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. Failure, here, means the inability of a component, which may be hardware or software, to perform its intended function.

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

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

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

- 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 local measures are taken to correct failures. Local recovery mechanisms will not make important decisions, e.g. to stop the run, but pass the information on to higher levels.
- Errors are reported in a standardized way to make it easy to automate detection and handling of well-defined error situations (e.g. with an expert system).
- Errors are automatically logged and the record made available for later analysis if necessary. Where the error affects data quality the necessary information will be stored in the conditions database.

The following actions can be distinguished.

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

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

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

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

Error recovery describes the process of bringing the faulty component back into a functioning state. This might involve manual intervention by the shift crew, an expert, or an automated response initiated by the expert system. The time-scale of this phase will typically be longer than the previous ones and can range from seconds to days (e.g. in the case of 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 of errors. It is assumed that error conditions are detected by DataFlow applications, controllers, event-selection software, and monitoring tasks. These conditions may be caused by failures of hardware they control, of components that they communicate with, or these may occur internally.

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

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

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

6.3 Error-reporting mechanism

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

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

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

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

6.4 Error diagnosis and system-status verification

Regular verification of the system status and its correct functioning will be a vital operation in helping to avoid the occurrence of errors. A customizable diagnostics and verification framework will allow verification of the correct status of the TDAQ system before starting a run or between runs, automatically or on request. It will make use of a suite of custom test programs, 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 component where the failure occurred. This allows failures to be dealt with in subsystems without necessarily involving any action from other systems. This decentralizes the knowledge required to react appropriately to a failure and it allows experts to modify the error handling in their specific subsystem without having to worry about the consequences for the full system.

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

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

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

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

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

6.6 Error logging and error browsing

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

6.7 Data integrity

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

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

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

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

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

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

6.8 Fault-tolerance and failure strategy for hardware components

6.8.1 The source end of Readout Links

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

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

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

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

6.8.2 The Readout System

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

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

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

The PCs used to house the ROBin modules will be commercial rack-mounted industrial PCs. The selection criteria for the PCs will require high reliability with such features as redundant power supplies. Every ROS unit is completely independent from the others. Thus the ROS system itself does not need to be reconfigured in the case of a failure with a particular unit/PC, and

it can work with any number of missing PCs. However, any hardware intervention on a ROS PC will imply the loss of all the data from the connected ROLs, and the need to reconfigure the DAQ accordingly. For example, the TTC system will have to be reconfigured to ignore any busy signal coming from the RODs which were connected to the missing ROS. The replacement of a full ROS PC is expected to be, in the best case, at least one hour.

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

6.8.3 The RoI Builder

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

6.8.4 Network

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

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

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

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

6.8.5 Rack-mounted PCs

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

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

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

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

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

All of the above hardware problems are likely to require operator intervention at some point, but the system is designed to minimize the impact of the failure and to make replacement of the

faulty components as simple as possible - e.g. ensuring that individual PC's can be removed from racks with minimal disturbance to cabling.

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

6.9 Use Cases

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

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

bly accepted for more detailed offline analysis (as assigning the same event to another LVL2 node could possibly bring this node down too).

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

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

6.10 References

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

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

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