC address table size

The switch MAC address table contains the correspondence between the Ethernet MAC addresses and the associated switch's ports. If the Ethernet destination address is not known by the switch, the frame will be forwarded through all its active ports. This is known as flooding. The DataFlow network may have up to ~ 2000 nodes and it is necessary to ensure that all nodes can be memorized by any switch that is deployed in order to avoid the effects of flooding.

There are two types of entries in the MAC address table: static and dynamic. The static entries are added using a switch management tool. Once the network topology changes the user needs update the MAC address table. This is a great inconvenience for networks with large number of nodes. No human intervention is needed if dynamic entries can be used.

With dynamic entries, when a frame is received, the switch looks at its Ethernet source address and if the address is unknown to the switch, it is added to the MAC address table. If the network topology changes and the same MAC address is received on a different port the MAC table will be updated correspondingly. When the switch no longer receives frames from an address for a certain amount of time (aging time), it will erase that entry from the MAC address table. A typical value for the aging time is some hundred seconds. The removal of entries from the MAC address table is known as aging. The choice of a request-response minimizes the probability of aging.

To establish the MAC address table size of a switch the following test has been performed. A client[1] sends requests to the switch. Two NICs are set in promiscuous mode (they will see all the frames that arrive to them, regardless of their Ethernet destination address). One of them is used as a server which responds using the destination address from the received request as it's own Ethernet source address. The server will emulate any number of Ethernet nodes. The second NIC is used as a listener for flooding. After clearing the switch MAC address table, the client generates 4096 requests with different Ethernet destination addresses. As the MAC address table has been re-initialised it will flood the request on all its ports. Both the listener and the server will see the requests from the client. The server replies will allow the switch to learn the injected Ethernet addresses. The generation of the same 4096 addresses by the client is repeated. In this setup, the number of MAC address that have been accommodated by the switch is given by 2*4096 - N, where N is the number of frames received by the listener.

This method has been applied for different MAC address patterns on several switches. Table 8-2 summarizes the results from a switch showing a vendor specific behaviour. If the lower two byte of the MAC address are linearly increased, while keeping the others fixed (row 1) the switch learns 4096 MAC addresses. Similarly when the lower three bytes are randomly generated (row 4), or when there is a mixture of the linear and random patterns (row 5). However, when either the fourth and the fifth bytes (row 2), or the third and the fourth byte (row 3) are linearly increased the switch MAC address table size is less than eighty.

Table 8-2 MAC address table measurements. xx, yy and zz are chosen arbitrarily but remain fixed. αα are linearly generated numbers, while ββ are random generated numbers.

No.	MAC address pattern	MAC address table size
1	00: xx: yy: zz: αα: αα	4096
2	00: xx: yy: αα: αα: zz	70
3	00: xx: αα: αα: yy: zz	80
4	00: xx: yy: ββ: ββ: ββ	4096
5	00: xx: yy: ββ: ββ: ββ, 00: xx: yy: zz: αα: αα	4096

All the nodes in the DataFlow except the RoBIns are interfaced via standard NICs. The MAC address of a NIC has the manufacturer code reflected in the first four bytes, while the last two bytes of the address are most likely random. This address pattern, row 4 in Table 8-2, causes no problem to the switch. The RoBIns are custom built hardware and therefore there is freedom in choosing the hardware addresses. It is natural to linearly increase two of the bytes from the

1. based on the Alteon traffic generators and which can generate up to 4096 MAC addresses with different patterns

MAC address. The measurements show that for a proper operation of this particular switch the lower two bytes of the MAC address should be used. When choosing the RoBIn MAC addresses care must be taken to ensure that the first four bytes do not coincide with any established manufacturer code.

8.4.1.2.4 MAC address aging

The MAC address aging time should be large enough to avoid flooding once the DataFlow system is operating. In the proposed request-response message flow scenario all the nodes send messages to the network periodically. In the case of the ROS and Central switches, the SFI has the lowest rate (around 30 Hz), which imposes an aging time larger than 30 ms. The aging time in the switches tested is typically some hundreds of seconds. Yet flooding may occur in the L2PU switches. These switches never forget the addresses of the L2PUs, which request with an average rate of 10 kHz, but they may forget the addresses of some RoBIns. The L2PUs receive data only from a fraction of RoBIns and there is a low probability that an L2PU cluster does not request data from a given RoBIn for a period larger than the aging time. This probability is fairly low, and flooding is almost harmless, as it will most likely stop at the level of the LVL2 central switch (which has a much lower probability of forgetting that ROB address, as it accommodates the traffic between all the L2PUs and all the ROBs).

8.4.1.2.5 Virtual Local Area Network

The extended header of the Ethernet frame may include a VLAN tag immediately after the Ethernet addresses. This tag contains the VLAN ID, and also a priority field. The VLAN ID allows many logical (virtual) LANs to coexist on the same physical LAN, while the priority field allows layer two traffic prioritizing. VLANs are crucial for the proposed architecture, as they ensure a loop free topology and provide QoS support.

8.4.1.2.6 Loops and the Spanning Tree Protocol

The network topology of the proposed architecture contains loops, see Figure 8-16, which are illegal in the use of Ethernet as they disturb the MAC address tables for unicast frames and they keep sending forever multicasts and broadcasts (broadcast storms). In general the Spanning Tree Protocol (STP) is deployed to cut off the redundant links from a LAN in order to maintain a loop free topology. In the proposed architecture a loop free topology is achieved by the use of VLANs: a LVL2 VLAN and an EB VLAN. Several nodes need to be part of both VLANs: the DFM, the pROS and the ROBs.

The setup described in Figure 8-16 allows us to verify that VLANs eliminate illegal loops, and also to check if the STP is aware of VLANs. The tests performed showed that VLANs provide a loop free topology when the STP is disabled. When the STP was enabled one of the links in the loop was disabled. This is due to the fact that the STP is not implemented per VLAN on those switches.

If STP is not implemented per VLAN it can be disabled and VLANs used for maintaining a loop free topology.

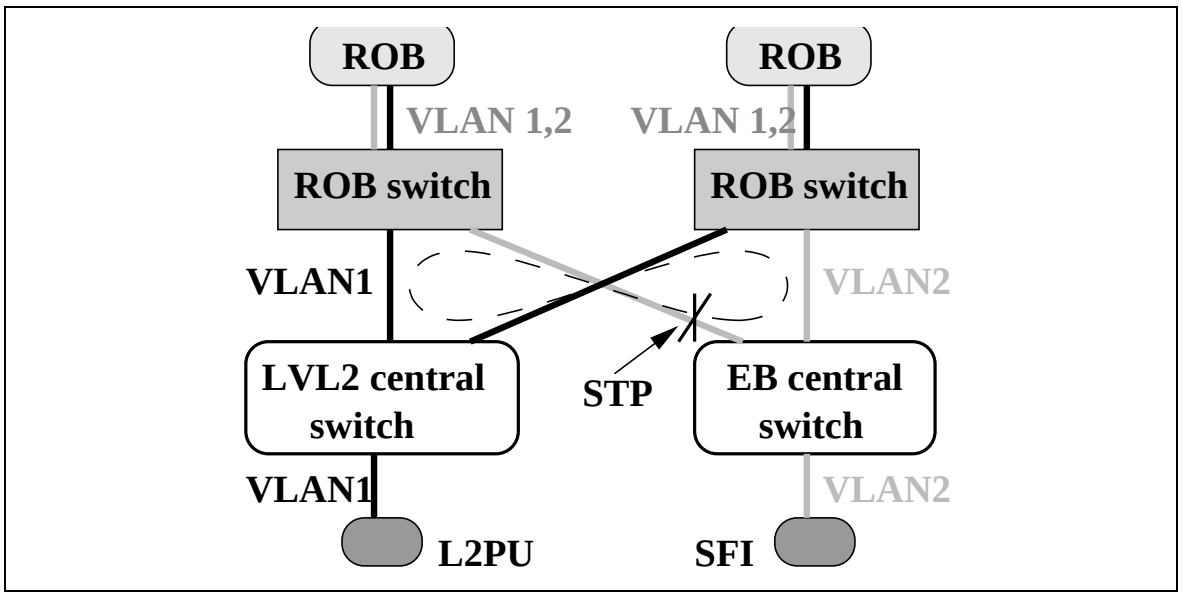

Figure 8-16 VLAN Ethernet loop setup. The potential loop appears in dashed line.

8.4.1.2.7 Traffic containment

Messages which are Multicast or broadcast can lead to the flooding of switches. VLANs ensure the containment of such traffic thus limiting the effects of flooding. For verifying this feature we use one transmitter and several receivers. The transmitter sends traffic to the investigated VLAN (unicast to an address which is not known by the switch, multicast or broadcast). One receiver is placed in every VLAN which is defined on the switch, plus an additional receiver outside any defined VLAN (i.e in the switch's default VLAN). All the receivers outside the VLAN we inject traffic to should receive no frames.

The procedure described above has been applied to switches in all of the following situations: one VLAN, two VLANs with no shared ports and two VLANs with shared ports. The VLAN bounds were not crossed.

This feature is useful as we can restrict the number of nodes receiving a multicast/broadcast message. For example, if we define a third VLAN in the DataFlow network, containing the DFM and the RoBIns, the clear message multicast by the DFM will be forwarded only to the nodes it should reach, i.e. the RoBIns.

8.4.1.2.8 Partitioning

VLANs are logical LAN within a physical LAN. This feature can be exploited to support partitioning. In the setup presented in Figure 8-17 independent traffic is sent to each of the two VLANs, which share no ports. For several traffic loads in VLAN2 the behaviour of the traffic in VLAN1 is measured. For the switches tested the performance of VLAN1 is insensitive to the amount of traffic flowing through VLAN2. Therefore this switch can be partitioned using VLANs. Note that this feature is highly dependent on the switch architecture, and no assumption should be made *a priori* for any switch.

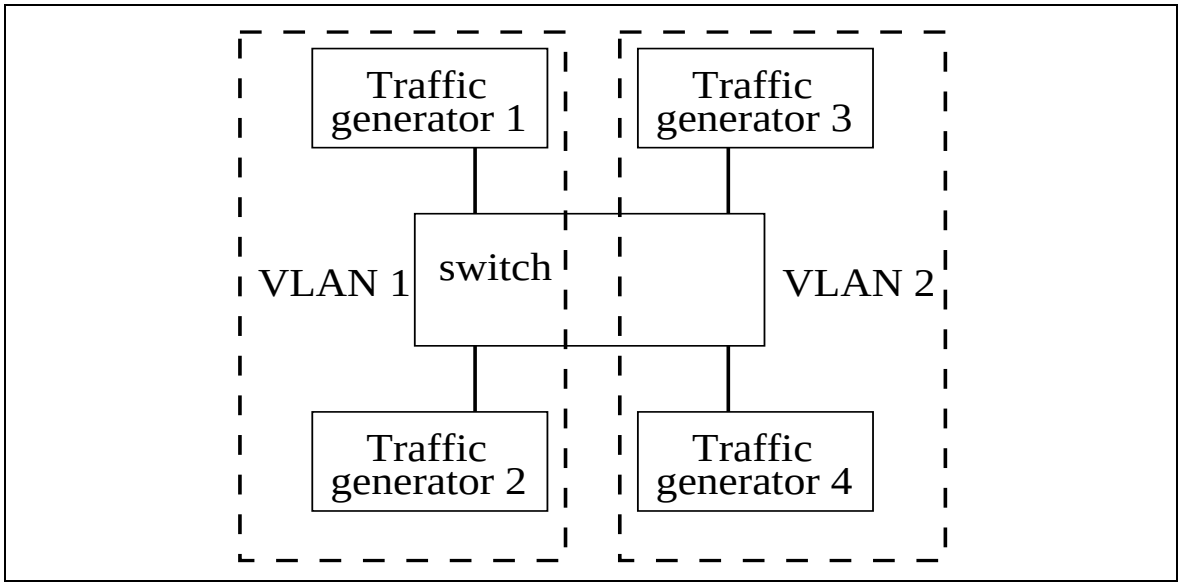

Figure 8-17 VLANs: switch partitioning setup.

8.4.1.2.9 Quality of Service (QoS)

Quality of Service support is provided through the priority field of the VLAN tag of the extended Ethernet header. Up to eight different priorities can be assigned to frames. Switches may adopt different QoS schemes such as strict priority (the highest priority is always served, and it can completely starve all the lower ones), or Weighted Round Robin (WRR) (bandwidth for all priorities is allocated according to an associated weight). Figure 8-18 presents the results of a QoS test. Eight GE ports send CBR traffic (1518 byte frames) to the same GE receiving port. Each of the senders sets a different priority. Once the capacity of the line is exceeded the strict priority algorithm gradually starves the lower priorities in favour of the higher ones. For the WRR algorithm, the bandwidth of each priority is proportional to its associated weight.

A possible use of QoS in the DataCollection is to assign control messages, e.g. the messages from the L2SV to the DFM, a higher priority than the main data flow. By applying QoS in an approximately 1/10th scale system it will be possible to assess the performance gain achievable.

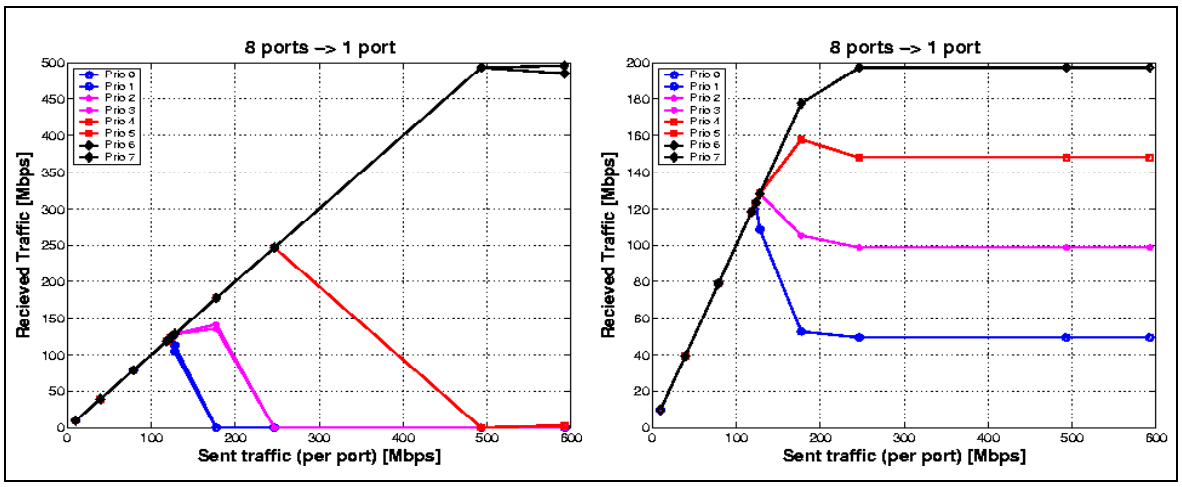

Figure 8-18 QoS measurements (a) strict priority algorithm (best-effort, normal, high, premium) (b) weighted round robin algorithm (weights of 10, 20, 30, 40).

8.4.1.2.10 Ethernet Flow Control

On a full duplex Ethernet line, a slow receiver can limit the sending rate of a fast transmitter using Ethernet Flow Control Frames (FC). The FC mechanism is presented in Figure 8-19. When the receiver becomes low in resources it sends a PAUSE frame, which will block the transmitter. Once enough resources are available on the receive side a PAUSE CANCEL frame tells the transmitter to resume sending the main data stream.

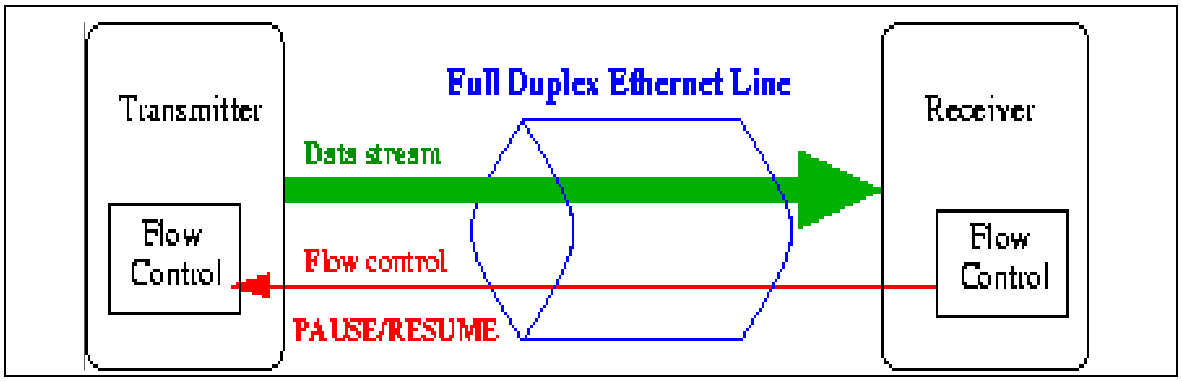

Figure 8-19 Ethernet Flow Control mechanism.

A correct FC implementation guarantees no overflow occurs at the level of the Ethernet MAC buffers. Yet we must investigate Flow Control propagation through the network switches, as well as inside the PC's (NIC, Operating System (OS) and user level application), in order to efficiently use this feature in the TDAQ system.

8.4.1.2.11 Propagation through the switches

In the setup illustrated in Figure 8-20 all the transmitters send CBR traffic at the same rate, denoted as α. X sends 100% of α entirely to A, Y splits its traffic 30% to A and 70% to B, while Z transmits 50% α to A and 50% to C. The value of α is gradually increased until line speed is achieved. When α becomes larger than 56% of the line speed the switch port from node A is oversubscribed.

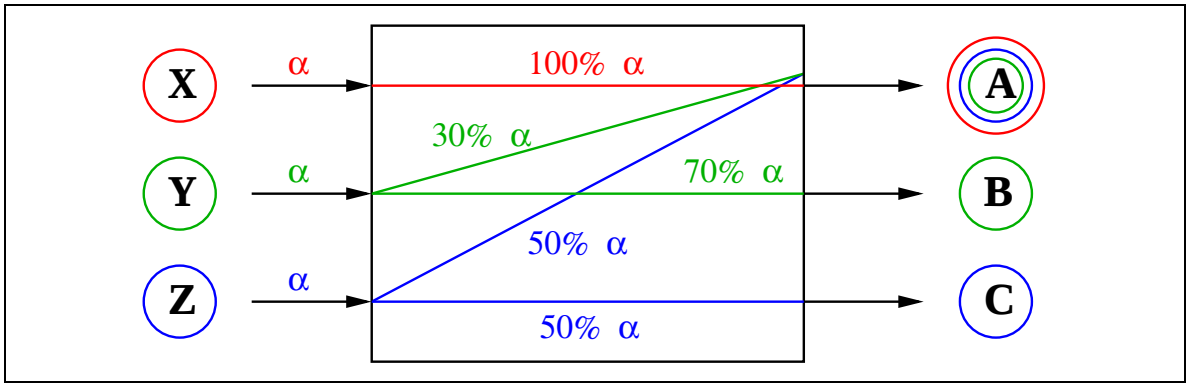

Figure 8-20 Flow control propagation testing setup.

When Flow Control is disabled on all the ports, packet loss occurs at the congested port (Figure 8-21a). It is important to see if the congested port affects the other traffic paths. Although some of the frames directed to A are lost, B receives all the traffic from Y with no packet-loss. The same observation is true for the Z transmitter. This proves the input buffers have separate queues for each of the outgoing ports, therefore the switch is not Head Of Line (HOL) blocking.

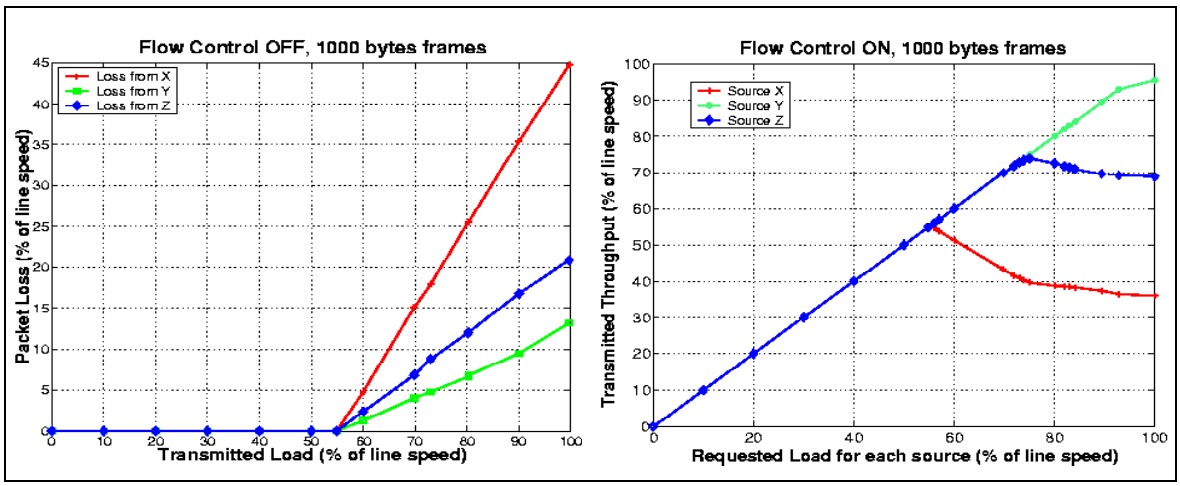

Figure 8-21 Congestion analysis with (a) flow control OFF, (b) flow control on.

When FC is enabled (both on the nodes and the switch's ports) packet loss is no longer observed at port A. This proves the switch is propagating Flow Control between it's ports. Flow control propagation across a switch is not achieved by forwarding the FC frames across a switch as they never get beyond the port's MAC, but by assuring a lossless frame transfer between the switch buffers. In achieving no packet loss the congestion spreads. The congestion from port A causes a slow down in the transmission rate of all the ports that send frames to this destination (X, Y and

Z). Thus the sending rate from Y to B, as well as from Z to C will be affected. Figure 8-21b demonstrates that the congestion effect from port A spreads towards all the ports contributing to it, proportionally with the amount of traffic sent to the oversubscribed point.

8.4.1.2.12 Fault tolerance

The spread of congestion is caused by the choice of propagating flow control through the switch, which results in undesired effects in the case of a node failure. Receiver 'A' was modified to emulate a dead node with active FC[1]. As the dead node cannot empty the received frames from the MAC queue, it will keep sending FC to the switch, as long as the latter tries to deliver packets to node A. The congestion spreads, and all the transmitters are completely blocked. No traffic reaches port B or C, as long as Y and Z keep sending towards A.

The conclusion for the baseline architecture, is that applications should realize when a node fails to respond for a long time, consider it dead, and no longer talk to it. If no such precaution is taken and FC is propagated through the switches, the requesting application will be brought to silence.

The choice of propagating Flow Control through the switches is manufacturer dependent. Some switches do not propagate FC. Packet loss occurs if the congestion cannot be absorbed by the buffers, but there is no spreading effect.

8.4.1.2.13 User level application -- traffic shaping

The deployment of a switching network based on switches that propagate FC does not guarantees lossless communication between the applications. To illustrate this a simple test was performed using the set-up shown in Figure 8-22.

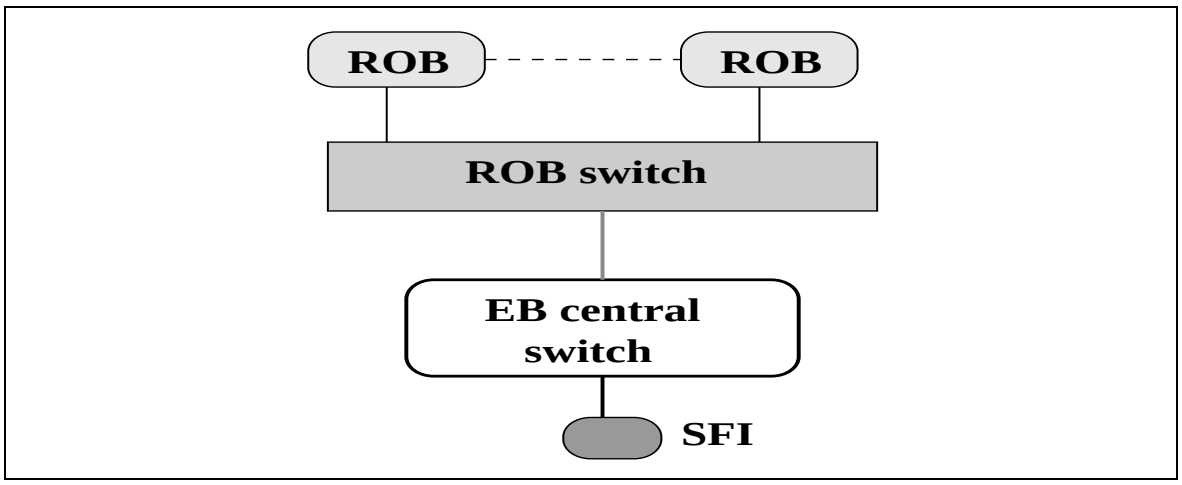

Figure 8-22 Request-response traffic Flow Control analysis.

In order to see if the PC sending rate can be reduced by Flow Control, the SFI interrogates only one RoBIn, which is artificially slowed down using a busy loop. The RoBIn cannot cope with

1. This may happen if the Linux kernel crashes but the PC's NIC remains active, or if a firmware crashes on a hardware device like the ROBs.

the request rate, asserts FC which propagates through the network, and slows down the PC's NIC. The kernel sending queues will fill up and a call to send will fail. If the error indicates no buffer space was available for sending, the sending thread should sleep for a while, then try to send again. Therefore the PC sending rate can be reduced using Flow Control.

On the receiver things are not that simple. If the PC's NIC or the kernel cannot cope with the received traffic rate, FC will be asserted by the NIC. The problem appears when the user level application cannot empty its communication socket receive buffer. If the kernel cannot push frames to the application's socket receive buffer, it will silently drop them. In other words, a lazy user level application is not allowed to slow down (or even block) other peer applications which use different communication sockets. On the other hand this kernel behaviour limits the user-level application's ability to assert Flow Control when it cannot cope with the incoming message rate.

In the request-response message flow scenarios each application can take care not to request more than it can receive: providing automatic traffic shaping. The sending part of an application is blocked if the number of outstanding request exceeds a certain threshold, and becomes active once responses arrive back.

8.4.1.3 Design of the message passing component

The requirements of the Message passing layer are detailed in [8-28]. It is responsible for the transfer of all control and event data messages between the DataFlow components. It imposes no structure on the data which is to be exchanged and it allows the transfer of up to 64 kbyte of data with a best-effort guarantee. No re-transmission or acknowledgement of data is done by this layer. This has allowed the API to be implemented over a wide rang of technologies without imposing an un-necessary 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 [8-29] defines classes that allow the sending and receiving of messages. The *Node*, *Group* and *Address* classes are used at configuration time to setup all 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 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.

8.4.1.4 Performance of the message passing

The prototype Message Passing layer interface has been implemented over raw ethernet frames, UDP and TCP. TCP provides additional reliability compared to UDP and raw ethernet. However, applications and message flow have been designed to ensure 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 doesn't need to be copied inside the application.

Extensive studies of the performance of the Message Passing layer have been performed [8-30], **Figure 8-23** shows its performance compared to direct socket measurements. Note that the latter are done in a single-threaded environment without any dynamic memory allocation and always send and receive data from a fixed location and with a fixed size that is known in advance. Raw ethernet measurements are performed with a maximum of 1460 byte, since no re-assembly of larger packets has been implemented. They therefore provide an upper bound for the possible performance.

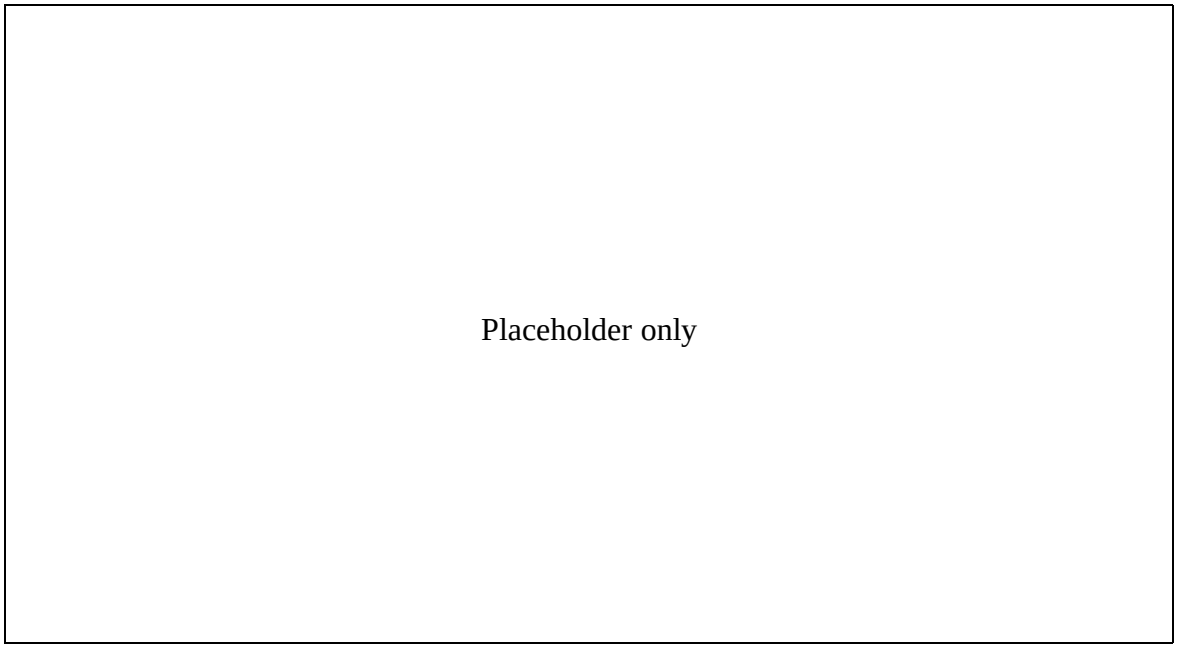

Figure 8-23 Message Passing performance compared to a direct sockets.

The measurements for request/response traffic and streaming show an overall overhead of about 6 µs compared to the direct socket measurements. The observed differences between UDP and raw Ethernet are negligible and are in the order of one microsecond.

The UDP and raw ethernet implementations use only a single socket for receiving data however, the TCP implementation must handle a large number of open sockets. By default TCP uses the 'select()' system call which is known not to scale. However, alternatives are available in the form of a POSIX conforming real-time signal in combination with non-blocking sockets or non-standard, ongoing developments, in the Linux kernel.

8.4.2 Data collection

8.4.2.1 General overview.

The requirements on the DataCollection are described in [8-31]. In summary it is responsible for the movement of event data from the ROS to the LVL2 trigger and EF and from the EF to mass

storage. It includes 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. However, DataCollection is not responsible for initializing and formatting (or preprocessing) of event fragments inside the ROS, neither is it responsible for preprocessing nor for trigger decisions in the L2PU or in the EF SubFarm.

A complete description in the DataCollection is described in [8-32]. The DataCollection implements the: L2SV, L2PUA (LVL2 Processing Unit Application, i.e. L2PU low layer functionality), DFM, pROS, SFI and SFO. In their prototype implementation a common approach to the design and implementation has been adopted. This approach leads to the definition of the common DataCollection framework, implementing a suite of common services:

- OS Abstraction Layer
- Configuration Database
- Error Reporting
- System Monitoring
- Run Control
- Message Passing

A typical application is built on top of a skeleton application and only the application specific functionality needs to be implemented.

Services are built from packages following a modular approach. Many of these packages consist only of interfaces whose implementation is provided by other packages which can be changed at configuration or run-time. This clear separation between interfaces and implementations exists down to the lowest levels like, the thread interface and access to system clocks and timers. Examples are the error reporting (switching between simple stdout/stderr and MRS), the configuration database (switching between OKS files and remote database server), the system monitoring (providing an interface to the Information Service of the Online Software and a local independent version). The Message Passing has been described in Section 8.4.1.

8.4.2.1.1 OS Abstraction Layer

The OS abstraction layer consists of packages hiding all OS specific interfaces. E.g. the *threads* package hides the details of the underlying POSIX thread interface.

8.4.2.1.2 Error Reporting

The ErrorReporting package allows the logging of error messages either to standard out and error or to MRS. Each package can define its own set of error messages and error codes. Error logging can be enabled/disabled on a package by package basis, with a separate debug and error level for each package. Furthermore debug logs and normal error logs are treated logically differently, so the debug message could go to stderr while all normal application logs go to MRS. The user only interfaces via a set of macros to the ErrorReporting system allowing optimization of the applications at compile time.

8.4.2.1.3 Configuration Database

All applications make use of the Online Software's configuration database and their design allows the underlying implementation to change without implying changes to the application. The application's view of the database is hidden by configuration objects which access the database, providing a more convenient way to access configuration information. The configuration objects themselves are created automatically from the Configuration Database schema file.

8.4.2.1.4 System Monitoring

This package allows every component to make arbitrary information available to some outside client. In practice this is used to publish statistics like counters and histograms. The packages makes this information available in various different ways, including the Information Service of the Online Software.

8.4.2.1.5 Run Control

The run control interface is responsible for translating the requests from the Online Software about state changes into commands for the application. It also provides a skeleton around which one can build an application. These classes realize most of the use cases for run control. They talk to a special DataCollection Run Controller on the one side and application specific code on the other side.

8.4.2.2 RoI data collection

8.4.2.2.1 Design

The interaction between the L2SVs, L2PUs, ROSs and pROS which results in the collection of RoI data leading to a LVL2 Decision with further details in the LVL2 Result is explained in Chapter 9.

8.4.2.2.2 Performance

Each L2SV controls a subfarm of L2PUs. The maximum size of a subfarm is determined by the rate at which the L2SV can handle each L2PU. This is shown in Figure 8-24 using a L2SV with an emulated connection to the RoI Builder. The maximum rate for a farm containing a single L2PU is ~ 30 kHz dropping off slowly as more L2PUs are added. Thus a few L2SVs are sufficient to achieve the maximum design rate of 75 kHz.

The maximum rate at which an L2PU can collect RoI data depends on the size of the RoI, the number of ROSs that contribute data and the number of Worker threads that collect RoI data in parallel on the same L2PU. Figure 8-25 shows 1/Rate for an RoI of 16 kbyte collected as 1, 2, 4, 8, 16 or 22 slices of 16, 8, 4, 2, 1 or 0.8 kbyte respectively, varying the number of Worker threads between 1, 2, 4 or 8. The L2PU as well as the L2SV and ROS emulators were all dual Xeon CPUs of 2.2 GHz interconnected by Gbit Ethernet. For this test, the L2PUs were completely dedicated to data collection. The plot shows that the time for acquiring RoI data is small compared to the execution time of selection software (currently aimed at 10 ms/event average).

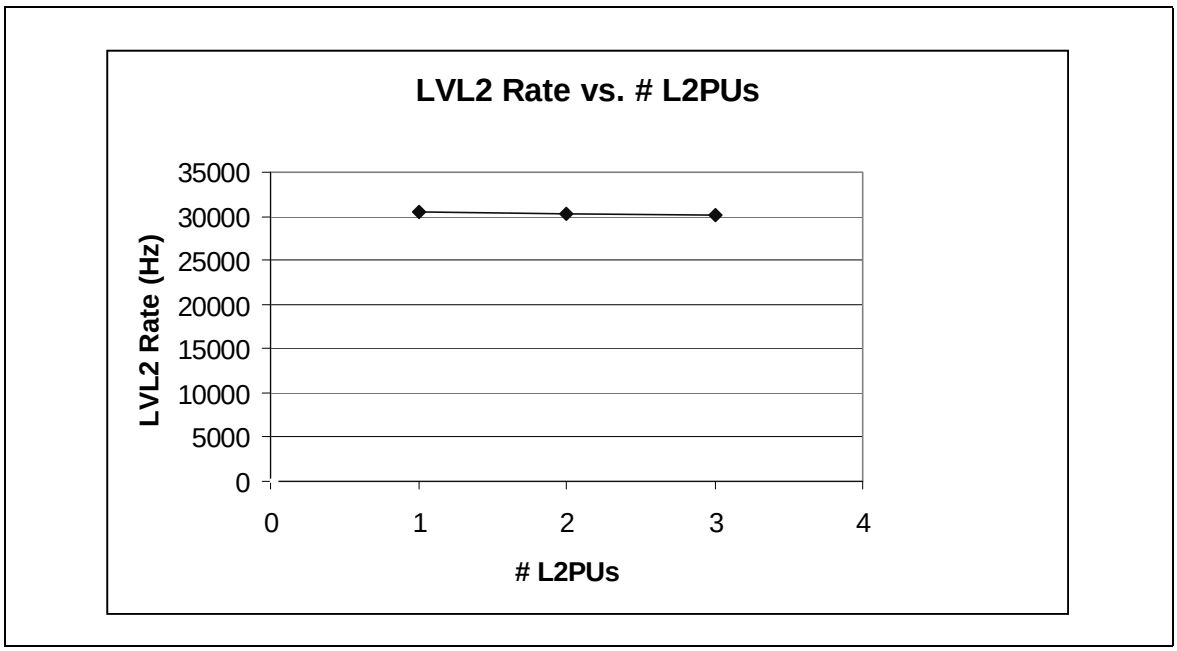

Figure 8-24 Maximum LVL2SV decision rate as a function of the number of L2PUs it controls.

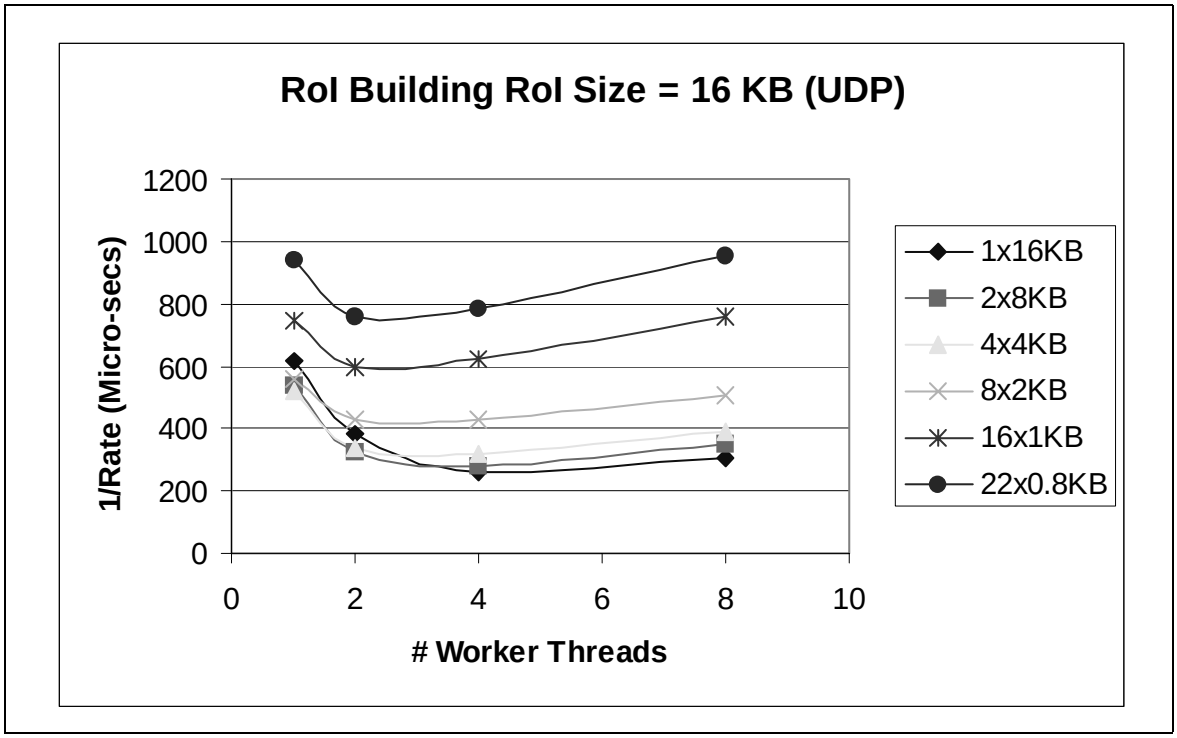

Figure 8-25 Performance of the RoI data collection for an RoI of 16kbyte as a function of the number of Worker Threads. The plot shows 1/Rate collecting data in slices of equal size from 1, 2, 4, 8 or 16 sources.

Figure 8-26 summarizes the performance of the RoI data collection for various combinations of RoI sizes and slices for four threads.

The scalability of the RoI data collection has been tested by using two L2SVs, 22 ROS emulators and varying the number of L2PUs from 1 to 8. All nodes were PCs equipped with dual Xeon processors at 2 or 2.2 GHz connected by Gigabit Ethernet. Figure 8-27 shows the obtained RoI

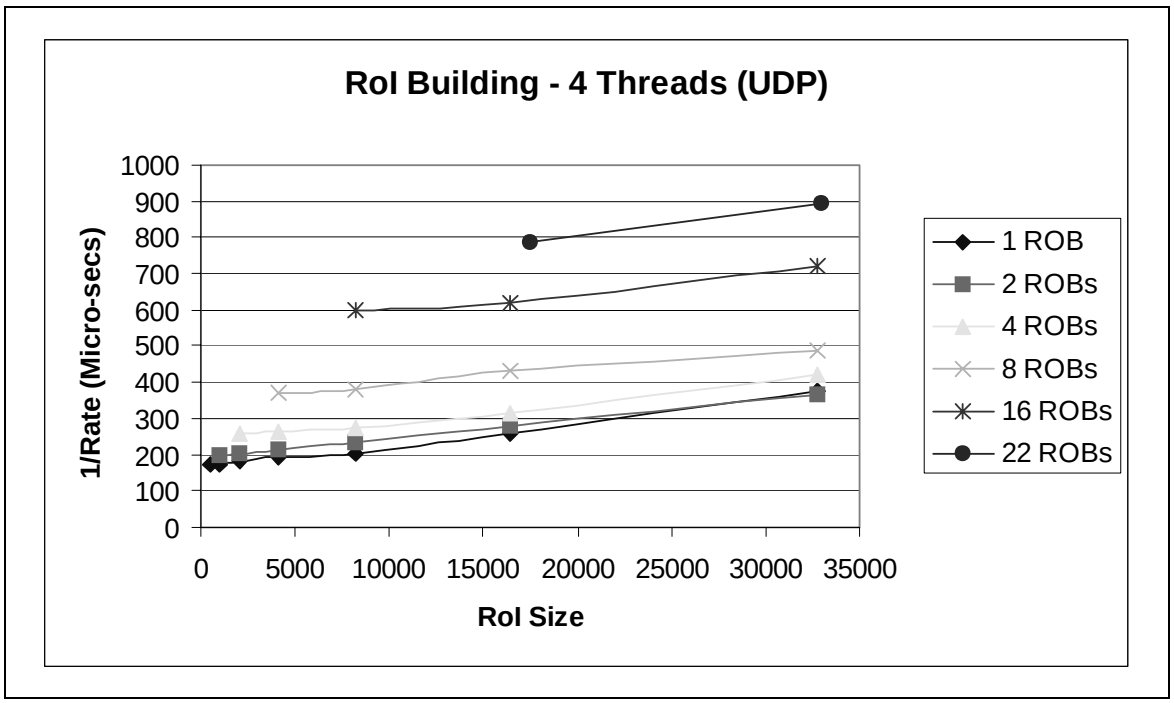

Figure 8-26 Summary of the performance of the RoI data collection for various combinations of RoI sizes and slices.

rate of the system for 1, 2, 4 or 8 L2PUs collecting RoIs as slices of 6 x 1 kbyte, 3 x 2 kbyte (6 kbyte RoI) and 6 x 4 kbyte or 12 x 2 kbyte (24 kbyte RoI). The unrealistically small number of ROS emulators available for the test causes a deviation from perfect scaling for 8 L2PUs.

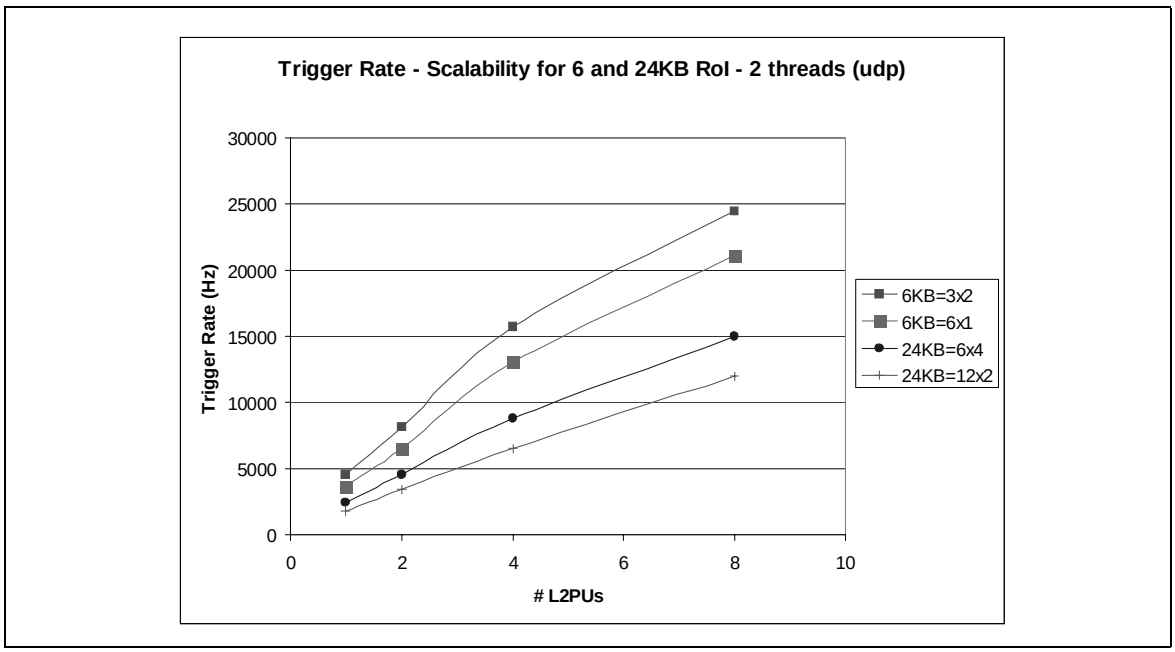

Figure 8-27 Scalability of the data collection is demonstrated by the rate at which as system of 1,2,4, or 8 L2PUs collect RoI data from 22 ROS emulators. The unrealistically high L2PU/ROS ratio causes a deviation from perfect scaling for 8 L2PUs.

8.4.2.3 Event Building

8.4.2.3.1 Design

The interaction between the ROSs, the DFM and the SFIs which implements the event building functionality for is shown in Figure 8-14 and explained in [8-25]. Two scenarios concerning the interaction between DataFlow components in Event Building are being studied:

- PUSH scenario: In this scenario, the DFM assigns an SFI to all the ROSs via a multicast mechanism. The ROSs then respond to the assigned SFI with their respective ROS event fragment. 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. The SFI then requests from each ROS its event fragment via a series of unicast messages. The SFI receives from each ROS individually 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 double in case of the pull scenario.

Although a doubling of the message rate at the level of the SFIs may seem problematic, the pull scenario offers the advantages with respect to controlling the flow of traffic. In this mode an SFI at any given moment in time never requests more fragments than it can handle. Thus it smooths out the traffic and reduces the risk for congestion within the network. In the case of the PUSH scenario, the ROSs will need to control the amount of traffic sent to each SFI individually; this can be achieved via applying QoS at the level of the ROS. Detailed studies have been made on the use of IP QoS to avoid congestion in the network. The results of these studies are summarised in Section 8.4.2.3.3 and further details can be found in [8-33]

8.4.2.3.2 Performance

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 1628 different buffers. This has to be performed at a rate of ~ 3 kHz. Detailed studies of the event building have been performed [8-34], [8-35] using prototype software, PCs, Ethernet switches and traffic generators, see Section 8.4.1.2.1for description of the traffic generators only the principle 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. Seeded by this rate, in the pull scenario, the DFM assigns the events to and SFI, receives notification when an event is built and sends clears to the ROSs. The performance of the DFM, defined as the sustained event building rate verses the number of SFIs is shown in Figure 8-28. It can be seen that the prototype implementation of the DFM can sustain an event building rate of up to ~ 23 kHz, an order of magnitude greater than the required performance. Within 5% this sustainable rate is independent of the number of SFIs deployed in the system.

In Figure 8-29 the sustained event building rate per SFI is shown as a function of the number of ROLs per ROS. The two curves represent the cases where the SFI forwards the built event to an EF subfarm or not. The results shown in the figure indicate that today's prototype implementation of the event building functionality deployed on PCs (dual processors clocked at 2.4 GHz)

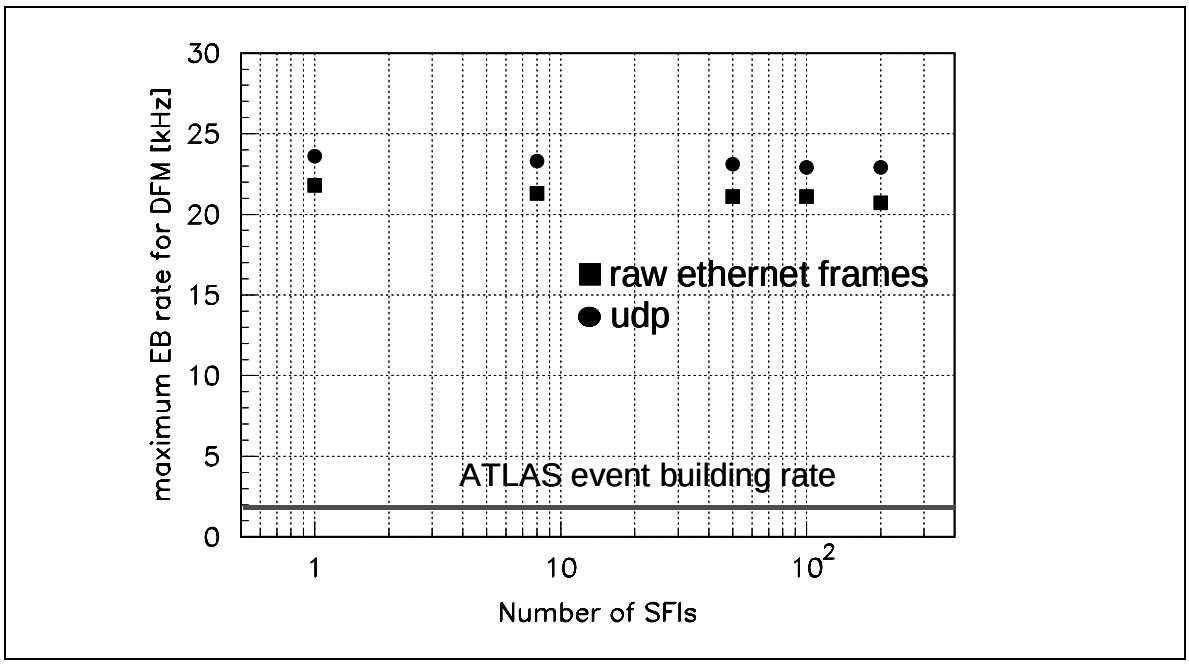

Figure 8-28 The DFM event building rate verses the number of SFIs. Each SFI concurrently builds two events

achieves an event building rate per SFI 35 Hz. In addition, the bus-based scenario gives a performance gain of ~ 30%.

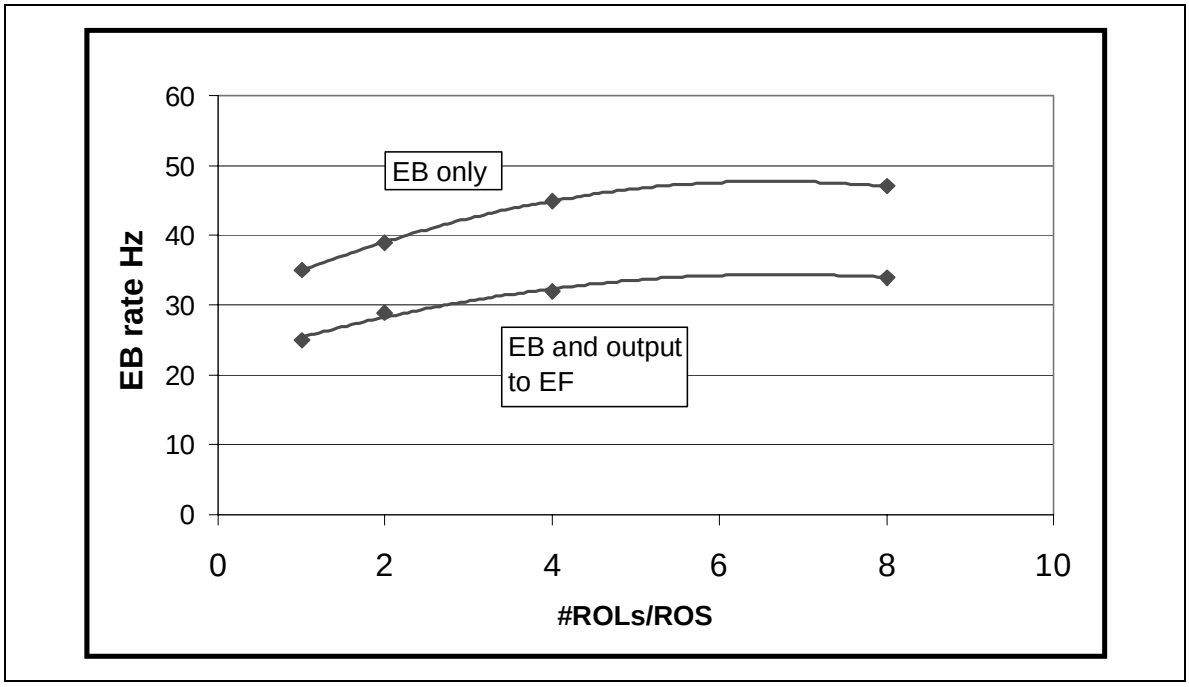

Figure 8-29 The event building rate verses the number of ROLs/ROS in a system with a single SFI.

The scalability of the event building is shown in Figure 8-30. In this test the number of SFIs in the set up was increased from one to eight and the corresponding event building rate measure. It can be seen that the sustained event building rate increases linearly with respect to the number of SFIs in the system and that every additional SFI contributes to the overall system performance by ~ 35 Hz.

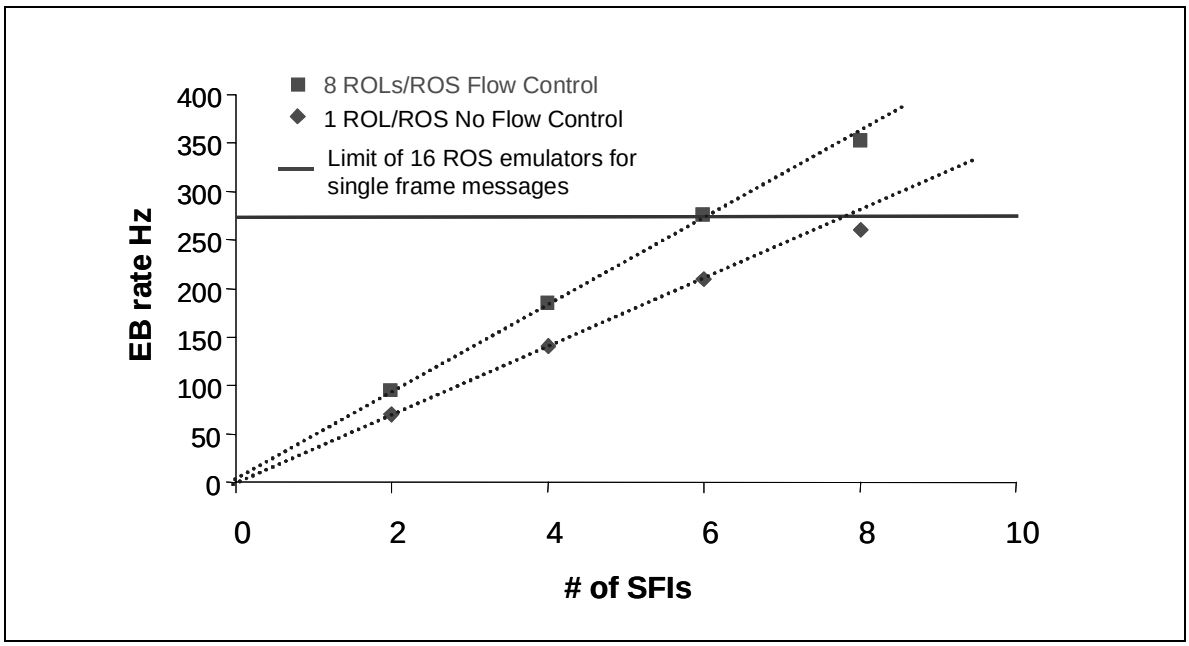

Figure 8-30 The event building rate verses the number of SFIsin the system.

It should be noted that the results in Figure 8-30 for eight ROLs / ROS were achieved with ethernet flow control active. The measurements with ethernet flow control disabled have yet to be understood.

8.4.2.3.3 Event Building with QoS

Quality of Service has been implemented in the standard Linux kernel at the IP level and it can be used to shape the traffic entering a switching network. This removes the necessity of implementing traffic shaping at the level of the applications.

QoS manages the flow of data at the IP level by employing packet classification, packet scheduling and traffic shaping techniques. Packet classification is used to classify incoming packets in groups, such as Class Base Queuing (CBQ). The packet scheduler arranges the scheduling for outgoing packets according to the queuing method and the buffer management selected. Token Bucket Filter (TBF) is an example of one method. The outgoing packet are sent at a rate determined by the size of the token buffer and the rate in which tokens are supplied. The traffic shaper is a technology to make the burst flat.

Note that QoS as implemented by the Linux kernel is performed only in the message output queues, i.e at the level of the ROSs, in coming 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. Event building is to be performed at a rate of ~ 3 kHz, therefore the data should be scheduled to at least the same rate for the traffic shaping to be effective. In the studies performed the Linux kernel scheduling frequency was set to ~ 4 kHz.

The event building performance with QoS applied at the IP level has been measured for the push scenario [8-33]. The results of these studies are shown in Figure 8-31.

In this figure 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 lose 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 is more preferment 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 is more preferment.

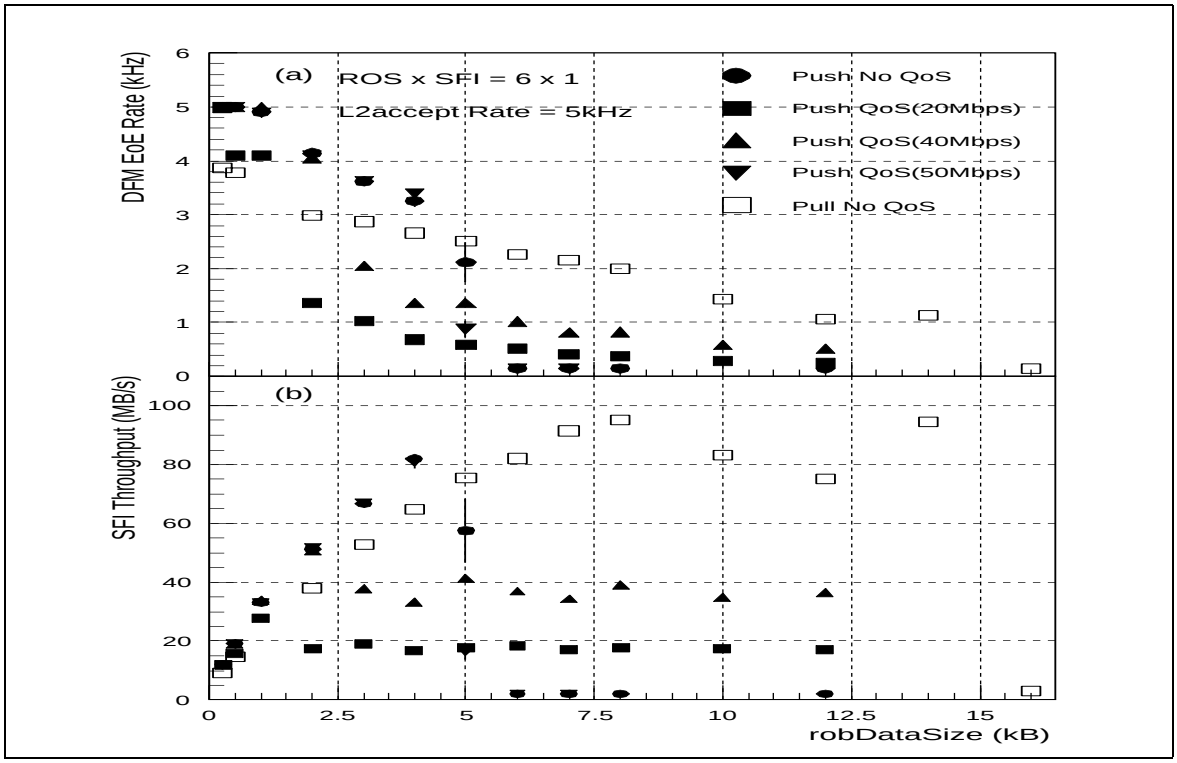

Figure 8-31 (a) Event Building rate and (b) throughput as a function of event fragment size and different QoS parameter values for the push scenario.

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

8.5 Scalability

8.5.1 Detector readout channels

This section describes quantitatively how the physical size, performance and control and configuration of the system scales with the 'amount' of detector to be read out.

8.5.1.1 Control and flow of event data

How the number of applications, messages and data volume changes.

8.5.1.2 Configuration and control

Amount of configuration data a function of the amount of detector.

8.5.2 LVL1 rate

How the system performance and physical size scales with respect to the LVL1 rate.

8.6 References

8-1 *Trigger & DAQ Interfaces with Front-End Systems: Requirement Document*, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/DIG/archive/document/FEdoc_2.5.pdf

8-2 *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

8-3 The S-LINK *interface Specification*, http://edmsorweb.cern.ch:8001/ceder/doc.info?documnet_id=110828

8-4 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

8-5 Recommendations of the Detector Interface Group - ROD Working Group, https://edms.cern.ch/document/332389/1

8-6 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).

8-7 Design specification for HOLA, https://edms.cern.ch/document/330901/1

8-8 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

8-9 B. Gorini, *ROS Software Architecture Document*, https://edms.cern.ch/document/364343/1.3

8-10 ReadOut Subsystem, *Summary of prototype RoBIns*, https://edms.cern.ch/document/382933/1

8-11 Readout sub-system test report (using DAQ -1.), *in preparation*.

8-12 R.Cranfield et. al., *ROS Requirements*, https://edms.cern.ch/document/356336/1.0.0

8-13 B. Green et. al., *RobIn TDR-Prototype HLDD*, https://edms.cern.ch/document/356324/2.4

8-14 B. Green et. al., *RobIn TDR-Prototype DLDD*, https://edms.cern.ch/document/356328/2.3

8-15 B. Green et. al., *RobIn TDR-Prototype Software Interface*, https://edms.cern.ch/document/356332/2.2

8-16 B. Gorini et. al., ROS Test Report, *in preparation*.

8-17 The MPRACE board, http://mp-pc53.informatik.uni-mannheim.de/fpga/

8-18 P. Werner amd A. Bogaerts, *Pseudo-ROS Requirements*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-034.pdf

8-19 P. Werner, *Pseudo-ROS Design Document*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-041.pdf

8-20 ROD Crate DAQ Task Force, *Data Acquisition for the ATLAS ReadOut Driver Crate (ROD Crate DAQ)*, https://edms.cern.ch/document/344713/1

8-21 R. Spiwoks, *Workplan for ROD Crate DAQ*, https://edms.cern.ch/document/364357/1

8-22 M.Abolins et. al., *Specification of the LVL1 / LVL2 trigger interface*, https://edms.cern.ch/document/107485/1

8-23 R. Blair et. al., 1. *A Prototype RoI Builder for the Second Level Trigger of ATLAS Implemented in FPGA's*, ATL-DAQ-99-016

8-24 B.M. Barnett et. al., *Dataflow integration of the calorimeter trigger CPROD with the RoI Builder and the ROS*, https://edms.cern.ch/document/326682/1

8-25 H-P. Beck and C. Haeberli, *Mesage Flow: High-Level Description*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-012.pdf

8-26 H-P. Beck and F. Wickens, *Mesage Format*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-022.pdf

8-27 ALTEON traffic generators, *in preparation*.

8-28 R. Hauser and H-P. Beck, *Requirements for the Message Passing Layer*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-008.pdf

8-29 R. Hauser, *Message Passing Interface in the LVL2 Reference Software*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-007.pdf

8-30 P. Golonka, *Linux network performance study for the ATLAS DataFlow System*, https://edms.cern.ch/document/368844/0.50

8-31 R. Hauser and H-P. Beck, *ATLAS TDAQ DataCollection Requirements*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-045.pdf

8-32 R. Hauser and H-P. Beck, ATLAS TDAQ/DCS DataCollection Architectural Design, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-012/DC-043.pdf

8-33 Y. Yasu et. al., *Summary of Studies on ATLAS DataCollection software with QoS*, https://edms.cern.ch/document/382921/1

8-34 M. Gruwe & B. Di Girolamo, *DataFlow performances measurements: the Event Building*, https://edms.cern.ch/document/375101/1

8-35 C. Haeberli et. al., *ATLAS DataCollection: Event Building Test Report*, https://edms.cern.ch/document/382415/1

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. 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. This common architecture is described in Section 9.4.

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 considered 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).

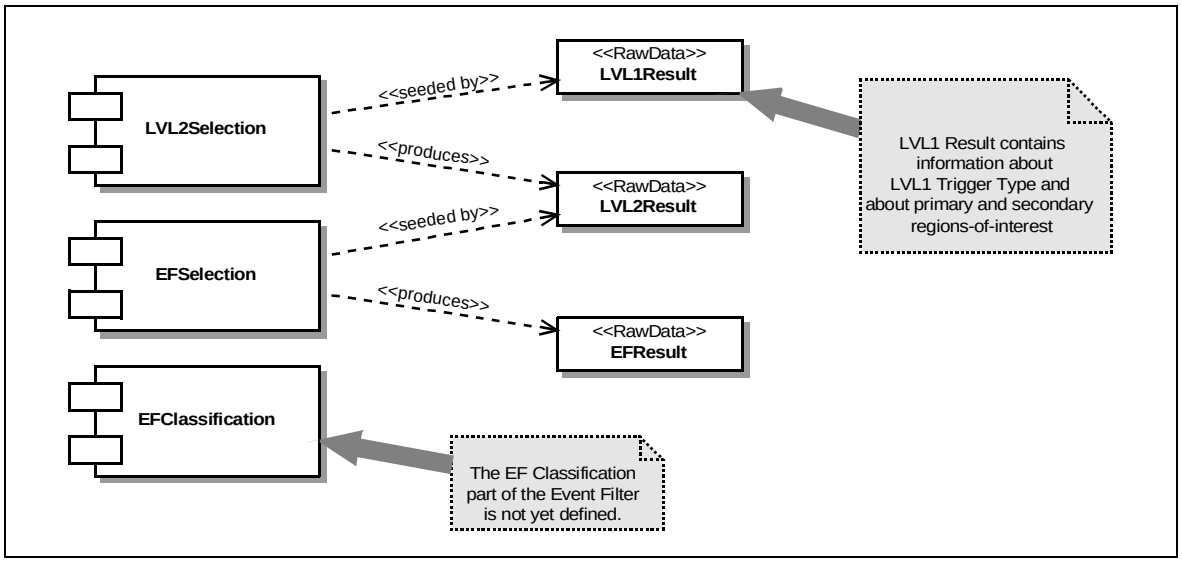

Figure 9-1 The high level trigger selection chain with the LVL2 and EF selection each seeded by the preceding trigger.

The LVL2 Result plays a similar role for the EF as does the LVL1 Result for the LVL2. The LVL2 Result provides the means to seed the EF selection. 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 which were satisfied and higher level reconstruction objects, will be appended to the raw event data.

The EF classification is yet to be fully defined. 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 ReadOut Buffers. In parallel, information on the location of RoIs identified by LVL1 is sent to LVL2 to guide the LVL2 event selection. Using this guidance specialised LVL2 algorithms request a sub-set of the event data to perform the second stage of event selection. In this way only a few percent of the event data need to be transferred to the LVL2 system — thus considerably reducing the network bandwidth required. Events selected by LVL2 are passed to the Event Builder, where the complete event is assembled into a single record. The built event is then passed to the Event Filter where the third and final stage of on-line event selection is performed. The Event Filter applies off-line algorithms guided by information from the LVL1 and LVL2 triggers to further refine the event selection. Events passing the Event Filter are then passed to storage for off-line analysis. Figure 9-2 shows the exchange of messages between the subsystems involved in the HLT proc-

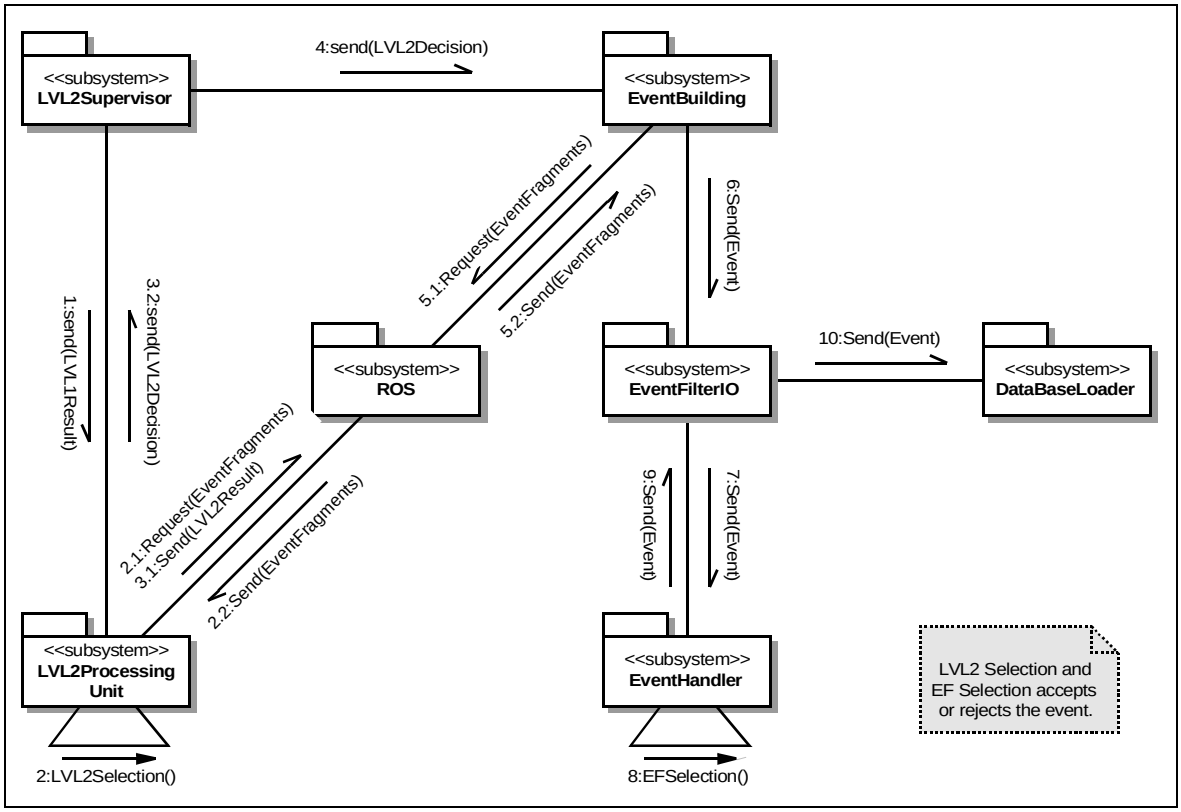

Figure 9-2 The exchange of messages between HLT Components.

ess. The LVL2 selection is done in the LVL2 processing unit and the EF selection in the event handler. Both LVL2 and EF are situated in dedicated processor farms. The LVL2 Processor receives the LVL1 result from the LVL2 supervisor. LVL2 is RoI guided and only requests the corresponding fragments of the events from the ROSs. After a positive LVL2 decision the event building collects all fragments, including the LVL2 Result. The full event is sent via the event filter IO to the event handler, where the EF selection is made. Accepted events are sent to the database loader for permanent storage of the event for offline reconstruction and analysis.

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, the LVL2 Supervisor, the LVL2 Processors, the ROS and the pROS. The RoI Builder assembles the fragments of information from the different parts of the LVL1 trigger and transmits the combined record to a LVL2 Supervisor. The LVL2 Supervisor selects a LVL2 Processor for the event and sends the LVL1 information to that processor and then waits for the LVL2 Decision to be returned. The LVL2 Processor runs the Selection code, requesting event data as required from the ROSs and returns the LVL2 Decision to the LVL2 Supervisor. For events which are to be passed to the Event Filter the LVL2 Processor 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 RoI Builder combines these fragments into a single record which is passed to a LVL2 Supervisor processor. In the baseline design each LVL2 Supervisor will only see a sub-set of the LVL2 Processors, thus the choice of LVL2 Supervisor affects the load-balancing between LVL2 Processors. This routing would normally be on a round-robin basis, but busy Supervisors can be skipped and the design also allows more complex algorithms, including use of the LVL1 trigger type. A fuller description of the RoI Builder is given in Section 8.3.2.

9.2.3 LVL2 Supervisor

The LVL2 Supervisors are a small group of processors (of order 10) that 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 S-LINK interface to receive the data from the RoI Builder (see Section 8.3.2).

The context of the Supervisor is indicated in Figure 9-3. The Supervisor receives information on the LVL1 trigger in a single record (LVL1Data) from the RoI Builder. It selects a LVL2 processor for the event and passes the LVL1 data to the processor in the LVL1Result message. Once the LVL2 processor has decided whether the event should be accepted or rejected it passes back a LVL2Decision message. If the decision is an accept or if the Supervisor has collected a predetermined number of rejects, the LVL2Decision Group message is sent to the DFM where the DFM coordinates clearing of the buffers and readout to the EB.

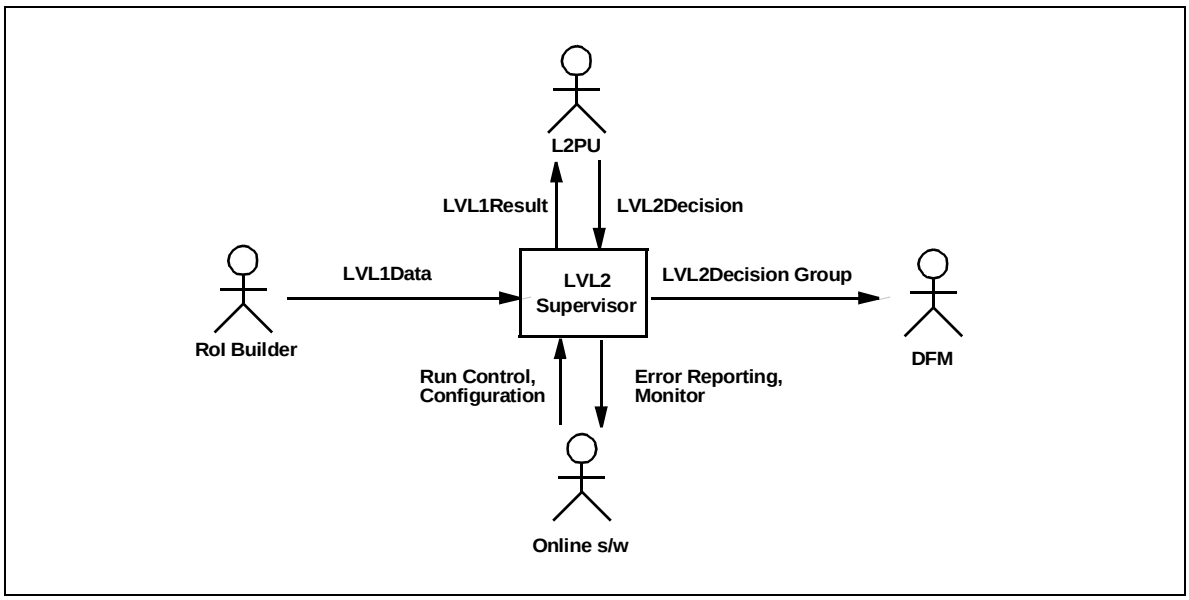

Figure 9-3 Context of LVL2 Supervisor.

In selecting a LVL2 processor the Supervisor exercises a load balancing function. Currently the Supervisor is designed to either select processors from its available pool via a simple round-robin algorithm or by examining the number of events queued and assigning the event to the least loaded processor. It is foreseen to automatically accept certain classes of LVL1 triggers without involving the LVL2 processors, for example detector calibration events. This could be extended

to provide unbiased samples to the EF.

The supervisor software is based on the DataCollection Framework described in Section 8.4.2. For normal data taking the LVL1Data is received from the RoI Builder, but for diagnostic purposes the Supervisor can also retrieve LVL1 data from a file or generate it internally.

9.2.4 LVL2 Processors

The LVL2 selection is performed in farm of processors, today assumed to be dual processor PCs, running at 4 GHz, in 1U rack-mounted format. The budget allowed for the mean latency 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 utilisation 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. The application has three main components: the L2PU; the PESA Steering Controller (PSC); and the Event Selection Software. The L2PU handles the DataFlow 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 LVL2 Supervisor 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 [9-2] has been studied for some CPU-intensive algorithms, for example non-guided track finding in the inner detector[9-2]. For the current trigger menus, however, it has been found that these are not justified and therefore they are not included in the baseline architecture.

9.2.4.1 L2PU

The design and implementation of the L2PU is based on the DataCollection Framework described in Section 8.4.2 from which it uses the following services: application control, initialisation and configuration, error reporting, application monitoring, message passing, and, for the purpose of performance evaluation, the instrumentation.

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) is requested from the ROSs, on instigation of the LVL2 Selection algorithms. For positive decisions, a LVL2 Result is sent to the pROS.

The actual selection algorithms runs 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 [9-12]. 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.

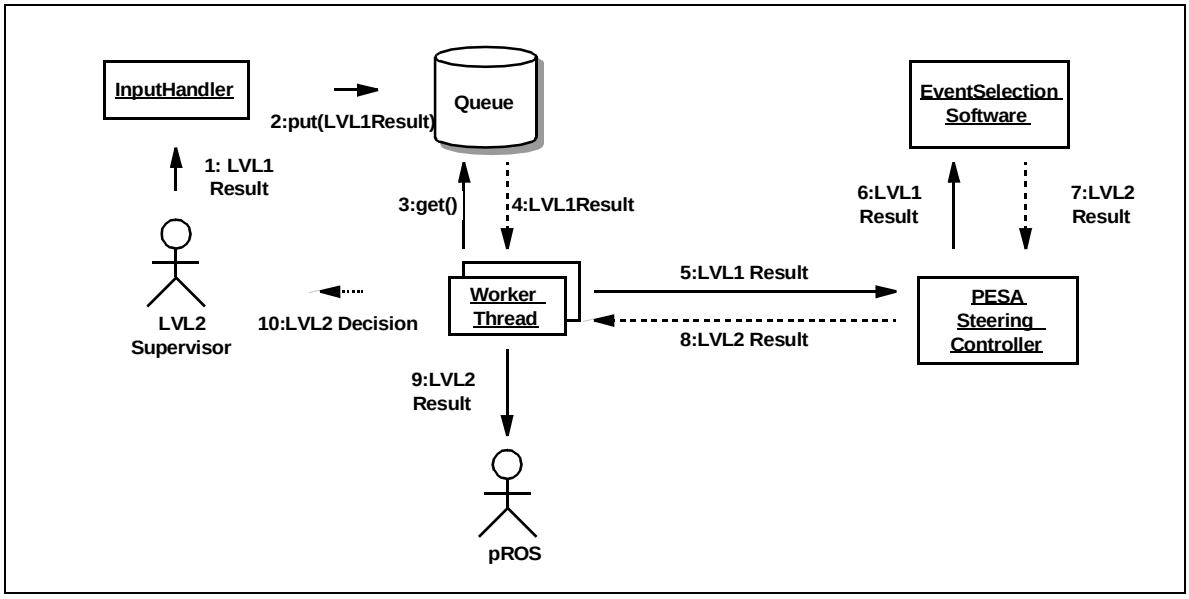

Figure 9-4 Collaboration diagram showing the successive steps that are applied to each event received from LVL1 leading to the LVL2 Decision.

Selection algorithms that need data from ROBs activate the ‘RobDataCollector’, which functions as shown in Figure 9-5. The DataCollector takes a ‘list of RoBs’ as input parameter and returns a ‘list of ROB Fragments’. The DataCollector 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 which are returned to the caller.

As reported in Chapter 8 the performance for collecting data for RoIs 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 LVL2 Event Selection Software. 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 with the Event Filter; and to provide a mechanism for transmitting the LVL1 and LVL2 Results between the dataflow system and the PESA software.

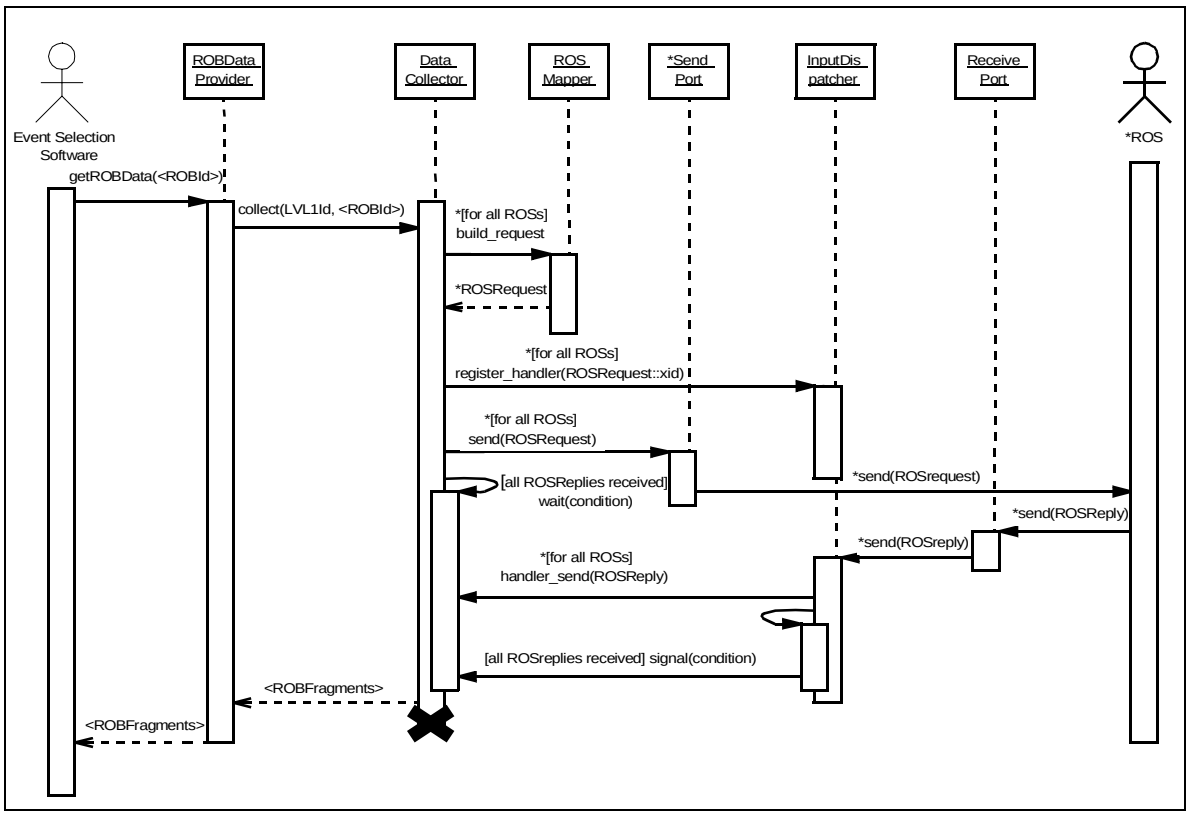

Figure 9-5 Data Collection by the L2PU of the data within a list of ROBs, corresponding to an RoI.

The key to the PSC design is to place this interface where the functionality of the dataflow and event selection frameworks 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 state changes from the dataflow software to the PESA software. Since the HLT Event Selection Software is being developed in the offline framework, ATHENA [9-13], which is itself based on Gaudi [9-14], the PSC has been designed [9-15] to re-use the framework interfaces defined in Gaudi.

Figure 9-6 illustrates the sequence of interactions of the PSC with the dataflow and the PESA software. The figure shows three states: *Configure*, *Start*, and *Stop*. During the *Configure* phase, configuration and conditions metadata are obtained from external databases via an HLT-online interface. These data are then used to configure the PESA software 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 PESA selection software on the event) the LVL2 Result directly to the Data Collection framework.

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 LVL2 supervisor, or with the pROS. The requests for event data fragments are hidden behind the DataManager.

After a *Stop*, the PSC terminates algorithm execution. At this stage, run summary information can be produced for the selection process.

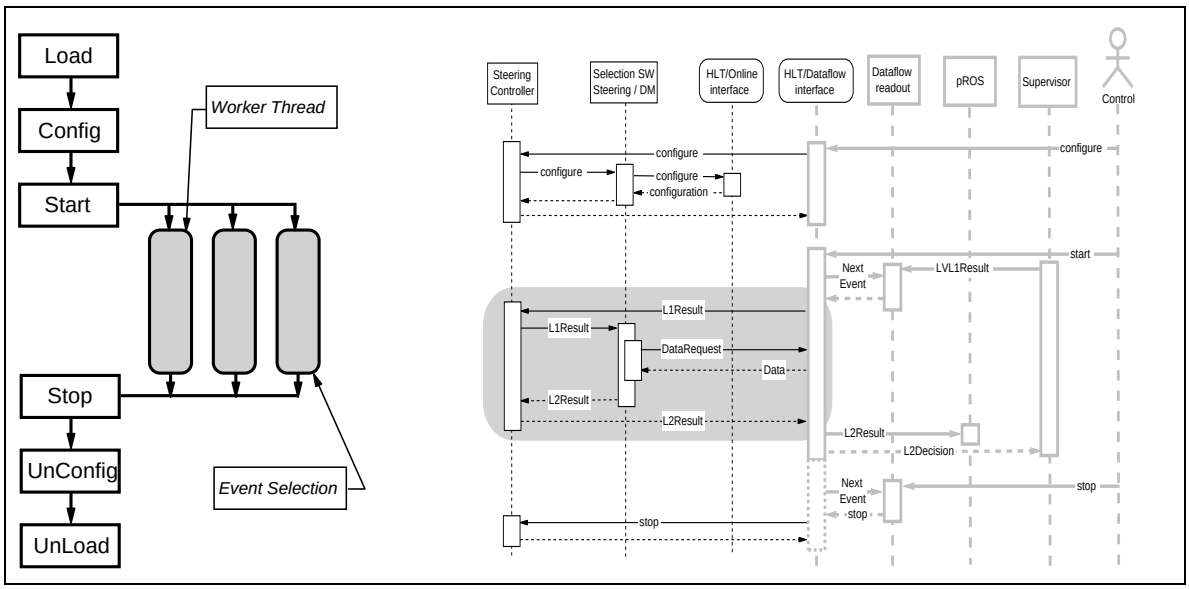

Figure 9-6 The L2PU Finite State Machine (left) and the PESA Steering Controller (right).

Since the Event Selection Software 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 DataCollection software, both performance and robustness tests were carried out on a dual-processor 1.533 GHz Athlon machine (for details, see [9-15]). The PSC ran for over 50 hours with three threads with an early version of the selection software prototype [9-16]. The prototype ran successfully on both single-CPU and double-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 Event Selection Software running in the LVL2 Processing Unit. The communication with external systems and subsystems, including the LVL2 Supervisor and the ROS, are hidden from the ESS. The ESS is initialised 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.

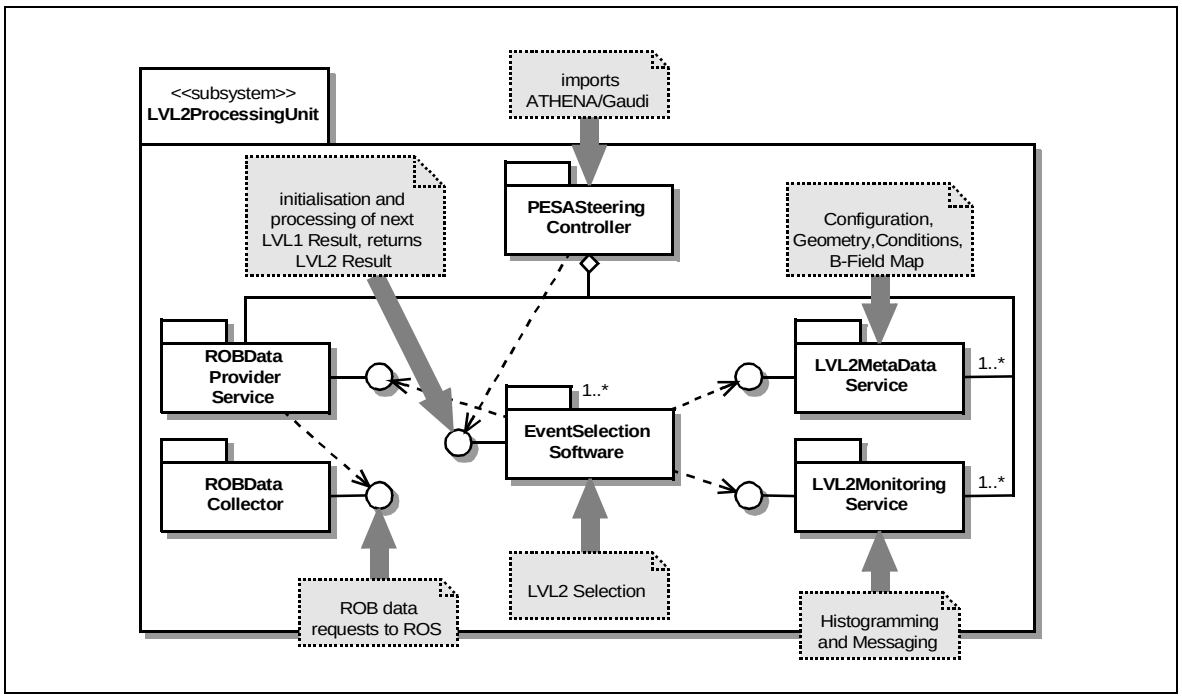

Figure 9-7 The dependencies of the Event Selection Software performing the LVL2 selection in the LVL2 Processing Unit.

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 (~ 2 kHz).

9.2.6 LVL2 Operation

The configuration and control of the LVL2 applications in the LVL2 processors are handled by standard DataCollection controllers, using the toolkit provided by the Online Software (see Chapter 10). 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 (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. The overall supervision of the LVL2 is integrated in the HLT Supervision system described in Chapter 12.

Here a brief description of how the farm-fabric is managed.

9.3 Event Filter

The Event Filter (EF) is the third and last step of the on-line selection chain. It makes use of the full event information. It will use the offline framework (ATHENA) to execute filtering algorithms which will be taken directly from the offline suite.

9.3.1 Overview

9.3.1.1 Functionality

The functionality of the EF has been logically distributed between two main entities:

- the Event Handler (EH) is in charge of performing the activities related to event selection. This includes the data flow between the main DAQ system and the EF as well as between the different steps of the selection itself. It also includes the framework to run the processing tasks (PT).
- the EF Supervisor is in charge of the control operations, in co-ordination with the overall TDAQ control system. Its responsibilities include the monitoring of the EF functionality.

The EF has been designed so that additional functionalities, not yet formally EF responsibilities, can be added without jeopardising the selection activity. Examples of such extra activities are the global monitoring of the detectors or tasks related to alignment and calibration.

9.3.1.2 Operational analysis

The operational analysis of the whole HLT has been described in a dedicated document [9-4] which describes in details the expected functionality and gives uses cases for the operation.

In the case of the EF, use cases are separated between the following operations:

- the *start-up operations*, which tentatively gives the list of the different operations which should be necessary to set the HLT processing farms in operation.
- the *Run Control* related operations, which illustrates how the EF will cope with the *Start of Run*, *End of Run*, *Pause*, *Resume* and *Checkpoint* commands.
- the *shutdown* operations which tentatively gives the list of the different operations which should be necessary to set the HLT processing farms back into the non-operating state.
- the *steady operation actions*, which are the list of the operations which do not stop the data taking process, but are rather associated with the *checkpoint* procedure. This includes changing the trigger menu during a given spill as well as modifying the computing power of the EF (by adding or removing CPUs in the farm).
- *unsolicited* events: this section is a first approach to error management in the EF. More information on error handling in EF can be found in Chapter 6??.

9.3.1.3 Baseline architecture

The baseline architecture of the Event Filter relies on commodity components, namely PCs linked by Ethernet networks. A detailed description of the architecture may be found in [9-5].

The EF Farm is organised in independent sub-farms, each one connected to a different SFI. Several SFOs can be connected to a given sub-farm. A given SFO may be accessed by different sub-farms. A possible layout is shown on Figure 9-8. Backend switches are Gigabit Ethernet, while the sub-farm switches are Fast Ethernet.

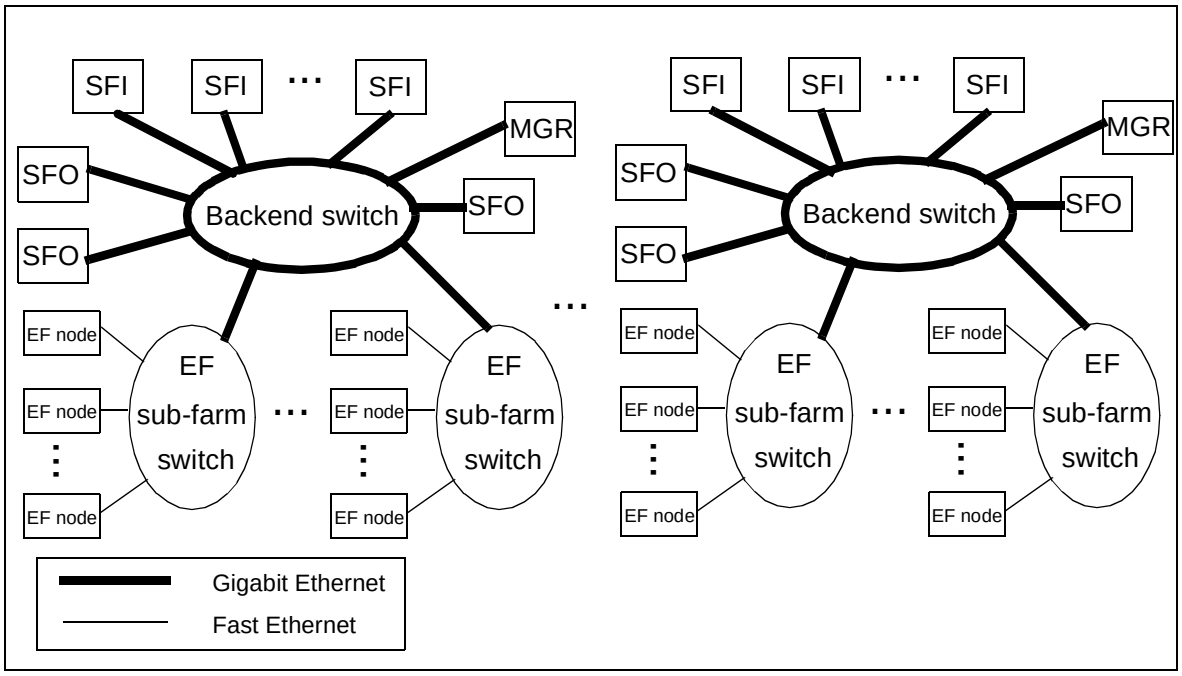

Figure 9-8 Possible layout for the EF Farm.

9.3.2 Event Handler

9.3.2.1 Requirements

A detailed list of requirements can be found in [9-6]. This list is based on the analysis of constraints coming from other systems and on some primary and general uses cases. A summary of these requirements is given here.

- The EH shall receive events from the main DataFlow system and send them back to it for permanent storage. The EH shall be capable of appending information to the event containing details related to the selection operations performed. Events may be directed towards dedicated output channels according to the results of the processing performed (specialised channels according to event classification, monitoring channels, channels for events which have produced errors during processing, etc...)
- The EH shall distribute events to specific processing tasks according to information contained in the event header. This requirement comes as an extra functionality in addition to the specialised distribution of events to dedicated sub-farms which can be performed by the DataFlow system.
- The EH shall provide the software infrastructure to perform the required operations inside the Processing Tasks. Filtering is mandatory, as well as functionality for EF monitoring purposes. Additional functionality for general monitoring, calibration checks, etc...

should also be provided. This infrastructure shall be compatible with the offline framework (ATHENA).

- The EH shall be scalable in the sense that increases in either the trigger rate or the event size or the required processing power for an event can be accommodated by increasing only the hardware resources. It shall be independent of the processor architecture.
- The EH shall provide the framework to recover from hardware or software failures while minimising the risk to lose events being processed.
- The EH shall continue to provide its functionality in case of a failure of the Supervision system.

The design of the EH has been made according to the following principles:

- data flow in the EH and data processing are provided by separated entities
- the flow of events is data driven, i.e. there is no data flow manager to assign the event to a specified target
- data copy 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 (PT). There is one EFD process per processing node (i.e. per PC hosting processing tasks). 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. Synchronisation is maintained via messages using UNIX sockets. Details can be found in [9-7] and [9-8].

9.3.2.2 Event Filter Dataflow

The processing of the events is decomposed into steps which 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 processing chain are implemented by ‘tasks’. All ‘tasks’ are derived from a base class `Task`. Each derived class is implemented to provide dedicated functionality. Examples are tasks providing the interface with the DAQ DataFlow, tasks to perform internal monitoring activity (e.g. counting the event which traverse them), tasks to sort events towards different data paths according to internal flags (e.g. the result of the reconstruction and selection process), tasks to duplicate events to send them in parallel towards different processing paths, etc. Some tasks provide an interface with external (with respect to EFD) processing tasks, where independent processes perform the selection process or pre or post processing. An example of an EFD implementation is given in Figure 9-9.

In this example, an *Input Task* makes the interface with the main DataFlow system. Events are counted in an *Internal Monitoring Task*. The *External PT Task* provides the interface for synchronisation and communication with PTs in charge of performing the selection. Events which have not been tagged as rejected 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.

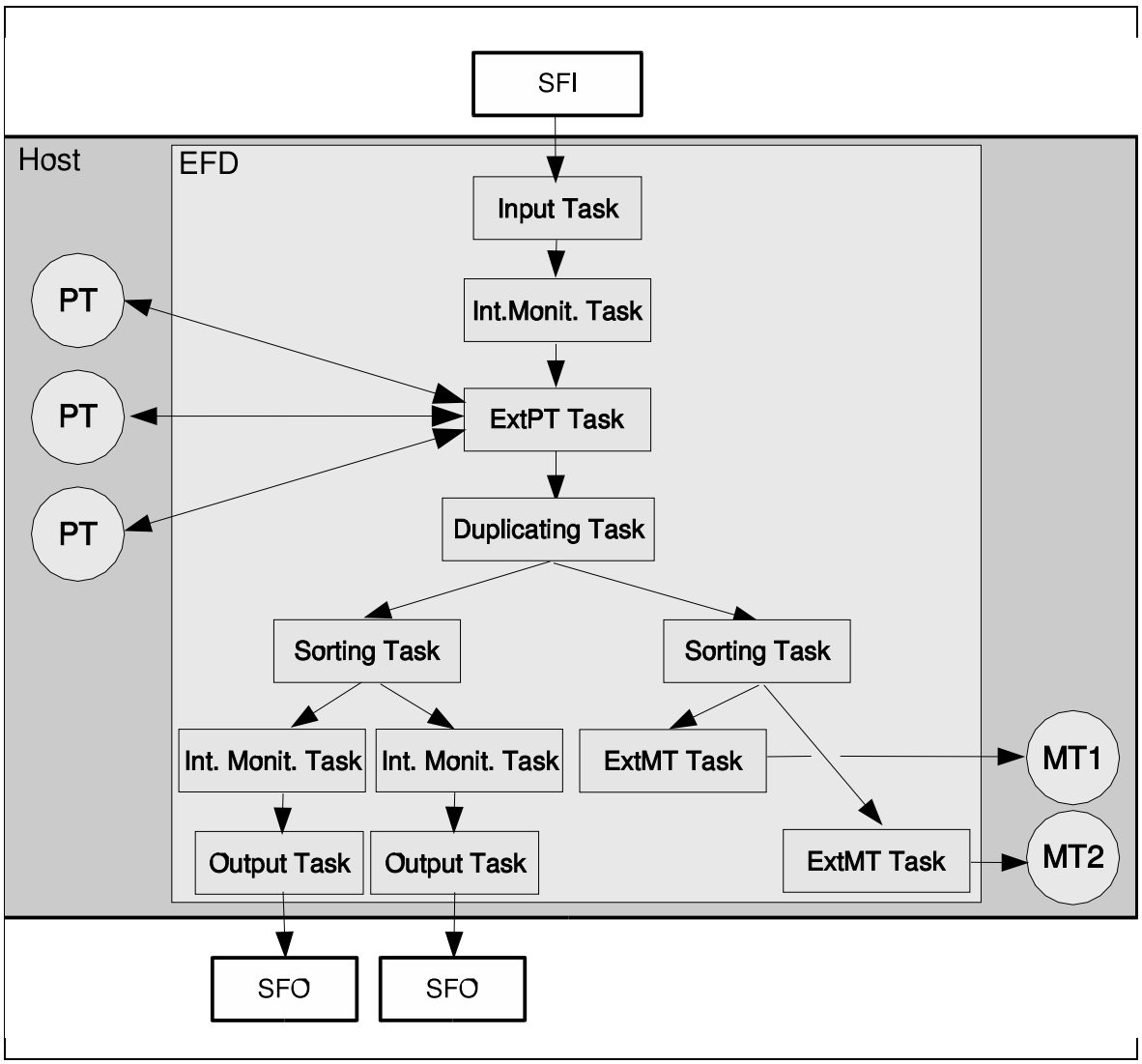

Figure 9-9 A example of an EFD implementation.

The `InputTask` maps events into shared memory (`SharedHeap`). 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 if it is stored in the `SharedHeap`. The file mapping the shared memory segment ensures that data is saved by the operating system in case of problems, e.g. a crash of the EFD process. It is the operating system which is in charge of saving the shared memory into the local disk where the segment is mapped. When the EFD is restarted, events received from the DataFlow can be recovered from the `SharedHeap`. 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 in the sense that each task knows the identity of the next task to execute for the current event. The `Task` base class has a method named `processEvent()` receiving a reference to an event pointer and returning a pointer to the next Task to execute. The backbone of the chaining mechanism is a `Worker` thread which first extracts an event from a Work Queue. The `getWork()` method returns from the Work Queue a pair (Event pointer, Task

pointer). It then calls the processEvent method of the Task passing the pointer to the Event. After processing, the method returns the pointer on the next Task, or the NULL pointer if it was the last task in the chain. Figure 9-10 shows the sequence diagram corresponding to this mechanism. 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.

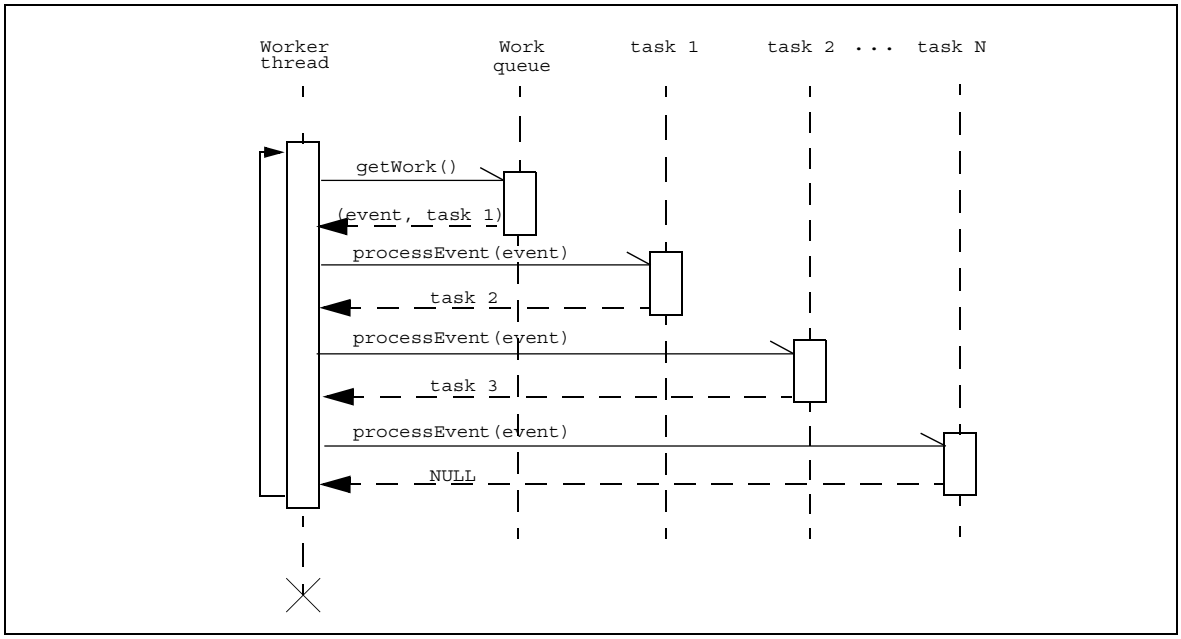

Figure 9-10 Sequence diagram for the work distribution in the EFD.

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 selection algorithms for the strategy described in Chapter 4 ??

The sequence diagram shown in Figure 9-11 shows the synchronisation 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 SharedHeap described in the previous section. Synchronisation 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 the SharedHeap. When processing is completed, PT returns an 'EF answer' to the 'External PT Task' in EFD as a string. This EF answer 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 SharedHeap, 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.

In more detail, communication between EFD and PT is done via the standard ATHENA service ByteStreamCnvSvc. Input is made by selecting ByteStreamEFHandlerInputSvc (instead of ByteStreamFileInputSvc). In the main event loop, the nextEvent method gets a pointer to an event in the SharedHeap, casts it to the standard Event Format Library (EFL) FullEventFragment and passes it to the ByteStreamCnvSvc. At the end of the selection process, an EFresult object is created and registered in the ATHENA Transient Event Store. The

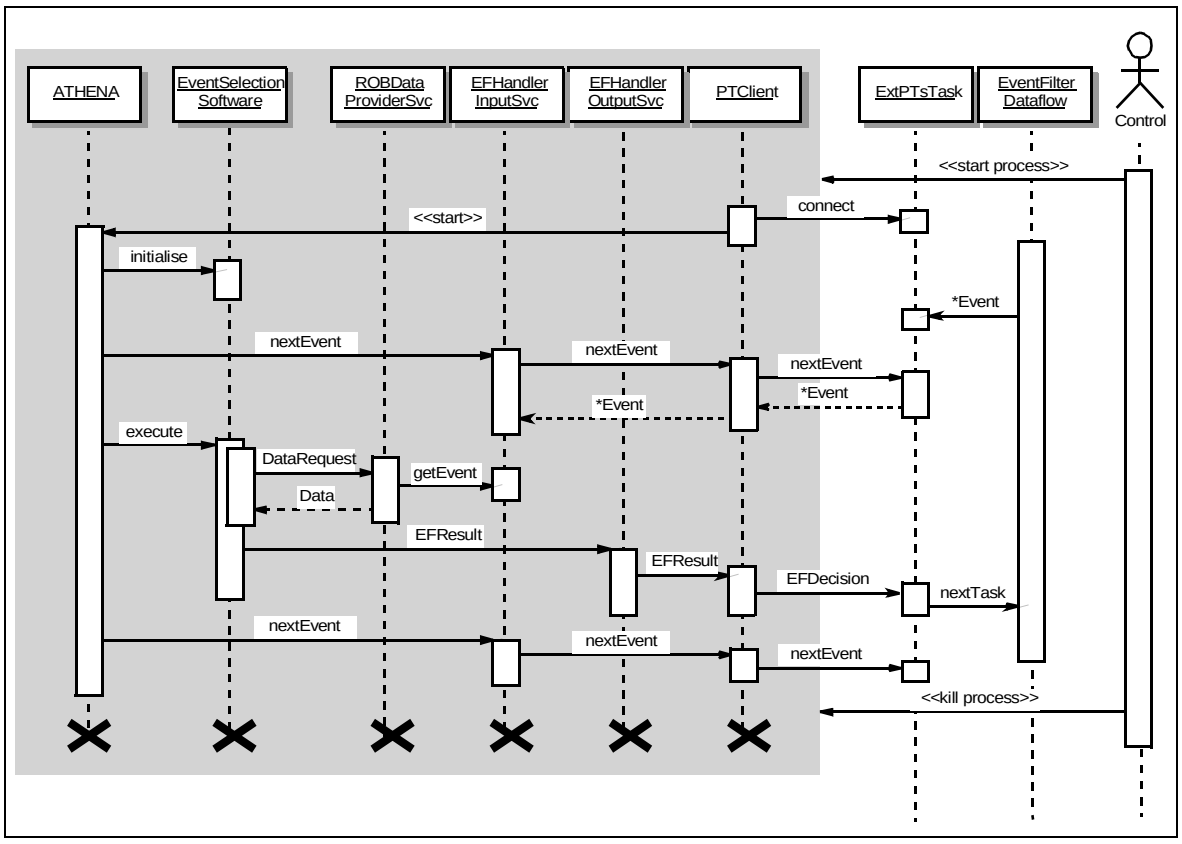

Figure 9-11 Sequence diagram for the interaction between the ESS and the EF PT

`ByteStreamCnvSvc` calls the corresponding converter that creates a ROD fragment containing the relevant data. Transferring to the EFD additional information produced by the algorithms during the selection process can be achieved via converters that serialise the information included in a reconstructed object in the form of a ROD fragment. The `ByteStreamCnvSvc`, will then construct an 'EF sub-detector fragment' of the raw memory type consisting in the standard 'ROS/ROB/RODs' fragment suite. The final step is done in the `ByteStreamEFHandlerOutputSvc`. The EF result is accessed in the Transient Event Store and the answer is checked. The EF sub-detector fragment is serialised directly into the requested `SharedHeap` extension. Finally, the EF answer is passed back to EFD which determines the next step of the processing chain.

9.3.2.3.1 Interfaces with Event Selection Software

In Figure 9-12 a package diagram is shown for the Event Selection Software running in the EF Processing Task. The Event Filter Dataflow has an interface to the external subsystems for the communication with the Event Filter IO. The pointer to the event is transmitted to the Processing Task running ATHENA, which requests 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 depends on interfaces of 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.

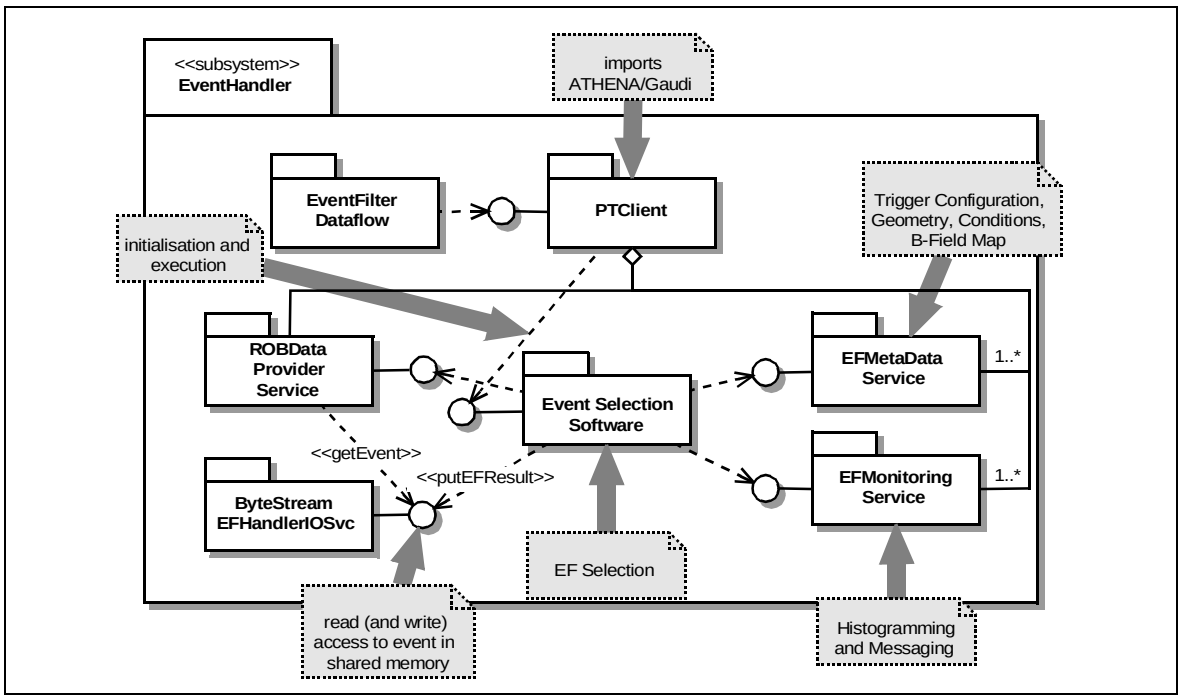

Figure 9-12 Package diagram showing the dependencies of the Event Selection Software performing the EF selection in the Processing Task.

9.3.2.4 Validation tests

9.3.2.4.1 EF data flow stand-alone performances

Two lightweight EFD and PT processes have been used to measure the overhead time introduced by the communication between EFD and PT. The EFD process contains a unique External PT task to provide the PT with dummy data. The PT performs the following sequence of actions: request an event; map the event in the `SharedHeap`; send the dummy EF answer; unmap the event. The measured time to perform this sequence is 66 µs. It does not depend on the size of the event. This overhead is negligible when compared to the expected average processing time (of the order of 1 s).

9.3.2.4.2 EF data flow communication with main DataFlow

In addition to the EFD and PT processes described above, two main DataFlow interfaces have been used: an SFI providing dummy events of variable size and an SFO receiving events and discarding them at once. Two kinds of tests have been made. The first one exercises the full sequence SFI → EFD → PT → EFD → SFO, while the second exercises SFI → EFD → PT → Trash. Both tests have been performed with all processes hosted on a single machine and with all processes hosted on different machines linked by a Gigabit Ethernet link. The results are summarised in Table 9-1.

9.3.2.4.3 Robustness tests

to be written

Table 9-1 EF DataFlow test results

	SFI → EFD → SFO		SFI → EFD (Trash)	
Event size [bytes]	**rate on a single machine [Hz]**	**rate on different machines [Hz]**	**rate on a single machine [Hz]**	**rate on different machines [Hz]**
5k	**6600**	**2400**	**10500**	**3400**
500k	**240**	**111**	**610**	**175**
2000k	**63**	**29**	**143**	**44**

9.3.3 EF Supervision

9.3.3.1 Design

A detailed list of requirements can be found in [9-9]. This list is based on the analysis of constraints coming from other systems and on some first and general uses cases. These requirements are summarised here.

Some constraints arise from the working environment:

- the Supervision system must work coherently with the general ATLAS TDAQ control. In particular, it must map the finite state machine of the TDAQ Run Control. It must comply with the partitioning system.
- 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.

The mandate of the Supervision system is to:

- configure the machines
- configure the software
- provide the Run Control facilities
- provide error handling and recovery facilities, as well as some bookkeeping facilities
- monitor the EF processes
- provide the user with access to the information data produced in the PTs
- provide (optionally) some farm management facilities (*this is still an open question*), i.e. hardware monitoring, operating system maintenance, code distribution on many different nodes, etc.

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 Supervision of the EF is integrated in the HLT Supervision system described in Chapter 12. It makes use of the toolkit provided by the Online Software (see Chapter 10). A tree-like structure has been adopted following the baseline architecture described in Section 9.3.1.3. 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 (Figure 9-13).

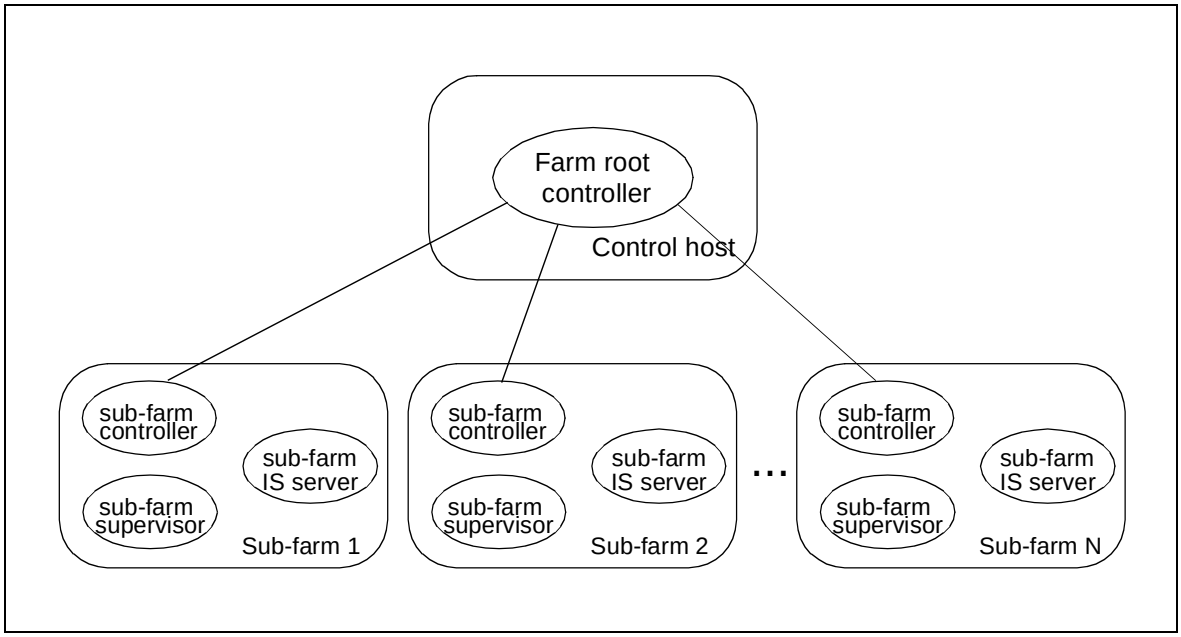

Figure 9-13 EF Farm Supervision organisation.

9.3.3.2 Scalability tests

Scalability and performance tests have been performed on the ASGARD cluster at ETH Zurich and on CERN-IT clusters. Detailed results are given in [9-10] and [9-11].

On ASGARD, the supervision system had been implemented with the 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 seconds for configurations of a size varying between 50 and 250 nodes.

9.3.4 Extra functionality possibly provided by EF

Although not strictly speaking part of the HLT, some functionality can be provided by the EF at a rather low cost in terms of resource usage. The idea is to take profit of some CPU consuming calculations which have been made for the sake of selection (mainly reconstruction) and which can be re-used for monitoring and / or calibration / alignment purposes. This could be done

- directly in EFD context (by-products of calculations performed for selection, in the filtering tasks or in independent monitoring tasks). This functionality has been illustrated in Section 9.3.2.1

- or in dedicated parts of the Farm, specially fed by the main DataFlow, and working under the control of the EF supervision

More details on monitoring in EF can be found in Chapter 7 ??

9.4 Event Selection Software (ESS)

The tasks of the Event Selection Software are “event selection” and “event classification”. Abstract objects representing candidates of e.g. electrons, jets, muons and $J/\psi \rightarrow e^+e^-$, are reconstructed from event data by using a particular set of HLT Algorithms and applying a set of cut parameters. An event is selected if the reconstructed objects satisfy at least one physics Signature given in the Trigger Menu. At both stages, the LVL2 and the EF, events are rejected if they do not pass any of the selection criteria designed to meet the signal efficiency and rate reduction targets of the trigger. The boundary between LVL2 and EF is not precise from a physics event selection point on view. Indeed, flexibility in setting the boundary should be retained in order to profit from the complementary features of both trigger steps.

The Event Selection Software involves the infrastructure and the selection algorithms. The latter are to be provided either by the PESA group or, in 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 Event Selection Software.

In the online the Event Selection Software will run in the software environments provided by the LVL2 Processing Unit and by the Processing Task of the EF, as is shown in Figure 9-9 and Figure 9-12. Hence the ESS needs to comply with the online requirements, like thread safety, online system requirements and services, as well as online performance goals.

It is essential though that the Event Selection Software is also able to run directly in the offline environment ATHENA [9-13] 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 team. For this reason the ATHENA framework was chosen as the framework to run the Event Selection Software inside the EF Processing Task and in the modified form of the PESA Steering Controller inside the LVL2 Processing Unit.

In the offline, it is the task of the ESS is to emulate the full online selection chain. Hence three so called top level ATHENA algorithms are needed, 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 is given in chapter??? (*add reference to chapter 13*).

9.4.1 An Overview of the Event Selection Software

The design and implementation of the Event Selection Software is based on the requirements and use cases documented in reference [9-19]. The current prototype implementation follows the analysis and conceptual design discussed in reference [9-20]. In the following an overview

of the subsystem is given and the design and implementation of the basic concepts are discussed.

In Figure 9-14 the package diagram of the Event Selection Software is shown, which also includes the most important external software dependencies. The ESS is subdivided into four subpackages:

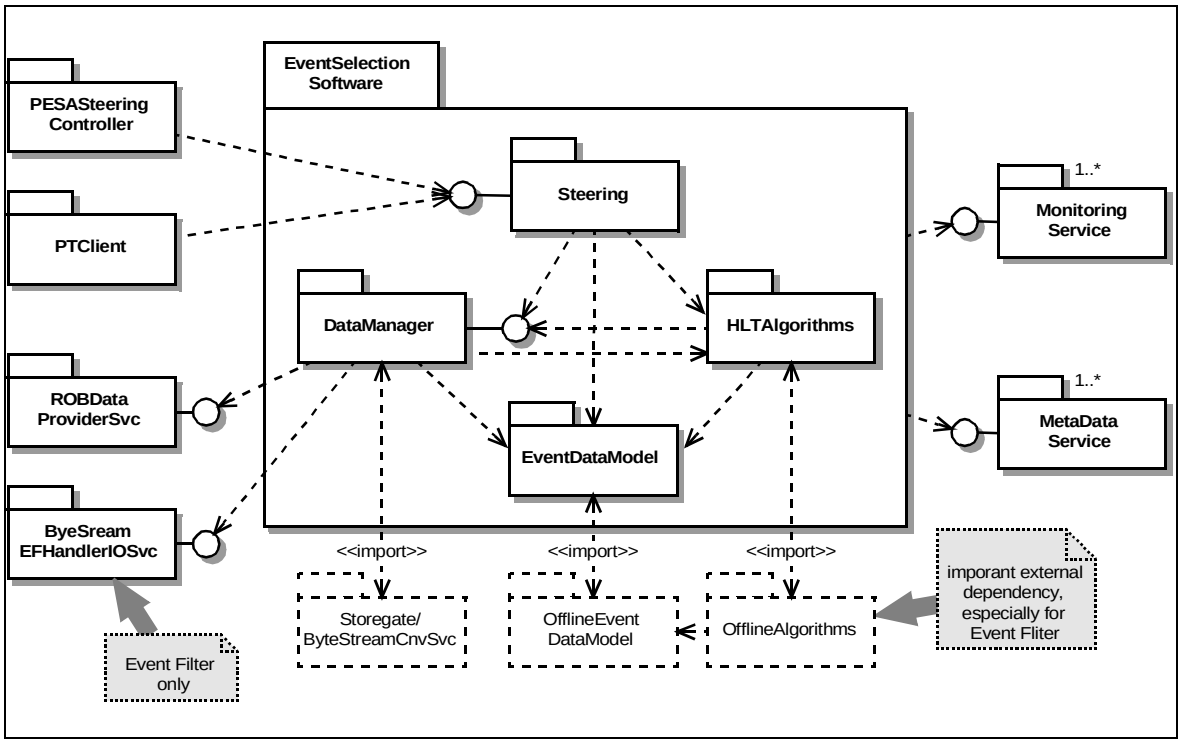

Figure 9-14 A package diagram of the Event Selection Software. Also shown are the dependencies of the subpackages. The dependencies on the offline Event Data Model and on the offline algorithms are explained in the text.

- The **Steering** controls the selection software. It organises 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 running in the LVL2 Processing Unit) 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 their relationships with each other. 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 [9-21] 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.

9.4.2 The Event Data Model Sub-package

The LVL2 and the EF selection is implemented as a software trigger that selects events by means of reconstruction, guided by the RoI information provided by the LVL1 system. Therefore the organisation 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 5 sub-packages:

- **Raw Data** coming from the ROS and the LVL1 system is in byte stream format. The LVL1 Result, the LVL2 and EF Results, as well as ROB Data from the sub-detectors and from the LVL1 system are Raw Data. Part of the Raw Data formats are headers and trailer from the RODs, ROBs, ROSs and DC system, that are defined in [9-22]. Note that for a given sub-detector several Raw Data formats might be used, e.g. different formats depending on the LVL1 Trigger Type. This includes different data content or compression schemes.
- The **Raw Data Objects (RDO)** are an object representation of the raw data from the different sub-detectors. In the EF the RDOs are created at input to the reconstruction chain. However, this object creation poses too much overhead for LVL2 and thus LVL2 converts Raw Data directly to Reconstruction Input Objects (RIOs), i.e. Calorimeter Cells or SCT Clusters.
- **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 Reconstruction Input Objects produced by calibration and clustering, 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 are given in chapter ??? (add reference to section 13.2.2) for the current prototype.
- **MC Truth** information. Together with the first three sub-packages of the EDM, the MC truth is common to both the Event Selection Software and the offline reconstruction. It is needed primarily for debugging and performance studies.
- Another part of the EDM is **Trigger Related Data**. It comprises RoI Objects and the LVL1(LVL2/EF) Trigger Type [9-17], as well as so called Trigger Elements (TEs) and Signatures. A TE labels a set of reconstructed Features and implies (by its label) a physical interpretation of these Features, such as a particle, missing energy or a jet. It represents thus an even higher level of abstraction. TEs are the objects used by the Steering of the ESS to guide the processing and to extract the trigger decision.

An important aspect of the EDM is the relations between different objects in the event. Navigability of such relation is essential for the HLT Algorithm processing, because navigation is used

to analyse and built upon the previously reconstructed event fragments. Examples of instances of EDM classes and their relations are discussed in Section 9.4.3.1 in the context of the Seeding Mechanism.

9.4.3 The HLT Algorithms Sub-package

The task of the HLT Algorithms is to analyse the Raw Data and to reconstruct event fragments 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 Conversion** comprises algorithmic code to convert the Raw Data into objects that serve as input to reconstruction. The task of this type of tool involves sub-detector specific information. Sub-detector groups are responsible for the implementation and maintenance of the code, taking the HLT requirements into account. In the current design the organisation of the Data Conversion is different for the LVL2 and the EF. The EF follows closely the offline reconstruction chain that starts from RDOs. Hence the Raw Data gets first converted into object form and is then processed in order to obtain STC Clusters or Calorimeter Cells. In the LVL2 the step of object creation is omitted in order to avoid the overhead. Here the data preparation is going in one step from Raw Data to RIOs.

  The boundary between Data Conversion and subsequent Feature Extraction is not exact, but note that only Data Conversion will see the Raw Data or Raw Data Objects.

- **Feature Extraction** algorithms operate on abstract Features and Trigger Related Data to refine the event information. They built upon the Data Conversion output and extract the necessary input data to derive the trigger decision. 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. Trigger specific is the use of the information in RoI Objects to restrict the processing to geometrical regions of the detector, which were identified by the LVL1 system. The TEs used by the Steering to “seed” the reconstruction algorithms represent these trigger relevant aspects of the event. 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.

- It is beneficial to structure the algorithm processing in such a way that algorithms from a library of **Algorithm Tools** carry out common tasks such as track-fitting or vertex-finding.

An overview of Reconstruction Algorithms and Algorithm Tools implemented in the current ESS prototype are given in chapters ??? (add reference to chapters 13.2.3-13.2.5).

9.4.3.1 The Seeding Mechanism

Logically the trigger processing is done starting from a LVL1 RoI using predefined sequences of HLT Algorithms. A so called “Seeding mechanism” is needed in order to guide the reconstruction to the event fragments relevant for preparing the trigger decision.

It is beneficial not to couple directly the Steering of the trigger selection to the details of the Feature Extraction algorithms. No “micro” sequencing is done at the level of Features, e.g. individual Calorimeter Cells or SCT Clusters. These complex event details are better handled by the algorithms themselves. Therefore TEs are to be used to characterise with their label the abstract physics objects, e.g. “electrons” or “jets”. Besides this label a TE does not have any properties or states of its own, because the list of possible states or properties is not well defined a priory. Instead it is the “uses” relation by which the TE is associated to the concrete event data, that should provide all necessary information to the HLT Algorithms.

Figure 9-15 shows the evolution of the event fragment associated to one LVL1 RoI at different

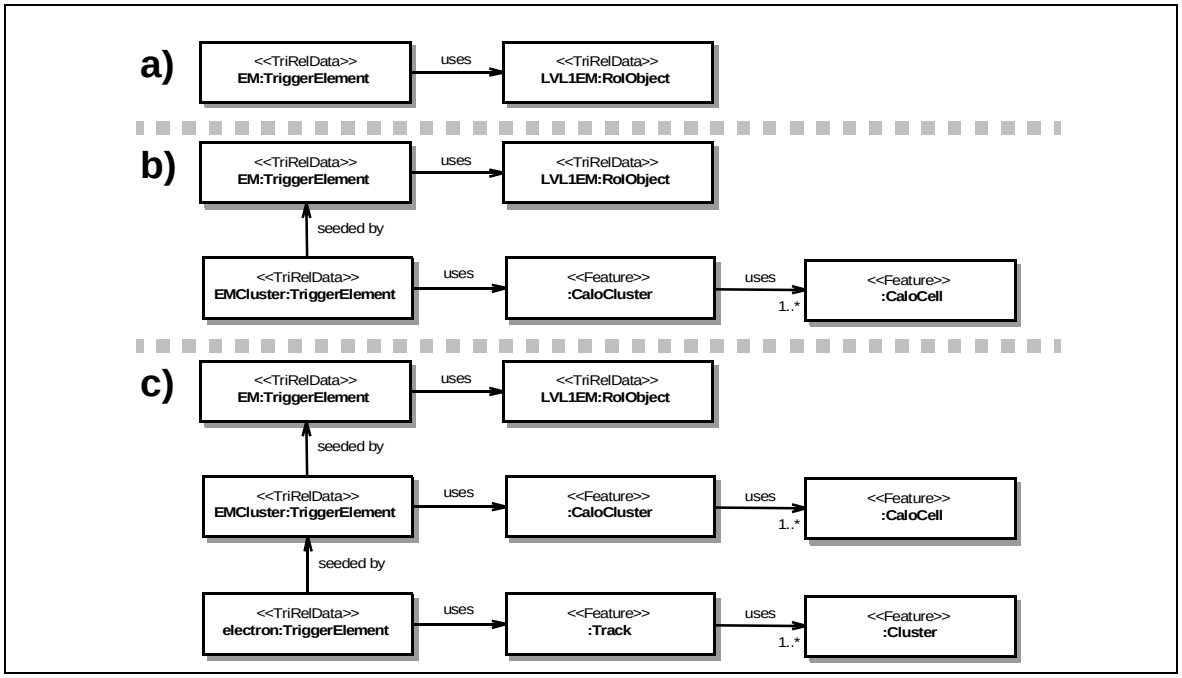

Figure 9-15 Three diagrams showing fragments of the event fragment associated to one RoI at different stages of trigger processing. See text for details

stages of the trigger processing. The upper part Figure 9-15.a shows an example of an electromagnetic RoI Object from the LVL1 calorimeter trigger. The RoI Object is “used” by an “EM” TE to label it for the Steering. Starting from this input TE it is the task of the first HLT reconstruction step to validate the physics hypothesis of having an electromagnetic cluster above a certain threshold. For this hypothesis an output TE with the corresponding label is created by the Steering. 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. It is the task of the HLT Algorithm to validate this output TE.

The first thing the algorithm needs to do is to obtain information about the geometrical position of the LVL1 RoI. It can do so by navigating via the “seeded by” and “uses” relation from the

output TE to the input TE to the “LVL1EM” RoI Object. The algorithm obtains η and ϕ from the RoI and does its pattern recognition work. In our example it creates a Calorimeter Cluster from a set of Calorimeter Cell objects. These objects are linked to the output TE to record the new event information associated with this RoI for later processing. Based on the reconstructed Cluster the algorithm decides if all cuts are passed to validate the hypothesis. The algorithm needs to transmit the result to the steering, which is done by “activating” the output TE in case of a positive decision. The steering will thus ignore all inactive TEs for further processing. The event fragment at the end of the algorithm execution is shown in Figure 9-15.b

The next algorithm in the example is a electron track finding algorithm. It is seeded with a new output TE with the label “electron”. The algorithm is able to navigate the full tree of data objects, as shown in Figure 9-15.b, 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. Because a Track is found the “electron” hypothesis is validated and a TE is activated. The resulting event fragment is shown in Figure 9-15.c.

It is important that the seeding mechanism is general enough to cover all physics use cases. In Figure 9-16 a similar diagram to Figure 9-15.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 one needs to reconstruct all tracks in the event after the initial validation of a “muon”. A TE “uses” the collection of Tracks and seeds, for example, a 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.

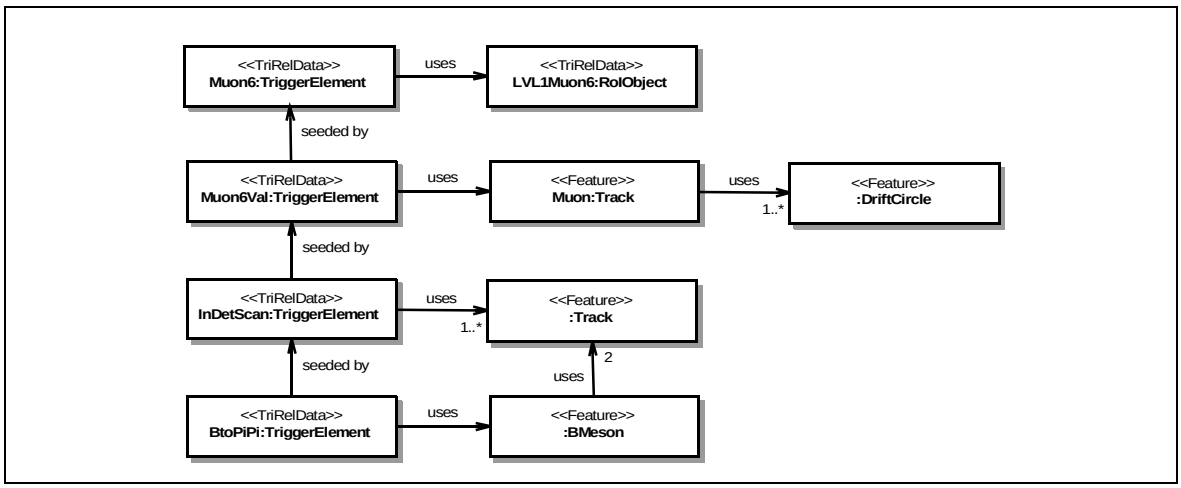

Figure 9-16 A diagram showing a fragment of the event for the case of LVL2 B-Physics trigger reconstruction. See text for details.

9.4.4 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 LVL2 Processing Unit (Figure 9-6) and to the PT Client for running in the EF Processing Task (Figure 9-14). In the offline the Steering is a top level ATHENA algorithm executed directly by the ATHENA event loop manager.

The LVL2 and the EF selection is data driven, in that it builds on the result of the preceding trigger level. It is the task of the Steering 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 event fragment 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 9-15 and Figure 9-16.

The goal of the ESS is to implement the physics selection strategy as discussed in chapter ??? (add reference to chapter 4). The HLT selection demands that an event matches at least one so-called 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 nothing else but a combination of required Trigger Elements with labels like “e25i’. An example for such a menu of Signatures is given in table ??? (add reference to table 4.1).

Many of the Signatures discussed in chapter ??? (add reference to chapter 4) involve more than one required TE. In this case it is beneficial not to completely validate the first required TE from the Signature, because the second may be e.g. due to noise and therefore may fail right away at the beginning of the trigger reconstruction. Therefore 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. Hence concept of “step processing” ensures an early rejection of events.

9.4.4.1 Implementation of the Steering

In Figure 9-17 a class diagram for the Steering sub-package is given. The **Step Controller** inherits from 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”.

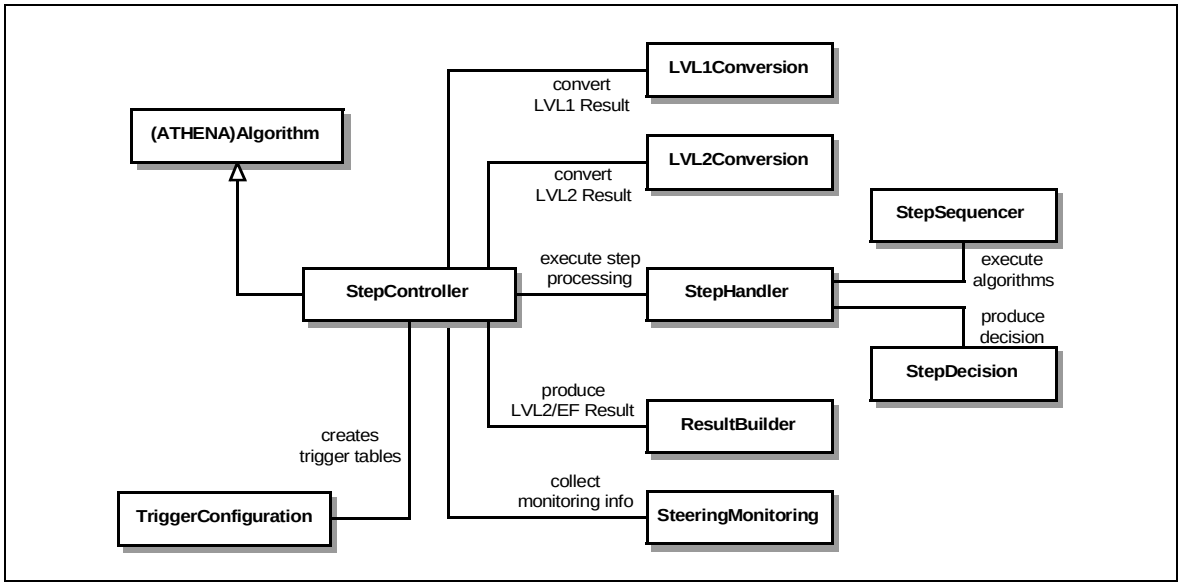

Figure 9-17 A class diagram for the Steering sub-package.

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 initialisation phase (before the start of a “RUN”), which is especially important for LVL2.

In case of the 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 needed for the Seeding Mechanism, as it was shown in Figure 9-15.a. In case of the EF the corresponding **LVL2 Conversion** translates the LVL2 Result to prepare the EF selection.

The trigger processing of each event carried out by the **Step Handler**. It uses the **Step Sequencer** to execute for each step 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 next step is processed by the Step Handler only if the event is still accepted, until all steps are executed. The role of the **Result Builder** is to produce the LVL2 or EF Results, depending on the HLT sub-system 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.

9.4.4.2 The Trigger Configuration

At initialisation time it is the task of the **Trigger Configuration** is to configure 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 ??? (refer to 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 an top-down approach starting from the final Signatures. That 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 gets parsed [9-23] into C++ objects. The first XML file contains the list of final physics Signatures in terms of TE labels, as shown in table ??? (refer to table 4.1). The second XML file contains a list of available Sequences of HLT Algorithms that are implemented and could be used by the trigger. Each Sequence specifies the required input in terms of TEs, the HLT Algorithms to be executed on 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 9-18 using as an example a 2 electron Signature. The final physics Signature “2 x e20i” requires two constituent TEs “e20i”. Such a TE is 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 x e20i” one requires to have in the previous step two “e20” TEs or in other words, the derived Signature is “2 x e20”. The recursion is continued until the LVL1 input TEs are reached (corresponding to the RoI Objects as shown in Figure 9-15.a). In the example from Figure 9-18 this corresponds to 4 recursion steps.

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 Sequence

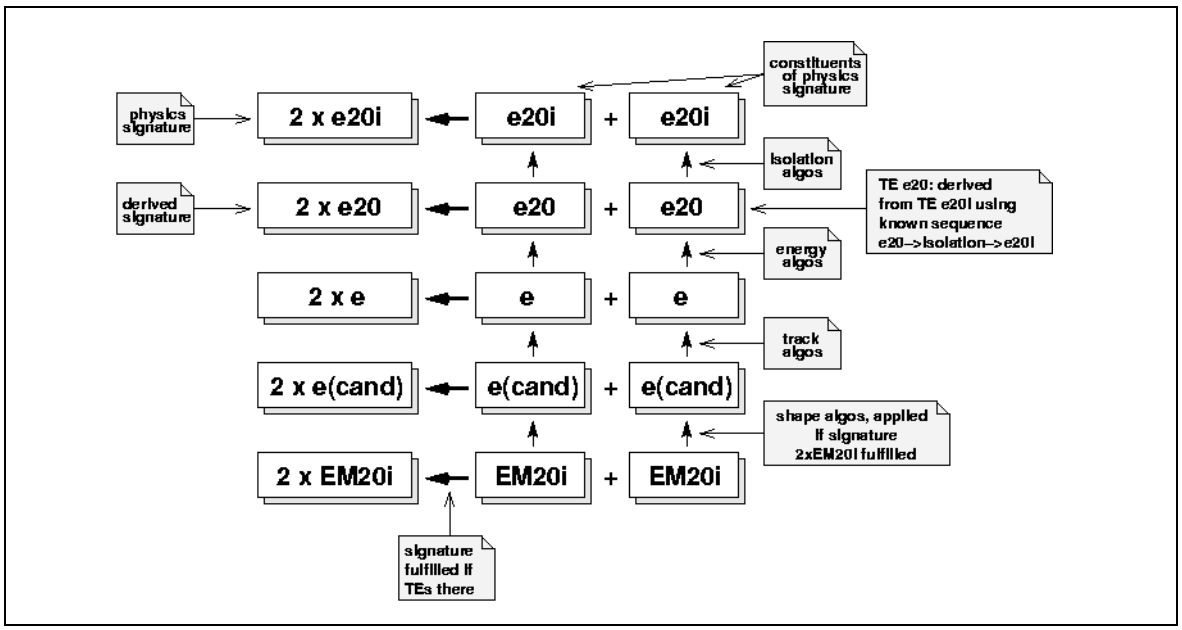

Figure 9-18 Illustration of the recursive configuration of the selection for a 2 electron trigger.

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 form of pairs of **Sequence Tables** and **Menu Tables** for each trigger step.

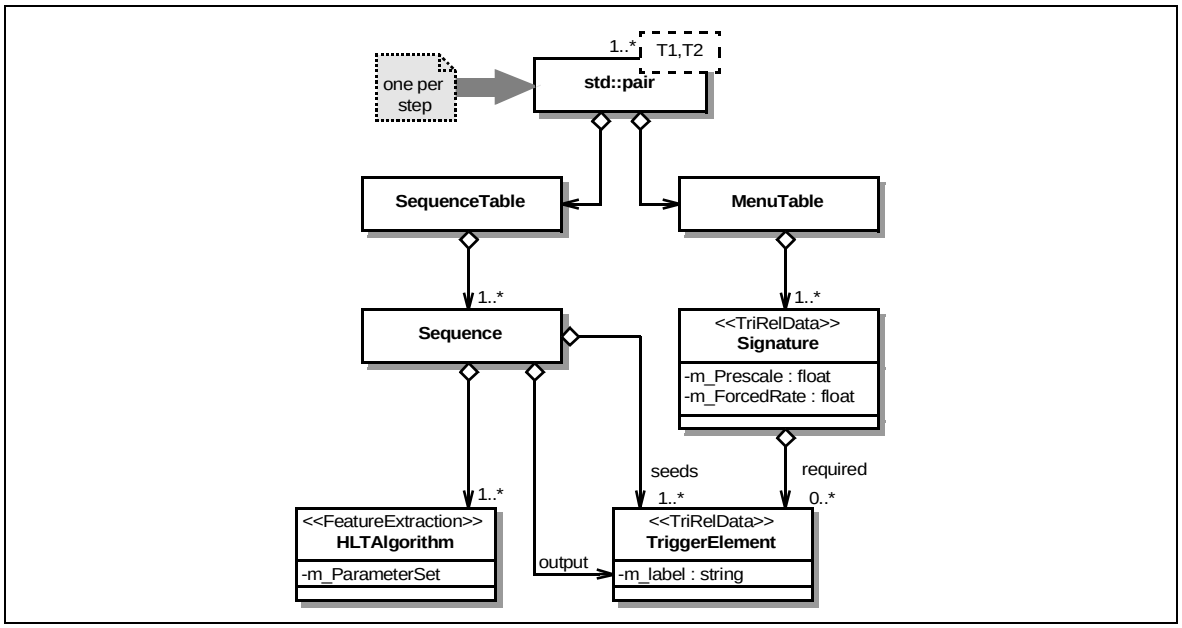

Figure 9-19 A class diagram for output of the Trigger Configuration, showing for each step a pair of a Sequence Table and a Menu Table.

The class diagram is shown in Figure 9-19. They 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.

9.4.4.3 The LVL1 Conversion

At the beginning 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 RoI Builder, see Chapter ???? (add reference to 13.1 LVL1)). 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 value in GeV. To obtain the values it is required to use the configuration of the LVL1 and subdetector specific mapping information. The RoI Objects are then stored for further processing and each RoI Object is decorated with a TE of the corresponding label to enable the Seeding Mechanism as shown in Figure 9-15.

The following of section 9.4 is still not updated and will be changed quite a lot !!!

9.4.4.4 The Step Processing

MONIKA AND GIANLUCA TO UPDATE THIS SECTION.

In Figure 9-20 a sequence diagram is shown for the step handler carrying out the step processing for an event. The step handler gets as arguments the trigger configuration and the initial list of TEs produced by the LVL1 (LVL2) conversion. Each step corresponds to a pair of a menu table and a sequence table. The reconstruction part for each step is controlled by the sequencer. It executes HLT algorithms in sequences to further process the event and to refine the content in terms of TEs. The step decision decides based on the outcome of each reconstruction step whether or not to reject the event.

Depending on the input list of TEs at the beginning of each step the sequencer executes HLT algorithms in sequences, which are defined in the sequence table. The task is to ‘‘seed’’ the HLT algorithms in the sequences executed for all (one at a time) matching combinations of TEs out of the original list of input TEs and to request validation of the resulting output TEs. The resulting list of TEs is passed to the step decision after all sequences in the sequence table of this step have been processed. The step decision then compares the list of output TEs to the signatures in the menu table for this step. For each matching combination of TEs a signature is stored in the data manager. The signature ‘‘uses’’ these TEs in order to prepare the LVL2 (EF) result. The step decision returns a list of TEs, which have been used to satisfy at least one signature. The remaining TEs are discarded from further processing. The step handler continues processing the next step either until it is rejected by the step decision, because no signature is satisfied, or until all steps are done, in which case the event is accepted.

9.4.4.5 Obtaining the LVL2 and EF Results

... GIANLUCA TO CHANGE THE SUBSECTION, SAY SOMETHING ON THE LVL2 RESULT TRANSFER TO PSC...

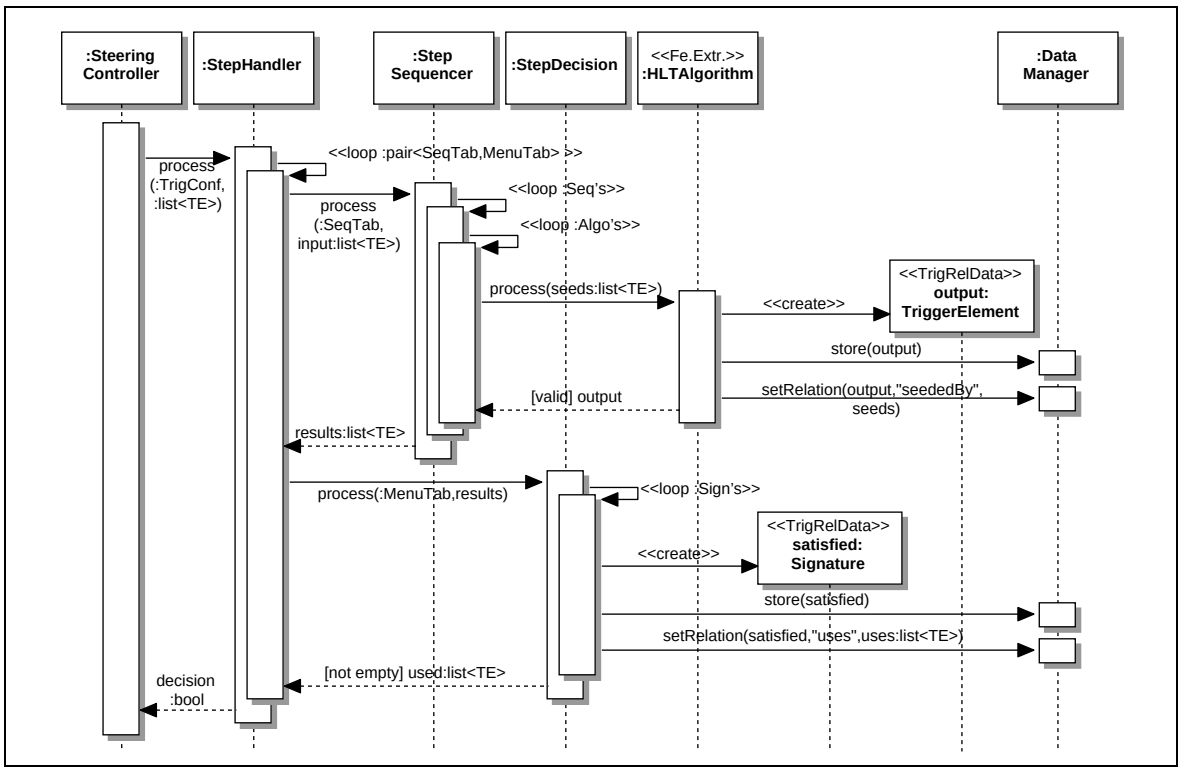

Figure 9-20 A sequence diagram for the step processing showing the role of the step handler, of the sequencer and of the step decision. *UPDATE DIAGRAM FOR THE TE ACTIVATION SCHEME -MONIKA*.

The event summary is executed after the step handler for each accepted event. Its task is to produce the LVL2 or respectively the EF result based on the event information produced. As shown in Figure 9-21, the event summary retrieves the list of validated signatures from the data manager. From this the LVL2 (EF) trigger type is determined. The exact content of the LVL2 or EF results, which are to be constructed from the event information, is not yet defined.

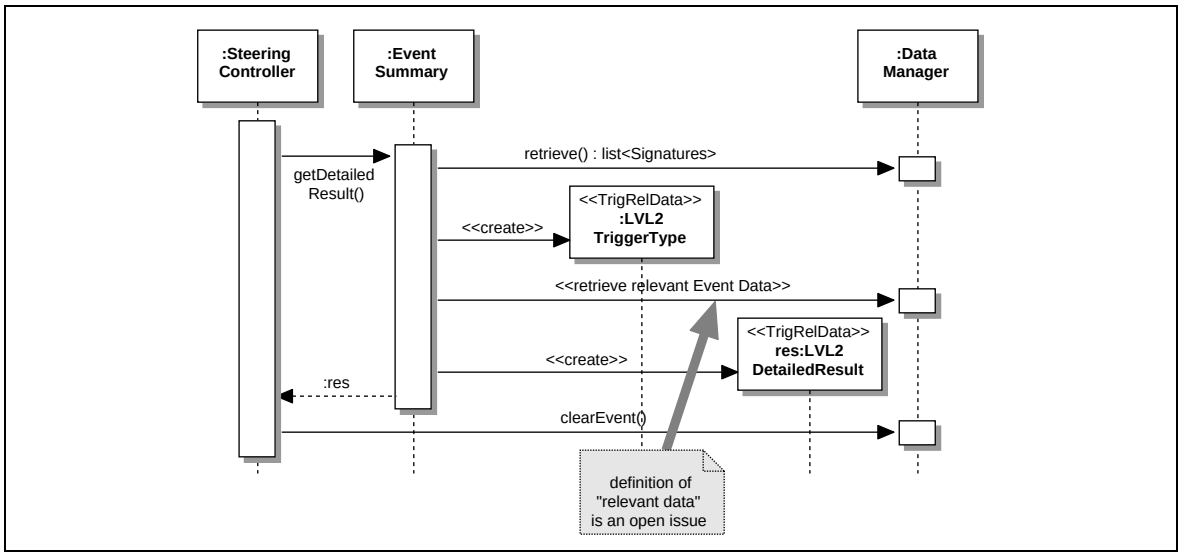

Figure 9-21 A sequence diagram for the Event Summary executed at the end of the processing of each accepted event. *GIANLUCA TO FIX THIS DIAGRAM*.

At the end of the event analysis the data manager has to clean the event data from its memory.

... CRISTOBAL MAY PUT SOMETHING ABOUT THE NEW MONITORING STUFF HERE...

9.4.4.6 Ending a Run of the Event Selection Software

NEEDS UPDATE!!!

The final component of the steering sub-package is the run summary. Its task is to collect the summary information from the HLT algorithms, which are in the sequences of the sequence tables in the trigger configuration. This summary information is published via the monitoring services.

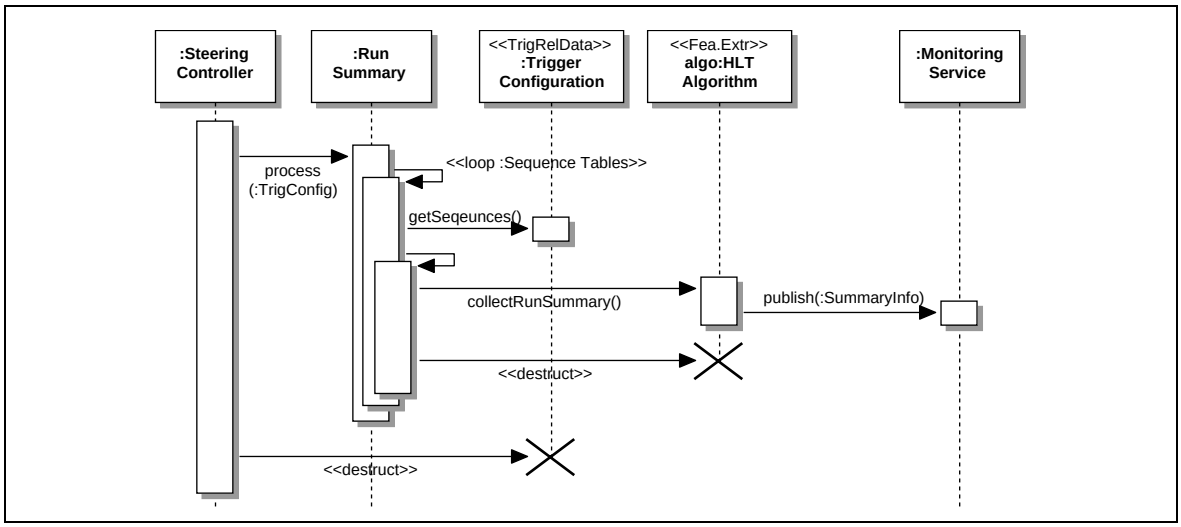

Figure 9-22 A sequence diagram for the Run Summary, which is executed at the end of a “RUN”. *CRISTOBAL MAY UPDATE THE DIAGRAM*

GIANLUCA, PLEASE CORRECT AND PUT THIS SOMEWHERE...

The run summary also destructs the instances of the HLT algorithms. Afterwards, the steering controller destructs the old trigger configuration.

9.4.5 The Data Manager Sub-package

THIS SECTION NEEDS MAJOR REWORK AS THE NATURE OF THE DATA MANAGER HAS CHANGE WITH THE USE OF ATHENA/STOREGATE. THE LONDON SCHEME AND THE BYTE STREAM CONVERSION SERVICE NEEDS TO BE INTRODUCED AS WELL AS THE REGION SELECTOR.

The Data Manager provides the means of receiving and storing the event data during the trigger processing. It therefore provides the necessary infrastructure for the EDM and for the navigation used for the seeding mechanism. In case of LVL2, the data manager does the communication with the ROB data provider. It thereby hides the online aspects the ROB data access from the HLT algorithm processing, which is essential for the PESA algorithm development model, which is based on the reuse of offline software.

DISCUSS THE PRINCIPLES OF THE LONDON SCHEME HERE.

- OFFLINE INTERFACES

- OFFLINE TO ROB MAPPING

- ROB DATA PROVIDER

- LAZY DATA PREPARATION

REGION SELECTION NEEDS TO BE INTRODUCED HERE.

The access to (prepared) data by a geometrical region is a clear requirement for most of the HLT algorithms that follows directly from the RoI concept. The **Region Selection** implements this detector geometry dependent data access pattern centrally. It can then be used easily by all HLT algorithms.

THE INTERFACE TO THE DATA IS A BIT MORE COMPLEX AS IT IS RDO/RIO LEVEL DETECTOR CONTAINERS. THE DESCRIPTION OF THE INTERFACE IS STILL WRONG IN THE FOLLOWING.

In Figure 9-23 a class diagram for the data manager sub-package is shown. The **Data Manager Interface** is implemented by the data manager. It is used by the HLT algorithms and by the steering. In order to provide the necessary functionality the data manager uses several other components:

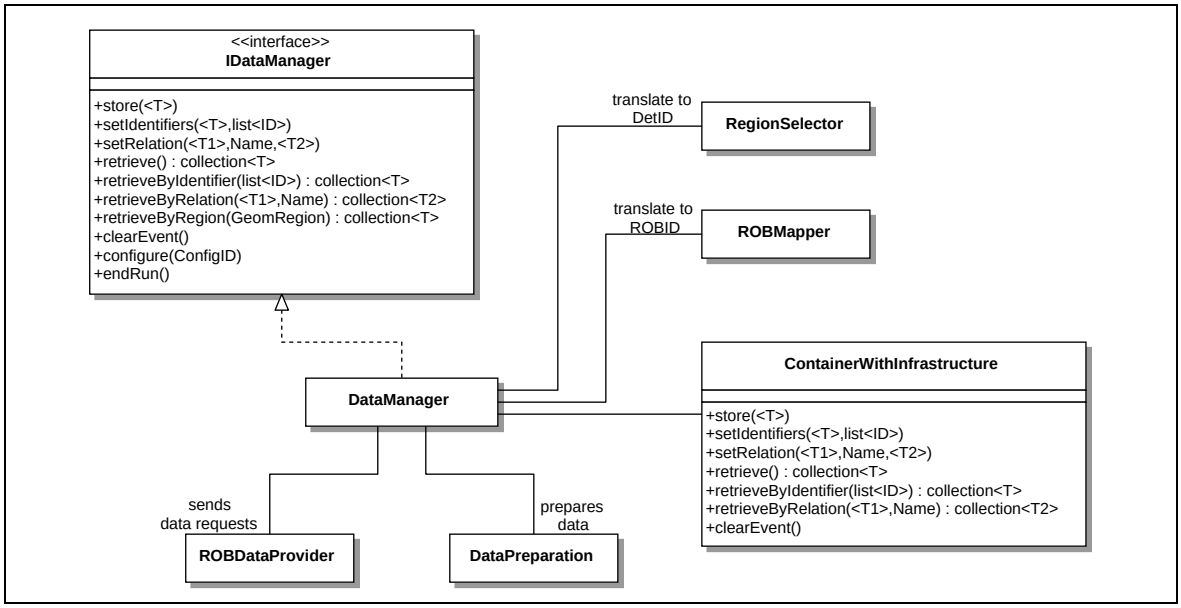

Figure 9-23 The class diagram for the data manager sub-package. See text for details. *THIS NEEDS MAJOR REWORK!!! REGION SELECTION, ROB MAPPER GOES, STOREGATE,...*

- The **Store Gate** is part of the ATHENA framework that provides both, the functionality of a transient store and the infrastructure to implement the raw data access via it persistency mechanism.
- *WRITE SOMETHING ABOUT THE **KEY2KEY** NAVIGATION - ANDREW?*
- The **Region Selector** is used to implement the ‘‘retrieve By Region’’ data access pattern. Their task is to translate the abstract geometry region into a set of (offline) **Identifiers** for **Identifiable Collections**. Those are used to access the data.
- *WRITE SOMETHING ABOUT THE RDO,RIO and **CONTAINERS** THAT HOLD THE DATA - HONG?*
- The **Byte Stream Conversion Service** is...

  EXPLAIN ITS FUNCTION
- The **Data Preparation Tools** are used by the byte stream conversion service to process raw data and to create in the case of the EF the raw data objects or to create in the case of the LVL2 the reconstruction input objects. It hides the detector dependent part of the raw data processing from the HLT algorithm.

The region selector and the byte stream conversion service needs to be configured at the beginning of a ‘‘RUN’’ to access meta data and to set up the data conversion tools used for the data preparation. The same limitations as for the steering and HLT algorithms apply to the meta data access by the data manager, especially for LVL2.

9.4.5.1 Storegate as the Transient Event Store

THIS NEEDS MAJOR REWRITE, IT IS THE OLD TEXT FOR THE CWI BEFORE WE DECIDED TO TAKE STOREGATE.

Storegate as the transient event store of ATHENA is used in the data manager for the function of the container with infrastructure.

STOREGATE:

-DATALINKS

-PERSISTENCY

-ONLINE TO OFFLINE MAPPING

-OWNERSHIP AND CLEANUP

9.4.5.2 The Support for Navigation

ANDREW - ADD SOMETHING ON THE KEY2KEY MECHANISM.

9.4.5.3 The Raw Data Access using the London Scheme

SECTION TO BE WRITTEN UP.

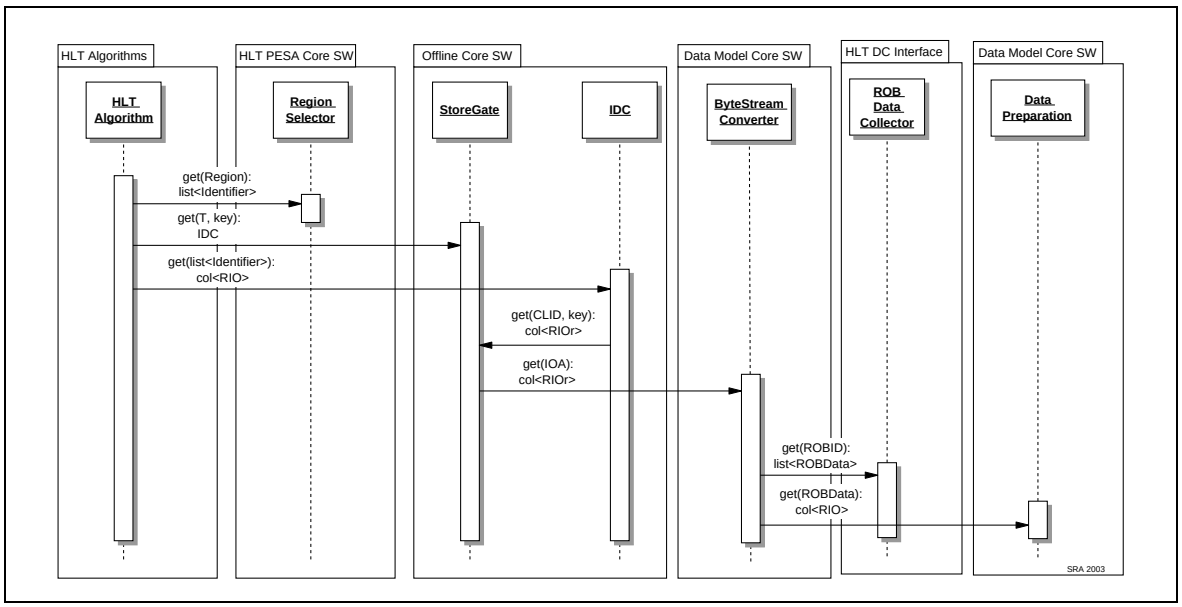

Figure 9-24 The London Scheme

9.4.5.4 Retrieve by Region

STEVE AND ALINE TO REWORK THIS SECTION

The data request by a **Geometrical Region** is an important requirement to implement the RoI concept in the event selection software. A geometrical region does not directly correspond to a RoI, but is a more general concept. A RoI defines a given volume in the detector of interest for the trigger processing, as was shown in figure???. But during the HLT algorithm processing more refined information becomes available that can be used to restrict the original volume and thereby the amount of data to be analysed. On the other hand, a geometrical region could be a full sub-detector or a group of sub-detectors, like for example for the full scan of the inner detector for certain B-physics triggers. Another application of the data request by geometrical region is for example the ‘‘Track Following’’ done by XKALMAN or IPATREC. Here those clusters are to be examined that are on the next detector surface intersected by the track candidate.

In Figure 9-25 a sequence diagram is shown....

STEVE - PLEASE ADD TEXT ACCORDING TO THE NEW DIAGRAM.

Note that the ‘‘retrieve By Region’’ access pattern effectively hides the details of the detector geometry model and of the Identifiers from the requesting HLT algorithms. This pattern does not apply to higher levels of reconstructed objects, i.e. tracks or TEs. Here the access to these objects is by their relation to other objects (seeding mechanism) in the event.

9.4.5.5 Identifiable Containers and the Reconstruction Input Data

WE NEED TO WRITE HERE SOME DETAILS ABOUT THE CONTAINERS AND THE DATA.

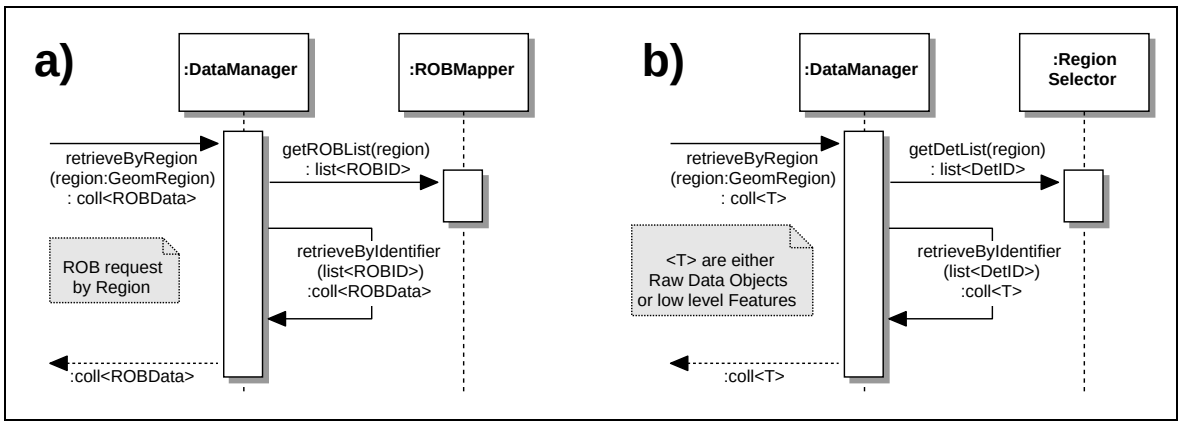

Figure 9-25 Sequence diagrams for the data requests by a geometrical region. The difference between the requests is in the kind of output data. See text for details. *DROP PART A) FROM THE DIAGRAM AND FIX UP PART B.*

It also supports access to data by **Detector Identifier** for raw data objects or low level features, e.g. calorimeter cells or clusters in the SCT. A Detector Identifier corresponds, for example, to a wafer for the Pixels [9-18].

9.4.5.6 Data Request by Detector Identifiers

THIS TEXT NEEDS REWORKING FOR THE LONDON SCHEME.

In Figure 9-26 the corresponding sequence diagram is shown for the data request by detector identifier. Again, the data manager checks, if the data for a given detector identifier is available in the Storegate. If not, the data preparation is used to prepare the requested output, because either raw data objects or low level features are requested. Again, the data is then stored in Storegate to allow for caching before returning it.

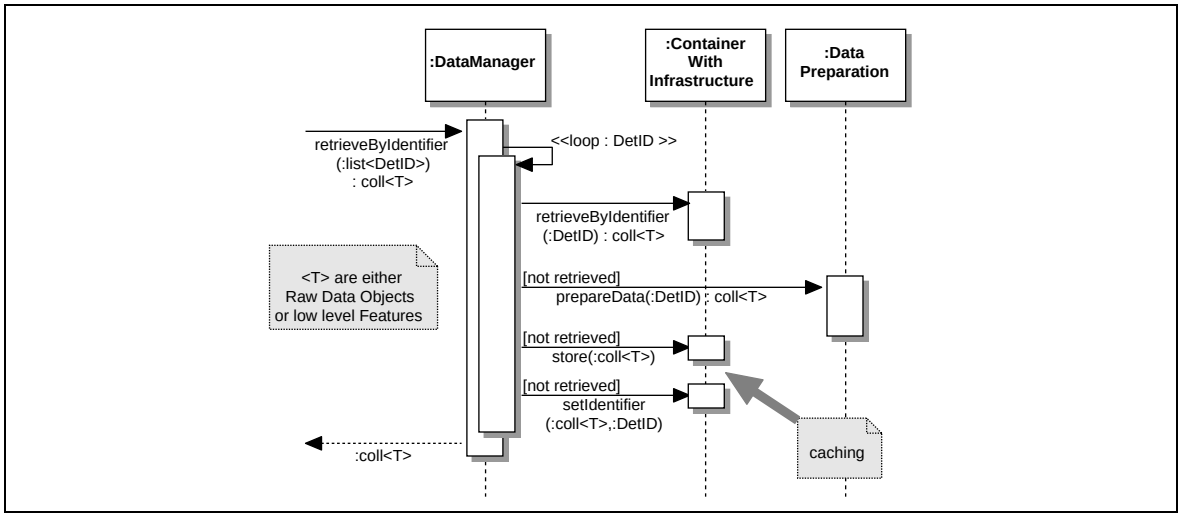

Figure 9-26 A sequence diagram for the data request by detector identifier. See figure??? for details of the data preparation. *UPDATE FIGURE FOR THE LONDON SCHEME IMPLEMENTATION. HENCE, IDC, BYTESTREAM-ADDRESS/PROXY, PERSISTENCY.*

9.4.5.7 ROB Data Request and Lazy Data Preparation

UPDATE THIS TEXT FOR THE LONDON SCHEME.

In Figure 9-27 a sequence diagram is shown for the data preparation. It prepares raw data objects or low level Features if they are not already stored in Storegate, as was shown in Figure 9-24. This way the important performance requirement of so called “lazy data preparation” is implemented, because only the requested part of the data is actually processed, when it is needed by the HLT algorithm.

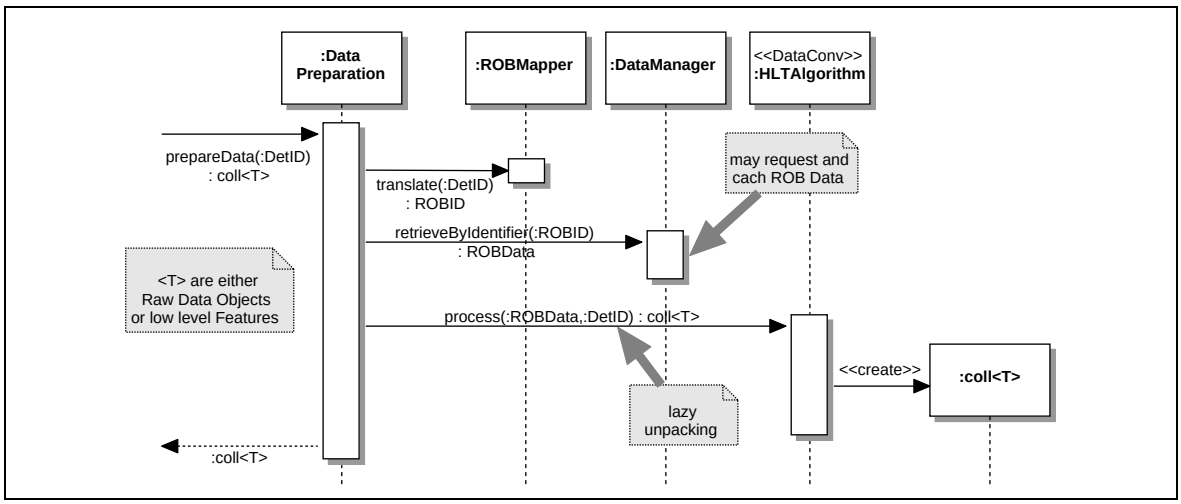

Figure 9-27 A sequence diagram for the data preparation that executes the data conversion algorithms inside the data manager. *REWORK - ROB DATA ACCESS AND BYTE STREAM CONVERTER.*

THIS TEXT BELOW IS WRONG, NEEDS UPDATE.

As shown in the figure, the data preparation uses the ROB mapper to translate the detector identifier into a ROB Identifier. Then the ROB Data is requested from the Data Manager itself. This request was discussed in figure???. The Data Preparation uses a Data Conversion Algorithm to create the requested output data from the part of the ROB Data that corresponds to a given Detector Identifier. It is an option, that only a part of the full ROB Data fragment gets transmitted from the ROS.

9.4.6 Further Issues

THIS IS A PLACEHOLDER FOR MORE STUFF TO COME. THE FOLLOWING STUFF NEEDS PROPER WRITE-UP FOR THE NEXT DRAFT:

- WRITE SOMETHING ON THE ISSUES OF BUNDLED ROB REQUESTS.
- An issue that needs to be addressed by the detailed design and implementation are HLT algorithm error conditions. They are to be handled by the sequencer and by the data preparation.
- The LVL2 supervisor or the PESA steering controller may timeout events that are being processed in the LVL2. In such a case the LVL2 result might not be sent to the ROS and might therefore not be available at input to the EF. Another possibility is that the event se-

lection software sends only a partial result because of an error condition. In both situations the EF might be forced to revert back to the LVL1 result to seed the selection, which needs to be taken into account in the detailed design of the EF selection strategy.

- The meta data services are needed to provided the necessary information for the event selection software. An important issue here is the lifetime of the information. In the LVL2 all meta data accesses are to be done before the start of the ''RUN'', while for the EF updates per time period may be possible.
- The monitoring 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 the information.
- It is desirable to allow for a graphical event display in order to visualise the trigger reconstruction and decision making process. This could be done on the basis of the event information transmitted via the LVL2 and EF results that may include all reconstructed Features for debugging purposes.
- For the online system a graphical user interface will be useful for both the control and the monitoring of the event selection software.

9.5 References

9-1 LVL2 URD

9-2 FPGA Processor note - from Andreas Kugel

9-3 FPGA Algorithm studies - from Andreas Kugel

9-4 F. Touchard et al., *HLT operational analysis and requirements to other sub-systems*, https://edms.cern.ch/document/363032/1

9-5 HP. Beck et al., *ATLAS TDAQ, a network-based architecture*, TDAQ Data Collection note 59, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DataFlow/DataCollection/docs/DC-059/DC-059.pdf

9-6 C. Bee et al., *Event Handler Requirements*, https://edms.cern.ch/document/361786/1

9-7 C. Meessen et al., *Event Handler Functional Design Analysis*, http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/EF/EFD-FunctionalAnalysis.pdf

9-8 C. Bee et al., *Event Handler Design*, https://edms.cern.ch/document/367089/1.1

9-9 C. Bee et al., *Supervision requirements*, https://edms.cern.ch/document/361792/1.1

9-10 C. Bee et al., *ATLAS Event Filter: Test Results for the Supervision Scalability at ETH Zurich*, (2001), http://documents.cern.ch/cgi-bin/setlink?base=atlnot&categ=Note&id=daq-2002-006

9-11 S. Wheeler et al., *Test results for the EF Supervision*, https://edms.cern.ch/document/374118/1

9-12 *Guidelines for writing thread-safe LVL2 algorithms*, https://edms.cern.ch/document/...

9-13 ATHENA

9-14 Gaudi

9-15 S. Gonzalez et al., *Use of Gaudi in the LVL2 Trigger: The Steering Controller*, https://edms.cern.ch/document/371778/1

9-16 A. Bogaerts et al., https://edms.cern.ch/document/375305/1

9-17 M. Abolins et al., *Specification of the LVL1/LVL2 Trigger Interface*, https://edms.cern.ch/document/107485/1

9-18 ID EDM requirements note

9-19 S.George et al., *PESA high level trigger selection software requirements*, ATLAS Internal Note, ATL-DAQ-2001-005 (2001)

9-20 M.Elsing et al., *Analysis and Conceptual Design of the HLT Selection Software*, ATLAS Internal Note, ATL-DAQ-2002-013 (2002)

9-21 Storegate

9-22 EventFormatLib

9-23 Xerces

9-24 RoIBResult (RoI word definition)

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 customizable 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.

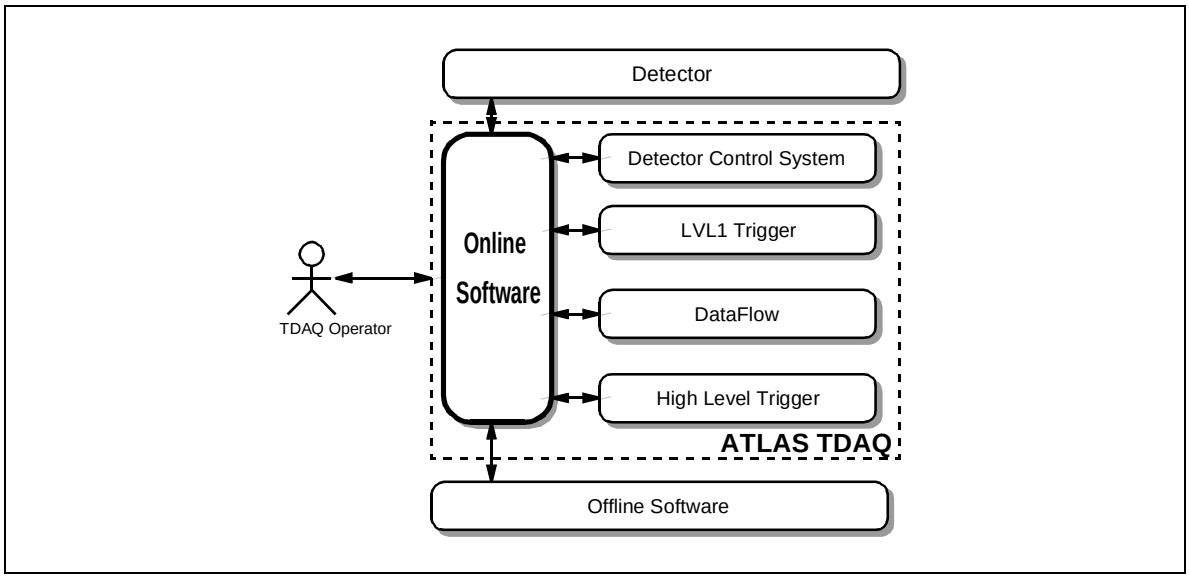

Figure 10-1 Context diagram of the Online Software

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 synchronisation of the states of a run in the entire TDAQ system and for process supervision. These procedures are aimed to take a minimum amount of time to execute 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 for early and efficient problem finding. Configuration database services are provided for holding the large number of parameters which describe the system topology, hardware and software components and running modes, based on the partitioning model. During data taking, access 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 is provided. User interfaces display the status and performance of the TDAQ system and allow the user to configure and control his operation. These interfaces provide comprehensive views of the various sub-systems at different levels of abstraction.

The Online Software distinguishes various types of users. 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 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.

The following types of Online Software users have been identified:

1. *TDAQ Operator - this user is responsible for using the TDAQ system to take data during a particular data taking session, for example during a shift. He has to be able to start, monitor and stop data taking. He is not expected to perform any programming tasks related to the Online Software.*
2. *TDAQ Expert - this user is responsible for running and maintaining the TDAQ system itself. He is responsible for the initialisation and shutdown of the TDAQ system or parts of it. He shall have a knowledge of the TDAQ structure and its components.*
3. *TDAQ Sub-System or Detector Expert - this user is responsible for the operation of a particular sub-system of the TDAQ or particular sub-detector of the ATLAS detector. He should be capable of describing the specific TDAQ sub-system or detector configuration and diagnosing the specific sub-system or detector problems which may appear during operation.*
4. *TDAQ Sub-System or Detector - this user represents an application of another TDAQ Sub-System or Detector. It will use the services provided by the Online Software for the sub-systems and detectors configuration and control.*

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 on the architecture and a comprehensive set of use cases are described in [10-2]. Each of the packages is associated with a functionality group 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 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 initialisation and shutdown, to provide control command distribution, synchronisation, 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 of the TDAQ system and detectors. 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.

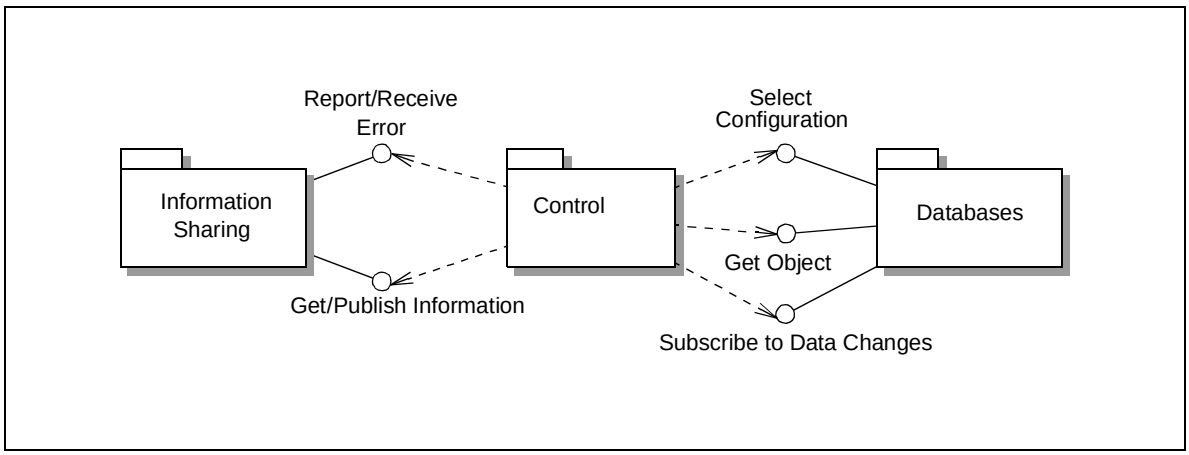

Figure 10-2 Internal Interaction between the Online Software packages

The following sections describe these packages in more details.

10.3 Information Sharing

The choice of name for this section is not final. A possible alternative could be 'Monitoring services'. This would then be applied to the whole of the document where one talks about these services.

There are several areas where information sharing is necessary in the TDAQ system: synchronisation 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 between TDAQ software applications. This chapter 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 the other TDAQ applications. The first type of application will be called in this chapter Information Provider, the second one will be 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.

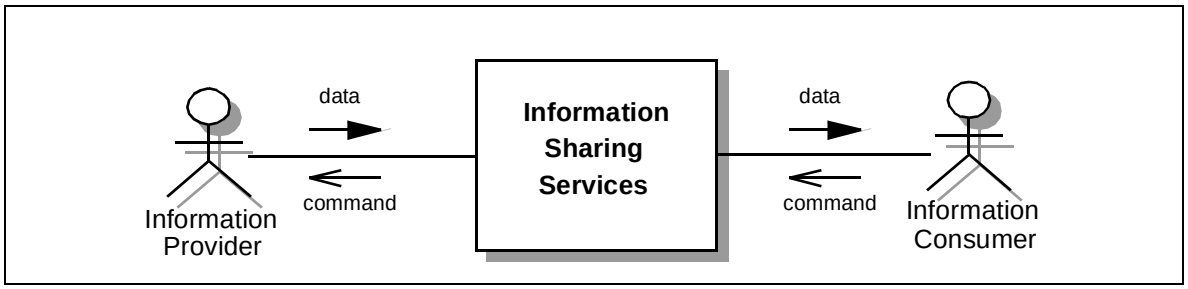

Figure 10-3 Information Sharing in the TDAQ system

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 (i.e. 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

Depending on the type of the information the performance and scalability requirements vary. 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 x 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 which monitor the status of a particular components 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 hundreds of 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 those 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, shall be able to handle several hundred Information Providers and several tens 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.

The Inter Process Communication (IPC) is a basic communication service which is common for all the other Online Software services. It defines a 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 to run several instances of the Online Software services in different TDAQ Partitions concurrently and fully independently.

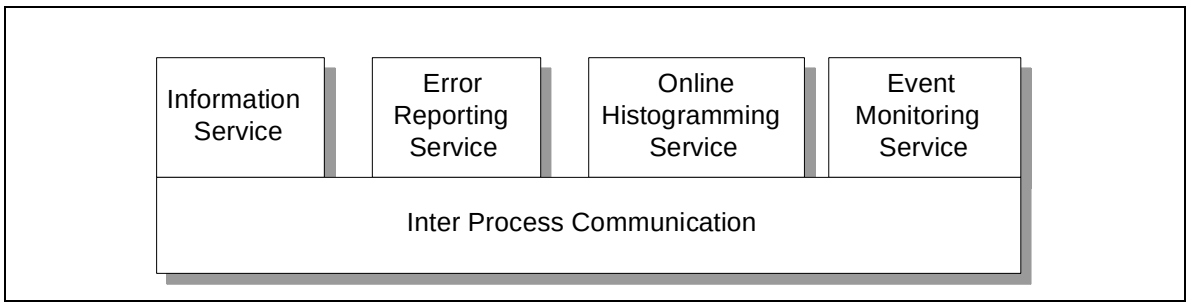

Figure 10-4 Information Sharing services

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.

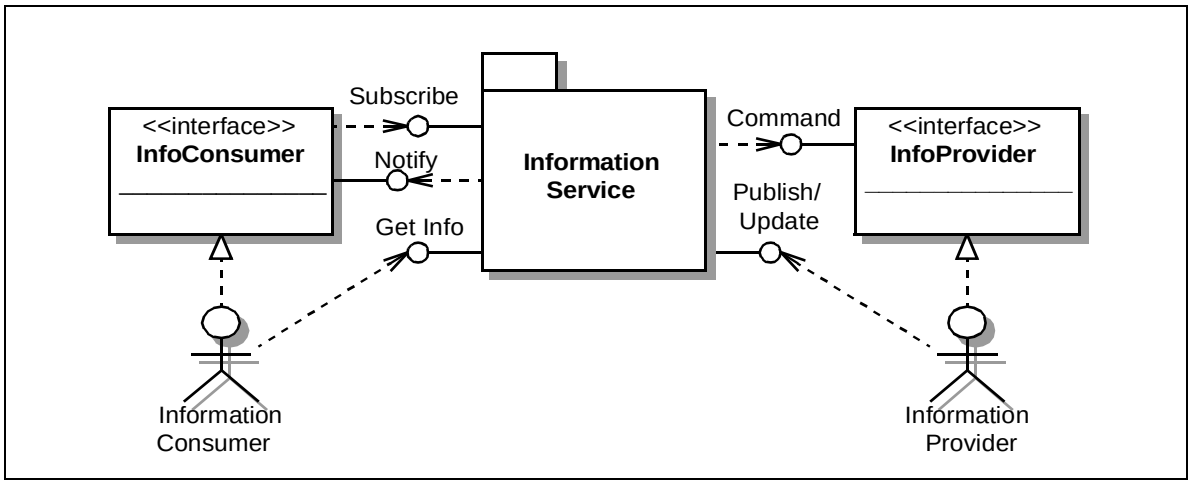

Figure 10-5 Information Service interfaces

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: either 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, or 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, i.e. setting time interval for the information updates.

There are also two types of Information Consumers. 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.

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.

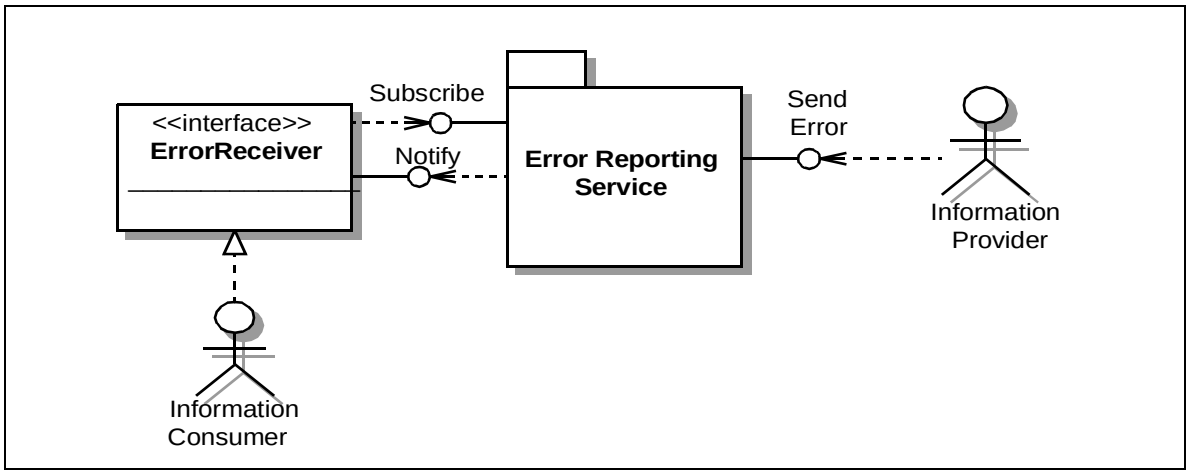

Figure 10-6 Error Reporting Service interfaces

An Information Provider can send the 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 the error messages an Information Consumer has to implement the ErrorReceiver interface and construct the criteria which 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 pre-defined format. Figure 10-7 shows interfaces provided by the Online Histogramming Service.

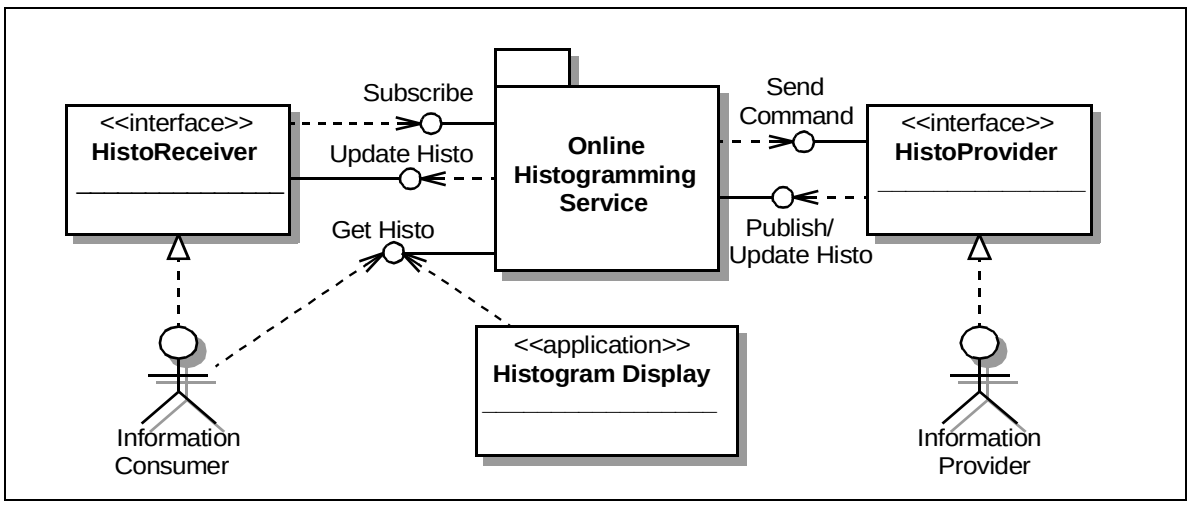

Figure 10-7 Online Histogramming Service interfaces

The OHS sub-package will provide also a human user interface in a 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 transportation of physics events or event fragments sampled from well-defined points in the data flow chain to the software applications which can analyse them in order to monitor the state of the data acquisition and the quality of physics data of the experiment. Figure 10-8 shows the main interfaces provided by the Event Monitoring Service.

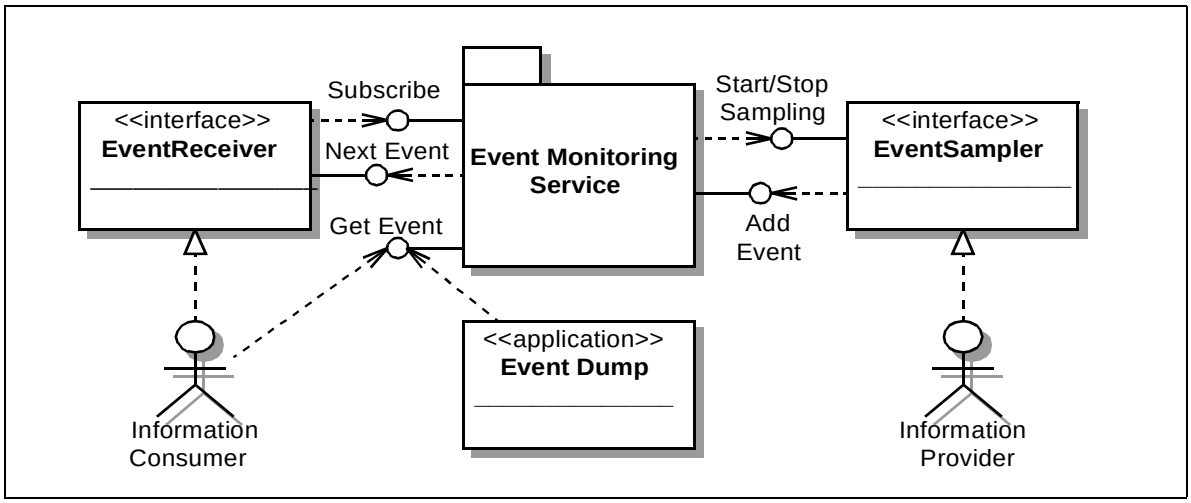

Figure 10-8 Event Monitoring Service interfaces

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 interfaces 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 because of the use of 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 it's own specific interfaces but reuses all the communication mechanisms provided by the IS.

10.3.4 Application of Information Sharing services to the TDAQ sub-systems

Usage of the information services by the other TDAQ systems, concentrating on differences with general use.

Should be provided by TDAQ systems

This sub-section should only exist if the information is not already covered in Chapter 7, "Monitoring".

There is already some information in the sections 10.3.1.1 and 10.3.2.

10.3.5 Prototype evaluation

The prototype implementations exist for all the Information Sharing services. These prototypes are aiming to verify the feasibility of the chosen design and the choice of implementation technology for the final TDAQ system, and are used in the ATLAS test beam operations. This chapter contains a description of the services implementation along with their performance and scalability test results.

10.3.5.1 Description of the current implementation

The Online Software provides prototype implementations [10-4] for all the Information Sharing services. Each service is implemented as a separate software package with both C++ and Java interfaces. All the services are partitionable in the sense that 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) [10-3] 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, object location transparency.

Currently the open source implementation of CORBA provided by Xerox company is used. It is called Inter Language Unification (ILU) [10-4]. Several other CORBA implementation are currently being evaluated. They are namely: 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 another CORBA implementation without affecting the architecture and design of the Information Sharing services.

10.3.5.2 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 the 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 frequency 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 last 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.

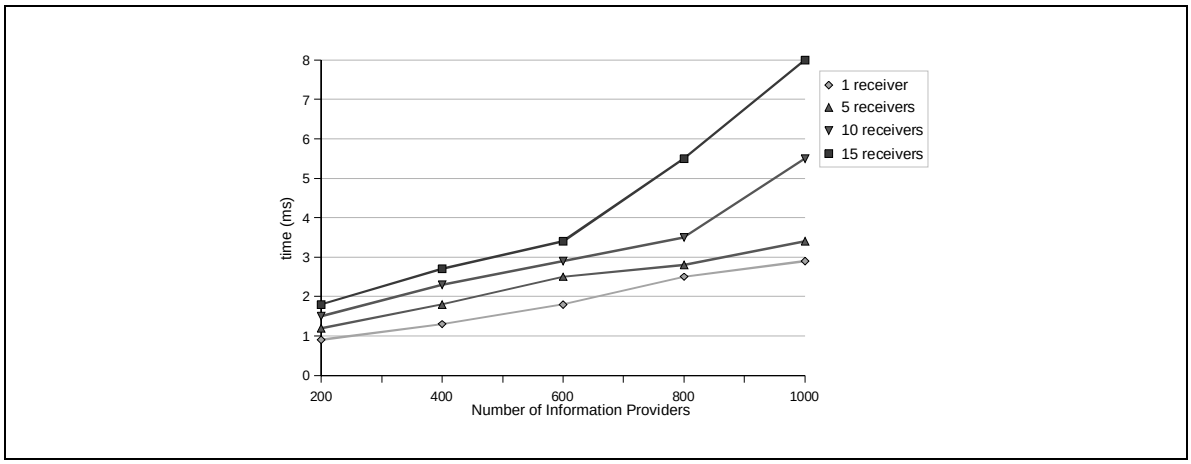

Figure 10-9 Time spent to transport one information object from one Information Provider to a number of Information Consumers versus the number of concurrent Information Providers

Figure 10-9 shows the average of the 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 thousand Information Providers and about 10 Information Consumers at the same time. The design of the IS allows to distribute the total load among a number of IS server which do not affect each other. 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.

10.4 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** to provide the description of the system configurations,
- the **online bookkeeper** to log operational information and annotation,
- the **conditions databases interface** to store and read conditions under which data were taken.

10.4.1.1 Configuration Databases

The configuration databases (ConfDB) are used to provide the overall description of the TDAQ systems and detectors hardware and software. It includes the description of partitions defined for the system and parameterized for different types of runs describing the hardware and software used by a given partition and their parameters.

The configuration databases are organized in accordance with the actual 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. The configuration databases provide graphical user interfaces for the human user to browse the data and to modify the data by authorized human experts. 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 configuration databases provide an efficient mechanism for fast access to the configuration data for huge number of clients during data taking. They do not store the history of the data changes which is the task of the Online Bookkeeper but provide archiving options. Configuration data which are important for offline analysis must be stored in the conditions databases.

10.4.1.2 Online bookkeeper

The online bookkeeper (OBK) is the system responsible for the online storage of relevant operational information and configuration description provided by the TDAQ systems and detectors. The OBK organizes the stored data on a per-run basis and provides querying facilities.

The online bookkeeper provides 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 online bookkeeper provides programming interfaces for user applications to browse the information or to append annotations.

10.4.1.3 Conditions Databases interfaces

The TDAQ systems and detectors use the conditions databases to store conditions which are important for the offline reconstruction and analysis. The conditions data are varying with time and parameters such as temperature, pressure and voltage. These conditions are stored with a validity range which is typically expressed in time or run number and retrieved again using time or run number as a key.

The conditions databases are expected to come from an LCG 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 insure the required performance during data taking runs.

10.4.2 Performance and scalability requirements on the Databases

The performance and scalability requirements for the database services strongly depend on the strategies chosen by the TDAQ systems and detectors to transport the data to their applications, to store the conditions and to keep the operational data. Roughly, for most TDAQ systems and detectors, the number of estimated clients of configuration and conditions database is equal to the number of local controllers, which varies by different estimations from 500 to 2000. For the Event Filter the situation is not defined yet and the number of database clients in the worst scenario can be $O(10^3)$, if each processing task will receive configuration description and conditions.

The complete databases information can be split into a number of independent parts which are specific for the TDAQ systems and the detectors. The complete description is required only to few applications, while the most others require to access only a small fraction of it. The typical database size which completely describe all necessary parameters of a TDAQ system or a detectors for a physics run is about $O(10^2)$ Mbyte. The DCS may produce up to 1 Gbyte of measured conditions per day.

The configuration databases data are read once during the start of the data taking and an acceptable time to get the description for the whole system is of O(10) s. During the data taking of physics data 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 Mbytes per hour.

10.4.3 Architecture of Databases

10.4.3.1 Configuration databases

The configuration databases 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 prepare the system description and to define configurations, which can subsequently be browsed by a user.

A TDAQ or detector application accesses 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 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 it’s changes. The DAL is also used by an expert process to produce the configuration data.

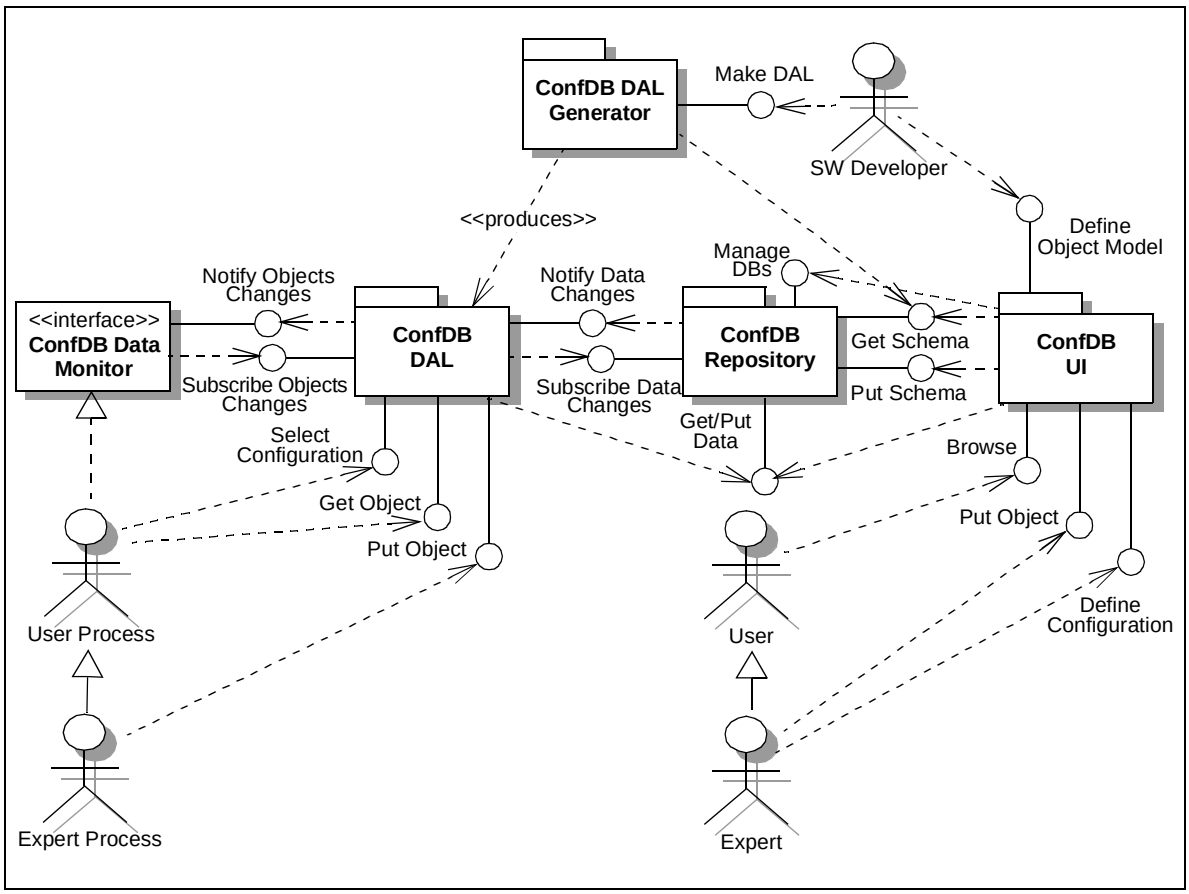

Figure 10-10 Configuration databases users and interfaces

The Figure 10-10 shows the main classes and interfaces provided by the configuration databases and their users.

The ConfDB system contains the following classes:

- **ConfDB Repository** - defines low-level interfaces to manipulate the configuration databases including databases management by users which 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 shall never use DBMS-specific methods directly;
- **ConfDB UI** (User Interface) - defines a user interface for object model definition, configurations definition, database browsing and population by human actors;
- **ConfDB DAL Generator** - defines an 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, being that some of them are human oriented, 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.

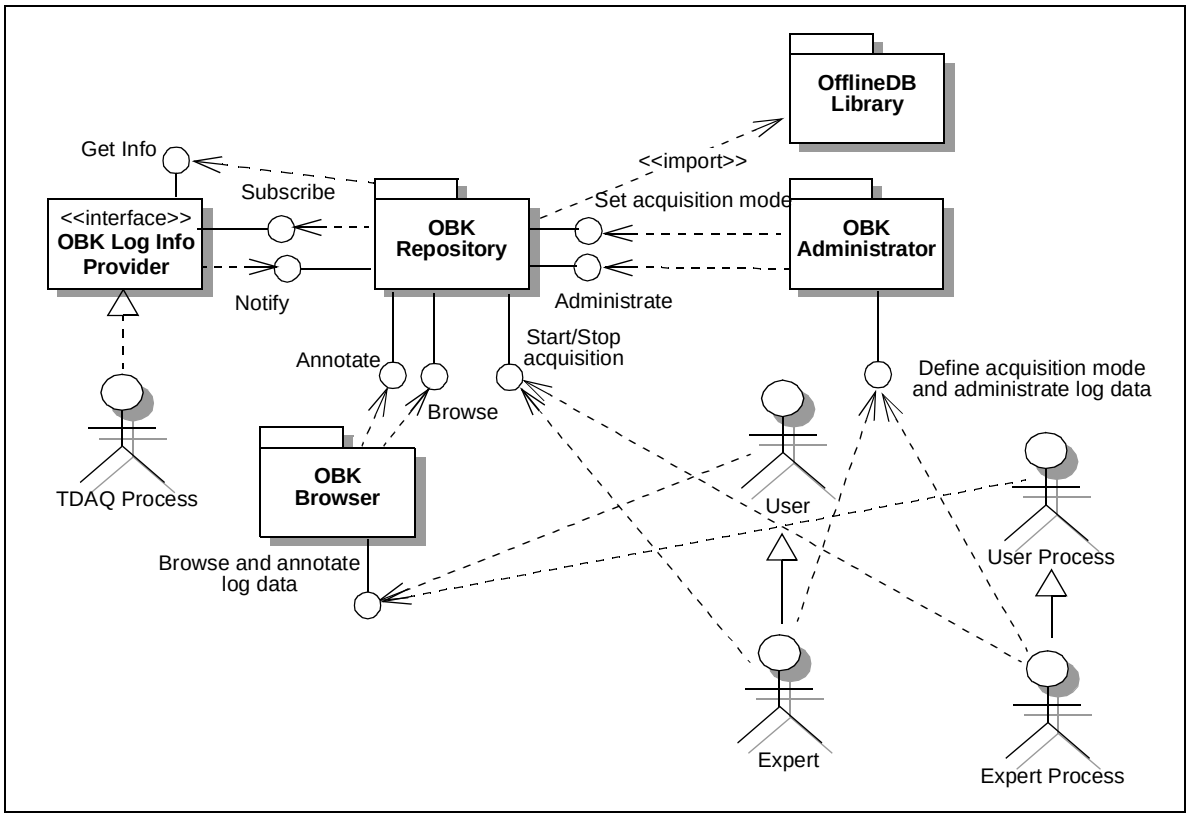

Figure 10-11 OBK users and interfaces

The OBK uses as persistency backbone the Conditions and Time-Varying offline databases services. In this sense, 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 request log data from the several TDAQ processes providing them. It allows a human or system actor to start or stop the acquisition of log data. In order to become a log data provider a TDAQ application will have to realize 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, OBK's 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 to the ATLAS offline conditions and time-varying databases and their architecture are not specified at the moment of the document writing. The conditions database interface will add functionality required by TDAQ and detectors users, if it will not be provided by the Offline Software.

10.4.4 Application of databases to the TDAQ sub-systems

Usage of the databases by the other TDAQ systems, concentrating on differences with general use.

Should be provided by TDAQ systems

10.4.5 Prototype evaluation

The main functionalities 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 beams operations, the integration with other TDAQ sub-systems and detectors, and to evaluate the suitability of the chosen technologies for the final system.

10.4.5.1 Scalability and performance tests of the Configuration Databases

The prototype of the configuration databases is implemented on top of the OKS [10-11] persistent in-memory object manager. It allows to store the database schema and data in multiple XML files. Several subsets can be combined to get a complete description. The access to the configuration description can be made via a file system (C++ interface) and via dedicated remote database server built on top of ILU [10-4] and tested for C++ and Java interfaces.

The 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 a common AFS file system have shown 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

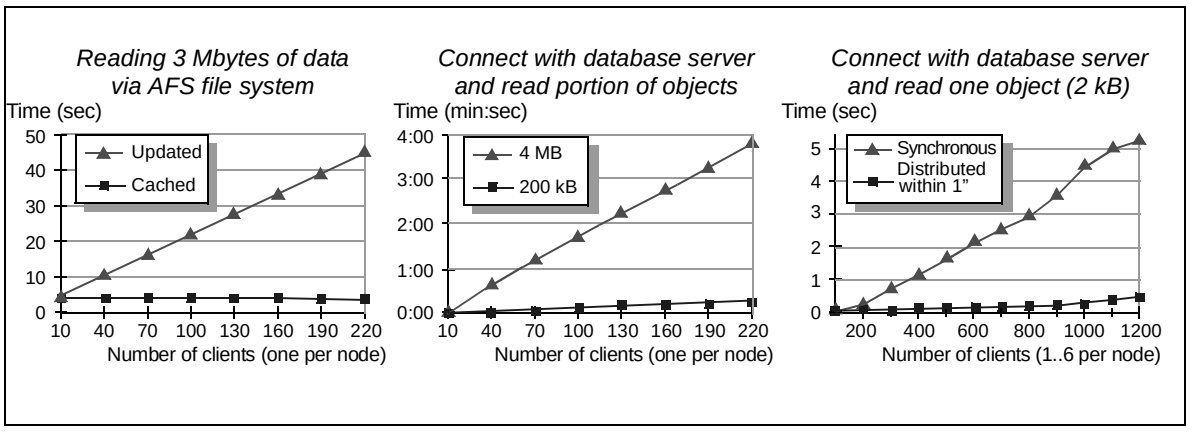

Figure 10-12 Results of the databases performance and scalability tests

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 synchronisation.

The proposed architecture of the configuration databases allows to switch between implementation technologies without affecting user code. Some other database technologies are studied for possible replacement of the ones used in the prototype including relational databases with possible object extensions and the POOL [10-13] LCG [10-14] project.

10.4.5.2 Online Bookkeeper

The prototype of the online bookkeeper was implemented on OKS persistent in-memory object manager and MySQL [10-12] freeware implementation of relational database management system. The results obtained during recent performance and scalability tests [10-9] have shown, that the current MySQL implementation allows to 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 they are described in Chapter 3, "System Operations". It provides the functionality of the TDAQ Control as shown in the controls view of the Chapter 5, "Architecture".

In addition the package has the responsibility for functionalities necessary in the computing environment for user interaction, process management and access control.

10.5.1 Control functionality

Control encompasses software components responsible for the control and supervision of other TDAQ systems and the detectors. The functionalities have been derived from the user requirements and are.

- **User Interaction:** Interaction with the human user like the operator or expert of the TDAQ system
- **Initialization and shutdown:** Operations for the initialisation of TDAQ hardware and software components is foreseen. The operational status of system components must be verified and the initialisation of these components in the required sequence is ensured. Similar considerations are required for the shutdown of the system.
- **Run control:** System commands have to be distributed to a range of several hundred to thousands of clients programs. The control sub-package is responsible for the command distribution and the required synchronisation between the TDAQ sub-systems and detectors.
- **Error handling:** Malfunctions can interrupt system operations or deteriorate 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 recover operations and assist the operator with diagnostic information.
- **Verification of System status:** The control package is responsible to verify the functionality 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-authorised users to corrupt TDAQ functionality.

10.5.2 Performance and Scalability Requirements on Control

The TDAQ system is a large and heterogeneous system composed out of a large number of items to be controlled. These items range from readout modules in VME crates to workstations within HLT computer farms. Typically these items are clustered such that modules are contained within crates or workstations are part of sub-farms. Such units are the preferred places to interface with the Online Software Control system. The number of these units is estimated to be in the range of 500–1000. To control these units the TDAQ control system is build in a hierarchical and distributed manner. More detailed explanation can be found in the chapter on Experiment Control, 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 to 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

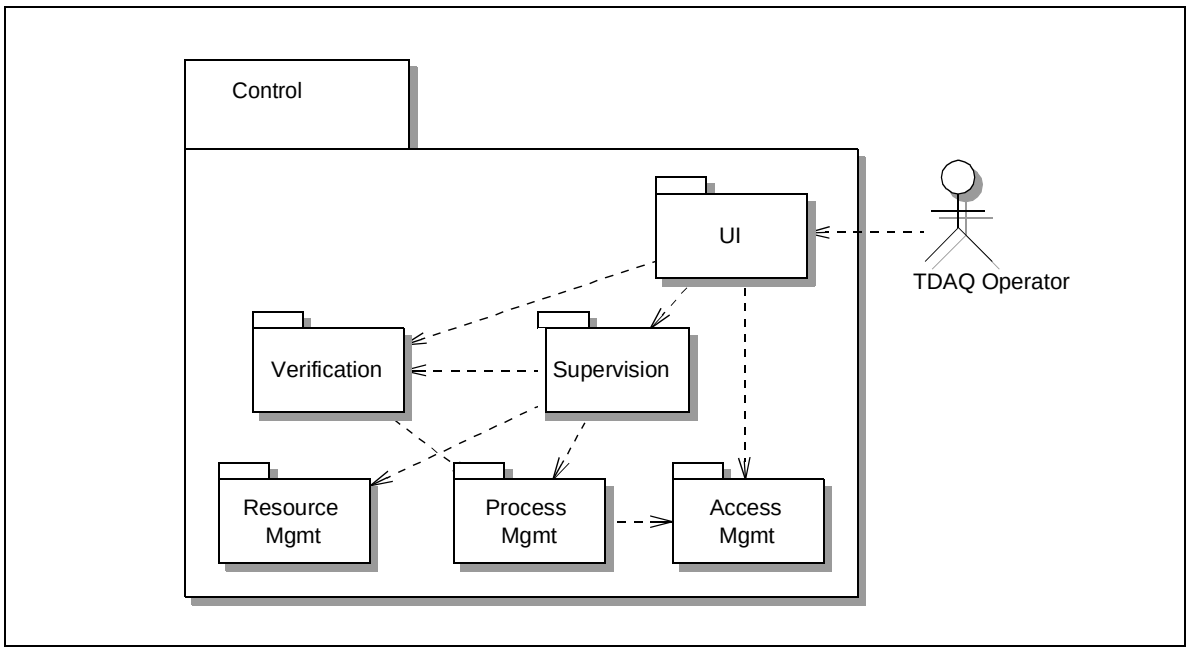

Figure 10-13 The organization of the control package

- 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 authorisation 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. It is foreseen to provide a flexible and extensible UI 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 so-called 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 via its higher level controller. One top level controller called root controller will take the function of the overall control and coordination of the system by interfacing to other TDAQ system or detector specific controllers. Via the User Interface sub-package it provides all TDAQ control facilities to the Operator. These are expressed as interfaces in Figure 10-14 and discussed below in more details.

The *Initialisation and Shutdown* is responsible for

- initialization of TDAQ hardware and software components, bringing 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 functionalities can reside on a so-called *Local Controller* and are extended by specific policies which the TDAQ sub-systems and detector expert developers implement. 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 subsystem.

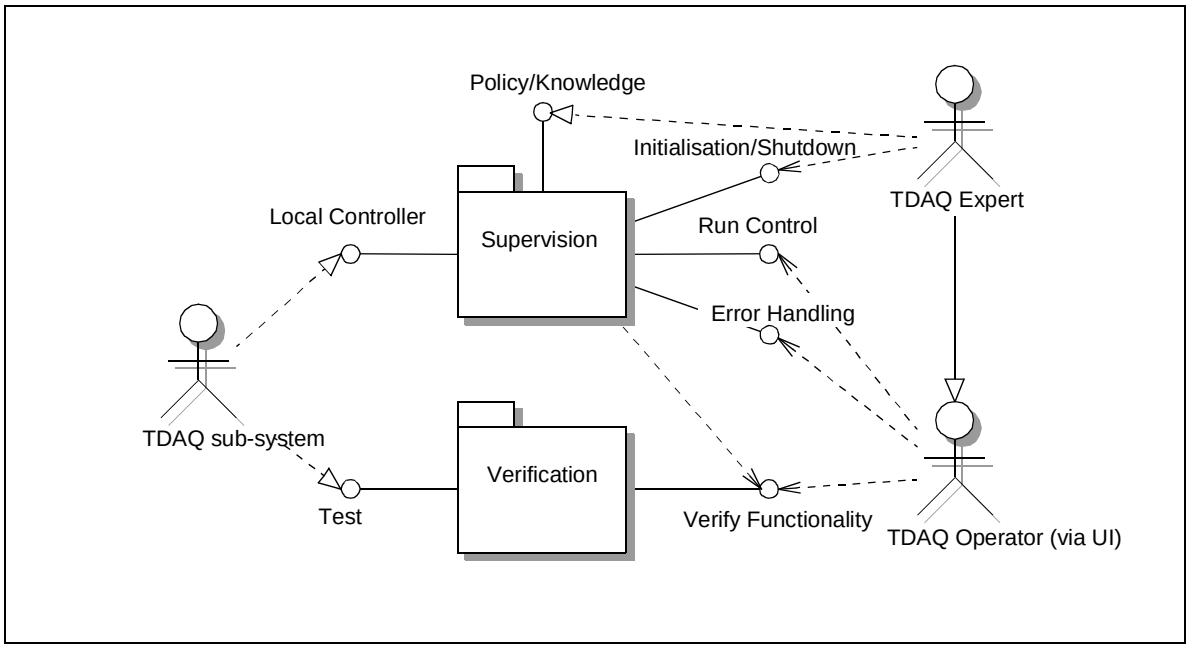

Figure 10-14 Interfaces of the Supervision and Verification sub-packages

10.5.3.3 Verification

The **Verification** sub-package is responsible for the verification of the functionality of the TDAQ system or any subset of it. It uses developer's knowledge to organize tests in sequences, analyse 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 scenario,

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 subsystem provides access to its functionality via the *Verify Functionality* interface. The subsystem is build on specific tests that probe the functionality of the Online Software, TDAQ subsystem 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 it is shown on 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.

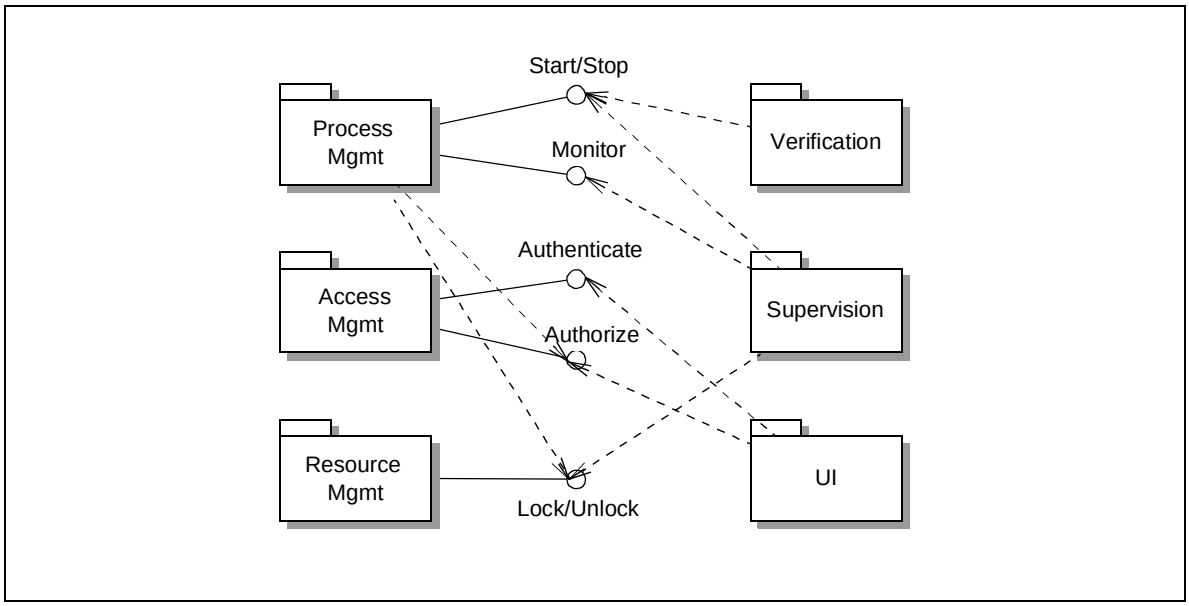

Figure 10-15 Interfaces of the Process Management, Resource Management and the Access Management

The **Process Management** provides basic process management functionality in a distributed environment. This functionality includes starting, stopping and monitoring processes on different TDAQ hosts.

The **Resource Management** is concerned with the allocation of software and hardware resources between running partitions. It prevents the operator from performing operations on resources which are allocated to other users.

The **Access Management** is a general Online Software safety service, responsible for TDAQ user authentication and implementation of an access policy, in order to disable non-authorised persons to corrupt eventually TDAQ functionality.

10.5.4 Prototype evaluation

Prototype evaluations have been performed for a number of technologies. The initial choice was based on experiences in previous experiments. Products were chosen, that fit well in the envisaged object oriented software environment.

- A Run Control implementation is based on a State Machine model and uses the State Machine compiler CHSM [10-15] as underlying technology.
- A Supervisor is mainly concerned with process management. It has been build using the Open Source expert system CLIPS [10-16].
- A verification system (DVS) performs tests and provides diagnosis. It is equally based on CLIPS.
- A Java based graphical User Interface (IGUI) is in use.
- Process Management and Resource Management are implemented 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 could be controlled reliably while meeting the performance requirements.

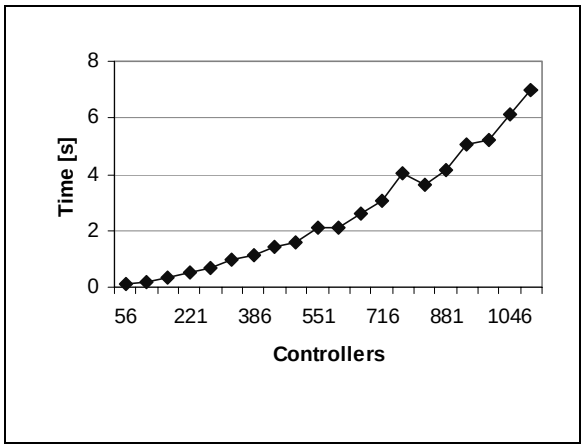

Figure 10-16 Time to perform three TDAQ state trannsitions for configurations in the range of O(10) to O(1000) controllers

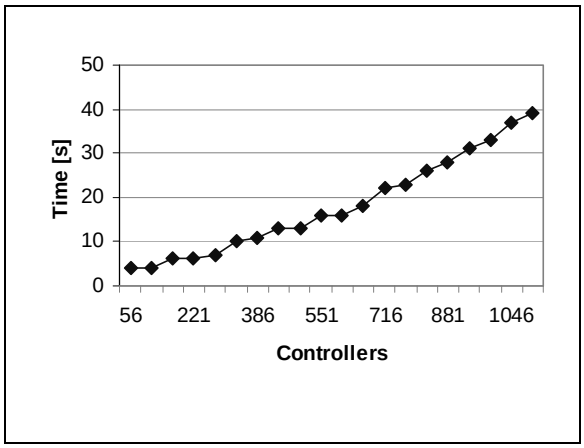

Figure 10-17 Time to start the processes for configurations in the range of O(10) to O(1000) controllers

One item of interest 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 from the overall online control system to the specific tasks to be performed in the TDAQ system. The time was measured to distribute a command from the top level and to obtain the confirmation from all controllers. Figure 10-16 presents the time taken to perform three successive state transition 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 world system each controller would perform its specific actions during such a state transition taking between a few seconds and tens of seconds. In relation to these values the overhead of the overall control system is small.

A second item of interest 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. While 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 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 have been identified. An important aspect is the lack of flexibility in the state machine implementation CHSM. In this context the expert system CLIPS [10-16] and related products have been studied. The general purpose nature of this product allows to implement the various aspects of the supervision-like initialization, control and error handling. The knowledge base provides the basis for customizable solutions, that can be specialized for different parts of the system. Another advantage is the extensibility of CLIPS. It can easily be interfaced with other components of the Online Software system. Alternative products also under consideration are Jess [10-17], a similar expert system implementation written in Java and a commercial alternative, Eclise by Haley. Inc. [10-18].

Further alternatives have been investigated: SMI++ is a system that models the controlled domain as a system of interacting state machines [10-19] and is in use at several HEP experiments. Another possibility would be the use of general purpose scripting languages, as Python [10-20]. 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 have been 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 Mbyte 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 to study the system performance aspects in detail. A number of performance measurements for the Online Software components have also been performed as a function of the system size. Details on those can be found in the component test descriptions on IS in Section 10.3.5.2 and on Databases in Section 10.4.5.1.

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 have been 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 subsystems were studied. Various special purpose controllers and infrastructure configurations were prepared to help identifying eventual shortcomings.

Different types of limitations were found in consecutive test cycle phases. After first having overcome limitations due to design or implementation flaws, limits build 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 lead 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 already the current Online Software system 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 additional safety margins. Furthermore, error handling and fault tolerance aspects will have to be extended and the user interface adapted.

10.6.2 Binary tests with the Online Software and the Event Filter software

will be written by the colleagues from HLT

10.6.3 Deployment

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 has been gathered and has lead to improvements in design and implementation. Details are described in [10-21].

10.7 References

10-1 *Online Software Requirements*,
http://atlas-onlsw.web.cern.ch/atlas-onlsw/documents/documents_page.htm

10-2 *Online Software Architecture*,
http://atlas-onlsw.web.cern.ch/atlas-onlsw/documents/documents_page.htm

10-3 CORBA home page, http://www.omg.org/corba/

10-4 ILU home page, ftp://ftp.parc.xerox.com/pub/ilu/ilu.html

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 Test Report of Large Scale and Performance tests, January 2003, in preparation

10-10 References to external documents on used or evaluated technology

10-11 OKS User's Guide, ATLAS DAQ TN # 033
latest version available from http://atddoc.cern.ch/Atlas/DaqSoft/components/configdb
http://atddoc.cern.ch/Atlas/DaqSoft/components/configdb/docs/oks-ug/2.0/pdf/OksDocumentation.pdf

10-12 http://www.mysql.com/

10-13 POOL = Pool Of persistent Objects for LHC; http://lcgapp.cern.ch/project/persist/

10-14 LCG = LHC Computing Grid Project; see http://lcg.web.cern.ch/LCG/

10-15 P.J.Lucas, *CHSM, An Object Oriented language system for implementing concurrent hierarchical finite state machines*, Thesis, University of Illinois.

10-16 CLIPS, A tool for building expert systems, http://www.ghg.net/clips/CLIPS.html

10-17 Jess, The Expert System Shell for the Java Platform, http://herzberg.ca.sandia.gov/jess

10-18 Eclipse, Rule based programming language and inference engine, http://www.haley.com

10-19 SMI++, State Model Interface, http://cern.ch/smi

10-20 Python, http://www.python.org

10-21 Atlas TDAQ Online Software, *Systems Integration and Deployment in Test Beam and Large Scale Tests*, http://atlas-onlsw.web.cern.ch/Atlas-onlsw/

11 DCS

We consider that the level of detail presented in the chapter, as well as its structure should be respected. However, there we expect changes in the wording and in the logical connections between the different sections.

11.1 Introduction

The principle task of DCS is to enable the coherent and safe operation of the ATLAS detector. Safety aspects are treated by DCS only at the least severe level. All actions initiated by the operator and all errors, warnings and alarms concerning the hardware of the detector are handled by the DCS. Concerning the operation of the experiment, an intense interaction with the DAQ system is of prime importance.

+ Description of what is contained in this chapter and what it has been presented in chapters 1, 2, 3 and 5 (These chapters are needed before).

11.2 Organization of the DCS

The architecture of the DCS and the technologies used for its implementation are strongly constrained by environmental and functional reasons. The DCS consists of a distributed Back-End (BE) system running on PCs, which will be implemented with a Supervisory Control And Data Acquisition system (SCADA), and of the different Front-End (FE) systems.

The DCS FE instrumentation consists of a wide variety of equipment, from simple front-end elements like sensors and actuators, up to complex computer systems that are connected to the SCADA stations by means of standard fieldbuses. A SCADA run-time database contains records of all equipment where the data values are stored.

The equipment of the DCS will be geographically distributed in three areas as shown in Figure 11-1: the main control room at the surface of the installations, the underground electronics rooms USA15 and the detector's cavern, UX15. The SCADA component will be distributed over the two first locations while the front-end equipment will be placed in USA15, US15 and UX15 as shown in figure XXX.

The Front-End (FE) electronics in UX15 (see figure 1.3 ???) is exposed to radiation and to a strong magnetic field up to 1.5 T. The instrumentation in the cavern must be radiation-hard or tolerant to levels of 1–10^5 Gy per year in the muon subdetector and inner tracker, respectively. In the following, only the DCS FE equipment located outside of the calorimeters in ATLAS where the dose rate is of the order of 1 Gy / year will be addressed.

The equipment in the detector's cavern will be interfaced via fieldbuses, to FE computer equipment located in the underground electronics room USA15 and US15. The former is accessible during operation and whereas the latter, will not be accessible during operation due to the remaining radiation levels. The equipment at this level will consist of workstations for subdetector experts for the supervision of individual partitions, mainly during commissioning and maintenance periods. It is foreseen to use a dedicated control computer for each detector. These systems are called Subdetector Control Stations (SCS) in figure *XXX Figure needs to be changed*

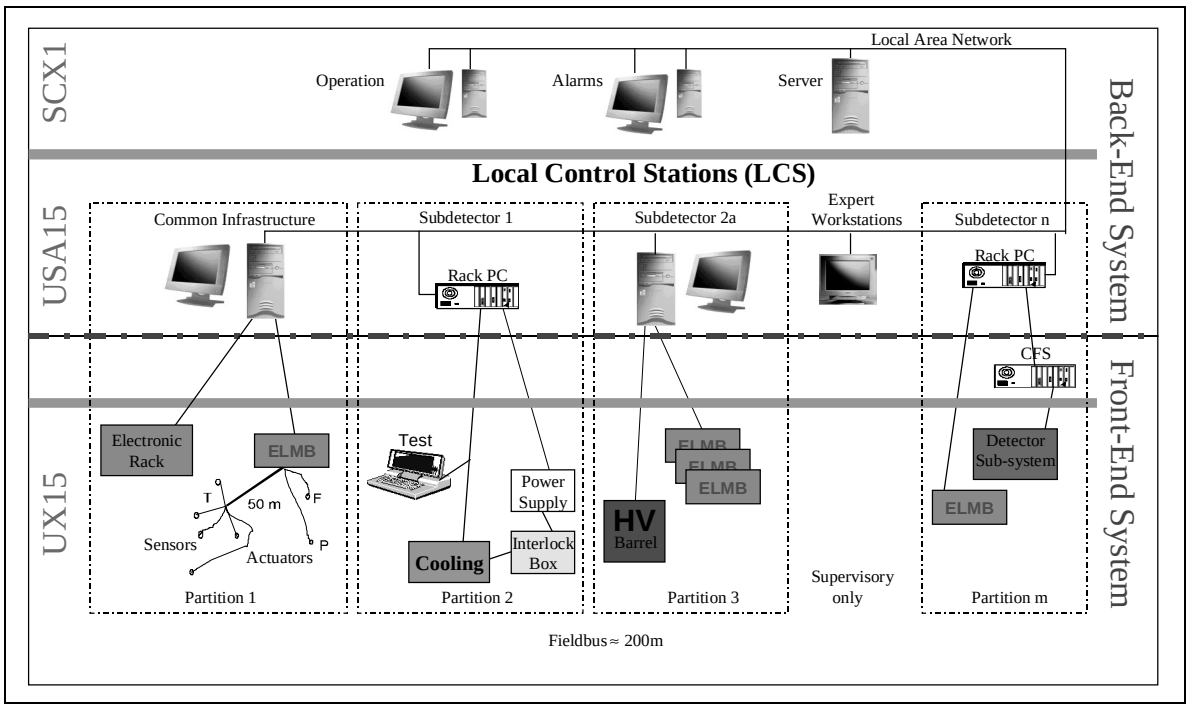

Figure 11-1 Geographical deployment of the DCS.

XXX and they will run the SCADA software collecting data from the front-end devices in their partition.

The Complex Front-end Systems (CFS) at this layer will not run SCADA and will be dedicated to specific tasks. The CFS are normally connected to SCS over a dedicated Local Area Network (LAN). CFS can also be placed in UX15 if they support the hostile environment.

The main control room will be located in the SCX1 building at the surface of the installations. This area will always be accessible to the personnel. The equipment will consists of general-purpose workstations, which will be linked to the control layer through a LAN providing TCP/IP communication. The workstations will retrieve information from the SCS of the different subsystems underneath and can be used to interact with them by means of commands or messages.

The DCS can be partitioned into vertical slices as shown in figure XXX (ref to the picture below). Such a partition can be operated completely independent from other slices of the DCS and offers full SCADA functionality to its users. A vertical slice controls a subsystems of the ATLAS detector, where a subsystem is defined as an arbitrary part of the detector, e.g. the high voltage system of a subdetector or the subdetector itself.

11.3 Front-End System

The front-end equipment consists of controllers, which connect to the hardware, either as separate modules or as microprocessors incorporated in the front-end electronics. Field instrumentation like sensors and actuators will be of various types and it will be tributary to the requirements for the detector hardware.

This equipment is distributed over the whole volume of the detector with cable distances up to 200 m. The distribution underground is governed by two conflicting constraints. Because of the

radiation level, magnetic field, limited space available for equipment and the inaccessibility at UX15 during beam time, the equipment should be located in USA15. However, complexity, cost and technical difficulties of cabling suggest condensing the data in UX15 and transferring only the results to USA15.

The harsh environment limits the types of technologies that may be used. CERN recommends a small number of fieldbuses and ATLAS has chosen the CAN fieldbus as the main standard for this area. The CAN fieldbus is reliable, may be used over large distributed areas, does not use magnetic sensitive components and has good support from industry. To ensure successful operation of equipment in the cavern, the 'ALTAS Policy on Radiation Tolerant Devices' (see [11-2]) has been formed to give specific rules concerning testing and qualification of radiation tolerant electronics.

11.3.1 Embedded Local Monitor Board (ELMB)

Due to the harsh environment, the large number of channels required and the low cost required per channel, there is no commercial solution which exists. Therefore, a general purpose IO device has been developed called the ELMB.

The ELMB is a single, credit card sized PCB which may be embedded onto custom front-end equipment or may be used in a stand-alone mode. It has been designed with low power consumption so that it may be powered remotely via the fieldbus. It provides 64 high-precision analog input channels each of 16-bit accuracy. As well as the analog inputs, there are 8 digital input lines, 8 digital output lines and 8 configurable (either input or output) digital lines. Other interfaces are available such as a serial port allowing JTAG or other protocols to be implemented.

The standard software that is pre-loaded into the ELMB allows for communication over a CAN field bus using the higher level protocol CANopen. The standard functionality gives 'plug and play' usage for the analog inputs and digital inputs and outputs.

The ELMB fulfills the majority of requirements of the ATLAS subdetector applications in terms of accuracy, stability and functionality. It has been tested and qualified to operate in radiation and a magnetic field giving long term operation without manual intervention.

11.3.2 Other FE equipment

All four experiments in the LHC have similar requirements for front-end equipment. JCOP provides solutions and support for standard devices such as high and low voltage power supplies, PLCs and other common front-end equipment, as well as their interfaces into the SCADA system. ATLAS will use many of these standards as recommended for the four LHC experiments.

Other non-standard FE equipment has to be mentioned here: e.g. aligment and calibration systems.

11.4 The Back-End System

The functionality of the BE system is two-fold: It acquires the data from the front-end equipment and it offers supervisory control functions, such as data processing, dysplaying, storing and archiving. This enables the handling of commands, messages and alarms.

The BE system will be hierarchically organized to map the natural partitioning of the experiment into subdetectors, systems and subsystem. The BE hierarchy allows for the dynamic splitting of the experiment into independent partitions, which can be operated in stand-alone or integrated modes. The operation of the different subdetectors will be performed by means of Finite State Machines (FSM), which will handle the states and transitions of the different parts of the DCS. It is envisaged to have a FSM per subdetector. The coordination of the different partitions will be performed by means of commands and messages. The command flow is downwards, whereas the message exchange take place in either direction the within the slice. No horizontal communication is foreseen between different slices or amongst the components of an slice.

11.4.1 Functional Hierarchy

In order to provide the required functionality the BE of the DCS will be logically organized in three levels as shown in Figure 11-2. The actions on the operator time-scale are performed at the upper level, while the RT operations are performed at the lower levels. In addition, data and alarm archiving and logging of incidents and commands will be provided at each of these levels. 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.

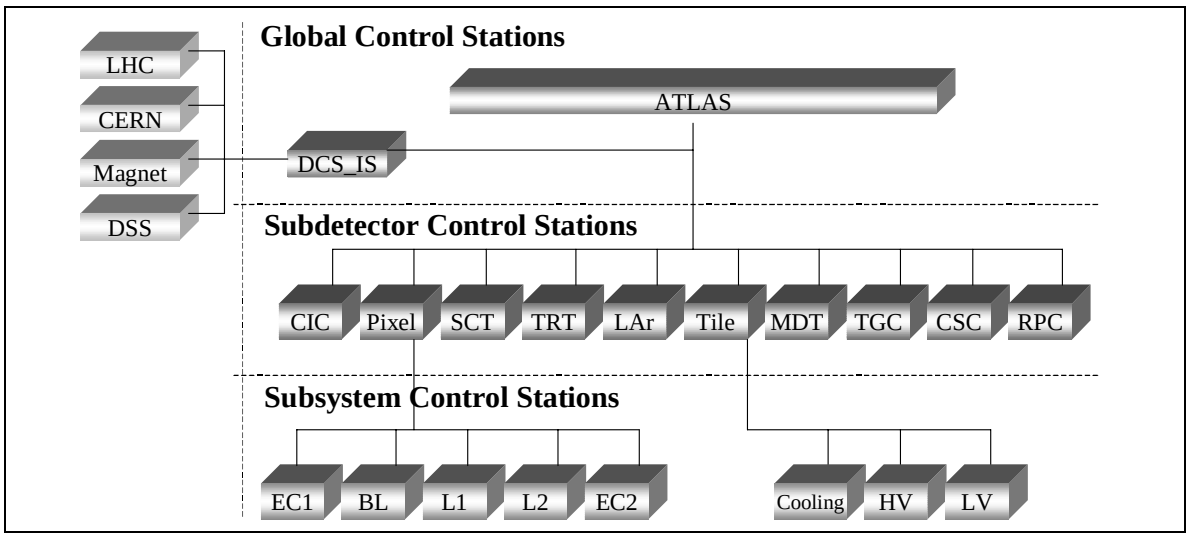

Figure 11-2 Hierarchical organization of the Back-End system of the DCS in three functional layers

Global Control Stations

The overall control of the detector will be performed by the uppermost level of the BE system, which consists of the Global Control Stations. These stations are envisaged to provided high level monitoring and control of all subdetectors and technical infrastructure. The full control of

the detector is provided at only lower levels in the hierarchy. At this level, different services will be provided like the DCS Information Service to handle the communication with the external systems, namely the LHC accelerator, the CERN infrastructure and the Detector Safety System, or web and database services. Information for these subsystems will be used to build the overall status of the experiment. Bidirectional data exchange between the DCS and the TDAQ system will also be managed at this level. No command exchange between the TDAQ and the DCS is foreseen at this level.

Subdetector Control Stations

The Subdetector Control Stations are placed at the intermediate of the BE hierarchy. There will be one SCS per subdetector and an additional SCS to handle the monitoring of the common infrastructure in ATLAS called Common Infrastructure Controls (CIC). The later will be interfaced with the DCS Information service in the layer above. All actions on a given subdetector provided at the Global Control Stations are also provided at this level. In addition, the SCSs allow for the full and stand-alone local operation of the subdetector by means of dedicated graphical interfaces. The SCSs also handle the communication with the services of the layer above. It is foreseen to have a direct connection from the SCSs to the DCS Information service to provide the different SCSs with the status of the external system, namely the LHC accelerator, the Detector Safety system, CERN services and the ATLAS magnet, as well as with the eviromental parameters of the common infrastructure. The SCS handle the co-ordination of all subsystems underlying in the layer below and are the responsible for the validation commands issued by the operator from the global control stations in the layer above or directly from the TDAQ run control. If low level control is required by the issued actions, e.g. ramp up high voltage, these commands can be propagated to the subsystems in the layer below for their execution. The overall status of the subdetector is assembled (collated???) and pass on to the TDAQ system via the DAQ-DCS communication software, which is described in Section 11.8, and to the control stations in the layer above.

Subsystem Control Stations

The bottommost level of the BE hierarchy is constituted by the Subsystem Control Stations, 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 detector could be performed attending to either geographical or functional criteria. In the former the arrangement follows the natural partitioning of the detector in sections, subsection, etc. whereas in the second approach, the organization is decided as a function of the different services of the subdetectors, e.g. cooling, high-voltage, etc. This 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 station placed at this level will executed the commands propagated from the SCSs in the layer above although they can also execute predefined automatic actions if required.

11.4.2 SCADA

The BE system of the ATLAS DCS will be implemented using a Supervisory Control And Data Acquisition (SCADA) product. SCADA systems [37] are commercial software packages normal-

ly used for the supervision of industrial installations. They gather information from the hardware, process the data and present them to the operator. Even though SCADA products are not tailored to LHC experiment applications, many of them have a flexible and distributed architecture and, because of their openness, are able to fulfil the demanding requirements of the ATLAS DCS.

Besides basic functionality like the Human Machine Interface (HMI), alarm handling, archiving, trending or access control, SCADA products also provide a set of interfaces to hardware, e.g. CERN recommended fieldbuses and PLCs, and software, e.g. Application Program Interface (API) to communicate with external applications, or connectivity to external databases via the Open or Java DataBase Connectivity (OBDC and JDBC respectively) protocols.

SCADA products constitute a standard framework to develop the applications leading to a homogeneous DCS. Its usage saves development effort reducing the work for the subdetector teams. In addition, they follow the evolution of the market, protecting against changes of technology like operating system or processor platforms.

11.4.3 PVSS

A major evaluation exercise of SCADA products [38] was performed at CERN in the frame of the Joint COntrols Project (JCOP), which concluded with the selection of the PVSS-II, from the austrian company ETM, to be used for the implementation of the BE systems of the four LHC experiments.

PVSS is a device-oriented product where process variables that logically belong together are combined in hierarchically structured data-points. Device-oriented products adopt many properties from object-oriented programming languages like inheritance and instantiation. These features facilitate the partitioning and scalability of the application.

PVSS provides the interfaces to connect to external databases or systems, like the DAQ system or LHC accelerator, and the capability to extend the functionality of the product to interface custom applications or equipment (e.g. availability of driver development toolkit).

It is conceived as distributed systems. The single tasks are performed by special program modules called managers. The communication among them takes place according to the client-server principle, using the TCP/IP protocol. The internal communication mechanism of the product is entirely event-driven. This characteristic makes PVSS specially appropriate for detector control since, systems which 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 PC running either Microsoft Windows or Linux. The communication between them is internally handled by PVSS-II. This has been one of the crucial points in the selection of this product since the DAQ system of the ATLAS experiment is been developed entirely under Linux, where as the DCS will widely use Windows.

PVSS allows to split the supervisory software into small application communicating over the network and it is imposed by the distribution of the DCS equipment in different locations in ATLAS.

11.4.4 PVSS Framework

Although PVSS-II will be used as the basis of the LHC experiment controls, this has been found not to be sufficient to develop an homogeneous and coherent system. An engineering framework on top of PVSS-II is being developed in the frame of JCOP. *XXXHas JCOP been mentioned before?XXX* The PVSS framework is composed of a set of guidelines, tools and components commonly used by the four LHC experiments like HV and LV systems. This framework will lead to a significant reduction of the development and maintenance work to be performed by the subdetector teams and to an homogeneous system by means of the standardization of the equipment utilized. It also addresses the interoperability of the different components included in the framework. The ELMB has been integrated into PVSS as a component of JCOP framework with the aim of facilitating the usability of the ELMB node to the ATLAS users but also to ensure the homogeneity of the SCADA software. The ELMB component provides all PVSS infrastructure needed to work with the ELMB in a standard mode. This package also comprises a so-called a 'top-down' configuration tool, which handles the configuration of all non-SCADA interfaces to the FE system.

All DCS systems from the global level to the local control stations will need some commonly used services. These applications will be implemented once and may be used at all levels. The main services to be provided will be for data and alarms, where both numerical displays and trending (with the native widgets and external trending tools) will be used, web based presentation of information and logging of all actions whether performed by an operator or an automatic process.

11.5 Integration FE-BE

The PVSS-II product will be used as SCADA system for the implementation of the supervisor layer of the ATLAS/DCS. There are several interfaces which allows to connect PVSS-II based systems to hardware.

- Dedicated drivers for PVSS-II; PVSS-II has the drivers for modbus devices, PROFIBUS and some others. It also contains the API to develop drivers by users.
- PVSS-OPC client; OPC is a wide used industrial standard. All commercial Low and High voltage systems are supplied with the OPC servers.
- DIM software, which is a communication system for distributed and multi-platform environments. DIM provides a network transparent inter-process communication layer developed at CERN.

The OPC due to the wide spread usage and the big support from industrial has been chosen as main interface from the SCADA to hardware devices. The main purpose of this standard is to provide the standard mechanism for communicating to numerous data sources. The OPC is based on the Microsoft Windows technology. The specification of this standard describes the OPC Objects and their interfaces implemented by OPC server. The architecture and specification of the interface was designed to facilitate clients interfacing to remote server. An OPC client can connect to more then one OPC Server, in turn an OPC Server can serve several OPC clients. All OPC objects, consequently, are accessed through interfaces.

In turn the ELMB is a CANopen device and will be widely used in the implementation of the subdetector front-end system.

CANopen is a high level protocol for the CAN-bus communication. This protocol is widespread as well. CANopen standardizes the types of CAN-bus messages (objects) and defines the sense of them. It allows to use the same software in order to manage of CAN nodes of different types and from different manufacturers.

To connect the ELMB to SCADA the OPC CANopen server has been develop. Others possibilities will also be used in suitable cases.

11.5.1 OPC CANopen server

On the market there are a lot of the CANopen servers. But all of them are tailored to their specific hardware interface cards and they do not provide the CANopen functionality required by the ELMB.

The OPC CANopen server works as the CANopen master of the bus handling network management task, node configuration and transmitting data to the OPC client. The OPC CANopen Server consists of two main parts.

- The main part is an OPC server itself. It implements all the OPC interfaces and main loops. Any application interacts with this part through interfaces. The CANopen OPC server transmit data to a client only on change, which results in a substantial reduction of data traffic.
- The second part ‘Can Bus component’ is hardware dependent. It interacts with a CAN bus driver and controls CANopen devices.

Several busses with up to 127 nodes each, in accordance with CANopen protocol, can be operated by the OPC CANopen server. The system topology in terms networks and nodes per bus is modelled in start up time in the address space of the OPC CANopen server from a configuration file.

11.6 Readout chain

The complete readout chain ranges from the I/O point (sensor or actuator) to the operator interface and is composed of the elements described above: ELMB, CANopen OPC Server and PVSS-II. Apart from the data transfer, it also comprises tools to manage the configuration and the settings and status of the bus. PVSS-II models the system topology in term of CANbus, ELMB and sensor in the internal database by data-points. These data-points are connected to the corresponding items in the OPC server in order to send the appropriate CANopen message to the bus. In turn, when an ELMB sends a CANopen to the bus, the OPC server will decode it, set the respective item in its address space, which transmits the information to a data-point in PVSS. The different elements of the readout chain perform the following functions.

The ELMB digitises the analogue 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 automaticly in predefined intervals, or when they have changed. As the ELMB is in most cases exposed to ionizing radiation, it 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 when they have changed. It can optionally perform calculations, e.g. to convert 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. The SCADA also archives the data and allows its visualisation.

11.6.1 Performance of the DCS readout chain

To investigate the performance and the scalability of the DCS readout chain to the size required by ATLAS, a full vertical slice (or full branch) consisting of 6 CANbuses having 32 ELMBs each was assembled.

The aim of this test was to study the behavior of the system with these characteristics to discover settings required in order to achieve the optimal results and to establish limits of the readout chain. These limits define the granularity of the system in terms of number of ELMB per CANbus and the number of buses per PVSS system. In particular, the following was to be inspected and investigated:

- Remote powering of the ELMB nodes via the bus. The radiation levels in the detector cavern impose that the power supplies will have to be placed in the underground electronics rooms US15 and USA15. Therefore, the power for the nodes will have to be fed remotely via the CANbus with distances up to 150 m.
- Bus loading, which determines the number of nodes per bus. The data traffic on the bus has to be uniformly distributed over time in order to keep the bus load low under normal operation. In ATLAS the bus occupancy will be kept below 60% in order to cope with a higher loads which may arise in case of problems like power cuts. In these cases, an avalanche of channel information which must be handled by the system.
- Optimization of the work balance amongst the different processing elements in the readout chain. The functions to be performed by the ELMB, CANopen OPC server and PVSS are homogeneously distributed to ensure equal load of each of these components and to avoid bottle-necks.
- Optimization of the system performance by tuning of different software settings such as update rates for OPC and the readout rate.
- Determination of the overall performance of the systems, which defines the number of CANbuses with these characteristics per PVSS system, and that will strongly condition the topology of the different subsystems.

The setup employed in the test, shown in figure 1. The system of CANbuses was operated from PVSS-II using the CANopen OPC server and a Kvaser CAN interface. The bus lengths were 350 m in all cases, in order to fulfil the ATLAS requirements with a broad margin. Up to 32 ELMB were connected at the end of each CANbus. The total number of channels in this system was: 12288 analog inputs, 3072 digital outputs, and 1536 digital inputs. It is important to note that the amount of channels in the set up described here, is of the order of magnitude of some large applications in ATLAS. During the test, the readout rate was increase in order to push the system to the limit. When big bursts of data arrive at PVSS-II very rapidly, the different messages are 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, in addition, the CPU memory usage remains constant, i.e. no buffering is performed at the PVSS-II or OPC level.
- The so-called *avalanche run*, where the fastest possible readout rate is estimated for a a short period, typically a few minutes. Under these circumstances, although all messages are archived to the PVSS database, the bus data flow is so high, that messages cannot be treated in real time and are buffered at the SCADA level therefore, leading to an increase of the memory usage. It is important to note that although long term operation under these conditions would not be possible, this situation can occur in case of major problems of the equipment monitored, like power cuts, and must be handled by the system.

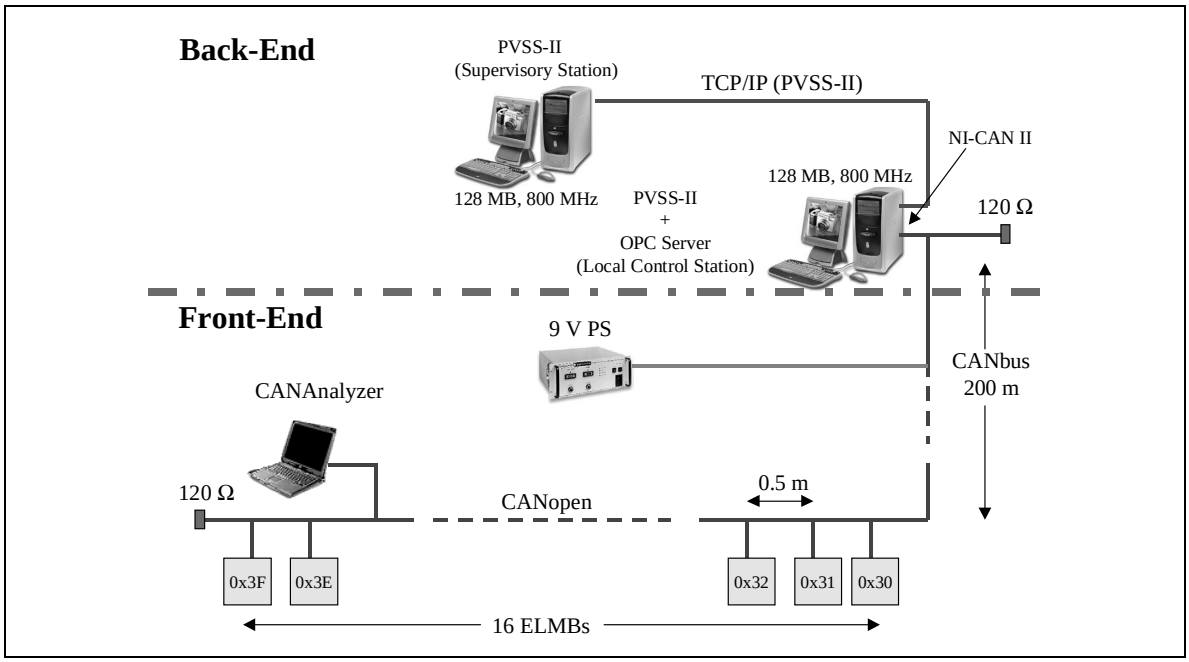

Figure 11-3 ELMB full branch test setup. *(To be changed: Show bus multiplicity)*

A readout rate of at least 30 s is required for a steady run in a system of 6 buses with 32 ELMB nodes each (12288 analog input channels). Under these conditions, the fastest readout rate in avalanche mode is limited to 8 s for several minutes. The examination of the CPU load showed that in all cases, the results presented are only limited by the CPU work load due to the PVSS-II managers as a consequence of the buffering of the CAN messages. These results indicate that the operation of the readout is strongly constrained by the performance of PVSS-II.

11.6.2 Long term operation of the readout chain

Mention as introduction that the DCS has to run 24/365.

The long-term operation of the full readout chain was tested with a number of ELMBs in a radiation environment similar to that expected in the ATLAS cavern, though with a much greater dose rate. This environment allows the different error recovery procedures implemented at the different levels of the readout chain, which are required due to radiation effects to be tested. A CAN bus of greater than 100 m was connected to a PC running the CANopen OPC server and PVSS-II. The test ran for more than two months, which was equivalent to more than 300 years at

the expected ATLAS dose rate in terms of TID. 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 were seen during the test at the ELMB using special test software, and these are also handled at the ELMB level. The OPC server ensures all ELMBs on a bus are kept in the operational state, monitoring the messages on the bus in case of power glitches. At the highest level, PVSS scripts were utilized to monitor the current consumption for the bus (where increase in current is an indication of latch-up or damage from long term TID) and to reset ELMBs if communication has been lost. Through this script, the power supply for the bus was controlled allowing for hard resets to be performed. No user intervention was necessary during the time of the test.

11.7 Applications

The components and tools described above are used to build the applications, which control and monitor the experiment equipment. The applications for the supervision of the subdetectors is the responsibility of the subdetector groups and is described in the relevant TDRs. All equipment, which does not belong directly to a subdetector will be supervised by a SCS called Common Infrastructure Controls (CIC), which is hierarchicaly situated at the level of a subdetector. It monitors the subsystems described below.

All racks, both in the elecronics rooms and in the cavern will comprise a control unit based on the ELMB. It monitors the basic operational parameters like temperatures, air flow, cooling parameters, etc. and also user-defined parameters. Some racks will have the option of controlling the electric power distribution. The crates which are housed in these racks have however usually 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 belong also this category. The individual subdetector cooling distribution systems are however supervised by the corresponding subdetector SCS.

All information collected is avalaible to all other PVSS stations via a central DCS information server. A subset of them will also be transmitted to the DAQ.

11.8 Connection to DAQ

In order to grant a coherent functioning of both DCS and the physics data acquisition the following functionality of communication between that systems is to be provided [11-6, 11-7]:

- Bi-directional exchange of data like parameters and status values;
- Transmission of DCS messages, like alarms and other error messages, to DAQ;
- Synchronization DCS with TDAQ run control and providing ability for DAQ to issue commands on DCS (with feedback).

In accordance of the concept of TDAQ partitioning [11-10] the communication functionality required should be provided for each needing it TDAQ partition independently of others.

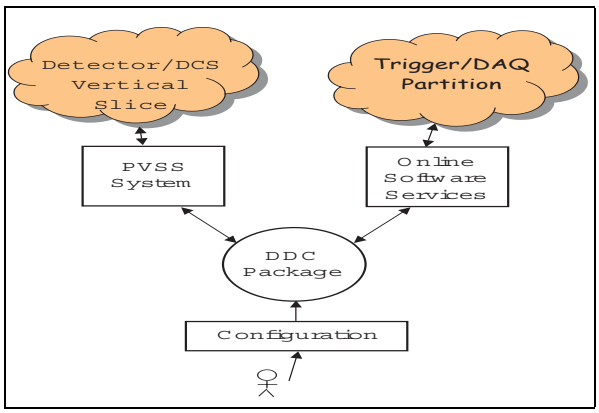

Figure 11-4 DDC package in relation with the DCS and the Trigger/DAQ system.

The TDAQ Online software package, described in the previous chapter, provides a series of services for Trigger / DAQ inter-application communications of the content declared above for DCS communication. They are the Information Service (IS) allowing to share the run time information , Error Reporting Service (ERS) distributing application messages (Section 10.3.3.2) and the Run Control package (Section 10.5 ??) running a finite state machine to represent and control / synchronize the states of TDAQ subsystems of a partition. These services / subsystems will be used as the DAQ-side connection points for the communication with DCS.

The PVSS II product (Section 11.4.3) is provided with a powerful application program interface (API) allowing full direct network access to the PVSS application runtime database. That API is DCS side law level interface for DAQ – DCS communication.

The DAQ–DCS communication package (DDC) is to be developed on top of the interfaces mentioned above as a generic tool configurable by end-user in terms of DAQ and DCS functionality (i.e., identification of data, messages and commands to be transferred). It provides the co-operation of DCS (PVSS) world and Trigger / DAQ world as it is illustrated in Figure 11-4.

The underlying subsection describe the DAQ – DCS communication software components with their interfaces. The features belonging to all of them:

- Implemented as a PVSS API manager [11-8] integrates the program interface of corresponding Online software service;
- Waits if a communication partner is not running and recovers lost connection;
- Being configured from the TDAQ configuration database – this interface is omitted in figures below.

The prototype of the DDC package has been tried in the test beam experiments of 2002–2003 and demonstrated satisfactory and reliable capability of working.

11.8.1 Data Transfer Facility (DDC-DT)

The data exchange in both directions is to be implemented via the Information Service of DAQ Online software. The application keeps the data elements (parameters of the systems) declared in the DDC-DT configuration being the same in both source and destination of that data. This is done on the base of the subscription mechanism available for both sides. The possibility for

DAQ of requesting single read of specified DCS data is also provided. Figure 11-5 shows the interfaces to be used.

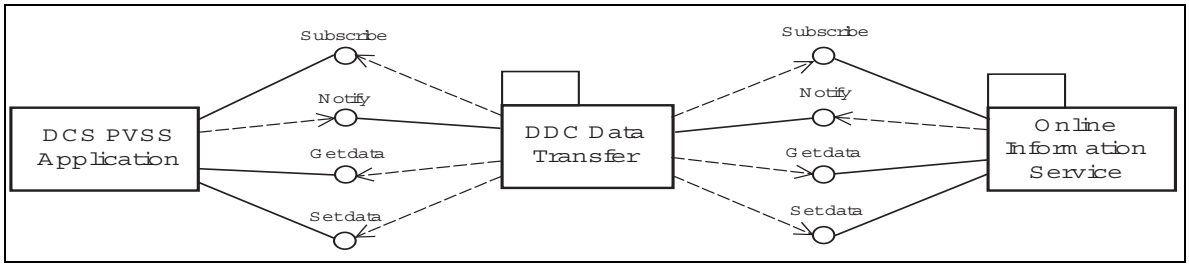

Figure 11-5 DDC data transfer interfaces.

11.8.2 Message Transfer Facility (DDC-MT)

The DCS message transferring to DAQ is to be implemented via the Error Reporting System of DAQ. The interfaces used are drawn in Figure 11-6.

This component delivers for DAQ DCS alarm messages on appearance and DCS text variables to be interpreted as messages for DAQ, which have been defined in its configuration.

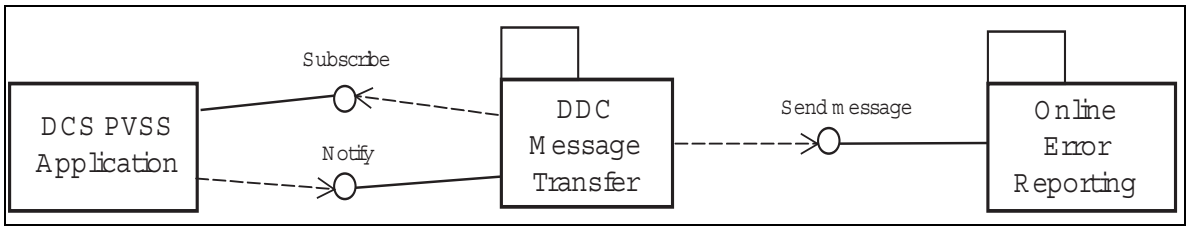

Figure 11-6 DDC message transfer interfaces.

11.8.3 Command Transfer Facility (DDC-CT)

The DDC-CT subsystem (being like other components a PVSS API manager) is implemented as a dedicated run controller (RC) to be included as a leaf into a TDAQ partition run control tree (see Section 12.2 ??). This run controller, like any other TDAQ run controller, is capable to execute standard commands causing its transitions as defined by the TDAQ partition finite state machine. Except of that, it allows sending for DCS so-called *non-transition* commands (***nt-commands***) via the Online software information service. The DDC-CT interfaces are shown in Figure 11-7.

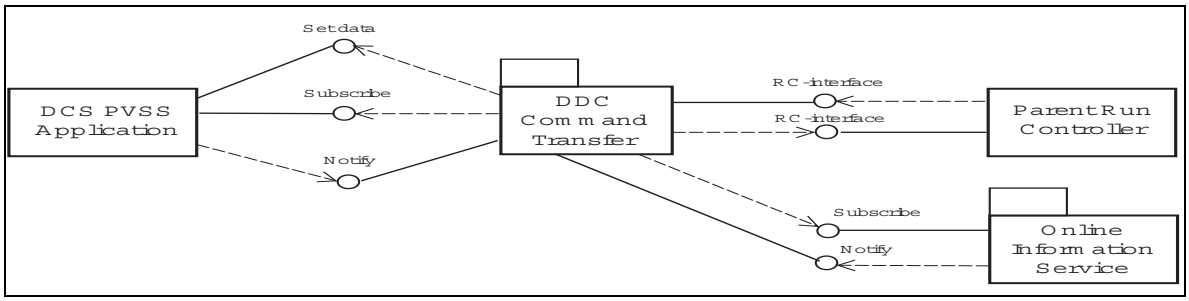

Figure 11-7 DDC command transfer interfaces.

The term *non-transition* emphasizes that those commands do not cause any finite state machine transition. An *nt-command* may be issued by any TDAQ application (including the parent run controller).

The content of a command (both run control and nt-command) and its execution is the responsibility of DCS's PVSS application. Mapping of the run control transitions onto certain set of commands on DCS is to be done in the DDC-CT configuration.

11.9 Interface to External Systems

The term External Systems designates systems having their own control system with which the DCS has to interact. ATLAS will gain access to external systems via the DCS. We can distinguish two main external systems: the CERN technical infrastructure and the LHC accelerator. The former consists of a number of subsystems like cooling and ventilation, electricity distribution, radiation monitoring, etc. The Detector Safety System (DSS) and the Magnet Control System are also considered part of the technical infrastructure. All these external systems must be concurrent into the general DCS. Although these systems are designed to react in case of problems, early indications of their status must be notified to the DCS since they may have consequences onto the detector and automatic corrective actions, driven by the DCS, may be required. The DCS will reflect also the states of all these systems and, in many cases, will act as their user interface.

The connection will support bidirectional information exchange and, in some cases, the sending and receiving of commands. This interface will be unique for the 4 LHC experiments and it will be developed in the framework of the JCOP.

11.9.1 CERN Technical Services

The technical services around the ATLAS detector include cryogenics and conventional cooling, ventilation, gas system, electricity, radiation monitoring, low and high voltage power supplies. The DCS will also have access to the control and status of infrastructure including AC mains, air conditioning etc. These services will monitor the environment to guarantee the safety of the personnel and equipment and will enable the different subdetectors of the experiment to function within their required operating conditions. Some of the systems will need feedback from the subdetector to operate. This is the case for the gas system and cooling, where part of their equipment consist of external stand-alone PLC or commercial I/O modules, whereas some information come from the detectors themselves. Therefore, slow closed-loops maybe needed between the DCS and this type of system.

11.9.2 Detector Safety System

As previously mentioned, the DCS is not responsible for the security of the personal 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 and stand-alone PLC and is the responsibility of the Detector Safety System. Although the information exchange between the DCS and the DSS must be bi-directional, actions must go only in one direction. The DCS must not disturb the operation of the safety system. However, warnings about problems

detected by the safety system must be notified to the DCS in order to take corrective actions or to shut down the problematic part of the detector. Control access will also be handled by the CERN services and it will be needed at the DCS side.

11.9.3 Magnet system

Due to its critical requirements [10????] and complication, a dedicated PLC-based control system will be implemented for the ATLAS magnet. The operator will not need direct control, although a detailed online status and knowledge of all important parameters of the magnets is essential for the operation of the detector and for the subsequent physics analysis. This dedicated system will supervise and control the cryogenics, the cooling system, the power supplies and the instrumentation of the magnet.

11.9.4 LHC

An robust interface between the experiment and the accelerator must be provided. Instantaneous beam parameters like the different types of background, beam position and luminosities observed in the detector must be transferred from the experiment to the accelerator for consequent tuning of the beam.

The experiment will also give all information on its status such as status of its magnets, in particular, the solenoid which acts directly on the beams, status of sensitive equipment like high voltage on the sub-detectors and other status signals as well as global status signals such as the operation state of the detector, setting up, etc. The DCS has to make sure that the detector is in an appropriate state (e.g. voltage settings) before LHC is allowed to inject particles.

ATLAS may need the possibility to request actions like a fast beam dump should the backgrounds become dangerous for the subdetectors or injection inhibit.

On the other hand, machine parameters like status signals for setting up, shut-down, controlled access, stable beams, beam cleaning must be transferred from the accelerator to the experiment. The machine should also provide information on the beam like emittance, focusing parameters, energy, number of particles and a horizontal and vertical profile needed for offline physics analysis. Information on the vacuum conditions in the vicinity of and in the experimental straight section, and position of the collimator are also of interest to the experiment.

The LHC has dedicated instrumentation for the comprehensive measurement of all these parameters. The subset of operational parameters of the accelerator, relevant to the operation of the detector or to the subsequent physics analyses have to be delivered to the DCS and must be logged.

Although the exchange of many of these parameters is only needed during data-taking, a subset of this information, like the integrated radiation doses in the different parts of the detector measured by the DCS, has to be known to the LHC at all times. Therefore, this communication is required regardless the state of ATLAS. This is one main reason why this communication will be handled by the DCS on the ATLAS side and not by the DAQ system.

11.10 References

11-1 New references to be added

11-2 ALTAS Policy on Radiation Tolerant Devices

11-3 H.J. Burckhart et al., *Vertical Slice of the ATLAS Detector Control System*, submitted to 7th Workshop on Electronics for LHC Experiments, September 2001, Stockholm (Sweden)

11-4 F. Varela Rodriguez et al., *ELMB Full Branch Test: Behaviour and Performance*, ATLAS DCS Internal Working Note 13 (2001)

11-5 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)

11-6 http://www.kvaser.com

11-7 V. Filimonov, *Description of the CANopen OPC server v2.5*, http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/ELMB/DOC/OPCCOUserGuide.pdf

11-8 H. Burckhart, M. Caprini, R. Jones, *Connection DCS - DAQ in ATLAS*, ATLAS DCS IWN8, (1999) http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/dcs_daq_0.6.pdf.

11-9 R. Hart, V. Khomoutnikov, *ATLAS DAQ - DCS Communication Software User Requirements Document*, (2000) http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/DCS/DDC/ddc_urd.pdf

11-10 <TDAQ Partitioning document> Probably, made already earlier at the document

11-11 <Reference to PVSS> Probably, made already earlier at the document

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: the TDAQ control and the Detector Control Systemm (Chapter 11). The TDAQ Control is in charge of controlling the hardware and software elements in TDAQ needed for physics data taking whereas the DCS handles the control of the detector equipment and related infrastructure. An architectural overview of the Experiment Control has been introduced in Chapter 5, "Architecture". The architecture of the systems involved has already been discussed in the previous chapters. They have to fulfil different tasks and meet different requirements concerning the experiment control. Whilst the TDAQ control is only required when taking physics or calibration data, during detector commissioning and tests, the DCS has to operate continuously to ensure the safe operation of the detector. The DCS is based on a SCADA system [12-5], while the TDAQ control is based on the TDAQ Online Software described in Chapter 10. The operation of the detector requires a strong coordination of these two systems with the LHC machine. The interaction with the LHC machine will be handled by the DCS as shown in Figure 12-1 and presented in detail in Chapter 11. The TDAQ system has the overall mastership for the control of the data-taking operations.

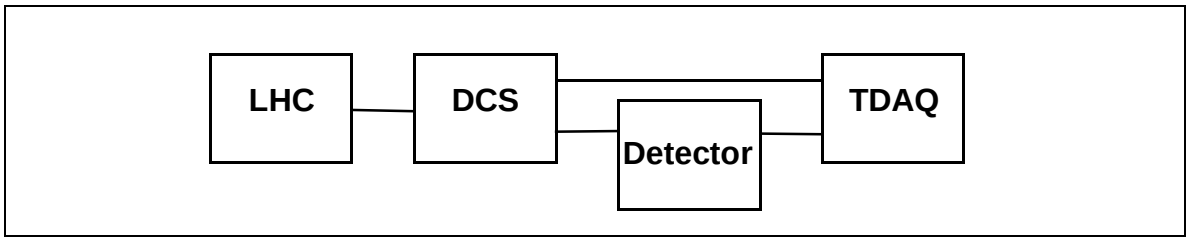

Figure 12-1 Connections between the principal experiment control elements and the detector

The general control of the experiment requires a flexible partitioning concept, 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

In order to provide all the functionality required for physics and calibration runs, the DCS system must provide the flexibility to follow the partitioning schema of the TDAQ system. The finest granularity of the TDAQ system, is given by the segmentation of the sub-detectors in TTC zones. For these reasons, the different sections of the sub-detectors will be logically represented in the back-end software of the DCS by means of the so-called control units, which will be operated as FSM. According to this model, the DCS of the Tilecal, for example, may be organized in

four independent control units, which can control the four sub-detector sections. The control units are hierarchically organized in a tree-like structure to reproduce the organization of the experiment in sub-detectors, DCS system and sub-systems as illustrated in Figure 12-2.

Each control unit may control a sub-tree consisting of other control units or device units, which are responsible for the monitoring and control of the equipment.

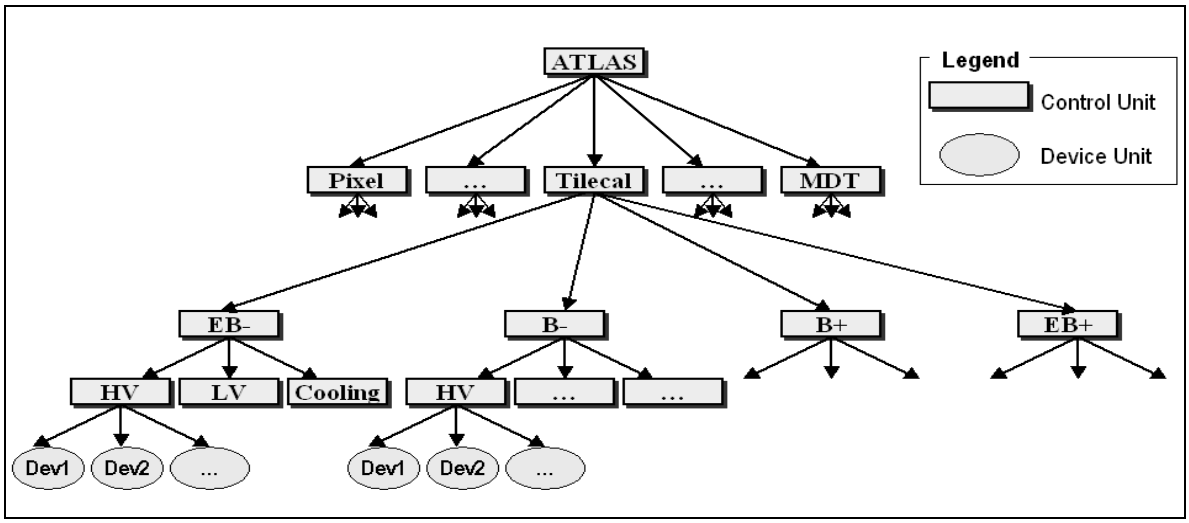

Figure 12-2 DCS Logical Architecture.

In order to map the dynamic structure of the TDAQ system, the control units will support different partitioning modes. Any control unit may be excluded from the hierarchy and operate in stand-alone mode for testing, calibrations or debugging of part of the system. The different modes of operation of the DCS, which allow for stand-alone or integrated mode with the TDAQ system, will require the implementation to handle the ownership of the different control units. This mechanism will be developed according to the recommendations of the JCOP Architecture Working Group [12-1].

12.3 Online Software Control Concepts

The TDAQ system is given by a large number of hardware and software components, which have to operate in a coordinated fashion to provide for the data-taking functionality of the overall system. The organisation of the ATLAS TDAQ system in detectors and sub-detectors leads to a hierarchical organisation of the control system. The basis of the TDAQ control is provided by the ATLAS Online Software, which is explained in detail in chapter 10.5.

The basic element for the control and supervision is a controller. The TDAQ control system is build of a large number of controllers which are distributed in a hierarchical tree following the functional composition of the ATLAS TDAQ detector.

The concept is illustrated in Figure 12-3. In the diagram 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 *root controller* has the overall control over the TDAQ system. It supervises the next level of controllers in the hierarchy, the *detector controllers*. It is the responsibility of the detector controller to supervise the hardware and software components which belong to this detector.

The next control level takes the responsibility for the supervision of the sub detectors which correspond to the TTC partitions [12-4]. On the lowest level the so-called *local controllers* are responsible for the control of readout crates and alike. Farm supervision and ROS hardware make use of the same controllers following a similar structure, which is further discussed in Section 12.3.1 and Section 12.3.2.

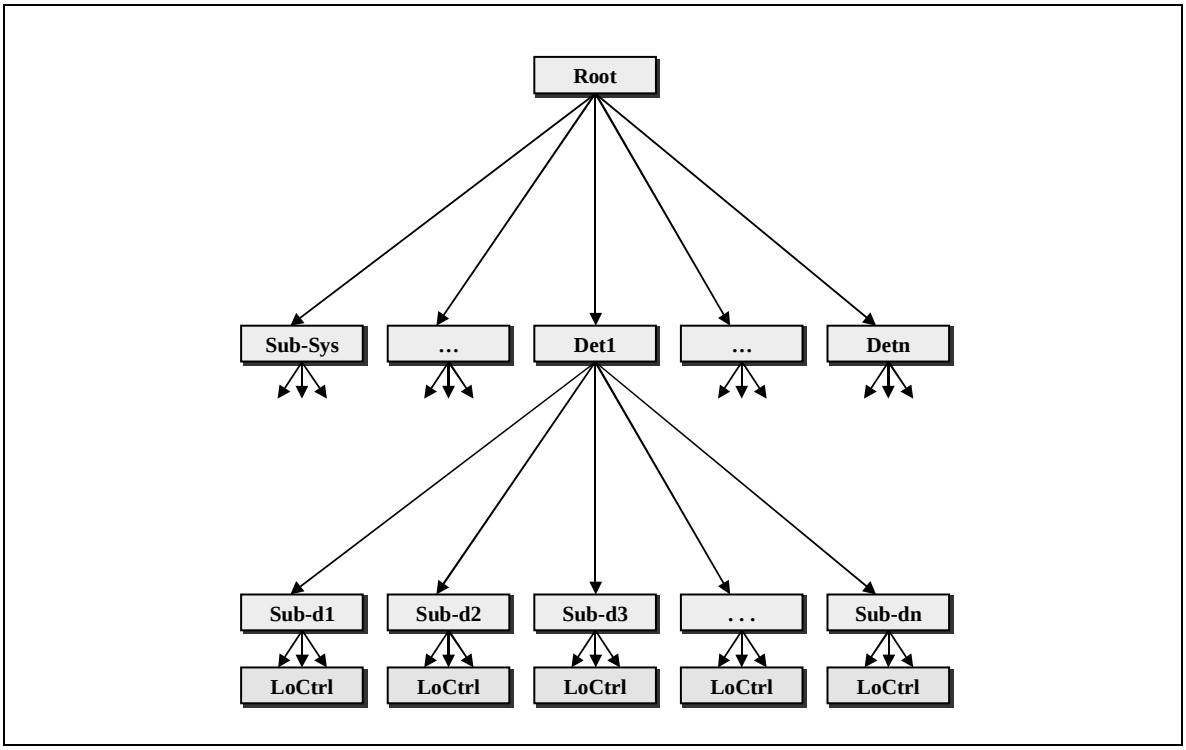

Figure 12-3 Online Software Control Hierarchy in TDAQ

A controller in the TDAQ system is characterised by its state following the TDAQ state model described in Section 12.4.3. In any place of the hierarchy, a change of state is initiated and synchronized from the higher level controller and sent down to the next lower level. From there information is returned to the next higher level when the requested transition has been performed. Possible error conditions are also reported back to the next higher level.

A controller framework allows to handle the operations described above in a coherent way on all the controllers in the system. On the other hand, it also gives the necessary flexibility to the detector expert to customize each controller for handling the individual tasks on the system under its control. These tasks take a wide range of variety from readout hardware to event filter farm control. The information on the relationship of the controllers and their responsibilities is detailed in the configuration database (Chapter 10.4.3).

The controllers have a number of responsibilities: Each controller is responsible for the initialisation and the shutdown of software and hardware components in its domain. It is also responsible for passing commands to child controllers and for signalling its overall state to its parent. Of particular importance is the synchronisation necessary to start the data-taking. This is performed by successive transitions through a number of intermediate states until data-taking is finally started as described below in Section 12.5.1, "Initialisation, Data-taking and Shutdown Phase". Interaction with the shift operator via the user interface drives the operations via commands to the root controller. The inverse series of synchronized transitions is traversed when

data-taking is stopped. If necessary, it is envisaged to introduce so called hidden states to allow for further synchronisation points on the sub-system level.

During all the operational phases, each controller is responsible for the supervision of the operation of elements under its direct control and for the observation of the operations of its children thus providing also for the task of error handling. In case of a malfunction of a detector, the controller can start corrective actions and/or signal the malfunction by sending messages. Severe malfunctions which are beyond the capabilities of a controller can be signalled by a state change to its parent. It is then the role of the parent controller to take further actions. The design of the control, supervision and error handling functionality is based on the adoption of a common expert system shell. Specific nodes will use different rules to perform their functions in addition to a common rule base which handles the generally valid aspects.

12.3.1 Control of the DataFlow

The DataFlow control encompasses the ROS/ROD control and the Data Collection control. It is comprised of the control of all applications and hardware modules responsible for moving the event data from the detector front-end electronics and LVL1 trigger to the high level triggers (LVL2 and EF). It includes the control of the ROD crates, the RoI Builder, the ReadOut System and the Data Collection applications, such as the Event Builder.

There are two flavours of local controllers in the DataFlow foreseen, both making use of the Online software infrastructure in the same way. The ROS controller is tailored towards the control of ROS software applications and hardware devices which cannot themselves access the online software facilities and the DC controller which handles the different types of DC applications and is optimized for the control of computer farms. A version of the latter is also used for the control and supervision of the high level triggers and is further described in Chapter 12.3.2. The main difference between the two controllers is that the one which controls ROD crates and the ReadOut System cannot assume that it controls active and intelligent elements. A ROD is a hardware device on which no standard software application is running: in this case the controller is the only access point to the databases as well as the only element communicating over IS/MRS to the Online system.

Both controllers can be deployed at different levels of the control hierarchy. As an example, a Data Collection controller can be used as top controller for all event building applications, as well as controller for a group of them. In general, such a controller can be in charge of other sub controllers or of endpoint data taking applications, transparently.

The DataFlow controllers make use of the configuration database to extract the information on the elements they are supposed to supervise. Their duty is to start, control and stop the data taking elements (hardware and software), to monitor the correct functioning of the system, gather operational statistics information and perform local error handling for those kinds of errors which couldn't be handled by the data taking nodes, but do not need to be propagated further to higher control levels.

12.3.2 HLT Farm Supervision

The emphasis for HLT control is on the management of the Computer farms. It is assumed that the farm for a HLT is divided into a set of subfarms, each under control of a specific controller. These controllers have well defined tasks in the control for the underlying processing tasks.

The High Level Triggers (HLT) perform the final selection before sending events to permanent storage. They consist of the Second Level Trigger (LVL2) and the Event Filter (EF). The two stages of the HLT are implemented on processor farms, divided into a number of subfarms. A key design principle 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 communalities between the two sub-systems need to be exploited as fully as possible. Bearing this in mind, a joint control and supervision system has been envisaged.

The Online Software configuration database will describe the HLT in terms of the software processes and hardware (processing nodes) of which it is comprised. The HLT supervision and control system will use the configuration database to determine which processes need to be started on which hardware and subsequently monitored and controlled. It is foreseen that the smallest set of HLT elements which can be configured and controlled independently from the rest of the TDAQ system (i.e. a ‘segment’) will be the subfarm. This allows subfarms to be dynamically included/excluded from partitions during data-taking without stopping the ‘run’. Supervision and control for each subfarm will be provided as a local run controller, which will interface to the Online Software run control via a farm controller. The controller will provide process management and monitoring facilities within the subfarm. The controller will maintain the sub-farm in the best achievable state 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.

It is foreseen that Online Software services will be used by the supervision system for monitoring purposes. For example, IS will be used to store state and statistical information which could be displayed (for example) by a dedicated panel in the Online Software graphical user interface.

12.4 Control Coordination

The control of the experiment is given by the interplay between three systems: The LHC machine, the Detector control and the TDAQ control. For each of the them the status of the system under control is expressed in distinct states.

12.4.1 Operation of the LHC machine

The phases of the LHC define a multitude of states [12-2] important for the internal functioning of the machine. In addition, measured values of the beam parameters are of prime importance for the operation and integrity of the detector. A subset is of direct interest for the interaction with the experiment control, in particular those states and parameters which describe the condi-

tion of the beam with consequences for the operation of the detector. Phases with stable beam and collisions indicate that the detector is operational for data-taking.

The main phases to consider here are of the following type: *filling* the beam from the SPS into the LHC, *ramp*, when the beam is accelerated up to its nominal energy, *squeezing* the beam, prepare for physics and *collide, physics* with stable beam, beam *dump and ramp-down* and *recover*. The various accelerator phases and beam parameters will be determined by either the LHC machine directly or by the monitoring of LHC equipment.

12.4.2 Operation of the DCS as a State Machine

Look for repetitions...

The DCS must enable the stand-alone operation of the sub-detectors, as well as the coherent and integrated operation of all sub-detectors for concurrent physics data-taking. For these reasons, the operation of the different sub-detectors will be performed by means of Finite State Machines (FSM). 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 required to fulfil the control needs in the various scenarios presented in the following. The FSM will handle the transition of the different parts of the DCS through internal states. It is envisaged to have one FSM per sub-detector and an additional FSM for the global operation of the experiment. The states of the sub-detector FSM will be assembled from the status of the different parts or services of the detector, which are determined by the status of the FE equipment, and from the status of the various environmental parameters monitored by the CIC station. The states of the external system, interfaced via the DCS_IS described in Section 11.8.1, will also be considered, e.g. the state of the LHC accelerator.

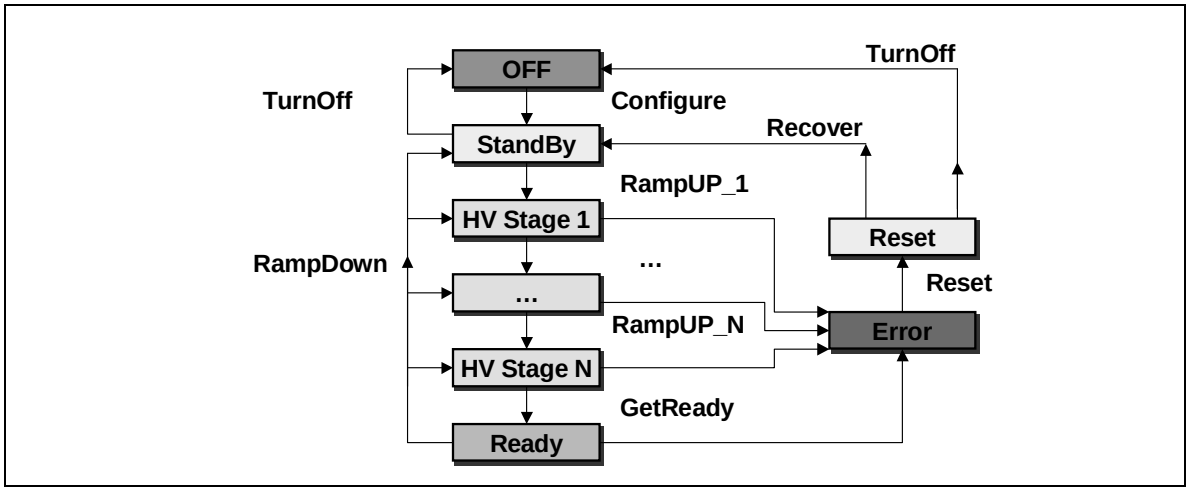

Figure 12-4 Sub-detector DCS states and transitions.

Figure 12-4 shows the DCS internal states for a given sub-detector. The DCS states are mainly determined by the state of the HV system although the status of the other systems of the detector as well as of the external systems will also be considered. The starting situation for a sub-detector is the OFF state. This subdetector may transit to the STAND_BY state after a successful configuration of the front-end equipment. 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 HV stages will be different depend-

ing on the sub-detector and will be defined according to recipes loaded from the configuration database. The transition to the READY state will be performed by means of the GET_READY command. This state implies that the sub-detector equipment is ready for physics data taking. The DCS will also permit to turn off or bring the sub-detector hardware into the STAND_BY state after a successful run in a controlled manner. If an error is detected during the transition to any of these states or during data taking, the subdetector will go to the ERROR state, where dedicated 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' FSM. Any command issued at this level, which triggers an state transition, will be propagated to the underlying sub-detectors' FSM. Similarly, any incident, which affect to the normal operation of a sub-detector, will be reported and it will trigger the state transition of the overall FSM to the ERROR state. As it will be described later in this chapter, the overall FSM is only needed when the TDAQ system is not running, otherwise it may be replaced by the Run Control state machine.

12.4.3 Operation of the TDAQ States

Three main TDAQ states from *initial* to *stand-by* and *running* have been introduced in Chapter 3.1. Here the states are further sub-divided as explained in [12-3] and shown in Figure 12-5. Two states are traversed between *initial* and *running*. Before arriving to the *initial* state the software infrastructure is initialized. The loading of the software and configuration data is performed which brings the system to the *loaded* state. The system configures the involved hardware and software and enters the *configured* state and the TDAQ system is 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 the L1 busy is set.

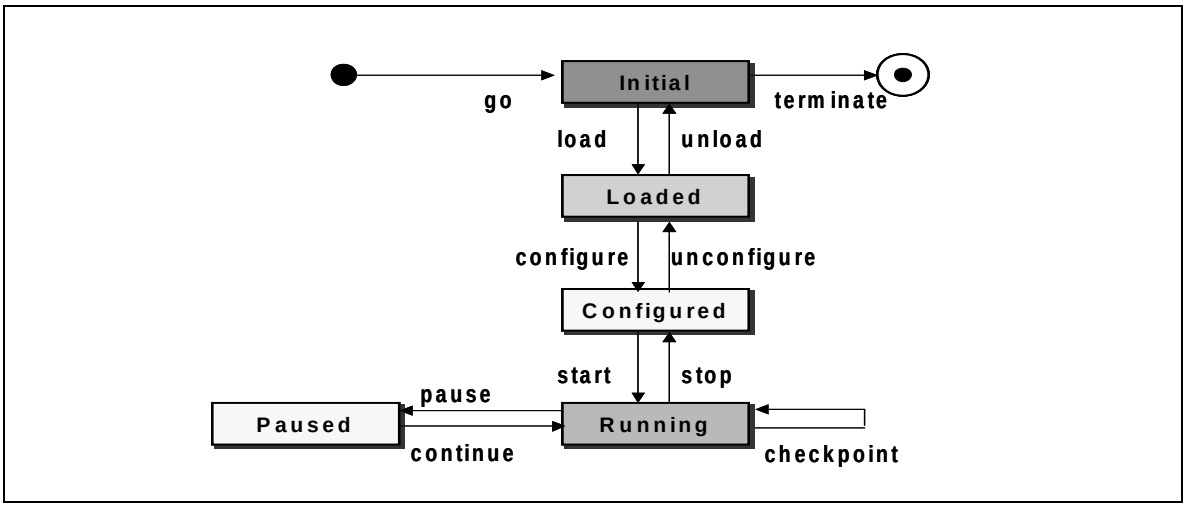

Figure 12-5 TDAQ states

The checkpoint is a transition in a running TDAQ system which is triggered by a change in conditions or by an operator. It results in the following events to be tagged with a new run number and does not need the synchronisation, via run control start / stop commands, of all TDAQ elements. Some components in the TDAQ control system require TDAQ sub-states which will be used for synchronisation during certain transitions.

12.4.4 Connections between States

As it has been presented in the previous sections, the LHC, the DCS, and the TDAQ system will each be operated through states. The synchronization of these systems is required in order to ensure coherent data taking and the integrity of the detector. The communication with the LHC are handled by DCS as described in Chapter 11. The DCS monitors the status of the LHC continuously and transfers this information in real time to the TDAQ system in order to prepare the detector for Physics data-taking. On the other hand, parameters measured by the TDAQ system like beam position or individual bunch luminosity, can be used to tune the beams and therefore, must be transferred to the LHC via the DCS.

Place holder for diagram illustrating the connection
between the TDAQ states, the DCS states and the LHC states

Figure 12-6 Connection between the TDAQ states, the DCS states and the LHC states

The actions on the sub-detector hardware performed by the DCS are coordinated with the states of the LHC machine to ensure the safe operation of the sub-detectors. This is the case of the ramping up of the high voltage of some sub-detectors, like the Pixel or SCT trackers. These sub-detectors are more vulnerable in case of insufficiently focused beam if the high voltage is on. For these sub-detectors this kind of actions on the detector will only be taken if the accelerator provides stable beams. For this reason, the DCS states will closely follow the operation of the LHC operation. The LHC state are related to a pre-defined set of operational conditions of the sub-detectors DCS and of the TDAQ system. Periods of particle injection or acceleration in the LHC may be used by the DCS or the TDAQ to initialize and configure the different parts of the systems, like the front-end electronics. Information on the internal states of the DCS is transferred to the TDAQ system via the DDC described in Section 11.8. Once the safe operation of the experiment instrumentation is ensured from the LHC machine, the DCS will bring the sub-detectors to the required state for data-taking and it will communicate their availability to the TDAQ system by means of the DDC. During Physics data-taking bi-directional communication between these systems take place to assure the correct coordination. An example is the request by the DCS to the LHC for beam dump if large backgrounds are observed by the TDAQ system in the detector.

The TDAQ control is only partially coupled to the LHC and the DCS states depending on the type of run. For physics runs it must be ensured that the LHC is providing stable beams and collisions are taking place and that DCS has brought the detectors into the *Ready* state. The TDAQ system can generally be brought from *initial* to the *configured* state while the LHC is ramping, squeezing and preparing for physics, and while the DCS prepares the detector for data-taking. It is then ready for taking physics data and waiting in a stand-by mode for LHC and DCS to be ready. For some of the calibration runs similar conditions apply for the selected sub-detector. For other calibration runs, for example with cosmic rays or with an external source the co-ordination with the DCS is required but not the co-ordination with the LHC states. For most TDAQ internal system tests no c-ordination with other states need to take place.

12.5 Control Scenarios

12.5.1 Initialisation, Data-taking and Shutdown Phase

The TDAQ states as described in Section 12.4 are traversed when the TDAQ system is run through initialisation, data-taking and shutdown phases. The preparation for data taking requires the initialization and configuration of all TDAQ hardware and software elements needed for the event readout, as well as a close coordination with the DCS, which acts on the sub-detector equipment. Unlike the TDAQ system, the DCS is always operational and ready to connect to TDAQ.

The described procedures rely on the Atlas partitioning concept which is described in Section 3.3.

12.5.1.1 Initialisation

The initialisation is performed in several steps walking through the states defined in Section 12.4. The states provide synchronisation points between the involved systems and subsystems. During the state transitions, actions specific to the sub-system like initializing software and hardware elements, loading software and parameters to hardware moduls, configuring them, or starting processing tasks are performed. Time-consuming operations are preferably performed during early state transition.

When initiating a data-taking session, the operations of the TDAQ system start from booted but idle machines. The TDAQ operator selects a configuration which is described as a partition in the database. The infrastructure, consisting of a number of servers in the distributed system (i.e. the Information Services), is started and initialized. The correct functioning of the hardware and software elements of the TDAQ infrastructure is then verified. Sequence and synchronisation of these start-up operations follow the dependencies described in the configuration database, see Section 10.4.1.1.

Once the TDAQ infrastructure is in place, the controller and the application processes, which are part of the configuration, are booted. The TDAQ process management is de-centralized and can therefore occur in parallel.

The TDAQ-DCS communication software is started and the communication between both systems is established. The TDAQ system passes the information of the chosen partition to DCS. The TDAQ controllers responsible for the command exchange with the DCS, connect to the individual DCS state machines of the sub-detectors. The preparation of the sub-detector equipment for data taking comprises the issuing of commands from the TDAQ system to DCS with the corresponding execution of several control procedures. These actions are defined according to recipes previously loaded from the configuration database and are sub-detector specific. The different procedures to be performed on the equipment are previously validated and cross-checked with the states of the external systems and of the common infrastructure to guarantee the integrity of the equipment, e.g. stable beams and acceptable backgrounds must be ensured by the LHC machine. In some cases, their execution can take up to several minutes depending on the characteristics of the DCS system.

These actions on the DCS equipment take place in parallel to the loading and configuring of the elements which are directly under TDAQ control. However, a callback is established between

the TDAQ controller and the PVSS application. The feedback on the requested actions is reported to the TDAQ system after the execution of the command, as well as the current state of the sub-detector DCS.

The TDAQ system is in the *initial* state.

Once all processes have been booted successfully the operator can cycle the system through the states which are used to synchronize the loading and configuring of software applications and hardware equipment which take part in the data-taking process.

During the loading transition, the initialisation of all the processing elements in the system including for example the *loading* of the software and configuration data is performed. During the following transition, called *configuring*, the configuration of a loaded system, for example the realization of connections between TDAQ elements or the setting of parameters, is performed.

At each of these stages, further synchronization with the DCS may be provided by issuing comands. These commands can be associated to state transitions of the TDAQ controllers, or be asynchronous commands issued directly by the TDAQ operator or by applications.

When these operations are terminated, the TDAQ system is in the *configured* state and ready to take data. The operations described up to here may be time-consuming and can therefore be performed a significant time before starting the run, for example when waiting for stable run conditions.

The availability of the sub-detector DCS for data taking is reported via the DDC to the TDAQ system. Generally the DCS will be in the *Ready* state to allow the TDAQ operator to start a run. However, the possibility to start a new run regardless of the state of the DCS is provided.

The TDAQ system is in the *configured* state and ready to receive the command for data-taking.

12.5.1.2 Data-taking

When the run is started, the L1 busy is removed and event data-taking operations are activated. If found necessary, a run can be paused and resumed later in an orderly manner. The transitions involved should only concern the direct data-taking activity to minimize time overhead. On the occurrence of special conditions the checkpoint transition as described in Section 3.2.6 can lead to a change in run number.

During the run, partitions which belong to one or more TTC partitions can be split off the main data-taking partition, for example in case of problems with the respective detector part. They can be joined at a later time, when the problem is solved. Depending on the TDAQ system elements which are involved, these actions may require a re-configuring of hardware or software modules and in this case it may be necessary to stop and re-start the run. However it is possible to remove and join sub-farms without affecting the data-taking and without stopping the run.

Component failures which cannot be handled locally are reported through the controllers to the system via the control communication mechanism. These mechanisms are described in Chapter 6, "Fault tolerance and error handling" and in Section 10.5.3, "Control Architecture".

When the operator stops the run, the L1 busy is set and all data-taking activities are stopped. The involved control and application processes remain active. No changes on the DCS side are foreseen and the sub-detectors will remain in the *Ready* state.

12.5.1.3 Shut-down

On receipt of the shut-down command the DCS will initiate a controlled shut-down of parts of the equipment. If the data-taking should terminate, the sub-detector DCS state will then be set to StandBy. In addition, clean-up operations ins software and hardware are performed in the TDAQ system. The previously started applications and then the controllers are stopped. Finally the TDAQ infrastructure is removed in an orderly manner to leave the system in a state in which a new and independent data-taking session can be started.

All the processes involved in the TDAQ system will be stopped in an orderly manner and to leave the system in conditions to be ready for the next data-taking period. The controllers are terminated and the finally infrastructure is removed.

12.5.2 Control of a Physics Run

remark: here the terms sub-detector and sub-sub-detector are used, this can be changed to whatever terms for those items will be decided to be used for the TDR.

The overall control system must ensure the safe and coherent operation of all sub-detectors in an integrated mode for concurrent physics data-taking. The online control system, which handles the control of all the readout elements in the system and the DCS which controls the detector hardware, are synchronized by means of commands issued from the TDAQ system to the DCS. The interface between TDAQ and DCS, called DDC, has been described in detail in Section 11.8. It consists of three aspects. Dedicated communication interfaces for data exchange, presented in Section 11.8.1 and for message exchange, described in Section 11.8.2 are available. During all data-taking phases, bi-directional data and message exchange between both systems may take place from any level of the DCS BE to the Online software information sharing and message reporting services. The command communication between TDAQ and DCS is provided via a dedicated controller called DDC_CT, which has been introduced in Section 11.8.3.

The TDAQ will be the master system and will drive the data-taking. Figure 12-3 shows the organization of the Back-End (BE) system of the DCS and the TDAQ hierarchy of controllers presented in Section 12.3. For Physics data-taking, all TDAQ controllers will be integrated in a single common partition.

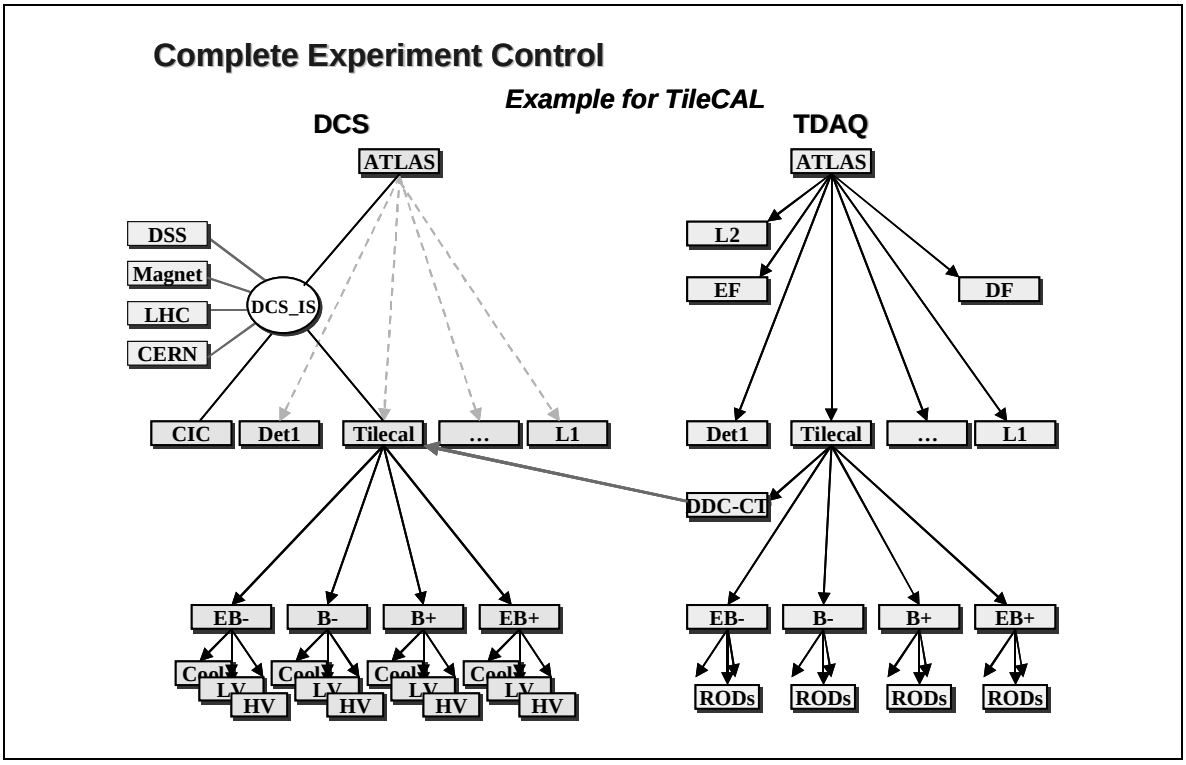

Figure 12-7 Complete Experiment Control mode. *Figure to be changed. DCS part should show a logical organization*

It is envisaged to have one DDC_CT controller per sub-detector. This controllers will send commands directly to the Subdetector Control Stations (SCS) on the DCS side. No command flow is foreseen from the TDAQ system and the DCS global operation station. The DCS SCS are directly connected to the sub-detector services underneath as well as to the external systems by means

of the DCS Information Server (DCS_IS). All information required to operate the detector is available at the SCS level. For these reasons, the SCS will have local decision capabilities. Therefore, the SCS represent the natural place to validate and execute commands issued by the TDAQ system, since they can cross-check the status of the different systems to ensure the safe operation of the detector.

This communication model implies that the TDAQ system will interact directly with the DCS FSM of the various sub-detectors. The availability of each of the partitions of the sub-detector for Physics data-taking will be notified to the TDAQ system from the SCS by means of the message transfer facility of the DDC.

Only a pre-defined set of high-level commands from the TDAQ system like the triggering of state transitions on the DCS side will be allowed. The interpretation and execution of the commands is the entire responsibility of the DCS, since the TDAQ system will not have the knowledge of the underlying structure of the DCS. The state of the command execution on the DCS side will be reported to the TDAQ system directly by means of the DDC_CT. The TDAQ Online software control system handles failures or time-outs from the DDC_CT in the same way as from other controllers in the system.

The TDAQ system control operates according to the hierarchical Online Software Control concept as further described in Section 12.3. As illustrated in Figure 12-7, the detector controllers, the farm control for EF and L2 as described in Section 12.3.2, and the DC controller operate at the same level in the hierarchy. Each detector controller supervises the sub-detector controllers and the controller which provides the detector connection to DCS. ROSs and RODs are supervised by the respective sub-detector controller. Global error handling and recovery is provided by the Online system control.

In case of malfunctioning of a detector or sub-detector, the respective partition can be removed from the control and readout partition tree. The global partition control will continue to supervise the readout if the detector part in question is not vital for data-taking for the type of physics chosen at the time. It can run in stand-alone mode to allow detector experts to repair eventual problems and join later the global control partitions and its readout chain.

HLT sub-farms can be removed or added to the global farm control without disturbance of data-taking actions. Breakdown and replacements of individual sub-farm nodes will be handled transparently and each of such operations will be logged.

this may have to be expanded in more detail

12.5.3 Calibration Run

the different types of calibration runs will be described similar to the description of the physics run

- *Three different type of calibration runs: Pure TDAQ, for example test pulses for LA calibration, pure DCS (calibration of the temperature sensors, adjustment of the cooling flow depending on the temperature) or both systems are involved, for instance the case of the calibration of the Tile Calorimeter using the Cs source.*
- *Shall we refer to only the third case in this section? My personal impression is yes.*
- *In all calibration where both systems are required, the TDAQ system will be the master of the process.*

- *The picture shows the case in which the Tilecal detector is operated in stand-alone mode for calibration where both systems are involved.*
- *The DCS will execute the commands issued from the TDAQ by means of the DDC_CT.*
- *case stand-alone of sub-sub-detector, also more than one in parallel*

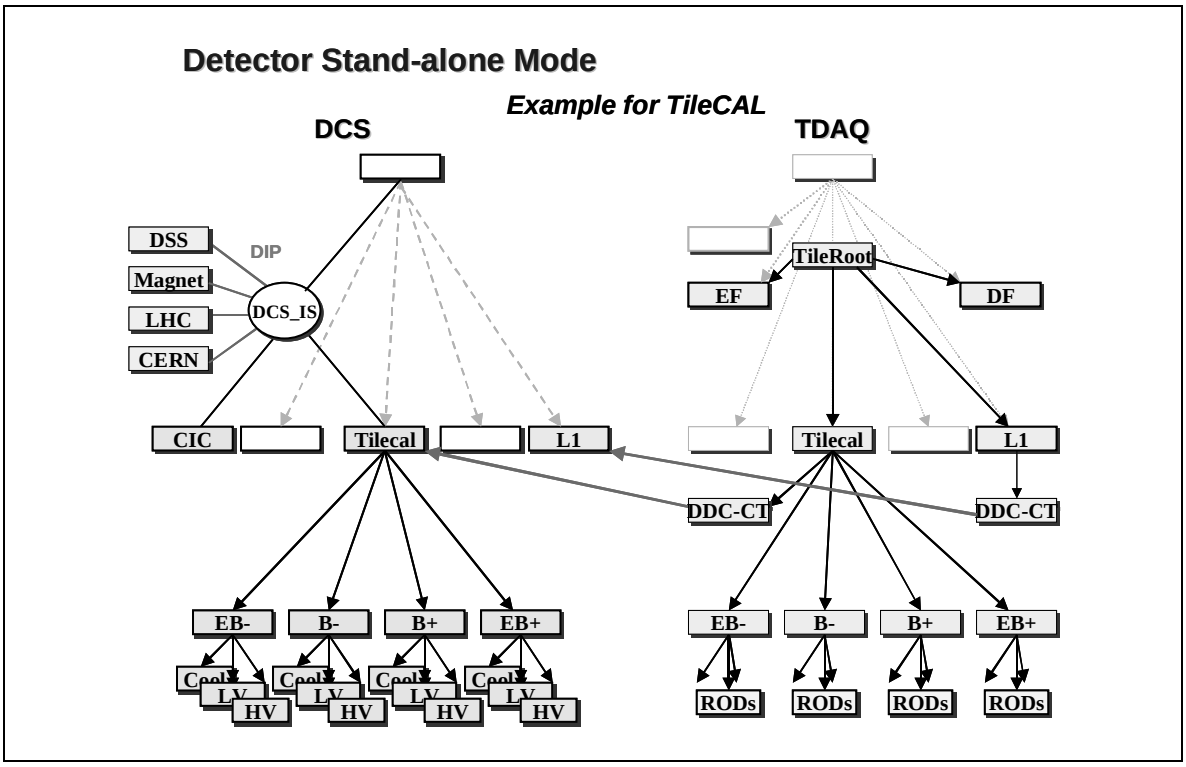

Figure 12-8 Detector Stand-alone mode. *Figure to be changed. DCS part should show a logical organization*

12.5.4 Operation outside a Run

Outside a run, operations take place between runs, with or without beams, and those performed during the LHC shutdown period.

Operations between runs: *to be written*

During shut-down periods and long term intervals without data-taking, the TDAQ system is not necessarily operational but the full functionality of the DCS will be required in order to supervise the operation of the sub-detectors and common services. In this situation, the DCS becomes the master of the detector. In order to reduce the cost, the operation at high power consumption equipment or the circulation of expensive gases in the sub-detectors will be interrupted in these periods. However some sub-detector services like the LAr and ID cryogenics, will continue to be operated during these periods. 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 sub-detector specific equipment will be required in order to enable a safe operation. For these reasons, the access of the DCS to the conditions and the configuration databases must be ensured outside a run. The DCS will also handle the communication with the ex-

ternal systems. The ATLAS magnet will be permanently switched on and therefore the interface with the DCS must be continuously available. The radiation levels monitored by the LHC control system must be accessible by the DCS at all times. The DSS will be able to trigger actions on the DCS in case of problems under these circumstances. Similarly, the interface to the fire brigade and to the access security system must be continuously operational.

In this scenario, the DCS will allow for the operation of the sub-detector in stand-alone mode, as required for system debugging or calibration runs, and also in an integrated mode. In the former case, the operation will be performed from the sub-detectors control stations having full control of the sub-detector, whereas in the latter, the overall control will be performed from the global operation station.

12.6 References

12-1 JCOOP Architecture Document

12-2 LHC Operations project, http://lhcop.web.cern.ch/lhcop/

12-3 *Runs and States - Global issues working group document*, http://atlas-project-tdaq-giwg.web.cern.ch/atlas-project-tdaq-giwg/Documents/Documents.htm

12-4 *Partitioning - Global issues working group document*, http://atlas-project-tdaq-giwg.web.cern.ch/atlas-project-tdaq-giwg/Documents/Documents.htm

12-5 The SCADA system

Part 3

System Performance

13 Physics selection and HLT performance

13.1 Introduction

In the Technical Proposal (TP) for the HLT, DAQ and DCS of the ATLAS experiment, a first understanding of the on-line event selection scheme and the corresponding physics coverage was presented. Since then, the studies have evolved, to cope with different machine scenarios and additional constraints coming from the detector itself. One of the major changes to take into account has been the LHC start-up phase, currently foreseen to deliver 10 fb^{-1} in one year, with a peak luminosity per fill of $2 \times 10^{33}\ cm^{-2}\ s^{-1}$, a factor of two higher than the Technical Proposal assumptions. This change has motivated a complete revisiting of the approach to the Physics and Event Selection Architecture of the experiment, leading to a novel way of reducing event rates and sizes, while retaining as much as possible of the ATLAS physics goals. Needless to say, only the availability of real data will allow this proposal to find a concrete implementation and the tuning of the relative weights of the selection will only be possible then, when confronted with the environment of LHC data taking.

As it has been explained in Chapter 9, the High Level Trigger (HLT) system of the experiment is composed of two separate data reduction steps, the LVL2 and the Event Filter (EF), each of them with distinctive and complementary features. The common denominator of these selections is that they will operate using software algorithms running on commercial computers to validate the hypotheses of particle identification. The LVL2 will do this with purpose built algorithms that need to operate in about 10 ms and use only part of the detector information at full granularity, the EF will have the fully built event at disposal, with a latency of the order of a second. An important aspect is to maintain a flexible scheme allowing for an easy adaptation to changes in conditions like luminosity or background: the modularity of the HLT will allow the implementation of different reduction steps at different stages.

A mandatory input concerns the seeding of the HLT selection, where a detailed simulation of the first level trigger (LVL1) result is needed: this level identifies the regions of the detector (Regions-of-Interest) where potential candidates for interesting physics objects are found. This detailed 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 and the distinctions of the LVL2 and the EF, it has been recognized that a coherent and organized approach to the software components of the trigger validation was needed to make a fundamental step forward with respect to the TP. The work that will be presented in Section 13.3 has concentrated on this issue, by deriving the common tools for the event selection and identifying the data model components and methods that can be shared across the different algorithms, in particular at LVL2. This will ease the implementation of different selection schemes, by making as well simpler the migration across levels.

Another important focus point for new developments has been the compliance with the updated detector geometry and with the realistic format of the data coming from the ReadOut System. This implies that algorithms will operate on streams of bytes organized according to the readout structure of each detector, in exactly the same way in which they will in the real experiment. This has allowed to study and understand the implication of converting those byte-

streams to the objects needed by algorithms in order to perform trigger selections as well as making preliminary measurements of the overheads stemming from these conversions.

In Section 13.4 the outcome of present studies are presented. Particular emphasis has been put on the selection of electrons and photons, and on the one of muons. For those “vertical slices” of event selections, a thorough implementation of the approach described above has been attempted. After LVL1 validation, data organized according to the readout format are used by LVL2 algorithms, operating within the framework of the PESA Steering and Control (refPESA). Trigger elements are then built using detector information and verified against hypotheses of particle identification. If LVL2 validation is successful, the EF reconstruction and analysis is performed (seeded or not by the LVL2 result) and the final selection published for off-line use. Rejection against dominant backgrounds and efficiencies for typical signals are reported, as well as the rates deriving from each of the selections.

To fully span the ATLAS physics coverage, also signatures involving jets, taus, ETmiss, as well as jets with b-quark content have been studied, and results are reported in the same section. As described in Chapter 4, the available on-line resources will also be used, for luminosities below the peak one, to evaluate b-production cross-section and make precision measurements with B-hadrons.

The global assessment, based on the present evaluation for each signature, of the ATLAS rates to off-line is made in Section 13.5, together with a preliminary description on how to reduce further the data volume by applying compression techniques or zero suppression to the detector information. A sketch of issues related to the initial phase of the experiment seen from the selection architecture point of view is also 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 result of which serves as input to the HLT trigger process. The ATLAS LVL1 trigger [13-2] 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.

The LVL1 trigger simulation is implemented in C++ in the ATLAS offline computing framework Athena and 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 shows a package view of the LVL1 simulation. It consists of packages simulating the resistive plate chamber (RPC) muon trigger (indicated by the package TrigT1RPC in Figure), 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 will be explained in the next sections. The simulation of the muon trigger in the endcaps, the signals for which are provided

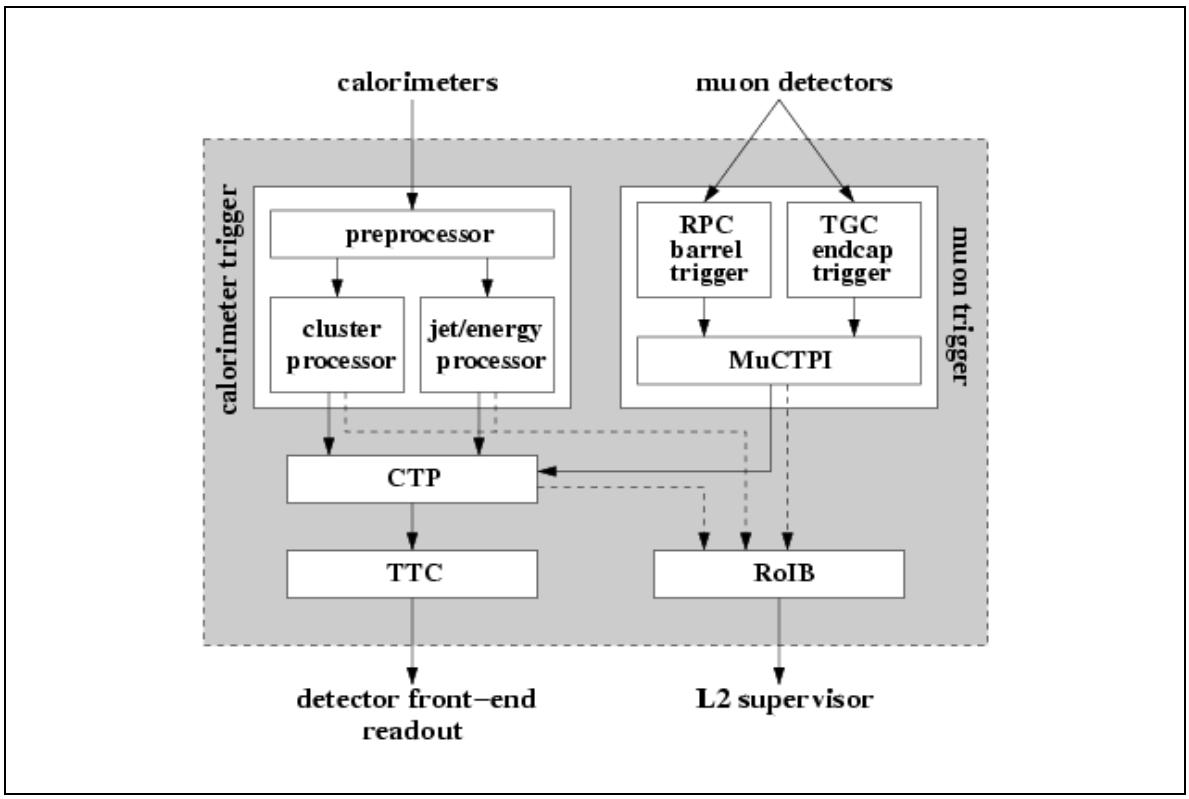

Figure 13-1 .An overview of the ATLAS LVL1 trigger system. The Region-of-Interest Builder (RoIB) formally is not a part of the LVL1 trigger. Its simulation, however, is done together with the simulation of the other parts of the LVL1 trigger

by the thin-gap chambers (package TrigT1TGC), so far exists only as a stand-alone program and will not be treated in detail.

The interfaces and data formats to be used in the simulation [13-3] were designed to follow as closely as was practical the data formats used in the LVL1 trigger hardware which are documented in [13-4]. Additional information on the LVL1 simulation can be found in [13-5].

13.2.1 Configuration of the LVL1 trigger

The task of the LVL1 trigger configuration is twofold: first, the trigger menu, i.e. the collection of event signatures LVL1 is supposed to trigger on, has to be translated into something that the simulation 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 delivered by the calorimeter and muon triggers found in the event (see later subsections for details about the calorimeter and muon trigger principles and simulations).
A simple example for a trigger item is ‘one (or more) electron / photon candidate with transverse momentum above 10 GeV and one (or more) muon candidate 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 integer numbers in front of and behind the string symbolize the required multiplicity and the required

transverse momentum, respectively. The combination of a candidate string and a threshold value (like 'EM10') is called a *trigger threshold*.

Second, the calorimeter and muon triggers have to be configured such that they deliver the information required for the event decision by the trigger menu, i.e. that the multiplicities for the required trigger thresholds are sent to the CTP simulation. For the implementation of the above mentioned example the calorimeter trigger has to be configured such that it delivers to the CTP the multiplicity count for the threshold 'EM10', i.e. the number of electron/photon candidate objects with transverse momentum above 10 GeV. It is obvious that the trigger menu and the trigger thresholds for the calorimeter and muon triggers have to be defined consistently. Particularly all thresholds used in the definition of any trigger condition in any trigger item must be delivered by the calorimeter and muon triggers and thus need to be configured.

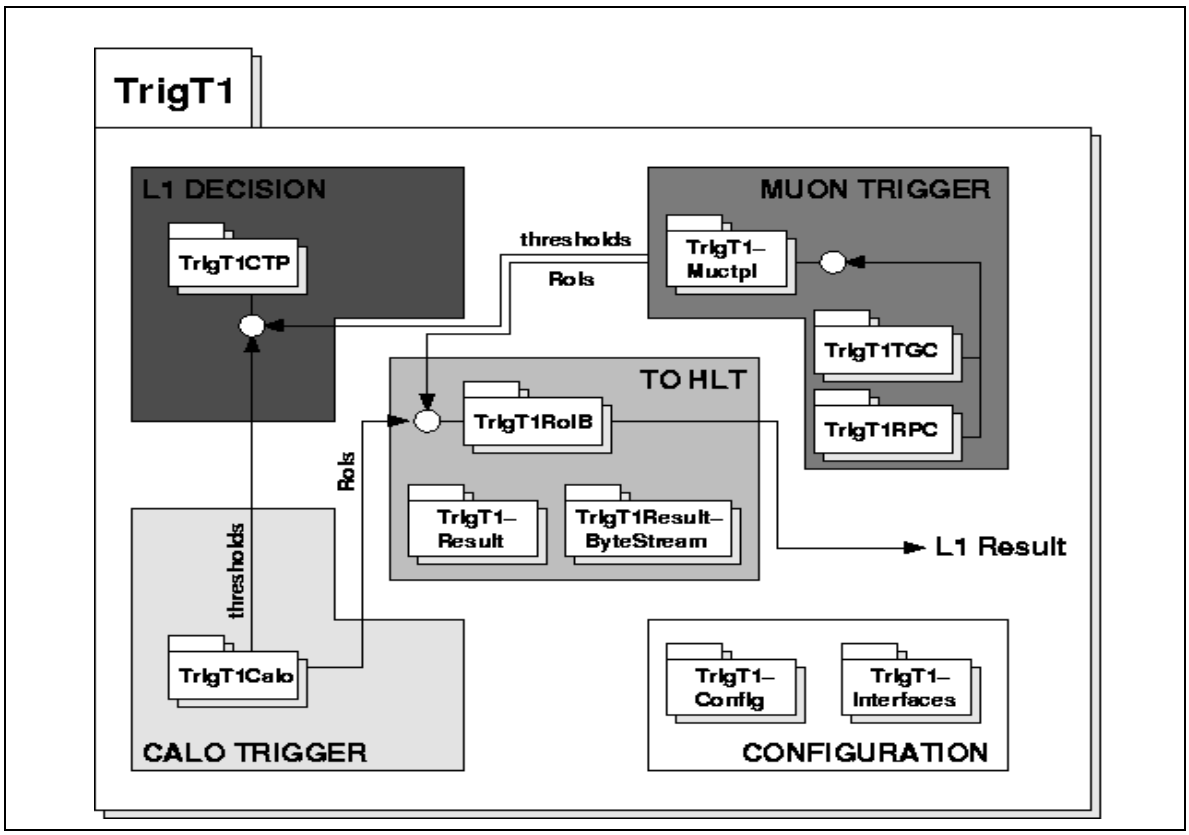

Figure 13-2 A package view of the LVL1 trigger simulation.

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 [13-6]. 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, in the configuration process configuration objects for the calorimeter and muon triggers are created 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 configura-

tion 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 [13-7] has to provide information on localised energy depositions in the ATLAS calorimeters, which might be due to single particles (electrons, photons, hadrons), to tau leptons or to hadronic jets with transverse energies (E_T) above a set of pre-defined thresholds and satisfying certain isolation criteria. The multiplicities of these candidate objects are counted and are passed on to the CTP to be used in the LVL1 event decision. In addition, the calorimeter trigger calculates global energy sums (total and missing transverse energy) which are discriminated against a set of configurable thresholds and are also used in the CTP event decision.

The ATLAS calorimeter starts with the signals of 7200 trigger towers which are analogue sums of trigger cells in the liquid argon and tile calorimeters. The trigger tower signals are digitized in the preprocessor electronics and then subject to two different trigger algorithms: Electron / photon and tau / hadron candidates are searched for within the 6400 trigger towers of granularity $\eta \times \phi = 0.1 \times 0.1$ in the central part of the ATLAS calorimeters ($|\eta| < 2.5$) using the Cluster Processor. For the algorithms searching for jet candidates and deriving global energy sums in the Jet / Energy Processor, coarser (jet) elements of granularity 0.2×0.2 are used, which are built from all 7200 trigger towers and are available for a larger rapidity range ($|\eta| < 3.2$ in case of the jet trigger).

In case of the electron / photon trigger a candidate object is defined by a local maximum of transverse electromagnetic calorimeter energy in a region of 2×2 trigger towers corresponding to a 0.2×0.2 region in $\eta - \phi$ space. In addition, vetos on the amount of hadronic energy in that region and on the amount of energy surrounding the 2×2 region are applied. The highest transverse energy sum that can be build from any neighbouring two of the four trigger towers in the 2×2 region defines the transverse energy of the candidate object which is discriminated against the predefined thresholds. See [13-2] and [13-8] for a more detailed description of the various calorimeter trigger algorithms.

In addition to the counting of object multiplicities, Regions-of-Interest (RoIs) are defined using the highest E_T thresholds passed by the candidate objects and a bit pattern indicating their location. These RoIs are transmitted to the Region-of-Interest Builder and serve to seed the HLT event decision process. All relevant calorimeter trigger information is also provided to the readout using S-LINKs.

The LVL1 calorimeter trigger simulation is designed to reproduce the functionality of the hardware, but does not entirely duplicate the dataflow. The primary reason 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 starts from input calorimeter cell signals; there exists no dedicated simulation of trigger tower signals. The cell signals can be taken from the fast ATLAS simulation or from the detailed GEANT calorimeter simulation. It is also possible to feed the software with hardware test vectors, i.e. test patterns for which the output of the trigger logic is known and which thus serve for functional tests of the hardware and software.
The cell signals are then 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 / tau / hadron finding), and are summed into the coarser Jet Elements to be used in the simulation of the Jet and Energy Processors (jet finding and building of energy sums). The multiplicity outputs for the CTP simulation are produced and stored in the TES for later retrieval, and the simulated candidate objects are formatted according to the S-LINK data protocol and are stored for later use in the RoIB simulations. So far there is no simulation provided for the readout data stream.

The HLT steering software requires the RoIs to be given not in terms of the LVL1 internal data format (basically a bit pattern detailing the electronics module the signal came from and a bit indicating the threshold that was passed), but using the coordinates in η–ϕ space and the value (in GeV) of the threshold that was passed. In order to provide this information, software converters are provided to translate the raw 32-bit RoI data words into objects, complete with methods to return the required data.

SAY SOMETHING ABOUT CALORIMETER TRIGGER VALIDATION HERE!!!!

13.2.3 The RPC muon trigger and its simulation

In the barrel part of the ATLAS detector ($|\eta|<1.05$) the muon trigger uses information from the resisitive plate chamber detectors (RPC) of which ATLAS has six layers organised in three so-called stations [13-9]. The middle RPC station (RPC2) is called the *pivot plane*. The algorithm that finds muon candidates works as follows [13-10]: 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 where the line hits the station RPC1. Since the ATLAS magnetic field will bend the trajectory of charged particles, the size of the coincidence window defines the transverse momentum p_T of muon tracks that can be triggered. 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. If, in addition, there is a coinciding hit in at least one of the two planes of the outermost station RPC3, a high-p_T candidate has been found. For each of the 64 sectors of the RPC trigger, up to two muon candidates can be selected and sent to the MuCTPI.

The input to the simulation of the muon trigger logic is provided by a package that performs the simulation of the RPC trigger detector system; this is done with the program ATLSIM. The muon detector layout used for this simulation is the version “P03” [13-11]; the geometry of the single RPC stations and the position of these stations in the muon spectrometer is reproduced in great detail, following with care the engineering drawings. The material composition and geometry of the single RPC units are also correctly simulated. The simulation of the RPC detector response is based on the results obtained in test-beam experiments. The hits produced by the simulation of charged particles crossing the RPC detectors are collected and stored in output files and can be used in downstream packages for the simulation of the trigger logic and also for the event reconstruction.

The detector simulation stage is followed by a set of Athena algorithms which are used to simulate in detail the logic of the LVL1 muon barrel trigger. There are basically two sets of objects:
The first set is a collection of objects corresponding to (and simulating the behaviour of) the basic elements of the hardware system: the Coincidence Matrix ASIC (CMA), the Pad board, the Sector Logic and the ReadOut Driver. All data belonging to a given CMA are recorded in a dynamic structure, and the input register bits corresponding to the fired channels are set to one. Channel masking, time alignment and the introduction of an artificial dead time to the fired channels are possible although not used yet in the present implementation. The trigger data of

the eight CMAs belonging to a given Pad board are processed following the same logic as the electronics, and the output of all Pads belonging to a given trigger sector are sent to the corresponding Sector Logic. The Sector Logic identifies the two highest transverse momentum muon candidates among all the Pads and provides the adresses of the relevant Regions-of-Interest.The output of the Sector Logic is finally stored in the TES from where it can be retrieved by the MuCTPI simulation.
The second set of objects comprises four packages which reproduce the architecture of the muon trigger system and have access to the geometry of the RPC trigger system. The main functions of these packages are the mapping between different hardware components (RPC detectors to CMAs to Pads to Sector Logic objects), the configuration of the CMAs according to the thresholds chosen, and the data transfer between the different parts of the simulation.

The CMAs supply information also to the readout system; this data path is also simulated. The resulting data are organised in a structure that follows exactly the one adopted in the hardware (*bytestream* format). This, together with software converters for the interpretation of the bytestream data, allows the use of the RPC data in LVL2 selection algorithms like muFast [13-12].
In addition to the simulation just described, a fast simulation of the trigger logic for a fast emulation of the LVL1 trigger logic at LVL2 is provided.

The simulation software was completely rewritten with respect to the software used for the HLT technical proposal [13-13] and has only recently been integrated into the overall LVL1 trigger simulation. Therefore, the validation process has just started; so far no results showing the equivalence of the current simulation with the one used in the technical proposal or demonstrating the improved performance of the trigger due to a more precise simulation and a better detector description are available. Some information on the performance of the RPC muon trigger can be found in [13-12] and [13-14].

13.2.4 The Muon-to-CTP interface and its simulation

The Muon-to-CTP Interface (MuCTPI, [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 six different muon p_T thresholds and are sent to the CTP. In case the event is accepted, up to 16 candidates (chosen are the ones with the highest p_T) are formatted to conform to the ATLAS ROD standard and are sent via the Region-of-Interest Builder to the HLT and, via a separate link, to the readout system. The MuCTPI suppresses candidates which are the result of double-counting due to overlapping muon chambers.

The MuCTPI simulation follows the hardware scheme as closely as possible, down to the data formats used in the hardware. The dataflow 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 a stand-alone program for extensive tests of the prototype MuCTPI hardware. It has recently been ported to the ATLAS offline framework, Athena. It has been integrated with the simulation of the RPC muon chambers and trigger on the input side, and with the simulations of the CTP and the RoIB on the output side. Also the output to the readout is simulated; this is however 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, [13-16]) in the afore mentioned two-step procedure:

The threshold multiplicities from the calorimeter and muon triggers are discriminated against the *trigger conditions* introduced in Section 13.2.1 — 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 by LVL1 as defined in the trigger menu; gates and prescales can be applied to each individual item. The LVL1 trigger result is then the logical OR of all trigger items. The final CTP design will probably foresee up to 96 trigger items.

The logical relations between the conditions and the items on one side, and the conditions and the input threshold multiplicities on the other side, are provided by the LVL1 trigger configuration (Section 13.2.1). The CTP provides identical output to the RoIB and to the readout; the information that is sent comprises bit patterns for the input signals and for the trigger items before and after prescales and vetos, and the L1A signal.

In the currently existing prototype hardware implementation of the CTP, the CTP-D ('D' for demonstrator, [13-17]), this selection procedure is implemented using two look-up tables (LUT) for the multiplicity discrimination and two programmable devices (CPLD) for the combination of conditions to items. The final design of the CTP will probably incorporate only one big programmable device in which both above mentioned steps can be performed.

The existing CTP simulation follows closely the CTP-D design — conversion to the final CTP design may eventually require some effort, depending on how close to the hardware implementation the simulation software is supposed to be.

First, the input threshold multiplicities provided by the calorimeter trigger and MuCTPI simulations are collected, and the multiplicities that are required in the trigger menu are discriminated against the respective conditions which are taken from the C++ object representing the trigger menu that is provided by the configuration step.

In a recursive algorithm the conditions are then combined to trigger items. Taking into account the logical relations is facilitated by the use of XML as the medium for the trigger menu definition.

Then all items are passed through a prescale algorithm, and the logical OR of all items is formed, resulting in the LVL1 event result (the LVL1 accept or L1A signal which might be 0 or 1, FALSE or TRUE). No effort has been undertaken so far to implement the deadtime algorithms realized in the hardware.

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, [13-18]) which formally is not a part of the LVL1 trigger. This device collects the information relevant for the HLT from the calorimeter and muon triggers and from the CTP, and combines all data into a single block which serves as input to the LVL2 supervisor. The data are transmitted in S-LINK format (four S-LINKs from the calorimeter trigger cluster processor, two from the jet/energy processor, and one from each the MuCTPI and CTP). The RoIB has to operate at the highest foreseen LVL1 output rates without introducing additional deadtime.

The RoIB simulation picks up the S-LINK information stored in the TES by the calorimeter trigger, MuCTPI and CTP simulations and constructs a LVL1 result raw data object (RDO). This object, the content of which is used to seed the HLT steering, can then be converted into persistent or transient hardware (or *bytestream*) format using a software converter which is provided within the LVL1 simulation effort, as is the converter for translating the bytestream format back into objects which serve to seed the HLT trigger and contain the value (in GeV) of the passed threshold and the location of the RoI in the η–ϕ space.

13.3 Common tools for selection

The HLT algorithms are the basic software components which provide data to derive the trigger decision. These algorithms operate within the context and environment of the PESA Core Software which is discussed from a conceptual design and architectural standpoint in ***Chapter 9.*** Section 13.3.1 provides an overview and description from the viewpoint of these HLT algorithms. The objects of a common Event Data Model which algorithms exchange and manipulate are described in Section 13.3.2. An inventory of HLT algorithms intended to operate in the LVL2 environment is given in Section 13.3.3 while those for the EF are described in Section 13.3.4.

13.3.1 Algorithmic View of the Core Software Framework

Unlike their counterparts in the Offline Software environment, HLT algorithms must allow themselves to be guided by the PESA Steering, to be seeded by Trigger Elements, and to operate with a restricted set of event data.

To accomplish the Steering guidance of algorithms using Sequence Tables and Trigger Elements, a Seeded approach is required. Trigger Elements characterizing abstract physics objects have a label (e.g., ‘electrons’ or ‘jets’) and effectively decouple the Steering and Physics Selection from details of the Event Data Model used by the algorithms. Via the Navigation scheme within the PESA Core Software environment, algorithms may obtain concrete event data associated with a given Trigger Element which define the Seed of restricted and relevant event data fragments upon which they should work.

The Trigger processing itself starts from a LVL1 RoI using predefined Sequences of algorithms. These LVL1 RoI objects are associated to Trigger Elements allowing them to be acted upon by the Steering. For each of these Trigger Elements, the Steering executes the required algorithms as defined in a Sequence Table. Hence, it is possible that a given algorithm may be executed *N* times per event. This is fundamentally different than the ‘Event Loop’ approach of the Offline reconstruction paradigm where a given Offline algorithm would act only once upon each event.

At LVL2, event data reside within ROBs until actively requested. This allows the LVL2 algorithms to request and process only a small fraction of event data from ROBs, representing a substantial reduction in the network and computation resources required. The first step in this process 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 RegionSelector .

The HLT RegionSelector [13-20] translates geometrical regions within the fiducial volume of the detector into a set of Identifiers. Presently these Identifiers are `IdentifierHash`es used in the offline software environment. They correspond to elements of appropriate granularity in each sub-detector, usually a DetectorElement. As such, the RegionSelector uses DetectorDescription information during its initialization phase to build an `EtaPhiMap` for each layer (or disk) of a subdetector. This map is essentially a two-dimensional matrix in η and ϕ. Each element consists of a `list` of `IdentifierHash`; the column indices are ϕ floating point numbers while a range (η_{min}, η_{max}) specifies row indices. The input to RegionSelector is the sub-detector under consideration (*i.e.*, Pixel, SCT, TRT, LAr, Tile, MDT, RPC, CSC, or TGC) and the physical extent of the geometrical region. Given the vastly different designs of each subdetector, a subdetector-dependent procedure is used. With knowledge of the layers and/or disks in the region, the RegionSelector searches the $\phi \rightarrow$`IdentifierHash` map which will give a set of `IdentifierHash` is relevant in ϕ region. The last step is to validate each `IdentifierHash` inside the `IdentifierHash`$\rightarrow (\eta_{min}, \eta_{max})$ map.

Interactions with the Data Collection system are hidden from the Algorithm behind a call to StoreGate. Within StoreGate, event data are aggregated into collections within an IdentifiableContainer (IDC) and labelled with an Identifier. Algorithms request event data from StoreGate using the set of Identifiers obtained by the HLT RegionSelector. If the collections are already within StoreGate, it returns them to the HLT algorithm. If not, StoreGate uses the `IOpaqueAddress` to determine which ROBs hold the relevant event data and requests it from the Data Collection system. A ByteStream converter converts the Raw Data into either Raw Data Objects (RDOs) or, by invoking a DataPreparation AlgTool, into Reconstruction Input Objects (RIOs). The obtained RDOs or RIOs are stored within the collections within the IDC within StoreGate.

13.3.2 Event Data Model Components

During 2002 and 2003, there has been a substantial ongoing effort within the HLT, Offline, and subdetector communities to establis a common Event Data Model (EDM) between HLT and Offline software in the areas of the raw and reconstruction data models. In the discussion that follows, the concept of a DetectorElement is used as an organizing and identifying principle for most EDM objects; these are discussed in Section 13.3.2.1. At the time of writing this document, there has been convergence with respect to the raw data model described in Section 13.3.2.2. Common reconstruction data model classes specific to LVL2 and EF algorithms have been developed and are described in Section 13.3.2.3 and Section 13.3.2.4.

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 IdentifiableContainer (IDC) with Identifier labels corresponding to the unit of aggregation. For most sub-detectors, the organizing principle is that of the DetectorElement.

In the Pixel detector a DetectorElement is a module, equivalent to a single Silicon wafer; hence there are 1744 Pixel DetectorElements. For the SCT, a DetectorElement 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 DetectorElements. For the TRT, a DetectorElement is a planar set of straw tubes representing one row at a constant distance from the module inner wall of straws in a barrel module (*i.e.*, a plane corresponding to the tangential direction in the barrel) and 1/32 in $r\phi$ at a given z of straws in an end-cap wheel; there are 19008 TRT DetectorElements [13-19].

For the calorimeters, the concept of DetectorElement is difficult to define. Instead, the organizing principle for event data is that of the Trigger Tower.

Within the muon spectrometer, for the MDTs, a DetectorElement is a single MDT chamber, where there is at most a single MDT chamber per station, and typically, an MDT chamber has two multilayers. An RPC DetectorElement is the RPC components associated to exactly one barrel muon station; there may be 0, 1 or 2 RPC doublet sets per station and a doublet set may comprise 1, 2 or 4 RPC doublets. A TGC DetectorElement 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, for a CSC DetectorElement is a single CSC chamber, where there is at most a single CSC chamber per station. A CSC chamber typically has two multilayers.

13.3.2.2 Raw Data Model Components

ByteStream Raw Data is ROB-formatted data produced by the ATLAS detector or its simulation [13-19]. It is defined by a set of hierarchical fragments, where only the bottom level, the ROD fragment, is defined by the sub-detector group. The format of the ByteStream has not yet been formally defined. Hence, preliminary “best guesses” have been made as to its structure which may undergo changes in the future.

A Raw Data Object (RDO) is uncalibrated Raw Data converted into an object representing a set of readout channels. Historically this has been referred to as a *Digit*. It is the representation of Raw Data which is put into the Transient Event Store (TES) and is potentially persistifiable.

The purpose of the RDO converters is dual: first a Raw Data ByteStream file can be created by taking the information from the already filled RDOs (in the transient store, from ZEBRA); second, this ByteStream file can then be read back by the converters to fill the RDOs (or the RIOs for LVL2). Since the RDOs are a representation of the specific detector output, its content can change with the life time of the sub-detectors.

A detailed description of the Raw Data Model components is available elsewhere [13-21].

13.3.2.3 Reconstruction Data Model Components

Algorithms interact with Reconstruction Input Objects (RIOs) as opposed to RDOs. For each subdetector system, classes of RIOs have been defined and are described in the following subsections.

13.3.2.3.1 Inner Detector

The implementation of the RIOs makes use of the `IdentifiableContainer` base class, and the collections are also according to the granularity of DetectorElements.

The Pixel and SCT RIOs are Clusters. A Cluster in the Pixel detector is a two-dimensional group of neighbouring readout channels in a DetectorElement. 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 EF and Offline and one used for LVL2. The one used at EF has Pixel and SCT sharing the same class. For LVL2 there is a common structure for Pixel, SCT and TRT, but they all have their own concrete classes. For Pixel and SCT there is a base class used for LVL2. There is also an *Extended* class which could potentially be used at EF (which inherits from the LVL2 base class) in the future. Both LVL2 and EF set 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 [13-19]. It is assumed that in the future there will be only one set of RIO classes to be used for LVL2, EF, and Offline.

At LVL2, Pixel and SCT RIOs are converted to 3-dimensional coordinates in the ATLAS global coordinate system using the `AlgTools` `SCT_SpacePointTool` and `PixelSpacePoint-Tool`. These tools accept as input a STL `vector` of pointers to Cluster Collections of the appropriate type, `SCT_ClusterCollection` or `PixelClusterCollection`, and return a STL `vector` of objects of the class `TrigSiSpacePoint`. A UML class diagram of the LVL2-specific SpacePoint class `TrigSiSpacePoint` and associated `InDetRecInput` classes is shown in Figure [Ref: fig:spacepoint]

For the Pixels, the creation of SpacePoints consists of combining the local coordinates of Clusters with information on the position and orientation of the DetectorElement 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 2-dimensional information. One species of SCT DetectorElement, phi-layer, has strips orientated parallel to the beam axis, the other, *u* or *v* layer, is rotated by ± 40mRad with respect to the phi-layer DetectorElements. The formation of SpacePoints 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 SpacePoint is formed for each (in this case one, at most, 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 DetectorElement transform to global coordinates.

Note that for the LVL2 SpacePoints some simplifications are made in the interest of speed, as follows:

1. There is a Cluster Collection per DetectorElement.

- No attempt is made to form SpacePoints from Tracks passing close to the edge of a module, where the corresponding stereo-layer Cluster is in a different module.
- Since the stereo and phi layers are separeted by a small distance, the trajectory of the track will influence the measurement of the second coordinate. Since the trajectory is not known at the time that SpacePoints are created, there will be a corresonding increase in the uncertainty in the measurement in the second coordinate (*R* or *z*).

The TRT RIO is the drift circle of a straw. In the case of the TRT, the same classes are used for LVL2, EF, and Offline: those classes are the `DriftCircle` classes part of the set of classes that are also used at LVL2 for Pixel and SCT. 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 and is just the direct content of the detector output, while in the case of the RIO the straw information is calibrated: out of the drift time, a drift radius is obtained. For now, the drift function applied is the same for all straws. In the future the constants that go into the parametrization of this drift function will come from the Interval of Validity Service [13-23].

13.3.2.3.2 Calorimeters

For the Calorimeters, the RIOs are calibrated calorimeter cells (`LArCells` and `TileCells`), imported from the offline reconstruction.

Both `LArCells` and `TileCells` have `CaloCell` as a common base class which represents the basic nature of a observation in the calorimters an energy, position, time, and quality. A `CaloCell` has been calibrated so that `energy()` 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 occured,relative to the trigger; it should be zero for good hits. Quality reflects how well the input to the system matched the signal model on which the algorithm is based. It is a number with a value betwen zero to one, giving the significance of the hypothesis that the actual signal is a sampling of the signal model (*i.e.*, it is the integral of a probability distribution from negative infinity to an observed value of a test statistic and ought to be uniformly distributed between zero and one if the hypothesis is correct).

13.3.2.3.3 Muon Spectrometer

[Need text here.]

13.3.2.4 Reconstruction Output

13.3.2.4.1 Tracks

A track is, in general, an object containing a parameterization of a hypothesized particle trajectory through space relating groups of RIOs and / or SpacePoints together. A Track trajectory consists of three position, two direction, and one curvature[1] parameters. If a track is evaluated at an intersecting surface, there are five parameters and a covariance matrix.

A proposed uniform Track class exists for LVL2 algorithms, the `TrigInDetTrack` class. A UML class diagram of `TrigInDetTrack` and associated classes is shown in Figure [Ref: fig:track] . No such uniform Track class yet exists in the Offline environment.[1]

13.3.2.4.2 Calorimeter Clusters

[To be written]

13.3.3 HLT Algorithms for LVL2

13.3.3.1 IDSCAN

IDSCAN (see Refs. [13-32] and [13-33]) is a track reconstruction package for LVL2. It takes as input SpacePoints found in the Pixel and SCT Detectors. A series of sub-algorithms (`ZFinder`, `HitFilter`, `GroupCleaner`, `TrackFitter`) then processes these and outputs Tracks and the SpacePoints associated with them.

The `ZFinder` finds the *z*-position of the primary interaction vertex. The algorithm puts all hits into narrow ϕ-bins and extrapolates pairs of hits in each bin back to the beam-line, storing the *z* of intersection in a histogram. It takes as the *z*-position the histogram region with the most entries.

The `HitFilter` finds groups of hits compatible with Tracks from the *z* position found by `ZFinder`. It puts all hits into a histogram binned in ϕ and η. It then finds clusters of hits within this histogram. It creats a *group* of hits if such a cluster has hits in more than a given number of layers.

The group of hits found by `HitFilter` is used by `GroupCleaner` which splits groups into Tracks and removes noise hits from groups. Each triplet of hits forms a potential track for which p_T, $\phi 0$, and $d0$ 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 `TrackFitter` verifies track candidates and finds the track parameters by using a standard Kalman-filter-type fitting algorithm adapted from SCTKalman [13-24]. It returns a list of SpacePoints 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 `SpacePoints` as input and outputs fitted reconstructed Tracks, each storing pointers to the `SpacePoints` used to build it. SiTrack is

1. The use of curvature assumes a homogenous magnetic field in which case this quantity is constant. For ATLAS and its significantly inhomonogenous magnetic field in the endcap region of the Inner Detector and in the Muon Spectrometer, this parameter may be replaced by an invariant quantity such as *charge* / *p*.
1. There are of course Track classes defined internally within ORPs such as iPatRec and xKalman++.

implemented as a single main algorithm `SiTrack` which instances and executes a user defined list of sub-algorithms (chosen among `STSpacePointSorting`, `STMuonVertex`, `STTrack-Seeding`, and `STThreePointFit`).

`STSpacePointSorting` collects pointers to `SpacePoints` 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.

`STMuonVertex` is a primary vertex identification algorithm mostly suitable for low luminosity events with an high p_T muon signature. It is based on track reconstruction inside the LVL1 muon RoI: the most impulsive track is assumed to be the muon candidate and its z impact parameter is taken as the primary vertex position along z.

`STTrackSeeding`, using the sorted `SpacePoint` map and a Monte Carlo Look-Up Table (MC-LUT) linking each B-layer module to the ones belonging to other logical layers, builds track seeds formed by two `SpacePoints` 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 improved. If the primary vertex has already been reconstructed by `STMuonVertex`, a fraction of fake track seeds can be rejected during their formation, applying a cut on their z distance from the primary vertex. Otherwise, if no vertex information is available, an histogram whose resolution depends on the number of seeds found is filled with the z impact parameter of each seed; its maximum is then taken as z position for the primary vertex. This vertexing algorithm, which can be operated in both RoI and full scan modes, is best suitable 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.

`STThreePointFit` extends track seeds with a third `SpacePoint`; it uses a Monte Carlo map 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.*). `SpacePoints` 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 SpacePoint) can be solved on the basis of the track quality, leading to an indipendent 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 tracking algorithm for track reconstruction in the TRT. It is described in detail elsewhere [13-25]. The algorithm takes as input Hits in the TRT. The algorithmic processing consists of Initial Track Finding, Local Maximum Finding, Track Splitting, and Track Fitting and Final Selection. It outputs the Hits used and Tracks with their parameters.

During the Initial Track Finding, every hit in a three-dimensional image of the TRT detector is allowed to belond to a number of possible predefined tracks characterized by different parame-

1. A road is a list of modules ordered according to the radius at which they are placed starting from the innermost one.

ters. All such tracks are stores in a Look-Up Table (LUT). Every hit increases the probability that a track is a genuine candidate by one unit.

The next step consists of Local Maximum Finding. A two-dimensional histogram is filled with bins in ϕ and $1/p_T$. A histogram for a single track would consists of a "bow-tie" shaped region of bins with entries at a peak in the center of the region. The bin at the peak of the histogram will, in an ideal case, contain all the hits from the Track. The roads corresponding to other filled bins share straws with the peak bin, and thus contain suib-sets of the hits from the track. A histogram for a more complex event would consist of a superposition of entries from individual tracks. Hence, bins containing a complete set of points from each track can be identified as local maxima in the histogram.

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 2 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 reconstruted p_T threshold of $0.5\text{GeV}/c$ is applied.

13.3.3.4 TRTKalman

TRT-Kalman [13-26] is a new 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.

TRT-Kalman incorporates following modified modules from xKalman:

- `XK_Tracker_TRT`: This reads TRT geometry from from ROOT files. It uses `InDetDescr`, `InDetIdentifier` to access necessary Detector Description information;
- `XK_Algorithm`: A strategy is added to perform TRT standalone reconstruction;
- `XK_Track`: A step has been added with fine-tuning of track parameters after the histogramming step and Least Squares fit;
- `XKaTrtMan, XKaTRTRec`: This contains xKalman++ internal steering algorithms;
- `XKaTRTClusters`: This component retrieves `TRT_RDO_Container` from StoreGate filled from a ByteStream file.

13.3.3.5 T2Calo

T2Calo (see Refs. [13-27], [13-28], [13-29], [13-30]) is a clustering algorithm for electromagnetic (EM) showers, seeded by the LVL1 EM trigger RoI positions [13-31]. This algorithm can select isolated EM objects from jets using the cluster E_T and certain shower-shape quantities.

The RIOs are calibrated calorimeter cells (`LArCells` and `TileCells`), imported from the offline reconstruction. Both `LArCells` and `TileCells` have `CaloCell` as common base class. The output (`T2EMCluster`) is a specific LVL2 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 from the cell with highest energy in the second sampling of the EM calorimeter. This position (η_1, ϕ_1) is later refined in the second sampling by calculating the energy weighted position (η_c, ϕ_c) in a window of 3×7 cells (in $\eta \times \phi$) centered in (η_1, ϕ_1). As described in Ref. [13-28], the steps to perform the jet rejection are the following:

- In sampling 2, $R^{shape}_{\eta} = E_{3\times7} \,/\, E_{7\times7}$ is calculated. The expression $E_{n\times m}$ stands for the energy deposited in a window of $n \times m$ around (η_1, ϕ_1).
- In sampling 1, $R^{strip}_{\eta} = (E_{1st} - E_{2nd}) \,/\, (E_{1st} + E_{2nd})$ is obtained in a window of $\Delta\eta \times \Delta\phi = 0.125 \times 0.2$ around (η_c, ϕ_c). E_{1st} and E_{2nd} are the energies of the two highest local maxima found, obtained in a strip-by-strip basis. The two ϕ-bins are summed and only the scan in η is considered. A local maximum is defined as a single strip with energy greater than its two adjacent strips.
- The total transverse energy E_T deposited in the EM calorimeter is calculated in a window of 3×7 cells around (η_1, ϕ_1).
- Finally, the energy that leaks into the hadron calorimeter E^{had}_{T} 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 standalone LVL2 tracking algorithm for the Muon Spectrometer. In the past, it existed in the Reference software from ATRIG, and this version is described in detail elsewhere [13-34].

The program is steered by the RoI given by the LVL1 Muon Trigger and uses both RPCs and MDTs measurements. At present this algorithm is limited to the barrel region and it is based on four sequential steps:

1. LVL1 emulation; the muon pattern recognition in the MDT system is initiated by the RPC hits that induced the LVL1 trigger accept. Among these hits, only those related to the pivot plane (middle RPC station) are provided by the muon trigger processor; the ones related to the coincidence plane (innermost and outermost RPC stations) have to be identified running a fast algorithm that simulates the basic logic of the LVL1selection.
2. Pattern recognition: it is performed using the RPC hits that induced the LVL1 trigger 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 trajectory;
3. A straight-line track fit is made to the selected tubes (one per each tube monolayer) within each MDT station. For this procedure the drift-time measurements is used to fully exploit the high measurement accuracy of the muon tracking system. The track sagitta is then evaluated.
4. A fast p_T estimate is made using LUTs. The LUT encodes the linear relationship between the measured sagitta and the Q/p_T, as a function of eta and phi.

The output of this algorithm is the measurement of the muon transverse momentum p_T at the main vertex, eta and phi.

13.3.3.7 muComb

The combination of the features of the track measured in the Muon Spectrometer and the Inner Detector (ID) at LVL2 provides a rejection of π and K decays to μ and of fake muons induced by the cavern background. Moreover the combination of the two measurements improves the momentum resolution of reconstructed muons over a large momentum range.

The matching of the Muon Spectrometer tracks and of the ID can be performed extrapolating the ID track to the muon system. The procedure needs to take into account the detector geometry, the material composition 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 is 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, the materials and of the magnetic field have been described by simple analytic functions of eta and phi. The extrapolation of the ID tracks to the entrance of the Muon Spectrometer is performed using linear extrapolation in two independent projections: the transverse and the longitudinal views. Two coordinates are extrapolated: the z-coordinate and the azimuthal angle phi. The linear extrapolation is corrected using average corrections. In the transverse projection the ID track extrapolation in phi is corrected as follows:

$$\Delta\varphi = \frac{\alpha}{p_T - p_T^0} \qquad \text{13-1}$$

where α is related to the field integral and p_T^0 allows for the transverse energy loss in the material of the calorimeter, that is approximately independent of the track transverse momentum p_T. Both alpha and p_T^0 have been determined by fitting $\Delta\varphi$ of simulated muons as a function of p_T. It is found that $p_T^0 \sim 1.5$, *i.e.* about half of the transverse energy loss of low energy muons, as naively expected. A similar approach has been followed in the case of the extrapolation of the z-coordinate in the longitudinal view.

The matching is done geometrically using cuts on the residuals in each of z and phi.

For matching tracks the combined transverse muon momentum is estimated through a weighted average of the independent p_T measurements in the Muon Spectrometer and in the Inner Detector. For each combined track, a χ^2 parameter is used to evaluate the quality of the p_T matching. Thanks to the high quality of the muon p_T measurements in both detectors, secondary muons from π and K decays give typically bad χ^2 matching, and thus can be rejected.

13.3.4 HLT Algorithms for EF

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/*c*. A more detailed description of this algorithm is available elsewhere [13-35].

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 number of sufficient clusters.

The primary track finding in the Pixels or SCT outputs a set of SpacePoints as an initial trajectory estimation. In the next step these set of space points serve as an input for the Kalman filter-smoother formalism that will add the information from the precision layers. Each reconstructed track is then extrapolated back 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 mechanism available in the offline environment: `XKaSeedsAll`, the reconstruction of the full event; `XKaSeedKINE` reconstruction of a region-of-interest and soon available EM calorimeter seeding. In the HLT environment 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 as `SimpleTrack` objects in a `SimpleTrackCollection`. The Track candidate contains the following information:

- Fit procedure used (m-fit or e-fit);
- 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 $\chi 2$ resulting from final fit procedure;
- List of all hits on track from all sub detectors;
- Total number of precision hits Np.
- Total number of straw hits Ns, empty straws crossed Ne, and of drift-time hits Nt.
- 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 $Ns/(Ns+Ne)$ is larger than 0.7 to 0.8;
- The ratio $Nt/N\backslash s$ is larger than 0.5 to 0.7;

- No previously accepted track has the same set of hits as the current one; this last cut removes full *ghost tracks*.

13.3.4.2 iPatRec

A detailed description of iPatRec is available elsewhere [13-36].

[Need text here.]

13.3.4.3 LArClusterRec

LArClusterRec is the reconstruction package for electromagnetic clusters in the calorimeter.

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 input of 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. In case a local maximum is found with a total energy in the window above a given transverse energy threshold, clusters are created which are 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 and leakage outside the cluster in a given window. In the region between the barrel and end-cap calorimeter 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, which is a geometrical effect. The ϕ position is corrected for an offset, which is also a geometry effect.

13.3.4.4 egammaRec

EgammaRec is designed to calculate useful quantities to separate clusters in the electromagentic calorimeter from jets. To do so, electromagnetic cluster information as well as tracking information is 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 `EMShowerBuilder` which calls different algorithms which calculate diverse quantities using the information in the first and second sampling of the electromagnetic calorimeter as well as the leakage into the first sampling of the hadronic calorimeter.

Cluster and track information is combined in the `TrackMatchBuilder`. For a given cluster all tracks are examined in the given 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 trackmatch 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 shapes, tight track quality cuts, E/p, and the position match in η and ϕ between the cluster and the tracks. Photons can be selected by analysing the EM shower shapes,

1. This window can be different from the one used for the sliding window algorithm.

reconstruction of conversions in the Inner Detector, and possibly a track veto for non-converted photons.

13.3.4.5 Moore

Moore (Muon Object Oriented Reconstruction) is a track reconstruction package for the Muon Spectrometer. A detailed description of Moore is available elsewhere [13-37].

Moore takes as input collections of digits or clusters inside the Muon Spectrometer (CSC, MDT, RPC, TGC) and outputs fitted reconstructed tracks whose parameters are expressed at the entrance of the muon spectrometer.

The reconstruction is performed in several steps and each step is driven by an Algorithm module, `MooMakeXXX`. Each algorithm is independent (*i.e.*, it retrieves objects created by the previous modules from StoreGate and it builds a transient object to be recorded in StoreGate where it is available for the subsequent algorithms). The only link between algorithms are the transient objects, in such a way that the algorithms depend on transient objects but transient objects do not depend on algorithms. The decoupling between data and algorithms and the natural step sequence of algorithm performing the reconstruction gives the opportunity to plug-in different reconstruction algorithms at run time.

As it is now, the overall reconstruction starts from the searches for ϕ regions of activity and builds `PhiSegments` (`MooMakePhiSegments`). For each ϕ-Segment, the associated MDTs are found and a *crude* `RZSegment` is built (this is essentially a collection of z hits) (`MooMakeRZSegments`).

Inside the MDTs the drift distance is calculated from the drift time, by applying various corrections: such as the TOF, the second coordinate, the propagation along the wire, the Lorenz effect. From the 4 tangential lines the best one is found. All the MDT segments of the outer station are combined with those of the Middle layer. The MDT hits of each combination are added to the phi-hits of the ϕ Segment, forming outer track candidates. All the successfully fitted candidates are kept for further processing (`MooMakeRoads`).

The successful outer track is subsequentely used to associate inner station MDT hits. A final track is defined as a successfully fitted collection of trigger hits and MDT hits from at least two layers (`MooMakeTracks`). The parameters of the fitted track are referred to the first measured point and are therefore expressed at the entrance of the Muon Spectrometer.

When dealing with data already selected by the trigger the first two steps (`MooMakePhiSegments`) and (`MooMakeRZSegments`) can be substitute with *ad hoc* makers that seed the track search in the regions selected by the trigger.

13.4 Signatures, rates and efficiencies

In the following subsections, the physics performance of algorithms for LVL2 and EF is summarized for five final-state classes: electrons and photons; muons; jets, taus and missing ET; b-jets; and B-physics. This broad classification stems from the physics goals of the ATLAS experiment, as explained in Chapter 4. Whenever possible, results will include the realistic use of data formats and associated converters (as described in previous section), steering control (as described in Chapter 9), 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. Except for the case of B physics, reconstruction of exclusive decays is not required and no topological variables (e.g. the calculation of invariant masses from a combination of several high-p_T objects) are used in the selection, although this is technically feasible at LVL2 or in particular in the EF (e.g. to select $Z \rightarrow l^+l^-$ decays exclusively).

It is worthwhile noticing that system performance (*e.g.* execution time, amount of data needed) is one of the major requirement in the HLT selection, to comply with the constraints imposed by the on-line environment and resources. In this chapter an indication of the compliance with those requirements will be given for the most important selections, whilst a detailed analysis of the different contributions to those figures will be given in Chapter 15. In general, all results have been achieved by optimizing concurrently physics and system performances.

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 from signal events from Standard Model physics processes containing real isolated electrons or photons ($W \rightarrow ev$, $Z \rightarrow ee$, direct photon production, etc.).

The electron and photon triggers can be viewed as a series of selection steps of increasing complexity. 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 and detailed calibration (see Section [13.3.3.5]). This selection is based on 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 LVL2 photon trigger object.

Electrons are identified at LVL2 by associating the e.m. cluster with a track in the Inner Detector. This association can be as simple as requiring the presence of a track with a minimum p_T in the e.m. RoI, but may, in addition, require position and momentum matching between the track and the cluster. Typically, 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 used for the studies presented here are described in Sections [13.3.3.1], [13.3.3.2], [13.3.3.4].

As currently planned by the HLT scheme, the EF will select events using as far as possible the algorithms of the ATLAS offline reconstruction system, which implies these algorithms have to comply with the stricter EF requirements in terms of robustness and system performance. Currently this is not yet achieved, however, work is in progress to change the algorithms accordingly. The present study uses the currently available ATLAS offline reconstruction software as discussed in Section [13.3.4] as a prototype of the future EF code. The criteria to identify electrons and photons need to be softer at the EF level in order not to loose events prematurely. In previous studies [13-42] and [13-44], the offline electron and photon selection has been applied using the same identification criteria as the offline selection just leaving out few "critical" criteria. For example a track veto for non-converted photons has not been applied on the EF level because is requires a 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.The EF algorithm components (calorimetry, tracking and particle identification) are treated in a similar way as for LVL2.

The main differences with respect to LVL2 derive from the availability at the EF of more detailed calibrations and more sophisticated algorithms with access to the full-event data. The improved performance results in sharper thresholds and better background rejection. In the case of electrons, bremsstrahlung recovery will be performed for the first time at the EF. In addition, a photon-conversion recovery procedure will be applied to photon candidates at the EF.

In the following the system and physics performance of the selection of electrons by the HLT will be reviewed in detail. The photon selection is not discussed here. These studies are currently in progress and no results are avialable yet. The physics performance of the electron and photon selection has been already studied in detail in the past by the HLT and are reported in [13-41]- [13-44]. The system performance part of this selection is discussed in ***Chapter 14.***

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 performance has been characterized in terms of efficiency for the signal channel, rate expected for the selection and algorithm execution time. The rates shown in this and in the following sections have been obtained using a sample of simulated di-jet events with pile-up added for the low and design luminosity scenario (see Ref. [13-45]). Compared to previous studies a more up to date detector geometry has been used and pile-up effects for the low luminosity scenario of 2×10^{33} cm^{-2} s^{-1} has been as well taken into account. In general, events with electrons and photons are selected on the basis of single high-p_T objects or of pairs of lower-p_T objects. The physics performance of the electron triggers is summarized here and documented in detail for the three trigger levels in Ref. [13-46] and Ref. [13-47]. An overview can be found in table [13-1].

Table 13-1 Performance of the isolated electron HLT trigger at design and low luminosity for the single electron selection. The results are presented in a single sequence, except for the starting point of the LVL2 tracking, where two alternatives (TRT and Precision) are shown. 'Matching' refers to position and energy–momentum matching between calorimeter clusters and reconstructed tracks (at LVL2 both Precision and TRT tracks are used). The efficiencies are given for single electrons of p_T = 30 (25) GeV a design (low) luminosity over the full rapidity range $|\eta| < 2.5$. The efficiencies and rates are given with respect to a LVL1 output efficiency of 9x% (9x%) and a LVL1 rate for e.m. clusters of xxx kHz (xxx kHz). The timing results quoted here are for events from the di-jet sample and are scaled to correspond to a 4 GHz machine running Linux. The terms m_{50} and m_{95} are defined in [13.4.1.2]. The quoted errors are statistical.

	Design Luminosity			Low Luminosity		
Trigger Step	**Rate [Hz]**	**Efficiency [%]**	**Timing m_{50} / m_{95}**	**Rate [Hz]**	**Efficiency [%]**	**Timing m_{50} / m_{95}**
LVL2 Calo	3490 ±± 160	97.1 ± 0.3	0.20 / 0.26 ms	1100 ± 30	96.0 ± 0.6	0.15 / 0.23 ms
LVL2 Precision	620 ± 70	90.3 ± 0.6	6.2 / 12.7 ms	150 ± 11	92.4 ± 0.8	2.4 / 5.8 ms
LVL2 TRT	1360 ± 100	89.7 ± 0.6	0.4 / 1.2 s	360 ± 17	89.2 ± 0.9	31 / 210 ms
LVL2 Matching	460 ± 60	85.3 ± 0.7	--	140 ± 11	88.1 ± 0.9	--
EF Calo	313 ± 50	83.5 ± 0.8	0.39 / 0.63 s	85 ± 8	86.4 ± 1.0	0.34 / 0.56 s
EF ID	149 ± 34	79.3 ± 0.8	11 / 71s	57 ± 7	82.4 ± 1.1	0.31 / 1.6 s
EF Matching	117 ± 30	77.6 ± 0.8	--	41 ± 6	80.8 ± 1.2	--

The performance of the single isolated electron HLT algorithm is summarized in table [13-1] as a function of the main steps in the LVL2–EF trigger chain. The trigger steps have been factorized by detector in order to show the overall computational load and rejection that each stage contributes to the trigger. The table shows that the input rate from the LVL2 electron trigger to the EF is xxx Hz (xxx Hz) at design (low) luminosity for a nominal p_T threshold of 30 GeV (25 GeV). The overall reduction in rate achieved by LVL2 is a factor of xx (xx) for a loss of efficiency of xx% (xx%) with respect to LVL1. The additional rate reduction provided by the EF amounts to a factor of xxx (xxx) for a relative efficiency loss of xxx% (xxx%). Using the offline electron selection the rate is reduced by xxx (xxx)%, for an additional loss of xx (xx)% in efficiency. This shows that the HLT selection is very powerful. The LVL2 selection has an efficiency of 9x% (9x%) for the events selected by the EF alone, and the additional loss of events is mostly due to the fast track selection at LVL2, showing the expected correlation of inefficiencies at the LVL2 and EF stages (e.g. due to bremsstrahlung). Compared to previous results (see Ref. [13-44]) the rates quoted here are higher. This is partly due to the looser selection on the EF level, but mainly due to the increase of material in the inner detector, resulting in more electons which undergo bremsstrahlung effects. Especially due to the new insertable layout of the pixel b-layer more material is found near the beampipe. Hard bremsstrahlungs effects occuring in this material cannot been identified in the tracking system and thus is unrecuperable. Due to these effects the rate has gone up by approximately a factor of 2 (1st guess, to be confirmed).

At low luminosity, the events remaining after the HLT electron selection consist of $W \rightarrow e\nu$ decays (xx ± x)%, isolated electrons from (b,c) → eX decays (xx ± x)% and background from high-p_T photon conversions and misidentified hadrons (xx ± x)%. At design luminosity, where a higher p_T threshold is applied, the corresponding proportions are (xx ± x)%, (xx ± x)% and (xx ± x)%. The quoted errors are the statistical uncertainties on the estimates. As seen around 30% (update later) of the selected events at the trigger level contain ‘real’ electrons, hence a further improvement of this selection can only be small.

Electron decays of the W are selected by the EF with an efficiency of (xx ± x)% at low luminosity and (xx ± x)% at design luminosity, in agreement with the values given in table [13-1] for single electrons of 25 GeV and 30 GeV transverse momentum respectively. Finally, as an example of the performance for a physics signal, the HLT selection efficiency (using both the single- and the double-electron trigger) for the decay H(130) → 4e is (9x ± x)% at low and (9x ± x)% at design luminosity. For Z→ ee events the efficiency is (9x ± x)% ((9x ± x)%) at low (design) luminosity. These high efficiencies are due to the large electron multiplicity in the final state.

13.4.1.2 HLT Electron/Photon Algorithm Optimization

The algorithm execution time has been measured in order to study the resource constraints they may place on the overall HLT / DAQ system. This exploratory study addresses the interplay between the physics and the system performance aspects. Timing measurements were carried out on the feature-extraction part of the algorithms, excluding as much as possible any I / O (data read / write), and thus characterizing the most computationally complex aspects of the algorithms. In order to assess the impact of tails on the timing results, the measurements are given in terms of the median (m_{50}) and the latency within which 95% of the events are processed (m_{95})[1].

1. The timing measurements were carried out on several different platforms, but have been converted to the same overall scale, corresponding to a 4 GHz Pentium PC equivalent.

To understand where the computing resources are being used in the trigger, the different parts of the LVL2 and EF algorithms have been profiled in the test-bed studies, which are summarized in **Chapter 15** and as well given in table [13-1]. The time concuming components in the selection chain have been identified and work has started to improve the system performance of theis part. It should be noted taht the current offline algorithms were not aimed at at running on the EF at the time they were developed. Significan progress to run on the EF level and speeding-up of the programs is expected in the near future. There are still quite some possibilities to speed up the execution time and the numbers given in section xxx will improve with time. To give an example one possibility to speed up the code is to optimize different algorithm parameters. Here the aim is to eliminate any resource-consuming tasks that contribute only marginally to the rejection. Similar studies have been performed for the EF. As an example, Figure xxx shows the execution-time dependence of the EF electron-tracking algorithm on the transverse-energy threshold of the calorimeter cluster used to *seed* the reconstruction. (The seed energy scale does not correspond to the calibrated electron energy scale.) Increasing the threshold, thus reducing the number of seeds, reduces the execution time (in particular m_{95}) with a negligible impact on the physics performance.

The present system performance of the electron / photon algorithms can be improved at all levels of the HLT. There are studies under way which will be documented in future TDAQ notes.

13.4.1.3 HLT Strategy and the LVL2–EF Boundary

The use of system resources in the electron HLT can be minimized by exploiting the modularity of the trigger. By ordering the trigger steps in such a way that events are rejected as early as possible, both overall processing times and data transfers are reduced. Factorizing the trigger algorithm components also provides flexibility to move the rejection power from LVL2 to the EF or vice versa, to optimize the following: the performance of the implementation of the algorithm; the robustness of the selection with respect to the rate; the load implied at each level; etc. These issues have been studies in the past and are reported in Ref. [13-41]. As an example, figure xxx shows that an increase in efficiency can be obtained, with a modest increase in the total HLT output rate, by moving the whole LVL2 tracking selection to the EF. However, in this case, the input rate to the EF would increase by a factor of about eight, with important consequences on the computing load on the EF.

An important aspect of optimizing the sharing of rejection between LVL2 and the EF is the determination of the rejection contributed by each trigger level at the same efficiency. After tuning the LVL2 and EF electron selections to yield the same efficiency for events selected by LVL1, the EF contribution to the total reduction in rate is still better than LVL2 by a factor of two (three) at design (low) luminosity.

In case the incoming trigger rate is too high and needs to be reduced two obvious ways to do so is either to raise the energy threshold of the trigger menu item or by stricter selection criteria. All of these will imply an additional loss in efficiency for physics signals. Part of this loss in physics can then be recovered by more selective triggers. The preferred and easiest way to reduce the rate is to raise the energy thresholds. The LVL1 rate is dominated by the contribution from single high-p_T e.m. objects. As an example, raising the thresholds by E_T=5 GeV of the single electron trigger would yield in a final HLT rate of xxx (xxx) at low (design) luminosity. This is also seen in Figure xxx, which illustrates the impact of raising the threshold for the single-electron HLT selection only (nominal threshold of 30 GeV), while keeping the double-electron trigger threshold at its nominal value (20 GeV for each electron). The upper plot indicates the reduction in rate for the sum of the single- and the double-electron trigger contributions. As the

threshold is increased, besides the reduction of fake electrons, also the contribution from real $W \rightarrow e\nu$ decays is gradually rejected. Figure yyy illustrates the impact on this physics signal, as well as for $Z \rightarrow e^+e^-$ decays: for thresholds below 35 GeV, the efficiency for Z's is only slightly reduced. Decays with more than two electrons are affected even less, e.g. in the case of $H(130) \rightarrow 4e$.

As illustrated above, the proposed strategy contains considerable flexibility. Various possibilities exist to reduce the required computing resources or to improve the physics performance. For many channels of interest, the selection scheme also provides considerable redundancy. More details on the trigger selection strategy is given in **Chapter 4**.

13.4.2 Muon selection

The main purpose of the high-level muon trigger is the accurate reconstruction of muon tracks in the RoIs indicated by the LVL1 muon trigger. LVL2 and EF must reject low-p_T muons, secondary muons produced in the in flight decays of charged pions and kaons and fake muons originating from the cavern background. The EF must be able to reconstruct additional muons present in the event not reconstructed or selected by the LVL2 trigger.

Whilst the LVL1 trigger system uses only hits from the dedicated trigger detectors (RPCs in the barrel and TGCs in the endcap), the LVL2 and EF has access to the full measurements of the Muon Spectrometer, including in particular the data from the Monitored Drift Tubes (MDTs). This allows the best muon track reconstruction. The high background environment in the Muon Spectrometer demands algorithms with robust and fast pattern recognition capable of rejecting hist induced by the cavern background.

The tracks found in the LVL2 Muon Trigger are extrapolated for combination with the Inner Detector and the Calorimeter. Matching between muon tracks measured independently in the Muon System and the Inner Detector selects prompt muons and reject fake and secondary muons. This is important in particular for the B-physics trigger in low-luminosity running, for which the selection of prompt low-p_T muons events defines the input of the B-physics trigger algorithm.

The studies presented in this section are limited to the barrel region ($|\eta| < 1$).

13.4.2.1 The Physics Performances of LVL2 Muon algorithms

The p_T resolution of reconstructed muons is crucial to the selection efficiency and to the rejection of low p_T tracks that can be achieved at LVL2. The distribution of $(1/p_T^{muon} - 1/p_T^{true})/(1/p_T^{true})$ obtained by the muFast algorithm is shown in figure [figure TP 8-5] for p_T=6 GeV/c. The non Gaussian tails arise largely from the presence of soft particles produced by the muon interacting with the material of the detector.

The p_T resolution of the muFast algorithm is shown as a function of p_T in figure [figure TP 8-7]. As shown in the figure, the resolution ranges between 4.0% and 5.5% for muon in the p_T interval 6-20 GeV/c. These results are well compared with the transverse momentum resolution obtained by the offline muon reconstruction program MUONBOX.

The selection efficiency of muFast for selecting prompt single muons at 6 GeV/c and 20 GeV/c thresholds, relative to muons accepted by the LVL1 muon trigger, are shown in figure [figure TP

8-9]. For a nominal threshold of 6 GeV/c, the efficiency is about 90%, including detector acceptance. This efficiency is 95% for the 20 GeV threshold.

The total rates after this algorithm, including the rejection provided by the LVL1 selection, have been evaluated by convolving the algorithm efficiency as a function of p_T with the muon differential cross section production of the dominant physics processes. Where the available statistics are too low (in particular for the high-p_T rate calculation) to evaluate the efficiency, the lowest p_T at which an efficiency estimate has ben possible (p_T=10 GeV/c) is assumed conservatively to constitute a plateau extending down to the lower limit of the p_T acceptance (p_T=3 GeV/c in the barrel). The rates from π/K decays are calculated using the predicted cross-sections from the DMPJET program, and would be lower by about 50% if the PYTHIA prediction were used.

Table 13-2 Total output rates [kHz] of the LVL2 muon trigger after application of the mu rithm for the 6 GeV/c low-p_T threshold at low and 20 GeV threshold at the design luminosity.

Physics Process	low-p_T	high-p_T
p/K decays	3.1	0.06
b decays	1.0	0.09
c decays	0.5	0.05
W→μν	negligible	0.05
cavern background	negligible	negligit
Total	**4.6**	**0.24**

The total rates after LVL2 are shown in Table 13-2.

[THE ABOVE NUMBERS NEEDS TO BE CHECKED].

Preliminary studies of the trigger rate arising from the cavern background as predicted by the FLUKA package have been done. The probability that a fake LVL1 muon trigger is accepted by the LVL2 is below 10^{-2}. This upper limit is sufficient to neglect the contribution from fake muons.

Preliminary studies have been made to evaluate the physics performances of the muComb algorithm. Figure [figure TP 8-10] shows the combined reconstruction efficiency of prompt and secondary muons, as a function of the muon p_T, where the standalone codes from muFast and the LVL2 Precision algorithm [reference to the related chapter] have been used. The requirement of a good muon track matching (z/phi and p_T matching) reduces the low p_T trigger rate to 1.0 kHz: a factor three reduction compared to the rate from the muFast algorithm. Including the further reduction in rate due to the increase in p_T resolution for prompt muons, the total rate from the muComb algorithm is 2.1 kHz from muons with p_T>6 GeV/c.

13.4.2.2 The Physics Performances of the Muon Event Filter

in preparation

13.4.2.3 The Timing Performances of the Muon Algorithms

The muFast trigger algorithm has been benchmarked on several processor. On a processor corresponding to 10 SPECint95, muFast takes ~2ms/RoI, fairly independent from the trigger threshold and the muon p_T analyzed. If the data access is taken into account the time increases to XX ms.

A realistic evaluation of the time needed to the LVL2 muon trigger to take a decision has to take into account the time needed to move the data from the RoBs to the LVL2 processor in the AT-

LAS TDAQ architecture. Testbed measurements have been performed and indicates that the global time taken from the LVL2 trigger is about YY ms.

13.4.3 Tau/jets/$E_T$miss

A major Standard Model source of tau leptons in ATLAS will arise from *W/Z* decay sources: $pp \rightarrow W \rightarrow \tau\nu$ ($\sigma \sim 19$ nb) and $pp \rightarrow Z \rightarrow \tau\tau$ ($\sigma \sim 3$ nb). The overall tau production rate from this source is of the order of 10 Hz, at low luminosity. The tau lepton will also play a key role in the search for new physics. For example, in the MSSM the heavy scalar (H) and pseudoscalar (A) Higgs boson decays to a tau-pair are strongly enhanced with respect to Standard Model Higgs boson case. The dominant process in the high $\tan\beta$ (≥ 15) region is gg, $q\bar{q} \rightarrow b\bar{b}A/H \rightarrow \tau\tau$ and for low values of $\tan\beta$ rates for $gg \rightarrow A/H \rightarrow \tau\tau$ are dominant. Also, a key decay channel for the charged Higgs boson is $H^{\pm} \rightarrow \tau\nu$. In minimal SUGRA models the lighter tau slepton is expected to be the second lightest superparticle over a large parameter range at high $\tan\beta$. Consequently, one expects a viable SUGRA signal involving taus originating from tau slepton decay.

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. At LVL1 the tau trigger described above would have similar inputs and much of the same logic, as the electron / photon trigger.

Missing transverse energy will provide a distinct and important signature for new physics at the LHC. Many elements of physics beyond the Standard Model e.g the production and decay of SUSY particles, the production and decay of the Higgs boson, require a good measurement of E_T-miss. One example is the possible production of the A / H bosons which then decays via the process, $A/H \rightarrow \tau\tau$. A precise and reliable measurement of E_T-miss requires good calorimeter performance and energy resolution, good linearity of response and hermetic coverage.

One of the basic building blocks of the LVL1 calorimeter trigger is the E_T sum in the calorimetry. The total scalar E_T, as well as its components , are computed in the Jet / Energy sum processor of the calorimeter trigger. An E_T-miss trigger is not implemented in the basic LVL1 inclusive triggers. However, it is an important part of the trigger when placed in combination with the tau / hadron, electron photon, single jet triggers. A high level E_T-miss trigger is applied at the Event Filter level only, where access to the complete event, after calibration, is available. At this stage one can combine tau and E_T-miss triggers as one would in an offline analysis program, for example in the search for MSSM Higgs production in the channel $A/H \rightarrow \tau\tau$

(*Presumably there will be more or different words when I get feedback from Giacomo and the ETmiss group. JLP*)

13.4.3.1 The First Level Tau Trigger

The motivation for a LVL1 tau calorimeter trigger is threefold. First, it could improve the efficiency for triggering on the process $Z^0 \rightarrow \tau^+\tau^-$ or on low mass $A \rightarrow \tau^+\tau^-$ on decays, in coincidence with an electron or a muon trigger. Second, it could provide a trigger on $A \rightarrow \tau^+\tau^-$, $W \rightarrow \tau\nu$ and $Z \rightarrow \tau\tau$ decays, in coincidence with missing E_T. Third, using the measured momentum from the

tracking system, it could be used to select high-E_T hadronic tau decays for calibration of the hadron calorimeter. Narrow tau jets containing 1(3) tracks give rise to narrow isolated energy depositions in the calorimetry. It is envisaged that an isolation requirement will be a valuable part of the trigger for the first and second trigger types mentioned above.

The e/gamma/tau LVL1 algorithms are described in detail elsewhere ([13-38] and [13-39]). The LVL1 Tau/hadron calorimeter trigger is based on a 4 x 4 array of "trigger towers" in the electromagnetic and hadronic calorimetry (within the region $|\eta| < 2.5$) where the tower granularity is (0.1 x 0.1) $\Delta\eta$ x $\Delta\phi$. A core E_T is defined in the trigger algorithm as the sum of the electromagnetic and hadronic E_T in a 2 x 2 trigger tower. 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". The requirements for a window to be accepted as containing a valid trigger objects are: at least one of the four (1x2/2x1) trigger clusters passes the "e.m. cluster threshold"; the trigger cluster formed from 2x2 hadronic towers pass the "hadronic cluster threshold". E_T sums in the e.m.isolation and hadronic isolation regions are less than the corresponding isolation threshold; and, the centre 2 x 2 RoI cluster is a local "E_T maximum", i.e.is more energetic than the 8 neighbouring clusters of the same type contained in the 4 x 4 trigger window. The hadron isolation requirements are defined as follows. First, the total E_T in the e.m.isolation region is less than the 12-tower e.m. "ring" isolation threshold. second, the total E_T in the outer hadronic isolation region is less than the 12-tower hadronic "ring" isolation threshold. A schematic description of the first level tau/hadron trigger is given in *Figure 13-3.*

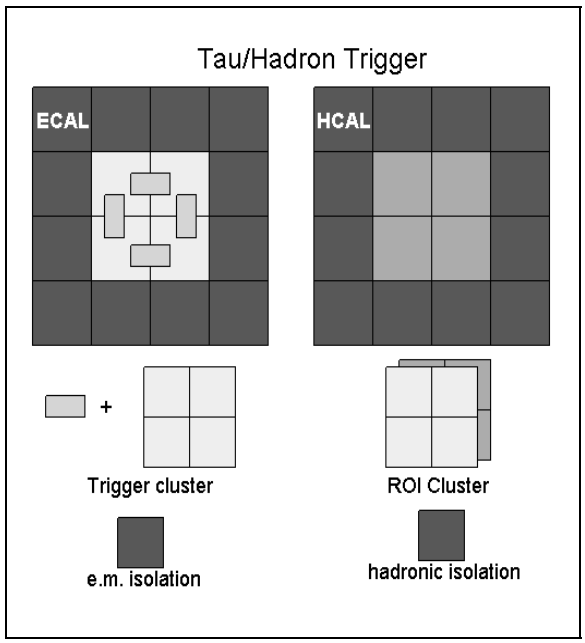

Figure 13-3 A schematic view of the tau/hadron LVL1 trigger algorithms

Although the tau/hadron LVL1 trigger its is very similar to the e/gamma slice LVL1 trigger there are two important differences First, there is 2x2 "hadronic cluster threshold" requirement not present in the e/gamma LVL1 trigger. Second, in the e/gamma LCVL1 trigger there is an inner hadronic, 2x2 tower, isolation threshold requirement that is no present in the tau/hadron trigger. The tau/electron trigger utilizes the full e/gamma slice machinery described previously. The signal selection is tuned using events of the type $Z^0 \rightarrow \tau^+\tau^-$. Background evaluation is performed using fully simulated di-jet events.

The LVL1 selection signatures studied will be (τXX + xEYY) (we will start with *the LVL1 trigger menu item is (τ25 + xE30) assuming that we have access to the LVL1 Etmiss trigger in the software.*) and (τZZ) (*we will try (τ50) initially*).

The results of the analysis detailing the precise cuts and isolation thresholds at LVL1 will be placed here in tabular form. JLP.

13.4.3.2 The Second Level Tau Trigger

The LVL2 tau trigger involves the verification of the LVL1 decision and tau identification using parameters that describe that describe the shower shape and the isolation of the narrow jet. Additional rejection of background jets can be achieved by using the information from tracks associated to the tau RoI. Again, The signal selection is tuned using events of the type $Z^0 \rightarrow \tau^+\tau^-$. Background evaluation is performed using fully simulated di-jet events. 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 XX GeV is required, and e.m.and hadronic isolation thresholds are set to XX GeV. Jet calibration is applied to the cells within the LVL1 RoI window. The energy weighted position of the tau-jet candidate ($\Delta\eta_\tau$ x $\Delta\phi_\tau$) 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 executed except that fine grained cell information is utilized and no threshold is applied to the trigger towers.At LVL1 this threshold is X GeV.

The next step in the LVL2 is to look at core energy and isolation. The performance of the algorithm is studied for several choices of e.m.core size: $\Delta\eta$ x $\Delta\phi$ = 0.1 x 0.1, 0.15 x 0.15, 0.2 x 0.2, where the core window is centred at the energy weighted position, ($\Delta\eta_\tau$ x $\Delta\phi_\tau$). The hadronic core size is chosen to be the same as LVL1 i.e. $\Delta\eta$ x $\Delta\phi$ = 0.2 x 0.2. Isolation windows and thresholds are defined separately for the e.m.or hadronic parts as the complement of the respective core and the 0.4 x 0.4 RoI region.In order to avoid the lose of high energy taus relative energy fractions were considered. The quantity considered here is fraction of e.m.energy in the core. A

(*A previous analysis [13-19] has required that: the e.m. plus hadronic transverse energy contained in a small core is required to be above 50 GeV; and, the fraction of e.m.energy in the core is required to be greater than 85%.Results of the study of performance as a function of the core size studied for various values of the integrated efficiency for jets, etc., etc will be included here. JLP.)*

The next stage of the LVL2 algorithm is to select tracks within a window of $\Delta\eta$ x $\Delta\phi$ = 0.4 x 0.4 centred at the tau cluster. Only tracks above a p_T threshold (X GeV or Y GeV) are used. We require exactly one track, or one to three tracks within the window. Assuming 100% tracking efficiency and requiring track p_T to be above X GeV and $1 \leq N_{trk} \leq 3$, the jet rate is reduced by approximately a factor of X.X, while keeping the tau efficiency close to XX%. The reduction of jets can reach approximately a factor of Y.Y when exactly one charged track is required, but in this case the tau efficiency is reduced to ZZ% Inefficiency in track finding reduces the jet rejection power as well as the tau efficiency. However, the effect is small for realistic values of track efficiency (~90%): about X% increase in jet rate and Y% decrease in tau efficiency. If a higher p_T cut is chosen, say, p_T>5 GeV, then the rejection power for jets is significantly diminished, with little effect on the tau efficiency. Thus, the capability to measure low p_T charged tracks at LVL2 will be important for tau / jet separation.

(*A table of tau and jet efficiencies after a cut on the number of generated charged tracks, when applied to LVL1 RoIs and after LVL2 tau selection will be given here. JLP*)

Electron identification at LVL2 is described previously in section xxx of this document. It is based upon the matching of a track reconstructed in the inner detector, including TRT hits, with the electron cluster in the calorimetry.This machinery will be used to identify tau / electron jets at LVL2.

13.4.3.3 Tau Selection in the Event Filter

At the event filter stage we have access to the complete, calibrated, event for the first time. Thus, it is possible to refine the LVL2 decision. Existing off-line studies of tau / hadron identification and jet rejection [13-40] provide the basis for the event filter (EF) trigger decision. The trigger criteria for tau / hadron jets with E_T > XX 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_{em} < 0.0X (*0.07 in [13-40]*);
- The difference between the E_T contained in cones of size ΔR =0.X and 0.Y (*0.2 & 0.1 in [13-40]*),, normalized to the total jet E_T, ΔE_T must obey the inequality: ΔE_T < 0.Z (*0.1 in [13-40]*);
- The number of reconstructed charge tracks N_{tr} is equal to 1 or 3, pointing to the cluster (within ΔR =0.X)

(*In the case of the tau/hadron trigger three additional tau selection requirements will be studied (if there is time) Tracks must be extrapolated to "hit" a hadronic cluster i.e. must satisfy R < 0.X (R^2= $\Delta\phi^2$ +$\Delta\eta^2$), where R is the "distance between the cluster centre and the extrapolated track position at the hadron calorimeter; 2) All tracks must be consistent with originating from the same event vertex as the cluster. That is, Δz, between the distance of closest approach of the hadronic shower direction and the track at the closest approach to the beam obeys the inequality: Δz < xx cm; 3) There must be rough agreement between the sum of the hadronic E_T measured in the calorimetry and the sum of p_T measured in the tracking. JLP*

In the case of the tau / lepton trigger the cuts employed are: The p_T of the lepton is required to be greater than XX GeV and $|\eta|$ < 2.5.

A method of improving the signal acceptance for final states involving taus as well as retaining an acceptable trigger rate is combining a tau trigger with an E_T-miss trigger. The corresponding HLT menu item is ((τ35+xE45). Using XXXXX di-jet events the probability for a QCD jet to satisfy the tau identification criteria is studied. The corresponding tau identification efficiency was investigated using using $Z^0 \rightarrow \tau^+\tau^-$ produced at low and high luminosity.

(*The results of the studies discussed above will be presented here. JLP*)

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 multi 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 lower jet E_T thresholds than those discussed in section XXX is feasible) or reduce the background (and hence the rate) for events containing b-jtes that have already been slected by other triggers.

The study presented in this section defines and characterizes, in the low and in the high luminosity case, an online b-tagging selection based on the information coming from the Inner Detector.

13.4.4.1 b-tagging at LVL2

The track reconstruction and the precise determination of the track parameters (in particular in the transverse plane d_0) are the crucial components for the b-jet trigger.

Several tracking algorithm for LVL2 have been presented in Section YYY. Table 12-1 shows the comparison of the track reconstruction efficiency, the track parameters resolution and the latency among the different algorithms for RoIs generated by b-jets in the low and high luminosity case.

Table 13-3 Comparison of the track efficiency, the track parameter resolution and the time latency for different tracking algorithm in low (high) luminosity case.

	IDSCAN	**SiTrack**	**...**
Track efficiency	xx(yy)	xx(yy)	xx(yy)
Fake track fraction	xx(yy)	xx(yy)	xx(yy)
$\sigma(d_0)$	xx(yy)	xx(yy)	xx(yy)
$\sigma(\phi)$	xx(yy)	xx(yy)	xx(yy)
$\sigma(p_T)$	xx(yy)	xx(yy)	xx(yy)
$\sigma(\eta)$	xx(yy)	xx(yy)	xx(yy)
$\sigma(z_0)$	xx(yy)	xx(yy)	xx(yy)
Execution time	xx(yy)	xx(yy)	xx(yy)

The b-tagging algorithm is based on the significance of the transverse impact parameter $S=d_0/\sigma(p_T)$. A b-jet estimator is build using the likelihood-ratio method: for each track (i), the ratio of the probability densities for the track to come from a b-jet or a u-jet is calculated: $f_b(S_i)/f_u(S_i)$; the product W of these ratios over all reconstructed tracks in the jet is computed and the final tagging variable $X=W/(1+W)$ is defined. Jets are tagged as b-jets if X~1 and u-jets if X~0.

13.4.4.2 Results on single b-jet tagging

The b-tagging algorithm has been characterized on single b-jets coming from H -> b $\overline{b}$ decays with m_H=120 GeV produced in association with a W at low and high luminosity, and corresponding u-jets obtained by artificially replacing the b-jets from the Higgs decay. The E_T spectrum of these jets covers the range up to E_T=120 GeV; they provide a good benchmark for many physics channels involving Higgs production.

The efficiencies for b-jets (ε_b) and rejection factors (R_u) against u-jets are given in Table 12-2.

Table 13-4 Rejection of the LVL2 b-tagging algorithm against u-jets for three different values of the b-jet efficiency: 50%(top), 60 %(middle) and 70 %(bottom). The results are shown for different intervals of E_T, η of the jet.

	E_T < 40 GeV	**40 GeV < E_T < 80 GeV**	**80 GeV < E_T < 120 GeV**
$\lvert\eta\rvert < 1.5$	xx	xx	xx
$\lvert\eta\rvert > 1.5$	xx	xx	xx

13.4.4.3 Comparison with Offline b-tagging

The performance of the LVL2 trigger algorithm has been compared to that of the offline algorithm.

The Figure 12-X demonstrates that the trigger and offline selection are well correlated and that, as long as the LVL2 efficiency is kept above XX%,it is possible to provide subsequent analyses with an unbiased sample in the region ε<XX%.

Different combinations of working points of LVL2 trigger selection and offline analysis could be chosen depending on the required offline b-tagging efficiency.

13.4.5 B-physics

About one collision in every hundred will produce a *bb* quark pair. Therefore, in addition to rejecting non-*bb* events, the B-trigger must have the ability to identify and select those events containing B-decay channels of specific interest. Important areas include CP-violation studies with the channels $B_d \to \pi^+\pi^-$ and $B_d \to J/\psi K_s$ (with both $J/\psi \to e^+e^-$ and $J/\psi \to \mu^+\mu^-$); measurements of B_s oscillations in $B_s \to D_s\pi$ and $B_s \to D_s a_1$ with $D_s \to \pi\pi$; analysis of $B_s \to J/\psi$ and $B \to J/\psi\eta$; rare decays of the type $B_{d,s} \to \mu^+\mu^- X$; *b*-production measurements and precision measurements with *B*-hadrons. Since these are precision measurements and searches for rare decays, high statistics are required. The large number of *bb* pairs produced at the LHC mean that ATLAS is well placed to make a significant contribution in these areas.

Since the Technical Proposal the B-trigger has been re-assessed in the light of a number of developments, including the likelihood of a reduced ID layout at the start of running, an increase in the target start-up luminosity and various trigger deferral scenarios. The aim is to provide 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 start-up conditions are uncertain, luminosity is expected to vary from fill-to-fill, and there are 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 at the start of higher luminosity LHC fills and introducing further triggers later in the beam coast or for lower luminosity fills (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.

- Require a LVL1 JET or EM RoI in addition to a single-muon trigger (p_T>8 GeV). At LVL2 and the EF, tracks are reconstructed within RoI using pixel, SCT and TRT information. The reconstructed tracks form the basis of selections for e.g. J/ψ(ee), B($\pi\pi$) and $D_s(\phi\pi)$. Since track reconstruction is performed inside RoI, the resources required are modest.
- A full-scan of the SCT and pixels is performed for events with a single-muon trigger (p_T>8~GeV). The reconstructed tracks form the basis of selections for e.g. B($\pi\pi$) and $D_s(\phi\pi)$. This promises better efficiency than the above method, but requires somewhat greater resources in order to perform the full-scan.

In all cases, at least one LVL1 muon trigger is required to initiate the B-trigger. Since the cross-section for inclusive muon production from pion and kaon decays falls more rapidly with p_T than that for prompt muon production from *b*-decays, an appropriate choice of p_T threshold gives a powerful reduction of the trigger rate due to background processes. For example, a threshold of p_T > 8 GeV would gives a single-muon trigger rate of 10 kHz at LVL1 for a luminosity of $1 \times 10^{33}\,cm^{-2}\,s^{-1}$. Most of this rate is due to muons with true p_T below threshold originating from pion and kaon decay, a large proportion of which can be rejected at LVL2 on the basis of more precise track measurements. After the LVL2 selection the trigger rate is about 2~kHz at a luminosity of $1 \times 10^{33}\,cm^{-2}\,s^{-1}$; 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 and render the measurements un-competitive.

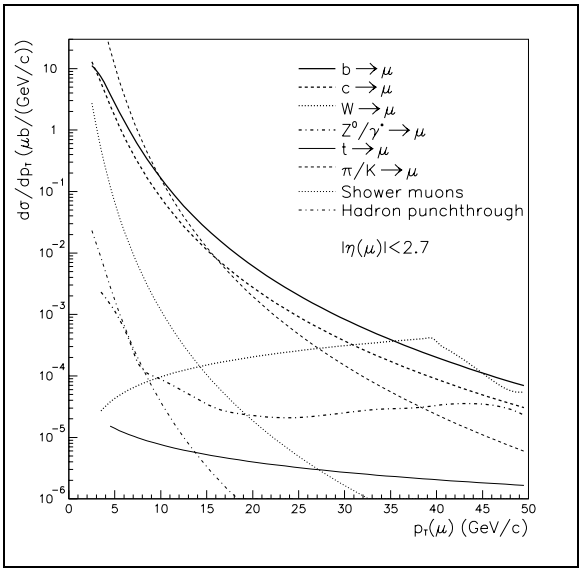

Figure 13-4 Differential cross-section ds/dp_T for inclusive muon production in ATLAS in the pseudo-rapidity range $|\eta| < 2.7$.

13.4.5.1 Di-muon triggers

A di-muon trigger provides a very effective selection for several important channels, e.g. B→J/ψ(μ⁺μ⁻)K_s and B→μ⁺μ⁻(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 600 Hz after LVL1 at a luminosity of $2 \times 10^{33}\,cm^{-2}\,s^{-1}$. These triggers are mostly due to muons from heavy flavour decays plus some single muons which are doubly counted due to overlaps in the end-cap trigger chambers. The later are removed when the muons are subsequently confirmed at LVL2 using information from the muon precision chambers and ID. At the EF, tracks are refit and specific selections made on the basis of mass and decay length cuts. These consist of semi-inclusive selections, for example to select J/ψ(μ⁺μ⁻) decays with a displaced vertex, and in some cases exclusive selections such as for $B_{d,s}$→μ⁺μ. The final trigger rate, after the EF, is about ~ 20 Hz at a luminosity of $2 \times 10^{33}\,cm^{-2}\,s^{-1}$.

13.4.5.2 Hadronic final states

For hadronic final states, two strategies have been studied based on, for events with a muon trigger, either an ID full-scan or a RoI-based selection. An ID full-scan consists of track-reconstruction within the entire volume of the SCT and Pixel detectors <Ref_idscan> and, optionally, the TRT <ref TRTscan>. The alternative 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, about 10% on average, there is potential for up to a factor of ten saving in execution time compared to the full-scan. Preliminary studies of efficiency and jet-cluster multiplicity have been made using a fast simulation which includes a detailed parametrization of the calorimeter. These studies indicate that a threshold of E_T > 5 GeV gives a reasonable jet cluster mul-

tiplicity, i.e. a mean of about two RoI per event for events containing a muon with $p_T > 6$ GeV, see Fig. <jet roi Mult>. A trigger based on this threshold would be efficient for $B \rightarrow D_s \pi$ and $B \rightarrow \pi\pi$ events with a *B*-hadron p_T above about 15 GeV.

Following the ID track reconstruction (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 refit at the EF inside RoI 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. At the EF, tighter mass cuts may be applied than at LVL2, due to the better track parameter resolution obtained from the EF reconstruction. In addition, EF selections may include decay vertex reconstruction, allowing further cuts on vertex-fit quality and decay length.

Studies using a full detector simulation have shown that an efficiency of about 70% 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 rate is about 60 Hz at LVL2 and about 6 Hz after the EF at a luminosity of 1×10^{33} cm^{-2} s^{-1}, using a single muon trigger threshold of $p_T > 8$~GeV. There is very little degradation of the trigger performance if the number of pixel layers is reduced from three to the two layers expected at the start of LHC running.

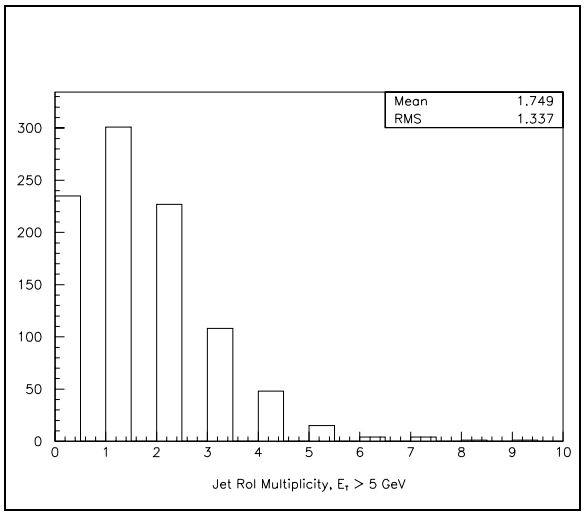

Figure 13-5 Jet RoI multiplicity ($E_T > 5$ GeV) for events with a muon $p_T > 6$ GeV.

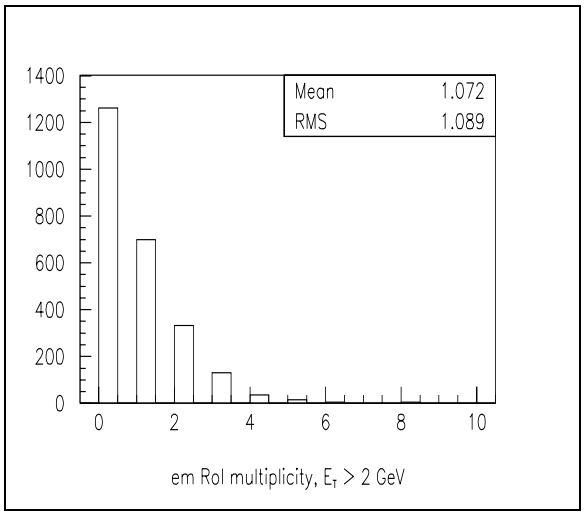

Figure 13-6 EM RoI multiplicity ($E_T > 2$ GeV) for events with a muon $p_T > 6$ GeV.

13.4.5.3 Muon-electron final states

A muon-electron trigger is used to select channels such as $B_d \rightarrow J/\psi(e^+e^-)K_s$ events with an opposite side muon tag, or $B_d \rightarrow J/\psi(\mu^+\mu^-)K_s$ with an opposite side electron tag. As for the trigger for hadronic final states, two different strategies have been studied using either an ID full-scan or RoI-based ID track reconstruction. In both cases a LVL1 muon trigger, confirmed at LVL2, is required.

For the full-scan based method, a histogramming technique <Ref_TRTLUT,Ref_xKalman> is used to search for tracks within the entire volume of the TRT. Good efficiency has been obtained for electrons with p_T down to about 1 GeV. However, since execution time scales as $1/p_T$, in practice higher thresholds may be used. To improve track parameter resolution, track candidates reconstructed by the TRT are then extrapolated into the SCT and pixels using a Kalman filter algorithm <Ref_SiKalman>. The TRT identifies e^+/e^- candidates on the basis of transition radiation information. Candidates passing track cuts are combined in opposite charge-sign pairs and $J/\psi(ee)$ mass cuts applied. An efficiency of about 40% can be obtained for $B_d \rightarrow J/\psi(e^+e^-)K_s$ events where both e^+ and e^- have $p_T > 1$~GeV. The corresponding LVL2 trigger rate is about 40 Hz, at a luminosity of $1 \times 10^{33}\ cm^{-2}\ s^{-1}$, using a $p_T > 8$~GeV muon trigger threshold. The tracks are refit at the EF, including a vertex fit. Decay length and fit quality cuts are applied, giving about a factor of ten further reduction in trigger rate.

An alternative strategy is based upon using the LVL1 trigger to find low E_T electron/photon clusters which define RoI 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 with a threshold of E_T>2 GeV. This gives a mean RoI multiplicity of about one for events containing a muon with $p_T > 6$~GeV, see Fig. <EM RoI mult>. The corresponding efficiency to create a RoI for both the e^+ and e^- from $J/\psi(e^+e^-)$ is about 80% 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. A search is then made, inside the RoI, for tracks in the SCT, Pixels and TRT. The RoI about each electron candidate can be quite small, of order $\Delta\eta \times \Delta\phi = 0.2\text{x}0.2$. This gives a large saving in reconstruction time, compared to a full-scan, but has a lower efficiency, particularly at low p_T.

13.4.5.4 Resource estimates

In order to estimate the computing resources required for the *B*-trigger, measurements of execution time are combined with estimates of trigger rate 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, and 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 multiplicity to give an estimate of the resources required for the B-trigger [13-51]. The results are shown in Table 13-5.

Table 13-5 B-trigger resource estimates.

Luminosity	B-Trigger	no. cpu
$2 \times 10^{33}\ cm^{-2}\ s^{-1}$	Di-muon only	2
$1 \times 10^{33}\ cm^{-2}\ s^{-1}$	Di-muon + RoI-based triggers	8
	Di-muon + fill-scan based triggers	26

The use of low E_T LVL1 RoI to guide reconstruction at LVL2 promises a full programme of B-physics for very modest resources. However multiplicities and efficiencies need to be verified in studies using a full detector simulation.

13.5 Event rates and size to off-line

Define present ideas about data compression and reduction, zero suppression for LAr (and TRT?): this might be probably be elsewhere as well. Differences between zeros at the EF and loss-less data compression in the ROSes.

Global table on rates for initial and high luminosity, implication for off-line reconstruction (costing, later)

13.6 Start-up scenario

Should be here? Picture a global approach on how we are going to handle, at the selection level, the first year of running, assuming a certain machine scenario. It is probably very appealing for LHCC

13.7 References

13-1 ATLAS detector and physics performance technical design report, CERN–LHCC/99–14/15 (1999)

13-2 ATLAS Collaboration, First-Level Trigger Technical Design Report, CERN/LHCC/98-14.

13-3 Schörner-Sadenius and T. Wengler, Formats and Interfaces in the Simulation of the ATLAS First Level Trigger, ATL-DA-ES-0029.

13-4 [M. Abolins et al., Specification of the LVL1 / LVL2 trigger interface, ATL-D-ES-0003.

13-5 E. Moyse et al., The ATLAS Level-1 Trigger Offline Simulation, ATL-COM-DAQ-2002-021.

13-6 http://xml.apache.org/xerces-c

13-7 E. Eisenhandler, Level-1 Calorimeter Trigger URD, ATL-DA-ES-0001.

13-8 A. Watson, Updates to the Level-1 e/gamma and tau/hadron Algorithms, ATL-DAQ-2000-046.

13-9 ATLAS Collaboration, more about the RPC detectors.

13-10 ATLAS collaboration, more about the RPC trigger.

13-11 http://atlas.web.cern.ch/Atlas/GROUPS/MUON/layout/muon_layout_P.html

13-12 A. Di Mattia et al., A muon trigger algorithm for Level-2 feature extraction, ATL-DAQ-2000-036; A. Di Mattia and L. Luminari, Performance of the Level-1 Trigger System in the ATLAS Muon Spectrometer Barrel, ATL-DAQ-2002-008.

13-13 ATLAS Collaboration, High-Level Trigger Technical Proposal, CERN/LHCC/2000-17.

13-14 A. Di Mattia, RPC Trigger Robustness: Status Report, ATL-DAQ-2002-015.

13-15 S. Veneziano, Preliminary Design Report of the MUCTPI Electronics, ATC-RD-ER-0001.

13-16 N. Ellis, Central Trigger Processor URD, ATL-DA-ES-0003; P. Farthouat, CTP Technical Specification, ATL-DA-ES-0006.

13-17 G. Schuler et al., Central Trigger Processor Demonstrator (CTPD), ATL-DA-ER-0005; I. Brawn et al., A Demonstrator for the ATLAS Level-1 Central Trigger Processor, ATL-DA-ER-0008.

13-18 R. Blair et al., ATLAS Second Level Trigger Prototype RoI Builder, ATL-D-ES-0011.

13-19 S. Armstrong et al., "Requirements for an Inner Detector Event Data Model", ATLAS-TDAQ-2002-011.

13-20 A.G. Mello, S. Armstrong, and S. Brandt, "Region-of-Interest Selection for ATLAS High Level Trigger and Offline Software Environments", ATLAS-COM-TDAQ-2003-005, ATLAS-COM-SOFT-2003-002.

13-21 PESA Core Algorithms Group, S. Armstrong editor, "Algorithms for the ATLAS High Level Trigger", ATLAS-COM-TDAQ-2003-00X.

13-22 K. Assamagan et al., "A Hierarchical Software Identifier Scheme," ATLAS-COM-MUON-2002-019.

13-23 C. Leggett and A. Schaffer, presentations at the ATLAS EDM-DD Workshop, 27 January 2003.

13-24 For more information on SCTKalman see I. Gaines, S. Gonzalez and S. Qian, in Proceedings of CHEP2000 (Padova); D. Candlin, R. Candlin and S. Qian, in Proceedings of CHEP01 (Beijing); J. Baines, et al. ATL-DAQ-2000-031.

13-25 J. Baines et al., "Global Pattern Recognition in the TRT for B-Physics in the ATLAS Trigger", ATLAS-TDAQ-99-012; M. Sessler and M. Smizanska, "Global Pattern Recognition in the TRT for the ATLAS LVL2 Trigger", ATLAS-TDAQ-98-120.

13-26 S. Sivoklokov, presentations made in PESA Core Algorithms Group meetings, December 2002 and January 2003. See also S. Sivoklokov, "High pT Level-2 Trigger Algortihm for the TRT Detector in ATRIG", ATLAS-TDAQ-2000-043.

13-27 M.P. Casado, S. González, and T. Shears, TrigT2Calo package, http://atlas-sw.cern.ch/cgi-bin/cvsweb.cgi/offline/Trigger/TrigAlgorithms/TrigT2Calo/

13-28 S. González, T. Hansl-Kozanecka, and M. Wielers, "Selection of high-pT electromagnetic clusters by the level-2 trigger of ATLAS," ATLAS-TDAQ-2000-002.

13-29 S. González, B. González Pineiro, and T. Shears, "First implementation of calorimeter FEX algorithms in the LVL2 reference software," ATLAS-TDAQ-2000-020.

13-30 S. González and T. Shears "Further studies and optimization of the level-2 trigger electron/photon FEX algorithm," ATLAS-TDAQ-2000-042.

13-31 ATLAS first level trigger technical design report, CERN-LHCC/98-14 (1998).

13-32 H. Drevermann and N. Konstantinidis, "Determination of the z position of primary interactions in ATLAS," ATLAS-TDAQ-2002-014.

13-33 H. Drevermann and N. Konstantinidis, "Hit FIltering and Grouping for Track Reconstruction," ATLAS-TDAQ-2003-XXX (Document in Preparation).

13-34 A. di Mattia et al. (Rome Level-2 Muon Trigger Group), "A Muon Trigger Algorithm for Level-2 Feature Extraction," ATLAS-DAQ-2000-036.

13-35 I. Gavrilenko, "Description of Global Pattern Recognition Program (xKalman)", ATLAS-INDET-97-165; also see: http://maupiti.lbl.gov/atlas/xkal/xkalmanpp/index.en.html.

13-36 R. Clifft and A. Poppleton, IPATREC: Inner Detector Pattern-Recognition and Track-Fitting, see http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/ DOCUMENTS/IPATREC/ipatrec.html.

13-37 The Moore Group, Moore – Muon OO REconstruction for ATLAS, see http://www.usatlas.bnl.gov/computing/software/moore/.

13-38 ATLAS Trigger Performance Status Report, CERN/LHCC 98-15, 25th August 1998

13-39 A. T. Watson, "Updates to the Level-1 e/gamma & tau/hadron Algorithms," ATL-DAQ-2000-046.

13-40 D. Cavalli and S. Resconi, "Combined Analysis of $A/H \rightarrow \tau\tau$ Events from Direct and Associated bbA Production," ATL-PHYS-2000-005., 22nd May 1998. D. Cavalli, L. Cozzi, L. Perini, S. Resconi, "Search for $A/H \rightarrow \tau\tau$ Decays", ATL-PHYS-94-051, 22nd December 1994. D. Cavalli and S. Resconi, "Tau Jet Separation in the ATLAS Detector", ATLAS PHYS-N0-118, 27th January 1998

13-41 ATLAS HLT, DAQ and DCS Technical Proposal, CERN-LHCC/2000-17

13-42 Performance studies for electron and photon selection at the event filter, ATLAS internal note ATL-DAQ-2000-007 (2000)

13-43 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)

13-44 Electron trigger performance studies for the event filter, ATLAS internal note, ATL-DAQ-2001-004 (2001)

13-45 ATLAS Data Challenge 1, see http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/DC/DC1/DC1-Report-V1.0-211002.pdf (will become ATL-SOFT note)

13-46 Physics Performance of the LVL1 EM triggers, to be written

13-47 Performance Studies of the HLT electron triggers, to be written

13-48 J.Baines et. al., B-Physics Event Selection for the ATLAS High Level Trigger, ATLAS Note ATL-DAQ-2000-031 (2000).

13-49 *Effects of Inner Detector Misalignment and Inefficiency on the ATLAS B-physics Trigger* by: J.Baines, B. Epp, S.George, V.M.Ghete, G.Hollyman, D.Hutchcroft, A.Nairz, S.Sivoklokov. ATL-DAQ-2001-006

13-50 *Event Filter Rate for the Ds Trigger* B. Epp, V.M.Ghete, A.Nairz. ATL-DAQ-2001-003

13-51 J.Baines et. al. Resource Estimates for the ATLAS B-physics Trigger ATLAS-COM-DAQ-2002-001

13-52 N.Konstantinidis and H.Dreverman, Fast tracking in hadron collider experiments, in Proceedings of the 7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Amer. Inst. Phys. Conference Proceedings, Vol. 583, 2001

13-53 J. Baines et al., Pattern Recognition in the TRT for the ATLAS B-Physics Trigger, ATLAS Note ATL-DAQ-99-007 (1999).

13-54 I. Gavrilenko, Description of Global Pattern Recognition Program (XKalman), ATLAS Note ATL-INDET-97-165 (1997).

13-55 P.Billoir and S.Qian, Simultaneous Pattern Recognition and Track Fitting by the Kalman Filtering Method, Nucl. Instr. and Meth. A225 (1990) 219.

14 Overall system performance and validation

14.1 Introduction

- Definition of validation of rate capability, its context and scope.

- Summary of validation process

14.2 Integrated Prototypes

Description of the prototypes:

- HLT / PESA prototype
- integrated 10% system

14.2.1 System performance of event selection

The High Level Trigger will select and classify events based on software largely developed in the offline environment. This approach minimizes duplication of effort 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 (“PESA software”) in a realistic trigger environment.

The resource utilization characteristics of the PESA 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.

14.2.1.1 Measurement and validation strategy

The scope of the work reported here is limited to a system with full event selection and a minimal dataflow system, 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 dataflow aspects relevant to PESA 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 may reveal weakness not foreseen in the original design. The integration and testing work described here followed, roughly, the following steps:

1. Individual component testing and validation (addressed in Chapters 8 and 13)
2. Functional integration of relevant components (e.g., Online, Dataflow, PESA) in a small testbed, providing feedback to developers.
3. Final system validation

4. Measurement program, including event processing times, network latencies, and thread scaling.

The last three steps were carried out for a LVL2 testbed, an EF testbed, and a combined HLT testbed in the context of validating the TDAQ architecture. The following sections summarize the outcome of this integration and measurement program.

14.2.1.2 Event selection at LVL2

All elements necessary to transport and process event data inside the L2PU were assembled in a LVL2 vertical slice prototype. As shown in Figure 14-2 (left), the following components were included in the prototype:

- L2PU (Described in Chapter 9.2.4)
- ROS or ROS emulator (Described in Section 8.2.2)
- LVL2 Supervisor (Described in Section 9.2.3)

The above were connected to the CERN network through a hybrid Fast/Gigabit ethernet switch, all controlled by the online software. The host machines for the applications were typically either Dual processor Pentium or Athlon machines (2.2 GHz) or single processor Pentium IV (2.4 GHz). A detailed description of the setup can be found in [14-2].

The L2PU, the application that effects the event selection, hosts both Dataflow and HLT software. In building the vertical slice prototype, the major challenge was achieving the integration of both software frameworks. As described in Section XXXX, the PSC acts as an interface between the control aspect of the dataflow and the event selection software (Section XXX). The selection software included all elements described in ChapterXXX, 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 problems and unresolved issues, can be found in [14-2].

The event data, generated from fully simulated samples of di-jet events and including the LVL1 trigger simulation, was loaded into the ROS before starting a run. A suite of trigger algorithms designed to select electrons and photons ran within the L2PU, together with the appropriate detector software to decode the raw data. For these tests, only the LAr and Tile calorimeters and the SCT/pixel detectors were used. In addition, in order to analyze the system performace of the software, the software suite was instrumented using NetLogger.

The latency distribution for processing the electron/photon selection in the vertical slice prototype is shown in Figure 14-1. INTERPRET RESULTS HERE. The total latency shown in the fig-

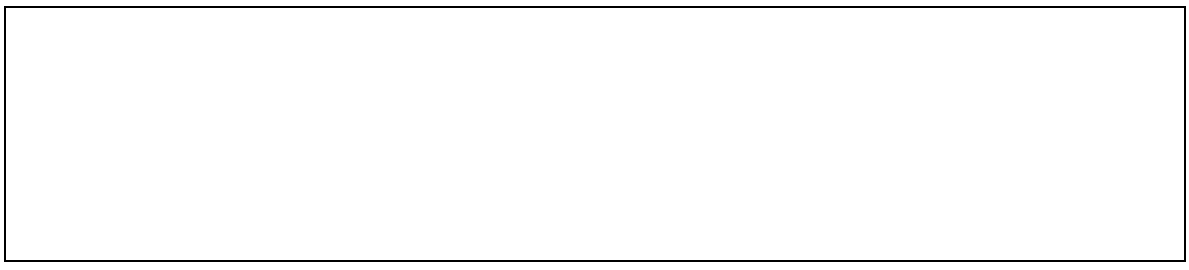

Figure 14-1 Latency distribution for processing events in the electron/photon trigger. The left plot corresponds to low luminosity, while the plot on the right corresponds to design luminosity. In each figure, the lower portion shows the fraction of events processed as a function of time.

ure includes contributions from network latencies, raw data conversion, and algorithm execution time.

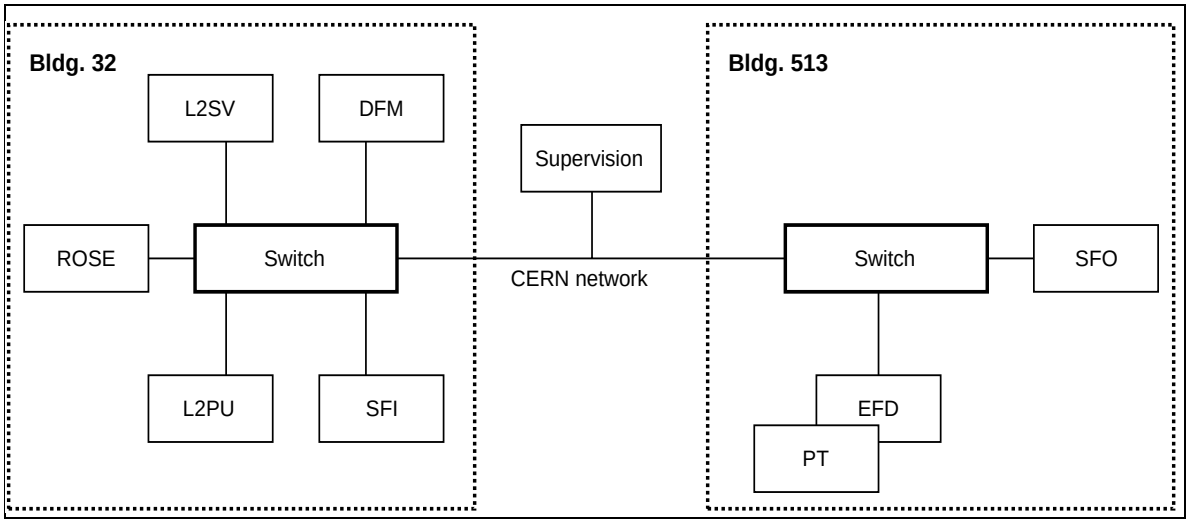

Figure 14-2 Setup for the LVL2 (left) and EF (right) vertical slice testbeds. The combined HLT testbed consisted of both LVL2 and EF testbeds connected via the CERN network infrastructure.

Here will provide a table with summary performance numbers of L2 slice. The table will present overheads per component (as in ATL-DAQ-2002-012) running in one thread:

- PSC+L1Result->SG
- above+dummy algorithm requesting data containers
- above-dummy+T2CALO
- above+steering
- above+IDSCAN
- above+L2result

For above, separate network latencies and PESA execution times. Compare with offline measurements.

Here describe system performance results for various components in different configurations, e.g., PSC overhead, framework overhead, algorithm usage of CPU resources, number of threads, etc. Only a few results shown in a table. The rest will be in a backup document.

Also give a sense of what sort of optimization can still be done in the software/strategy so that performance can be brought to an acceptable level.

Open issues: STL., etc.?

Address robustness requirements (runs more frequently in online that in offline)?

14.2.1.3 Event selection at the Event Filter

In the Event Filter, the event selection process is carried out by the processing task (ref. to chapter 9). In the first section, the baseline choice of using the offline ATHENA framework as the processing task is described, as well as the integration of the HLTSSW and ATHENA with the Event Filter. In the second section we will describe the current testbed implementation, the measurements and the validation strategy.

14.2.1.3.1 The Event Filter Processing Task

In the Event Filter, the PESA strategy consists in using the offline reconstruction algorithms with the minimum set of adjustments required to comply with the performance goals. Although the full detector information is available in the Event Filter, it will not always be necessary to unpack the full event to reach the trigger decision. Hence, it should be possible to seed the algorithms, in particular, with the LvL2 result and the corresponding lazy data unpacking mechanism should be supported.

The baseline implementation choice for the Event Filter Processing Task is to use the offline ATHENA framework. ATHENA is the ATLAS concrete implementation of the underlying architecture GAUDI. The GAUDI architecture separates Algorithm Objects from Data Objects. The Algorithm view of the framework is shown in Figure 14-3. Each algorithm can access a set

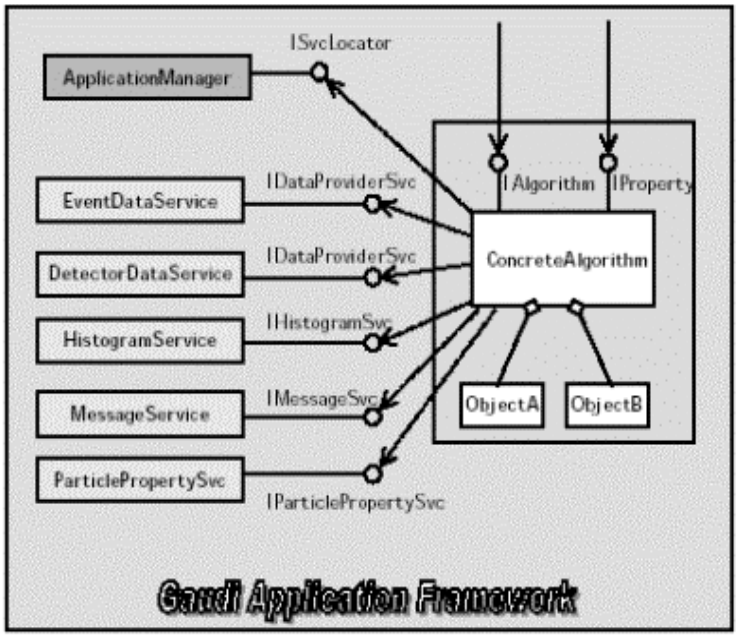

Figure 14-3 The Gaudi application framework architecture.

of services via an interface: for example the Event Data Service is accessed via the IDataProviderSvc interface. Data is exchanged between algorithms via the Transient Event Store.

The HLTSSW software is an algorithm suite steered by the Step Controller algorithm (refer to chapter XXX). While the Data Manager contributes to data preparation, all event data entities are defined in the Event Data Model. The HLTSSW is being developed in the ATHENA frame-

work. This will facilitate the development of algorithms, the adaptation of offline algorithms, the study of the boundaries between LvL2 and EF and the physics performances studies.

As we shall see to integrate the HLTSSW and ATHENA in the Event Filter will consist in making concrete implementations of some of the ATHENA services. In the Event Filter, each Processing Task consists of a standard ATHENA process, as in the offline case, that instantiates all the necessary services and runs an 'infinite' event loop.

In the following paragraphs, we will explain the event input mechanism, the production of the EFresult data fragment and how it passed to the EFdataflow, the access to the LvL2 result and the implementation of some of the services.

In the ATHENA / GAUDI concept, the infrastructure for reading from and writing to a particular persistency is provided by a conversion service (ref. Athena Developer Guide V 2.0.2). In this case the persistency is a full event in ByteStream format and the transient form are collections of Raw Data Objects while registered in the TES. The package that implements the necessary classes for a Gaudi conversion service is the ByteStreamCnvSvc. In addition, converters are responsible for converting a particular data type, in this case one for each collection type, to and from that particular persistency type.

Event Input

The service used to access the event is the ByteStreamCnvSvc. One of the elements of the service is the Input Source. Currently ATHENA requires each Event Input Source to implement the EventSelector and EventIterator interface. It is the EventSelectorByteStream that locates the requested ByteStreamInputSvc whose name is specified in the jobOptions. In case of running in the Event Filter farm, the specified input service is a class called ByteStreamEFHandlerInputSvc that specifies the event source as coming from the EFDataflow. To run in offline mode instead, reading data from a file containing events in ByteStream format, is achieved simply by requesting in the jobOptions file, a different source, the ByteStreamFileInputSvc.

When running in the Event Filter farm, the ByteStreamEFHandlerInputSvc interfaces with the EFDataflow, at initialization time, by connecting to an instance of the EFD PTclient singleton class (refer to Chapter 9). The method ByteStreamEFHandlerInputSvc::nextEvent requests a pointer to the next Byte Stream event in the EFD Shared Heap. It extracts the event size from its header and uses this information to construct an instance of the RawMemoryStorage class defined by the Event Format Library (EFL) online package. This object is in turn used to create and return an object of type RawEvent,a typedef for an Event Filter Library FullEventFragment object that is constructed from a RawMemoryStorage instance. From this point on, the treatment of data is exactly the same if running in the EF or offline.The IdentifiableContainer,used to contain the Raw Data Objects, and the corresponding converters provide a mechanism for creating the RDOs on demand.

After the HLTSSW processing is completed, its result is wrapped in an object of type EFResult which is derived from the Athena DataObject class. The ByteStreamCnvSvc is again responsible for the conversion to persistency. The list of CLIDs of objects that should be converted to persistency is declared in jobOptions. When running in the offline mode, the output service is used to simulated the events in ByteStream format; in that case the list of CLIDs of the various types of RDO collections is declared in the jobOptions. On the other hand, when running in the Event Filter farm, one needs only to append the EFresult to the original event residing in the shared memory. Hence only the CLID of the EFresult object is declared in the jobOptions.

Output of EF selection process

After the event filtering process, an algorithm creates the EFresult object that contains a bit pattern corresponding to the result of the selection and some more detailed information about the selection, like which trigger element and menu items have been validated. The Athena ByteStreamCnvSvc calls the EFResultByteStream Converter that creates a ROD fragment which payload contains the bit pattern. The ByteStreamCnvSvc automatically takes care of collecting all existing ROD fragments and, using the Event Format Library, constructs the Full Event Fragment. The correspondence between a ROD and its corresponding ROB / ROS / SubDetector is provide at initialization. Again there is similarity between the offline case, where the aim is to produce ByteStream format event files and the case of the Event Filter test bed. Different jobOptions will select the list of CLID for the relevant RDO collections in the first case and the EFresult CLID in the latter. The same mechanism will be used to pass back to the EF additional information contained in reconstructed objects during the selection process. For example, the TES object representing an identified electron, given the corresponding converter that serializes the information and insert it in a ROD fragment, can also be included in the Event Filter 'Subdetector Fragment' and will be appended to the original event.

Access to the LvL2 result

The access to the LvL2 result is a trivial matter. The converter associated to the LvL2result object will be automatically called by ATHENA and the object created in the TES after unpacking the LvL2 'Subdetector' fragment when the first access request will be issued by one the HLTSSW algorithm.

Other services

Various others ATHENA services will be interfaced to the EFsupervison, like messaging, error reporting, exception handling or to Condition Database services, etc. Again, to switch running in offline mode or running in the EF farm will consist in selecting the appropriate concrete service implementation via jobOption files.

Conclusion

We have seen that the use of the Athena software as the EF Processing Task makes it completely transparent to switch from the offline development environment to the EF farm which was one of the important requirement. Next we have to validate that this approach meets the performance requirements of the Event Filter.

14.2.1.3.2 Event Filter Prototype

To validate the choice of the ATHENA Framework for the Processing Task, an implementation of the HLTSSW running in ATHENA in the Event Filter has been prototyped in the Magni Cluster (ref. to XXX).

The strategy of validation consists in running some of the most relevant triggers in terms of rates: the electron trigger and the muon trigger. Efforts have been directed to the electron trigger first. The HLTSSW suite for electron identification is run starting with ByteStream data files corresponding to single electrons and dijet events, the main background source. The files have been simulated offline and the result of the LvL2 selection in form of a LvL2 'Subdetector' fragment is included.

The current HLTSSW selection suite includes tracking and calorimeter reconstruction adapted from the offline electron identification software. The interface to the EF Dataflow PTclient described in the above section has been implemented. For now, the EFresult object contains a set of bits matched to the decision of the selection process: 'Accept', 'Reject', 'ForcedAccept', 'Error'. No additional reconstructed objects are serialized in the current test. The implementation of the services are the following:

- The message service simply writes to a log file specific for each PTtask (name declared in jobOption file and known by the EFSupervision)
- The histogram service is interfaced to the Web based histogram service (refer to section XXX??)

Validation procedure:

1. The basic log files are monitored by the Supervision has a simple monitoring and debugging tool.
2. For timing measurement, the TrigTimeAlgs package is used. This allows to compare timing measurement done offline with the ones made in the test bed. It should be pointed out, that there is a complete decoupling between latency due to the EFdataflow and the latency of the ATHENA and HLTSSW selection process. One the pointer to the SharedHeap is passed to Athena there is no difference between that case and the offline case where the ByteStream has been read from a file and copied in the local memory. Comparing the two modes, running with equivalent processors, allows to measure the additional overhead that could arise when running, in the offline case, in an uncontrolled farm environment. The latency in the EF that may result when the request is issued for a new event can be measured with a dummy PT. This complete decoupling will not be true any more when the database access at run time will be enabled. This is not included in the current test.
3. The histogramming package is exercised. Histograms are produced by the various processing tasks and are collected by EF Supervision.
4. The Efresult will be appended to the original Event. These events are analysed and their content compared to the result of processing the same events offline.

14.2.1.4 The HLT vertical slice

The LVL2 trigger and the Event Filter were integrated in a single testbed as shown in Figure 14-2. The LVL2 slice (described in Section 14.2.1.2) and the EF slice (described in Section 14.2.1.3) were connected to form an 'HLT vertical slice' using the CERN network infrastructure. The DFM and the Event Building were located geographically close to the event fragment sources. In order to pass the LVL2 result to the EF, one of the ROSes was configured as a pROS (described in Chapter 8 ??). The entire system was configured and controlled by the Online software using the CERN network.

During the tests, the ROSes were pre-loaded with LVL1-preselected events. NEED TO ADD RESULTS HERE.

14.2.2 The 10% prototype

14.2.2.1 Description of the 10% testbed

In Chapter 8, "Data-flow", the performance of the individual components in the ATLAS Data-Flow system has been presented. These components, when operated together, carry out the functions of RoI collection and Event Building. The performance for these two functions using independent subsystems has also been presented in that chapter. The final ATLAS Data Collection system requires simultaneous operation of RoI collection and event building. This section describes results obtained from a testbed with a size of approximately 10% of the final system, with full data collection functionality. Although the testbed necessarily is a scaled down version of the final system, individual components have been operated at rates similar to those expected in the final system. The primary aims of the 10% testbed are to demonstrate full functionality of the data collection in both the LVL2 and the EB subsystems simultaneously and to check for possible interference between the subsystems. The latter is especially important with respect to the choice to be made between a switch or bus based ROS. The testbed results have also been used to calibrate and validate computer models of components and systems. The approach taken is to assure that the measured performance of 10% systems can be successfully reproduced prior to drawing conclusions from modelling full size systems. Finally, the testbed results have been used to study possible ways to achieve a staged approach to a full size system by the progressive installation of components.

The first testbed studies have been done with a single Data Collection application and associated test environment. Next, small setups with one instance of each Data Collection component, aimed at demonstrating functionality, were used. Studies with more complex setups of either EB or LVL2 subsystems followed. The largest testbed inherits from the previous ones and represents about 10% of the final system. It's organization, as presented in Figure 14-4, reflects the architecture of the final system.

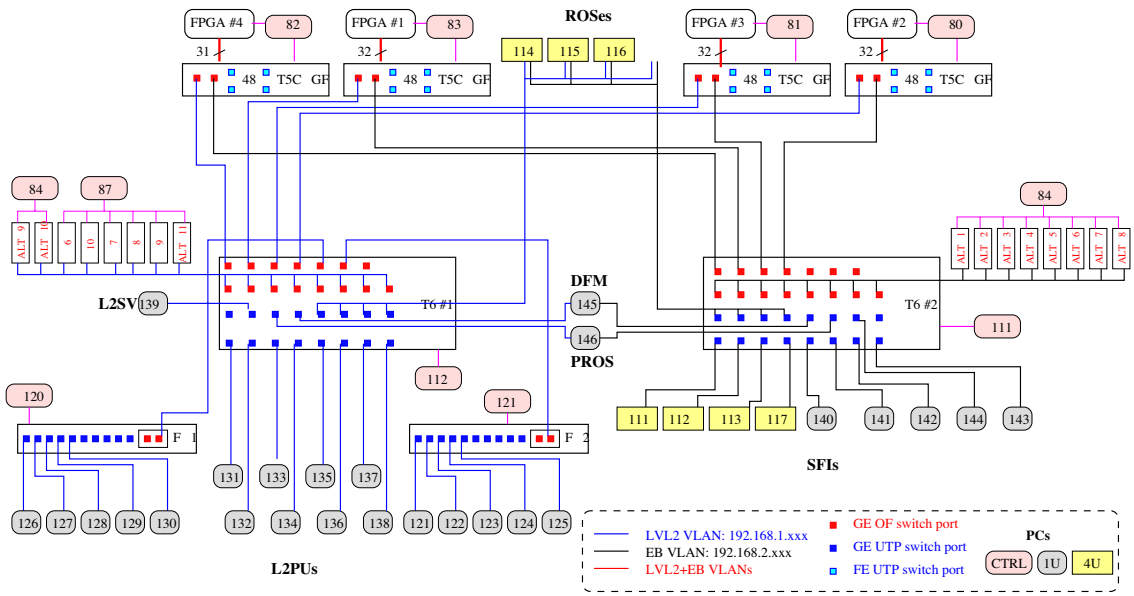

Figure 14-4 Organization of the 10% testbed

The central part of the 10% testbed consists of two central switches. The central switch T6 #1 has been assigned to the LVL2 subsystem whereas the central switch T6 #2 plays the central role in the EB subsystem. Both central switches are 32 port all-Gigabit Ethernet switches (fibre / UTP ports). For the final system the possibility of installing more central switches is foreseen. For example the number of central switches can be four when ROB concentrating switches with four uplinks are used. The consequences of using more than one central switch per subsystem have been studied in the testbed and the performance has been measured. This could be achieved in a straightforward way as the functionality of the processing nodes is defined by the type of application executed: changing it from SFI to L2PU, or vice-versa is possible by reloading appropriate software into a PC.

In the LVL2 subsystem (T6 #1) some of the processing nodes are attached directly to the central switch (nodes 131 - 138) and some via the LVL2 grouping switches (F-1 and F-2 in Figure 14-4). For the final system it is planned to connect LVL2 processing nodes via the LVL2 grouping switches, but for the testbed not enough LVL2 grouping switches (like F-1 or F-2) were available. Therefore PCs running the L2PU application were connected directly to free ports in the central switch in order to produce more traffic in the LVL2 subsystem.

In the EB subsystem PCs 111-113, 117 and 140-143 act as SFIs directly connected to the EB central switch. The current paper model predictions assume 80 SFIs for the final system, the number of SFIs installed in the testbed amounts to 10% of the final system.

Three options have been explored with respect to how the detector data stored in the ROBs are accessed. The FPGA ROB emulators (FPGA #1 - FPGA #4) allow to study the option of using ROBs with individual network access via Fast Ethernet connections. This option is described in more detail in the [ATLAS TDAQ Switch-based architecture note reference]. The ROBs are connected to concentrating switches (4 * T5C). Each concentrating switch has two uplinks, connected to the central switches. The second option studied consists of readout of the data via bus-based ROS PCs. In the testbed PCs act as bus-based ROS emulators (114-116).(*perhaps someone could provide more details, or at least a reference to the bus-based ROS..?*). In the third option two or more ROBIns share a single network connection, as in the prototype ROBIn. For this option ROB / ROS emulators are used based on programmed Ethernet Gigabit NICs (ALT1-8 and ALT??). Depending on the software loaded, the ROB / ROS emulator can reply with either individual ROBIn data, or produce reply messages with data from a number of ROBIns. The hardware emulators (FPGA and Alteon NICs) allow to output the data from a number of ROBIns. Therefore with only a limited number of emulators (128 FPGA channels and tens of Alteons) the data from more than 1600 ROBIns can be provided.

The architecture with the two central switches and the ROB concentrating switches creates Ethernet loops. The Spanning Tree algorithm is used to switch off the redundant links and VLANs are used to avoid changing the configuration of the testbed. As at the time of writing the Data Collection applications did not support VLANs on a single network interface, the DFM and the LVL2 Supervisor (the two applications which communicate across VLANs) were connected with two network interface cards to the EB and LVL2 central switches respectively.

14.2.2.2 Description of measurements

The organization of the measurements with the 10% testbed aimed at having the components running at the nominal rates, as required for the final system. The basic quantities measured are the rate and the latency (time needed for producing a LVL2 decision or completion of event building).

Possible packet loss in the full size system is a source of concern. Therefore the effectiveness of the traffic shaping schemes used by the LVL2 and EB subsystems (the SFI uses credits to limit the number of requests into the network, while the L2PU limits the number of worker threads working in parallel) to avoid packet loss have been studied.

The amount of network resources (ports and switches) necessary for operation of the full size system is presented in [ATLAS TDAQ Switch-based architecture note reference]. In this document 60% Ethernet link utilization is assumed. In the testbed, the limited number of network resources (switches) limits the rates and throughputs. In the tests processing applications running at their nominal rates have been added until the network became a bottleneck. Then the traffic generated on the links was reduced to 60% of their nominal rate and it was verified that this safety margin guarantees the absence of packet loss.

Simultaneous operation of the EB and LVL2 subsystems as well as of multiple instances of the EB subsystem and of multiple instances of the LVL2 subsystem have been studied.

For each configuration separate measurements were done with either the FPGA ROB emulators, the Alteon ROB emulators, or the PC ROS emulators delivering data to the tested subsystem(s). Also measurements were done with all available ROB emulators delivering data. For the configuration without the LVL2 subsystem all processing nodes were loaded with the SFI application and the DFM was set into autorun mode (this avoids the need to communicate with the LVL2 Supervisor to get lists of accepted events). For the configuration without the EB subsystem all processing nodes were loaded with the L2PU application and the Supervisor was set in autorun mode (the Supervisor generates LVL1 accepted events without communication with the LVL1 subsystem). Also for simultaneous operation of the LVL2 and EB subsystems the Supervisor was set in the autorun mode and communicated to the DFM lists of accepted events for which the Event Building is necessary. The interference of the two subsystems when accessing the ROB buffers was investigated measuring the effect of changing the number of accepted events sent in the list from the Supervisor to the DFM on the maximum event rate.

14.2.2.3 Results

Distributions for the event building latency and maximum event building rate measured as a function of the number of credits in the SFIs are presented in Figures .... These measurements were done with the testbed configured for event building only (i.e. without LVL2 traffic) using both central switches and with each of the three different types of emulators separately and with all types simultaneously active. *Discussion of results to be added. Forward reference to discussion in Section 14.5.1 on comparison with model predictions.*

PLACE-HOLDER FOR RESULTS FROM EB-ONLY TESTBED CONFIGURATION

Distributions for the RoI building latency and maximum RoI handling rate measured as a function of the number of threads in the L2PUs are presented in Figures ..... These measurements were done with the testbed configured for LVL2 triggering only (i.e. without EB traffic) using both central switches. *Discussion of results to be added. Forward reference to discussion in Section 14.5.1 on comparison with model predictions.*

PLACE-HOLDER FOR RESULTS FROM LVL2-ONLY TESTBED CONFIGURATION

Distributions for the RoI building latency and maximum RoI handling rate, event building latency and maximum event building rate are presented in Figures ..... These measurements were done with the testbed configured for both LVL2 triggering only and event building. *Discussion*

of results to be added. Forward reference to discussion in Section 14.5.1 on comparison with model predictions.

PLACE-HOLDER FOR RESULTS FROM LVL2+EB TESTBED CONFIGURATION

14.3 Functional tests and test beam

During prototyping phases, often the performance of a system is put in foreground with respect to its stability and maintainability. Functional user requirements have in this phase of development a lower priority than the achievement of the performance requirements. This is to some extend true also for the TDAQ system, which has focused its efforts in the area of trigger rates, speed of data acquisition, etc. Nevertheless we have decided to also stress the global functionality of the TDAQ system, by carrying out a series of functional tests and exposing the system to non expert users at the ATLAS test beam sites.

Three different aspects of the functionality have been covered:

- a)Dynamic system configuration
- b)Stability in cycling through TDAQ finite states
- c)Operational monitoring and system recovery in case of errors

All these aspects have first been tested in dedicated laboratory setups and then verified in a 'real' environment, during test beam data taking.

- a)A TDAQ system has to be easily reconfigurable in order to accommodate the substitution of hardware, the change of trigger conditions, etc. This means that on one side all the tools to keep the configuration parameters in a database have to be developed and on the other side that the Run Control, DataFlow and Trigger software has to be designed to be dynamically reconfigurable.

To verify the flexibility of our system the following tests have been carried out:

- -substitution of a data taking machine
- - exclusion and reinsertion of a Run Control branch
- -change of communication protocol between the ROS and the L2/EF (= change of Data Collection protocol)
- -change of run parameters.

More detailed test description and measurement results to be included here.

- b)When performing a series of measurements with different configuration options, the TDAQ system must be capable of cycling through its finite states stably. This functional requirement has been checked via automated scripts cycling repeatedly through the finite state machine.

More detailed test description and measurement results to be included here.

- c)In a distributed system such as the TDAQ it is important to constantly monitor the operation of the system. Furthermore, the fault tolerance is a fundamental aspect of its functionality. In this area several improvements are still to be achieved, but we decided to carry out a series of tests in order to assess the present performance of the system in case

of errors. In particular we tried to test the fault tolerance of the system in the presence of a fatal error which prevents on or more data taking computers to continue their working.

- Verification that all applications provide regular information on their status
- -Failure of a SFO
- -Failure of a EF subfarm (distributor or collector)
- -Failure of a EF processing task
- -Failure of a SFI
- -Failure of a L2PU
- -Failure of a DFM
- -Failure of a L2SV
- -Failure of the RoI builder
- -Failure of a ROS
- -Failure of a ROBIN
- -Failure of a ROL
- -failure of online sw servers (is, mrs, ipc, ….)

More detailed test description and measurement results to be included here.

The results of the various tests will determine the summary and conclusions of this section. It is premature to indicate them now.

14.4 Paper model

Estimates of average message frequencies, data volumes and the amount of processing power required have been made with the help of the “paper model”. Due to the RoI driven nature of the LVL2 trigger and the different processing sequences and associated data request patterns, a “back-of-the envelope” calculation on the basis of the LVL1 and LVL2 accept rates and average event fragment sizes is not feasible. The quantities of interest have in the past been calculated with the help of a spreadsheet. Due to problems with easy modifiablity of e.g. trigger menus and processing sequences, with the extraction of the results and with checking the correctness of the computing procedures implemented, the spreadsheet has been replaced by a program written in C++. The problems mentioned are solved, as all parameters are specified in separate input files, as results output by the program, an array of tables in ASCII format, are chosen by parameters in an input file, and as the program source gives a clear overview of the computing procedures followed.

The most important results of the paper model have been presented in Chapter 2. In Appendix A a description of the model and detailed results can be found.

With the help of the paper model a direct comparison may be made between sequential and non-sequential processing in terms of the quantity of data to be transported and of the processing resources needed to execute the respective trigger algorithms (non-sequential processing refers to the scenario in which all data that possibly could be needed is requested and processed). In particular the time consuming analysis of the inner tracker data (see Appendix A) is less fre-

quently required for sequential processing. This results in sequential processing being an order of magnitude faster than non-sequential for the current processing time estimates, see Table 14-1. The table also shows the increases in RoI request rates and amount of data transport if non-sequential processing is assumed. Another benefit of sequential processing is a shorter average decision time than that resulting from non-sequential processing.

Table 14-1 The relative increase in request rates, LVl2 data volume and size of the LVL2 farm if non-sequential instead of sequential processing is used.

	Low luminosity	Design Luminosity
LVL2 total request rate (all ROBIns)	1.7	1.9
LVL2 total data volume	1.4	2.0
Number of LVL2 PCs	6.2	13.5

14.5 Computer model

Because a full scale testbed is not feasible only simulation with a “computer model” can provide information on the dynamic behaviour of the full system. Computer models have been developed to get answers to very basic and fundamental questions like throughput and latency distribution of the LVL2 and EB subsystems when operating together, 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. Also the interaction between multiple instances of the LVL2 subsystems and between multiple instances of the EB subsystems has been studied.

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. The type of simulation used for the computer models is known as “discrete event simulation”. Basically 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” occurring. Only at the time of occurrence of an event the modelled system is 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 for a later time, which are entered at the correct position in the 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 trigger/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 Collection applications) or data items. The models of the hardware and software items 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 [*reference to the DC note*] and Ethernet switches [*reference to DC note on parameterization of switches*] have been developed. The calibration of the models of the Data Collection applications was determined by analysing time stamps. These 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 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 any possible limitations in the switches which may affect the performance required for the full ATLAS trigger / 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 measurements usually led to a modification in the model(s) and associated parameters and another calibration phase.

Two simulation programs have been used, the at2sim program and the Simdaq program. The at2sim program makes use of the general purpose simulation environment of the Ptolemy system. 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 a part of the program. The component models used in the at2sim program are based on testbed components and calibrated as described above. The component models in the Simdaq program are less specific.

14.5.1 Results of testbed models

The testbed configurations for which results have been presented in Section 14.2.2 have been modelled. In Figures ... model predictions and measurement results are compared. ..... agreement is observed. *Discussion to be added.*

PLACE-HOLDER FOR COMPUTER MODEL RESULTS OF TESTBEDS

14.5.2 Results of extrapolation of testbed model and identification of problem areas

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.

The computer models of the full system can be used to identify possible problem areas.

The full system model implemented in at2sim is the ATLAS network-based architecture with 1564 ROBIns with individual network connections (ROBIns of which the data is not used in the LVL2 trigger have not been taken into account). The LVL2 subsystem is 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 L2Super, DFM, pROS are connected via a dedicated small Gigabit Ethernet switch to the 4 central switches. The ROBIns are connected via concentrating switches with 4 links to the central switches. The ROBIns were grouped as in the full size ATLAS: groups of ROBIns from a subdetector connected to a number of concentrating switches. The number of

concentrating switches per subdetector depends on the average fragment size produced by ROBs from a given subdetector (calculations are presented in the network-based architecture note). In total there were 46 concentrating switches. Results were obtained for the following configurations:

1. “individual ROBIn FE”: 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 GE”: each ROBIn has a single Gigabit Ethernet connection to a concentrating switch,
3. “2 ROBIns aggregate”, “4 ROBIns aggregate”, “6 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 final 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 step: for each event data from 10 randomly chosen ECAL ROBs was requested and a decision was produced and sent to the Supervisor. The acceptance factor chosen resulted in 3 kHz event building rate). The L2PUs were not calibrated - they were used only to provide the LVL2 traffic and get the EB latency in the more realistic environment. The SFIs were requesting data from randomly addressed ROBs.

In the full system model implemented in Simdaq the same LVL1 trigger menus as used for the paper model are used to generate an appropriate number and type of RoIs for each event. In both models the eta and phi coordinates of the RoIs are chosen at random from the possible eta, phi coordinates (as defined by the LVL1 trigger). ROBIns are grouped together in groups of 12 in ROS units, each with two Gigabit Ethernet connections, one to a central LVL2 switch and one to a central EB switch. ROBIns of which the data is not used in the LVL2 trigger have not been taken into account. The mapping of the detector on the ROBIns is the same as for the paper model. Also the same processing times and LVL2 processing sequences are used. Average message rates and volumes and total CPU power utilized obtained from the paper and the computer model of the full system therefore should be equal within the statistical errors. The component models (processors, switches) however have not been calibrated as was done for at2sim. The main use of Simdaq therefore is for studying general trends in the behaviour of the full system with respect to building up of queues and of effects of possible choices for processor allocation strategies.

14.5.2.1 Load balancing

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 consists of the LVL2 supervisor or DFM maintaining 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 this should be straightforward to implement. A new event can then be assigned to the L2PU or SFI with the smallest number of events to process. Simulations of the LVL2 system have shown this to be a very effective strategy, with which high average loads of the L2PUs are possible (*reference to TPR or new results, if available*).

PLACE-HOLDER FOR COMPUTER MODEL RESULTS ON LOAD-BALANCING

e.g. results for round-robin compared to results for least-number-of-events-assigned

14.5.2.2 Congestion in switches and buffer sizes

The effect of the credit based event building traffic shaping on the event building latency and queue build-up has been investigated with the configuration modelled in at2sim. The latency plot in Figure 14-5 shows that increasing the number of credits per SFI above 10 does not improve the latency for event building except for the “individual ROBIn FE” configuration. The latency for this configuration will for more than 10 credits still depend on the Fast Ethernet link transfer time, as it is unlikely that all 10 requested ROBIns will reply at the same time and the replies will form a queue in the concentrating switch for the uplink to the central switch. The shorter latency for setups with ROBIn aggregates with respect to individually accessed ROBIns is explained by 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 scales with the number of ROBIn aggregates. Relative to the time required for receiving the replies the SFI can therefore exhaust the credits assigned faster than for individually connected ROBIns (the CPU time is shared between the process receiving replies and the process generating requests). This also causes longer queues in the central switch, as can be seen in Figure 14-5.

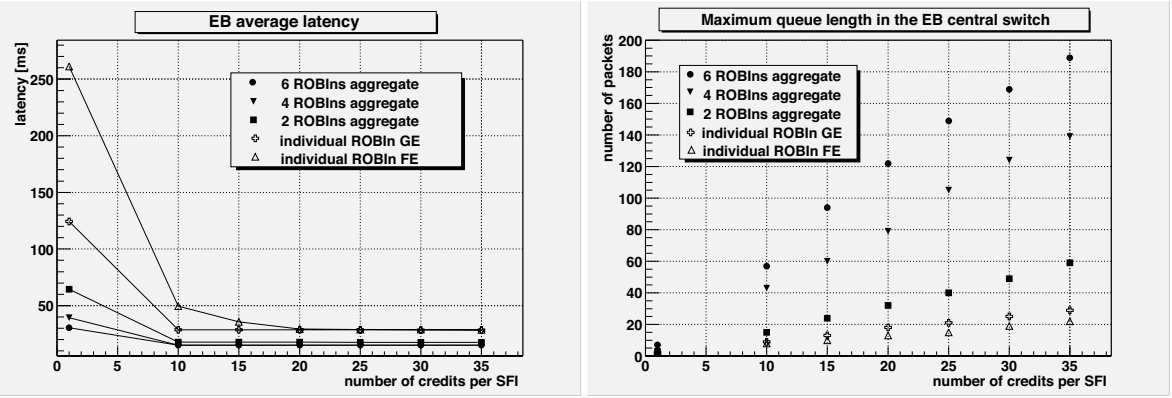

Figure 14-5 Average event building latency and maximum queue length in the EB central switches for different ROBIn configurations obtained with the at2sim full system model

More quantitative information on link and processor utilization to be added.

From Section 14.4 and from Chapter 2 it can be concluded that for the base-line architecture the total available output bandwidth from the ROS (two Gigabit Ethernet connections per ROS PC) is considerably higher than the required bandwidth (about 28 Gbyte/s vs. 6 Gbyte/s for design luminosity and a LVL1 trigger rate of 75 kHz).

Conclusion from previous result to be added.

14.5.2.3 Spare processor capacity and spare network bandwidth

PLACE-HOLDER FOR COMPUTER MODEL RESULTS ON BUILD-UP OF QUEUES AND LATENCY AS FUNCTION OF PROCESSOR UTILIZATION

14.6 Technology tracking

14.6.1 Status and Prospects

The ATLAS TDAQ system is to a large extent comprised of off the shelf commodity equipment; personal computers (PCs) and Ethernet links and switches; the exceptions being the RoI builder and ROBIns where specialized equipment has had to be developed. The technical evolution of commodity computing, communications equipment, as well as pricing, is therefore a significant element in the performance, costing and life cycle of the TDAQ system.

Impressive price and performance improvements have occurred over the last two decades. In this section we consider the prospects over the next decade, a period which covers the run up to the commissioning of ATLAS and the first few years of running.

14.6.1.1 The personal computer market

Moore's Law, the doubling of the number of transistors on a chip every couple of years, has been maintained over three decades, and still holds true today. Intel expects that it will continue at least through 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 Intel's 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. The only blip on the horizon being economic: the turndown in the world economy and an unwillingness to invest the large sums of money required to deliver new generations of microprocessors.

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 4 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 farms where the processing time will be long compared to the I/O time. Components in the system with high I/O requirements will also benefit from improvements in processor performance but will be more bounded by link speed. Figure 14-6 shows the performance of the DFM as a function of processor clock speed.

14.6.1.2 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

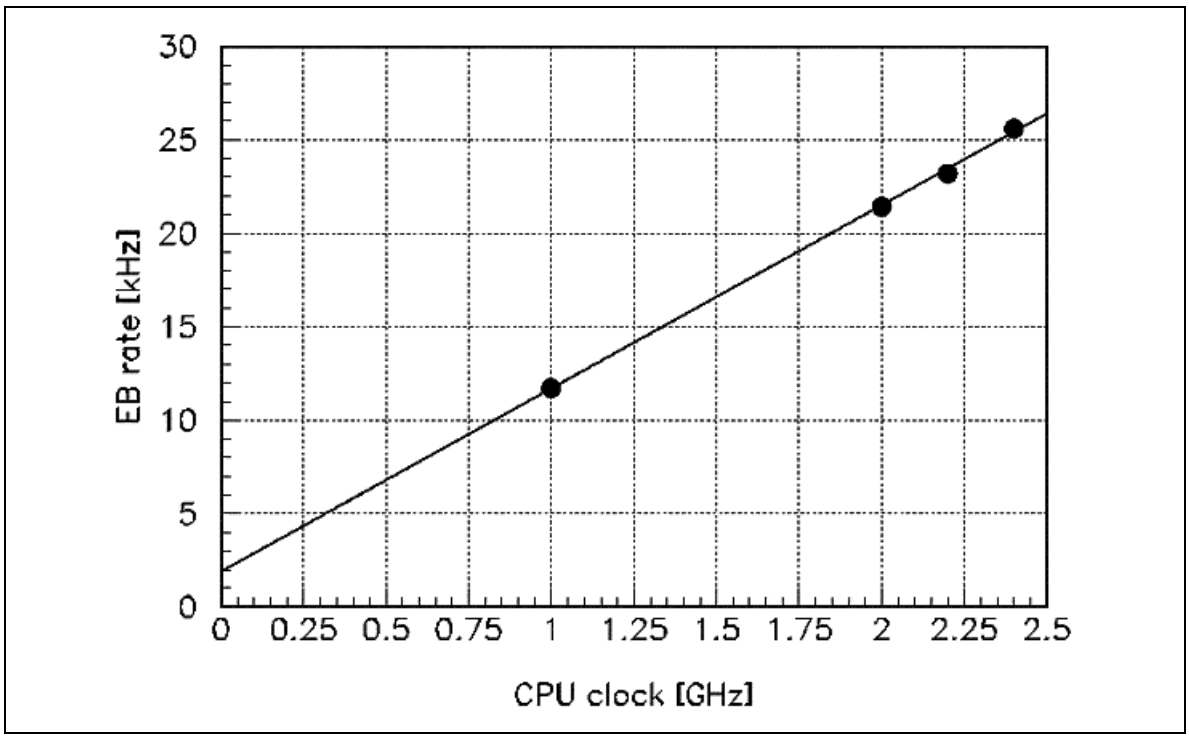

Figure 14-6 The performance of the DFM as a function of processor clock speed.

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. Long-term, the optimization of the operating system will continue, fueled 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.

14.6.1.3 PC Buses

I think we have to put in something about the future of PCI bus.

14.6.1.4 Networking

The ATLAS baseline system uses Ethernet network technology for RoI collection and event building, as well as data distribution to the event farms. It is also used in other networks associated with control and monitoring. Ethernet is, throughout the world, the dominant local area network (LAN) technology. It has evolved from the original IEEE standard, based on a 10 Mbps shared medium, to today's point to point links running at speeds of up to 10 Gbps [14-4].

The price of Ethernet technology has followed a strong downward trend driven by high levels of competition in a mass market. Figure 14-7 shows the price of 100 Mbps (FE) and 1 Gbps 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 hap-

pened earlier with FE. This trend is coupled to the increasing provision of a GE connection in all PCs.

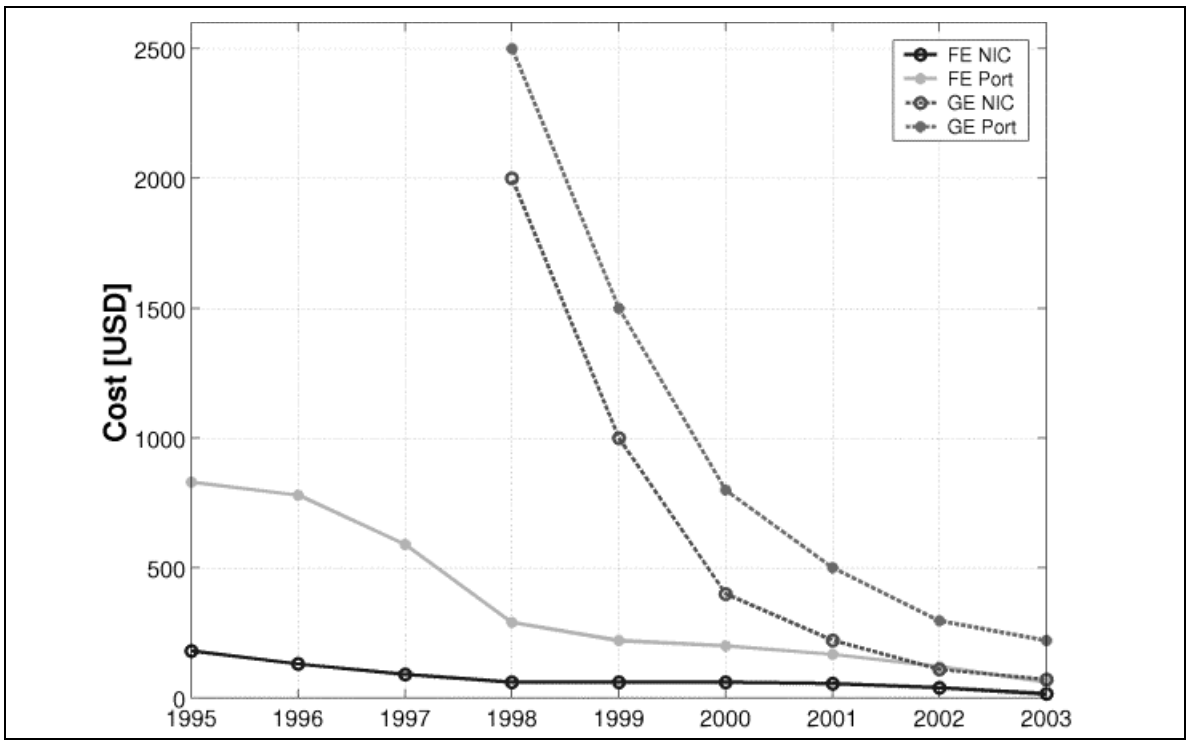

Figure 14-7 The evolution of the cost for Fast and Gigabit Ethernet switch ports and network interface cards.

The proposed baseline system can be built using Ethernet switches available today. Even the most demanding components in the system, the large central GE switches in the data collection system, 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 Gbps transmission between CERN and NBI, Copenhagen over the GEANT pan European backbone network [14-5]. Figures 14-8 and 14-9 show the setup used and some of the results obtained.

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 GE 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.

14.6.2 Survey of non-ATLAS solutions

(a reality-check on ATLAS approach?)

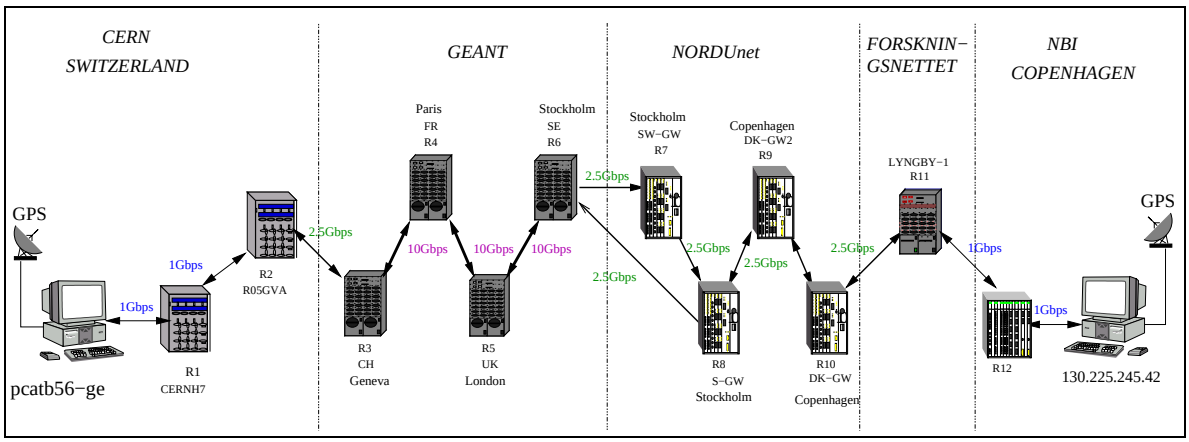

Figure 14-8 The network infrastructure between CERN and NBI (Copenhagen) over which Gbps tests have been carried out.

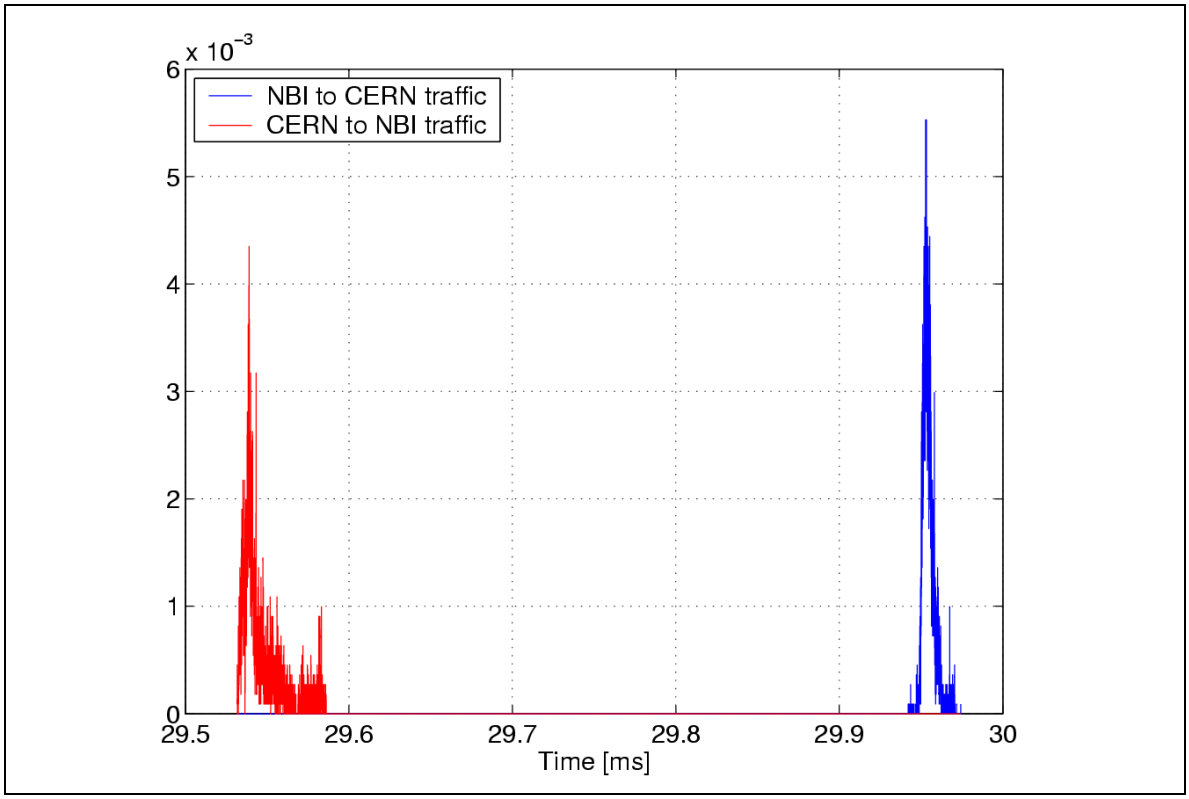

Figure 14-9 Packet latency measured between CERN and NBI (Copenhagen) at a 1 Gbps transmission rate.

14.7 Implication of staging scenarios

Re-interpretation of performance numbers for staging scenarios

14.8 Areas of concern

14.9 Conclusions

14.10 References

14-1

14-2 LVL2 vertical slice results

14-3 HLT vertical slice results

14-4 Dobinson et al RT2001 Lyon

14-5 Santiago di Compestella, RT2003

14-6

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 for with the adoption of a development framework for DAQ components. The development framework consists of distinct development phases. At the end of each phase a set of deliverables is provided. This framework is complemented by guidelines, checklists and standards, internal reviews, templates, development and testing tools and coding standards. Those are being adopted as common working practice and help for error removal and error prevention in 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 mend to be a help for 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 automized validation testing

A powerful release management system and a convenient working environment provide the necessary technical working basis.

15.2 The Development Process

The software development process (SDP) in Atlas TDAQ [15-2] 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 phases will evolve within time.

15.2.1 Inspection and Review

Written material including documents and code are 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 Tom Gilb's Software Inspection method [15-3]. An important feature of the

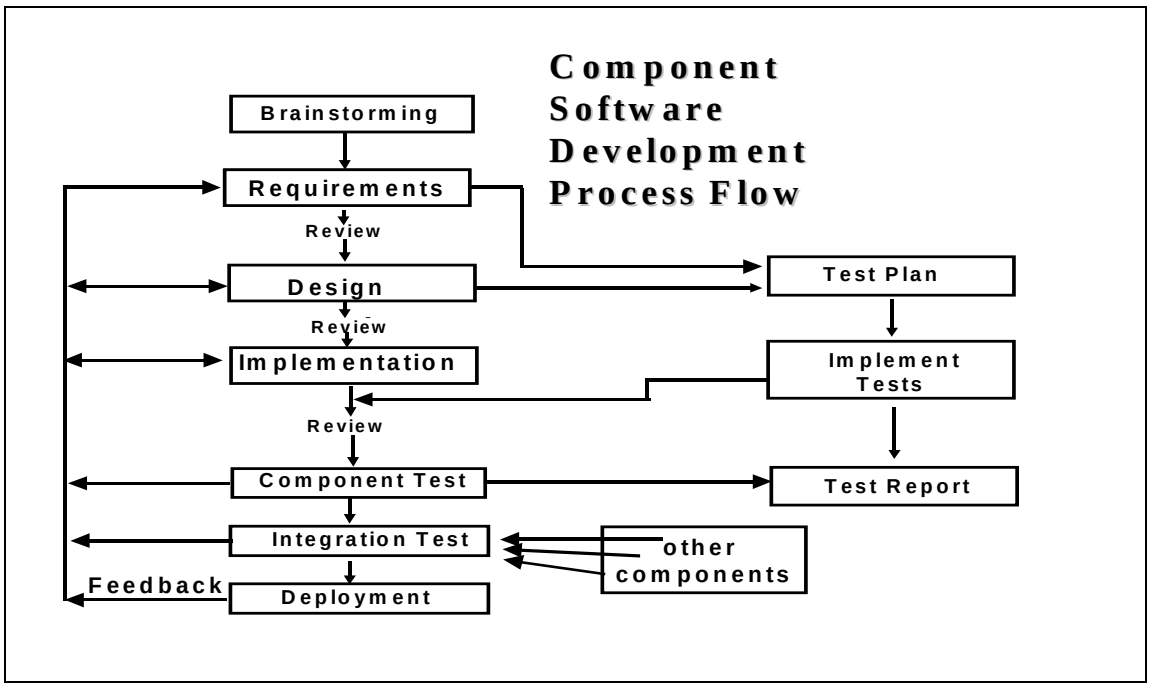

Figure 15-1 Phases and flow of the Software Development Process

inspection procedure is its flexibility, allowing it to evolve as needs change during the lifetime of the project [15-4].

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].

15.2.2 Experience

The Software Development Process 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 meets the requirements within a predictable schedule.

However, one of the key differences in adopting the SDP in an HEP as opposed to 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 [refs].

15.2.3 The Development Phases

15.2.3.1 Requirements

The Requirements phase [15-5] 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 & review is described [15-5]. 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 22 rules for the requirements document itself, for which a template has been provided in each of the supported documentation formats.

15.2.3.2 Architecture and Design

The Architectural Analysis and Design Phase of the SDP [15-6] follows the Requirements phase and takes as its starting points the User Requirements & 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 'Howto' note has been produced describing the overall process.

For this phase, we are largely following the approach of the Rational Unified Process (RUP), which contains descriptions of concepts, artifacts, guidelines, examples and templates. In particular, we have highlighted the RUP descriptions of architectural analysis and design concepts and 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. We have also prepared recipes for producing appropriate diagrams and incorporating them into documents.

15.2.3.3 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 [15-7] are being applied to newly written code and being introduced for existing code still in evolution. In the case of Java we await the outcome of the Atlas investigation of coding conventions. DCS will follow the coding standards provided by the JCOP Framework for PVSS [15-8].

Guidelines [15-9] 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. No standards have been identified for the hardware so far.

15.2.3.4 Component Testing and Integration Testing

Testing occurs during the entire life-time of a component, group of components or entire system. Referring to figure [SDP], the initial test plan is written during the requirements and design phases of the component, so as not to be biassed 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 [15-12] 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 life-time 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.

15.2.3.5 Maintenance

As with testing, maintenance occurs during the entire life-time 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. Preventative 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 fixed 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.

15.2.4 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 [15-13] 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-system like the use of compiler versions and platforms are coordinated.

Development tools like design tools, memory leak checking tools, automatic document production tools and code checking tools are vital elements of the development environment.

No standards have been identified for the hardware so far.

15.3 Quality Assurance During Deployment

15.3.1 Quality Assurance of operations during data taking times

The quality of the DAQ 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 shall 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, 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 inititated. This may include running special diagnostic code, replacement of faulty equipment, rebooting of processors, restarting of applications, re-establishing network connections, re-configuration 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 is 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 point of failures 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 technology ones. 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 to incorporate smoothly new parts into a running system, in particular also when staging of the system is required.

15.4 References

15-1 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/

15-2 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/- SDP

15-3 Software Inspection - Tom Gilb, Dorothy Graham - Addison-Wesley 63181

15-4 Impact of Software Review and Inspection - D. Burckhart et.al. CHEP2000, Padova

15-5 http://atlas-connect-forum.web.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;

15-6 http://atlas-connect-forum.web.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.

15-7 ATLAS C++ Coding Standard Specification The coding conventions - http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/Development/qa/General/index.html

15-8 Joint Controls Project - http://itco.web.cern.ch/itco/Projects-Services/JCOP/

15-9 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/- SDP - Implementation Multi-user multi-platform scripting guidelines.

15-10 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/ - Tools

15-11 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/ - Templates

15-12 http://atlas-connect-forum.web.cern.ch/Atlas-connect-forum/ - Testing

15-13 Configuration Management Tool - http://www.cmtsite.org/

16 Costing

16.1 Initial system

16.2 Final system

16.3 Deferral plan

16.4 References

16-1

16-2

17 Organization and resources

Should the geographical, racks, power supplies, and cooling issues be addresses in this chapter or in the system component ones?

17.1 ...

17.2 References

17-1

17-2

18 Work-plan

Post TDR. Need to change the title of the chapter I think

18.1 Schedule

This section will present the overall schedule for the HLT/DAQ up to LHC turn-on in 2007. The principal milestones coming from: detector needs for TDAQ in installation, TDAQ component production (in the case of custom components like the ROBin & the RoIB), component purchasing & associated tendering, component testing time etc. Probably require a MSproject style diagram plus associated details.

18.2 Commissioning

Description of first ideas for the commissioning process of the TDAQ as well as the steps in the detector commissioning which need TDAQ components.

18.2.1 TDAQ

How are the various elements of the system will be commissioned. What other elements will be required for this

18.2.2 Tools for detectors

More details on which elements of TDAQ will be required by the detectors, and when, to progress with their commissioning plans

18.3 Workplan up to June 2005

Here we should discuss in some detail the work needed to be done in the coming 2 years to arrive at decisions and final choices in the various areas of the system. The year 2003/4 should be presented in as much detail as possible, enumerating specific dates where possible. The work for the following year will probably contain somewhat less detail.

18.4 References

18-1

18-2

A Paper model results

A.1 LVL1 trigger menu

The exclusive rates for the different LVL1 trigger menu items are specified in Table A-1. E.m./gamma (EM, the I refers to ‘isolated’), muon (MU), jet (J) and hadron (TAU) RoIs are identified and labelled with the LVL1 transverse momentum threshold. XE refers to the LVL1 missing energy trigger. For a discussion of these menus see Chapter 4, "Physics selection strategy" (NB: 5 kHz of ‘Other items’ are not taken into account).

Table A-1 Exclusive rates for the LVL1 trigger menu items. For items where two possibilities are indicated, the latter one corresponds to the design luminosity menu item.

LVL1 Trigger menu item	Low luminosity (kHz)	Design luminosity (kHz)
MU20	0.8	4.0
2 MU6	0.2	1.0
MU10 + EM15I	0.1	0.4
EM25I (EM30I)	12.0	22.0
2 EM15I (2 EM20I)	4.0	5.0
J200 (J290)	0.2	0.2
3J90 (3J130)	0.2	0.2
4J65 (4J90)	0.2	0.2
J60 + XE60 (J100 + XE100)	0.4	0.5
TAU25 + XE30 (TAU60 + XE60)	2.0	1.0

A.2 Event fragment sizes

Estimates of the fragment sizes are presented in Table A-2. To each fragment a ROD header and trailer (together 48 Bytes) and a ROBIn header (56 Bytes) are added. A ROS subsystem in general concatenates several fragments and then adds a 52 Byte ROS header. The Data Collection software and the network protocol will add further headers (36 Bytes for the Data Collection software, 30 Bytes for raw Ethernet with VLAN tag) which are removed on receipt of the data.

Table A-2 Estimates of maximum data fragment sizes in bytes.

Subdetector	Number of ROLs	Participating in LVL2	Low luminosity (2×10^{33} cm^{-2} s^{-1})	Design luminosity (1×10^{34} cm^{-2} s^{-1})
Pixels	120	YES	200	500
SCT	92	YES	300	1100
TRT	256	YES	300	1200
E.m. calorimeter	724	YES	752	752
Hadron calorimeter	32 (Tilecal)	YES	752	752
	24 (LAr)	YES	752	752
Muon precision	192	YES	800	800
Muon trigger (RPCs and TGCs)	48	YES	380	380
CSC	32	NO	200	200
FCAL	16	NO	1400	1400
LVL1	56	NO	1200 (average)	1200 (average)
Total event size, raw			1006864	1346864
Total event size, with headers			1183352	1523352

A.3 Parameters relevant for LVL2 processing

The LVL2 processing consists of several steps and after each step a decision is taken on whether data from other subdetectors within the region-of-interest should be requested for further analysis. Table A-3 shows the subdetector data requested by different procesing steps of the LVL2 trigger for the four different types of RoIs. The associated acceptance factors are also specified in the table. As the LVL1 trigger defines a finite number of possible RoI locations, the data rates can be estimated using these factors along with information on the sizes and locations of the regions-of-interest, and the mapping of the detector on the ROBins. A small region in eta-phi space corresponds to each location. A hit in this region satisfying appropriate LVL1 trigger criteria generates a RoI with a location corresponding to the region. The relative RoI rate for each location is assumed to be proportional to the surface of this region, while the sum of the rates for all possible locations should be equal to the LVL1 menu RoI rate. This makes it possible to determine the rate for each possible location. In combination with the RoI sizes (see Table A-4) and the mapping of the detector on the ROBIns, the RoI data request rates for each ROBIn can be calculated.

In order to establish the processing resources needed for the LVL2 trigger the algorithm execution times and the overheads for sending requests and receiving data are needed. See Table A-5 for current estimates, assuming execution on 4 GHz machines. The numbers specified include estimates of the time needed for data preparation. Furthermore, for each Ethernet frame sent and received overheads of 4 µs and 8 µs respectively are taken into account. These values have been estimated from measurement results for the SFI (see Section 8.4.2.3). The processing step

resulting in a decision is assumed to take 50 µs. Merging of event fragments into a larger fragment suitable for input to the algorithms is assumed to proceed at 160 Mbyte/s.

Table A-3 Subdetector 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
EM	E.m. calorimeter	0.19 (design lum.: 0.16)	Hadron calorimeter	0.11 (design lum.: 0.16)	TRT/SCT/Pixels
JET	E.m. and hadron calorimeters	1.0			
TAU	E.m. and hadron calorimeters	0.2	TRT/SCT/Pixels		
MUON	Muon precision and trigger detectors	0.39	SCT/Pixels	0.086	E.m. and hadron calorimeters (only for design luminosity)

Table A-4 LVL2 RoI sizes

Type of RoI	Size in eta	Size in phi
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)

Table A-5 Estimated execution times (in ms) of LVL2 algorithm steps on a 4 GHz processor, for low and design luminosity respectively. The estimated time needed for data preparation has been included in the RoI processing times. The algorithm execution times are the m_95 values (see chapter...).

Type of RoI or trigger	Muon detectors	Calorimeters	TRT	SCT + Pixels
EM		0.088/0.123 (e.m.) 0.023/0.032 (hadron)	8.33/24.56	1.36/3.88
JET		0.68/0.68		
TAU		0.044/0.061 (e.m.) 0.011/0.016 (hadron)	8.33/24.56	1.36/3.88
MUON	0.5/0.5	0.044/0.061 (e.m.) 0.011/0.016 (hadron)	8.33/--	1.36/3.88

A.4 Parameters relevant for Event Builder and Event Filter

Events need to be fully built at a rate equal to the acceptance rate of the LVL2 trigger (0.6 or 1.5 kHz) and then to be analysed by the Event Filter. The Event Filter is expected to reduce the rate by a factor of 10 (see ch....) with a typical processing time of one second per event, which requires a farm of at least 300 or 750 dual-CPU PCs.

A.5 Data rate summaries

The LVL2 system and the Event Builder both send requests for data to the ROBIns. The rate of the requests from the Event Builder is equal to the event building rate, i.e. 0.6 or 1.5 kHz at nominal LVL1 trigger rate. The LVL2 request rate per ROBIn for a given subdetector, averaged over all ROBIns, and the average number of requests for the ROBIn with the highest average are presented in Table A-6. The total data volume (data sent to the LVL2 trigger and to the Event Builder) output per ROBIn for a given subdetector, averaged over all ROBIns, and the volume output for the ROBIn with the highest average are presented in Table A-7. Similar numbers are given in Table A-8 and Table A-9 for ROS units handling data from 12 ROLs (for each subdetector, groups of 12 ROLs have been formed, without regard for the partitioning of the subdetector; 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). Data Collection and Raw Ethernet wrappers have been taken into account in the data volumes output by the ROS units.

Table A-6 LVL2 request rate per ROBIn in kHz at nominal LVL1 rate. Here ‘overall average’ denotes an average over all ROBIns and ‘maximum average’ is an average for the ROBIn with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low (overall average)	0.02	0.04	0.41	0.27	0.03	0.11	0.13
Low (max. average)	0.04	0.06	1.49	0.40	0.04	0.15	0.20
Design (overall average)	0.10	0.22	0.60	0.31	0.01	0.27	0.34
Design (max. average)	0.20	0.30	2.19	0.45	0.02	0.37	0.49

.

Table A-7 Output data volume per ROBIn in Mbyte/s (LVL2 data and data sent to the Event Builder) at nominal LVL1 rate. Here 'overall average' denotes an average over all ROBIns and 'maximum average' is an average for the ROBIn with the highest average number of requests.

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadron calorimeter	TRT	SCT	Pixels
Low (overall average)	0.56	0.31	0.86	0.75	0.26	0.29	0.22
Low (max. average)	0.58	0.32	1.79	0.86	0.26	0.30	0.24
Design (overall average)	1.45	0.83	1.80	1.55	1.97	2.13	1.11
Design (max. average)	1.53	0.87	3.15	1.67	1.98	2.25	1.20

Table A-8 LVL2 request rate per ROS unit (12 ROLs) in kHz. Here 'overall average' denotes an average over all ROBIns and 'maximum average' is an average for the ROBIn with the highest average number of requests

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadr. calorimeter	TRT	SCT	Pixels
Low (overall average)	0.2	0.4	2.4	1.9	0.2	0.8	1.0
Low (max. average)	0.3	0.5	4.3	2.1	0.3	0.9	1.5
Design (overall average)	0.9	1.9	3.6	2.1	0.1	2.0	2.6
Design (max. average)	1.4	2.4	6.4	2.4	0.1	2.2	3.9

Table A-9 Output data volume per ROS unit (12 ROLs) in Mbyte/s (LVL2 data and data sent to the Event Builder) at nominal LVL1 rate. Here 'overall average' denotes an average over all ROBIns and 'maximum average' is an average for the ROBIn with the highest average number of requests.

Luminosity	Muon trigger	Muon precision	E.m. calorimeter	Hadron calorimeter	TRT	SCT	Pixels
Low (overall average)	6.9	3.9	10.9	9.4	3.2	3.7	2.9
Low (max. average)	7.1	4.0	14.1	10.1	3.2	3.8	3.1
Design (overall average)	17.9	10.5	22.5	19.3	24.3	26.6	14.0
Design (max. average)	18.6	10.8	27.2	20.1	24.3	27.5	14.9

A.6 Overview of paper model results

Table A-10 and Table A-11 contain an overview of results obtained with the paper model for the nominal LVL1 rate and extrapolated to 75 kHz LVL1 rate respectively.

Table A-10 Overview of paper model results for the nominal LVL1 trigger rate

	Low luminosity	Design luminosity
LVL1 trigger rate (kHz)	20.1	34.5
Average number of ROBins receiving a RoI request, per LVL1 trigger	17.9	16.2
Average number of groups of 12 ROBIns receiving a RoI request, per LVL1 trigger	9.0	8.5
Maximum average output bandwidth (for LVL2 and Event Builder data) per ROBIn (Mbyte/s)	1.8	3.2
Maximum average RoI request rate per ROBIn (kHz)	1.5	2.2
Maximum average output bandwidth (for LVL2 and Event Builder data) per 12 ROBIns (Mbyte/s)	14.1	27.2
Maximum average RoI request rate per 12 ROBIns (kHz)	4.3	6.4
Total bandwidth of LVL2 traffic (Gbyte/s)	0.31	0.52
LVL2 farm size	32	65
Fragment rate in = request rate out per L2PU (kHz)	5.7	4.5
Fragment volume in per L2PU (Mbyte/s)	9.9	7.7
Decision rate per L2PU (kHz)	0.63	0.53
Total bandwidth of traffic to Event Builder (Gbyte/s)	0.72	2.3
Event Building rate (kHz)	0.6	1.5
Number of SFIs required	12	38

Table A-11 Overview of paper model results for 75 kHz LVL1 trigger rate

	Low luminosity	Design luminosity
Average number of ROBIns receiving a RoI request, per LVL1 trigger	17.9	16.2
Average number of groups of 12 ROBIns receiving a RoI request, per LVL1 trigger	9.0	8.5
Maximum average output bandwidth (for LVL2 and Event Builder data) per ROBIn (Mbyte/s)	6.7	6.9
Maximum average RoI request rate per ROBIn (kHz)	5.6	4.8
Maximum average output bandwidth (for LVL2 and Event Builder data) per 12 ROBIns (Mbyte/s)	53	59.
Maximum average RoI request rate per 12 ROBIns (kHz)	16	14
Total bandwidth of LVL2 traffic (Gbyte/s)	1.2	1.1
LVL2 farm size	120	140
Fragment rate in = request rate out per L2PU (kHz)	5.7	4.5
Fragment volume in per L2PU (Mbyte/s)	9.8	8.0
Decision rate per L2PU (kHz)	0.63	0.54
Total bandwidth of traffic to Event Builder (Gbyte/s)	2.7	5.0
Event Building rate (kHz)	2.2	3.3
Number of SFIs required	45	82

B Glossary

B.1 Acronyms

B.2 Definitions

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