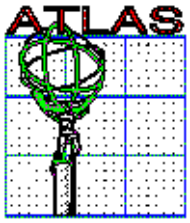

ATLAS
High-Level Triggers, DAQ and DCS

Technical Proposal

Issue:	1
Revision:	0
Reference:	CERN / LHCC / 2000–17
Created:	31 March 2000
Last modified:	19 September 2000
Prepared by:	ATLAS HLT / DAQ / DCS Group

ATLAS Collaboration

Armenia
Yerevan Physics Institute, Yerevan

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

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

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

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

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

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

CERN
European Laboratory for Particle Physics (CERN), Geneva

China
Institute of High Energy Physics, Academia Sinica, Beijing, University of Science and Technology of China, Hefei, University of Nanjing and University of Shandong

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

Denmark
Niels Bohr Institute, University of Copenhagen, Copenhagen

Finland
Helsinki Institute of Physics, Helsinki

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

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

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

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

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

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

Japan
Department of Information Science, Fukui University, Fukui
Hiroshima Institute of Technology, Hiroshima
Department of Physics, Hiroshima University, Higashi-Hiroshima
KEK, High Energy Accelerator Research Organisation, Tsukuba
Department of Physics, Faculty of Science, Kobe University, Kobe
Department of Physics, Kyoto University, Kyoto
Kyoto University of Education, Kyoto-shi
Department of Electrical Engineering, Nagasaki Institute of Applied Science, Nagasaki
Naruto University of Education, Naruto-shi
Department of Physics, Okayama University, Okayama
Department of Physics, Faculty of Science, Shinshu University, Matsumoto
International Center for Elementary Particle Physics, University of Tokyo, Tokyo
Physics Department, Tokyo Metropolitan University, Tokyo
Department of Applied Physics, Tokyo University of Agriculture and Technology, Tokyo
Institute of Physics, University of Tsukuba, Ibaraki

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

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

Norway
University of Bergen, Bergen
University of Oslo, Oslo

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

Portugal
Laboratorio de Instrumentação e Física Experimental de Partículas (University of Lisboa, University of Coimbra, University Católica-Figueira da Foz and University Nova de Lisboa), Lisbon

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

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

JINR
Joint Institute for Nuclear Research, Dubna

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

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

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

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

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

Taiwan
Institute of Physics, Academia Sinica, Taipei

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

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

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

Acknowledgments

The authors would like to thank Mario Ruggier for preparing the FrameMaker template upon which this document is based. The authors also warmly thank the CERN Desktop Publishing Service for their professional help in the organization of publishing the document, as well as the CERN Printshop staff for their helpful, friendly and efficient service.

Table Of Contents

1 Introduction

1.1 Purpose and Scope of the Technical Proposal

The Technical Proposal (TP) follows on from the work presented in the *ATLAS Technical Proposal* [1-1], published in December 1994; the *ATLAS DAQ, EF, LVL2 and DCS Technical Progress Report* [1-2], published in June 1998; the *ATLAS Trigger Performance Status Report* [1-3], published in August 1998; and the *ATLAS Detector and Physics Technical Design Report* (Chapter 11) [1-4], published in May 1999. The LVL1 trigger is not addressed in this document as its Technical Design Report (TDR) has already been published [1-5].

The purpose of the TP is threefold. Firstly, to summarize the requirements on the High-Level Triggers (HLT), Data Acquisition (DAQ) and Detector Control System (DCS) from the physics point of view and from external systems, namely the ATLAS detector systems, the LVL1 trigger and the offline software. Secondly, to summarize the work, measurements and conclusions of the DAQ/EF-1 Prototype Project, the LVL2 Pilot Project and the DCS group. Thirdly, based on the above, to propose a strategy for high-level triggering and an architecture for ATLAS High-Level Triggers, Data Acquisition and Detector Controls, and following this, to outline topics and issues which will have to be addressed between now and the submission of the Technical Design Report, currently scheduled for June 2001. Detailed technical choices and implementation issues are not discussed in this document.

1.2 Organization of the Technical Proposal

The document can be considered in three parts. The requirements placed on the HLT/DAQ/DCS system by physics constraints and external systems are explained in Part 1. Extensive summaries and conclusions of the work done in the various areas of the HLT/DAQ/DCS community can be found in Part 2. The reader wishing to study the proposed architecture for the HLT/DAQ/DCS system and analysis of future work can go directly to Part 3.

1.2.1 Part 1 – Requirements of Physics and External Systems

Chapter 2 summarizes the physics requirements and discusses briefly the challenging aims of the HLT/DAQ system to exploit the ATLAS physics potential. Chapter 3 examines the interfaces of the HLT/DAQ/DCS to the detectors and Chapter 4 discusses interfaces to other external systems (the LVL1 trigger and the Offline software).

1.2.2 Part 2 – Summary of work done in the HLT/DAQ/DCS Community

Chapter 5 discusses the results and conclusions of the DAQ/EF-1 Prototype Project which has been working on the challenging issues of high-rate and high-bandwidth data acquisition, experiment online control, and event filtering. Chapter 6 summarizes the work of the LVL2 Pilot Project, which has concentrated on the issues related to designing and building a LVL2 trigger

capable of dealing with a LVL1 trigger accept rate of up to 100 kHz. Chapter 7 addresses the work done in the DCS group and, in conjunction with the other LHC experiments in the context of the Joint Controls Project to develop a system capable of coordinating the control and monitoring of all the ATLAS detector. Chapter 8 summarizes the work done in the Physics and Event Selection Algorithm (PESA) group, and proposes an event-selection strategy for the HLTs.

1.2.3 Part 3 – Architecture Proposal and Issues for Further Work

Chapter 9, founded on the work described in the preceding chapters, proposes an architecture for the HLT / DAQ / DCS system. Following on from this, Chapter 10 then identifies some topics and issues which will have to be studied in the period leading up to the publication of the Technical Design Report.

In order to keep the present document to a reasonable size, many details of the work done have been omitted and references are made to supporting back-up documents. This is particularly true of Part 2.

1.3 References

1-1 *ATLAS technical proposal*, CERN / LHCC / 94–43 (1994)

1-2 *ATLAS DAQ, EF, LVL2 and DCS technical progress report*, CERN / LHCC / 98–16 (1998)

1-3 *ATLAS trigger performance status report*, CERN / LHCC / 98–15 (1998)

1-4 *ATLAS detector and physics performance technical design report*, CERN / LHCC / 99–14 (1999)

1-5 *ATLAS first-level trigger technical design report*, CERN / LHCC / 98–14 (1998)

2 Physics Requirements

2.1 Physics at the LHC

The LHC will provide proton–proton collisions at $\sqrt{s}$ = 14 TeV with a design luminosity of 10^{34} cm^{-2} s^{-1}. One of the main goals of ATLAS is to understand the mechanism of electroweak symmetry breaking (search for one or more Higgs bosons) and to search for new physics beyond the Standard Model. In addition, precision measurements will be performed for Standard Model processes (e.g. the masses of the W boson and of the top quark, the Cabibbo–Kobayashi–Maskawa(CKM) matrix elements and the proton structure), and for new particles (properties of the Higgs boson(s), properties of supersymmetric particles).

Measurements will be performed by detecting and measuring the momenta of charged leptons, photons, jets (with and without a b-tag) and missing transverse energy. The usage of these high-p_T objects will allow e.g. to search for Higgs bosons in a variety of decay modes and to perform precision measurements of the W boson and the top-quark mass, as well as of gauge-boson couplings. The reconstruction of exclusive decays of B hadrons into particles with relatively small transverse momentum is needed in order to determine elements of the CKM-matrix and to measure CP violation.

The range of cross-sections for the various processes is enormous: the total proton–proton inelastic cross-section is estimated to be 80 mb, whereas the cross-section for the production of a Standard Model Higgs boson with m_H = 120 GeV is about 20 pb. In addition, for several channels very small branching ratios have to be taken into account (e.g. the branching ratio for $H \rightarrow \gamma\gamma$ can be as small as 10^{-3}). The challenge is to provide an efficient rejection of high-rate backgrounds online, while maintaining excellent and unbiased efficiency even for rare signals.

At the LHC, two categories of final-state properties can be identified, which influence the requirements on the High-Level Triggers (HLTs). Firstly, there are events containing few (usually high-p_T) signatures, such as charged leptons or jets. As will be shown below, the typical minimal p_T value is larger than 15 GeV (this number depends on the type of signature). Secondly, there are events where the final state is made up of several low-p_T particles all of which have to be identified. These two categories put different demands on the strategy and the possible implementations of the HLT and DAQ system.

2.2 The ATLAS Detector

ATLAS is a general-purpose detector with a solid-angle acceptance close to 4π. The interaction region is surrounded by silicon-pixel and silicon-strip detectors, followed by a transition-radiation tracker (TRT). These tracking elements (called Inner Detector) have an pseudorapidity coverage of $|\eta| < 2.5$ and are located inside a 2 T solenoidal field. The solenoid is surrounded by calorimetry. For $|\eta| < 5$ the electromagnetic section consists of a liquid argon(LAr) calorimeter with a fine-grained section for $|\eta| < 2.5$ for precision measurements. The coverage of the hadronic calorimeter (a combination of scintillator tile and LAr components) also extends over $|\eta| < 5$. Muon detection and momentum measurement is performed with an air-core toroid system, equipped with muon chambers covering $|\eta| < 2.7$. In total the detector consists of more

than 10^8 electronic channels and the average event size is about 1–2 Mbyte. More details on the ATLAS detector and the performance of its components are given in Ref. [2-1].

2.3 Physics Requirements on the High-Level Triggers

Ref. [2-1] gives a description of many detailed physics studies for various processes which are envisaged at the LHC. Based on the event-selection criteria used in these studies, a list of physics process signatures has been derived by restricting the selection criteria to the main handles (p_T cuts, isolation measures, etc.), namely the ones which are more suitable for an online event selection. These characterizations are summarized in Ref. [2-2], where a fully organized layout of the physics requirements and of the corresponding selection strategy can be found. In the following, the underlying concept will be illustrated for a few examples. This compilation serves as input to the physics and event-selection strategy as documented in Chapter 8. Note that extensive studies of trigger performance have been already been presented in Ref. [2-3] and Chapter 11 of Ref. [2-1].

2.3.1 Physics Process Signatures

In the following, a few representative examples from Ref. [2-1] are given to illustrate the principle used in deriving the physics process signatures summarized in Ref. [2-2].

For the measurement of the top-quark mass, the inclusive single-lepton plus jets channel will be used. The selection, as documented in Section 18.1.3.1 of Ref. [2-1], requires an isolated lepton ($p_T > 20$ GeV, $|\eta| < 2.5$), $E_T^{miss} > 20$ GeV, at least four jets with $p_T > 40$ GeV and $|\eta| < 2.5$, out of which at least two have to be b-tagged. The analysis then further selects (non b-tagged) jet pairs to reconstruct the $W \rightarrow jj$ decay and combines these with a b-tagged jet to reconstruct the top decay. In this example, the essential signatures are the lepton, missing transverse energy and the jets with their associated properties.

A more exotic example is the associated production of the supersymmetric Higgs bosons *H* or *A* with two b-quarks, where the Higgs decays to two b-quarks, leading to a final state containing four b-jets. The detailed selection procedure is described in Section 19.3.2.8 of Ref. [2-1]; the essential criteria are four b-tagged jets with minimum transverse energies of 70, 50, 50 and 30 GeV (for a Higgs mass of $m_A = 300$ GeV). In the trigger menus for LVL1 and LVL2 documented in Section 11.7.3 of Ref. [2-1] most of these events would not be accepted due to high-p_T jet thresholds. If an increase of the LVL1 rate is affordable, then the acceptance for these channels could be increased by lowering the LVL1 thresholds and applying b-tagging at LVL2.

A different area of the study is the measurement of B-hadron decays and properties (referred to as B-physics), which are used to measure CP violation and elements of the CKM-matrix. Here the reconstruction of exclusive decays into particles with low momentum is required, which leads to more complex criteria than in the high-p_T cases discussed above.

2.3.2 Definition of the Key Characterizations

Based on the summary of physics analysis signatures, it is obvious that part of the key characterizations will be given by electrons, photons, muons, taus, jets (b-tagged or not) and missing

transverse energy. These objects have properties, of which the main ones are the transverse energy (or momentum), isolation and pseudorapidity. The transverse energy is typically of the order of a few tens of GeV, leading to the label 'high-p_T'.

2.3.3 Other Physics-Related Requirements

In the following, an overview is given of various criteria for event selection, where not every event passing the criteria might be accepted. Only a fraction (via prescaling) of such events could be written to mass-storage:

- Cross-section measurements

  In order to extend measurements of cross-sections over a wide kinematic range, it is necessary to accept a fraction of the events which contain, for example, jets with small transverse energies.

- Background studies

  To be able to obtain the properties of background processes from the data, it will be important to accept a fraction of the background events, this can be achieved by loosening the cuts.

- Calibration and alignment

  Special data-taking is foreseen on selected triggers to allow subdetectors to perform calibration and alignment tasks, e.g. with $Z \rightarrow e^+e^-$ or $Z \rightarrow \mu^+\mu^-$ events.

- Determination of trigger efficiencies

  To reduce the dependence of measurements on the modelling of proton–proton interactions and the detector response, it is desirable to be able to determine the trigger efficiency from the data. This can be achieved by accepting for relevant triggers part of the events, which are selected by a complementary trigger and by recording samples with lower threshold for a given trigger signature.

2.4 Summary

The selection by the HLT should be as inclusive as possible in order to maximize the discovery potential of ATLAS. Furthermore, the selection process has to be very flexible to adapt to luminosity variations over the duration of a fill, to changes of background conditions, and to incorporate new selection criteria based on observation of new phenomena. It should also allow more and more sophisticated selections to be done online, as the knowledge and understanding of the detector and the physics improves.

2.5 References

2-1 *ATLAS detector and physics performance technical design report*, CERN / LHCC 99–14 / 15 (1999)

2-2 *Physics requirements for the ATLAS high-level trigger*, ATLAS internal note, ATL-DAQ–2000–033 (2000)

2-3 *ATLAS trigger performance status report*, CERN / LHCC 98–15 (1998)

3 Detector Requirements

3.1 Introduction

In this chapter we examine some of the issues relevant to the interface between the detectors and the HLT/DAQ/DCS system. Given the overall ATLAS time-scale, many of the issues addressed here are incomplete or uncertain in their details. However the various issues have been discussed in the Detector Interface Group(DIG)[1] as well at a recent HLT/DAQ workshop [3-1], and what is presented here represents the best understanding we have at this time.

The official ATLAS policy to date has been that the ReadOut Drivers(RODs)[2] and their environment are the responsibility of the individual detector groups[3], while the ReadOut Buffers(ROBs)[4] are the responsibility of the central data acquisition group. The ReadOut Links(ROLs)[5] will be common across the detector systems and will be specified by the DAQ group and the detectors, but remain the responsibility of the detector groups [3-2]. Discussions in the ATLAS ROD Workshop held in December 1998 [3-3], involving representatives from all detector systems, led to the conclusion that many of the basic data acquisition requirements of the detectors in the ROD crate are similar or identical. This led to the proposal that ATLAS should strive for some degree of commonality for the DAQ functions within the detector's ROD crates. This would have the following very obvious advantages:

- Prevents multiple developments of the same functions.
- Minimizes the overall maintenance requirements.
- Minimizes the variety of hardware (e.g. controller CPUs) required in the ROD crates.
- Interfaces to external systems (e.g. databases) would be implemented once in a coherent fashion.
- Facilitate the final integration of each detector's acquisition system in the ATLAS HLT/DAQ/DCS system.

This proposal has been analysed in detail by the DIG during 1999 in terms of detector requirements. It was also discussed at the recent HLT/DAQ workshop [3-1], together with the closely connected issue of test-beam data acquisition for the detectors in the coming years. The following conclusions were agreed upon:

- The ROD-crate DAQ and test-beam DAQ systems will be based on an evolution of the appropriate elements of the DAQ/EF -1 system.

1. The DIG is a forum for discussion between representatives of the ATLAS detectors and the Trigger/DAQ.
2. Functional element which gathers data from the derandomizer buffers over one or more data streams and builds pieces of events to be sent to the ROB.
3. Note that for the purposes of this discussion, the LVL1 trigger acts in the same way as a detector subsystem.
4. Standard module which receives data from the ROD, stores them and makes them available to the LVL2 trigger, and, for LVL2-selected events, to the Event Filter.
5. Physical link between ROD and ROB through which the data are sent at the event rate of the LVL1 trigger.

- They must be viewed as a part of the overall ATLAS HLT / DAQ / DCS system, and must not become isolated or disconnected.
- Developments of the ROD-crate DAQ and test-beam DAQ must proceed in unison with the *mainstream* developments.

Following these conclusions, the DAQ group has started to liaise with the detector groups to analyse the requirements with a view to making a considered proposal for a common DAQ in the ROD crates as soon as possible.

3.2 Functional Requirements

3.2.1 Data Acquisition

We present here a list of basic functional requirements for data acquisition in the ROD crate, which reflect the statements made in the previous section.

- A local detector data acquisition system (DetDAQ) is required in the ROD crate, running on its CPU.
- The DetDAQ should be common to all ATLAS systems.
- Common software and hardware components should be used to guarantee maximum uniformity throughout ATLAS, adapted where necessary to the particular detector requirements.

3.2.1.1 Access to Databases

Various classes of parameters will need to be accessed at the ROD-crate level from databases, notably detector parameters and initialization parameters, detector geometry and calibration constants, and DAQ configuration parameters. The following requirements on databases have been noted by the DIG:

- All configuration data relevant for the DetDAQ should be stored in a database.

  This database must also be accessible to the offline where necessary.
- Read / write access is required to the database from the ROD-crate CPU.

  This is required both at configuration time (to boot and load the electronics), and at run time (to store and retrieve information). Examples of data to be stored and received are: front-end electronics parameters and software, ROD parameters and software, calibration information and data, and monitoring data (including histograms).
- It must be easily possible to ensure that calibration parameters produced in the ROD crate are available coherently to other systems which require them.
- Information in the database should be time-stamped to facilitate parameter monitoring as a function of time (or run-stamped for some parameters).
- A reference scheme is required in order to track changes in the detector behaviour and set up, to be able to compare monitoring histograms, and to validate new calibration values.

3.2.1.2 Normal Running

In normal running mode a detector forms part of the *active* experiment's readout. Standard monitoring and continuing non-intrusive calibration functions will go on in parallel with data-taking. Local monitoring functions will take place in both the ROD crate and at the level of the ROBs whereas procedures requiring data from all elements of a given subdetector, or full event data from the entire ATLAS detector, will be run in the Event Filter.

3.2.1.3 Standalone Running

Standalone mode is defined as the case when all or part of a detector's readout system does not form part of the *active* experiment's readout, but operates independently with its own trigger and dead time logic. The readout may be split into several independent and possibly concurrent partitions.

The ability to partition the system is an important detector requirement. Each partition is a fully independent system. Several partitions can be run and operated in parallel with other partitions. The following requirements have been identified in this area:

- DetDAQ partitioning is required at the ROD-crate level.
- The maximum number of partitions is defined by the number of TTC (Timing, Trigger and Control) partitions available [3-4].
- Different DetDAQ partitions should be able to run simultaneously and independently.
- The DetDAQ should be capable of running on a single partition (with one or several ROD crates).

  Individual crates may be *masked* out of a partition to allow local specialized debugging operations. Event building between multiple ROD crates may be necessary for detector installation, commissioning and calibration, and could be used as an additional monitoring resource during normal running provided that the main flow of data is not perturbed.
- The DetDAQ should be capable of running both coherently with and independently of the main ATLAS DAQ system
- No event-building functionality is required at the ROD level for the main data stream

Note that further aspects of partitioning are addressed in Section 3.2.2.

3.2.1.4 Test Beam

The detectors have a wide spectrum of test-beam plans for the period running up to installation and commissioning. It is important that they have a stable and upgradeable DAQ system for the test-beam work. With this in mind, it was decided that future testbeam DAQ would be based on the DAQ/EF -1 system, and that developments in the test-beam area would be coordinated and coherent with those in the main ATLAS DAQ. This decision is also useful to encourage ROD software development work within the framework of something approaching the final DAQ.

3.2.1.5 DAQ–DCS Interface

The interaction between the DAQ and the DCS is discussed in Chapter 7. We present here the principal requirements:

- Access to the DCS is required at the ROD crate level. This access should be as direct as possible.
- The DCS should be partitionable, and should be partitioned, if possible, in a similar fashion to the DAQ.

3.2.1.6 Interface between the DCS and Detector Configuration

The DCS includes: environmental monitoring; control and monitoring of detector support systems such as high voltage and gas; configuration, control and monitoring of the detector electronics. The first two functions have a well-defined separation from DAQ (although not a complete separation, e.g. data readout during calibration as a function of high voltage). The third function addresses calibration runs, and standalone procedures for diagnosis, debugging and set-up. To implement this functionality, some detectors will use the data readout path and others will use the same path (CANbus) for the environmental monitoring. The muon detector electronics, for example, are spread out over a very large area with environmental sensors in proximity to the electronics. Many hundreds of CANbus nodes are required for the environmental monitoring. It makes sense to use the same nodes and high-reliability CANbus network for both functions. Other more compact detectors may prefer using the data readout path (via the ROL). Either approach requires that there be an interface between the DCS and the ROD crate.

When the event data readout path is used, the ROD crate must respond to configuration requests from the DCS. When the CANbus pathway is used, the DCS must transport the interaction between the ROD crate and its detector electronics. In both cases this implies that a detector's DCS be partitioned compatibly with its DAQ partitioning.

3.2.2 Triggering Aspects

3.2.2.1 Introduction

The ATLAS readout elements, such as the front-end electronics, the RODs and possibly the ROBs, need the bunch-crossing signal (BC) and the LVL1 accept signal (L1A). The TTC system allows these signals to be distributed to the readout electronics elements. The timing signals comprise the LHC clock (BC) and the synchronization signals (BCR, ECR). The trigger signals include the L1A, test and calibration triggers. The TTC system allows the timing of these signals to be adjusted, and is described in some detail in Ref. [3-4].

The way the TTC system will be used in different subdetectors depends on the specific requirements of each of them. Most of the subsystems will use more than one partition to allow concurrent running of different parts of the detector in different trigger modes during commissioning or calibration periods.

In ATLAS, the TTC system will be used in different ways:

- In normal running

  Each TTC partition receives its clock from the LHC machine and the L1A from the LVL1 Central Trigger Processor(CTP). The BCR is derived from the LHC ORBIT signal. After each L1A, an 8-bit trigger type is forwarded to the destinations as well as (optionally) a 24-bit event identifier (L1ID). The trigger type is formed in the CTP and contains information on what gave rise to an L1A, while the L1ID is formed in each TTC–VME interface (TTCvi). The TTC system can also transmit subdetector-specific data and commands without introducing dead time, e.g. fire test pulses when there are no bunches of protons (LHC gap), or load front-end parameters (e.g. delay values).

- During commissioning and for test and calibration runs

  Triggers can be injected locally in each TTC partition under the responsibility of the detector groups.

The ATLAS front-end electronics and readout systems contain many levels of buffering. Information may be lost at any of a number of stages of the readout chain if buffers become saturated. Different strategies can be adopted to handle this situation, the two extreme ones being:

- Introduce dead time to avoid uncontrolled information loss.
- Accept information loss and build a readout system able to accept incomplete events and possible loss of synchronization.

The first of these strategies has been chosen and it has been decided to introduce dead time in the CTP in order to:

- Easily control and monitor the dead time of the experiment.
- Have a relatively simple and safe readout system relying on the presence of data for every event.
- Simplify the front-end electronics systems by imposing an upper limit on the event rate and a minimum time between consecutive events.

Despite the introduction of dead time, exceptional cases can arise where buffers become full, e.g. in the tracking detectors that produce variable-length data. Means of handling such situations are described in [3-2].

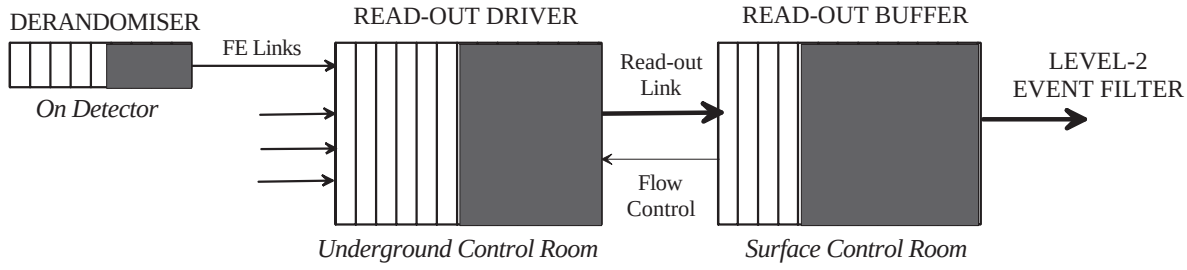

Figure 3-1 Buffers in the readout chain.

As shown in Figure 3-1, there are buffers at different places in the readout chain:

- At the front-end level

  A derandomizer buffer, located just after the LVL1 pipeline, can store data for a few events. This is needed in order to match the limited bandwidth of the front-end links, designed to cope with the average LVL1 trigger rate, to the random arrival time of the L1A.

- In the RODs

  There are buffers at the input stage before the data processing is done and/or at the output stage before the readout link.

- In the ROBs

  The data are stored until the LVL2 trigger has made a decision.

Any of these buffers can in principle become full, so some dead time must be introduced if data loss is to be avoided. The dead time is introduced in three ways:

1. A short dead time of four bunch crossings after each L1A is systematically introduced in the CTP to accommodate front-end electronics limitations.
2. The CTP limits the number of L1A signals that can be generated within a given period of time to prevent derandomizer overflows.
3. The CTP can be vetoed with an external signal to handle the occupancy of the ROD and ROB buffers.

The third mechanism is described in the following. Note that both the TTC and the dead-time handling are partitionable.

3.2.2.2 Dead-Time Handling in the RODs and ROBs

In the current estimate, there is a total of about 1500 RODs in ATLAS. Most of these modules will be located in crates in the underground control room (USA15). They contain buffers which can fill up. These modules have to produce a signal (ROD_BUSY) when their buffer is close to being full, in which case dead time must be introduced. The logical OR of these ROD_BUSY signals will be used to veto the CTP during normal running.

In order to avoid having up to 1500 signals to be ORed and monitored at the CTP level, a module (ROD_BUSY module - see Section 3.2.2.3) will be provided to subdetector groups.

For the ROBs, an identical strategy to that of the RODs could be applied. However, as the ROBs are located in the surface counting room and the CTP in the underground counting room, it is not deemed a very practical solution. Furthermore, dead time could be introduced too early as a given ROB which is filling up does not know about the availability of buffer space in the ROD to which it is connected.

It is planned to use a flow-control mechanism of the readout link to apply back pressure on the ROD. If a ROB has no available space in its buffer it disables the data transmission from the ROD. The buffer of the ROD may then fill up, in which case the ROD_BUSY will be asserted in due course (if necessary).

3.2.2.3 ROD_BUSY Module

As mentioned earlier, there will be about 1500 RODs in ATLAS, each of them providing a ROD_BUSY signal and capable of introducing dead time. It is therefore very important to:

- Monitor and control all the ROD_BUSY signals so that any pathological ROD can be prevented from introducing dead time.
- Monitor the duration of these signals.
- Gather the ROD_BUSY signals in a tree structure.

  This way, only one BUSY signal per subdetector appears at the level of the central trigger logic. An additional ROD_BUSY module will handle these signals and provide a VETO signal to the CTP.

A VME module handling up to 16 ROD_BUSY signals will be made available.

Each input BUSY_IN signal can be individually masked and its integrated duration is measured. A 16-bit counter, clocked by the BC divided by four (100 ns), associated with a 512-word deep FIFO, will be used to maintain the dead time history of the last three seconds.

The logical OR of the unmasked BUSY_IN signals is made available as an output. The state of the BUSY_IN inputs can be read out through the VME interface for debug purposes. A VME interrupt can be issued if the duration of one of the BUSY_IN signal exceeds a time limit, in order to warn the processor controlling this module that action has to be taken.

In order to be able to chain the modules in a tree structure, a CARRY input is made available. This input is connected to the BUSY_OUT of the previous module in the tree.

At the central trigger logic level, an additional module of the same kind is used to handle one ROD_BUSY signal per subdetector, and its output is used to veto the CTP.

Such a structure allows efficient control of the dead time in the experiment and provides an easy way to detect a faulty module introducing dead time. It also allows the partitioning of the readout, as a subdetector BUSY contribution can easily be removed from the CTP VETO. In the same way, additional subpartitions within subdetectors can be implemented.

It should be noted that in some cases, the functionality of the VME busy module will be implemented on other boards, but using the same design - i.e. not all detectors will use the busy module for all their busy handling.

3.2.2.4 Partitioning

A partition is a subset of the experiment with the capability to run independently (see Section 3.2.1.3). It can be a complete subdetector, a part of a subdetector or the combination of several subdetectors (in other words, several partitions can be combined into a higher-level partition). The data-taking can be done at the level of the RODs or through the complete DAQ system. A partition requires an independent TTC system and an independent handling of the dead time.

Figure 3-2 shows an example of two sub-detectors (Det1 and Det2), each divided into four partitions. Each of these partitions has its own TTC system (TTCvi and TTC crate) and generates its own BUSY signal. Each partition can operate autonomously, generating private TTC signals (lo-

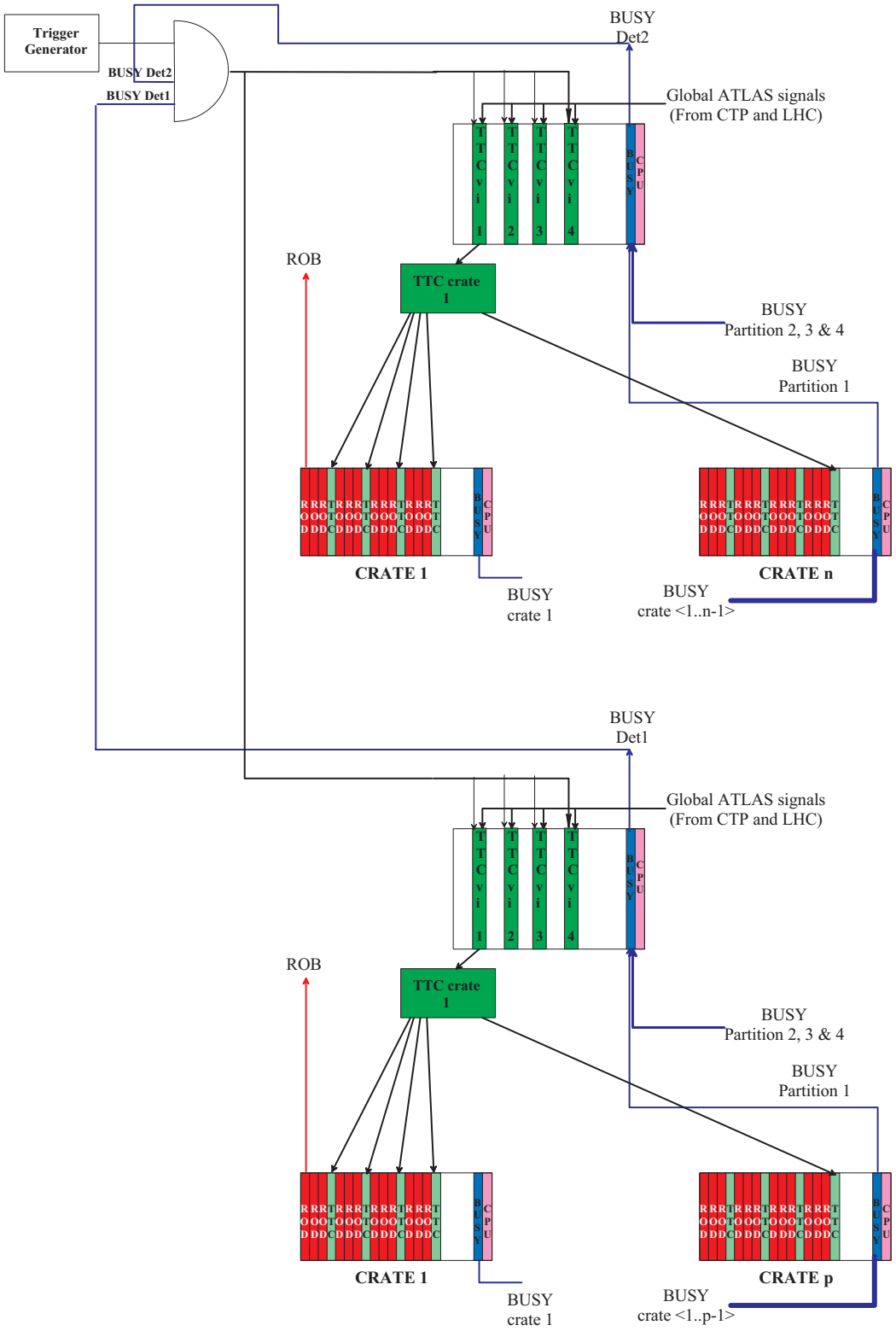

Figure 3-2 Example of two subdetectors each divided into four partitions.

cal trigger), and with the BUSY signal being used locally to control the trigger rate as shown in Figure 3-3. Alternatively, two or more partitions can be combined, using the global TTC signals

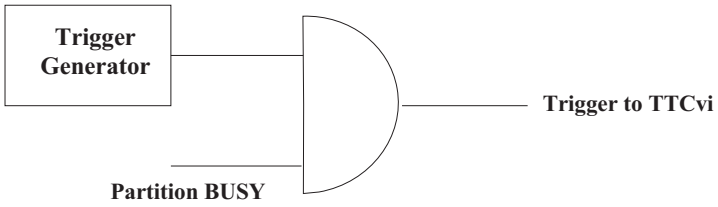

Figure 3-3 Control principle of the trigger of a partition.

of the experiment (L1A coming from the CTP) and using the BUSY signals in the global trigger logic.

If the data-taking for a partition is done at the level of the RODs, nothing more is required. If it is done by the main DAQ at the level of the ROBs, the BUSY signals of the ROBs have to be taken into account; if the ROB crates receive the TTC signals, they must be in the same TTC partition as the corresponding RODs.

3.3 Interface Requirements

3.3.1 Front-End Interfaces

The interface of the HLT/DAQ/DCS system to the detector front-end electronics is defined at various levels (data and control paths, trigger and busy signals, etc.). A summary of the current understanding of this interface can be found in Ref. [3-2]. Note that the specific interfaces between the LVL1 trigger and the detectors have been addressed elsewhere, see Ref. [3-4].

3.3.2 ReadOut Links

3.3.2.1 Introduction

The principal use of the ReadOut Link (ROL) is to connect the RODs to the ROBs. However, other areas of the HLT/DAQ system, for example the LVL1/LVL2 interface, are planning to use the same link since it meets the requirements and a separate development is not justified. Ref. [3-2] details the advantages of having common ROD outputs and ROLs for all subdetectors.

3.3.2.2 Prototype Work

A specification for a prototype ROL interface, the S-LINK [3-5], was drafted in 1995. This interface specifies the signalling and protocol of each end of a link and recommends a connector and mezzanine card. Based on this specification a family of links and test equipment has been designed and commercialized. The family includes optical and electrical links, test modules and interfaces to PCI/PMC bus. These components have been successfully used in the LVL2 Pilot Project, in DAQ/EF -1 and in several test beams. In addition all prototype RODs will soon be equipped with S-LINK outputs and prototype ROBs will have S-LINK inputs.

3.3.2.3 User Requirements of the ReadOut Link

During the 1998 ROD workshop [3-3] the user requirements of the ROLs were refined. They can be summarized as:

- Data width and rate: 32 bits at a maximum of 40.08 MHz (i.e. LHC bunch-crossing rate).
- Control bit to identify start and end of event.
- Xon / Xoff flow control.
- Error detection.
- Error rate < 10^{-12}.
- Maximum length: 300 m for optical version, 25 m for electrical version.

3.3.3 Event Format

The internal details of the individual detector's data content and format are of no direct relevance to the DAQ, but are, however, extremely relevant to the LVL2 trigger as they have a strong bearing on the computing resources required for the LVL2 trigger algorithms. This issue is addressed in the following section. The overall ATLAS data format in terms of the event and detector headers, book-keeping information, general DAQ parameters and link data is, however, important also for the DAQ. The detailed event format used by DAQ/EF-1 is presented in Ref. [3-6], and forms the basis of a proposal for the ATLAS event raw data format.

3.3.4 Data Mapping into the ReadOut Drivers

A proper organization of the mapping of the front-end readout channels into the RODs is a key element to optimize and simplify the extraction of Region-of-Interest (RoI) data for the LVL2 processing. The basic principle for optimized mapping at the ROD level is to organize the readout channels of each subdetector in groups which match LVL1 trigger towers. The ROB is a natural element for such groupings. The current specifications are detailed in Ref. [3-7]. This chapter summarizes the mapping for each subdetector. Table 3-1 summarizes the number of RODs and the expected average event fragment size per ROD (for high luminosity), per detector. This yields an estimated full event size of ~ 2 Mbyte.

Table 3-1 Number of RODs and average event fragment size per ROD.

Detector	Number of RODs	Average event fragment size per ROD (Kbit)
TRT	256	8
SCT	92	13
Pixel	110	15
LAr	794	14
TILE	64	7
MDT	192	6
CSC	8	2
RPC/TGC	48	~ 0
Total RODs	1564	

The Transition Radiation Tracker (TRT) is physically divided into half-barrels and end-caps. Each half-barrel contains 32 projective readout wedges and each end-cap is logically divided into 96 sectors, giving a total of 256 RODs.

For the Silicon Strip Semiconductor Tracker (SCT), a similar projective approach is now assumed for the readout. Elementary units of readout channels from the four layers of the barrel are grouped to form projective wedges with approximately equal φ coverage. The barrel is thus divided into 22 segments per half-barrel, and each end-cap into eight octants in φ and three groups across the nine wheels, giving a total of 92 RODs.

The Pixel detector is also read out in φ-regions. In each end-cap, each of eleven RODs will receive data from a small number of modules from each disk. For the outer two barrel layers, each ROD will receive data from two complete staves, giving 49 RODs. The B-layer is treated separately due to mechanical differences and its limited lifetime. Its 234 modules are distributed among 39 RODs for a total of 110 RODs in the Pixel system.

For the liquid-argon calorimeter (LAr), *layer* and *tower* mappings have been compared. Mapping by *towers* has the advantage of building a full trigger tower at the ROD level, but the disadvantage of a complex connection between the front-end board and the ROD. In addition, applying the calorimeter algorithm sequentially to layers allows a reduction in the average number of ROBs needed to be accessed. Consequently, *layer* mapping is now assumed for the readout of the LAr calorimeter. The entire LAr calorimeter (electromagnetic and hadronic end-caps) readout uses a total of 794 RODs.

In contrast, the scintillator-tile hadronic barrel calorimeter (TILE) is organized into 64 projective sectors, one per ROD.

The readout chosen for the Precision Muon Chambers (MDT) has the barrel arranged into projective towers and the end-cap as overlapping layers, with a total of 192 RODs. The Cathode Strip Chambers (CSC) consist of two end-caps, each of which will be instrumented with four RODs (eight in total). Each ROD will instrument one quarter of the solid angle subtended by an end-cap. The Muon Trigger Chamber barrel (RPC) and end-caps (TGC) follow the mapping of the LVL1 trigger towers with a total of 48 RODs.

3.4 References

3-1 *Beatenberg HLT/DAQ/DCS Workshop*, 6–10 December 1999, `http://lnxatd01.cern.ch/~beck/Workshop/Welcome.html`

3-2 *Trigger & DAQ interfaces with front-end systems: requirement document version 2.0*, ATLAS internal note, ATL–DAQ–98–103 (1998)

3-3 *ROD Workshop*, Geneva, December 1998, `http://schp5.unige.ch/atlas/atlaspage/workshops/rod98/rod_workshop`

3-4 *ATLAS first-level trigger technical design report*, CERN / LHCC / 98–14 (1998)

3-5 R.A. McLaren et al., *An application of S-LINK, a data link interface specification, in the ATLAS readout system*, presented at the 2nd International Data Acquisition Workshop (DAQ96) on Networked Data Acquisition systems, RCNP Osaka, Japan, November 1996

3-6 T*he event format in the ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–98–129 (1998)

3-7 *Detector and readout specification, and buffer-RoI relations, for LVL2 studies*, ATLAS internal note, ATL–DAQ–99–014 (1999)

4 External Interfaces

4.1 Introduction

The external interfaces to the HLT/DAQ/DCS system are indicated in the context diagram shown in Figure 4-1. The LVL1 trigger provides LVL2 with region-of-interest(RoI) and other data needed to guide the LVL2-trigger data selection and processing; this interface is discussed in detail in Section 4.2. The Timing, Trigger and Control (TTC) system provides signals associated with events that are selected by the LVL1 trigger, as discussed in Section 3.3.2. 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 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.

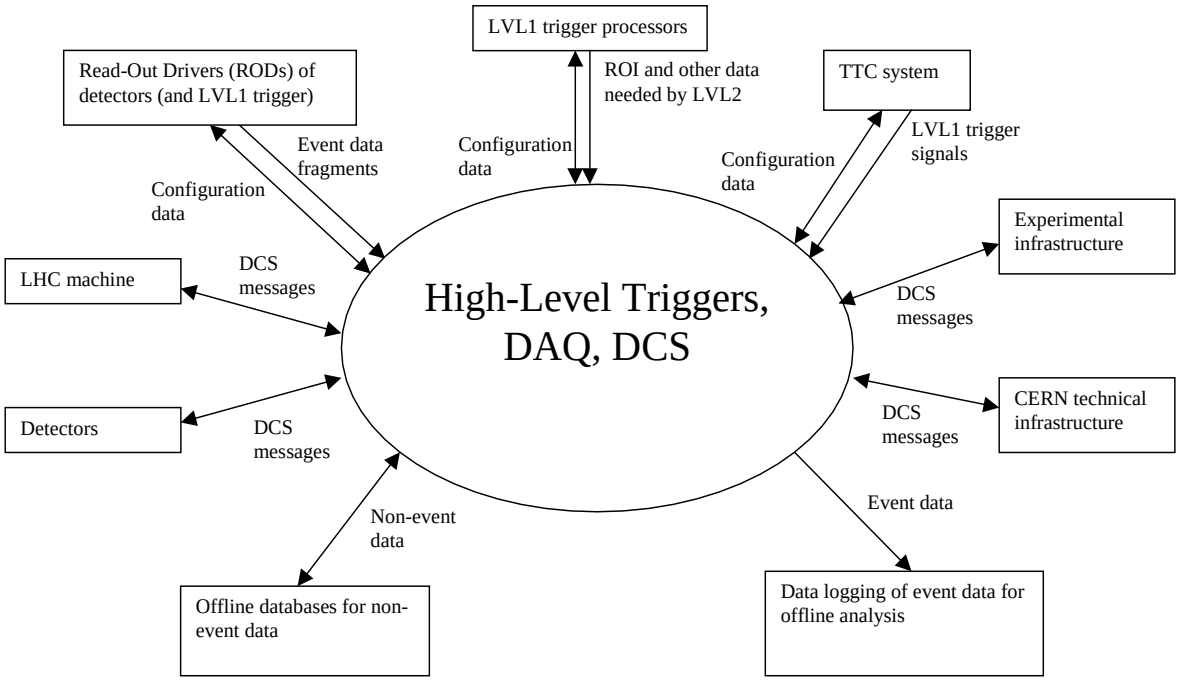

Figure 4-1 Context diagram of high-level triggers, DAQ and DCS.

The DCS components of the HLT/DAQ/DCS system have a number of external interfaces, as illustrated in Figure 4-1. 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 HLT system, 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 in the Event Filter.

This chapter addresses the external interfaces other than those to the RODs of the detectors, which are discussed separately in Chapter 3, and those associated with DCS, which are discussed in Chapter 7.

4.2 LVL1–LVL2 Interface

4.2.1 Introduction

The central idea in ATLAS LVL2 triggering is to use RoIs found at LVL1 as starting points for LVL2 triggering. For the two detector systems participating in LVL1 triggering, the muon and calorimeter, the RoI data indicate the positions of potentially interesting features of events found by LVL1, along with the threshold values passed. To make sense of these data the LVL2 system must have access to configuration databases where these items are defined. RoI data are also generated by the LVL1 Central Trigger Processor (CTP) to indicate the triggers satisfied by the event, and some global quantities such as missing transverse energy are also generated by LVL1 and passed to LVL2 along with the RoIs. The data interconnections to accomplish this are shown in a schematic way in Figure 4-2. The individual elements in this diagram are discussed below. A more detailed treatment of the subject can be found in Ref. [4-1].

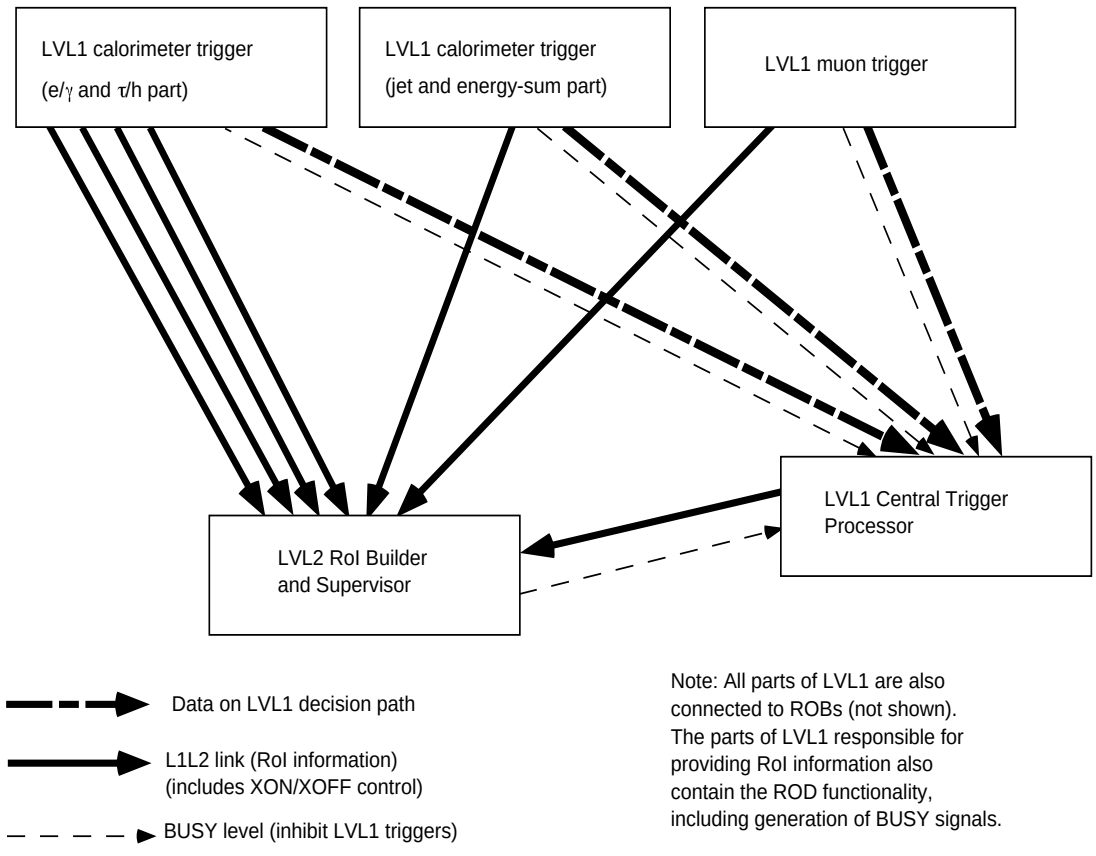

Figure 4-2 Block diagram of LVL1–LVL2 interface.

4.2.2 LVL1 Trigger System

RoI signals are generated by elements of the LVL1 trigger system as indicated in Figure 4-2. Shown are the links from the calorimeter trigger which transfer RoI and global data, a link from the muon system and a separate link from the CTP.

4.2.3 RoI Builder

The data from LVL1 are received by the RoI Builder (RoIB) [4-2], an element of the LVL2 trigger system that is connected to the LVL1 trigger system by a number of fast links (currently seven S-LINKs). A block diagram showing the RoIB/Supervisor structure and associated connections can be seen in Figure 4-3. The data from LVL1 are received in parallel on RoIB cards. Each card is connected to (at most) two RoI processors of the LVL2 Trigger Supervisor. The design is scalable by adding more RoIB cards and more RoI processors.

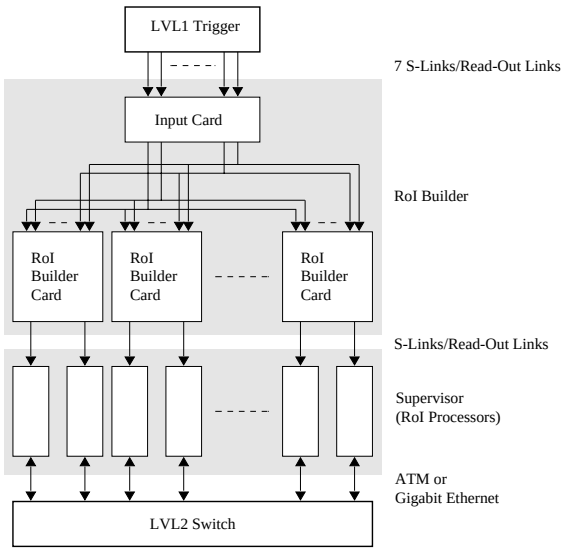

Figure 4-3 Block diagram of RoIB/Supervisor system.

On a LVL1 accept signal (L1A), the LVL1 trigger prepares and transfers to LVL2 RoI fragments that pinpoint interesting features of the event in specific subdetectors used by LVL1. The transfer is via fast links with event formatting following the standard ROD protocols with the event number embedded in every fragment. The RoIB receives these event fragments, assembles them into event-specific records, and selects an RoI Processor to which it transfers the record. The RoIB is capable of operating at the full LVL1 accept rate (100 kHz). To achieve the necessary performance the design is done in hardware making extensive use of large-scale FPGAs. The RoIB is discussed further in Section 6.5 and Section 9.3.2.

4.2.4 LVL1-to-LVL2 Links

The following LVL1-to-LVL2 links are foreseen:

- Muon trigger

  A single link carries all of the information from the muon trigger to the RoIB.

- Calorimeter trigger

  Four separate, geographically determined, physical links carry the e/γ and $\tau/$hadron RoI information to the RoIB, while a single physical link carries both the jet and energy-sum information.

- Central Trigger Processor

  A single link carries all the information from the CTP.

The data sent on the links are required to conform to the standard ATLAS ROD data format [4-1], [4-3].

4.2.4.1 Event Type

The field *Detector event type* is available for labelling event types. Current plans are to use it to distinguish:

- Physics events.
- Test events (artificially generated).
- Events monitored through the LVL1 Trigger ROD system with help of a special flag.
- Cosmic-ray events.
- Calibration events.

4.2.4.2 Status Words

In the ATLAS ROD data format a non-zero first status word indicates possible corrupted data.

Two kinds of status information must be transferred to LVL2:

- Indication of whether or not the corresponding data fragment is corrupted.
- Status information useful for interpreting the content of the data block.

A 32-bit word is proposed for each of these. In the first instance a 0 indicates good data. Both words are split into two 16-bit pieces, one common the other private, with the 16 bits in the common piece having the same meaning for every LVL1 subsystem. The private piece would have bits with meaning only to the specific LVL1 subsystem.

4.2.4.3 RoI Data Elements

The RoI data for both the muon and calorimeter systems is embedded in a single 32-bit word per RoI. In the case of muons the transferred information contains, among other things, the p_T threshold and the geographical location of the muon passing the LVL1 trigger criteria. Similarly, for the calorimeter RoIs, the 32-bit word contains the threshold set passed and the location of the RoI in the calorimeter.

The energy-sum data are packed into two 32-bit words, the first containing the 16-bit x and y components of the missing energy, and the second the summed scalar transverse energy, along with the thresholds passed by these global quantities.

4.3 LVL1–ROB Interfaces

As mentioned in Chapter 3, the LVL1 trigger produces data that have to be read out. There are no special requirements for this interface, which follows the same requirements and specification as for detector readout. The LVL1 trigger RODs provide data to the associated ROBs over standard readout links, following the requirements documented in Ref. [4-3].

If required, the LVL1 trigger may provide the L1A directly to the ROBs. This may be useful, for example, during partitioned calibration runs. This could be achieved by adding extra destinations in the appropriate TTC zones. Here, the ROBs would receive the same TTC signals as the corresponding RODs.

4.4 Interface to the Offline

There is a close, bidirectional relationship between Trigger/DAQ and offline computing. Trigger/DAQ depends on the provision of suitable offline reconstruction algorithms for use in the EF, and a framework in which to perform LVL1 trigger simulation, develop LVL2 algorithms and evaluate EF performance. Physicists require a good trigger simulation to be able to estimate the effects of the online selections on their analyses. The close collaboration necessary between offline and online software is positively evolving, based on the efforts made in the past, and is being improved with the new ATLAS Computing Steering Group, where two aspects of representation for the online are foreseen. One representative follows issues related to the code developed in the normal offline context (e.g. ATRIG [4-4] and its successors), the other deals with computing issues which are relevant for the offline software development performed in other systems (e.g. EF code). The synergism between these two aspects of interaction with the offline world is mandatory to achieve a coherent approach to this problem, in order to progress smoothly in an integrated way.

4.4.1 Software Architectural Considerations

The ATLAS Computing Review [4-5] and the Architecture Task Force [4-6] have given general guidelines along which the new implementation of the ATLAS offline software should develop. Among the conclusions which can be found in the reviews, the following points are worth noting:

- The choice of the Object Oriented(OO) approach, and corresponding paradigm.
- C++ as the implementation language for infrastructure packages, including the general framework (but keeping an eye on Java).
- A single source of detector-description related information for all applications.
- The separation of transient and persistent worlds, in order to be independent of the underlying OODBMS technology.
- A Transient Event Data Store used to communicate objects between modules, with access on demand.

From these points it is natural to extract some areas which should be carefully monitored by online algorithmic software developers, during the evolution of the new global software architecture. In particular the role of the *event* has to be decided in agreement with the online community because of its strong connection to the raw data. The relationship with offline here is not only at the level of the raw data format. One also has to clarify the need for an interface layer at the EF level, to build the “objects” needed by the reconstruction steps, and to understand the overhead implied by this conversion mechanism.

Another interface layer that has to be addressed coherently is the use of the many databases needed by the online processing. One is obviously the event data store, where the issue of

maximizing the output rate while preserving full data integrity with no drawback on online operations is the most important one. The other data stores are usually referred to as condition (or non-event) databases and span a wide range of time-stamped or run-stamped information, including calibration / alignment constants and magnetic field maps, DCS results (detector status, temperatures, voltages, accelerator condition, etc), data-taking configurations for each run, and many others. All these databases will need to be accessed (often in read / write mode) by several clients concurrently at all trigger levels, hence imposing strict requirements on the access methodology.

As a general architectural point one should note that a fully working suite of program modules, which includes LVL1 simulation, and LVL2 and EF algorithms as preselection to the offline reconstruction, all in the same architecture, is a mandatory goal. To achieve this, the LVL1 trigger simulation, the LVL2 selection and the EF reconstruction should be included from the beginning in the development of the new software architecture, as pluggable modules. The final aim is to allow each physicist to perform any of the following actions:

- Run purely the reconstruction software without preselection.
- Study trigger efficiencies at any level, by plugging in the relevant modules.
- Derive the EF implementation of the algorithms to be used online (e.g. by selecting a specific *job control option* set from a configuration database).
- Run the full suite by first simulating the online response (LVL1+LVL2+EF), and then applying the offline reconstruction and analysis chain.

This would be a fundamental step towards the understanding of the response of the detector as a whole, and will be beneficial for the many physics analyses that ATLAS wishes to perform.

4.4.2 Software Issues

The development of software used to simulate or to perform the selection at the different trigger levels has progressed in the past along a variety of paths. In the absence of a single framework with the full required functionality, several packages exist today, tailored to specific needs or goals. This does not ease the comparison of results or the maintainability of the software. Efforts have therefore been made in recent times to approach the different issues of physics and system performance in a more coherent way.

For the future LVL1, LVL2 and EF software, in order to comply with and benefit from the new OO analysis and design, a clean approach from the start is highly desirable, taking into account the specific requirements of each level. The LVL1 software simulates and reproduces the response of the LVL1 trigger and will need to follow closely the evolution of the hardware counterpart; its algorithms are relatively simple because they are implemented in fast custom hardware processors. In the case of LVL2, algorithms can be more complex because they will run on commercial processors, possibly with co-processors, but they are still constrained by the latency limits of the LVL2 trigger. At the EF, algorithms will be inherited as much as possible from the offline ones. This has strong implications on the design of the reconstruction software, because it implies a “funnel shape”[1] of the selection sequence, to take advantage of an early, fast

1. By funnel shape, we mean that the sequence is arranged to minimize the computional load in the initial selection steps, with the computationally heavy parts treated as late as possible, while retaining good signal efficiency and background-rate reduction.

rejection of unwanted events. In this spirit, since the policy of inheriting algorithms stems from the desire to simplify the transition between the two worlds, avoiding parallel development and allowing direct comparison of physics objects, the EF code should be considered as the core of the reconstruction software. The possibility of moving algorithms across the different levels (included the offline stage) should also be considered a desirable goal, depending on the availability of hardware resources and the understanding of the detector and of the selection scheme.

A final remark must be made on the quality of the software used in the online environment. In particular, for LVL2 and the EF, the potential impact on the physics performance resulting from errors or inefficiencies in the selection code is of course tremendous. Events that are wrongly rejected are lost forever and code not robust enough is bound to crash often, reducing the lifetime of the online data acquisition and hence the luminosity integrated by the experiment. Particular attention should be paid to the testing and validation procedures of the software which will run in the online farms in order to ensure the highest possible robustness, and this should be done with the help of the Quality Control group [4-7].

4.5 References

4-1 *Specification of the LVL1 / LVL2 trigger interface, version 1.0*, ATLAS internal note, ATL–DAQ–99–015 (1999)

4-2 *A prototype RoI builder for the second level trigger of ATLAS implemented in FPGAs*, ATLAS internal note, ATL–DAQ–99–016 (1999)

4-3 *Trigger and DAQ interfaces with front-end systems*, ATLAS internal note, ATL–DAQ–98–103 (1998)

4-4 *ATLAS trigger performance status report*, Chapter 4, CERN–LHCC–98-15 (1998).

4-5 *Report from the ATLAS computing review*, ATLAS project document, AT–RO–MR–0001 (1999)

4-6 *Report from the architecture task force*, `http://www.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/atf/report/report_2.7.ps` (1999)

4-7 *A policy for ATLAS software quality control*, ATLAS internal note, ATL–SOFT–99–003 (1999)

5 DAQ/EF -1

5.1 Introduction

The Data Acquisition and Event Filter Prototype '-1' (DAQ/EF -1) project [5-1] takes its name from the definition of the project to be able to support the full functionality of the final system, but not necessarily to achieve the final performance. The prototype consists of an implementation of a full *vertical* slice of the functional DAQ and EF architecture outlined in the ATLAS Technical Proposal [5-2] and detailed in Ref. [5-3]. It includes all the hardware and software elements of the data-flow, its control and monitoring, as well as all the elements of a complete online system. This project follows, and in part is based upon, previous investigations of DAQ-related issues for the LHC experiments [5-4]. Although aimed at providing the functionality described in the ATLAS Technical Proposal architecture, the prototype is designed to support the evaluation of various technological and architectural solutions. Issues related to the software development environment, from the operating systems to software engineering and CASE tools, are also addressed.

Started in December 1995, after approval by the ATLAS Collaboration, the project has been planned in three phases:

- Predesign studies and high-level design (1996–97).
- Detailed design and implementation (1997–98).
- System integration and exploitation (1998–99).

The project has been organized in a number of systems, following the features of the global architecture, as shown in Figure 5-1.

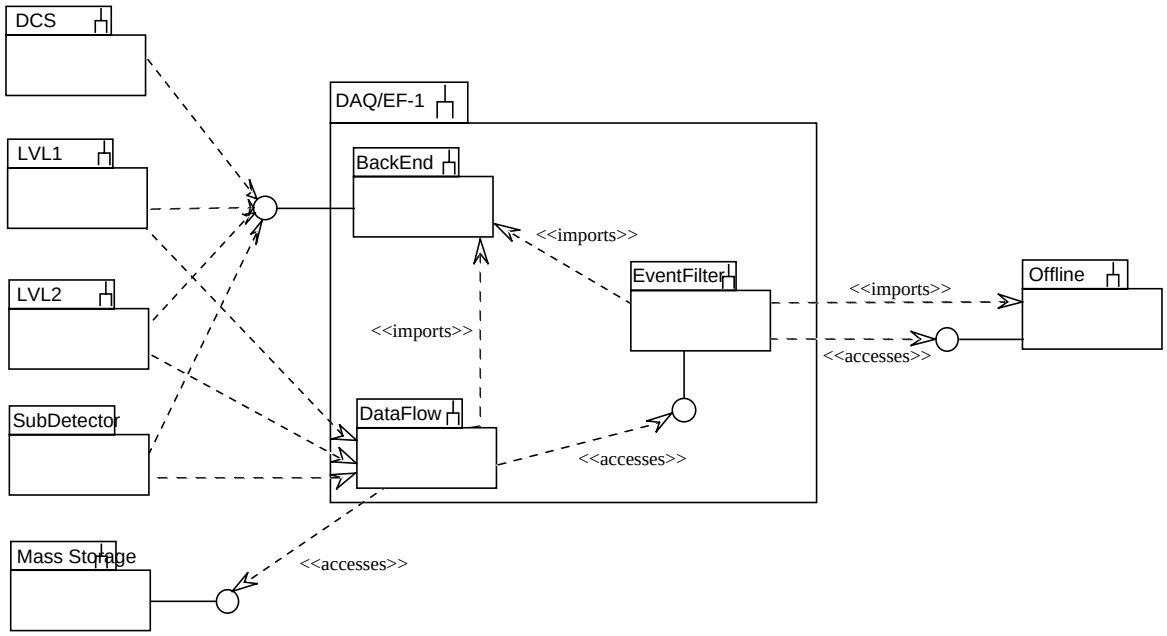

Figure 5-1 Top-level diagram of the DAQ/EF -1 architecture and its context.

The systems are:

- DataFlow
- Back-End
- Event Filter

All the connections to the experiment systems interfaced to the DAQ/EF -1 have been, for convenience, grouped in a *virtual system* of the project called

- Detector Interface

The DataFlow system is responsible for moving the event data from the detector readout links (see Section 3.3.2) to the final mass storage. It provides event data for monitoring purposes and implements local control for the various DataFlow elements.

The Back-End system encompasses the software needed to configure, control and monitor the DAQ, but excludes the processing and transportation of physics data. The Back-End software is essentially the *glue* that holds the subsystems together. It does not contain any elements that are detector specific as it will be used by all possible configurations of the DAQ and detector instrumentation. The DAQ system includes interfaces required by the triggers, processor farms, accelerator, event builder, data-flow and detector control system (DCS).

Inside DAQ/EF -1, the Event Filter system is in charge of distributing events to the different processing nodes of the SubFarms connected to the ports of the Event Builder(EB) and of providing the means to supervise (control and monitoring) the whole Farm. Algorithms to be used for the filtering operation are the concern of the PESA (Physics and Event Selection Algorithm) group. However, a strong collaboration between the two groups has been organised since the characteristics of the Event Filter Farm will be directly linked to the performances of the chosen algorithms.

The Detector Interface is not a system in itself, but rather a convenient way of grouping all the requirements and critical detector information necessary for the prototype design.

The system integration was successfully completed in June 1999, when a baseline version was made operational in the laboratory. The prototype has since been used as the basis for a number of performance measurements and operability studies which will be described later in this chapter and constitute the input for the DAQ and EF parts of the overall architecture proposed in this document, see Chapter 9. DAQ/EF -1 was selected by ATLAS to be the basis of the DAQ system on the ATLAS test beams, as well as baseline of the evolution of the ATLAS DAQ and EF.

5.2 DataFlow System

5.2.1 Introduction

The DataFlow system of the ATLAS DAQ/EF -1 [5-5] is responsible for moving the event data from the detector readout links to the final mass storage. It provides event data for monitoring purposes and implements local control for the various DataFlow elements. It interfaces with other parts of the ATLAS detector, in particular the first (LVL1) and second (LVL2) level trigger. A global view of the DataFlow is shown in Figure 5-2.

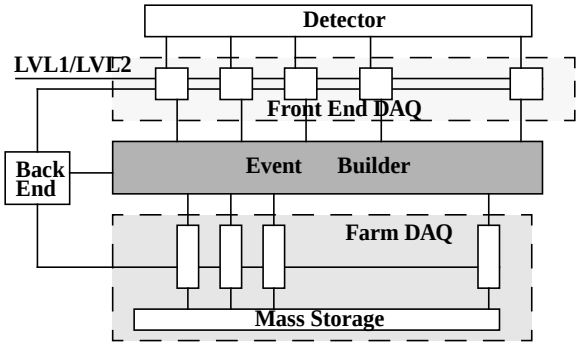

Figure 5-2 The DataFlow system.

Three main functions are provided by the DataFlow: the collection and buffering of data from the detector (the Front-End DAQ), the merging of fragments into full events [the Event Builder(EB)] and the interaction with the EF (the Farm DAQ). In addition to the Event Builder function we have identified two modular subsystems: the ReadOut Crate (ROC) and the SubFarm Crate (SFC)[1].

The ReadOut Crate: the segmentation of the detector readout into ReadOut Driver (ROD) modules suggests to organise the Front-End DAQ into a number of modular, independent elements (ROCs) each supporting the readout from one or more RODs and having one or more connections to the EB.

The SubFarm Crate: The Event Filter Farm is (logically or physically) segmented into independent modular elements, each connected to one of the EB outputs and treating one or more events. Such a modular element is called a SubFarm, its DAQ part is the SubFarm Crate.

The DataFlow is also responsible for fulfilling the DAQ partitioning requirement.[2] A DataFlow partition is a subset of the DataFlow hardware and software capable of reading-out, recording and monitoring/calibrating a subset of the detector. Given the design of the DataFlow system, and a partition granularity of a ROC, it is the role of the EB subsystem to support partitioning.

5.2.1.1 DataFlow Factorization

The functional overview of the DataFlow as described above suggests a factorization into three subsystems [5-6].

The DAQ-Unit is the component of the ROCs and SFCs responsible for receiving, buffering, merging and distributing detector event data. The ROC receives, buffers and merges ROD fragments from the detector and forwards fragments to the LVL2 system and to the EB subsystem. The SFC receives and buffers full events from the EB; sends events to and retrieves events from the Event Handler component of the Event Filter [5-6]; and sends events to mass storage.

The EB is the component of the DataFlow which merges data fragments coming from different parts of the detector into full, formatted events. It interfaces to a ROC via an Event Builder Interface (EBIF) element and to an SFC via a SubFarm Input (SFI) element.

The Local DAQ (LDAQ) subsystem provides all the common DAQ functions which are not related to moving event data. These functions are common throughout the DataFlow system: local control, initialization and configuration, support for event monitoring. In addition the LDAQ subsystem is also the interface and integration point between the DataFlow and Back-End systems. An integrated DAQ/EF -1 system will include instances of the LDAQ, spe-

1. The term *modular* here refers to the fact that a ROC (or an SFC) works concurrently and independently with respect to any other ROC (SFC) in the system. Each modular element connects independently to the EB. It is indeed the role of the EB to combine the various *modules* into a coherent DataFlow system.
2. Partitioning is defined as the capability of running multiple, independent, concurrent DAQ systems with full functionality on a subset of the detector.

cialized towards a specific DataFlow system: one LDAQ per ROC, one LDAQ per SFC and one LDAQ per partition (i.e. per EB).

5.2.2 The DAQ-Unit

5.2.2.1 Design

The design of the DAQ-Unit is given in Ref. [5-7]; only its major features are summarized here.

A DAQ-Unit is connected to one or more external I/O channels e.g. detector ReadOut Links (ROLs); all the functionality required to handle an external I/O channel is called a Task. To provide the required DAQ-Unit functionality, Tasks communicate with other Tasks and depending on the deployment, may be placed on one or more processors. These two features have led to the concept of internal I/O channels. A Task handles a single external I/O channel and one or more internal I/O channels. Communication between Tasks has been achieved by the design and deployment of a message-passing package [5-8].

Tasks are combined to form I/O Modules(IOMs) which provide the framework to schedule, configure and control their Tasks. Examples of instances of IOMs are the ReadOut Buffer (ROB), the EBIF and the Trigger Interface (TRG). Each of these instances is associated to a single external I/O channel and one or more internal I/O channels. Other IOMs combine the Tasks designed and implemented for the aforementioned IOMs in different ways, an example being a single IOM having the full ROC or SFC DAQ-Unit functionality. These different scenarios have been fully studied with respect to performance and flexibility [5-9].

An example of an interaction diagram of a ROC DAQ-Unit based on three IOMs (ROB, EBIF and TRG) is shown in Figure 5-3. A similar diagram is shown in Figure 5-4 for an SFC DAQ-Unit based on two IOMs (SFI and SFO).

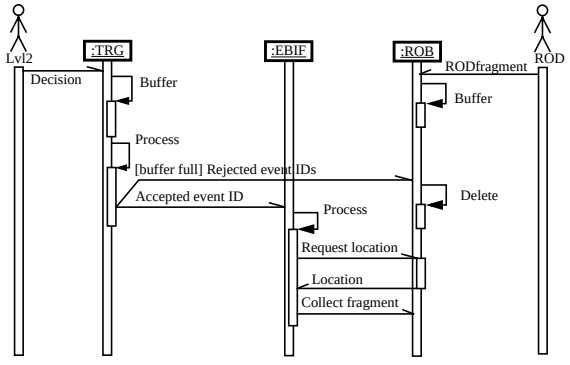

Figure 5-3 Interaction diagram of the ROC DAQ-Unit.

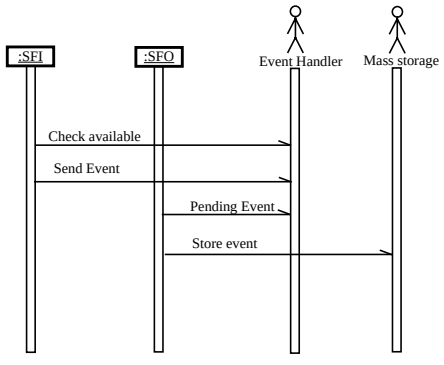

Figure 5-4 Interaction diagram of the SFC DAQ-Unit.

5.2.2.2 Implementation

The implementation of the DAQ-Unit software is based on layering. That is to say that the hardware and operating-system-independent software packages have been isolated from the hardware and operating-system-specific packages, thus supporting portability and maintenance.

Though maximum performance was not a major requirement of the design, some attention has been given to maximizing performance during implementation: only single-threaded processes

are used; interrupts are not used; drivers are not used; minimum operating-system functionality is used at run time.

The fundamental principle in the deployment has been the maximum use of (Commercial Of The Shelf) COTS products. To this end the IOMs have been implemented on CES RIO 806x VMEbus Single Board Computers (SBCs) running the LynxOS operating system [5-10] and an additional secondary bus providing broadcast functionality, specifically PVIC [5-11]. Implementations have also been performed on Motorola SBCs [5-12] and porting to Linux running on Intel platforms has started. The message passing package has been implemented on all communication technologies in the system (VMEbus, PCI bus and PVIC) and system memory.

At an early stage in the implementation it was clear that the demands on input to the ROB could not be met by the hardware described above, especially if more than a single detector ROL is connected with a ROB. Therefore the concept of a ROBIN has been used. Two developments have been undertaken based on a COTS PMC, the MFCC [5-13]; and a non-COTS PMC [5-14]. Both implementations have been used in the ROC DAQ-Unit.

5.2.2.3 Performance Measurements

A key element in the performance of the ROC and SFC is the message passing package. This has been extensively studied and reported [5-8]. The performance has also been extensively studied [5-8],[5-15],[5-16] as a function of various parameters and configuration settings, e.g. ROD fragment size, ratio of trigger messages and grouping of messages. In addition the performance of the DAQ-Unit has been understood by looking at the processing requirements of the different instances of IOMs, i.e. their Tasks [5-16]. The performance of a ROC DAQ-Unit consisting of a TRG, EBIF and one to five ROBs is shown in Figure 5-5. In Figure 5-6 the event rate as a function of the number of ROBINs per ROB and different IOM configurations, using optimal parameter settings, is shown. These results are reported in full detail in Ref. [5-9].

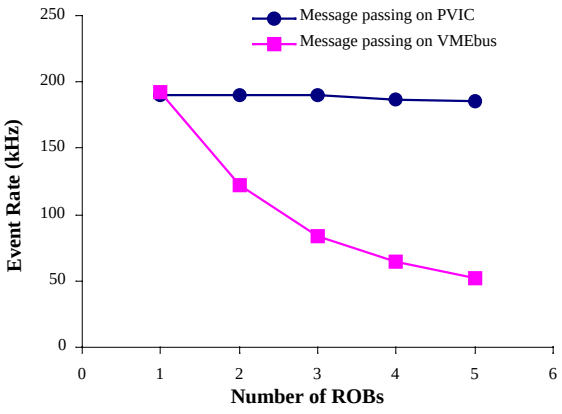

Figure 5-5 The ROC event rate with up to five ROBs.

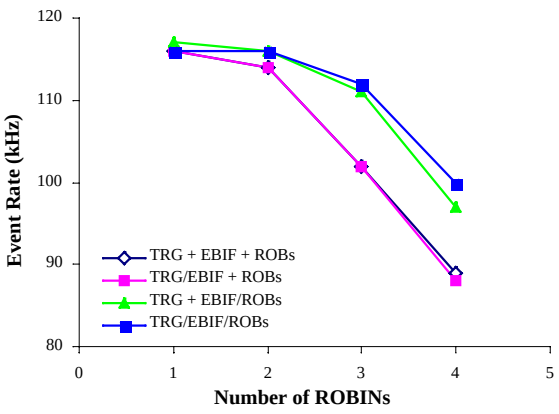

Figure 5-6 The ROC event rate for different IOM configurations vs. the number of ROBINs used.

The performance of the SFC is largely determined by the networking technology and the communication protocol used for the sending and receiving of events to and from the Event Handler. Using ILU / CORBA on a 10 Mbit / s Ethernet network and generating events of 1 kbyte, an event rate of 120 Hz has been measured. In a configuration where all the tasks building up an SFC (including a dummy event handler) were deployed on one single board computer, an event rate of 1.5 kHz has been measured.

5.2.3 Event Builder

5.2.3.1 Design

The EB high-level design [5-17] individuates five elements: the Data Flow Manager (DFM), Sources (Srcs) which are part of the EBIFs, Destinations (Dsts) which are part of the SFIs, the switching network and the Network Manager. The first three elements are interrelated via a high-level protocol, which defines the logical model in which the actual event building is performed. A schematic view of it is given in Figure 5-7. When an event is scheduled for event building its LVL1 ID[1] (L1id) is sent to the DFM, which assigns the destination in which this event will be built and broadcasts the L1id and the Dst ID to all Srcs (GetId arrow in Figure 5-7). The Srcs send the fragments and the assigned destination receives them (Transfer arrow). For each fragment an End-of-Transfer message (EoT arrow) is issued. When all fragments have arrived an End-of-Event (EoE) message is sent from the DFM to the Dst in charge with the current event. The event is then formatted and its L1id is removed from the DFM internal tables. Each Dst can send a Busy (B) message to the DFM to prevent new events being assigned to it; the assignment is reactivated when a Not-Busy ($\overline{B}$) message is sent.

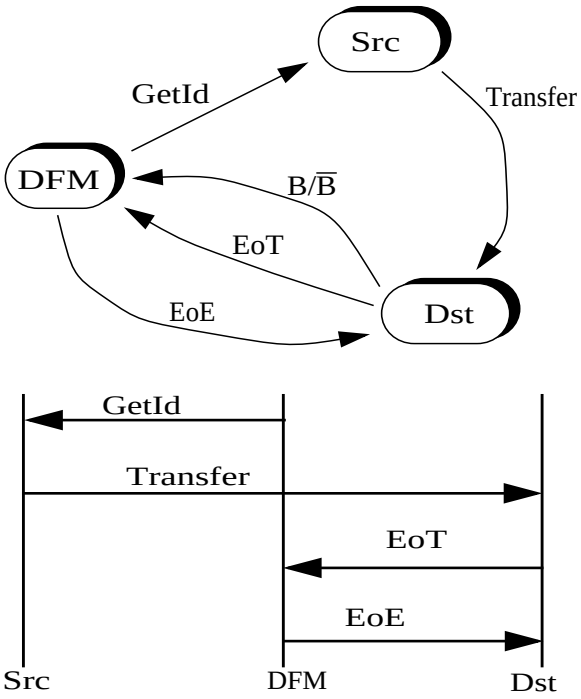

Figure 5-7 The EB high level protocol.

The EB has been designed to be physically partitionable, i.e. to allow concurrent disjunct data-taking sessions for different detectors. Furthermore, logical partitions are supported, allowing events to be assigned to different Dsts according to their event type. A single DFM will support several logical partitions, while it is foreseen to have a different DFM for each physical partition.

The EB has been structured in two layers, the first corresponding to the high-level applications and the second implementing, with a common API, the technology-specific networking aspects.

5.2.3.2 Prototype Implementation and Performance

The logical elements of the EB have been mapped onto physical network nodes in a way which ensures that, independently of the number of nodes in the system, none of the EB applications has to process data and control messages with a rate significantly higher than the EB rate.

The EB prototype implementation consists of two set-ups. One is based on RIOII VME SBCs, running the LynxOS operating system, and PCs, running Linux; this set-up runs on top of ATM and Fast-Ethernet networks. The other is a 16 node Gigabit-Ethernet set-up, based on PCs running Linux and Alteon hardware (network interface cards and switch) [5-18]. The performance of the EB prototype for different configurations (see Table 5-1) is shown in Figure 5-8.

1. In the context of the DAQ/EF -1 project, the L1id was taken as such, on the understanding that the final system will need to define a global event identifier.

Table 5-1 EB test-bed configurations.

No. of Srcs	No. of Dsts	Processors	Operating System	Network Technology	Communication Protocol
4	4	RIOII 8062 @ 200 MHz	LynxOS	ATM	AAL5
7	7	Pentium PC @ 450 MHz	Linux	Gigabit Ethernet	TCP / IP

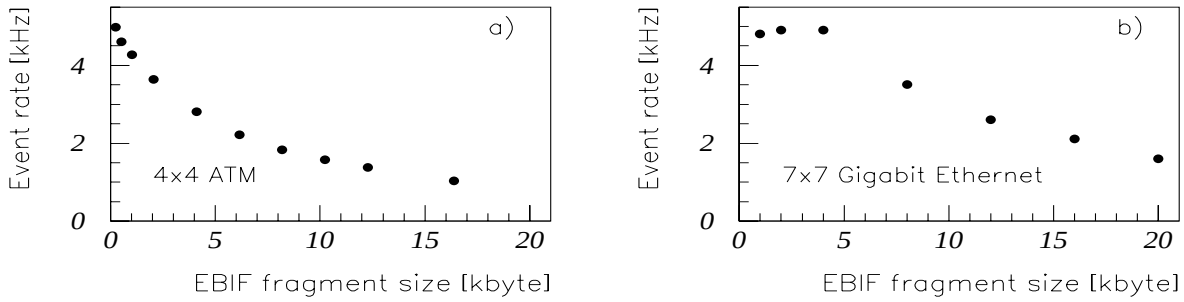

Figure 5-8 The EB performance for a) ATM 155 Mbit/s and b) Gigabit Ethernet.

5.2.3.3 EB Scalability Studies

The performance of the whole DataFlow system at the final ATLAS size is strongly dependent on the behaviour of the EB, as the number of nodes attached to the network increases. In the absence of a large scale prototype, modelling has been used to evaluate the scalability of the design and implementation of the EB.

A simulation program using Ptolemy [5-19] has been developed according to the design and implementation of the EB prototype and calibrated with the measurements carried out using the ATM 155 Mbit / s technology and 200 MHz SBCs from CES. For this purpose all the processing times in the prototype applications have been measured. The switching network has been modelled as an ideal routing element which only introduces a constant delay between input and output, justified by the fact that, for ATM, the traffic congestion avoidance can be handled at the individual nodes via Quality of Service (QoS) techniques.

The latest estimated event size of ATLAS is ~ 2.2 Mbyte [5-20]. The required EB rate is in the order of 1–2 kHz, resulting in an aggregate bandwidth of 4–5 Gbyte / s. Assuming a balanced load on all the Srcs and if one imposes that single links shall not be utilized to more than 70% of their capability, the minimal network configuration has to foresee 400 EBIFs and 400 SFIs for ATM 155 Mbit / s, or, alternatively, 100 EBIFs and 100 SFIs for ATM 622 Mbit / s. Figure 5-9 shows the simulation results for a) ATM155 and b) ATM622 using the AAL5 protocol. The extrapolated rate is strongly dependent on the assumptions made on the evolution of processing times as a function of the number of nodes.

The rate upper limit is the case in which the processing time per fragment stays constant as the EB system grows.[1] The lower limit is the case which assumes a linear dependence of the

1. There is perfect scaling behaviour and the time to build an event increases linearly with the number of fragments it is composed of. The time (in μs) to build an event as a function of the number of sources was measured (with up to 8 Srcs) and fitted with a linear function leading to $f(x) = 111x + 697$ with a χ^2 of 3.9 for 6 degrees of freedom.

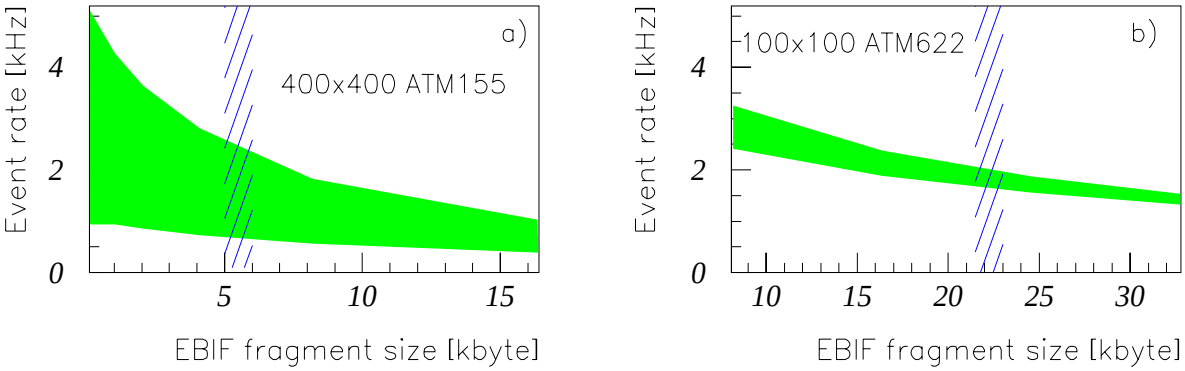

Figure 5-9 EB Simulation results for a) 400 EBIFs x 400 SFIs EB with ATM155 and b)100 EBIFs x 100 SFIs EB with ATM622. The shaded area shows the expected event rate as a function of the fragment size. The dashed area shows the region interesting for ATLAS assuming that the event data are uniformly distributed over the EBIF links.

processing time per fragment on the number of EBIFs, thus a quadratic increase of the event building time.[1] Although in the worst-case scenario the estimated event rate is not sufficient to cover the required ATLAS performance, it has to be taken into account that the processing times in the EB nodes will diminish as more powerful processors become available.

5.2.4 LDAQ

5.2.4.1 Introduction

The LDAQ is operated in two modes:

- Integrated mode

  The LDAQ is the interface point to the Back-End DAQ system for: run control, configuration databases and information exchange.

- Stand alone mode

  In the absence of the Back-End, the LDAQ provides suitable emulation of the Back-End functions, allowing a single subsystem (e.g. a ROC) to be operated.

5.2.4.2 Design

Based on an analysis of the required LDAQ functionality, the LDAQ subsystem [5-21] has the following logical elements:

- The LDAQ communication

  The LDAQ communication integrates the LDAQ with the IOMs in the ROCs, EB or SFCs. The LDAQ exchanges messages with the IOMs on a transaction basis (send / reply) which is always initiated by the LDAQ. Two different types of transactions are supported: control transactions which are typically short messages (up to a few hundred bytes) ex-

1. The time (in µs) to build an event as a function of the number of sources was measured (with up to 8 Srcs) and fitted with a quadratic polynomial leading to $f(x) = 2.2x^2 + 93x + 723$ with a χ^2 of 3.6 for 5 degrees of freedom.

changed at low rate (a few per second maximum) and transfer of event-sample data (event or crate fragments up to one Mbyte in size) on request from the LDAQ.

- The Local Control

  The local control provides run control and error handling within an ROC, EB or SubFarm. The local control is a Finite State Machine (FSM), which ensures that the controlled DataFlow elements (IOMs) are always in a coherent state. A defined set of commands for initializing, configuring, starting and stopping data-taking sessions are accepted. The execution of an LDAQ control command is associated with an action which is executed remotely on one or more IOMs. The FSM also handles asynchronous error conditions and / or notifications of state changes raised by the IOMs during data-taking sessions.

- The Event Monitoring

  Based on early discussions with detector representatives there was a requirement for statistical sampling at the ROC and SubFarm level, and no requirement for sampling all events nor for sampling correlated fragments from different sources. In addition, the effect of event sampling on the DataFlow should be deterministic. Following these requirements, the design of the LDAQ monitoring has identified two main tasks: the sampling task which collects and buffers event / crate fragments from the IOMs and the distribution task which handles user requests for analysis / monitoring purposes.

- The Configuration Data

  The configuration data is the information needed by the LDAQ to start-up the DataFlow and configure data taking sessions. This information consists of the description of the DataFlow topology (for example how the DataFlow is partitioned) as well as the DataFlow parameters (such as buffer sizes) needed for the configuration of data-taking sessions.

- The Back-End Interface

  The LDAQ interfaces with the Back-End core components (see Section 5.3.1.1): the configuration databases to initialize and configure the DataFlow, the Message Reporting System (MRS) to report errors and the Information Service (IS) to retrieve run parameters and publish run-time IOM statistics. The LDAQ run control is integrated in the hierarchy of Back-End run controllers via a TCP / IP socket interface. The LDAQ receives commands from its associated Back-End run controller and sends back a status after executing a command.

- The LDAQ GUI

  In the absence of the Back-End, the LDAQ can control a single ROC, EB or SubFarm. This is achieved by connecting a local GUI to the LDAQ. From this GUI control commands can be sent to a crate, errors can be logged and IOM statistics information can be collected and displayed on user request.

5.2.4.3 Implementation

An implementation of the LDAQ exists for VMEbus-based processors running LynxOS and Linux. The LDAQ communication, which is based on the DAQ-Unit message passing (see Section 5.2.2), is implemented on top of VMEbus and Fast Ethernet. The LDAQ control is implemented as a multithreaded application using POSIX threads. The database describing the DataFlow is based on the OKS in-memory database (see Section 5.3.2). The LDAQ monitoring is only available as a prototype and has not yet been integrated in the full system.

Most of the LDAQ measurements have been done in the context of the integrated DataFlow system (Section 5.2.5). These tests have shown that the use of message passing between the LDAQ and the IOMs, as well as the use of MRS and IS during data-taking have little effect on the global DataFlow performance.

5.2.5 Integrated DataFlow

Owing to the careful definition of interfaces between subsystems in the design phase, their integration into a full DataFlow system has been straightforward. The interface to the LDAQ is uniform throughout the system. The interface between ROCs and EB is at the EBIFs while that between EB and SFC is at the SFIs. In these elements EB tasks and DAQ-Unit tasks share the same IOM and most of the software infrastructure.

In the integrated DataFlow the LDAQ GUI is used for run-control operations and error reporting. Even in standalone mode, the Back-End configuration database is used to retrieve all the relevant configuration parameters.

The prototype has been implemented on VMEbus based SBCs running LynxOS as operating system. As an alternative, PCs running Linux have been introduced at the SFC and LDAQ level. The performance measurements which will be described in Section 5.6 have been carried out on a two ROCs by two SFCs prototype. The results presented correspond to a homogeneous SBC environment in which all processors are RIO II 8062 (200 MHz) and RIO II 8061 (100 MHz), for LDAQs only, from CES. The VMEbus is used for intra-crate communications and ATM as event building-technology. Each ROC houses two ROBs and performs local data collection to a single EBIF. The EB consists therefore of five nodes (2 EBIFs + 2 SFIs + 1 DFM). Data are generated locally in the ROBs and a trigger module distributes emulated LVL2 decisions to the two ROCs and to the DFM via PVIC. Data recording, compatible with the Central Data Recording system, has been implemented. A total of 18 processors are employed for the DataFlow elements plus an additional SBC for trigger emulation and a workstation for the LDAQ GUI.

5.2.6 DataFlow System Assessment

We assess the DAQ/EF-1 DataFlow project from three complementary points of view: the system, subsystem and final ATLAS experiment.

5.2.6.1 System view

The design and the implementation are two major assets of the DataFlow project. The design features the following.

- Modularity

  This is true at different levels in the system.The factorization of the DataFlow in terms of ROC, EB and SFC allows subsystems to be developed and exploited independently. The factorization in terms of an LDAQ and a DAQ-Unit allows the replacement of part of the functionality independent from the rest of the subsystem. The concept of the IOM and its structure, in terms of basic infrastructure and application tasks, allows, for example, collapsing of functions and the implementation of different organizations of the ROC.

- Technology independence

  The DataFlow has been designed in such a way as to isolate its functions from the evolution of technology, typically processor hardware, links and operating systems. Software critical with respect to underlying technology has been layered accordingly; application software has been designed and developed to support technology independence. Examples of the former are the EB high level-design and the message-passing library; examples of the latter are again the message-passing library and the system-independent library.

The implementation of the DataFlow features a baseline system, based on VME modules and a number of options, including the possibility of using PCs running Linux in parts of the system or a secondary bus in the ROC.

We believe that the DataFlow design is a good basis for the next phase of the development of the ATLAS HLT/DAQ system. The design has proven to be very flexible during the three years of the DAQ/EF-1 project. It has allowed the evolution of views and different concepts to be tested; and technology trends to be followed (i.e. extension of implementations to Linux and Intel platforms). This is an important consideration given the long period of time before the final decisions. Flexibility has also been maintained in the implementation. This has been achieved by software layering and the clear separation between hardware and operating system dependent and independent software packages. Another important point about the design is that it has led to a fully functional implementation.

The DataFlow implementation will be used as the basis for a test-beam DAQ system, while parts of it are very good candidates as the basis of future prototypes.

5.2.6.2 Subsystem view

- *DAQ-Unit*

  The cornerstones of the DAQ-Unit design are Tasks, IOMs and message passing between Tasks. Extensive studies and measurements have shown that this flexibility has not led to performance penalties. The use of COTS products has decreased the resources required during the design and deployment. Based on today's technologies and the potential requirement of more than a single ROL per ROB, the concept of a ROBIN has been shown to be important if the ROB input requirements are to be met. Open questions remain with respect to the influence of event monitoring, the integration of a LVL2 interface and the use of blocking Tasks in the current implementation. A ROC and SFC based on the DAQ/EF-1 design and the deployment of COTS products, capable of meeting the ATLAS requirements, is within the grasp of the ATLAS HLT/DAQ community.

- *Event Builder*

  The EB elements and protocol have been designed and prototyped. The layered approach taken in the software structuring allows the same applications to be run on different technologies. The present implementation of the EB protocol has been proven to be scalable. The simulation also shows that particular effort has to be put in the software engineering to eliminate any strong dependence of the processes' execution time on the number of nodes in the system.

- *LDAQ*

  The LDAQ design which separates the LDAQ functionality from the DAQ-Unit has allowed implementations on VMEbus SBCs as well as on Unix PC/workstations. Another important feature of the LDAQ is the possibility to operate parts of the DataFlow (e.g. a

single ROC) in isolation for testing and debugging purposes. The LDAQ has been shown to be a resource-consuming task in terms of CPU, memory and network, thus justifying the concept of separating it from the DAQ-Unit.

5.2.6.3 ATLAS HLT/DAQ view

The main objective of the DAQ/EF-1 project is the study of a functional HLT/DAQ architecture for the ATLAS experiment. It does not have the ambition to completely fulfil the ATLAS requirements in terms of performance. Nevertheless the DataFlow project has produced an implementation and detailed performance studies have been carried out. The implementation has demonstrated that a ROC using ROBINs and PVIC as a secondary bus shows strong indications that the 100 kHz rate required by the ATLAS experiment is in reach. Therefore, we believe that a ROC based on the DAQ/EF-1 design can be the basis to build a system that will meet the ATLAS required performance. The same design and implementation can also support the LVL2 interface. As regards the EB subsystem, the scalability studies indicate that the protocol scales with the size of the switch.

5.3 Back-End DAQ System

5.3.1 The Software Component Model

The user requirements gathered for the Back-End subsystem have been divided into groups related to activities providing similar functionality. These groups have been further divided into components of the Back-End with a well-defined purpose and boundaries. The components have interfaces with other components and external systems, specific functionality and their own architecture.

From analysis of the components, it was shown that several domains recur across all the components, including data storage, interobject communication and GUIs.

5.3.1.1 Core Components

The following components are considered to be the core of the Back-End subsystem. They constitute the essential functionality of the Back-End subsystem and have been given priority in terms of time-scale for development in order to have a baseline subsystem that can be used for integration tests with the DataFlow subsystem and EF.

- Run control

  The run-control component controls the data-taking activities by coordinating the operations of the DAQ subsystems, Back-End software components and external systems. It has user interfaces for the shift operators to control and supervise the data-taking session and software interfaces with the DAQ subsystems and other Back-End software components. Through these interfaces the run control can exchange commands, status and information used to control the DAQ activities.

- Configuration database

  A data acquisition system needs a large number of parameters to describe its system architecture, hardware and software components, running modes and the system running

status. One of the major design issues of ATLAS DAQ is to be as flexible as possible, parameterised by the contents of databases.

- Message Reporting System

  The aim of the Message Reporting System (MRS) is to provide a facility which allows all software components in the ATLAS DAQ system and related subsystems to report error messages to other components of the distributed DAQ system. The MRS performs the transport, filtering and routing of messages. It provides a facility for users to define unique error messages which will be used in the application programs.

- Process Manager

  The purpose of the Process Manager (PMG) is to perform basic job control of software components of the DAQ. It is capable of starting, stopping and monitoring the basic status (e.g. running or exited) of software components on the DAQ workstations and LDAQ processors independent of the underlying operating system. In this component the terms process and job are considered equivalent.

- Information service

  The Information Service (IS) provides an information exchange facility for software components of the DAQ. Information (defined by the supplier) from many sources can be categorized and made available to requesting applications asynchronously or on demand.

5.3.1.2 Trigger / DAQ and Detector Integration Components

Given that the core components described above exist, the following components are required to integrate the Back-End with other online subsystems and detectors.

- Partition and Resource Manager

  The DAQ contains many resources (both hardware and software) which cannot be shared, so their usage must be controlled to avoid conflicts. The purpose of the Resource Manager is to formalize the allocation of DAQ resources and allow groups to work in parallel without interference. The Partition Manager is intended to extend this functionality to whole partitions.

- Integrated Graphical User Interface

  The Integrated Graphical User Interface (IGUI) allows the operator to control and monitor the status of the current data-taking run in terms of its main run parameters, detector configuration, trigger rate, buffer occupancy and state of the subsystems.

- Online Book-Keeper

  The online book-keeper archives information about the data recorded to permanent storage by the DAQ system. It records the information to be later used during data analysis on a per-run basis and provides a number of interfaces for retrieving and updating the information.

- Event Dump

  The event dump is a monitoring program with a GUI that samples events from the data-flow and presents them to the user in order to verify event integrity and structure.

- Test Manager

  The purpose of the Test Manager (TMGR) is to organize individual tests for hardware and software components. The individual tests themselves are not the responsibility of the TMGR, which simply ensures their execution and verifies their output. The individual tests are intended to verify the functionality of a given component. They will not be used to modify the state of a component or to retrieve status information. Tests are not optimized for speed or use of resources and are not a suitable basis for other components such as monitoring or status display.

- Diagnostics Package

  The diagnostics package uses a knowledge base and the tests held in the TMGR to diagnose problems with the DAQ and verify its functioning status. By grouping tests into logical sequences, the diagnostic framework can examine any single component of the system (hardware or software) at different levels of detail in order to determine as accurately as possible the functional state of components or the entire system. In case any faults are found, it reports the diagnosis and suggests actions necessary to restore the affected component.

5.3.2 Underlying Technologies

The various components described above all require a mixture of facilities for data storage, interprocess communication, GUI, complex logic-handling and general operating-system services. Candidate freeware and commercial software packages were evaluated to find the most suitable products for each technology. C++ with a general-purpose library, called Tools.h++, is the primary programming language. The Objectivity / DB object-oriented database and custom-made in-memory persistent object manager (OKS) are used for data persistence.[1] CORBA (ILU) and a custom-made package on top of it (IPC) are used for communication. Finitestate-machines are used for implementing object behaviour (CHSM). The CLIPS expert-system tool is used as the basis for a diagnostic system. Motif and Java are used for GUIs.

5.3.3 Integrated Back-End DAQ

5.3.3.1 Component Tests

For all core components an implementation exists and functionality and performance tests have been performed. For most detector integration components an implementation is available and testing has started and is expected to be completed soon. Each component is subjected to unit tests to assess its functionality, performance, scalability and reliability. For each component a test plan has been prepared and the test results have been reported [5-22]. The integration of the core components was made in a step-wise manner, according to the dependence between components and the underlying external packages (both commercial and freeware) see Figure 5-10. Integration and scalability tests are based on the two most relevant scenarios for an integrated DAQ system representing the likely configurations for the DAQ / EF -1 project and the final ATLAS DAQ / EF system.

1. The Back-End DAQ, with the exception of the online book-keeper component, can run without Objectivity / DB by relying solely on OKS.

For further information on the results of the unit tests for Back-End components see Ref. [5-23].

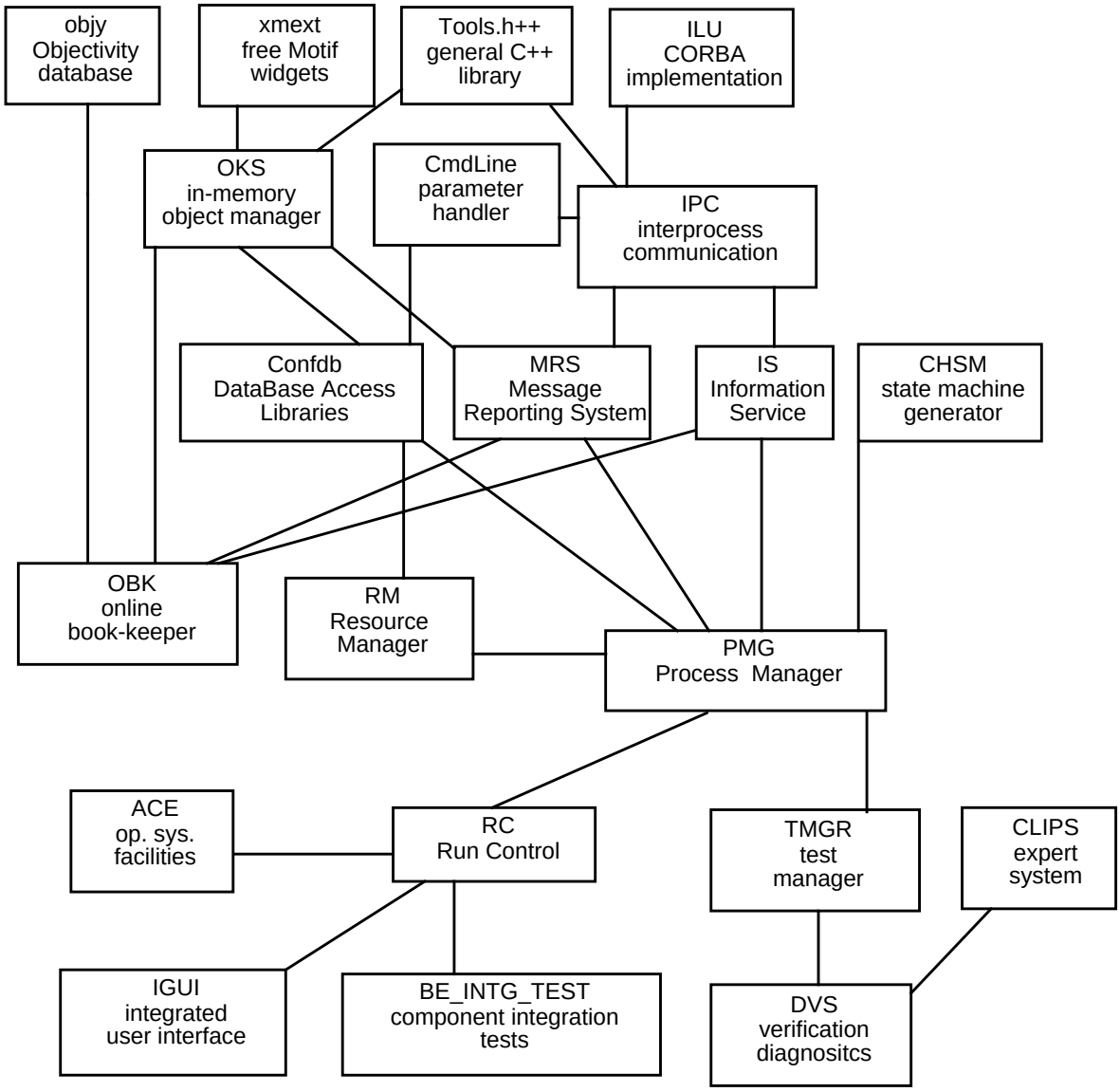

Figure 5-10 Back-End DAQ software package dependencies.

5.3.3.2 Integration Tests

Following individual component unit tests, integrated tests have been performed by employing the majority of the components developed in the Back-End DAQ. Such integration tests have been performed with the goal of verifying the correct interoperation of the components, the ability to operate in a distributed, heterogeneous, multi-platform environment and to gather performance measurements relevant to the operation of the DAQ in a production environment. The test configurations have included workstations, PCs and VME-based processors. The operating systems involved were Solaris, LynxOS and Linux. Configurations using up to 21 processors have been tested. The tests have been made using a shell script that goes through the initialization, operation and shutdown phases. For all these tests the Back-End communication server processes, DAQ supervisor and Run-Control (RC) root controller were always started on the same workstation (i.e. the host on which the benchmark script was launched). The other

processes (PMG agent, local DAQ emulator and corresponding RC controller) were started on separate PCs running Linux, or VME-based PowerPC CPU boards running LynxOS. The computers were not dedicated to the benchmark and they were used by other developers at the same time.

The tests were performed several times for each configuration and the average values for each test were calculated. The time taken for the tests strongly depends on the load of the computers and may vary by a few tens of per cent for lengthy operations (e.g. set-up and stop).

The PMG agents are started during the set-up stage via a remote shell that requires long delays (20 seconds) for synchronization purposes. In test beam and production use the agents will be started during the boot of the machines.

The results of these tests are presented below in Figures 5-11 and 5-12 for different configurations and types of operations.[1] For a discussion of the results obtained see Section 5.6.

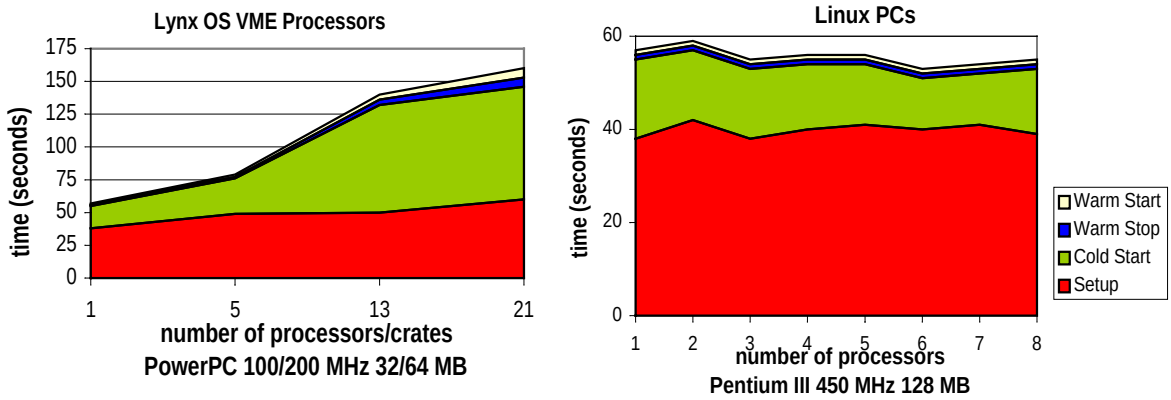

Figure 5-11 Test results for start-up and warm stop/start operations.

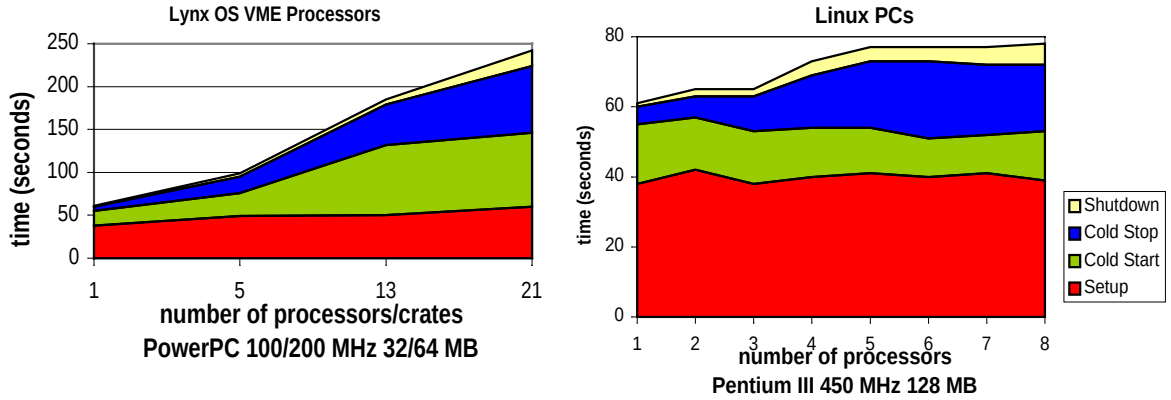

Figure 5-12 Test results for start-up and close operations.

1. **Start-up** operation means *set-up* and *cold start*. **Close** operation means *cold stop* and *shutdown*.

5.3.4 Software Process and Inspection

The development has been divided into a number of sequential phases intended to help pace and organize the work. Each phase has been defined to produce an obvious deliverable (i.e. document and/or code), which is reviewed before progressing to the next phase. The phases are: collect requirements; identify and evaluate candidate technologies and techniques capable of addressing the common issues identified from the requirements; produce a design for each component covering the most important aspects; refine the design to add more detail; implement and unit test according to the design; integrate with other components. Figure 5-13 shows

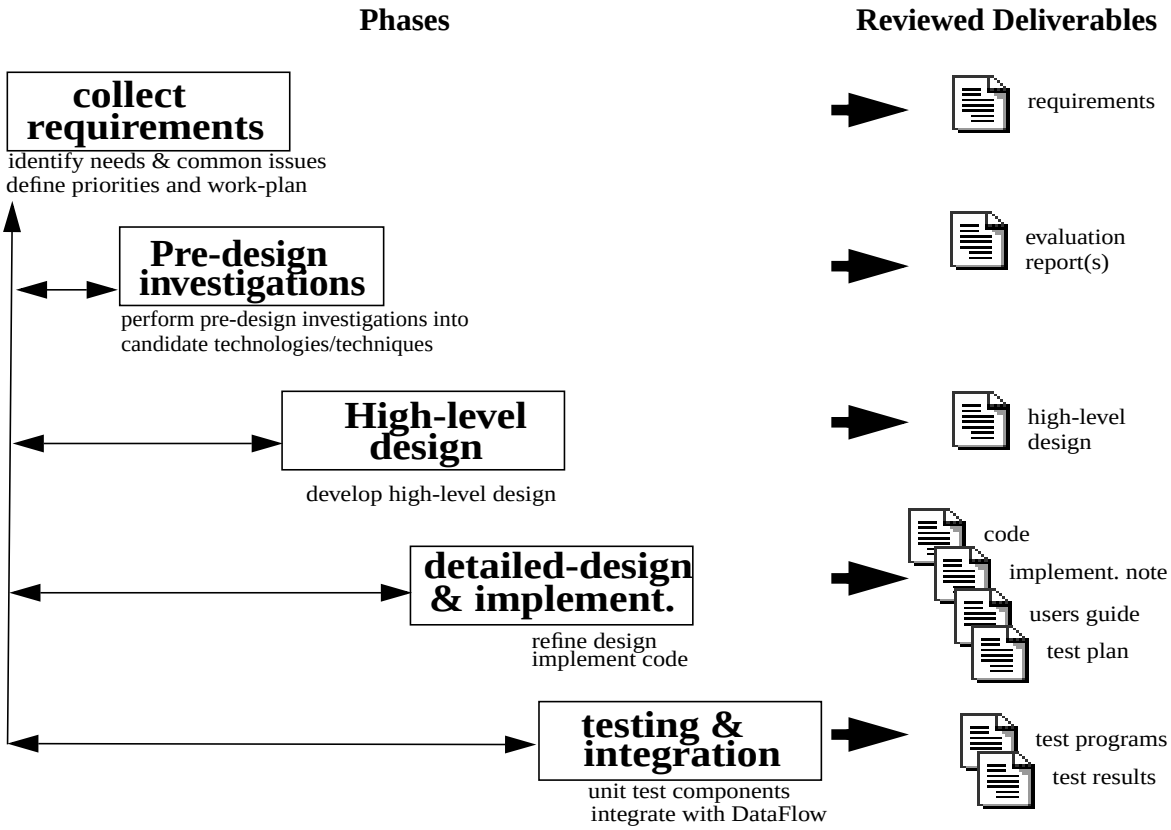

Figure 5-13 Software process employed for the development of the Back-End DAQ.

the phases of this software process and the artefacts produced by each phase, including documents used during development, code and user documentation. The deliverables from each phase have been reviewed. Initially, reviews took the form of presentations followed by discussions during an open meeting with all developers involved in the project. Later more formal inspections were introduced using a system of peer review supported by guidelines and checklists for documentation and code. In total 8000 lines of code which is about 20% of the in-house written code of the Back-End software, and 180 pages of documentation were inspected. Nineteen inspectors and one inspection leader participated in the inspections. For more information on the use of reviews and inspections in the Back-End DAQ refer to Ref. [5-24].

The people involved in the Back-End DAQ come from many institutes and have not, in general, been able to work full-time on the project. Faced with this situation, we have tried to organize the work along the component structure. Typically, a single institute has taken responsibility for developing a component, thereby simplifying communication and reducing travel. Such com-

ponent groups are small (up to a maximum of five individuals). The same individuals have tended to follow a single component through the various phases and hence ensured the continuity of the work.

5.4 Event Filter System

There are several possible candidate hardware architectures for an EF computing farm (EFF) capable of treating an input data rate of ~ 1 Gbyte / s and achieving a factor of ten reduction of this rate by applying complex physics selections to the events. The required processing power was estimated to be at least 25 kSPECint95, or 1 TIPS. Any final hardware choice must clearly be left until the last possible moment compatible with building and commissioning the EFF in time. It was also noted that given the estimated running lifetime of more than 15 years, the Farm must be both easily and gradually upgradable. This indicates the possibility that a variety of hardware and even operating systems may be running in parallel during the lifetime of the EFF.

5.4.1 Prototypes

Because the building of the event is performed by a switch where the different sources are connected to independent destinations, it is natural to factorize the EFF into independent SubFarms, each of them connected to a different output port of the EB as well as to the mass storage (Figure 5-14).

In addition, this architecture provides modularity, easy reconfiguration capabilities and straightforward redundancy properties. The required number of SubFarms will be determined from the available technology and the total CPU power needed.

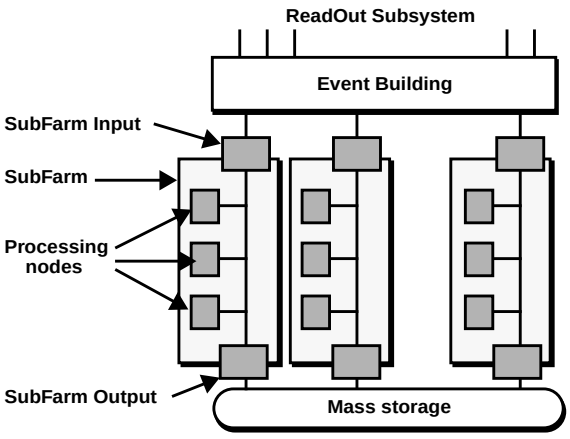

Figure 5-14 Overview of the Event Filter Farm.

Each SubFarm can contain one or several processors linked together. The EF group has produced a high-level design of the EFF software architecture [5-25] which is deliberately independent of any specific hardware or OS features. The data flow has been factorized in three main logical components: a Distributor, several Processing Tasks and a Collector, themselves built from four elements providing core functionality, I / O capabilities and control. Some parts of these logical components can possibly be collapsed depending on the hardware conditions. Events are pulled by the downstream components so that a natural load balancing is obtained. Configurations including machines of very heterogeneous computing power are easily handled by this mechanism. This design has been implemented in C++ in a modular fashion, and giving specific attention to insulating any hardware or OS specific areas, thereby leaving the body of the code invariant under hardware or OS changes.

According to this common design, three EFF prototypes have been implemented and assembled based on three somewhat different configurations: Commodity PCs, Commercial Symmetric Multi-Processor (SMP) and Intel Commodity Multi-Processors.

5.4.2 Commodity PC Prototype

This prototype aims to exploit the ever-decreasing price/performance ratio of the commodity component computers. Communication should not be a bottleneck in an EF SubFarm since applications are largely CPU limited. The distributed nature of the EF complies rather well with the hardware architecture of PCs, facilitating scalability and hardware upgrades. However, management of very large distributed configurations will be a complex task and will necessitate specific developments. Also hardware reliability is likely to be weaker than for a fully commercial solution.

This prototype has been tested on three different test beds based on bi-processor Pentium II machines (up to 35 nodes) interconnected with Fast Ethernet, and with a Gigabit Ethernet up-link to an external workstation used to inject events in the Farm.

The prototype has been firstly implemented with Windows NT as operating system. It uses distributed computing techniques. The various SubFarm components are implemented by different processes running on different nodes. Transport protocols for the communication among SubFarm components based on CORBA (ILU) and on TCP protocols have been studied. An implementation with the Linux operating system and TCP/IP is now available. It has been successfully tested on a configuration involving a thousand processing tasks.

This prototype has been used as a test bed for the development of the data-flow code. Different versions of the code have been released. The present one (*Version 3*) separates clearly the transport protocol from the higher level layers, therefore providing a very portable implementation of the generic design. The supervision of the prototype makes large use of the capabilities of the Java mobile agents technique and is described below.

5.4.3 Symmetric Multi-Processor Prototype

Since the processing power of each SubFarm is supplied by several processors working in parallel, a complete SubFarm can be implemented on a single SMP machine. The SMP architecture offers advantages in data sharing and transfer between the different hardware (and software) components: The main memory and other system resources can be accessed symmetrically by all the processors through a very high speed system interconnect (system bus, crossbar switch, etc.).

In order to avoid as much as possible interference of critical operating system aspects in the SubFarm code implementation itself (and to obtain a better reliability of both hardware and software component) the prototype has been developed on a commercial SMP architecture with a proprietary operating system, but has also been ported on an SMP commodity PC running Linux.

To achieve a better exploitation of the hardware resources all the components of the SubFarm have been implemented within a single subthread process. Every SubFarm component is assigned a thread scheduled directly by the OS kernel. The choice of the multi-threaded implementation stems from the fact that it eases in particular the communication and the synchronization among the different components: the event data-flow through the SubFarm can be achieved by passing to the different components the pointers to the events stored in the process memory space.

This SubFarm implementation has been tested on three different HP servers (4 CPU K220, 8 CPU N4000 and 20 CPU V2500) running HP-UX 11 and on a 4 CPU PC Intel-based (COMPAQ ProLiant 5500) running Linux.

5.4.4 Intel Commodity Multiprocessor Prototype

The main goal of the THOR project is to provide a commodity component prototype for the EFF that may be used to study various algorithms and implementations of the data flow and analysis. THOR aims to provide a 1/10 size (128 processor) prototype of the global ATLAS EFF in its final version which is planned for the end of the year 2000. The current THOR prototype consists of twenty dual PII/III 450 MHz machines connected with Fast Ethernet. In addition, nine of the nodes are also connected using Scalable Coherent Interconnect (SCI).

The design of the prototype is based on a farm of independent processors and thus many processes are running on the different machines in each SubFarm. The operating system chosen is Linux, mainly because of the ready availability of the system and the fact that upgrades and bug fixes are easily found in the public domain. The interprocess communication transport is based on raw TCP sockets. TCP/IP is a well-established, well-tested protocol and it offers excellent guarantees of reliability. Moreover, some specific features have been implemented on top of TCP/IP to improve message routing between the different components.

5.4.5 Results and Comparison of the Prototypes

5.4.5.1 Data Redundancy and Robustness

The prototypes implement a form of the Distributor Global Buffer [5-25]. In all implementations currently this has the form of a disk partition, where events are stored during their passage through the SubFarm. This is important in order to recover from a crash of the processing task or the Distributor, or even of the SubFarm itself. As noted from the measurements in all the prototypes, the time spent in disk writing is relevant only if the processing task time is shorter or comparable to it. With the prospected speed of disks, or disk-arrays, compared to the ATLAS reconstruction and analysis time, one could say that this item should not be a limitation to the performance of the SubFarm. On the other hand, since an error-free SubFarm is clearly not achievable, a better estimate of the tolerable, unbiased data loss rate for the ATLAS detector should be given in the future.

5.4.5.2 Data Communication Mechanisms

Several protocols have been studied in the prototypes to send data to the various processing tasks on different processors. In the distributed implementations (PC based) this is achieved via either CORBA or TCP protocols. The TCP solution is lighter than the CORBA/ILU one, and general enough for the needs of a SubFarm. In this context, the least demanding solution in term of resources is probably the one adopted for the SMP SubFarm, where the computer memory is used to hold the events and hence pointers, passing event addresses to the processing threads, are the only communication mechanism used. Given the ever-decreasing RAM costs, the size of the required memory should not constitute a problem for reasonably sized SubFarms, even though the hardware of the SubFarm will be intrinsically more costly. An interesting alternative solution is represented by the SCI link between processors, currently studied in the THOR prototype, which builds a SubFarm of the cc-NUMA type, using commodity com-

ponents. One might note that, in all these examples (with respect to CORBA / ILU), some more work is needed to accomplish the correct identification of objects which need to be known in the outside world (e.g. for Back-End software purposes).

5.4.5.3 Throughput and Scalability

The output of the ATLAS EB, after LVL2 selection, is currently estimated at about 1–2 kHz. From this number, together with guesses related to the processing time needed for each event, figures have been estimated, related to the size and the granularity of the EF subsystem: The final system could be made of a hundred of SubFarms, each containing ten-odd processors rated at 25 SPECint95. Assuming that the processing time of each event is on average 1 s, the required throughput for each SubFarm would be of the order of 10 events per second.

From the measurements obtained by the three prototypes, one could easily imply that no prob-

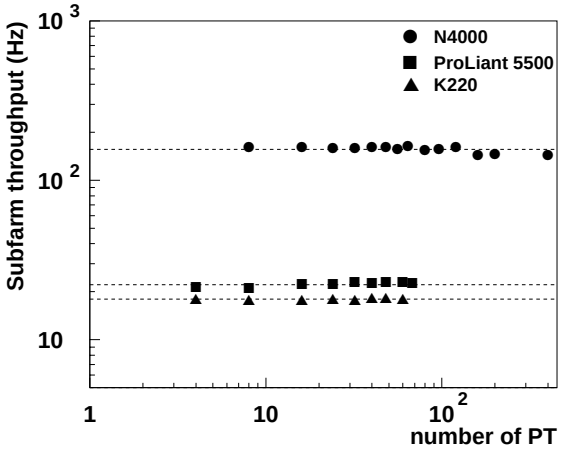

Figure 5-15 Throughput of SMP prototypes vs. number of processing tasks.

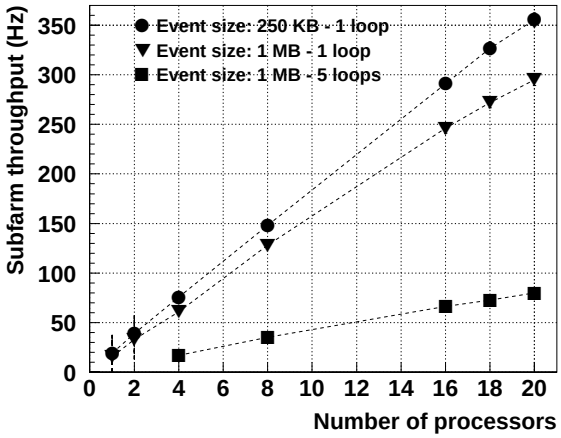

Figure 5-16 Scalability of the SMP prototype on HP V-class 20 CPUs (1 loop = 50 ms).

lem is seen on the horizon, concerning the fulfilment of the requirements imposed by the ATLAS DAQ system. In fact, all SubFarms are currently achieving, or are close to, throughputs aligned with the required ATLAS numbers. Moreover, complex scenarios arise from the mixture of physics and non-physics data (e.g. calibration data) flowing through the SubFarm. In this case, the current understanding is that load balancing will be a key issue, if one wants to merge those data types in the same SubFarm. In Figure 5-15 the throughput of different SMP prototypes is shown: Note that the global behaviour is independent of the number of processing tasks running on the SubFarm.

The following scalability studies have been made. In the SMP case, where the use of kernel threads implies relying heavily on the POSIX operating-system structure, the prototype has been shown to scale linearly up to 20 processors (see Figure 5-16) and hundreds of processing tasks (up to 400 tasks have been used on the N4000 HP machine without notable degradation of the performance).

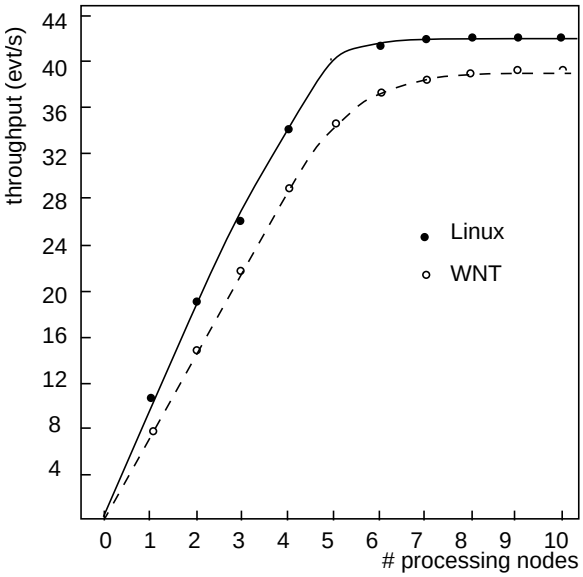

Figure 5-17 Scalability of the distributed architecture.

In the distributed architectures (Figure 5-17), assuming the average event size of 1 Mbyte, the use of new generation commercial networking should be capable of providing the necessary performance. Even in the case of the more realistic size of 2 Mbyte now commonly envisaged, the advent of Gigabit Ethernet switchs should provide easily the 20 Mbyte/s throughput. A particular role is played by the THOR prototype, where the scalability studies are extending beyond the scope of DAQ/EF -1, addressing the problem of multiple SubFarms and their partitioning.

5.4.5.4 SubFarm Configuration and Monitoring

Configuring the SubFarm and keeping track of its behaviour is an essential ingredient which all the prototypes have considered in their studies. In the commodity PC implementation a lot of progress has been made on the SubFarm supervisor element, based on Java mobile agents (Voyager). This is particularly relevant to demonstrate the feasibility of a complex distributed architecture, where bottlenecks might arise from the inability to communicate with and keep control of thousands of processors. The implementation scheme is shown in Figure 5-18. There are three levels of monitoring: system supervision (at the machine level, possibly provided by the operating system), Mobile Agent for event dataflow control and monitoring, offline analysis tools for monitoring data archived in the database.

The achieved results are extremely positive and are of course beneficial also to other implementations: The tests show that the supervision has been able to control 1023 distributed components without any problem. The monitoring aspect has been studied in the same context, using a graphical interface to the SubFarm performance. The interface has been extended across many SubFarms for THOR. In the SMP case, internal monitoring is provided through proprietary tools which control the behaviour of the processing tasks. For general supervision a tool similar to the one used for the commodity PC prototype could be adopted.

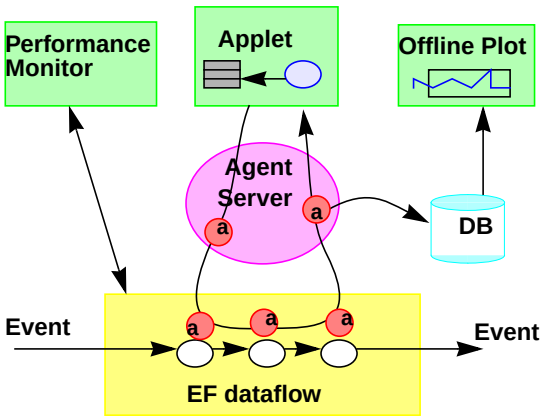

Figure 5-18 Implementation scheme of the supervision.

5.4.6 Event Filter System Assessment

The concept of independent SubFarms for the EF has been shown to be desirable, because it naturally solves problems related to farm partitioning and, more generally, flexibility and scalability. The clean definition of an API connecting the EFF to the DataFlow has been of extreme help in integrating different hardware and software implementations of EF SubFarms. No problem has been met in running EF SubFarms geographically separated from their DAQ data source.

Different data communication mechanisms have been studied, showing that many types of EF SubFarm are possible, from widely distributed configurations to large SMP ones. The importance of including data security features, such as the Distributor Global Buffer (DGB) for facilitating error recovery in the EF has been shown. The configuration and monitoring of the EF system has been shown to be a key issue for the maintainability of large Farms and solutions (e.g. based on Java mobile agents) have been proposed.

The values of global throughput and the performance scalability (both in terms of processing elements and components) measured in all the prototypes are already close to the ATLAS requirements.

5.5 Detector Interface System

The Detector Interface is not a system in a classical sense. In the context of the DAQ/EF-1 overall system, it can be seen as a *virtual subsystem* encompassing the set of requirements and critical detector information necessary for the prototype design. The collection of such requirements and detector parameters and the understanding of the impact of relevant detector features on the DAQ/EF-1 design was done via a Detector Interface Group (DIG).

Originally set up in the context of and as an integral part of the DAQ/EF-1 project, the DIG soon grew to assume the role of a very useful forum between the DAQ/EF-1 team and the detector and other ATLAS system communities to discuss and analyse common DAQ issues. Areas such as run control, database access, system partitioning, monitoring, calibration and local data acquisition were addressed by this requirement collection.

Two major requirements of the detectors are for a coherent test-beam data acquisition suite, and for some common (cross-detector) DAQ functionality at the level of the ROD crates. These issues have been latterly addressed in the DIG. The result has been a fairly clear definition of common requirements and associated time-scales for these areas. A more detailed description of the DIG's current work can be found in Chapter 3.

One of the principal stages of the DAQ/EF-1 project was a phase of testing in the test-beam environment, integrated with one or more detector ROD systems. Now that DAQ/EF-1 has been selected by ATLAS to be the basis for future data acquisition in the test beam, planning for this is well advanced, and detailed discussions with the detector groups are under way.

The DIG has proved to be an extremely important element in the development of the DAQ/EF-1 prototype. Viewing detectors as systems which have to be *serviced* by the DAQ has been very valuable.

5.6 Global System Performance

5.6.1 Introduction

Tests of the fully integrated DAQ/EF-1 prototype were carried out to understand the global performance and its scalability aspects. The tests have been designed with the intention of better understanding how to use the integrated prototype and to optimize configuration parameters of stability and performance.

5.6.2 Configuration and Environment

The fully integrated DAQ/EF-1 prototype brings together the Integrated DataFlow system (Section 5.2), the integrated Back-End system (Section 5.3) and the Event Filter system (Section 5.4). The external Trigger system is emulated and event data are generated internally by the ROCs as no detector was ready to be integrated. The baseline prototype consists of two ROCs, two SFCs, ATM as Event-Building technology and was controlled by the Back-End. This system will later be referred to as the 2×2 *baseline system*.

The PCs and workstations used for these tests were not entirely dedicated to the benchmarks. Workstations were used by developers while the tests were ongoing; in addition no private Ethernet segment was used. The VMEbus SBCs were NFS diskless systems running LynxOS, served by a Linux PC. When applicable the tests were performed several times for each configuration and the average values for each test were calculated.

5.6.3 Tests and Results

The tests carried out on the fully integrated DAQ/EF-1 prototype covered mainly the functionality, performance and scalability aspects of the system.

5.6.3.1 Functionality

While putting together the integrated DAQ/EF-1 prototype, the typical DAQ functionality has been checked. This included starting and stopping DAQ sessions, changing run parameters, recording data to disk, checking data integrity of full events from an Event-Handler task and displaying statistics about front-end IOMs. Several configurations have been tried (1 ROC by 1 SFC, 2 ROCs by 1 SFC and 2 ROCs by 2 SFCs using both ATM and Fast Ethernet as Event-Building technology), with each configuration mapped to a single partition. The 2×2 baseline system ran continuously for two days at a low trigger rate until completion.

5.6.3.2 Performance

Tests on the performance of the integrated DAQ/EF-1 prototype have two aspects: those related to the control and configuration of data-taking sessions and those related to the flow of physics data. The former is directly associated with the Back-End system while the latter is directly associated with the DataFlow system. The following performance tests were run on the 2×2 baseline system.

- The interesting performance figure related to the control and configuration of the DAQ/EF-1 prototype is the elapsed time taken to perform a number of data-taking oper-

ations (cold start, warm start, warm stop, etc.). The corresponding series of tests is started by a shell script which takes the whole system through a series of states as shown in Figure 5-19:

a. Start the Back-End server processes to initialize communication services and the Process Manager (PMG).

b. Launch configuration specific processes via the DAQ supervisor as described in the database.

c. Marshal the hierarchy of run controllers through different states (initialized → loaded → configured → running → configured → running → configured → loaded → initialized).

d. Stop configuration specific processes via the DAQ supervisor.

e. Stop the Back-End server processes.

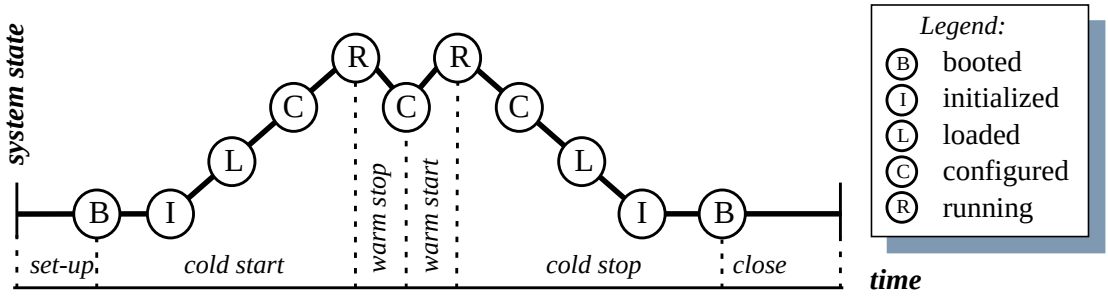

Figure 5-19 Activity diagram showing the actions performed by the benchmark script.

The results for the 2×2 baseline system (control on Solaris, LDAQs on VMEbus SBC / LynxOS) are:

a. The *set-up* phase, which only involves the Back-End, takes 50 s.

b. The *cold start* phase, which involves the Back-End and the DataFlow, takes a bit more than 1 min (25 s for the Back-End part and 40 s for the DataFlow part).

c. The *cold stop* phase, which involves the Back-End and the DataFlow, takes roughly 2 min (mostly Back-End, 6 s for the DataFlow part).

d. The *close* phase, which only involves the Back-End, takes almost 1 min.

e. The *warm start* and *warm stop* phases, which involve the Back-End and the DataFlow, are very fast operations which take less than 1 s.

f. During a run, reporting errors using MRS and publishing statistics using IS has a negligible effect on the system performance.

One can observe that operations which require interaction with the operating system (Solaris, LynxOS) for process creation and with the network for remote starting applications via NFS take most of the time. Further testing is required to determine whether the environment (controlling workstation load, concurrent NFS access to a single server) plays an important role and to understand why stopping processes generally takes much longer than starting processes.

- The interesting performance figure for the global DataFlow system is the rate at which events can be pushed through the whole DAQ chain. This is required to be in the order of 1–2 kHz, the expected LVL2 trigger accept rate. The DataFlow performance tests have been run on the 2×2 baseline system with SFCs which have a collapsed SFI/SFO to avoid the network latency. Results are shown in Figure 5-20 where the sustained event rate is plotted as a function of the ROB fragment size. One can observe that for a typical ROB fragment size (1.28 kbyte), the rate achieved with full Event Building control is stable at 2.3 kHz with two ROBs per crate and 95% of LVL2 rejects.

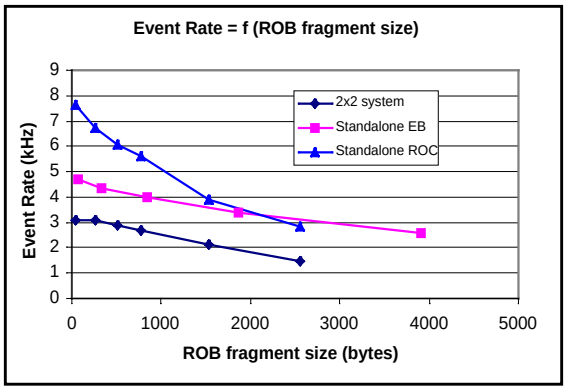

Figure 5-20 Event rate as a function of the ROB fragment size.

A simple calculation has shown that from the standalone ROC and the standalone EB results we can deduce the performance of the 2×2 system. For fragment sizes below 2.1 kbyte, the variation of the rate is governed by the EB source while above 2.1 kbyte the rate is dominated by the non EB source functionality of the EBIF. In the present implementation the EBIF application performs data collection over VMEbus, receives control information from the DFM of the EB and sends out fragments over the ATM network. The EBIF performance could be improved with the introduction of a secondary intelligent element (e.g. an intelligent PMC) which could take care of the network operations associated with Event Building, hence offloading the main CPU.

5.6.3.3 Scalability

As for performance, tests on the scalability of the integrated DAQ/EF -1 prototype have two aspects: those related to the flow of physics data and those related to the initialization, control and configuration of data-taking sessions.

The performance of the DataFlow in a final ATLAS size system is strongly dependent on the behaviour of the EB, as the number of nodes attached to the network increases. In the absence of a largescale prototype, modelling has been used to evaluate the scalability of the design and implementation of the EB. A simulation program using Ptolemy has been developed according to the design and implementation of the EB prototype and parameters describing the performance of the underlying network technology have been calibrated with the measurements carried out using the ATM (155 Mbit/s) technology. The results of this simulation are shown in Section 5.2.2.

- The interesting scalability figure related to the control and configuration of the DAQ/EF -1 prototype is directly associated with the Back-End system of DAQ/EF -1. Tests have been performed on configurations with a variable number of run controllers, LDAQs, servers and workstations. Because of limited hardware resources, large configurations require that most of the DataFlow elements have to be emulated by software. This is achieved by using an LDAQ emulator calibrated against the 2×2 baseline ROC LDAQs (4 IOMs), SFC LDAQs (2 IOMs) and DFM LDAQ. The tests are the same as those defined in Section 5.3.3. Results for different configurations and types of operations are presented

in Figures 5-21 and 5-22. The control runs on a Solaris workstation and the LDAQs run

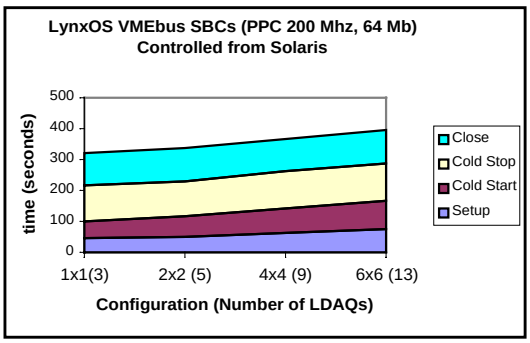

Figure 5-21 Tests results for set-up, cold start/stop and shutdown operations (Solaris / LynxOS).

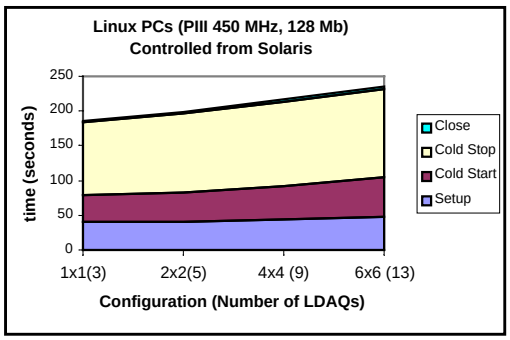

Figure 5-22 Tests results for set-up, cold start/stop and shutdown operations (Solaris / Linux).

either on VMEbus SBC / LynxOS (left) or on PC / Linux (right).

For configurations of up to 13 LDAQs controlled from a Solaris workstation, we observe that:

a. The *set-up* operation scales almost linearly with the size of the configuration. A better synchronization during the Back-End set-up will reduce this delay.

b. The *cold start* scales almost linearly with the size of the configuration.

c. The *cold stop* operations are unexpectedly long. More testing and analysis is needed to understand why this is so.

d. The *close* operation initiated from a Solaris workstation is abnormally long (more than 100 s) compared to the same operation initiated from a Linux PC (around 5 s). The problem with the Solaris workstation has been understood and will be fixed in a later release.

e. Once all the processes have been started, the time taken to change the system state (e.g. *warm start*, *warm stop*) is short and remains constant, which indicates good scalability of the distributed control.

We have also observed that the best results were obtained when running on a homogeneous system (e.g. a Linux PC controlling LDAQs running on Linux PCs).

5.7 Conclusions

The successful implementation of a full *vertical* slice of the Data Acquisition and Event Filter architecture presented in the ATLAS Technical Proposal has allowed us to draw a number of conclusions, which have been used for the architecture proposed in this document. The prototype has been constructed using existing technologies and has been used to extract information at the functional level. It does not have the ambition to completely fulfil the ATLAS requirements in terms of performance. Nevertheless the project has produced an implementation and detailed performance studies have been carried out. Details of the assessment and of the conclusions reached have been presented throughout this chapter. Here we summarize the main conclusions of the individual systems in which the project has been factorized, as well as of the global system.

The DataFlow

The design and the implementation are two major assets of the DataFlow system. The design features modularity and technology independence. The implementation features a baseline system, based on VMEbus SBCs and a number of options including the possibility of using PCs running Linux in parts of the system or a secondary bus in the ROC.

We believe that the DataFlow design is a good basis for the next phase of the development of the ATLAS HLT/DAQ system. The design has allowed different concepts to be tested and technology trends to be followed. Another important point about the design is that it has led to a full implementation.

The implementation has demonstrated that a ROC using ROBINs and PVIC as a secondary bus gives a strong indication that the 100 kHz rate required by the ATLAS experiment is within reach. Therefore, we believe that a ROC based on the DAQ/EF-1 design can be the basis for building a system meeting the ATLAS performance requirements. The same design and implementation also supports the distribution of data to the LVL2 system. As regards the Event-Builder subsystem, the scalability studies indicate that the protocol scales with the size of the switch. The layered approach taken in the software structuring allows for running the same applications on different technologies.

The Back-End DAQ

The successful completion of almost all the Back-End DAQ packages has proven the validity of the component model adopted in this part of the data-acquisition system. This model has also proven to be particularly suitable for an efficient distribution of the work amongst collaborators who are geographically dispersed.

The use of a formal software development process has clearly shown its usefulness in terms of software quality and day-to-day organization of the development work. This adaptation of a standard software development process to the project has given important insights into how such techniques can be successfully introduced into a HEP research environment.

Our assessment of the Back-End DAQ components is that their design can be adopted as a basis for the corresponding components of the final HLT/DAQ system.

The Event Filter

The Event Filter, unlike other levels of the event-selection process, is on the main data-acquisition path and hence it has been important to study it in conjunction with the DAQ so that global conclusions can be drawn.

The architecture is characterized by a clean boundary and interface between the EF and the data-acquisition functionality. The resulting flexibility in the integration of different hardware (PCs and SMPs) and software implementations, of the same designs, of EF SubFarms has supported the concept of independent SubFarms, which naturally solves problems related to Farm partitioning and, more generally, flexibility and scalability. No technical problems were encountered in running EF SubFarms geographically separated from their DAQ data source.

The Detector Interface Group

The role played by the DIG in the development phase was instrumental to the proper definition of the data-acquisition system. It is via this group that the detector requirements were specified and taken into account in the DAQ designs.

The validity and importance of such an approach was recognized ATLAS-wide and the scope of the DIG was officially extended to a more general discussion and information exchange forum between detectors, LVL1 and LVL2 triggers, DAQ/EF and DCS.

The Global DAQ/EF -1 System

A full DAQ/EF-1 prototype design and implementation has been demonstrated. Measurements carried out on the fully integrated system have shown that, at least for small-scale prototypes, the DAQ/EF-1 implementation meets the required performance. The simulation of the Event-Building system has demonstrated that the DAQ/EF-1 design is scalable and, therefore, the design could be scaled up to meet the full ATLAS DAQ system. Areas of further possible improvement have been identified concerning both hardware and software.

The DAQ/EF-1 prototype has been selected by ATLAS to be the basis of the DAQ system on the ATLAS test beams, the baseline for the evolution of the ATLAS Data Acquisition and Event Filter, as well as the basis for ROD crate data acquisition. The system is now ready to be integrated with the detector readout links and exploited as data-acquisition system for the ATLAS test beams.

5.8 References

5-1 G. Ambrosini et al., *The ATLAS DAQ and event filter prototype '-1' project*, CHEP97, Berlin, (1997)

5-2 *Technical proposal for a general purpose experiment at the large hadron collider at CERN*, CERN/LHCC/94-43 (1994)

5-3 *Global architecture for the ATLAS DAQ and trigger*, ATLAS internal note, ATL–DAQ–95–022 (1995)

5-4 *A scalable data taking system at a test beam for LHC*, CERN/LHCC/95-47 (1995) and references therein

5-5 *The dataflow for the ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–98–095 (1998)

5-6 *F/E DAQ discussion group summary document and work plan*, ATLAS internal note, ATL–DAQ–98–100 (1998)

5-7 *The design of the DAQ-Unit in ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–2000–054 (2000)

5-8 *Intra and inter-IOM communications summary document*, ATLAS internal note, ATL–DAQ–2000–051 (2000)

5-9 *The implementation and performance of I/O module variants in ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–2000–049 (2000)

5-10 *The RIO2 8060x*, product catalogue, Creative Electronics Systems, Geneva, Switzerland

5-11 *PCI Vertical InterConnect (PVIC)*, product catalogue, Creative Electronics Systems, Geneva, Switzerland

5-12 *MOTOROLA MVME2000*, `http://www.mcg.mot.com`

5-13 *The MFFC based ROBIN*, ATLAS internal note, ATL–DAQ–2000–052 (2000)

5-14 *The UK-ROBIN*, ATLAS internal note, ATL–DAQ–2000–013 (2000)

5-15 *The ROB summary document*, ATLAS internal note, ATL–DAQ–2000–053 (2000)

5-16 *Performance summary of the DAQ-Unit in DAQ/EF -1*, ATLAS internal note, ATL–DAQ–2000–050 (2000)

5-17 *A logical model for event building in DAQ -1*, ATLAS internal note, ATL–DAQ–98–112 (1998)

5-18 *The DAQ/EF -1 event builder system on Linux/Gigabit Ethernet*, ATLAS internal note, ATL–DAQ–2000–008 (2000)

5-19 J.T. Buck et al., *Ptolemy: a framework for simulating and prototyping heterogeneous systems*, Int. Journal of Computer Simulation, special issue on *simulation software development*, vol. 4, pp. 155-182 (1994); `http://ptolemy.eecs.berkeley.edu/papers/94/JEurSim`

5-20 *Detector and readout specifications, and buffer RoI relations, for the LVL2 trigger demonstrator program*, ATLAS internal note, ATL-DAQ-97-062 (1997)

5-21 *The LDAQ in ATLAS DAQ prototype -1*, ATLAS internal note, ATL–DAQ–98–094 (1998)

5-22 I. Alexandrov et al., *Performance and scalability of the Back-End subsystem in the ATLAS DAQ/EF prototype*, RT–99, Santa Fe, 1999

5-23 *Back-end summary document*, ATLAS internal note, ATL–DAQ–2000–001 (2000)

5-24 I. Alexandrov et al., *Impact of software review and inspection*, CHEP2000 conference, Padova Italy, February 2000

5-25 *Event Filter summary document*, ATLAS internal note, ATL–DAQ–2000–005 (2000)

6 The LVL2 Pilot Project

6.1 Introduction

This chapter gives an overview of the LVL2 Pilot Project and the conclusions drawn from it. It also recalls the conclusions, but not the details, from the earlier Demonstrator Programme, which was described in a Technical Progress Report (TPR) [6-1]. These and related physics and algorithm studies have led to a major evolution of the scheme proposed for the LVL2 trigger from that given in the ATLAS Technical Proposal in 1994 [6-2].

6.1.1 Principles of the Scheme in the ATLAS TP

The scheme described for the High-Level Triggers in the ATLAS TP used Region-of-Interest (RoI) guidance from LVL1 to restrict the data to be processed by the LVL2 system to a few per cent of the event data. With a few RoIs per event (typically five) data within each RoI and each detector were to be pushed in parallel from the ROBs to feature-extraction processors, which were to be either small *local* farms or very fast data-driven FPGA-based processors. The results of the feature extraction were then to be passed to a separate farm of *global* processors which would combine the features and make the LVL2 decision, with the aim of achieving a reduction factor of ~ 100. This scheme was seen to have the following advantages: trigger latency was reduced by the parallel processing, thus reducing the buffer memory size needed in the ROBs the aim was to achieve an average LVL2 decision latency of a few ms; and only small networks were required. This scheme was based on considerations for high-luminosity running; for B-physics it was realized that a modified scheme would be required with additional sequential data access and processing. However, although studies on this had started no such modified scheme had been defined at that time.

6.1.2 Studies Prior to the LVL2 Pilot Project

Between the submission of the ATLAS TP and the start of the LVL2 Pilot Project early in 1998 considerable progress was made in the understanding of the LVL2 problem. Physics studies had shown that much of the rejection could be obtained by considering just the RoIs which had caused the LVL1 trigger: the primary RoIs, typically only one or two per event. Furthermore, an initial check on only the subdetector used in LVL1 could be done very quickly and already provided a very useful reduction in the rate. The rate could be reduced further by checking the data within the primary RoIs from the inner detector; however, the algorithms for this are much slower. Thus using a sequential selection strategy, and performing the check of the LVL1 trigger with the inner-detector data only after the initial confirmation of the trigger, reduces the average latency compared to performing these checks in parallel - even though the latency for some events increases. Similar arguments apply to the processing of the secondary RoIs: the RoIs flagged by LVL1 as areas of activity, but which did not contribute to the LVL1 trigger decision. Interest arose in applying more complex algorithms for some types of events in LVL2 (e.g. inner detector full scan for B-physics, b-jet tagging). Such complex algorithms could only be run at a low rate and require a sequential strategy. In addition some of these algorithms used RoIs coming not from LVL1, but from LVL2 processing; thus a mechanism was required for the data from these LVL2-RoIs to be obtained from the ROBs.

The LVL2 Demonstrator Programme studied a number of architecture and technology options which might be used for the LVL2 trigger. These included studies of components required for the TP architecture; FPGA data-driven components; the use of a single network and single LVL2 farm; and general network studies. Small testbeds were used to study the different components and architectural ideas. An important aspect of this phase of the work was the development of *paper models*, which use average processing times, component latencies, bandwidths, trigger menus, etc. to calculate processor and network loads for a given complete LVL2 system. The results of the Demonstrator Programme were described in Chapter 4 of the TPR [6-1], but the principal conclusions were as follows:

- There was increased confidence that affordable commercial networks would be able to handle the traffic in a single network – a total of a few Gbyte/s between of the order of 1000 ports.
- Standard commercial processors (especially PCs) were favoured for the general LVL2 processing, rather than VME-based systems, since they offer a better price/performance ratio.
- FPGAs could provide fast and powerful processors for specific tasks (e.g. pre-processing and TRT Full Scan).
- Sequential processing steps are needed and sequential selection offers clear advantages, e.g. reduced network bandwidth and processor load.

  Some parallel processing should still be used (e.g. data collection, pre-processing, TRT Full scan).
- The general Push architecture was dropped.

  Control messages should pass via the same network as the data – an alternative path had been studied for a Push architecture, but it led to significant complications and no clear advantage.
- The LVL2 Supervisor should pass full event control for each event to a single processor in the farm.

6.1.3 The LVL2 Pilot Project

The Pilot Project started in early 1998, and was based on a hardware architecture which has evolved to that shown in Figure 6-1. The one difference between the architecture shown and that at the start of the project is that previously there were a small number of special FPGA processors connected to the network, to which any of the main processors could delegate some of the processing. These special processors have been superseded by co-processors, which could be included in every LVL2 processor, if required. This architecture aims to use commodity items (processors, operating system (OS) and network hardware) wherever possible.

The RoI Builder combines the fragments of RoI information from the LVL1 processors into one event record, which it passes to the Supervisor farm. The latter is made up of general-purpose processors each with an input card for receiving the LVL1 event records. The Supervisors assign each event to one of the LVL2 processors. The LVL2 processor performs one or more steps of data collection and analysis from relevant ROBs. The controlling processor for an event can delegate part of the processing to another processor, for example the FPGA-based co-processor for the TRT Full Scan. The trigger decision can be issued at any step. It is returned to the Supervisor which distributes it to the ROBs. Rejected events are discarded; accepted events are passed to

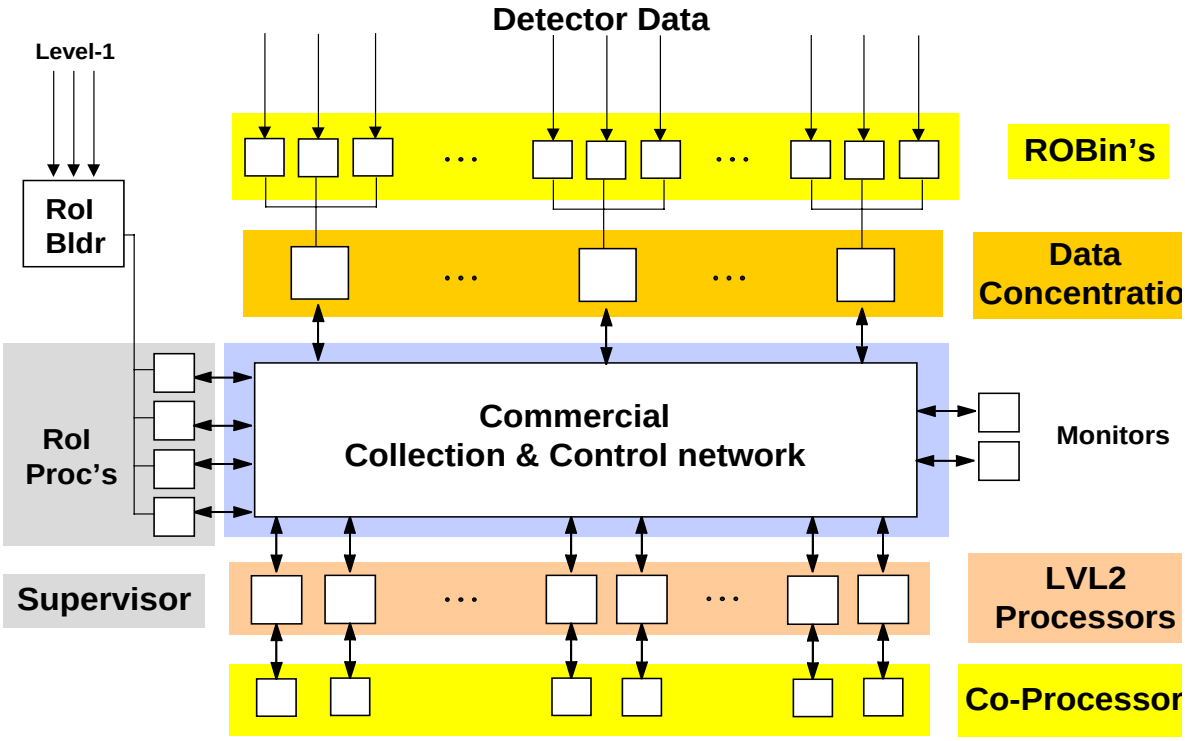

Figure 6-1 The Pilot Project LVL2 hardware architecture.

the Event Filter (EF) for further analysis. Within this architecture the event selection process is sequential, all processors can access all ROBs, and data collection from the ROBs is initiated by the processors using a request–response protocol.

The principal aims of the Pilot Project were to produce a validated LVL2 architecture for the present TP and to investigate technologies likely to be required for its implementation. To validate an architecture it was considered necessary to demonstrate at least one implementation of the hardware and software, with component performance at the required level, in a system which is shown to be scalable and affordable. The work and conclusions of the Demonstrator Programme had produced a preferred architecture, described above, and prototype components for some parts of an implementation. However, it was clear that further work was required for some key components (primarily hardware). There was also a clear need for a common software and hardware basis to test the components in an integrated way, preferable in moderately sized systems. It would also be necessary to consider integration with other parts of Trigger / DAQ and the scaling to a full-sized system with computer simulations. The work of the Pilot Project was thus divided into three main areas: functional components, testbeds and system design. The functional components covered optimized components for the Supervisor and RoI Builder, the ROB Complex[1], networks and processors (especially FPGAs). Testbeds covered the development of the Reference Software, a prototype implementation for the complete LVL2 process, and the construction and use of moderately large application testbeds to use this software. Finally, system design covered modelling activities and an integration activity to consider issues related to how the LVL2 system integrates with other subsystems and the requirements it has to meet.

1. The ROB Complex, described in Section 6.4 below, comprises a number of input buffers operating under a common local intelligence with a common output interface to LVL2.

Each activity had one or two convenors to organize the work in that activity and to aid communications between the activities. In addition to meetings of the participants within an activity, there were several meetings to review the whole project and to interchange information between all those involved. Because of the geographical spread of the participants much use was made of telephone- and video-conferences, with a bi-weekly video-conference for the whole project.

Particular emphasis was placed on the Reference Software and the application testbeds. The Reference Software was to provide a single software framework across the Pilot Project, building on the conclusions of the Demonstrator Programme. The testbeds were to use this software with the following aims: to check that individual components meet the required performance; to provide information on scaling up to moderate size systems; and to provide data for the full-system computer models. The further optimization of the functional components built on earlier work in the Demonstrator Programme, and would provide optimized components for later integration into the testbeds. The modelling was to check the extrapolation of the testbed results to the final ATLAS system and integration was to keep an overview of the final system and its integration into the entire Trigger / DAQ system.

Indicative performance requirements for the LVL2 components are given by the paper models [6-3]. For a 75 kHz LVL1 rate at low luminosity, and using current ROB mappings and trigger menu assumptions with sequential selection, these give the values shown in Table 6-1.

Table 6-1 Indicative parameters and performance requirements for LVL2 components.

Parameter	Value	Comment
Total bandwidth in LVL2 network	~ 5 Gbyte / s	Reduces to ~ 3 Gbyte / s with pre-selection of calorimeter data and TRT data compression
Max RoI request rate per ROB	~ 14 kHz	
Max output bandwidth to LVL2 per ROB	~ 9 Mbyte / s	Reduces to ~ 4 Mbyte / s with pre-selection of calorimeter data and TRT data compression
Max RoI Builder / Supervisor rate	75 kHz	
Average number of ROBs required to supply data per RoI	10–35	Depends on RoI type
Typical number of ROBs per data request	~ 4	In e.m. calorimeter only true if layers treated separately
Data for TRT full scan	~ 200 kbyte from ~ 256 ROBs	Reduces to ~ 80 kbyte with TRT data compression
Typical number of RoIs per event	1–2 primary / ~ 3 secondary	
Average number of sequential steps executed	~ 2	
Event rate per LVL2 processor	> 0.1 kHz	

6.2 Reference Software

The aim of the *Reference Software* [6-4] was to provide a prototype implementation for the complete LVL2 process and a common software framework for the Pilot Project activities. These activities include evaluation of networking technologies, in particular ATM, Fast and Gigabit Ethernet, and SCI (Scalable Coherent Interface); evaluation of components such as the RoI Builder, ROB Complexes and co-processors; development and evaluation of physics and run-time performance of LVL2 event-selection algorithms; measurements of critical parameters on multinode testbeds to obtain indications of the system scalability; and validation of the software architecture.

6.2.1 Software Process, Requirements and Design

The project started at the beginning of 1998 with the participation of eight ATLAS institutes in Europe and the US, a total of ~ 30 collaborators with ~ 20 contributing to code. An object-oriented (OO) approach with C++ as the main implementation language was adopted from the beginning. With the distributed team, some of whom had no experience of OO programming, and the very tight time constraints it was decided to use an informal software process: requirements (until April 1998), design (until September 1998) and implementation (first release March 1999). The software development was managed using a web server,[1] a common file system (AFS), a software repository (CVS) and weekly meetings with telephone conferencing.

The requirements, many from the LVL2 URD (User Requirement Document)[6-5], others specific to the Pilot Project, include platform independence with support for Linux and WNT on PCs; independence of networking technologies with implementations for ATM, Ethernet and SCI; deployment of the same version of algorithms and data sets for physics performance studies and online testbeds using emulated detector data; simple run control, error reporting and monitoring adequate to support testbeds of up to ~ 100 nodes. Figure 6-2 shows the main features of the design which is strongly influenced by the conclusions of the earlier Demonstrator Programme [6-1]. Communication is based on a request–response protocol to transfer data between functional components.

6.2.2 Implementation

The implementation had to provide a Trigger Processor (which combined the functionalities of *steering* and *feature extraction*), plus emulations of the ROB Complex and the Supervisor. These objects could be implemented on the same processor, e.g. for development of algorithms on the desktop, or distributed over multiple nodes for online testbeds. For some tests the emulations of the ROB Complex and Supervisor would be replaced by prototype components.

The Reference-Software framework is based on the (remote) proxy *design pattern* [6-6] to make communications transparent to the applications which are unaware of the possible distribution of functions over different processors. Algorithms may be executed on a single- or multinode system without change – this includes the possibility of running part of the algorithm in a co-processor. The framework handles all communication, dispatching and conversion of data.

1. http://www.cern.ch/Atlas/project/LVL2testbed/www/

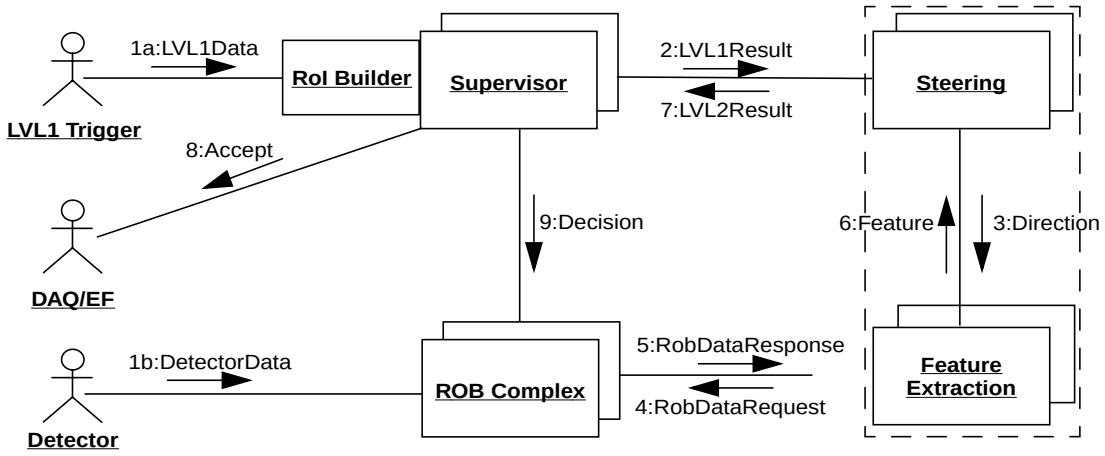

Figure 6-2 Collaboration diagram for the Reference Software showing the LVL2 functional components and external actors, and the communication between them.

Multiple *worker threads*, each dealing with one event, may run in parallel on a single node to allow overlap between communication and processing and exploit multiple CPUs within a node.

The required versatility – multiple platforms (Linux, WNT), multiple networking technologies (ATM, Ethernet, SCI), multiple environments (desktop, online testbeds, hybrid systems with co-processors) and extensibility to allow for new concepts – has been achieved by organizing the software as a set of layered packages as shown in Figure 6-3.

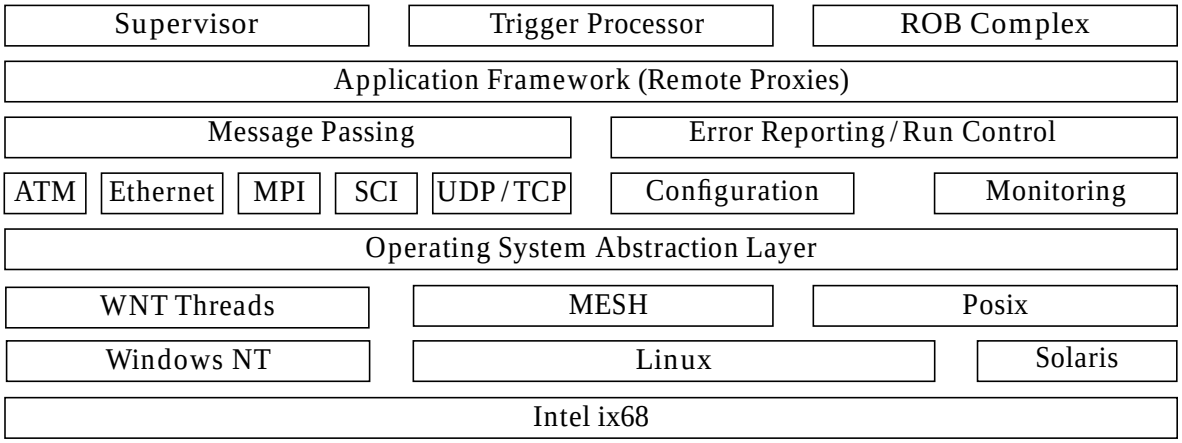

Figure 6-3 Reference-Software layering. MESH [6-7] is a low-latency message passing and scheduling code used with Fast/Gigabit Ethernet.

The full functionality is implemented as one application for a single node or split into three applications (Supervisor, Trigger Processor, ROB Complex) for the distributed version. A farm of each of these is used to obtain a high trigger frequency, enough CPU power for the algorithms and enough bandwidth to supply the detector data to the Trigger Processors.

A single PC or workstation (Linux or WNT) may be used for the development and evaluation of algorithms. LVL1 information and RoI data are read from ASCII files; a configurable menu *steers* the execution of feature-extraction algorithms (chosen from an algorithm table) to produce trigger elements which are matched with the physics signatures of the menu. This results in a LVL2

decision block which terminates the processing of the event. Feature-extraction algorithms (in C++) include calorimeter clustering (electron, gamma, jet, tau), TRT (RoI driven and full scan) and precision tracking (LUT based or Kalman filter). A typical example of a two-step sequential process consists of a confirmation of the calorimeter RoIs, followed by the TRT algorithm.

Configuration, Error Reporting & Logging, Monitoring, Process Management and Run Control, were implemented in a minimal way, but based on the design of the DAQ/EF -1 Back-End software [6-8] to facilitate a change to that software when it became available. The implementation used a client/server approach. The entire Reference Software is self-contained and consists of ~ 25 packages containing ~ 100 000 lines of code. To use it an ANSI standard C++ compiler and Standard Template Library (STL) are required.

6.2.3 Results

Dependencies on the OS have been successfully encapsulated in an OS interface layer. The standard thread scheduling provided by the native OS (Linux, WNT or Solaris) has been used in the ATM, SCI and MPI [6-9] testbeds. The Ethernet testbed used an optimized package MESH [6-7] to provide fast thread switching and support the optimized Ethernet driver. Real-time scheduling recently introduced in Linux needs further investigation as efficient context switching is critical. The software is SMP-ready but tests to date have concentrated on single and dual CPU PCs.

Performance measurements with ATM and Ethernet used optimized drivers. The much heavier TCP and UDP protocols were used mainly for software development and testing. Some of the tests with SCI used MPI, with only a small performance loss. However, the performance of MPI can be very dependent on the implementation and the network technology. Similarly *in-house* proxies have been used, rather than the heavier CORBA/ILU/ACE. Further optimizations, such as avoiding unnecessary copying and conversion of data and possibly inefficient use of OO are under investigation.

A comprehensive set of C++ algorithms developed by the Physics and Event Selection Algorithm (PESA) group and included in the framework produce identical results on desktop workstations and testbeds. Configurable menu and algorithm tables allow for the evaluation of different trigger scenarios.

6.2.4 Conclusions for the Reference Software

Though not yet optimized the I/O performance obtained with the testbeds, a few per cent of the final size, gives good indications that the requirements of the LVL2 trigger can be met with this software architecture. Further details are given in the following sections. The software has been proven to scale to systems of moderate size using a commercial cluster of ~ 100 dual CPU PCs with a fast interconnect (SCI) at the University of Paderborn [6-10]. These tests also demonstrated stable performance of the Supervisor versus the number of processors; and correct operation of the Reference Software on a large system.

The request–response based architecture has been validated.

Tests indicate that the required component performance can be obtained with commodity hardware (PCs) and OS software (such as PC/Linux). A variety of networking technologies (ATM, Fast/Gigabit Ethernet, SCI) satisfy the requirements on small testbeds, but need optimized

drivers. Similar performance was also seen in other tests with a commercial OS (WNT/Solaris); Message Passing protocol (MPI); on an integrated commercial cluster (at the University of Paderborn).

6.3 The Testbeds

The testbeds [6-11] were established to use the Reference Software with the following aims: to check that individual components meet the required performance; to provide information on scaling up to moderate size systems; and to provide data for the full-system computer models. The testbed systems vary in size from 25 to 50 nodes (plus as mentioned above the ~ 100 node system at Paderborn University). These systems correspond to a few per cent of the final ATLAS system. Ethernet (Fast and Gigabit), ATM and SCI technologies have been studied for the network. All the testbeds were based on the hardware architecture shown in Figure 6-1, although for most tests there were no co-processors and the Supervisor/RoI Builder and ROB Complexes used the Reference-Software emulations running on PCs. However, prototype components (i.e. Supervisor/RoI Builder, ROB Complex, an FPGA processor) developed in the functional-component activities were integrated for some tests.

The Ethernet and ATM testbeds (see Figure 6-4) share the same PCs, which are single- or dual-processor machines with processor speeds of 200–450 MHz. The ATM testbed also uses ten PowerPC single-board computers running LynxOS. The network equipment for ATM is a 48-port, 155 Mbit/s per port, FORE switch. For Ethernet three BATM Titan 4 Fast or Gigabit switches with up to 32 Fast ports or 4 Gigabit ports per switch were used.

Figure 6-4 The Ethernet and ATM testbeds in CERN.

The SCI testbed has 23 single- or dual-processor PCs with processor speeds of 300 – 450 MHz. The SCI switch is a 16-port Dolphin switch - each port connects to an SCI ringlet which can contain more than one processor node. The Siemens cluster at Paderborn University, has 96 dual-processor nodes (450 MHz Pentium II) and also uses SCI for the interconnect, but with multiple SCI ringlets in a Torus configuration, without a switch.

The Reference Software has been run on all testbeds under Linux and Windows NT, and at Paderborn under Solaris. The ATM testbed was also run with the C-based *ATM Testbed Software* developed for the Demonstrator Programme under Windows NT, Linux and LynxOS. A series of measurements [6-12] were devised for the testbeds to obtain performance results of the various LVL2 components. The following sections consider each of the components in turn. Results from the testbed measurements are given together with the description of the functional-component activities.

6.4 Requirements of the ROB Complex and Implementation Studies

The ReadOut Buffers (ROBs) are a key component affecting the performance of the LVL2 system and have consequently been the subject of much investigation by the LVL2 community. The study of the ROBs within the Pilot Project thus built on the considerable experience gained from previous LVL2 work in which prototype buffers were successfully operated in the vertical-slice configurations of the demonstrator and earlier programmes.

The effort in the Pilot-Project phase was directed towards consolidating this earlier work with a comprehensive review of requirements and design issues, and further exploration of the problem space via independent design studies. The aim was to record the points that could be established within the limits of available knowledge and decisions, and provide checklists of the remaining issues for the next development phase.

Although account was taken of all issues affecting the ROB design, the R&D efforts concentrated on the areas closest to the experience of the LVL2 group, namely LVL2 requirements and optimizations, and hardware implementation options. The approach was to combine paper design and system modelling with prototyping of hardware. Key parameters for the assessment of different LVL2 processing strategies and for different system scenarios were obtained from the paper models [6-3] and computer models [6-13], whilst the feasibility of different implementation approaches was demonstrated in performance measurements of several hardware prototypes.

A key focus of the work was to investigate the trade-offs in grouping sets of buffers into what was termed a *ROB Complex*. A canonical ROB Complex comprises a number of input buffers (ROBins) operating under a common local intelligence (ROB Controller) with one or more common output interfaces (ROBout). Such grouping potentially offers efficiency gains, and consequent cost savings, on both architectural and technological grounds. It was noted that elements of the DAQ/EF -1 ReadOut Crate (ROC) could be considered to constitute a ROB Complex with suitable renaming. The study of ROB Complexes included the limiting case in which the Complex comprises just a single input buffer.

As the ROB Complex subproject progressed, design issues and options, and project information, were documented in a *master working document*. This document, entitled *Options for the ROB Complex* [6-14], is available, along with fuller documentation on individual prototype designs and measurements, and scenario studies, via the ROB Complex Web page.[1]

6.4.1 Operation of a ROB Complex

The overall architectural framework assumed in the development work is specified in the master working document [6-14] and in UML descriptions [6-15]. The basic operation is as follows:

- Data are received into the ROBs from the detectors across readout links with a bandwidth of up to 160 Mbyte/s and at an average event rate of up to 100 kHz.
- Selected data are requested from the buffers by the LVL2 system, at a maximum rate of about 14 kHz for any given buffer.

1. http://www.nikhef.nl/pub/experiments/atlas/daq/ROB.html

- The final LVL2 decisions are passed back to the ROBs so that memory occupied by rejected events can be released.

  To reduce message handling overheads it is more efficient to pass the decisions back in groups (of 20 or more decisions).

- Data for accepted events are passed downstream for processing by the EF.

  The LVL2 accept rate is projected to be about 1–2 kHz and the required buffer size per ROB to be 2–64 Mbyte depending on the assumed processing model and buffer-management scheme.

The Pilot-Project studies have demonstrated that buffering from ATLAS compatible readout links (S-LINK) at the projected ATLAS event rates can be achieved with a number of variant designs; the NIKHEF ROBin prototype [6-16] has directly achieved a 160 Mbyte/s input rate, whilst prototypes [6-17] built to an earlier 100 Mbyte/s specification have been successfully operated in testbeds and are believed to be upgradable without major problems of principle.

The LVL2 sequential selection strategy requires that the RoI data are supplied to the LVL2 processors only when a sequential step requests it. Thus, within an event the LVL2 system can generate several asynchronous requests for data. In addition to supplying data from RoIs identified by LVL1, the ROB Complex should also be able to supply the LVL2 system with data from a complete subdetector and data from new RoIs identified by LVL2. This capability is needed, for example, to run a B-physics trigger at LVL2 using a full track search in the inner detector followed by analysis in the calorimeters of any muon or electron candidates identified.

Grouping buffers in the ROB Complex can significantly reduce the total number of network interfaces to LVL2 and the total number of messages in the network (see *Tables of fractional reduction of the output fragment rate* [6-14] and [6-3]). Whether this leads to financial savings depends on the relative cost of external and internal links. Studies of ROB Complex prototypes indicate that the internal communication can be supported with available point-to-point or bus-based links.

Measurements of ROB Complex emulators (running on PCs) in testbed configurations have demonstrated the operation of ROBout interfaces. The principal aim was to provide a data source, consistent with that expected from a ROB Complex, for testing other components in the testbed. The performance of this emulation was measured in different testbeds. The rate of requests for many processors which can be met by a single ROB Complex emulator is shown in Figure 6-5. The performance is consistent with that used in paper models and gives confidence that the network connection from the ROB Complex to the LVL2 system is attainable over the network technologies studied.

In addition, a prototype of the Saclay ROBin was integrated into the ATM testbed. A ROB Complex composed of 1, 2, and 3 ROBins was tested on VME, CompactPCI (LynxOS) and PC-Linux platforms. The average rate of requests coming from the processors that can be serviced by the ROB Complex is shown in Figure 6-6. The ROB controller was a 400 MHz Linux-PC that served the requests from ~ 10 processors via a 155 Mbit/s ATM link. In this test, ROBins were not connected to external sources and random event data were returned to the requesters. For typical fragment sizes of 1–2 kbyte, the maximum measured service rate almost reached the bandwidth limit of the ROB Controller link (16 Mbyte/s in this implementation). Although these results are compatible with current estimates of requirements, further investigations on a more complete set-up are needed.

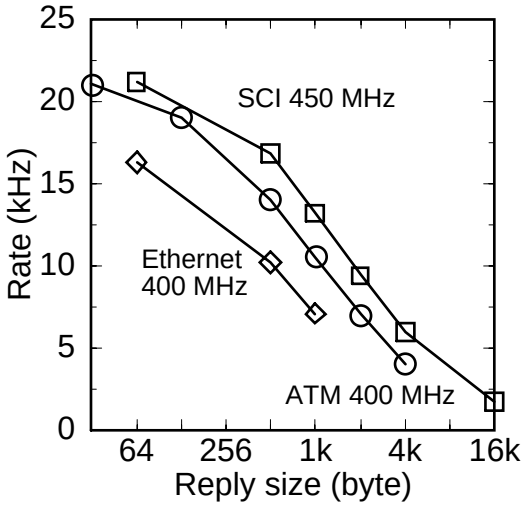

Figure 6-5 ROB Complex Emulator service rate. Testbed measurements of the rate at which a single Reference Software ROB Complex emulator could service requests from many processors.

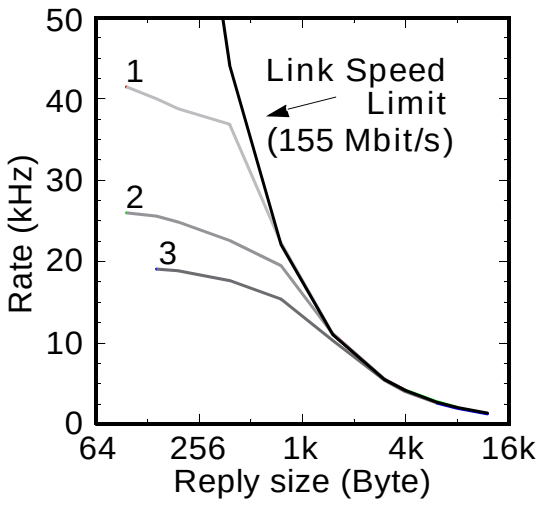

Figure 6-6 Prototype ROB Complex service rate. Measurements of the request–response rate for a prototype ROB Complex containing 1, 2 and 3 ROBins.

Further performance gains can be made if *preprocessing* is performed on the data within the ROB Complex. Pre-processing options considered range from simple data collection and reformatting, to full feature extraction. (Some studies [6-18], [6-19] refer to an *Active ROB Complex* which implies a ROB Complex with considerable processing capabilities.) The options with the most promise are: zero-suppression and bit compaction of the TRT data; suppression of irrelevant calorimeter data; clustering of adjacent SCT silicon strips or pixels into single-track hits and conversion of coordinates from local to global. These offer reductions either in the bandwidth of data to be transferred across the LVL2 network or in the processing time in the LVL2 processors. Feature extraction, however, requires all of the data from the RoI to be available and consideration of the probability of an RoI being contained within a practical single Active ROB Complex leads to the conclusion that this is unlikely to be practicable [6-18]. An initial demonstration of a prototype Active ROB Complex [6-19] has been made using a four-processor PC board, equipped with four FPGA-based µENABLE boards [6-20] acting as ROBins. With 1 kbyte fragments an aggregate bandwidth of 160 Mbyte/s was collected from the four ROBins via two PCI buses. Other tests have demonstrated reformatting of TRT data within the µENABLE based ROBin. It has also been shown that the selection of calorimeter data relevant for LVL2 can be performed by ROBins without an overall performance degradation of the ROB Complex [6-21].

The mechanism for passing data to the Event Filter remains to be decided, but there is no great problem in principle with any of the current prototype designs, whether the data are pushed immediately to an Event Builder or held in the ROB until received by the Event Filter, or whether or not the Event Filter connection is separate from the LVL2 network.

6.4.2 ROB Complex Conclusions

From the above and more detailed aspects of these studies the following conclusions are drawn.

It has been demonstrated that the requirements can be satisfied with current technology: projected input rates can be handled; output to LVL2 and to the Event Builder is achievable at the

necessary rates and bandwidth; some *on-the-fly* pre-processing is possible with the use of spare processor capacity or of FPGAs in the ROBins or in an *Active ROB Complex*; both page-managed and circular-buffer-based buffer management work; both point-to-point and bus systems are viable for internal communication in the ROB complex; and software control provides flexibility.

The implementation studies show that for the ROBin a compact design with acceptable power dissipation is possible. This would allow the construction of a ROB Complex with several (3–6) ROBins on a single Eurocard-size board, or from a single-board computer with ROBins implemented on mezzanine boards. COTS[1] hardware seems to be able to achieve the requirements with respect to output to the LVL2 system and to the EB. It has been shown that advantage can be taken of multi-processor and multi-bus COTS systems with minimal effort. However, true COTS hardware is not necessarily able to handle the input rates.

The scenario studies and modelling results show that there are potential advantages to (flexibly) grouping buffers in a ROB Complex and performing some degree of local processing. The optimal degree of buffer grouping within a ROB Complex will be strongly influenced by the cost and bandwidth of network ports and it is too early to draw any conclusion. The scenarios suggest that it is also worth studying further two limiting case variants of the ROB Complex: one in which the complex is collapsed into a *simple ROB*, potentially reducing cost and complexity by directly coupling a buffer on each input link directly to the output network; the other in which a farm processor, directly connected to a group of buffers, is used to provide both ROB Controller functionality and network interfacing, thus significantly reducing the total number of components.

The Pilot Project has provided the following checklists for the guidance of future design work: requirements (including UML description); data formats; run-control states; errors, error handling & timeouts; status & statistics reporting; implementation issues; testing.

6.5 The Supervisor and RoI Builder

During the Demonstrator Programme the concept of using a small farm of processors for the LVL2 Supervisor was developed. However, an additional unit is required to combine the different streams of RoI information from LVL1 and distribute the data to processors within the Supervisor farm. During the Pilot Project a prototype unit, the RoI Builder, was designed, produced and tested [6-22]. The implementation of the Supervisor farm was further developed and integrated with the Reference Software. This section describes these developments and the tests carried out with these units.

6.5.1 Overview

The basic organization and functions of the RoI Builder and Supervisor are described in Section 4.2.3 and a block diagram is given in Figure 4-3. On each LVL1 accept signal (L1A), the RoI Builder receives RoI information fragments from the LVL1 processors [6-23]. These RoI fragments are organized and formatted into a record for each event. The RoI Builder then transfers the record to a selected Supervisor processor (RoI processor). The Supervisor processor

1. COTS is here used as Commodity Off The Shelf (cf. Commercial Off The Shelf used elsewhere in this document).

manages the event through LVL2. It allocates the event to a LVL2 processor; forwards the RoI record to this target processor; receives the decision back; updates the statistics; packs the decisions and multicasts them to the ROBs.

This prototype development is intended to demonstrate the feasibility of a candidate architecture capable of building RoI records at the maximum LVL1 trigger rate of 100 kHz without introducing dead time.

6.5.2 Design and Implementation of the RoI Builder

The LVL1 processors send a number of RoI fragments per event, which may be considerably skewed in time. The RoI Builder prototype must receive these, provide assembled RoI records for up to eight RoI processors, and operate without introducing deadtime at event rates as high as 100 kHz. In order to meet these requirements, the prototype was implemented entirely in hardware, using FPGAs from the Altera 10K family, with a design emphasizing parallelism. This extremely dense logic allowed an architecture where essentially there are eight RoI Builders operating in parallel. The LVL1 event ID, embedded in every RoI fragment, is used to identify the RoI fragments belonging to a given event and for the assignment to an RoI Builder channel using a hardware allocation algorithm (round-robin in this implementation). The channel for which the event ID is relevant passes the fragment to a second buffer where the record is built, the other channels discard the fragment. The RoI record is built in 2 μs after receipt of the last fragment of an event, and is immediately transferred via S-LINK to the target RoI processor.

An important feature of the design is that by using custom hardware for the most demanding tasks, it reduces the demands on the other Supervisor components, so that they can use standard processors.

As implemented, each RoI Builder card can build RoI records from as many as 12 input data streams, and can service two RoI processors. The RoI Builder is completely data driven and scalable. Figure 6-7 shows one of the prototype RoI Builder cards. There are 12 inputs for data streams from LVL1 carried via copper in S-LINK format, six input FPGAs which are configured as 12 input buffers, 12 FPGAs which provide the two secondary buffers in which the records are built, and two FPGAs on pin grid arrays which manage transfer of the assembled records to the two target RoI processors served by this card.

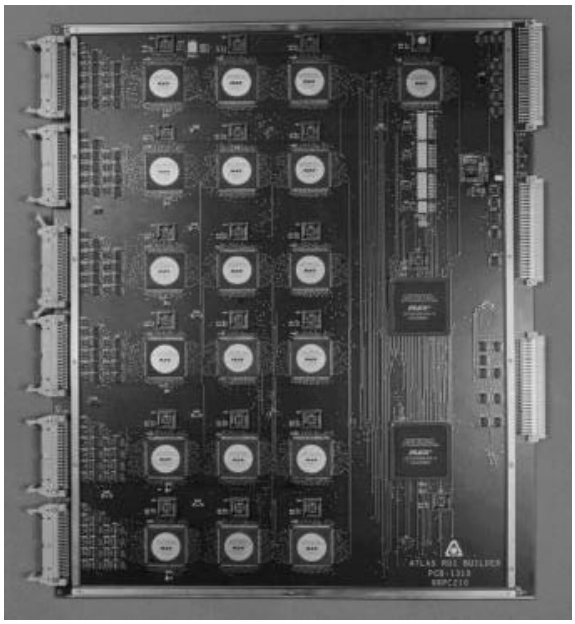

Figure 6-7 Prototype RoI Builder card.

6.5.3 RoI Builder Tests

An input card was designed and built for tests of the RoI Builder and for use as a LVL1 emulator in the testbeds. This input card emulates six LVL1 processors, and supplies RoI fragments to the RoI Builder. This input card can be loaded from VME with RoI fragments for 1024 events, and these events can be initiated at 12 software selectable rates, or under VME control.

The RoI Builder hardware was first tested with C++ diagnostics, running under LynxOS. Further tests of the RoI Builder were run in a system containing one input card; two RoI Builder cards; two S-LINK output cards (four S-LINK output channels); Four i686 PCs running under Linux or four RIO2s running under LynxOS. The purpose of this testing was to make sure the system ran without errors, and to investigate the LVL1 trigger rates that could be supported without having the Supervisor create deadtime. In these measurements, 1, 2, and 4 Supervisor nodes were used. The event rate was measured both for S-LINK transfer only and for S-LINK transfer and data unpacking where the system output was checked for errors.

6.5.4 Integration into Testbeds

The Supervisor / RoI Builder was integrated into ATM and Ethernet testbeds. Here the results of tests run in a 32-node ATM network are given. The goals of this work were to operate the RoI Builder with 1, 2, or 4 Supervisor nodes, to find the limits of Supervisor performance using existing processors (200 and 300 MHz RIO2s), and to investigate the effect that S-LINK flow control may have on the performance. Extensive testing was conducted, and a brief summary of results is presented in Table 6-2.

Table 6-2 Summary of RoI Builder/Supervisor Tests.

Configuration	Max Rate (kHz)	μs/event
RoI Builder card Readout	67	15
1 Supervisor + 1 RoI Builder card	29	34
2 Supervisors + 1 RoI Builder card	27 + 28 = 55	18
4 Supervisors + 2 RoI Builder cards	23 + 23 + 24 + 24 = 94	10.6

Within the testbeds, the Supervisor concepts were also tested using Supervisor emulators. Measurements included scaling with the number of Supervisor emulators and the dependence of the rate of a single emulator as a function of the number of RoIs per event (see Figure 6-8).

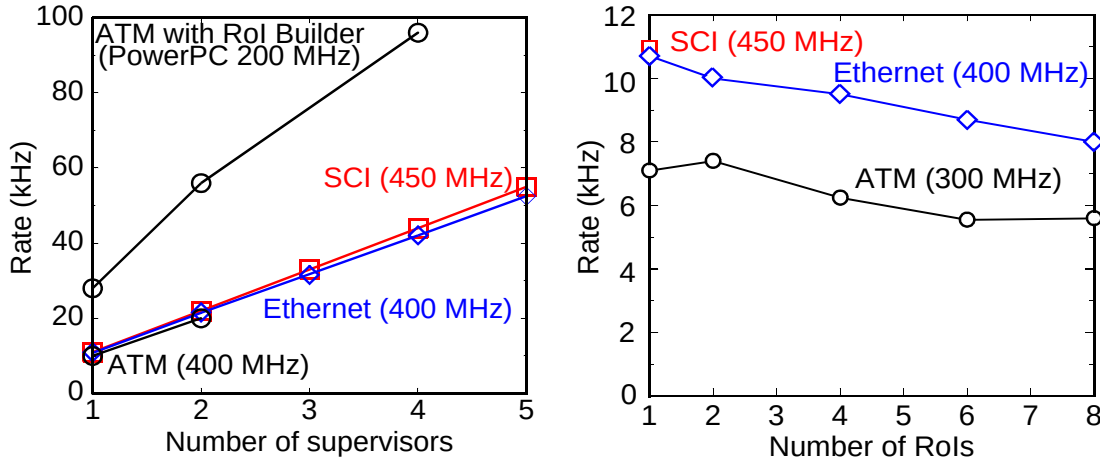

Figure 6-8 Supervisor scaling and variation with number of RoIs per event.
a) Scaling of the Supervisor rate (both the prototype RoI Builder and Supervisor, and the Supervisor emulator) with the number of Supervisor processors.
b) Variation in the Supervisor emulator rate with number of RoIs per event.

With a single RoI, a rate of ~ 11 kHz per Supervisor emulator is reached. The rate is independent of the number of ROBs when a hardware multicast is used. The results also show that the system rate scales with the number of Supervisors and a rate of 120 kHz was achieved with twelve Supervisor emulators (no RoI Builder) on the Paderborn cluster. It was also found that the rate versus the number of LVL2 processors increases linearly until the Supervisor is saturated.

6.5.5 Conclusions

A prototype RoI Builder has been built using FPGAs in a highly parallel architecture. It has been integrated with a Supervisor farm into ATM and Ethernet testbeds. The RoI Builder plus a small Supervisor farm have been shown to satisfy the requirements for the LVL2 trigger, i.e. up to a rate of up to 100 kHz.

6.6 Processor Requirements and Measurements

The first task of a LVL2 processor is to collect data from many sources and the second task is to process the data received. The emphasis in the testbeds has been on the first task and on quantifying the resources needed for this. The system was configured to saturate a processor using up to 16 ROBs. To study the effect of the protocol overheads, as a function of the number of ROBs involved, short data blocks of 64 byte were collected from each ROB and the event rate which could be sustained by a processor with no algorithm processing time was measured. As shown in Figure 6-9 a processor can sustain a rate of 5–7 kHz for one ROB, for a typical RoI request involving four ROBs this drops to 3–4 kHz. The data size collected from the ROBs was also varied. For the collection of more representative 1 kbyte and 4 kbyte data sizes (from a single ROB) rates of 4 kHz and 2.5 kHz are achieved, respectively. Whilst the data collection times are significant, the total farm sizes envisaged imply much lower rates per processor, leaving the larger part of the time for algorithm processing.

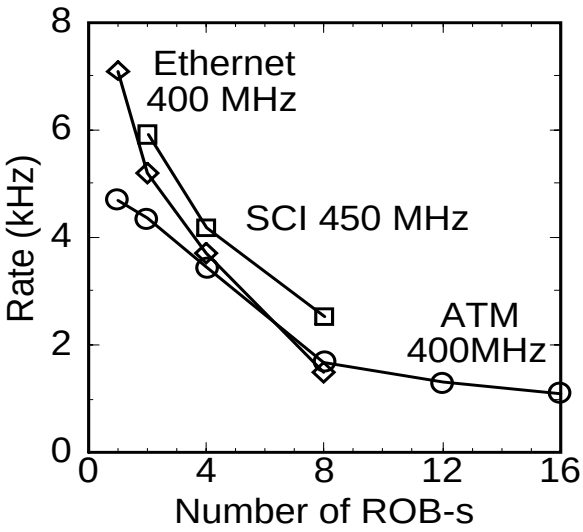

Figure 6-9 Processor data gathering rate. The rate a single processor can sustain gathering data as a function of the number of ROBs per RoI is shown.

While the number of ROBs involved in data collection for an RoI is ~ 4–16, several other tasks require gathering data from all the ROBs of a given subdetector (e.g. all TRT ROBs in a complete search for tracks), several subdetectors, or in the case of event building the whole detector. This data collection was investigated with the ATM testbed software. The task of each processor was to collect data from 20 ROBs. The processors used a hardware multi-cast over ATM to distribute to the ROBs the request messages initiating the data transfers. The transmission rate was controlled at the ROB level to avoid traffic toward any given output port of the switch over-running the buffer and bandwidth capacity of the port. Measurements show that the system throughput is proportional to the number of destination processors. With 20 processors, a sustained global

throughput of 260 Mbyte/s and 328 Mbyte/s is measured for ROB data fragments of 2 kbyte and 4 kbyte respectively. Data blocks of 80 kbyte equally spread across 20 ROBs are gathered at 4 kHz.

As already noted the use of sequential selection reduces the network and processor requirements and allows more complex algorithms to be run at lower rates. To validate the principle of multi-step data transfers and processing a testbed run was made on the cluster at the University of Paderborn [6-24] with the Reference Software including algorithms for three detectors: calorimeter e.m. clustering [6-25], TRT (Hough transformation) tracking [6-26] and SCT/pixel precision tracking [6-27]. The Supervisor and RoB emulators were preloaded with a data file containing ~ 3000 jet events with no pile-up, preselected to contain at least one LVL1 e.m. RoI. about 15% of the events contained two RoIs. The menu consisted of three consecutive steps (calorimeter alone, calorimeter followed by TRT and finally calorimeter + TRT + SCT/Pixel) each selecting 20 GeV electrons. The applied cuts were close to those described in Chapter 8, but without tuning for detector effects. The fraction of events accepted after each step was 0.19, 0.05 and 0.02.

The latency is measured in the Supervisor as the time interval between sending the LVL1Result and the reception of the LVL2Result (see Figure 6-2). It includes communication, data preparation and actual processing time (steering and feature extraction). Communication delays contribute ~ 500 μs. The RoI data size is ~ 10–20 kbyte, contributing another ~ 300 μs for each RoI/detector combination. The distribution in Figure 6-10 reveals the sequential execution of the three algorithms: calorimeter e.m. clustering predominantly below 4 ms, subsequent TRT tracking at 4–7 ms and final SCT/pixel precision tracking extending beyond 7 ms. The average and median values are 3.7 ms and 3.1 ms, respectively. The effect of rejection at early stages in the sequential process is shown explicitly in Figure 6-11: 50% of the events finish within 3.1 ms, 95% within 7.2 ms and 99% within 10.8 ms. (Note the processors are 450 MHz dual Pentium II.)

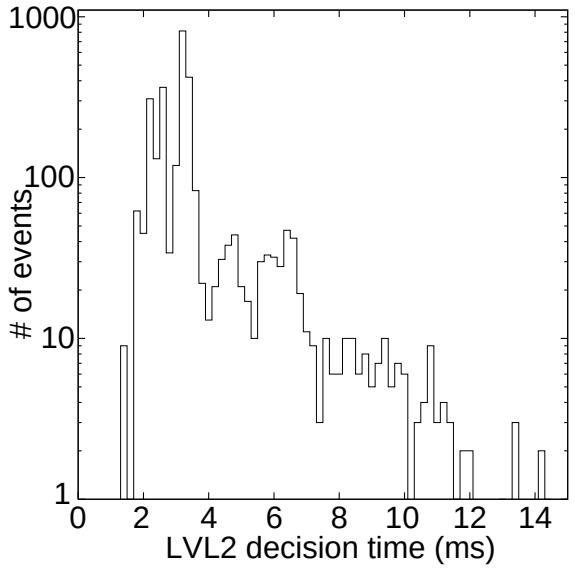

Figure 6-10 Testbed LVL2 trigger latency. Contributions around 3 ms, 5 ms and 9 ms correspond to the calorimeter, TRT and SCT/pixel feature-extraction algorithms.

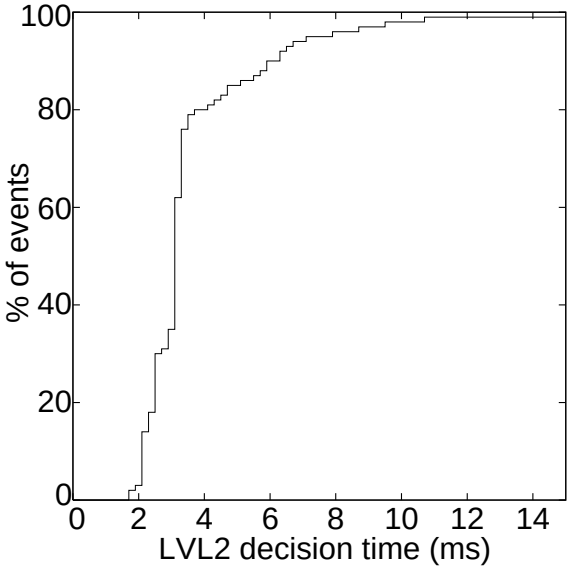

Figure 6-11 LVL2 trigger latency (integrated). The fraction of events which finish within a given time interval is shown.

The main conclusions for the processors are:

- The collection of data within an RoI from a single detector can be done at an acceptable rate.
- Large amounts of data, corresponding to all of the data from a single detector, have also been gathered into the processors as would be required for an inner detector full scan.
- A three–step sequential selection strategy has been demonstrated with prototype algorithms running on a multi-node testbed with the processor requesting simulated event data from ROB emulator nodes.

6.7 Use of FPGAs as Co-Processors

Modelling [6-3] shows that the size of the LVL2 trigger farm required may be determined primarily by the need of the B-physics trigger to execute a track search in the full inner-detector volume. Tracks could be found by a full scan of the pixel detector and / or the TRT, followed by a Kalman filter (or Hough transform) algorithm in the SCT. The full-scan algorithms allow considerable parallelism and are good candidates to run in FPGA-based processors. The performance expected from FPGAs should allow a considerable reduction in the number of LVL2 processors required. This section describes studies made of FPGA systems, especially using them as co-processors and for an implementation of the TRT full scan.

6.7.1 The ATLANTIS Processor System

The main focus of the FPGA implementation studies during the Pilot Project has been in the ATLANTIS processor system [6-28]. This is a combined FPGA and CPU-based computing system housed in a CompactPCI crate. A standard Intel Pentium PC – a CompactPCI computer – which plugs into one of the ATLANTIS active backplane slots is used for external connections. The FPGAs are mounted on the ATLANTIS Computing Board (ACB), (Figure 6-12). Each ACB can accept up to four RAM boards. In an eight-slot ATLANTIS system up to seven ACBs can be used. Communication between the ACBs and the PC is via the PCI backplane. When connected to a testbed the ATLANTIS system appears as a normal PC with accelerator features.

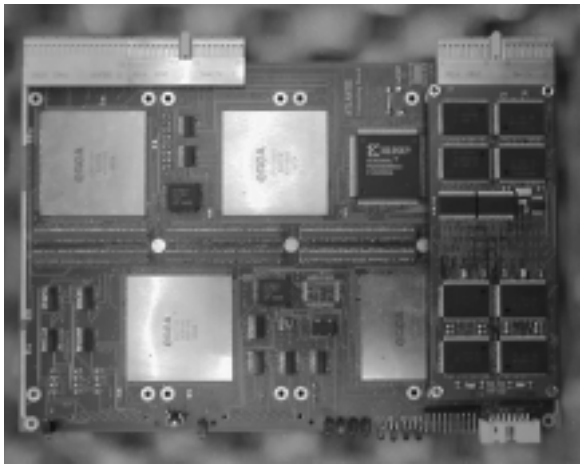

Figure 6-12 The ATLANTIS computing board with one RAM board.

6.7.2 Implementation of the TRT Full-Scan Algorithm in ATLANTIS

The full-scan TRT algorithm [6-29] was implemented on ATLANTIS, with the most time-consuming parts (the Hough transform, histogramming and peak-finding, which account for ~ 89% of the execution time on a Pentium) performed in the FPGAs. The main difference between the FPGA and CPU implementations of this algorithm is that the inner loop, which is executed once per active straw and increments many histogram counters, is executed sequentially

in the CPU and in parallel in the FPGA. The transform is performed on-the-fly as the TRT hits arrive in the ACB and the execution time is largely determined by the PCI data transfer rate. Tests with a single ACB processing the algorithm for one half of the TRT barrel led to a factor 6 improvement in the total execution time compared to a 300 MHz Pentium II (a factor of 9 in the part transferred to the FPGA) [6-30]. These tests continue, including investigation of several ways to improve the speed-up factor, even with the current generation of FPGAs.

In addition to standalone tests, the ATLANTIS system, running WNT, was successfully integrated into the ATM testbed [6-30]. The ATLANTIS system appeared to the Reference Software as a normal steering processor. At the time of these tests it was not possible to make meaningful latency and bandwidth measurements within the testbed environment. However it was demonstrated that event data could be transferred from ROBs to the ATLANTIS system, fully integrated into the testbed, and that the algorithm quality using the ACB was identical to the CPU-only implementation.

6.7.3 Prospects for FPGA Systems

The future prospects for FPGA systems are good. Detailed models [6-31] have shown that as early as 2001 an FPGA co-processor with one large FPGA and SRAM with large word length in combination with a commodity PC will be capable of executing the full scan of the TRT in 3.17 ms. This is more than ten times faster than a current 600 MHz PC. Putting this time in a paper model [6-3] reduces the number of LVL2 processors required from 608 to 369.

In general the computing power of FPGAs is increasing very rapidly and it seems likely that FPGA systems will also become easier to use and cheaper [6-31]. Thus using FPGA-based co-processors should be a cost-effective and powerful way to augment the processing power of standard PCs for appropriate algorithms.

6.7.4 Conclusions

The TRT full-scan algorithm has been implemented on the hybrid ATLANTIS system with a factor ~ 6 speed-up compared to a CPU-only implementation. The ATLANTIS system was integrated into the Reference Software and a testbed, appearing to the rest of the system as a standard node. Various ways are being investigated to overcome the limited data transfer rates via PCI and a further speed-up of 1.5–2 should be possible. For the future, higher speed PCI (64 bit @ 66 MHz, cf. 32 bit @ 33 MHz today) or the new Infiniband [6-32] could offer better I/O to the FPGA board. This work indicates how FPGA co-processors could be included in standard processors in a transparent way, offering significant performance improvements for suitable compute-intensive algorithms and hence a reduction in the size of the processor farms required.

6.8 Meeting the Network Requirements with Available Technologies

6.8.1 Network Requirements for the ATLAS HLT/DAQ

Networking technologies for the ATLAS HLT/DAQ system have to fulfil many important requirements. They should allow large data collection networks to be built connecting the ROBs

(either simple ROBs or ROB Complexes) to hundreds of destination processors. Depending on the detector readout and event selection strategy, the raw bandwidth requirement is estimated to be in the range of 4–6 Gbyte/s. The networks have to transport various types of traffic with different requirements in terms of bandwidth, message rate and latency. Protocol messages are characterized by a relatively small size (~ tens of bytes), a high rate (~ tens of kHz per node), and mainly flow from the destination processors toward the ROBs. Multi-cast capability is likely to be required (e.g. to distribute trigger decisions to the ROBs). Data traffic, characterized by the concentration of messages toward the processors from a number of ROBs, requires a high bandwidth. Data collection of RoIs requires low communication overheads, whilst for the full scan or event building, care must be taken to resolve network congestion, minimize data loss and sustain the event rate.

A general trend, for both HLT and DAQ systems, is to use the same network to transport protocol messages and event data. Use of the same networking technology for the LVL2 and DAQ systems would clearly simplify development and facilitate maintenance. In addition, it would give the possibility to have a common network for data collection at LVL2 and EF, and facilitate resource sharing between the two systems. Finally, using commercial products with a wide user base promises the best price/performance ratio and long-term maintainability.

6.8.2 Studies of ATM

Investigations on ATM for the ATLAS HLT/DAQ have been pursued since 1995. The work has been conducted using complementary methods: modelling and the construction of demonstrators. Ref. [6-33] describes in detail these architectural, conceptual and technology studies. Some important results relevant to ATM are summarized here.

Modelling studies indicate that ATM components are adequate to build high-performance networks capable of transporting simultaneously protocol messages and event data. Computer simulations show that ATM can handle the data traffic in large event builder systems. Mechanisms are available in ATM to avoid congestion and minimize the influence of network contention on HLT/DAQ system performance. A network common to the data collection systems for LVL2 and EF is feasible for the hypothesis made in these studies.

On ATM testbeds at Saclay and Osaka, and on the 48-node ATM testbed at CERN, many architectural principles have been validated: request–response protocol, sequential event selection, integrated data and control network. Although this is hidden for the applications in the Reference Software and the ATM testbed software, the underlying protocol for message passing is ATM Adaptation Layer 5. For performance reasons, the TCP/IP stack is removed and when available, zero-copy user level-drivers are used. Running the ATM testbed software, the demonstrators were operated in the following different configurations: a LVL2 system with sequential event selection; a standalone event builder; a combined LVL2/event builder with a common network and separate processor farms; a combined LVL2/event builder with a common network and a single farm of processors.

In all modes of operation, the same request–response protocol is used. Logical partitioning of the network was validated by running concurrently on a testbed several independent subsystems in different modes of operation. Within the size of the testbeds, no congestion is detected even at a load close to the saturation point of the network when using a best effort service for RoI data collection and rate division for full event building. In event builder mode, global throughput scales linearly with the size of the system until the link speed limit is reached. Mul-

ti-switch topologies were investigated to build a large network that can satisfy the bandwidth and connectivity requirements for the ATLAS HLT/DAQ.

6.8.3 Studies of Ethernet

Over the last two years the possible use of Fast and Gigabit Ethernet in the ATLAS LVL2 trigger has been investigated. Work has been carried out in a number of complementary areas including:

- The evaluation and optimization of the connection between the network and processor nodes [6-7], [6-34].
- Evaluation, testing and characterization of different manufacturer's products [6-35], leading to the construction and calibration of computer models for switching components [6-36].
- Performance modelling of full-size Ethernet networks suitable for the LVL2 trigger [6-36].
- The use of Ethernet as an infrastructure on which to build application testbeds using the Reference Software.

The conclusions are as follows:

- A high node I/O data rate can be obtained by using optimized drivers and by discarding the TCP/IP stack.
- A wide range of high-performance switches exists in the rapidly evolving market place.

  The leading edge of these products would already allow us to assemble networks capable of meeting the needs of the LVL2 trigger. This has been verified by the preliminary results of computer simulation.
- The use of Ethernet with optimized drivers in the application testbeds was able to demonstrate the level of component performance (ROBs, Supervisor and processors) required by the final trigger system.

6.8.4 SCI Studies

The application of SCI in ATLAS HLT/DAQ has been studied for several years, during which the equipment available has continued to evolve. During the Pilot Project SCI interfaces supporting links running at 400 Mbyte/s were tested together with a 16 port SCI switch [6-37], [6-38]. The hardware support for memory mapping on the interfaces allows a CPU or DMA engine on one node to write directly to memory in another node without going via a network driver. The bandwidth between two nodes is then determined by the PCI performance, provided that the interface receives data fast enough to use the optimum 64-byte data transfers. Under these conditions bandwidths of more than 80 Mbyte/s were observed. (The PCs used had 32-bit PCI buses; however the interfaces also support 64-bit PCI, which allows even higher rates.) For these applications a message-passing protocol was added [6-39], which ensured correct receipt of the data; this had a negligible effect for long messages, but significantly reduced the bandwidth for short messages. The switch was tested with over 20 nodes (each port can be connected to a ringlet with several nodes) and a total throughput of > 700 Mbyte/s was measured in a worst-case configuration of traffic.

6.8.5 Future Networking Trends

ATM is a well-established technology among telecommunication operators and internet service providers. It has large industrial support and availability of products in the long term is expected. Many vendors offer switching fabrics with a capacity of several tens of Gbit/s and a few hundred 155 Mbit/s ports. Higher speed links (e.g. 622 Mbit/s or 2.4 Gbit/s) are likely to be deployed only to interconnect switches. Prices for a 155 Mbit/s switch port and the associated network interface card are ~ $1000 and ~ $500, respectively.

Ethernet is a very well-established international standard for 10 Mbit/s, 100 Mbit/s and 1 Gbit/s networks. There is a new 10 Gbit/s standard under rapid development. Its longevity on the time-scale of the ATLAS experiment is assured. Ethernet has 80% of the enormous LAN market ($37 billion in 2000) and is the clear winner on the desktop. Competition between the numerous manufacturers is intense and prices are dropping in almost all product areas. The current price of a 100 Mbit/s connection, including the cost of a network interface card and a switch port, is $200. The corresponding cost for Gigabit Ethernet is $2000. Gigabit Ethernet prices fell by almost a factor of two in the last year.

SCI is becoming more widely adopted; however, it is likely to remain in a niche market with small volumes and few sources. The technology continues to improve, but is probably only of relevance to ATLAS inside commercial clusters (such as that at Paderborn). Given this it has been decided that it is not appropriate to make further studies of SCI for ATLAS.

The evolution and deployment of ATM and Ethernet will continue to be followed closely. ATM and Ethernet remain as candidate technologies for the ATLAS HLT/DAQ application. In both technologies the cost of a large network (~ 1000 ports), including switches and host adapters, appears to be within the ATLAS cost estimates.

6.9 Extrapolation to a Full System

Extrapolation to the full system has been done using both *paper models* and discrete event simulations. Discrete event models were first shown in the ATLAS TP. Since that time the original version written in ModSim has evolved into a version written in C++ [6-13]. In addition a separate model [6-40], using an Ethernet switch model, has been developed based on Ptolemy. These models can simulate sequential processing based on the LVL1 trigger menus also used for the *paper models* [6-3]. These tools have also been used to model behaviour in the testbeds, to obtain a deeper understanding of the testbed results and to calibrate and check the models.

The architecture of the model of a full-scale system is shown in Figure 6-13. The interconnecting switch is assumed to be capable of multicasting (for decision blocks and data requests for inner-tracker full scans). Events are generated internally in the program on the basis of a trigger menu. Different RoIs for the same event are assumed to be uncorrelated and a random position is chosen in the available η–ϕ space for each RoI. The mapping of the detector into the ROBs [6-41] is used to determine for each RoI position which ROBs should be sent the data requests. A simplifying assumption is that the size of an event fragment sent by a ROBin to the LVL2 processor farm for a given subdetector and RoI type is always the same.

The trigger menus, parameters and models used for paper modelling are also used in the simulation. Very good agreement has been found between corresponding results from the paper model and the C++ program (when average values, rather than distributions, are used for

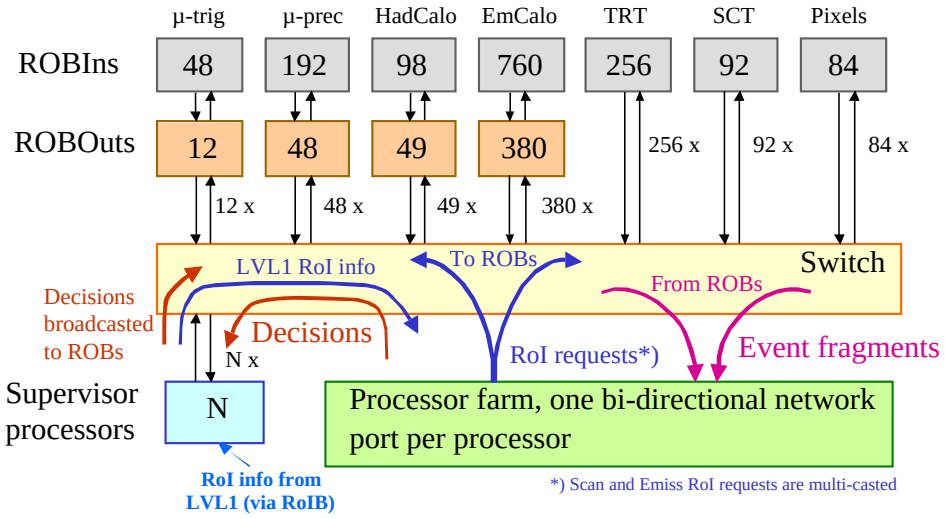

Figure 6-13 Architecture of the full-scale system model. Two Supervisor processors are used.

processing times and event-fragment sizes). The Ptolemy simulation work, which started in the last year, is expected to help with further verification of results and will also provide an environment for detailed network switch models. The full system results presented here are from the C++ model.

The parameters and models have been defined using estimates and experience based on results available at the beginning of 1999. During 1999 only the information with respect to the mapping of the detectors on the ROBs and the sizes of the event fragments has been updated. Results from recent testbed and associated modelling work need to be fed into the current full-system model. However, these results show that most assumptions made are reasonably realistic. The model assumes a simple single large crossbar switch. For reasons of simplicity the model uses two uni-directional switches, although in reality a single switch will support bi-directional traffic. Note it is believed that there will be little interference between the different data streams so this should not be significant. The effective link speed and internal bandwidth out of and into each buffer inside the switch are important parameters. The results shown are for a link speed of 15 Mbyte/s and for 60 Mbyte/s per crossbar connection.

The processor assignment strategy used by the Supervisor has been shown to have a significant effect on the performance of the system. The sequential processing, and variations with the RoI type and position give rise to a large variation in processing time per event. Thus rather than a simple round-robin algorithm it is better to assign the event to the processor with the least number of events queued in it. For the low-luminosity trigger, if B-physics processing is done at LVL2, it is also best to avoid queueing multiple events with one or more muon RoIs, since these will frequently continue to the relatively long B-physics trigger algorithms. Each Supervisor processor should manage the part of the LVL2 farm assigned to that Supervisor processor only.

Another factor that has a significant impact on the operation of the system is the size of the event fragments. The LVL2 trigger only needs the first 1024 bytes of the 1800 bytes of each calorimeter event fragment. Sending only these first parts reduces the transfer time across the network (in particular for jet RoIs), the amount of queueing and therefore the decision time. For the low-luminosity trigger it probably is feasible to reduce the TRT fragment size from 750 to 300 bytes. This has also a beneficial effect on queueing in the system. The results shown are for the case in which the type of pre-processing described is performed in the ROBs.

Figure 6-14 shows the results from the model for the distribution of the decision time for 1.2 million triggers (30 s of real running) for the low-luminosity trigger for a farm of 450 1000-MIPS-processors (the processing requirements for the trigger would be fulfilled by 360 processors). With this decision-time distribution, less than 1000 fragments need to be buffered in the ROBs, so a ROBin buffer memory size of 2 Mbyte would be sufficient. The peak for decision times around 45 ms is due to the B-physics trigger – note the peak is sharper than it would be in reality due to the assumption of fixed algorithm execution times. The small number of events with decision times to the right of this peak and just below it shows that the assignment algorithm is working well. Using a round-robin algorithm results in a distribution extending to long decision times (see Ref. [6-13]).

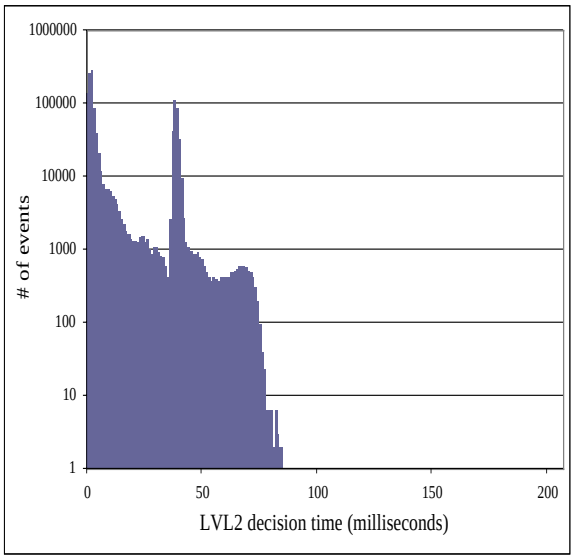

Figure 6-14 LVL2 decision time distribution for the low luminosity trigger.

For the high-luminosity trigger it is assumed that there is no B-physics trigger, so the decision-time distribution is similar to the part of Figure 6-14 below 20 ms.

The model allows the simulation of LVL2 and event-building traffic on the same network. Preliminary studies indicate that when the crossbar connection bandwidth is 100–200 Mbyte/s or lower interference can occur. The degree of interference depends on internal details of the switch; this will be studied further as the more detailed switch models become available.

The components on the Ptolemy model have been calibrated using the Ethernet testbed measurements. For the limited testbed systems studied the model agrees with measurement to within 5% [6-40]. The calibrated model components have been used to simulate the first step of the trigger using a multi-stage Ethernet switch in a three subdetector system. The multi-stage switch is composed of a central Gigabit Ethernet switch connected to a number (currently ~ 30) of Fast Ethernet switches. The latter connect to groups of ROBs or processors. Initial results have been obtained for the LVL2 decision latency with such a system [6-40]. A full-system model with all subdetectors and trigger steps is being finalized. It is expected that this system model will be used to study the influence of different parameters and configurations on the performance of the system.

The principal conclusions from the modelling studies are as follows:

- The high p_T LVL2 triggers may be handled by a similar number (100–200) of 1000 MIPS processors – at both high and low luminosity.
- The maximum data volume for LVL2 through the network is 20–40 Gbit/s for low luminosity (including the inner detector full scan for B-physics) and 10–25 Gbit/s for high luminosity.
- The inner detector full scan increases the processing power and network bandwidth required in the LVL2 processors significantly. Current estimates indicate that ~ 450 processors, each of 1000 MIPS and with an FPGA co-processor, would be needed. Without the

co-processors the number increases to ~ 770. (The increase compared to the high-p_T case could also be used as a safety margin for the high-p_T running.)

6.10 LVL2 Integration

The Pilot Project also included activity to consider various integration issues. This work included producing an update of the LVL2 User Requirements Document (URD) [6-5]; producing an updated description of the ROB mappings and data formats for the detectors [6-41] (originally requested for the modelling activity this compilation has generated wider interest and it is therefore proposed to add appendices to include corrections); providing a forum to discuss general LVL2 ideas, such as mechanisms to provide the LVL2 data to the Event Filter.

6.11 Conclusions

The principal conclusions stated above are reiterated here and some general conclusions drawn.

The Reference Software has been run in many configurations, using various network technologies. It has been shown to scale to systems of up to ~ 100 nodes. The tests indicate that the required component performance can be obtained with commodity hardware (PCs) and OS software (such as PC/Linux). The request–response based architecture has been validated. Though not yet optimized the I/O performance obtained with the testbeds gives good indications that the requirements of the LVL2 trigger can be met with this software architecture.

Extensive results were obtained in testbeds of up to 48 nodes with three optimized network technologies (ATM, Fast/Gigabit Ethernet and SCI) and over MPI on a 96-node commercial cluster. Optimized hardware components were integrated into the testbeds for the Supervisor and RoI Builder; ATLANTIS FPGA hybrid processor; and a ROB Complex (with the ATM software from the Demonstrator Programme).

It has been demonstrated that the ROB requirements can be satisfied with current technology: projected input rates can be handled, output to LVL2 and to the Event Builder is achievable at the necessary rates & bandwidth, some on-the-fly pre-processing is possible. A compact design for the ROBin is achievable, allowing the construction of a ROB Complex with several (3–6) ROBins. COTS[1] hardware seems to be able to support the output requirements but perhaps not the input rates. The Pilot Project has provided the following checklists for the guidance of future ROB design work: requirements (including UML description); data formats; run-control states; errors, error handling & timeouts; status & statistics reporting; implementation issues; testing.

A prototype RoI Builder has been built using FPGAs in a highly parallel architecture. The design uses custom hardware for the most demanding tasks, which reduces the demands on the other Supervisor components, so that they can be implemented using standard processors. The RoI Builder has been integrated with a Supervisor farm into ATM and Ethernet testbeds. The RoI Builder plus a small Supervisor farm have been shown to satisfy the requirements for the LVL2 trigger, i.e. up to a rate of 100 kHz.

1. COTS is here used as Commodity Off The Shelf (cf. Commercial Off The Shelf used elsewhere in this document).

Processors in testbeds using the Reference Software have demonstrated data collection within an RoI from a single detector at an acceptable rate; a three-step sequential selection strategy with prototype algorithms running on a multi-node testbed with the processor requesting simulated event data from ROB emulator nodes.

The TRT full-scan algorithm has been implemented on the hybrid ATLANTIS system with a factor ~ 6 speed-up compared to a CPU-only implementation. The ATLANTIS system was integrated into the Reference Software and the ATM testbed, appearing to the rest of the system as a standard node. This work indicates how FPGA co-processors could be included in standard processors in a transparent way, offering significant performance improvements for suitable compute-intensive algorithms and hence a reduction in the size of the processor farms required.

Extensive tests were carried out with three network technologies (ATM, Fast and Gigabit Ethernet, and SCI). The first two are *commodity* and therefore particularly interesting candidates for ATLAS. Using optimized drivers and discarding the TCP/IP stack allowed all three to give high node I/O data rates. Link speeds are increasing from the 100 Mbit/s to the Gbit/s range – already a wide range of high-performance switches exists in these technologies. ATM and Ethernet remain as candidate technologies for this application. In both technologies the cost of a large network (~ 1000 ports), including switches and host adapters, appears to be within the ATLAS cost estimates.

Models of a full-scale system indicate that the high-p_T LVL2 triggers may be handled by similar number (100–200) of 1000 MIPS processors at both high and low luminosity; the maximum data volume for LVL2 through the network is 20–40 Gbit/s for low luminosity (including the inner detector full scan for B-physics) and 10–25 Gbit/s for high luminosity; the inner-detector full scan increases the processing power and network bandwidth required in the LVL2 processors significantly. Current estimates indicate that ~ 450 processors, each of 1000 MIPS and with an FPGA co-processor, would be needed. Without the co-processors the number increases to ~ 770. (The increase compared to the high-p_T case could also be used as a safety margin for the high-p_T running.)

In general it can be concluded that:

- The main aims of the Pilot Project have been reached.
- The options chosen at the end of the Demonstrator Programme have been validated.
- The software architecture has been validated.
- Choices of components and candidate technologies for LVL2 now have a large overlap with choices for DAQ/EF (PCs for processors, ATM and Fast/Gigabit Ethernet for networking).

6.12 References

6-1 *ATLAS DAQ, EF, LVL2 and DCS technical progress report*, CERN/LHCC/98–16 (1998)

6-2 *Technical proposal for a general purpose experiment at the large hadron collider at CERN*, CERN/LHCC/94–34 (1994)

6-3 *Paper modelling of the ATLAS LVL2 trigger system*, ATLAS internal note, ATL–DAQ–2000–030 (2000)

6-4 *The ATLAS LVL2 reference software*, ATLAS internal note, ATL–DAQ–2000–019 (2000)

6-5 *ATLAS LVL2 trigger user requirements document,* ATLAS internal note, ATL–DAQ–2000–034 (2000)

6-6 E. Gamma, R. Helm, R. Jonson, J. Vlissides, *Design patterns,* Addison–Wesley (1995)

6-7 M. Boosten et al., *Fine-grain parallel processing on commodity platforms*, in Architectures, Languages and Techniques, Ed. B.M. Cook, IOS Press, pp. 263–276

6-8 *Back-End summary document*, ATLAS internal note, ATL–DAQ–2000–001 (2000)

6-9 Message Passing Interface (MPI), `http://www-unix.mcs.anl.gov/mpi/`

6-10 Paderborn Center for Parallel Computing, `http://www.uni-paderborn.de/pc2/systems/psc/`

6-11 *Results from the LVL2 pilot project testbeds*, ATLAS internal note, ATL–DAQ–2000–040 (2000)

6-12 *A minimum set of measurements to be made on the application testbeds,* ATLAS internal note, ATL–DAQ–99–005 (1999)

6-13 *Computer modelling of the ATLAS LVL2 trigger,* ATLAS internal note, ATL–DAQ–2000–035 (2000)

6-14 *Options for the ROB complex*, ATLAS internal note, ATL–DAQ–2000–027 (2000)

6-15 *A UML description of the ATLAS ROB viewed from the LVL2 trigger,* ATLAS internal note, ATL–DAQ–2000–009 (2000)

6-16 *A SHARC based ROB complex*, ATLAS internal note, ATL–DAQ–2000–021 (2000)

6-17 *The UK ROBin, a prototype ATLAS read-out buffer input module*, ATLAS internal note, ATL–DAQ–2000–013 (2000)

6-18 *The active ROB complex*, ATLAS internal note, ATL–DAQ–2000–022 (1999)

6-19 *The use of low-cost SMPs in the ATLAS LVL2 trigger*, ATLAS internal note, ATL–DAQ–2000–010 (2000)

6-20 O. Brosch et al., *MicroEnable - a reconfigurable FPGA coprocessor*, Proc. 4th Workshop on Electronics for LHC Experiments, Rome, Italy (1998), 402–406, CERN / LHCC / 98–36

6-21 *A scheme of read-out organisation for the ATLAS high-level triggers and DAQ based on ROB complexes*, ATLAS internal note, ATL–DAQ–2000–014 (2000)

6-22 *A prototype RoI builder for the second level trigger of ATLAS implemented in FPGAs*, ATLAS internal note, ATL–DAQ–99–016 (1999)

6-23 *Specification of the LVL1/LVL2 trigger interface, version 1.0,* ATLAS internal note, ATL–DAQ–99–015 (1999)

6-24 *Running the ATLAS second level trigger software on a large commercial cluster,* ATLAS internal note, ATL–DAQ–2000–024 (2000)

6-25 *First implementation of calorimeter FEX algorithms in the LVL2 reference software*, ATLAS internal note, ATL–DAQ–2000–020 (2000)

6-26 *Global pattern recognition in the TRT for the ATLAS LVL2 trigger,* ATLAS internal note, ATL–DAQ–98–120 (1998)

6-27 *Performance of a LVL2 trigger feature extraction algorithm for the precision tracker,* ATLAS internal note, ATL–DAQ–99–013 (1999)

6-28 K. Kornmesser et al., *ATLANTIS – a hybrid approach combining the power of FPGA and RISC processors based on CompactPCI*, International Symposium on Field Programmable Gate Arrays, Monterey, California, February 1999

6-29 *Global pattern recognition in the TRT for B-physics in the ATLAS trigger*, ATLAS internal note, ATL–DAQ–99–012 (1999)

6-30 *LVL2 full TRT scan feature extraction algorithm for B-physics performed on the hybrid FPGA/CPU processor system ATLANTIS: Measurement results*, ATLAS internal note, ATL–DAQ–2000–012 (2000)

6-31 *Prospects of FPGAs for the ATLAS LVL2 trigger*, ATLAS internal note, ATL–DAQ–2000–006 (2000)

6-32 Infiniband, `http://www.futureio.org/home.html`

6-33 *An integrated system for the ATLAS high level triggers; concept, general conclusions on architecture studies, final results of prototyping with ATM*, ATLAS internal note, ATL–DAQ–2000–011 (2000)

6-34 M. Boosten, *Fine-grain parallel processing on a commodity platform: a solution for ATLAS*, Draft thesis to be submitted to Eindhoven University of Technology, `http://home.cern.ch/mdobson/mesh/M_Boosten_thesis_141299.ps`

6-35 *Investigation of the performance of 100 Mbit/s and Gigabit Ethernet components using raw Ethernet frames*, ATLAS internal note, ATL–DAQ–2000–032 (2000)

6-36 *Modelling Ethernet switches for the ATLAS LVL2 trigger*, ATLAS internal note, ATL–DAQ–2000–044 (2000)

6-37 *Evaluation of commercial SCI components and low-level SCI software for the ATLAS second level trigger*, ATLAS internal note, ATL–DAQ–2000–029 (2000)

6-38 *ATLAS LVL2 trigger SCI demonstrator evaluation report*, ATLAS internal note, ATL–DAQ–2000–041 (2000)

6-39 *Implementation of the message passing software layer over SCI for the ATLAS second level trigger testbeds*, ATLAS internal note, ATL–DAQ–2000–028 (2000)

6-40 *Ptolemy simulation of the ATLAS LVL2 trigger*, ATLAS internal note, ATL–DAQ–2000–039 (2000)

6-41 *Detector and read-out specification, and buffer-RoI relations, for LVL2 studies*, ATLAS internal note, ATL–DAQ–99–014 (2000)

7 DCS

7.1 Introduction

The principal task of the DCS is to enable the coherent and safe operation of the ATLAS detector. The main requirements have been laid down in Ref. [7-1]. Early in 1998 the four LHC experiments and the CERN controls group, IT/CO, started a joint controls project(JCOP) [7-2] with the aim of arriving at a common solution.

Besides the obvious supervision of the subdetectors and the common technical infrastructure of the experiment, a homogenous way of communication with the infrastructure services of CERN and the LHC accelerator is needed. Special attention has to be given to the interaction of the DCS with the ATLAS DAQ system.

The DAQ system and the DCS are complementary in as far as the first treats all aspects of the physics event-data, which are identified by an event number, and the second deals in general with the other data, which are normally categorized with a time stamp. All data of the second type, which are needed for understanding the behaviour of the detector and for the subsequent physics analyses, have to be acquired, analysed and stored by the DCS.

The DCS has to continuously monitor all operational parameters, signal any abnormal behaviour to the operator and give him guidance. Besides executing operator commands (e.g. high voltage ramp), the DCS must also have the capability to automatically take appropriate actions if necessary and to bring the detector into a safe state.

It is mandatory that, concerning the hardware of the detector, all actions initiated by the operator and all errors, warnings and alarms are handled by the 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 go via the DCS. In this way, it can be verified internally that the operations requested are safe for the equipment. In safety-critical areas a special interlock system, which is also monitored by the DCS, has to be implemented in parallel. However the DCS is not responsible for the safety of the personnel. For this a dedicated system, which the DCS must not be able to influence, will exist.

In the following the overall architecture will be presented first. Then the two main components of the DCS, the supervisory system and the front-end I/O system, will be discussed in detail. The communication with external systems and the connection mechanisms with the DAQ system, including aspects of interaction during operation, will follow. Finally the workplan is presented.

7.2 Architecture

7.2.1 General Ideas

The ATLAS DCS is a distributed control system. It consists of a Supervisory Control And Data Acquisition system (SCADA) and of front-end systems. The name SCADA indicates that the

functionality is two-fold: It acquires the data from the front-end equipment and it offers supervisory control functions, such as data processing, presenting, storing and archiving. This enables the handling of commands, messages and alarms.

The front-end systems can be described in terms of devices and of I/O points. Front-end systems can range from simple I/O devices to complex computer-based systems that are connected to the SCADA systems by the network. A SCADA Real-Time (RT) database contains records where the data values are stored.

The DCS can be partitioned into vertical slices. Such a partition can be operated completely independently from other slices of the DCS and offers the full SCADA functionality to its users. A vertical slice controls a subsystem of the ATLAS detector, where a subsystem is defined as an arbitrary part of the detector (the high-voltage system of a subdetector, a subdetector itself, etc.).

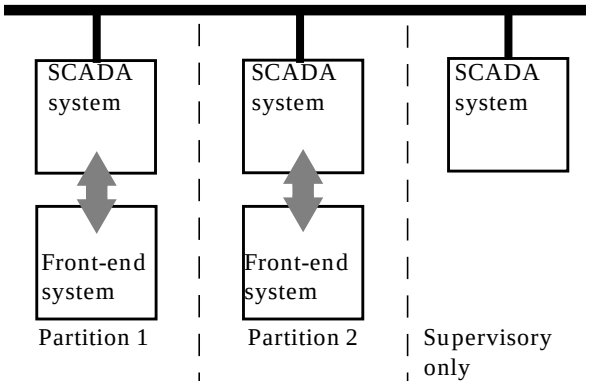

Figure 7-1 DCS partitioning.

Two or more partitions can be combined into one control domain by connecting the RT databases to each other. Obviously, a control domain can consist of a single partition as well. Partitions within this control domain exchange information with each other by reading and writing the records in the distributed RT database. Apart from this database access, there are no other communication links established between individual partitions of the control system. Figure 7-1 shows an example of a fragment of a control system that consists of two partitions connected to front-end systems and of a SCADA system dedicated to the supervisory control tasks without a direct connection to the front-ends. These partitions can be dynamically set up, e.g. on request of the DAQ system in order to match its partitioning.

7.2.2 Logical Organization

Based on the architecture described above, the ATLAS control system can be logically structured in a hierarchical way. Figure 7-2 shows this structure, where DCS partitions are represented as *bubbles*. During construction and maintenance of the detector, each partition can operate autonomously with respect to the others. From the lowest level on, one can start to combine the partitions into a logical structure. For instance the control partitions of the MDT detector can be combined into one control domain to be able to control the MDT as a logical entity. The MDT control system can be combined with the controls of the other muon detectors, like the TGC, CSC and RPC, to be able to obtain the integrated control of the muon detector and, as a final step, the muon control domain can be combined with the controls of all other subdetectors into the overall ATLAS control system.

Besides the partitions of the detector, there are also some components of a general nature:

- The *common services* component implements ATLAS-wide services, like the overall status display, alarm handling, archiving and so on.
- The *common infrastructure* component is responsible for the control of equipment that is shared by all subdetectors, such as electronics racks, global cooling, etc.

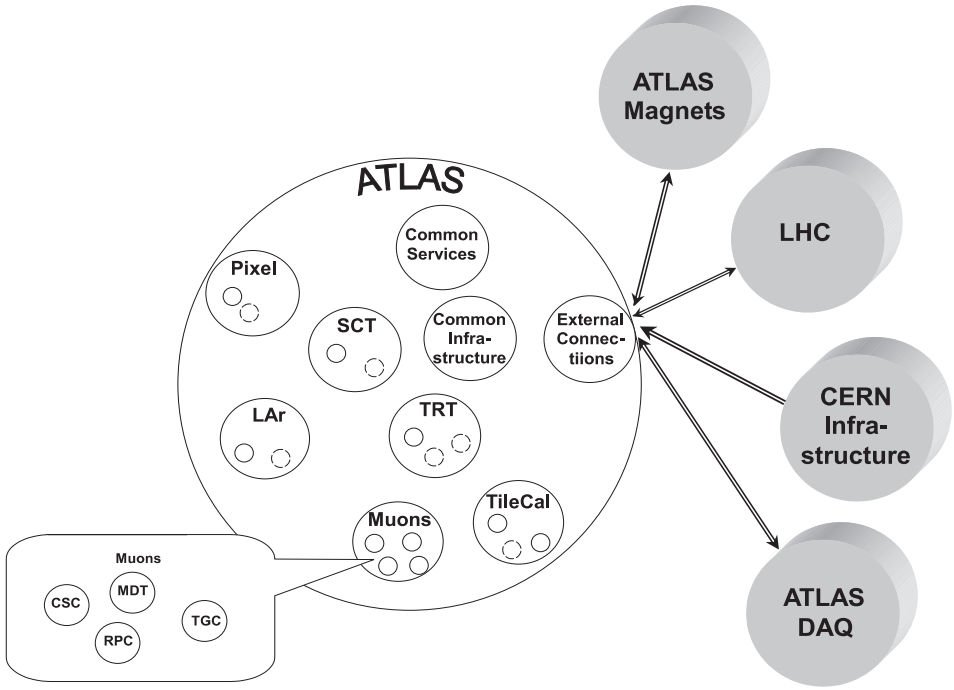

Figure 7-2 DCS logical structure.

- The *external connection* component is responsible for the communication with the external systems, such as the ATLAS DAQ system, the LHC machine, the CERN infrastructure, and the ATLAS superconducting magnets including their cryogenic system.

7.2.3 Software Components of DCS

The same types of software components should be used for all the different DCS partitions as shown together with the data-flow connections in Figure 7-3. They are the following:

- A distributed SCADA system.
- Control applications using the SCADA tools.
- Interfaces to the front-end.
- Interfaces to the external systems.
- Non-SCADA front-end control applications.

A *SCADA* system provides a wide range of facilities to develop and run a dedicated control application. The interfaces to the front-end hardware can be implemented either as native drivers in the SCADA to access the devices directly connected or using the client–server mechanism. In particular, the SCADA products have a set of embedded hardware drivers, the OPC client software to connect to either an industrial or a custom-made OPC server[1] and the Application Program Interface (API) library, allowing an external application to access the distributed run-time database. More features of SCADA are described in Section 7.3.

1. OPC is specified by the major manufacturers of hardware and software control components as a standardized interface defined for process-control applications, which is widely accepted in the control-system area.

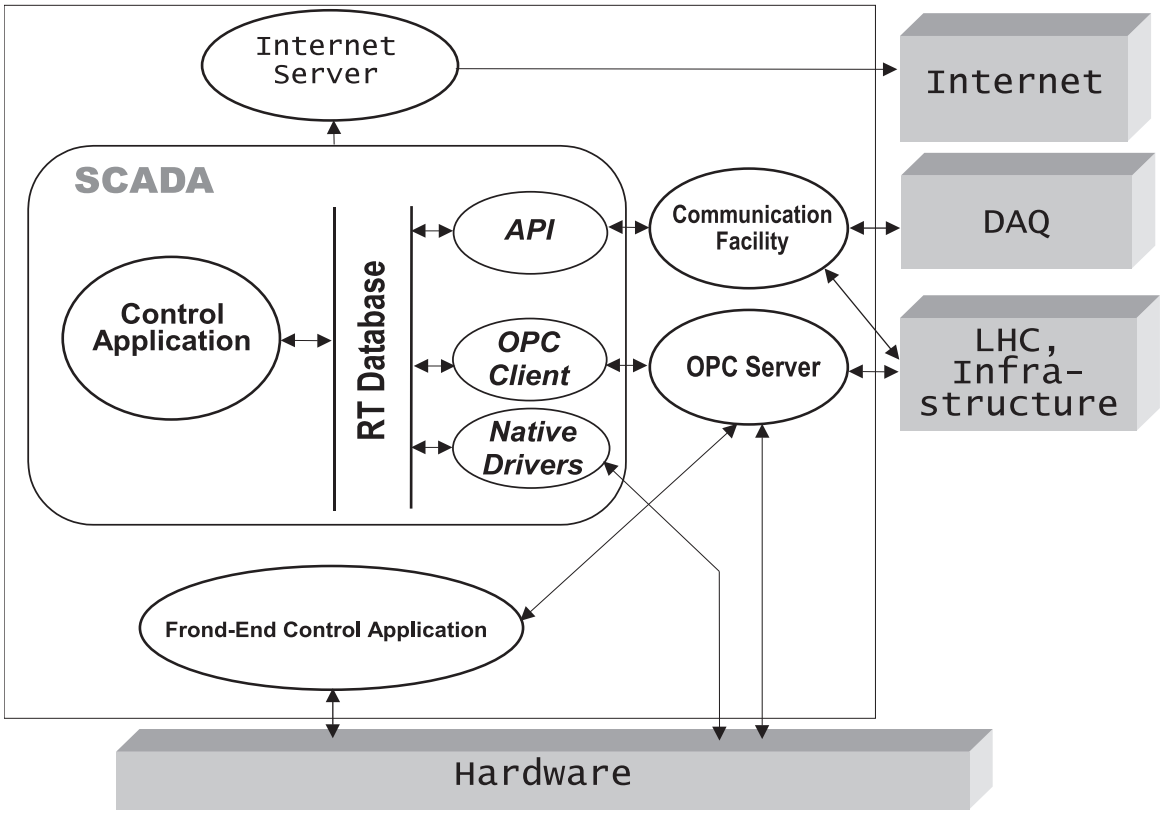

Figure 7-3 DCS software components.

The SCADA-based control application implements most of the functions necessary for the operation of the detector. Nevertheless, a certain set of control functions may be provided where appropriate by a front-end control application such as a VME-based system with an embedded computer, or a software emulation of a PLC. The connection of that front-end system to the SCADA should be established preferably with a dedicated OPC server.

The interfaces to the external systems shall be preferably developed as OPC servers or as custom-made communication facilities when it is reasonable to do so because of specific characteristics of the data exchange, as in the case of communication with the ATLAS DAQ system. These interfaces are specified in more details in Section 7.5 and Section 7.6.

Remote access to the DCS will be granted by an Internet server in accordance with the access rights of the different types of users.

7.2.4 Hardware Architecture

The DCS hardware consists of a wide variety of equipment, from simple front-end elements like sensors and actuators, up to complex computer systems. This equipment will be geographically distributed over three areas as shown in Figure 7-4:

- The experiment's cavern UX15, exposed to radiation and magnetic field.
- The underground electronics area USA15, always accessible to personnel.
- The main control room at the surface in building SCX1.

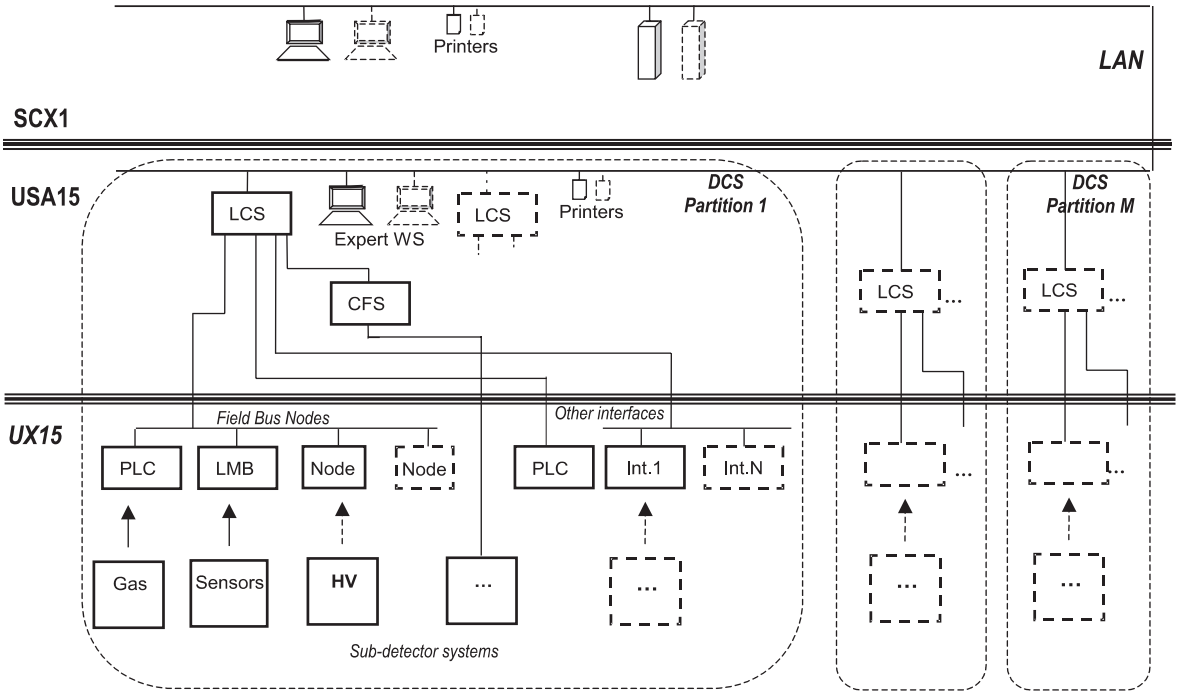

Figure 7-4 DCS hardware architecture and geographical distribution. The broken lines indicate that the corresponding items are repeated n-times.

The distribution underground is governed by two conflicting constraints. Because of the radiation level, the magnetic field and the inaccessibility at UX15 during beam time, it is advantageous to place equipment in USA15. However, complexity, cost and technical difficulties of cabling suggest condensing the data in UX15 and transfering only the results to USA15.

The equipment in UX15 must be radiation-hard or tolerant to levels of 1–10^5 Gy per year in the the muon subdetector and inner tracker, respectively. In addition, depending on the location, a magnetic field of up 1.5 T has to be tolerated. The data read out from sensors are collected by I/O concentrators which can be either embedded commercial fieldbus modules or general-purpose I/O nodes like the LMB, which will be discussed in more detail in Section 7.4. Some systems (e.g. gas) require the use of more complex devices like Programmable Logical Controllers (PLCs) that perform control loops at the front-end level. The data are transmitted by means of fieldbuses or dedicated links to the equipment in USA15, which consists of the following elements:

- Workstations, foreseen for subdetector experts for the supervision of individual partitions, mainly during commissioning and maintenance periods.
- Local Control Stations (LCSs) running the SCADA software and collecting data from the front-end devices in their partition. The LCS allows to run a partition either independently in standalone mode or integrated as part of the whole detector.
- Non-SCADA Complex Front-end Systems (CFSs), which are computer-based systems, dedicated to a specific task. An example is the alignment system of the muon subdetector, where a dedicated processor reads the images of CCD cameras and calculates the alignment constants of the individual chambers. The CFSs are normally connected to LCSs over a dedicated Local Area Network (LAN). CFSs can also be placed in UX15 if they support the more hostile environment.

The control room SCX1 houses the workstations for the operator and the central servers for database, external communications, global status of the detector, etc. These services are connected to the equipment in USA15 by means of a LAN. The central SCADA operator's consoles in SCX1 retrieve information from the LCSs of the different partitions and can be used to interact with them by means of commands or messages.

7.3 Supervisory System

SCADA systems are commercial software packages normally used for the supervision of industrial installations. They gather information from the hardware, process the data and present them to the operator. In general, the architecture of these products is centralized around the RT Event Manager. However, different components can be arranged in functional layers as shown in Figure 7-5.

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.

The Front-end Interface layer provides the communication with the hardware by means of dedicated drivers or communication standards such as OPC. Many SCADA products provide drivers for standard fieldbuses and PLCs. Some packages allow the data processing in the higher layers to be minimized by filtering the data at the Front-end Interface layer.

The Management layer handles the bi-directional transmission of the messages between the different modules. This layer is also responsible for managing RT data, archiving, alarm handling and administration of the user privileges.

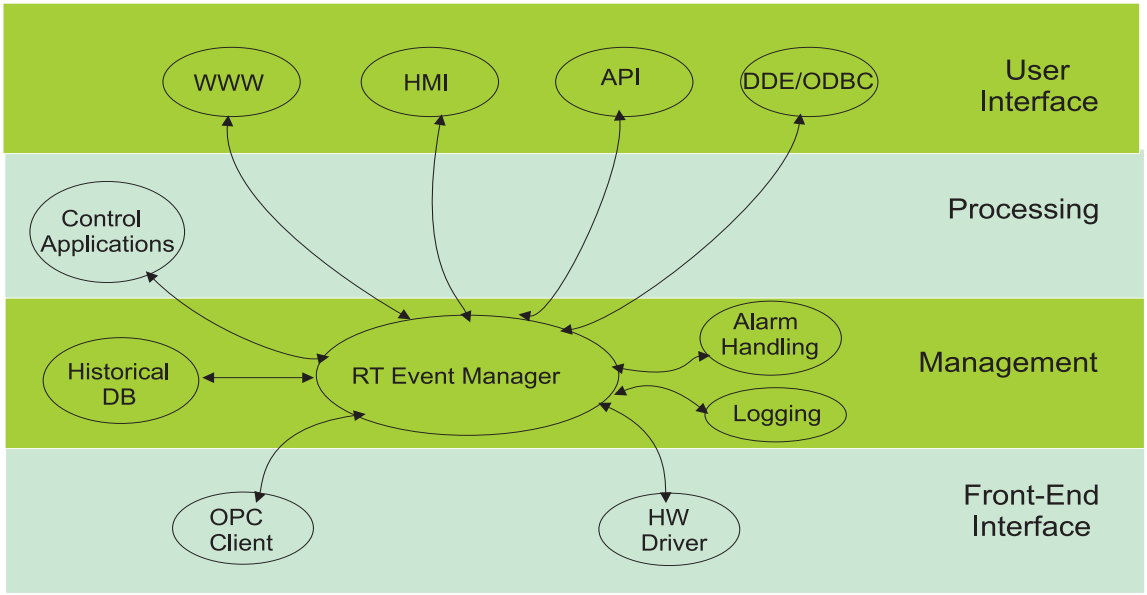

Figure 7-5 Architecture of a SCADA system.

The Processing layer contains control applications which are written within the SCADA system using its own programming language. They can be of two types: the first implements proce-

dures dedicated to monitoring and controling the detector; the second consists of specialized programs which extend the SCADA functionality, such as Finite State Machine (FSM) or additional reporting features.

The user interface layer takes care of external interactions. All SCADA products provide a powerful Human–Machine Interface (HMI) easily customizable by means of graphical objects that can be linked to the process variables in the application. In many cases the user interface is configurable through parameter templates, e.g. alarms or object libraries. Some SCADA systems include WWW servers for remote access to the application. External programs communicate with the Management layer via the API, allowing external applications to subscribe to any RT data. SCADA packages running in a PC / Windows environment usually provide Windows-standards such as Dynamic Data Exchange (DDE) and Open DataBase Connectivity (ODBC), to import / export data to applications like Excel and external databases.

The following example illustrates the internal operation of a SCADA system:

- A peripheral device sends a data telegram with information on a value change.
- After processing this telegram, the driver converts it into a *SCADA event*.
- The event manager writes the new value to the corresponding variable and sends the updated value to the appropriate components:
    - Control applications perform predefined calculations with the new variable: comparison with old value, thresholds, set-points, etc.
    - The database archives the value change.
    - The HMI displays the result value and status.
- If a threshold is crossed, the appropriate *event* is created and sent to the event manager.
- The event manager processes this *event*, i.e. stores it in the database and calls a hardware driver if necessary.
- The driver interprets the event manager's command and acts on the hardware.

The main results of the evaluation of SCADA products carried out in the frame of the JCOP project were the following:

- Device-oriented products, which allow complex data structures to be built to model hardware equipment, fulfil better the DCS requirements than simple I / O point-based SCADA systems, as they allow a better modelling of the natural partitioning and hierarchy of the detectors.
- Owing to the large number of parameters to supervise, the internal communication mechanism must be entirely event-driven. Systems which poll data values and status at fixed intervals present too big an overhead and have too long reaction times.

Suitable commercial SCADA systems have been identified. At present the technical specifications for a call for tender are being written.

7.4 Front-End I/O

The front-end I / O system is geographically distributed over the experimental cavern UX15 and the underground electronics room USA15. LANs and fieldbuses make the interconnections be-

tween the different equipment and I/O points and the SCADA stations. The LAN is used more for complex and higher level devices, and the fieldbuses more for supervision of sensors and actuators.

All equipment in UX15 is inaccessible during beam time and has to tolerate radiation and magnetic field. The levels depend on the exact position. We consider here only the volume outside of the calorimeter, where the levels do not normally exceed 10^{11} neutrons/cm^2 and 1 Gy per year. Special requirements also result from the restricted accessibility. Continuous functional tests have to be carried out and, in case of a failure, selective reset and initialization has to be possible. Remote diagnostics tools are essential.

Two classes of front-end system can be defined. The first is a general-purpose system, which is configurable to be used for many subdetector applications. The other consists of dedicated designs for specific applications.

7.4.1 General-Purpose I/O System

We have developed a general-purpose I/O system called Local Monitor Box (LMB). Apart from the special requirements listed above the reasons not to employ an industrial solution are the required high density of channels and the price, as we need several tens of thousands of channels. Nevertheless, in using the CAN fieldbus and the CANopen software protocol, we follow industrial standards as closely as possible. The intention is to use a standard solution for the connection to the SCADA system.

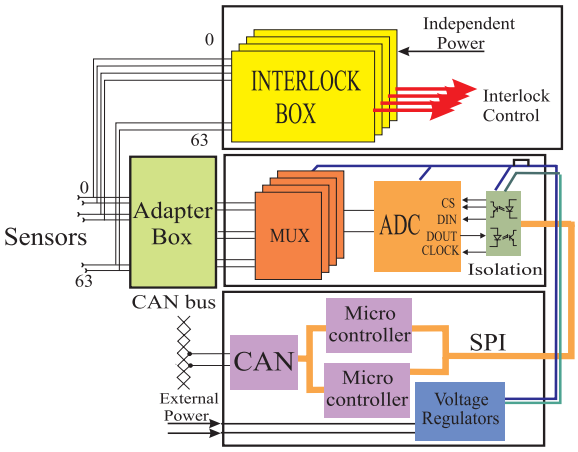

Figure 7-6 LMB schematic diagram.

A schematic diagram of the LMB is shown in Figure 7-6. At present it comprises the micro-controller, a multiplexed ADC (16-bit resolution, 7-bit gain range) and, as add-on, a hardware interlock system. The most demanding application in this area is the high-precision temperature measurements of the Liquid-Argon subdetector. In this application a resolution of 1 mK and an absolute precision of 3 mK have been achieved. This corresponds to a relative error of 4×10^{-5}, which is a factor of three better than required. Also the long-term stability, observed as 50 ppm over one month, is excellent. Irradiation tests have revealed some sensitive components (e.g. optocouplers), which have, in the meantime, been replaced by radiation-tolerant ones. The LMB has been operated in a magnetic field of 1 T and no effect has been observed. We conclude from these and other results [7-3] that the design is sound. Based on this experience, we have now started an implementation which best suits the various subdetector needs.

7.4.2 Subdetector-Specific I/O Systems

For applications which are embedded in the subdetector electronics and have very special requirements, dedicated solutions have to be employed. There are several approaches, such as dedicated and specialized CAN nodes, PLCs and complex standalone systems.

An example of the first approach is the supervision of the electronics and of the high voltage distribution of the Tile Calorimeter, where a custom-built CAN node is implemented on the electronics boards inside the detector. Another route is investigated by the muon group [7-4] who use a commercial CAN processor module. This would be plugged onto a dedicated motherboard, which implements the different I/O functions needed. Another approach is a standalone computer based system such as the CFS in Figure 7-4.

7.5 Connection to External Systems

The term *External Systems* designates here systems which have their own control system with wich the DCS has to interact. The connection will be made via the network and it will support information exchange and, in some cases, also the sending and receiving of commands.

7.5.1 Technical Services

The status of the CERN infrastructure services like cooling, ventilation, and electricity distribution has to be known to the DCS. Problems in these systems may have consequences on the detector and the DCS may need to take appropriate actions. After the temporary stop of one of these services, the DCS has to prepare the detector for the restart.

Some services will consist of several components, of which some belong to the detector and others are external. Cryogenics is a typical example. It consists of the refrigerator, the transfer system and the distribution inside the detector. The refrigerator will be delivered from industry with its standalone control system built in. However the operational parameters of the detector as consumer are tightly connected with the operation of the producer, the refrigerator. Therefore even slow feedback loops might need to be implemented with external systems.

The toroid magnets and the solenoid will also have their own process controls, which will be implemented with PLCs. The operator will not need direct control of them. However detailed online status and knowledge of all important parameters are essential for the operation of the detector and for the physics analysis.

Radiation monitoring is an area where information from many sources will be used. The subdetectors themselves are sensitive radiation probes, but also dedicated sensors and information from the monitoring of the environment will be used. The DCS will handle, present and log this data. This will also enable interlocks.

The safety system is a special case in so far as actions go only in one direction. The DCS must not be able to act on the safety system in order not to disturb its operation. On the other hand, the safety system must be able to command the DCS to take corrective actions, before it is forced to take more severe measures required for safety reasons. Detailed information exchange has to be done in both directions.

The information exchange with all external systems should use the same communication mechanism. We expect JCOP to address this question.

7.5.2 LHC Accelerator

Interaction with the LHC accelerator is required whenever it is ready to operate, regardless of the state of ATLAS. This is one main reason why this communication should be handled by the DCS on the ATLAS side and not by the DAQ system.

Information has to flow in both directions. ATLAS has to provide information about the beam parameters observed in the detector. In order to obtain detailed data, such as the different types of background, beam position, individual bunch luminosities, etc., information from different subdetectors, which is read out in some cases by the DAQ, has to be combined. This is normally only required during physics data-taking. During other periods, like machine development and equipment studies, only summary information about luminosity and background is needed, which the DCS has to provide independently from the DAQ system.

The LHC has dedicated instrumentation for the comprehensive measurement of all parameters of the accelerator. ATLAS will need a subset of this data for the operation and calibration of the detectors and for physics analysis. These data, which include parameters like total beam currents and possibly individual bunch currents, beam energy measurements, beam position and size, magnet and vacuum status around ATLAS, etc., will be made available to all subdetectors via the SCADA and will be stored in the ATLAS database.

Interlocks constitute another aspect of interaction between LHC and ATLAS. 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 a fast beam dump should the backgrounds become dangerous for the subdetectors. This important interaction has to be implemented by a fast interlock system.

All the information exchange should be done with the same mechanism as used for the communication with the other external systems, as described in the previous section.

7.6 Connection to the DAQ System

The ATLAS DAQ system and DCS perform different tasks, but have many aspects in common. The type of data to be treated gives guidance for defining the boundary between them. The DAQ system collects, transports, analyses and stores the physics event data, whereas the DCS, in general, takes care of the more slowly changing parameters of the detector, which are correlated with the physics events by a time stamp. Normally separate paths will be used for these different types of data. These data are assumed to be valid until a new value is entered in the database. This time window may extend over a whole DAQ run.

7.6.1 Requirements

The main requirements and the interconnections between the DAQ system and the DCS are described in detail in Ref. [7-5]. The DCS needs to be operational continuously in order to guarantee the supervision of the detector at all times, whereas the DAQ system needs to be operational only when physics and calibration data are taken. Experience from the LEP experiments shows that the periods in between will sometimes be used for further optimization and upgrade of the DAQ system. Therefore both systems have to operate independently and, in particular, the DCS must not rely on services provided by the DAQ system. Moreover, the monitoring of some of

the DCS functions will already be needed when the installation of the detector starts at the beginning of 2003, much earlier than the integration of the overall DAQ system. Even before this date some subdetector groups wish to use the DCS for quality assurance and calibration of equipment which will be pre-assembled on surface. The factorization into two independent systems must, however, not result in any limitation of the functions needed experiment-wide or of their performances. In particular seamless exchange of information and commands is required.

7.6.2 Connection to DAQ Components

The supervisory functions of the DAQ system will be implemented by using dedicated, purpose-built software tools and services, as described elsewhere in this document. The DCS, however, will be based on a commercial SCADA package, which comprises these components internally. A connection between these two different implementations of partly the same functions is needed. As a general rule, the DCS has to export all information which needs to be stored and accessed offline into the standard ATLAS environment. Using the SCADA API tool the connections are deemed necessary as described below.

The instantaneous values of parameters and status information are handled in SCADA in the RT Event Manager, and the corresponding component in the DAQ system is the Information Service. Between them a bi-directional interface is foreseen with an appropriate mechanism (e.g. publish / subscribe, filter) to limit the exchange to the necessary subset of data.

Asynchronous events and messages of general interest originating from the DCS have to be sent to the DAQ Message Reporting System and logged in the Book-keeper. Typical examples are failure of subdetector equipment or warnings and alarms of operational parameters.

The DAQ Run Control is based on a hierarchy of interconnected controllers each responsible for a well-defined element of the apparatus. The implementation will be in the form of Finite State Machines. One or more controllers will be dedicated to the DCS and will interchange commands and messages about state changes.

Both the DAQ system and the DCS will have their own HMI. As the physics data-taking will be driven by the Run Control, the information of the hierarchically highest level of the DCS has to be accessible via the DAQ HMI.

Databases are the most prominent external software components with which the DCS has to interact. The following data types have to be handled by the DCS:

- Configuration data:
    - Hardware set-up (electronics modules, crates, etc.).
    - Software (e.g. programs to load into fieldbus nodes).
    - Parameters (e.g. high-voltage settings).
    - Calibration constants (e.g. alignment constants).
- Command logging (operator and *automatic* commands).
- Incident logging (e.g. warnings, alarms).
- Storing results of measurements (e.g. currents, temperatures).

The master copy of all data sets, which need to be accessed from outside the DCS, will reside in the ATLAS-wide database. A local copy will be held inside the DCS in order not to rely on external connections during operation.

7.6.3 Interaction during Operation

In the following, a few examples for the interaction between the DCS and the DAQ system during normal operation are described. The interaction on the DCS side is performed at the SCADA level - no direct interaction is foreseen with the front-end I/O level. From the DAQ side the main point of interaction during physics data-taking will be the Run Control, with one or more controllers exchanging commands with the DCS. The starting of a physics run will be performed in several steps, both on the DAQ side and on the detector hardware side. This is assured by the interplay between the DAQ system and the DCS, the latter also synchronizing with the status of the LHC accelerator.

Calibration of the detector is a special case of operation and very different procedures will be used. Three main classes can be distinguished. The procedures can happen entirely inside the DAQ system, inside the DCS, or can involve both. We consider in this section only the last case, where two types exist, depending on whether the DAQ system or the DCS drives the procedure, i.e. whether the calibration happens during physics data-taking or without a DAQ run.

Some subdetectors will be calibrated during a physics run, e.g. by injecting special triggers. The data are transported on the main data-flow path and analysed by the DAQ system, which may, as a result, send commands to the DCS, e.g. to adjust the high voltage. In such a case the DAQ system is the master of the calibration procedure.

Other subdetector systems are calibrated by a DCS-driven procedure. For example, the DCS varies operational parameters like a gas composition. It may need results from the DAQ system, calculated from physics event-like data, to find the optimal set of parameters. In this case the DCS is the master.

In all cases described, the interaction between the DAQ system and the DCS happens between the DCS SCADA system and components of the Online Software of the DAQ system. These components are expected to extend the DAQ ROD crate, and hence enough functionality regarding exchange of commands and information is provided. Only in exceptional cases could it be envisaged to allow direct interaction between the DAQ front-end level and the DCS front-end system. In such a case special arrangements have to be foreseen to inform the supervisor levels in order to avoid possible conflicts.

7.7 Work Plan

The overall architecture of the DCS is now well established and has been verified with prototypes. The work to do until the Technical Design Report consists in choosing final components like the commercial SCADA product; designing the connections to the external systems; defining and prototyping the interaction with the DAQ system; agree with the subdetectors about their interfaces to the DCS and make a close-to-final implementation of the general-purpose part of the front-end system. In the following we concentrate on the work to be done internal to the DCS, the work related to the DCS–DAQ connection is listed in Section 10.3.4.

7.7.1 Responsibilities

The general approach is to build the DCS out of components which are selected and supported in the frame of JCOP. It has to be defined up to which level common solutions are possible and sensible for all four LHC experiments. This has consequences on the attribution of responsibilities which the different JCOP members have to assume. It is natural that the CERN controls group should be responsible for the common part and the experiment's DCS teams for the detector-specific part.

The SCADA component is clearly common and includes both organizational aspects (licences, distribution, training, etc.) and technical aspects. The latter comprise the provision of standardized connection to the front-end system and of additional software components, which are not included in the SCADA system (e.g. FSM, expert system). Common applications like interfacing to CERN-wide services, the LHC accelerator and also experiment-related services like gas systems should be done in common as well. The interfacing to the individual DAQ systems is clearly different for each case and hence is the responsibility of each experiment.

The front-end system is in general experiment-specific and hence is the responsibility of each experiment. This does not exclude that a solution which is implemented and supported by one experiment be also employed by another. It is up to the subdetectors to define and implement their process-control procedures and to interface them to the DCS, using the tools centrally provided.

7.7.2 SCADA

The final selection and the procedure of purchase of the SCADA system have to be done. The connection to the front-end system has to be developed in a robust and fault-tolerant way, both in terms of hardware drivers (e.g. for CANbus) and of software interfaces like OPC. Generic templates for the common SCADA services such as data presentation, incident handling, and database access have to be established, which will be used by the subdetector groups.

7.7.3 The Front-End System

The choice of the front-end system is the task of the subdetector groups. In order to harmonise this area in the best possible way and to use a common solution, the LMB has been developed. The principle has been proven and now the details of the implementation, such as types of functions, number of channels, packaging, etc., have to be defined and implemented and have to follow the ATLAS rules concerning radiation tolerance. Apart from normal operation in a radiation environment, the susceptibility to *single event upsets* has to be investigated. Software for the LMB, which detects and corrects automatically error conditions, radiation-induced ones included, has to be developed. For subdetector-specific front-end systems, the interface points and protocols to the central DCS have to be defined.

7.7.4 Prototype Applications, Test Beams

Applications together with subdetector groups will be done and will serve several purposes:

- Test and verification of components of the DCS.

- Training of subdetector controls personnel.
- Standalone systems for acceptance tests for subdetector components.
- Final calibration of detector elements.

These systems will monitor and control the subdetector hardware and also connect to the DAQ system in test-beam operation.

7.8 References

7-1 H. J. Burckhart, *The ATLAS DCS user requirements document*, DCS–URD1, http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/urd951123.ps

7-2 *Joint controls project (JCOP)*, http://itowww.cern.ch/jcop

7-3 B. Hallgren et al., *A low cost I/O concentrator using the CAN fieldbus*, Proc. of the ICALEPS 99 conference, Trieste, Italy, 4–8 October 1999, see also http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/lmb.html

7-4 P. Mockett and M. Twomey, *A low cost CAN node for A/D measurements in ATLAS*, Proc. of the Fifth Workshop on Electronics for the LHC Experiments, September 1999, CERN 99–09 and CERN / LHC / 99–33

7-5 H.J. Burckhart, M. Caprini, R. Jones, *Connection DCS–DAQ in ATLAS*, DCS–IWN 8, November 1999, http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/dcs_daq_0.6.pdf

8 Physics and Event Selection Strategy

8.1 Definition of the Strategy

The strategy for physics and event selection should provide an implementation-independent scheme that maximizes the potential of ATLAS for physics discovery and precision measurements. In this context, the High-Level Triggers (HLTs) are seen as a single logical unit. An implementation will determine how to make best use of the distinguishing features of the LVL2 and Event Filter (EF) systems. Flexibility is needed to adapt to changes in the luminosity (even during a fill of the LHC), variations in the background conditions, and new requirements derived from physics and from improved understanding of the detector. Selections in the HLT stem from the identification by the LVL1 trigger of Regions of Interest (RoIs) containing candidates for electron, photon, tau, jet and muon objects: essential ingredients to form the physics analysis schemes. Primary and secondary RoIs[1] can be distinguished: Most of the rejection is achieved after processing only the primary ones. This reduces the average number of RoIs to be inspected to less than two per event.

The physics and event selection strategy proposed in this chapter is composed of four elements related to classification, selection, sequences and algorithms. In the following, a brief description of these four elements is given; more details and results of studies can be found in subsequent sections. Algorithms are used to identify objects (such as electrons or jets) together with their properties, or to determine global features of the event. The sequence defines the order of execution of the algorithms (e.g. according to complexity) and aims to reduce the size and cost of the HLT system, while maintaining the flexibility needed and maximizing the physics potential. The selection is the procedure which, using the objects with their properties and a set of criteria, decides whether the event is to be rejected or not. Classification does not reject any event; it groups the events into categories to help the subsequent physics analyses.

The objects used for the classification and for the selection are based on physics process signatures, as presented in Section 2.3.1 and described in more detail in Ref. [8-1]. In order to keep the selection as unbiased as possible towards new physics processes, only inclusive selections using combinations of very few objects are used, wherever feasible.

8.2 Classification Scheme

The classification scheme summarizes the signatures used to classify the accepted events for subsequent physics analyses. The schemes [8-1], which are different at low luminosity (10^{33} cm^{-2} s^{-1}) and design luminosity (10^{34} cm^{-2} s^{-1}), have been derived from the analysis-level selection criteria for the various physics channels, as documented in Ref. [8-2].

In Table 8-1, an example from a classification scheme is given for the case of the Standard Model Higgs boson at low luminosity. For each classification criterion, the physics channel covered is given. As can be seen, at most three different kinds of objects (with variable multiplicity) are used for the classification. The properties associated with the objects are their transverse momentum and in some cases an isolation requirement. The criteria always contain high-p_T objects

1. Primary RoIs are RoIs that contributed to the LVL1 selection; secondary RoIs are additional regions flagged by LVL1.

Table 8-1 Classification scheme for the Standard Model Higgs boson searches (at low luminosity). The notation of a lepton ('*l*') refers always to only electrons and muons (*l* = e, μ). The number given for each object refers to the nominal p_T threshold, the term 'i' indicates an isolation requirement.

Signatures	Related physics channel
γ40i + γ25i	$H \rightarrow \gamma\gamma$
2γ25i + *l*25	associated H with $H \rightarrow \gamma\gamma$
γ60i + γ40i	H + jet with $H \rightarrow \gamma\gamma$
*l*20i + 2b15	$t\bar{t}$ H, ZH and WH with $H \rightarrow b\bar{b}$
2*l*20i + 2*l*7i	$H \rightarrow ZZ^{(*)} \rightarrow 4l$
*l*20i + *l*10i + E_T^{miss}40	$H \rightarrow WW^{(*)} \rightarrow 2l\,2\nu$
2*l*40i + E_T^{miss}50	$H \rightarrow ZZ \rightarrow ll\nu\nu$
*l*50i + E_T^{miss}50	qqH with $H \rightarrow WW$

(with transverse momenta typically above 20 GeV); however, additional objects with lower transverse momenta are used in some cases. The values indicated for the transverse momentum

Table 8-2 Classification scheme for B-physics (at low luminosity).

Signatures	Related physics channel
μ6 + μ3 + $m_{J/\psi}$ +2h0.5 + m_{K^0}	$B_d \rightarrow J/\psi K^0_s(\pi\pi)$ for sin 2β
μ6 + 2e0.5 + $m_{J/\psi}$ +2h0.5 + m_{K^0}	$B_d \rightarrow J/\psi K^0_s(\pi\pi)$ for sin 2β
μ6 + μ3 + $m_{J/\psi}$ +h1.5	control channel for sin 2β
μ6 + 2e0.5 + $m_{J/\psi}$ +h1.5	control channel for sin 2β
μ6 + μ3 + $m_{J/\psi}$ +2h0.5 + $m_{K^{0*}}$	control channel for sin 2β
μ6 + 2e0.5 + $m_{J/\psi}$ +2h0.5 + $m_{K^{0*}}$	control channel for sin 2β
μ6 + 2h4 + m_{B_d}	$B_d \rightarrow \pi^+\pi^-$ (sin 2α)
μ6 + μ3 + $m_{J/\psi}$ +2h0.5 + m_ϕ	$B_s \rightarrow J/\psi\,\phi$
μ6 + 2h1.5 + m_ϕ + h1.5 + m_{D_s}	$B_s \rightarrow D_s(\phi(KK)\pi)\pi$
μ6 + 2h1.5 + m_ϕ + h1.5 + m_{D_s} + 3h0.5 + m_{ρ^0} + m_{a_1}	$B_s \rightarrow D_s(\phi(KK)\pi)a_1(\rho^0\pi)$
μ6 + 2h3 + $m_{K^{0*}}$ + h3 + m_{D_s} + 3h0.5 + m_{ρ^0} + m_{a_1}	$B_s \rightarrow D_s(K^{0*}\,K)\,a_1(\rho^0\pi)$
2μ6i	rare decays: $B_{d,s} \rightarrow \mu\mu$, $B_{d,s} \rightarrow \mu\mu K^{0*}$, $B_{d,s} \rightarrow \mu\mu\phi$ and $B_{d,s} \rightarrow \mu\mu\rho^0$

are in most cases the selection cuts foreseen for the offline analysis. To allow studies of the background and to provide more flexibility to the analysis, lower thresholds will be chosen for the trigger selection wherever possible and appropriate. Where the system requirements will not allow this to be done for all events passing the cuts, prescaled samples will be accumulated for the lower thresholds.

For the case of B-physics (at low luminosity), the main classifications are listed in Table 8-2. Besides the identified objects such as leptons (which have much lower transverse-momentum thresholds compared to the previous example), additional objects ('h') representing charged tracks appear. Based on these tracks and the leptons, invariant masses are calculated, upon which cuts are placed to identify specific particle decays.

The full list of the classification schemes (for the currently anticipated physics coverage of ATLAS, and including those for design luminosity) can be found in Ref. [8-1]. The classification does not reject any events; the event is only categorized according to the signatures listed in order to facilitate subsequent analysis.

8.3 Selection Scheme

Based on the classification schemes described in the previous section, selection schemes are 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. Where inclusive selection cannot be maintained to the end of the HLT, because of constraints coming from the total output data flow, one way of reducing the rate is to use secondary RoIs in the selection. This adds a first level of non-inclusive selection, increasing the flexibility of the system, but of course imposes some limitations on the overall physics capability. 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 the EF (e.g. to select $Z \rightarrow l^+l^-$ decays exclusively).

8.4 Selection Algorithms and Selection Sequence

In this section, the performance of the selection algorithms for various final-state signatures is summarized. Two aspects of the performance are distinguished: the physics performance of an algorithm (i.e. the signal efficiency and the background rejection) and the system performance (e.g. the execution time, the amount of data needed). These two aspects must be optimized together for the full selection chain to reach the best possible performance at affordable cost. The sequence in which algorithms are executed (and the related selection sequence) also influences the system performance.

In the following subsections, detailed studies on the physics and system performance of algorithms for LVL2 and EF are summarized for five final-state classes: electrons and photons; muons; jets, taus and missing E_T; b-jets; and B-physics. This classification reflects the organization of studies on the detector performance, as documented in Ref. [8-2]. Emphasis is put on highlighting the interplay between physics and system performance and the flexible boundary between LVL2 and EF.

It should be pointed out that the studies presented in this section represent the present understanding of the trigger selection which will evolve further. A complete assessment of the correlation between the various selection stages in the chain LVL1–LVL2–EF–offline, including a complete assessment of trigger biases, has not yet been performed. The modularity of the selection scheme, as discussed below, will allow the implementation of different parts of the selections at different stages of the HLT. Only the availability of real data will allow one to make the final judgement on this sharing of algorithms.

It should be noted that the studies documented in this section build on earlier work reported in Ref. [8-3].

8.4.1 Selection of Electrons and Photons

In the present view of the ATLAS trigger menus [8-2], 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 ($W \rightarrow e\nu$, $Z \rightarrow ee$, 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 [8-4], the LVL2 trigger performs a selection of isolated e.m. clusters [8-5], [8-6] using the full calorimeter granularity and detailed calibration. 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 [8-7].

Electrons are identified at LVL2 [8-8] 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 [8-9] and SCT / Pixel ('Precision') detectors [8-10] in the region identified by the LVL1 RoI. A histogramming method is used to find an initial set of track candidates; the best track candidate is then selected on the basis of a fit. Other tracking algorithms [8-11] have been considered and are under study.

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. The present study therefore uses the available ATLAS offline reconstruction software [8-2] as a prototype of the future EF code. 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 [8-12]. 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.

8.4.1.1 HLT Electron/Photon 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. $H \rightarrow \gamma\gamma$, $Z \rightarrow ee$, $H \rightarrow 4e$). 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 and without pile-up (for more details see Ref. [8-13]). 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 and photon triggers is documented elsewhere for both the LVL2 trigger [8-7], [8-8] and the EF [8-12], [8-14] algorithms.

Table 8-3 Performance of the isolated electron/photon HLT trigger at design and low luminosity for the single electron and the single photon selections. 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 (20) GeV and single photons of p_T = 60 (40) 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 94.6% (92.6%) and a LVL1 rate for e.m. clusters of 21.7 kHz (5.8 kHz). The timing results quoted here are for events from the di-jet sample and are scaled to correspond to a 500 MHz Pentium II machine running Linux. The terms m_{50} and m_{95} are defined in Section 8.4.1.2. 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}**
Electrons	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	--
Photons	L2 Calo	250 ± 43	97.7 ± 0.4	0.20 / 0.26 ms	83 ± 8	94.0 ± 0.3	0.2 / 0.26 ms
	EF Calo	144 ± 33	87.3 ± 0.9	0.39 / 0.63 s	57 ± 7	84.7 ± 0.6	0.34 / 0.56 s
	EF ID	114 ± 43	82.9 ± 0.9	19 / 106 s	51 ± 7	81.2 ± 0.6	0.44 / 1.2 s

The performance of the single isolated electron / photon HLT algorithm is summarized in Table 8-3 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 460 Hz (140 Hz) at design (low) luminosity for a nominal p_T threshold of 30 GeV (20 GeV). The overall reduction in rate achieved by LVL2 is a factor of 47 (41) for a loss of efficiency of 14.7% (11.9%) with respect to LVL1. The additional rate reduction provided by the EF amounts to a factor of 3.9 (3.4) for a relative efficiency loss of 7.7% (7.3%). The LVL2 selection has an efficiency of 91% (96%) 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).

At low luminosity, the events remaining after the HLT electron selection consist of $W \rightarrow e\nu$ decays (19 ± 6)%, isolated electrons from $(b,c) \rightarrow eX$ decays (42 ± 7)% and background from high-p_T photon conversions and misidentified hadrons (39 ± 7)% [8-15]. At design luminosity, where a higher p_T threshold is applied, the corresponding proportions are (40 ± 13)%, (13 ± 9)% and (47 ± 13)%. The quoted errors are the statistical uncertainties on the estimates.

Electron decays of the W are selected by the EF with an efficiency of (90 ± 9)% at low luminosity and (75 ± 15)% at design luminosity, in agreement with the values given in Table 8-3 for single electrons of 20 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 $H(130) \rightarrow 4e$ is $(97.7 \pm 0.3)\%$ with respect to the LVL1 efficiency of 98.8% at low luminosity; these high efficiencies are due to the large electron multiplicity in the final state.

The photon trigger is based mostly on a calorimeter selection at the LVL2 and EF levels and, in addition, on the reconstruction and identification of photon conversions in the EF [8-12]. The nominal thresholds used for the single-photon trigger at LVL2, 40 GeV (60 GeV) at low (design) luminosity, are much higher than the single-electron ones. This is because track information is not used in the LVL2 trigger in order to keep good efficiency for converted photons. Table 8-3 shows that the input rate from the LVL2 photon trigger to the EF is 250 Hz (83 Hz) at design (low) luminosity and that the EF provides an additional reduction factor of about two for a relative efficiency loss of 15%. In the EF, photon conversions are reconstructed. Calorimeter clusters are rejected if they have a matching charged track not associated with a conversion. This leads to a reduction in the background rate for the photon trigger by only 15–20%.

After the EF photon selection, 33% (43%) of the remaining events [8-12] at design (low) luminosity contain a direct photon, while the rest are background jet events. The corresponding fraction of direct photons, when the conversion identification is not done, is 23% (35%). The loss of events at LVL2 with respect to the EF selection is found to be very small. For design (low) luminosity, about 0.3% (0.06%) of events selected by the EF alone are not accepted with the LVL2 criteria. As an example of physics performance, the HLT selection efficiency for $H \rightarrow \gamma\gamma$ is 97.9% (97.8%) with respect to the LVL1 efficiency at design (low) luminosity, including both the single- and the double-photon trigger.

8.4.1.2 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].

In order to understand where the computing resources are being used in the trigger, studies of algorithm performance have been made for different algorithm parameters. Here the aim is to eliminate any resource-consuming tasks that contribute only marginally to the rejection. As an example, Figure 8-1 shows the LVL2 trigger efficiency as a function of rate [8-5] for different levels of zero-suppression in the calorimeter [8-17]. As the calorimeter cell thresholds are increased and the number of cells considered by the LVL2 trigger is reduced, the execution time is also reduced. Figure 8-1 shows that the execution time can be reduced by a factor of four with a marginal increase in the LVL2 rate of less than 20% (for efficiencies above 90%).

Similar studies have been performed [8-12] for the EF. As an example, Figure 8-2 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

1. The timing measurements were carried out on several different platforms, but have been converted to the same overall scale, corresponding to a 500 MHz Pentium II equivalent.

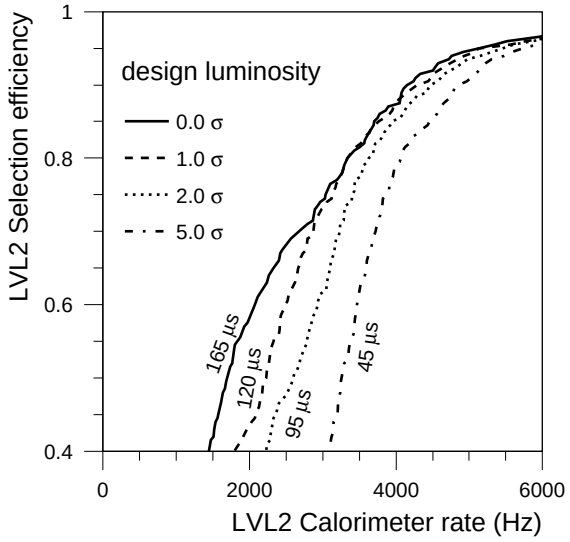

Figure 8-1 Performance of the LVL2 electron trigger calorimeter algorithm as a function of the calorimeter cell threshold (in units of noise level σ). The algorithm execution time corresponding to each threshold is indicated on the curves. The times are given for a 500 MHz Pentium II equivalent.

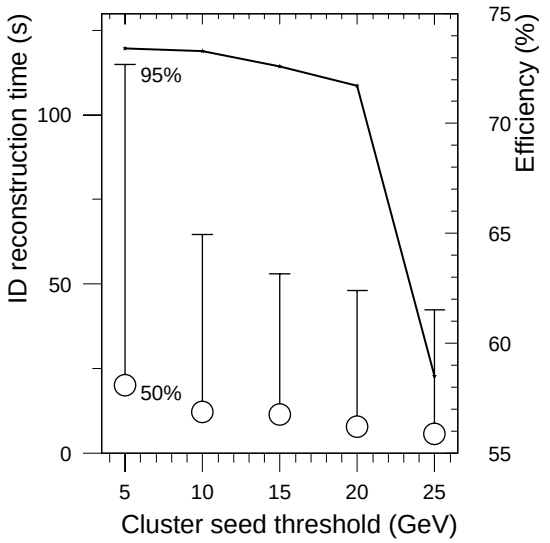

Figure 8-2 EF Inner Detector reconstruction time (`xKalman` [8-16]) as a function of the calorimeter seed threshold at design luminosity. The efficiency corresponding to each threshold is also shown. Times are expressed for m_{50} and m_{95} (see text) and are given for a 500 MHz Pentium II equivalent.

number of seeds, reduces the execution time (in particular m_{95}) with a negligible impact on the physics performance. Further studies have to be done to quantify possible trigger biases due to such optimizations.

The present system performance of the electron / photon algorithms can be improved at all levels of the HLT. Reducing the RoI area to the minimum needed for feature extraction [8-17] decreases the number of cells that the LVL2 calorimeter trigger has to consider and, in turn, the overall algorithm latency. A confirmation of the LVL1 isolation can further reduce the rate into the EF [8-18], at the expense of increasing the amount of data to be transferred to LVL2. An alternative look-up table (LUT) based algorithm for the LVL2 TRT tracking [8-11], although not yet fully optimized for physics performance, gives at design luminosity a significantly reduced latency of (m_{50}, m_{95}) = (3.4 ms, 6.2 ms): considerably less than for the algorithm of Ref. [8-9] (see Table 8-3 for comparison).

The performance of the EF algorithms can also be optimized by reducing the amount of data to be processed [8-12]. Reducing the size of the road for the track search to a smaller region (similar to the one used in offline studies [8-2]) gives a reduction factor of four on the reconstruction time at design luminosity. A similar improvement is obtained by increasing the p_T threshold of the tracking reconstruction. When taken together, all these improvements reduce the EF electron tracking reconstruction time (as shown in Table 8-3) by a factor of ten. First studies of the selection efficiency have shown that no significant losses are induced by this optimization; however, the full impact on physics performance (e.g. bremsstrahlung recovery) has still to be evaluated.

Studies [8-12] with an alternative tracking algorithm (`iPatRec` [8-19]) also show reduced execution times for a comparable physics performance (compared to that described above and obtained using `xKalman` [8-16]).

Based on these studies, an affordable implementation (in terms of computing) for the electron and photon HLT seems reachable, especially for LVL2. As shown in Section 6.6, an implementation of the LVL2 electron trigger algorithms on available processors already fulfils today the latency requirements of LVL2.

8.4.1.3 HLT Strategy and the LVL2–EF Boundary

The use of system resources in the electron / photon 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. As an example, at design luminosity, the EF total electron / photon trigger execution time is reduced by a factor of two [8-15] by rejecting events immediately after the calorimeter reconstruction. Similar gains are possible at LVL2 [8-17].

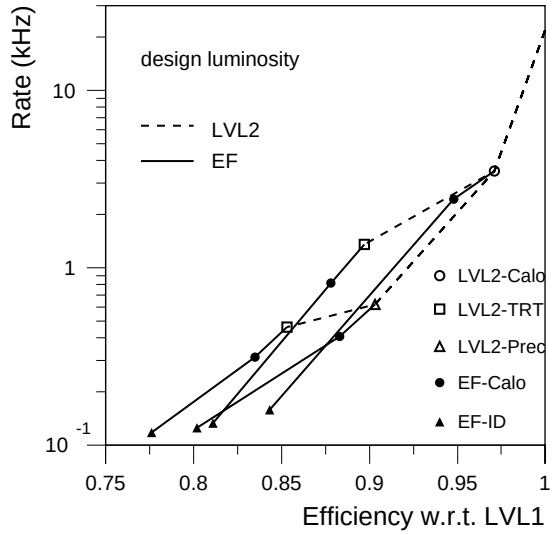

Figure 8-3 Different rate reduction *paths* in the LVL2–EF selection chain for electrons at design luminosity as a function of the HLT steps. The efficiencies are given with respect to LVL1.

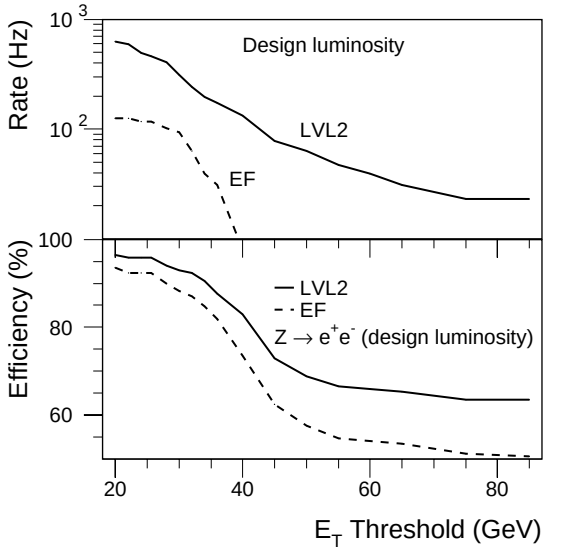

Figure 8-4 Total rate at LVL2 and EF for the single- and double-electron trigger (standard nominal thresholds e30i and 2e20i, upper plot) and efficiency of this trigger at LVL2/EF for $Z \rightarrow ee$ events (lower plot) as a function of the E_T threshold for the single-electron trigger.

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. As an example, Figure 8-3 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 [8-15].

Since electrons and photons are localized objects with similar reconstruction paths at LVL2 and the EF, it is natural to extend the RoI-based approach of LVL2 to the EF. The LVL2 trigger already provides detailed information (including refined position and energy measurements) about the e.m. cluster. This information can be used to *seed* the EF processing, thus freeing EF resources for other tasks. For example, at design luminosity, this approach [8-15] reduces the median processing time (m_{50}) of LVL2 accepted events in the EF by a factor of three; at the same time the value of m_{95} is reduced from 115 s to 20 s.

At design luminosity, the LVL1 rate is dominated by the contribution from single high-p_T e.m. objects. Further reduction of the HLT output rate could be achieved by applying an exclusive selection, or by further improving the rejection power using additional selection criteria, or by increasing the E_T threshold. All of these will imply an additional loss in efficiency for physics signals. Figure 8-4 shows a case study of 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. The lower plot shows the impact on another physics signal, the $Z \rightarrow e^+e^-$ decay: for thresholds below 35 GeV, the efficiency for Zs is only slightly reduced. Decays with more than two electrons are affected even less, e.g. in the case of $H(130) \rightarrow 4e$. If an additional rate reduction is required, further studies are needed to choose, from the different options described, the one which least affects physics performance and selection efficiency.

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.

8.4.2 Selection of Muons

8.4.2.1 Introduction

The purpose of the high-level muon trigger is the identification of muon tracks in RoIs indicated by the LVL1 trigger, and the accurate measurement of their position and transverse momentum. LVL2 and the EF must reject LVL1 muon candidates arising from the cavern background, non-prompt muons (such as those from decays of K and π mesons), and muons of p_T below the required threshold.

Whilst the LVL1 trigger uses only hits in the dedicated muon trigger chambers (RPCs and TGCs), the LVL2 trigger has to access the full data of the Muon Spectrometer, including the high-precision chambers of the Monitored Drift Tubes (MDTs). The high background environment in the MDT chambers demands algorithms capable of rejecting background hits due to chamber activity accompanying the muon track and to the cavern background.

The tracks found in the LVL2 muon trigger are extrapolated for combination with Inner Detector and Calorimeter information. Matching between muon tracks measured independently in the Muon Spectrometer and in the Inner Detector validates the prompt muon selection whilst reducing the contamination of secondary muons from π and K decays. This is especially important for the LVL2 B-physics trigger in low-luminosity running, for which the selection of prompt low-p_T muon events defines the input sample to the full scan of the Inner Detector (see Section 8.4.5).

The EF will have access to all the data of the event and will be able to run offline algorithms to reconstruct muons in the ATLAS detector. The use of such algorithms for the EF has not been studied in detail, but EF code could be developed starting from the current analysis tools [8-2], [8-16], [8-20], or the LVL2 algorithms discussed in Section 8.4.2.2.

The LVL1 trigger rates for the 20 GeV high-p_T threshold have been studied further recently and found to be a factor ~ 2 larger than those quoted in Ref. [8-4]. This change is largely due to a higher statistics study of the acceptance to low-p_T muons. In the following, these updated rates will be used for comparisons with LVL1.

8.4.2.2 LVL2 Muon Standalone Trigger Algorithms

Two independent algorithms (μFAST and BMC_TRIG) exist to perform LVL2 feature extraction in the muon spectrometer. Both algorithms aim to satisfy the demands described above, but do so using somewhat different approaches.

The μFAST algorithm has been designed expressly for the online environment in which the LVL2 code will run. The algorithm is steered by the RoI given by the LVL1 trigger and uses both trigger-chamber and MDT measurements. The feature-extraction procedure of the μFAST algorithm described here is confined at present to the barrel region of the detector, $|\eta| < 1$. It is performed in three sequential steps [8-21]:

- Pattern recognition is performed using information from the muon trigger chambers (RPCs) to define a road in the MDT chambers around the muon trajectory. MDTs lying within the road are selected and a contiguity algorithm is applied to remove background hits not associated with the muon trajectory.
- A straight-line track fit is made to the selected hit tubes within each MDT station. For this procedure the drift-time measurements and a linear time–distance relation are used; the track sagitta is then evaluated.
- A fast p_T estimate is made using an LUT. The LUT encodes the linear relationship between the measured muon sagitta and Q/p_T, as a function of bins in η and ϕ.

The output of the algorithm is a measurement of η, ϕ, the charge sign and an estimate of the muon p_T.

The second approach is based on track finding derived from an offline analysis algorithm, optimized for the LVL2 online environment, using a basic strategy similar to that presented above. The BMC_TRIG algorithm operates over the region $|\eta| < 2$ and is described in Ref. [8-22]. The main differences with respect to the μFAST algorithm are:

- No contiguity demand is made prior to the local track finding in the MDT stations: all possible tracks are formed using the MDT hits available within the defined roads; the tracks are evaluated using a quality factor; the track with the best quality factor is then selected.
- The track is fitted to a circle (or to a circle and a line tangent to the circle for tracks at high rapidity) to evaluate the radius of curvature of the muon track candidate[1]. LUTs are used to evaluate the function that relates p_T to the measured radius, ρ, as a function of η and ϕ.

1. The use of the radius is preferred to the sagitta because the radius is less sensitive to the layout inhomogeneities present at high rapidity.

8.4.2.3 Performance of the Standalone LVL2 Muon Algorithms

The performance of the LVL2 muon algorithms can be evaluated using several parameters. Ultimately the concern is the efficiency for selecting true muons with p_T above a given value, and the corresponding background rejection factor. These depend upon the track-finding efficiency, the fake-track rate and the resolution with which p_T is reconstructed. It is also esssential that the final online algorithm satisfies the timing constraints of code running in the LVL2 trigger processors.

For both the algorithms discussed above, some modelling of the cavern background has been done. For μFAST, a dedicated simulation within the DICE [8-23] framework reproduces the particle flux predicted by the FLUKA [8-24] program in the muon spectrometer. This background simulation package is general and can be used for any study of background in the Muon Spectrometer. For BMC_TRIG, incoherent background (corresponding to a 10% occupancy) has been simulated for the muon system.

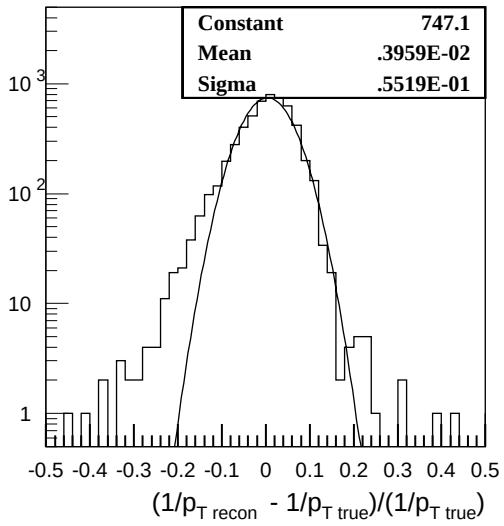

Figure 8-5 The distribution of $(1/p_T^{muon} - 1/p_T^{true})/(1/p_T^{true})$ for the μFAST algorithm, for (p_T = 6 GeV) muons in the region $|\eta| < 1$.

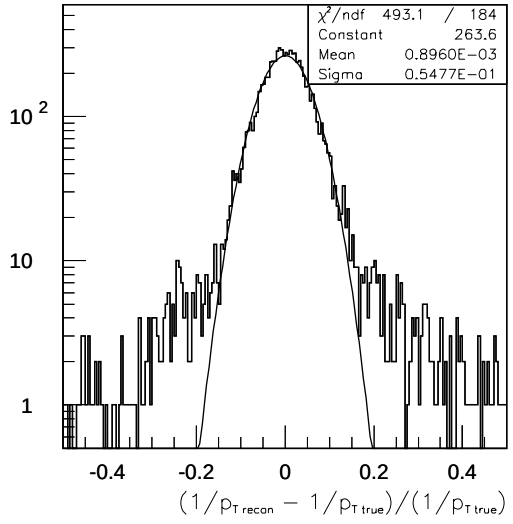

Figure 8-6 The distribution of $(1/p_T^{muon} - 1/p_T^{true})/(1/p_T^{true})$ for the BMC_TRIG algorithm, for (p_T = 20 GeV) muons in the region $1 < |\eta| < 2$.

The p_T resolution of the algorithms is a key factor in determining the selection efficiency and rate reduction that can be achieved at LVL2. The distribution of $(1/p_T^{muon} - 1/p_T^{true})/(1/p_T^{true})$ is shown in Figures 8-5 and 8-6, for the μFAST algorithm in the barrel ($|\eta| < 1$) for (p_T = 6 GeV) muons, and for the BMC_TRIG algorithm in the region $1 < |\eta| < 2$ for (p_T = 20 GeV) muons respectively. The non-Gaussian tails arise largely from the production of soft particles accompanying the muon.

The p_T resolution of the μFAST algorithm is shown as a function of p_T in Figure 8-7 (for $|\eta| < 1$). The p_T resolution of the BMC_TRIG algorithm is shown as a function of $|\eta|$ in Figure 8-8, for 6 GeV and 20 GeV p_T muons. The behaviour of this resolution as a function of p_T in the region $|\eta| > 1$ is still under study, and could be improved by further optimization of the LUT calculation, given the strong magnetic field and layout inhomogeneities present in this region. In the plots, the effect of the cavern background has not been taken into account. Howev-

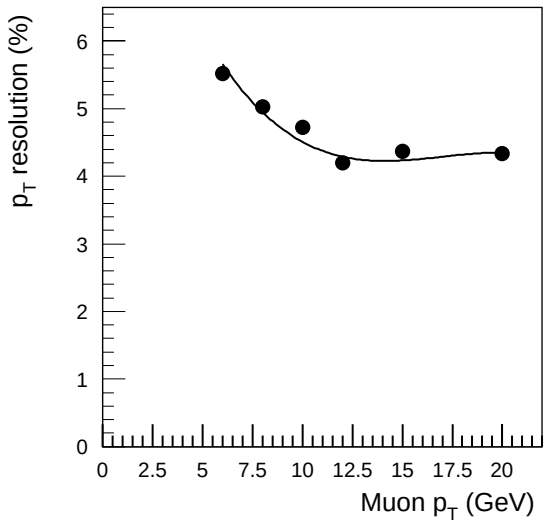

Figure 8-7 The p_T resolution of the μFAST algorithm as a function of the muon p_T, for $|\eta| < 1$.

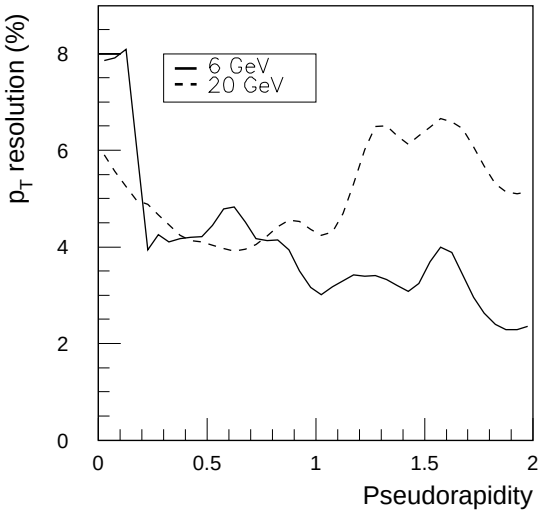

Figure 8-8 The p_T resolution of the BMC_TRIG algorithm as a function of $|\eta|$ for two values of the muon p_T.

er, an evaluation has been made of the effects of this background on the resolution for nominal values of the background at design luminosity, and they are found to be negligible.

As shown in Figure 8-7, the μFAST resolution ranges between 5.5% and 4.0% for muons in the p_T interval 6–20 GeV; similar resolutions are obtained from BMC_TRIG, as shown in Figure 8-8. These results compare reasonably well, in particular in the low-p_T case, with the transverse-momentum resolutions obtained with the offline reconstruction programs [8-20], which are about 4.5% at p_T = 6 GeV (and 2.5% at p_T = 20 GeV). The efficiencies of the two algorithms for selecting prompt muons at 6 GeV and 20 GeV thresholds, relative to muons accepted by the LVL1 muon trigger, are shown in Figure 8-9. For a nominal threshold of 6 GeV, the efficiency is about 90%, including the geometrical acceptance. This efficiency increases to about 95% for the 20 GeV threshold.

The total rates after these algorithms (including the rejection provided by the LVL1 trigger) have been calculated by convolving the algorithm efficiency as a function of p_T with the p_T distribution of the relevant contribution after LVL1. 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 been possible (p_T = 10 GeV) is assumed to constitute a plateau extending to the lower limit of the p_T acceptance (p_T = 3 GeV in the barrel and p_T = 2 GeV in the end-cap at $|\eta| = 2$). The rates from π/K decays are calculated using the predicted cross-section from the `DPMJET` [8-25] program, and would be lower by 50% if the `PYTHIA` [8-26] prediction were used.

The total rates after LVL2 are shown in Table 8-4. Rates seen from the μFAST algorithm are shown in the region $|\eta| < 1$, whilst those from BMC_TRIG are shown for $|\eta| < 2$. The reduction in rate with respect to LVL1 is:

- A factor of ~ 2 in the barrel ($|\eta| < 1$) and ~ 5 in the end-cap ($1 < |\eta| < 2$) for the 6 GeV low-p_T threshold at low luminosity;
- A factor of ~ 10 in both the barrel and the end-cap for the 20 GeV high-p_T threshold at design luminosity.

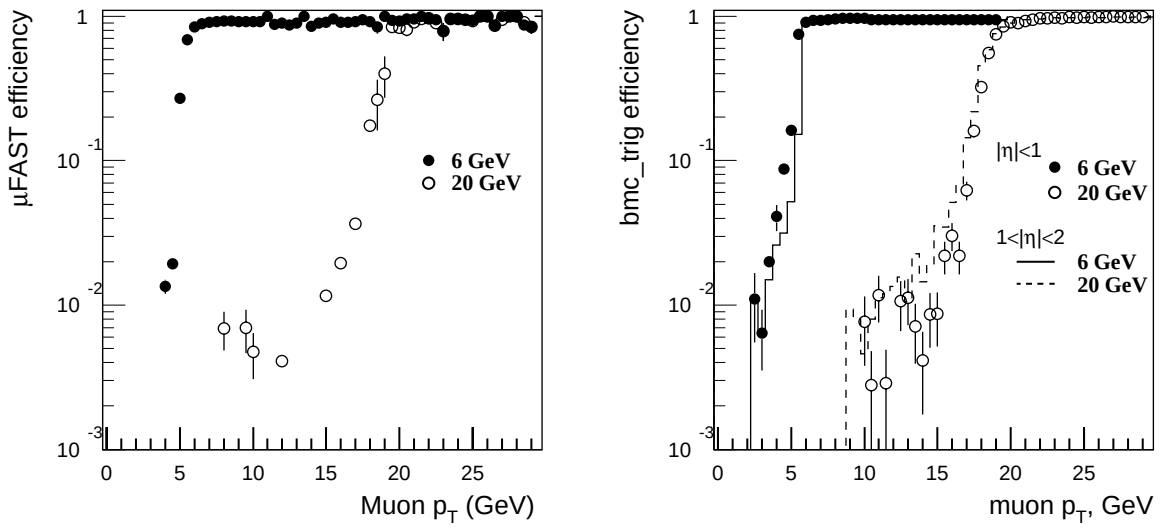

Figure 8-9 The selection efficiency for prompt muons as a function of p_T for the LVL2 muon algorithms *relative to the LVL1 output*, for p_T thresholds of 6 GeV and 20 GeV. The left plot shows the results from the μFAST algorithm in the region $|\eta| < 1$, and the right one those from the BMC_TRIG algorithm in the region $|\eta| < 2$.

Table 8-4 Total output rates of the standalone LVL2 muon trigger after application of the μFAST and BMC_TRIG algorithms for the 6 GeV low-p_T threshold at low luminosity and 20 GeV high-p_T threshold at design luminosity.

	Rate (kHz)			
	Low p_T (6 GeV)		**High p_T (20 GeV)**	
Contribution	**μFAST (|η|<1)**	**BMC_TRIG (|η|<2)**	**μFAST (|η|<1)**	**BMC_TRIG (|η|<2)**
π / K decays	3.1	5.0	0.06	0.13
b-decays	1.0	1.9	0.09	0.23
$W \rightarrow \mu\nu$	negligible	negligible	0.05	0.11
c-decays	0.5	1.1	0.04	0.09
cavern background	negligible	under study	negligible	under study
Total	4.6	8.0	0.24	0.56

Preliminary studies of the trigger rate arising from the cavern background indicate that the probability of a fake LVL1 muon trigger passing LVL2 is below 10^{-2}. This upper limit is sufficient to neglect the contribution from fake muons for the region $|\eta| < 1$.

The two LVL2 trigger algorithms have been benchmarked on several processors. On a processor corresponding to 10 SPECint95 units, the μFAST algorithm takes ~ 2 ms per RoI, independent of the trigger threshold and the true muon p_T, and including a simulation of the nominal background. The BMC_TRIG algorithm, which has not yet been optimized, takes at present ~ 13 ms.

8.4.2.4 Combined Muon Reconstruction Trigger Algorithm

The combination of features from the Muon Spectrometer and the Inner Detector (ID) at LVL2 and in the EF provides some rejection of π and K decays to μ, and an improvement in the momentum resolution of reconstructed muons over a large momentum range.

The matching of ID and Muon-Spectrometer tracks requires the extrapolation of the ID track, accounting for the detector geometry, for the material composition and, in particular, for the magnetic field which in some regions is very inhomogeneous. An accurate extrapolation is expensive in terms of CPU time and demands a detailed geometry and magnetic-field description, which limits its use at LVL2. To provide fast tracking procedures, the effects of geometry and magnetic field on the muon trajectory have been described by simple analytic functions that account for the field integral as a function of η and φ. The extrapolation of ID tracks to the entrance of the Muon Spectrometer is performed using linear extrapolations in two independent projections: the transverse and the longitudinal views [8-27]. The resulting residuals between the muon and ID tracks are then calculated in both φ and *z*, and average corrections are applied. In the transverse projection the ID track extrapolation in φ is corrected as follows:

$$\Delta\phi = \frac{\alpha}{p_T - p_T^0},$$

where α is related to the field integral, $\int B\mathrm{d}l$, and p_T^0 allows for the transverse momentum loss in the material of the calorimeters, which should be approximately independent of p_T. It is found that $p_T^0 \sim 1.5$ GeV, corresponding to half the transverse energy loss in the calorimeters, as would naïvely be expected. Both α and p_T^0 have been determined by fitting the Δφ residuals of simulated muons as a function of p_T. In the longitudinal projection, the *z*-coordinate is corrected empirically as a function of p_T. The matching is done geometrically using cuts on the residuals in each of *z* and φ.

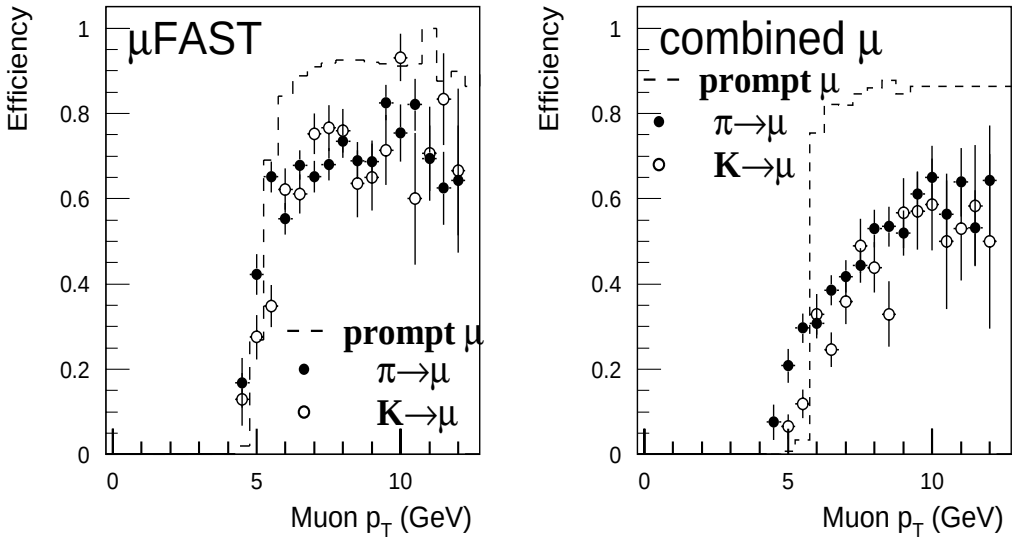

Figure 8-10 The efficiency with respect to LVL1 of the combined reconstruction at LVL2 for prompt muons and for muons from π/K decays. The lefthand plot shows the efficiency of the standalone LVL2 muon algorithm, μFAST, and the righthand plot shows the efficiency of the combined muon algorithm.

Since measurements of the muon direction in the ID are more accurate than those in the muon spectrometer, the ID ones are used to describe the combined muon object. The muon p_T measurement is performed by combining in a weighted average the independent transverse-momentum estimates from the muon spectrometer and the Inner Detector (using the appropriate resolution). For each combined track, a χ^2 parameter is used to evaluate the quality of the p_T matching; secondary muons originating from π and, especially, K decays in the central cavity typically have poor χ^2 values. A χ^2 cut can therefore be used to identify and reject non-prompt muons, which have lower transverse momenta than the parent meson.

As an example of these studies, Figure 8-10 shows the combined reconstruction efficiency for prompt and non-prompt muons, as a function of muon p_T, where the standalone codes from μFAST (Section 8.4.2.2) and the LVL2 Precision (SCT and Pixel) algorithm [8-10] have been used. The requirement of a good muon-track match reduces the low-p_T muon trigger rate in the barrel region from π/K decays to 1.0 kHz: a factor of three reduction compared to the rate from the μFAST algorithm. Including the further reduction in rate due to the increase in p_T resolution for prompt muons, the total rate from the combined muon algorithm in the region $|\eta| < 1.0$ is 2.1 kHz for muons with $p_T > 6$ GeV.

A more complete discussion of the combined muon algorithm at LVL2 and of the expected rates can be found in Ref. [8-27]. A preliminary timing analysis of the combined algorithm indicates that the processing time is about 20 μs on a 10 SPECint95 processor. Studies have begun to investigate the region $1.0 < |\eta| < 2.4$, and initial results suggest that similar fast algorithms may work equally well in this region. These results are encouraging for the B-physics analysis programme, where the high rate of muons from π/K decays has to be reduced as much as possible before an unguided track search in the Inner Detector is performed. More reduction is certainly possible at the EF level, using offline algorithms.

A further reduction of the rate of high-p_T muons can be achieved by an isolation requirement. As documented in Ref. [8-3], a rejection factor of about 10 against muons from b-decays was obtained (compared to the standalone muon selection), while keeping excellent efficiency for muons from W and Z decays.

The muon selection algorithms for LVL2 described above have a good efficiency for selecting low- and high-p_T muons and are able to reduce the rate after LVL2 significantly. The performance obtained in terms of computing resources is, with today's processors, already adequate for the requirements on the LVL2 system.

8.4.3 Selection of Events with Missing Transverse Energy, Jets and Taus

8.4.3.1 The High-Level Missing-E_T Trigger

The E_T^{miss} + jet trigger is an example of a trigger based on the combination of a global variable (E_T^{miss}) and localized Regions of Interest (RoIs) in the detector. The bulk of the trigger rate will result from fluctuations in the energy measurements of QCD jets, partly as a result of the presence of large amounts of material in front of the calorimeters at the interface regions between different elements of the calorimetry. The main instrumental effects arise from the difference in response between the various calorimeter technologies used and from the fact that the e.m. calorimeter is highly non-compensating.

The contribution of the EF to reducing the LVL1 missing transverse energy (E_T^{miss}) trigger rate should be significant for essentially three reasons. Firstly, accurate calorimeter calibration and

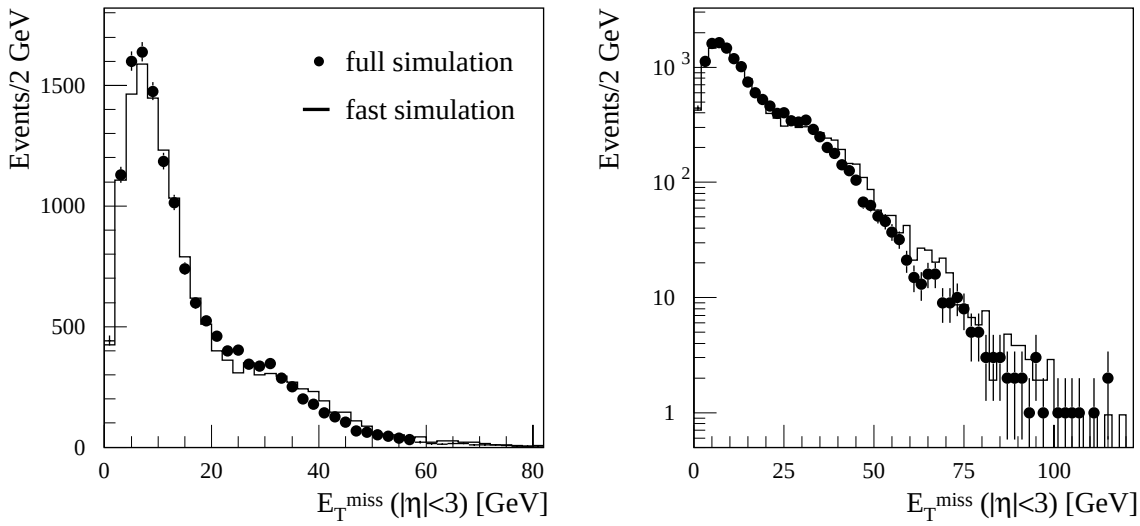

Figure 8-11 Comparison of $E_{\mathrm{T}}^{\mathrm{miss}}$ spectra for full simulation and fast simulation with ideal calibration (black dots = full simulation, full line = fast simulation) for |η| < 3.0. The comparison is shown both on a linear (left plot) and a logarithmic (right plot) scale.

inter-calibration are available [8-2]. Secondly, the effect of using a different calibration for low-energy calorimeter cells and for cells outside clusters can be taken into account. Thirdly, the cell E_{T} cutoff applied to suppress the noise contribution can be tuned accurately [8-28].

In the initial studies described here only the effects of an accurate, essentially ideal, calibration of the ATLAS calorimeters are taken into account: more details can be found in Ref. [8-29]. The $E_{\mathrm{T}}^{\mathrm{miss}}$ spectrum within $|\eta| < 3.0$ was calculated for samples of QCD di-jets (from PYTHIA [8-26]) using both full and fast simulation. The fast simulation studies used a version of ATLFAST [8-30] modified [8-31] to incorporate a better description of the longitudinal energy sharing between the e.m. and hadronic calorimeters, the effects of the interface regions, and the response and resolution functions obtained from a full ATLAS GEANT simulation [8-32]. The excellent agreement between full and fast simulation in the region of $|\eta| < 3.0$ is seen in Figure 8-11. The $E_{\mathrm{T}}^{\mathrm{miss}}$ spectrum for the full simulation has been calculated by reweighting the $E_{\mathrm{T}}^{\mathrm{miss}}$ components from the different calorimeters with the inter-calibration factors used in Ref. [8-2].

The value of $E_{\mathrm{T}}^{\mathrm{miss}}$ obtained from the fast simulation (extended to cover the full pseudorapidity range $|\eta| < 5$) will be referred to as the *software* $E_{\mathrm{T}}^{\mathrm{miss}}$ variable in the following. The selection at the EF corresponds to applying a threshold on this variable. For a given applied LVL1 threshold, the EF threshold is set to a higher value for which LVL1 will retain 95% of the events selected by the EF alone. The value for the software threshold is about 10–15 GeV higher than the corresponding LVL1 threshold. If the software threshold is set equal to the applied LVL1 threshold, the LVL1 acceptance for events selected by the EF is about 70%.

The reconstruction of jets in the HLT is done using a cone algorithm with a cone of radius $\Delta R = 0.7$ in the fast simulation. The software jet threshold is applied on the transverse energy inside this cone and the threshold value is derived as described for E_T^{miss} above. The difference between the LVL1 and software jet thresholds varies between 15 GeV (LVL1 jet[1] threshold of 10 GeV) and 30 GeV (LVL1 jet threshold of 60 GeV). A discrepancy between the efficiency curves obtained here and in Ref. [8-31], which may be due to a change in the jet scale in ATLFAST of ~ 10%, is currently under study. In contrast to Section 8.4.3.2, the jet threshold definition used in this study is not with respect to a reference jet defined at particle level.

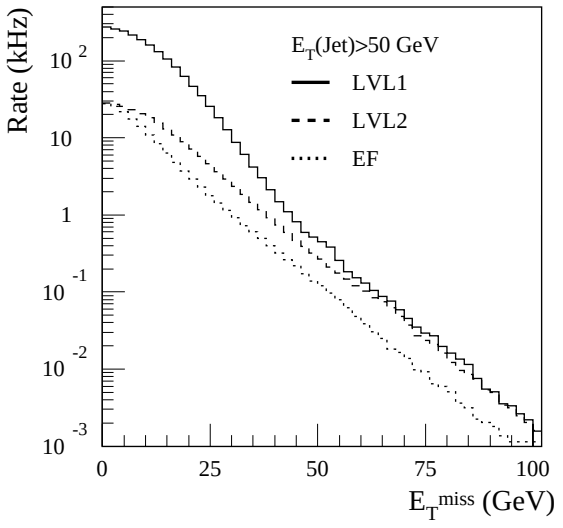

Figure 8-12 Missing-E_T rates from QCD di-jets (with $\hat{p}_T > 35$ GeV) as a function of the appropriate E_T^{miss} threshold for a jet E_T threshold of 50 GeV. The rates are shown for the three trigger levels for the case of low luminosity with the superposition of an average of 2.3 minimum-bias events.

The LVL1 rates are derived using the thresholds obtained from the procedure described above, to yield a 95% efficient LVL1 selection. For the rate determination at LVL2, it is assumed that no E_T^{miss} recalculation is done and the LVL1 E_T^{miss} value is used. In Table 8-5, the rates at the three trigger levels are shown for four values of the software E_T^{miss} threshold and two different software jet thresholds.

Table 8-5 Rates of a jet + E_T^{miss} trigger in Hz for software jet E_T thresholds of 50 GeV and 60 GeV, showing the three trigger levels. For the low-luminosity case shown, on average 2.3 minimum-bias events have been superimposed. The E_T^{miss} figure is the value of the software E_T^{miss} for which the applied LVL1 thresholds achieve 95% efficiency.

E_T^{miss}	50 GeV jet E_T Threshold			60 GeV jet E_T Threshold		
(GeV)	LVL1	LVL2	EF	LVL1	LVL2	EF
40	40400	6450	400	21000	4000	260
50	4190	1450	140	2550	1050	90
60	600	390	50	430	290	40
70	150	120	15	130	90	15

The reduction factor between LVL1 and EF varies with the cuts applied. For $E_T^{miss} > 50$ GeV and $p_T^{jet} > 50$ GeV (LVL1 thresholds of 40 GeV and 30 GeV, respectively) it amounts to about 30, corresponding to a LVL1 rate of about 4 kHz. The effect of the 10 GeV shift between the LVL1 and the software E_T^{miss} threshold can be seen from the curves shown in Figure 8-12. For the same value of the abscissa, in the region of 50 GeV, the reduction from the LVL1 curve to the EF curve is only a factor of 4–5. The additional reduction because in order to preserve high efficiency for a software E_T^{miss} threshold of 50 GeV, the corresponding LVL1 selection must apply a threshold

1. At LVL1, a jet is defined using a sliding window of size $\Delta\eta \times \Delta\varphi = 0.8 \times 0.8$ and the jet transverse energy is the sum of the trigger towers inside this window.

of 40 GeV. Conversely, if the applied LVL1 thresholds are fixed at 50 GeV to obtain a trigger rate of 400 Hz, 95% efficiency with respect to the EF selection is only achieved for software $E_{\mathrm{T}}^{\mathrm{miss}}$ and jet thresholds of ~ 60 GeV. In this case, an output rate of 40 Hz for the EF is found, corresponding to a reduction factor of about 10.

This preliminary analysis has only been performed for low luminosity and is based on a parametrized approach. A detailed understanding of the reduction of the $E_{\mathrm{T}}^{\mathrm{miss}}$ + jet trigger rate in the HLT awaits the results of an analysis that takes into account the use of full simulation up to $|\eta| = 5$, accurate calorimeter calibration and inter-calibrations (including the ones for low-energy calorimeter cells and for cells outside clusters), and an accurately tuned cell-energy cutoff. So far, no timing measurements have been made, but it is expected that the $E_{\mathrm{T}}^{\mathrm{miss}}$ algorithm will not contribute significantly to the required computing resources for the EF. At present, it is not foreseen at LVL2 to recalculate $E_{\mathrm{T}}^{\mathrm{miss}}$ using the full-granularity calorimeter data, because the large amount of data to be transferred in this case. One could envisage, however, improving the LVL1 result by taking into account overflows in the LVL1 trigger towers.

8.4.3.2 The High-Level Jet Trigger

The high-level jet trigger is required to reduce the rate of events containing jets compared to LVL1 by improving the transverse-energy measurement, using refined energy calibration and jet definition. It is important to keep in mind that jets are the dominant high-p_{T} process, so rate reduction cannot be expected from removing fake jet objects, as is possible with, for example, electromagnetic clusters. Currently, there are two sets of jet thresholds [8-3] envisaged for LVL1: j180, 3j75, 4j55 at low luminosity; and j290, 3j130, 4j90 at design luminosity. Jets are reconstructed in the region $|\eta| < 3.2$. LVL1 is optimized for a 95% jet efficiency for jets at the trigger threshold of the various menu items. The optimization is performed with respect to reference jets (jets reconstructed at the generator level). After LVL2, the overall jet-finding efficiency is reduced to 90%. Compared to LVL2, the EF may gain from an improved energy calibration. Also, for the EF, the jet thresholds are tuned for a 90% efficiency compared to the reference jets. A detailed description of the HLT for jets is given in Ref. [8-33].

Table 8-6 Expected rates of the high-level jet triggers at low and design luminosity.

Low Luminosity	Hz		Design Luminosity	Hz	
Trigger	**LVL2**	**Event Filter**	**Trigger**	**LVL2**	**Event Filter**
j180	278 ± 28	253 ± 26	j290	385 ± 103	275 ± 87
3j75	385 ± 32	286 ± 28	3j130	550 ± 123	440 ± 110
4j55	184 ± 23	127 ± 15	4j90	220 ± 78	175 ± 67
TOTAL	**619 ± 41**	**506 ± 37**	**TOTAL**	**853 ± 153**	**633 ± 132**

The LVL2 jet algorithm is described in detail elsewhere [8-34]. For every LVL1 jet RoI, jets are reconstructed in a window of size $\Delta\phi \times \Delta\eta = 1.0 \times 1.0$. To reconstruct jets, a cone algorithm with a radius of $\Delta R = 0.4$ is used. At the EF level, the same jet-finding algorithm is used, but jets are sought in the whole rapidity range and are no longer guided by the RoI. Table 8-6 summarizes the performance. It should be noted that the rates quoted here are higher than those given in Ref. [8-3], where the applied threshold cut is chosen to be the energy given by the menu item, thus leading to a much lower efficiency in selecting jets with true p_{T} equal to the nominal threshold. The rates are also dependent on the choice of reference jets. Compared to LVL1 and

using the same reference jets, LVL2 reduces the total jet trigger rate by a factor of 1.8 (2.1) at low (high) luminosity.

Reconstructing jets at the EF level improves the total jet trigger rate compared to LVL2 by only 20–30%. No significant reduction is achieved unless the algorithm is tuned for lower efficiencies. Raising the E_T thresholds by 20 (60), 14 (5), 8 (19) GeV for the single-, three- and four- object triggers, respectivley, reduces the rate by a factor of two at low (high) luminosity. For example, to achieve a final trigger rate of 25 Hz at low luminosity nominal E_T thresholds of 360 GeV, 150 GeV and 100 GeV are needed for the single-, three- and four- jet triggers, respectivley (the corresponding actual thresholds are 320 GeV, 130 GeV and 80 GeV). The values of the jet E_T thresholds for LVL1 [8-2] were deliberately chosen to be lower than those expected to be used for QCD studies to leave open the possibility to do b-jet tagging at LVL2 (see also Section 8.4.4)

Measurements of the algorithm execution times for two different jet-finding algorithms, the standard cone algorithm and a k_T-clustering algorithm [8-35], are shown in Table 8-7 for LVL2 and the EF. It can be seen that the algorithm execution time for the k_T algorithm increases if zero

Table 8-7 Performance of the jet HLT at low luminosity: the benchmarking results are quoted for a 500 MHz Pentium II machine running Linux. The times are given in terms of the quantities m_{50} and m_{95} of the timing distribution (as defined in Section 8.4.1.2) .

	LVL 2		**Event Filter**	
Algorithm	**$m_{50}(m_{95})$ for 2σ cut (ms)**	**$m_{50}(m_{95})$ for 0σ cut (ms)**	**$m_{50}(m_{95})$ for 2σ cut (ms)**	**$m_{50}(m_{95})$ for 0σ cut (ms)**
Cone	1.7 (3.4)	1.9 (3.3)	17 (58)	22 (59)
k_T-clustering	201 (390)	240 (456)	3.3 (9.4) x 10^6	5.7 (13.2) x 10^6

suppression is not applied on the calorimeter cells. The time at the EF level is given by running the jet finder algorithm over the whole event. In the case where one runs the jet algorithm only in a window around the LVL2 RoI, the time taken at the EF level will be the same as at LVL2. The k_T algorithm is based on a combinatorial method and hence is much slower than the cone algorithm. The execution times obtained for the cone algorithm on an available processor are well within the requirements both for LVL2 and the EF.

8.4.3.3 The High-Level Trigger for Taus

The importance of a tau trigger stems more from the additional flexibility that it adds to the HLT selection scheme than from the strength of the physics requirements [8-2]. Existing results [8-3] have not yet been updated and in the following a brief summary is given. Tau identification requires the selection of a narrow isolated jet associated with one, or at most three, charged tracks. The average fraction of energy deposited in the e.m. calorimeter is 60%, with a rather narrow shower shape. At low luminosity, the expected rate is 160 Hz for a nominal threshold of 60 GeV (on the hadronic tau energy), with an efficiency close to 60%. Further rejection against jets can be obtained by demanding exactly one track, which however reduces the efficiency to ~ 30%. Further reduction of the tau trigger rate could be achieved using the EF. This reduction is expected to come from more accurate calorimeter calibration and inter-calibration and from more efficient tracking.

8.4.4 Selection of b-Jets

The flexibility in the HLT scheme can be exploited further by making use of b-jet tagging at LVL2 and / or the EF, which for some classes of events could possibly extend the physics performance. In particular, for topologies containing multi b-jets, the ability to separate, at LVL2 / EF, b-jets from light quark and gluon jets could lead to an increase of the acceptance for signal events if the use of lower jet E_T thresholds than those discussed in Section 8.4.3.2 at LVL1 is feasible.

The study presented here concentrates on the use of a b-tagging algorithm at LVL2, because an implementation at the EF, although it would surely provide a much better signal / background performance, would not permit the use of lower LVL1 jet trigger thresholds and therefore would permit a considerably larger acceptance for some physics signals of interest (see Section 8.4.4.4). The use of b-tagging at LVL2 requires, however, fast track reconstruction and therefore precludes the use of the standard EF / offline track-reconstruction packages. This section summarizes the results of a LVL2 study; more details can be found in Ref. [8-36].

8.4.4.1 b-Tagging Algorithm for LVL2

The reconstruction of the impact parameter in the transverse plane (d_0) is a crucial component for the b-jet trigger. The optimal way to obtain this information is through a complete track reconstruction and track fit using all the information from the Inner Detector. However, this approach may prove difficult at LVL2 on account of the computing power needed and volume of data to be accessed. There are significant advantages to algorithms based solely on the pixel system (PixTrig) [8-37], which benefit from the direct availability of the three space-points over $|\eta| < 2.5$.

In this study an ideally aligned pixel detector has been assumed. Beam-position stability and effects of misalignment should be addressed in future studies.

The b-tagging selection proceeds as follows. PixTrig performs a three-dimensional track reconstruction in the RoIs defined by each relevant LVL1 jet trigger, using the pixel-hit clusters contained in a region $(\Delta\phi, \Delta\eta) < (0.5, 0.5)$ around the RoI axis. Every possible triplet of points in different layers of the pixel detector, compatible with a track of $p_T > 2$ GeV, is formed. Ambiguities are removed on the basis of the track quality; then, for each track, the value of the impact parameter in the transverse plane, d_0, is calculated together with its error (parametrized as a function of the reconstructed p_T) is calculated.

A simple b-tagging algorithm, based on the significance of the transverse impact parameter, $S = d_0 / \sigma(p_T)$, is applied, using the likelihood-ratio method described in Ref. [8-2]. 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 $X \sim 1$ and u-jets if $X \sim 0$.

8.4.4.2 Results on Single b-Jet Tagging

The b-tagging algorithm has been characterized on single b-jets coming from $H \rightarrow b\bar{b}$ decays with $m_H = 100$ GeV produced in association with a W at low 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 the physics

channel which has been studied in the following ($H \rightarrow hh \rightarrow b\bar{b}b\bar{b}$, with m_H = 300 GeV). The efficiencies (ε_b^{LVL2}) for b-jets and rejection factors (R_u) against u-jets are given in Table 8-8. The processing time has been measured on a Pentium III (500 MHz) running Linux and found to be on average 4.4 ms per jet, dominated by the time needed for the pattern-recognition stage of the track reconstruction.

Table 8-8 Rejection of the LVL2 algorithm against u-jets for three different values of b-jet efficiency: 50% (top), 60% (middle) and 70% (bottom). The results are shown for different intervals of the jet E_T and the jet η.

	E_T < 40 GeV	**40 GeV < E_T < 80 GeV**	**80 GeV < E_T < 120 GeV**
|η| < 1.5	43 ± 5	41 ± 6	33 ± 9
	21 ± 2	22 ± 2	20 ± 4
	7.1 ± 0.4	12 ± 1	12 ± 2
|η| > 1.5	11 ± 1	10 ± 1	11 ± 3
	8.0 ± 0.7	8.0 ± 0.8	5.6 ± 1.1
	3.3 ± 0.2	5.2 ± 0.4	3.6 ± 0.6

The algorithm has also been tested on the same type of events with pile-up for the design-luminosity case. The average processing time increases to 96 ms per jet and a higher level of fake associations degrades the overall performance (at fixed efficiency the average rejection decreases by a factor of five with respect to the low-luminosity case). However, it has been found that the reconstruction of the primary vertex, computed as the barycentre of the impact point of the tracks along the *z* coordinate, available in good quality for 55% of the RoIs, allows one to recover the light-quark rejection factor obtained at low luminosity for b-jet efficiencies above 80%. For lower b-jet efficiencies the rejection power is degraded by a factor of between 1.5 and 2. Since the b-jet tagging is relevant for multi-jet events in which the probability of a primary-vertex reconstruction failure is quite low (0.45^n where *n* is the number of RoIs), it can be expected that the b-jet tagging performance will not be completely spoiled at the design luminosity.

8.4.4.3 Comparison with the Offline Algorithm

The performance of the LVL2 trigger algorithm has been directly compared to that of the offline algorithm. As an example, a trigger selection corresponding to $R_u = 3$ and $\varepsilon_b^{LVL2} = 85\%$ has been applied as a first step to a sample of b-jets and u-jets coming from $H \to b\bar{b}/u\bar{u}$ decays with $m_H = 100$ GeV. (This rejection is the one used in the multi b-jet analysis described below.) For jets selected in this way, the offline selection was then applied and the results are shown in Figure 8-13. It is found that the full offline performance is restored for a final b-tagging efficiency $\varepsilon_b < 70\%$. This demonstrates that the trigger and offline selection are well correlated and that, as long as the LVL2 efficiency is kept above 80%, it is possible to provide subsequent analyses with an unbiased jet sample in the region $\varepsilon_b < 70\%$. Different combinations of working points of LVL2 trigger selection and offline analysis could be chosen depending on the required offline b-tagging efficiency; examples of which can be found in Ref. [8-36].

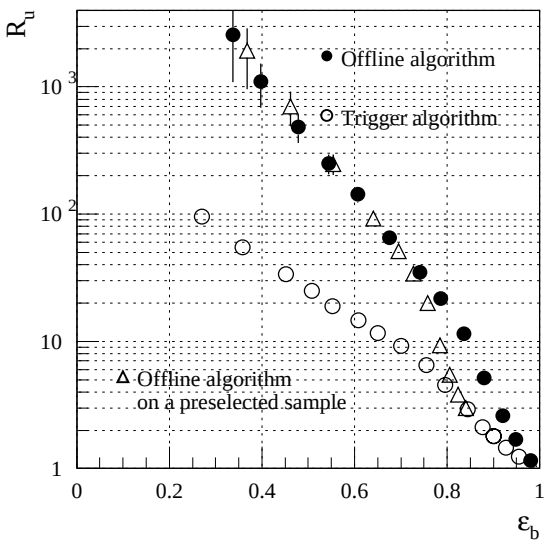

Figure 8-13 Rejection R_u against u-jets as a function of the b-jet efficiency ε_b for $H \to b\bar{b}/u\bar{u}$ decays with $m_H = 100$ GeV at low luminosity as obtained for the trigger and offline algorithms.

8.4.4.4 Multi b-Jet Tagging

Final states containing four b-jets have been proposed as signatures with a discovery potential for Higgs bosons in the MSSM ($b\bar{b}H$, $b\bar{b}A$ with $H/A \to b\bar{b}$ or $H \to hh \to b\bar{b}b\bar{b}$) [8-2]. The main drawbacks of these channels are the enormous background from QCD multijet production and the low production rate. In order to maintain a reasonable LVL1 trigger rate from jets, rather high E_T thresholds are set on the individual jets. With the LVL1 menu for three- and four-jets [(3j75, 4j55) at low luminosity and (3j130, 4j90) at high luminosity] presented in Ref. [8-2], some of these channels are selected with a poor efficiency. A study has been made to evaluate whether the jet E_T thresholds at LVL1 could be lowered usefully if a b-tagging algorithm were to be implemented at LVL2.

Table 8-9 Impact of a b-jet trigger at LVL2 for the channel $H \to hh \to b\bar{b}b\bar{b}$ with $m_H = 300$ GeV. The minimum value for the b-jet tagging variable required on each jet is identified by its rejection against individual u-jets.

	Trigger rate (kHz)	**Acceptance for $H \to hh \to b\bar{b}b\bar{b}$ events**
LVL1_standard: j180 or 4j55 or 3j75	0.3	0.19 ± 0.01
LVL1_ loose: j180 or 4j35 or 3j40	2.7	0.76 ± 0.01
LVL2_btag1: j180 or [4b35 (R_u=1.4)] or [3b40 (R_u=2.6)]	0.7	0.58 ± 0.02
LVL2_btag2: j180 or [4b35 (R_u=1.8)] or [3b40 (R_u=3.5)]	0.5	0.53 ± 0.02

The impact of a LVL2 b-jet trigger on the LVL1 trigger menu at low luminosity has been studied for the channel $H \to hh \to b\bar{b}b\bar{b}$ with $m_H = 300$ GeV, since this is the case where the largest po-

tential improvement could be obtained. The LVL1 multijet trigger E_T thresholds have been lowered to 4j35 and 3j40 ('LVL1_loose') and then a soft b-tagging selection has been applied on multi-jet events that would not have been selected by the *standard* LVL1 menu (in order to minimize the impact on physics channels of interest not containing b-jets). The results are summarized in Table 8-9 (for this example, LVL2 jet algorithms were not applied). The rejection factors obtained for LVL2_btag1 and LVL2_btag2 *(*about 6 and 12, respectively compared to LVL1_loose) are less than the third and fourth power of the rejection against single jets, because of the presence of gluon radiation, and b- and c-quarks in the background sample.

The results presented illustrate to what extent the acceptance for this particular channel could be enhanced, while keeping the constraint that the offline sample remain unbiased for ε_b up to 70%. The price to pay is a significant increase in the LVL1 rate, and to a lesser extent in the LVL2 rate; these could obviously be reduced at the expense of a smaller increase in acceptance. The final tuning of this trigger component will be a trade-off between the maximum acceptable LVL1 rate and the relevance of this physics channel based on its discovery potential.

8.4.5 Selection of B-Physics

The ATLAS B-physics trigger is initiated by a LVL1 muon which is confirmed at LVL2 in the muon spectrometer and in combination with an Inner Detector (ID) track (see Section 8.4.2.4). The next step is an unguided search for tracks in the ID. These tracks are used to reconstruct selected decay channels semi-exclusively. The parameters of tracks are combined in order to identify specific parent particles on the basis of the invariant mass. The requirements on the ID track-reconstruction algorithms for B-physics differ from those for RoI-guided track searches: the entire detector volume must be searched through, and the minimum p_T required for track reconstruction is, in general, as low as possible. Being able to calculate invariant masses requires three-dimensional track reconstruction with good efficiency and track reconstruction quality. Various candidate algorithms have been assessed for suitability for implementation in the LVL2 trigger and EF, including fast algorithms specific to the trigger (some of which are suitable for implementation on FPGAs), and algorithms taken from the offline reconstruction code.

The assessment of performance is based on measurements on simulated data of efficiency for signal events which would pass an offline selection, trigger rate and execution time. The performance has been measured for three channels of interest for physics studies. The easiest of the three, from the point of view of track reconstruction, is the case of $B_d \rightarrow \pi^+\pi^-$, giving rise to a final state with two high-p_T pions. The study of B decays with the subsequent decay $D_s^- \rightarrow \phi(K^+ K^-)\pi^-$ has to consider several final-state particles with lower transverse momentum (e.g. four in case of $B_s \rightarrow D_s^-\pi^+$ decay). The most challenging modes for track reconstruction concern *B* decays with the subsequent decay $J/\psi \rightarrow e^+e^-$, where the electrons have to be identified down to very low p_T and where allowance must be made for bremsstrahlung. The final HLT selection will obviously include triggers for many other channels, but these representative channels have been chosen to demonstrate the performance of the track-reconstruction algorithms and to show that a viable trigger can be constructed which will meet the requirements for B-physics. A summary of the selections and their performance is given here; more details can be found in Ref. [8-38].

8.4.5.1 Track Reconstruction

The track search can be initiated from the inner or outer parts of the ID, using information from either the Pixel or TRT detector, respectively. The result of the unguided *full-scan* track search is a set of tracks which form seeds for extrapolation into the SCT. In both cases the extrapolation is performed by a Kalman filter algorithm [8-39]. For TRT seeds, the extrapolation continues into the pixels. In the case of Pixel seeds, it would be beneficial to continue the extrapolation into the TRT in order to improve track parameter resolution and to benefit from the electron-identification capability of the TRT. More details on the two approaches can be found in Ref. [8-38].

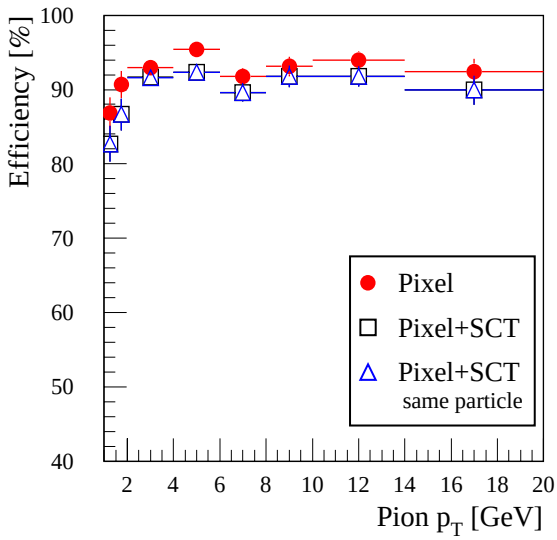

Figure 8-14 The reconstruction efficiency using the pixel scan and the pixel scan extrapolated to include the SCT for pions from the decay $B_d \rightarrow \pi^+\pi^-$ (filled circles = pixel scan, open squares = Precision algorithm with a pion $p_T > 1$ GeV and $|\eta| < 2.5$) as a function of the pion p_T. Also shown (open triangles) is the efficiency when the SCT and Pixel track segments are required to originate from the same particle.

The efficiency for the pixel scan [8-37] to reconstruct pions with $p_T > 1$ GeV from the decay $B_d \rightarrow \pi^+\pi^-$ is shown in Figure 8-14 as a function of the pion p_T. Also shown is the efficiency for reconstruction in the full Precision tracker, seeded by the pixel scan. These plots have been constructed by associating each track segment in the Precision tracker with the particle which contributed the majority of the hits; the track segment is then said to be due to this particle. A pion is defined to be found if there is a reconstructed track associated with it. In some cases, when there is a high density of tracks, it is possible that the SCT and Pixel segments of the same reconstructed track are due to different particles. This causes errors in the parameters of the combined track fit. As shown in Figure 8-14, the efficiency for both track segments to originate from the same particle is very close to the combined (Pixel + SCT) efficiency, as expected at low luminosity.

The efficiency for the TRT-scan and TRT-seeded Precision algorithm to reconstruct electrons is shown as a function of p_T and $|\eta|$ in Figure 8-15. These plots show results for the TRT-XK algorithm, which is based on the offline reconstruction package `xKalman` [8-16]. Similar results are obtained with the TRT-LUT algorithm, which is a look-up-table based implementation [8-40]. Both these algorithms use a histogramming method for the initial track search, followed by a fit. The TRT provides electron identification based on the fraction of hits passing a second, higher, discriminator threshold (transition radiation hits). The efficiency for electron tracks to be reconstructed and to pass the electron identification cuts is also shown in Figure 8-15. The correct extrapolation of tracks from the TRT into the SCT and Pixel detectors is affected by multiple scattering and by bremsstrahlung energy loss, which causes kinks in the tracks. These processes can lead to a failure to find a track in the SCT; mis-association can result in SCT and TRT track segments which are due to different particles. Also shown in Figure 8-15 is the efficiency for tracks identified as electrons, where both the SCT and TRT track segments are due to the same particle.

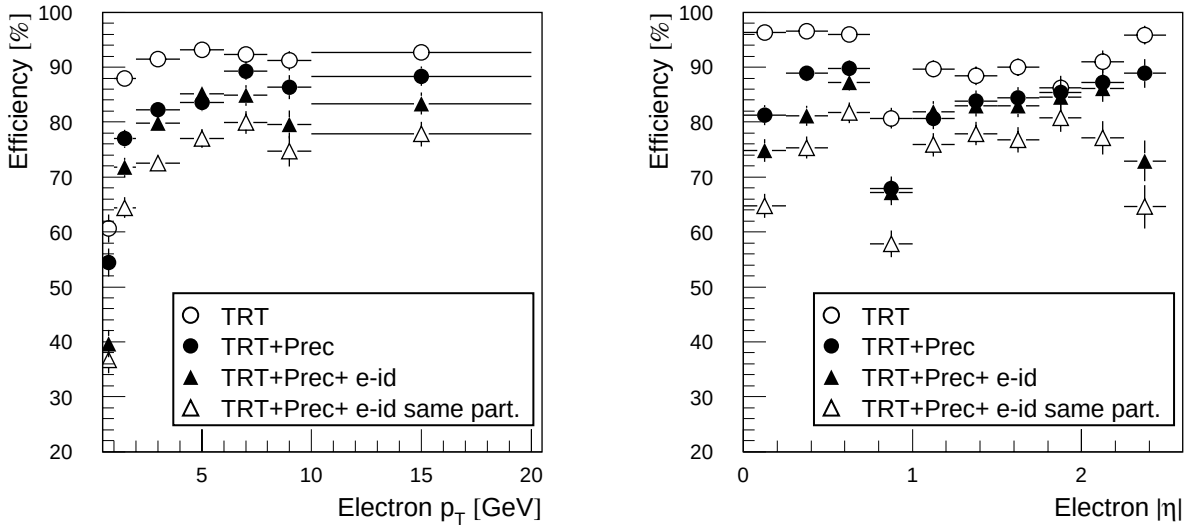

Figure 8-15 The efficiency to reconstruct electrons from $B \rightarrow J/\psi \rightarrow e^+e^-$ for the TRT-XK algorithm (open circles) and TRT-seeded Precision algorithm (filled circles) as functions of p_T (left: electrons with $p_T > 0.5$ GeV and $|\eta| < 2.5$) and of $|\eta|$ (right: electrons with $p_T > 1$ GeV and $|\eta| < 2.5$). Also shown is the efficiency for tracks identified as electrons based on the fraction of high-threshold hits on the track (filled triangles). The efficiency for tracks identified as electrons where both the TRT and Precision track segments are due to the same particle is also shown (open triangles).

8.4.5.2 Execution Time for LVL2

The execution times for the various algorithms have been measured on a 450 MHz Pentium-III processor with a 512 kbyte L2 cache and on a 600 MHz AMD Athlon processor with 128 Mbyte of 100 MHz memory, using a sample of $B \rightarrow \mu X$ events with pile-up corresponding to low luminosity. The results of these measurements are given in Ref. [8-38].

The execution time for the TRT full scan scales linearly with the number of hits. The mean execution time on the Athlon is 400 ms for the TRT-XK algorithm and 150 ms for the TRT-LUT algorithm. The most CPU-intensive parts of the TRT-LUT have also been implemented on FPGAs used both as a co-processor to a Pentium CPU and in the FPGA systems ENABLE++ and ATLANTIS [8-40]. The speed increase of the CPU-intensive part of the TRT algorithm resulting from using FPGAs was found to be about nine times; the speed increase of the total execution time (including those parts which are always done on a CPU) is about six times [8-41].

The pixel scan uses a combinatorial method, so the execution time contains contributions that scale as the second and third powers of the occupancy. It is therefore important to limit the number of point-combinations to be considered. The information from the LVL2 muon track can be used to determine the *z*-coordinate of the primary interaction point, which in turn can be used to reduce the number of hit combinations to be tried and hence the execution time. The mean execution time of the pixel scan on the Athlon is 23 ms with and 180 ms without muon guidance.

The Precision algorithm takes a mean of ~0.5 ms per pixel-track seed on the Athlon. A longer time of ~1 ms per seed is required per TRT-track, since in this case, the track search encompasses the Pixels as well as the SCT. A cut on the p_T of the Pixel or TRT track can be used, where physics allows it, to reduce the number of seeds and hence the overall execution time. The mean numbers of pixel and TRT seeds are given in Table 8-10 for a range of p_T-threshold values, together with the corresponding mean execution time for the Precision algorithm. The results are shown for the muon-guided pixel scan; without muon guidance the number of seeds is ~ 80% higher. For triggers involving only electron tracks, the number of TRT seeds can be reduced by using only TRT tracks passing electron identification cuts as seeds. The results shown in Table 8-10 are for TRT-XK; for TRT-LUT the number of seeds is ~ 50% higher.

Table 8-10 The mean number of tracks reconstructed by the muon-guided pixel scan and by TRT-XK in $B \rightarrow \mu X$ events with pile-up for various cuts on the reconstructed p_T. Also shown is the corresponding mean execution time for the Precision algorithm on a 600 MHz AMD Athlon both for pixel and TRT seeds.

	Pixel scan		TRT scan			
			All seeds		Electron seeds	
p_T^{min} (GeV)	No. seed	Time (ms)	No. seed	Time (ms)	No. seed	Time (ms)
0.0[a]	59	27	122	117	35	34
0.5	36	18	114	112	32	33
1.0	16	9	55	54	--	--
1.5	10	7	38	38	--	--

a. No additional p_T cut is applied, however the pixel and TRT algorithms make internal cuts.

8.4.5.3 LVL2 and EF Selections

Specific selections are applied for the different channels. For the $B_d \rightarrow \pi^+\pi^-$ trigger, tracks with $p_T > 4$ GeV are combined in all possible oppositely charged pairs. In order to select tracks from B_d decay candidates, for each pair cuts are applied on the scalar sum of transverse momenta, the difference in z-intercept of the two tracks, and a loose mass window cut. The mass cut provides a very effective selection of events for $B_d \rightarrow \pi^+\pi^-$ physics studies. Figure 8-16 shows the

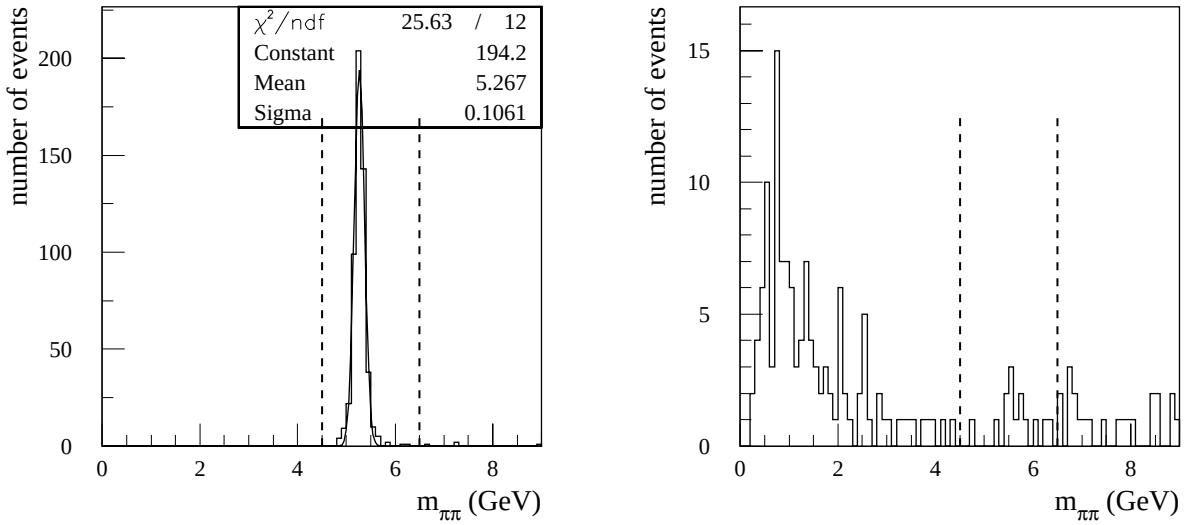

Figure 8-16 The invariant-mass distribution for the reconstructed π^+, π^- track pair from $B_d \rightarrow \pi^+\pi^-$ decays in events with pile-up (left) and for the best opposite-sign track pair per event in $B \rightarrow \mu X$ events with pile-up (right). The distributions were obtained using the pixel scan to seed the Precision algorithm.

invariant-mass distributions for signal (left) and background (right). The efficiency obtained using the muon-guided pixel scan to seed the Precision algorithm is given in Table 8-11 for signal events with pion $p_T > 4$ GeV. Also shown are the efficiency for events which would be selected by an offline analysis [8-2] and the rate for $B \rightarrow \mu X$ events with pile-up, normalized to a 9 kHz rate[1] for the LVL2 muon.

In the case of the $B_s \rightarrow D_s(\phi(KK)\pi)\pi$ trigger, first the ϕ and then the D_s are reconstructed from tracks with $p_T > 1.5$ GeV in a similar way to the $B_d \rightarrow \pi^+\pi^-$ decay. The resulting efficiencies and the rate obtained using the muon-guided pixel scan and Precision algorithm are given in Table 8-11. Also shown is the execution time on a 600 MHz Athlon.

A similar selection is performed for the $B \rightarrow J/\psi(ee)$ trigger using pairs of oppositely charged electron candidates. In this case a lower p_T cut of $p_T > 0.5$ GeV is applied and the tracks are required to pass electron-identification cuts. The best performance has been obtained using a TRT scan. The efficiencies and rates obtained for the $B \rightarrow J/\psi(ee)$ trigger using TRT-XK are given in Table 8-11. Similar results are obtained with TRT-LUT. Also given in Table 8-11 is the execution time for a full-scan by TRT-XK plus the time for the Precision algorithm seeded by electron candidate tracks with $p_T > 0.5$ GeV.

Table 8-11 Performance summary for the B-physics HLT. The efficiencies for the signal samples and the efficiencies for signal events that would be selected offline are given for each channel after LVL2 and after the LVL2 and EF selections. Also given are the rates and execution times for $B \rightarrow \mu X$ events with pile-up. Details of the selections can be found in Ref. [8-38]. The LVL2 results have been obtained with the Precision algorithm seeded by a pixel scan for $B_d \rightarrow \pi^+\pi^-$ and $D_s \rightarrow \phi(K^+K^-)\pi$, and seeded by TRT-XK for $J/\psi \rightarrow e^+e^-$. The rates quoted are normalized to a 9 kHz LVL2 muon rate. The LVL2 execution times were measured on a 600 MHz AMD Athlon. The EF execution time is based on the execution time for the offline reconstruction program `xKalman` [8-16].

	LVL2				LVL2 + EF		
Trigger	ε[a] **(%)**	ε **w.r.t. offline (%)**	**Rate (Hz)**	**Time (ms)**	ε **w.r.t. offline (%)**	**Rate (Hz)**	**Time (s)**
$B_d \rightarrow \pi^+\pi^-$	78	94	78	40	94	5	~10
$D_s \rightarrow \phi(K^+K^-)\pi$	53	56	196	40	56	27	~10
$J/\psi \rightarrow e^+e^-$	46	62	860	430	62	56	~10

a. Efficiencies for $B_d \rightarrow \pi^+\pi^-$ events with pion $p_T > 4$ GeV, $D_s \rightarrow \phi(K^+K^-)\pi$ events with all three final-state tracks with $p_T > 1.5$ GeV, and $J/\psi \rightarrow e^+e^-$ events with electron $p_T > 1$ GeV.

A further level of selection is possible at the EF. Shown in Table 8-11 are measurements of the efficiency and rate that could be attained after the LVL2 and EF selections; further details can be found in Ref. [8-38]. These measurements have been made using the offline reconstruction program `xKalman` [8-16]. Two-track (or three-track) fits are performed to pairs of oppositely charged tracks (to a pair of oppositely charged tracks and a further track) and a χ^2 cut is applied to select pairs (three tracks) consistent with having originated from a common vertex. A cut is also applied to the transverse decay length of the B_d, D_s and J/ψ candidates for the three channels. An angular cut is finally applied, in the transverse plane, to ensure consistency between the reconstructed momentum vector and the direction of flight of the parent meson determined from the reconstructed decay vertex. These cuts give a factor of 14 reduction in the total trigger

1. This assumption on the LVL2 muon rate is conservative, as described in Section 8.4.2.3; a further reduction in rate is expected by matching the track found in the muon spectrometer to a track found in the ID.

rate for the three channels, bringing it down to 90 Hz, with no loss of signal events passing the offline selections.

It is important that the trigger is robust with respect to detector inefficiency, noise and mis-alignment. The data sets used in this study contain a simulation of detector inefficiency (3% for the SCT, TRT and Pixels) and of the expected levels of front-end electronic noise in the SCT and Pixels. The effects of detector mis-alignment were not simulated, but have been studied by displacing the reconstructed points which form the input to the track-reconstruction algorithms. In a preliminary study for the $B_d \rightarrow \pi^+\pi^-$ trigger (for pions with $p_T > 4$ GeV), mis-alignment of the SCT and Pixel detectors using Gaussian distributions equal in width to the intrinsic detector resolutions causes no change in efficiency or rate. Increasing the level of mis-alignment by a factor of three causes a loss in efficiency of ~ 10% without change in the trigger rate.

8.5 Examples of Selection Sequences

The implementation of the HLT will determine which type of algorithm (for a given signature) is executed at LVL2 and at the EF (e.g. according to the degree of complexity), and also the sharing of the related decision sequence between the two trigger levels. In addition, within each trigger level, the sequence of algorithms and selection might depend on the implementation. The choice of implementation will have an impact on the required size of the system and the allocation of resources between the two trigger levels. In the following, examples are given for the sharing between LVL2 and EF for two representative signatures (inclusive electron and B-physics).

Table 8-12 Examples of algorithm and selection sequence for LVL2 and EF in the case of inclusive electrons.

Sequence A	Sequence B
LVL2 calorimeter	LVL2 calorimeter
LVL2 tracking	
LVL2 match (calorimeter, ID)	
EF calorimeter	EF calorimeter
EF tracking	EF tracking
EF match (calorimeter, ID)	EF match (calorimeter, ID)
EF bremsstrahlung recovery	EF bremsstrahlung recovery

For the inclusive-electron selection, the list of steps used as the baseline is given in Section 8.4.1 and repeated in Table 8-12 as ‘Sequence A’. It can be seen that, for the two trigger levels, several steps in the algorithm (and the related selection decisions) are conceptually similar. Possible differences are, for example, in the available detail of calibration constants. The recovery of bremsstrahlung and primary-vertex reconstruction are examples of more CPU-intensive calculations, which are likely to be done only at the EF. As shown in Table 8-12, an alternative sequence (‘B’) would only require the calorimeter algorithm to be executed at LVL2. As shown in Section 8.4.1.3, this would result in a much smaller loss of efficiency at LVL2 (compared to Sequence A), but the LVL2 output rate from this selection alone would be more than 3 (1) kHz at design (low) luminosity, with consequences for the subsequent data movement and processing.

Table 8-13 Examples of algorithm and selection sequence for LVL2 and EF in the case of B-physics.

Sequence A	Sequence B
LVL2 muon	LVL2 muon
LVL2 tracking	LVL2 tracking
LVL2 match (muon, ID)	LVL2 match (muon, ID)
LVL2 unguided search in ID	
LVL2 track refinement	
LVL2 calorimeter	
LVL2 muon	
EF muon	EF muon
EF tracking	EF tracking
EF match (muon, ID)	EF match (muon, ID)
EF unguided search in ID	EF unguided search in ID
EF track refinement	EF track refinement
EF calorimeter	EF calorimeter
EF muon	EF muon
EF invariant mass / topology	EF invariant mass / topology

In the case of B-physics, the present strategy builds on an unguided search for low-p_T particles over the full volume of one or more of the tracking subdetectors. Two possible implementations of sequences are shown in Table 8-13. The first example (Sequence A) performs an unguided track search at LVL2 and offers a significant reduction in the output rate of LVL2 B-physics, as shown in Section 8.4.5. It requires however more resources for the LVL2 system. The second example (Sequence B) shows an implementation, where LVL2 is used only in a RoI-guided mode. Once the best possible muon-track measurement has been obtained at LVL2, the event building is done and the EF starts to further improve the muon measurement, before initiating an unguided search for track candidates in the ID. Similarly to the case of the inclusive-electron selection, Sequence A places fewer system requirements on LVL2, but increases substantially the demands on the event building and the EF processing.

8.6 Global Performance of the HLT Selection

The studies summarized in Section 8.4 have shown that the currently available HLT selection algorithms have good physics performance, i.e. they provide good efficiency for physics objects and therefore high acceptance for the physics channels of interest. At the same time, they lead to a sizeable rate reduction of LVL1 accepted events. This physics performance is generally achieved while simultaneously fulfilling the requirements on the system performance; e.g. most of the RoI-guided LVL2 algorithms are already close to running within the final LVL2 latency constraints.

In the following, a brief summary of the global performance of the HLT selection is given for the physics objects which have been discussed in the previous sections in more detail. For the vari-

ous inclusive selections, the expected rate after the HLT, the corresponding reduction factor with respect to LVL1 and the single-particle efficiency are quoted, where available. In addition, the sharing of the rejection between LVL2 and the EF is given together with the corresponding efficiency losses.

8.6.1 Electrons

The expected rate after the HLT for an inclusive selection of isolated electrons [nominal threshold of 20 (30) GeV] at low (design) luminosity is 41 ± 6 (117 ± 30) Hz. This corresponds to a reduction in rate of a factor of 141 (185) and an efficiency of 80.8% (77.6%) with respect to LVL1. Of this reduction, LVL2 contributes a factor of about 41 (47) and the EF adds a further factor of about 3.4 (3.9). The efficiency after the LVL2 part of the selection is 88.1% (85.3%). When requiring no further efficiency loss at the EF, it has been shown that the EF is still capable of providing an additional rejection with respect to LVL2, albeit of a smaller size (a factor of 3 (2) at low (design) luminosity). The HLT electron selection is composed of a sequence of modular algorithms, which can be migrated from LVL2 to the EF (and vice versa).

After the HLT selection, the sample (at low luminosity) consists of 19% electrons from $W \rightarrow e\nu$, and 42% isolated electrons due to b- and c-production, with the remainder (39%) being fake electrons (mostly due to jets faking the electron signature) and electrons from photon conversions. At design luminosity, owing to the higher E_T threshold applied, about 40% of the HLT selected events are due to $W \rightarrow e\nu$; only 13% contain isolated electrons from b-/c-decays, and the remainder (47%) is due to fake jets and again electrons from photon conversions.

8.6.2 Photons

After the HLT photon selection (excluding the conversion identification algorithm), the inclusive rate for single isolated photons (nominal threshold of 40 (60) GeV) is 57 ± 7 (144 ± 33) Hz for low (design) luminosity. This corresponds to a reduction in rate of a factor 100 (150) and an efficiency of 84.7% (87.3%) with respect to LVL1 (where nominal thresholds of 20 (30) GeV are used), to which LVL2 contributes with a factor of 70 (87). The efficiency after the LVL2 selection alone is 94% (97.7%). Since the LVL2 calorimeter trigger has access to the full granularity information, most of the overall HLT rejection comes from this step. The additional rejection by the EF is at the cost of a loss of about 10% of the single-photon efficiency. In the reduction factors, however, the effect of the increased threshold at the HLT (compared to that at LVL1) is included. The HLT rate reduction, when calculated for the same nominal LVL1 thresholds, amounts to a factor of about 10 (25).

After the HLT (excluding the conversion identification algorithm), the inclusive single photon selection contains a fraction of 35% (23%) from direct photon production at low (design) luminosity. The remainder is due to fake photons from jets and photons produced by quark bremsstrahlung.

8.6.3 Muons

At low luminosity, the expected rate for $p_T > 6$ GeV muons in the region $|\eta| < 2.4$ after the standalone LVL2 selection is 9.4 kHz. Assuming that the combined reconstruction in the end-cap system is able to achieve a performance similar to that of the barrel system (see Section 8.4.2), a rate of 4.3 kHz is expected after matching to a track in the ID. This corresponds to a rate reduc-

tion of about a factor of five with respect to LVL1. Out of these rates, about 25% of the events before the matching (35% after matching) are expected to be due to b-decays. The fraction of π/K decays surviving is expected to be 65% (45% after matching); the remainder of the rate is due to c-decays.

At design luminosity, the expected LVL2 rate for 20 GeV muons in the region $|\eta| < 2.4$ is 690 Hz, giving a reduction in rate of a factor of approximately 10 with respect to LVL1. The corresponding efficiency (including the geometrical acceptance) with respect to LVL1 is 95%. About 19% of the events are from $W \rightarrow \mu\nu$, about 23% are from π/K decays, and the remainder are due to muons from b- and c-decays. It should be noted that the p_T resolutions achievable using the LVL2 standalone muon algorithms (5.5% at 6 GeV and 4% at 20 GeV) are close to the ultimate offline performance (4.5% and 2.5%, respectively). The possible improvement coming from offline analysis can only be checked with EF studies, which are not yet available.

Further reduction of the inclusive muon rate at design luminosity can be expected by requiring isolation around the muon direction. This should reject about 90% of the muons from b- and c-decays, while maintaining excellent efficiency for the isolated muons from $W \rightarrow \mu\nu$. Without taking into account a possible contribution from π/K decay, which will generally not be isolated, a lower limit on the 20 GeV isolated muon rate (due to W and b-/c-production) is about 140 Hz in the region $|\eta| < 2.4$.

8.6.4 Missing Transverse Energy

Emphasis has been put on the exploitation of the EF capabilities (full event data, improved calibration, noise control, etc.) to reduce the LVL1 rate. Depending on the different requirements imposed by the physics analyses (either keeping the LVL1 nominal threshold or using the EF sharper threshold), a reduction factor ranging from 10 to 30 is achievable at the EF. The expected rates after the EF selection amount to 38 Hz for $E_T^{miss}60$ + j60 and 139 Hz for $E_T^{miss}50$ + j50.

8.6.5 Jets

The expected reduction of the LVL1 jet rate after the HLT selection is approximately a factor of two. For low (design) luminosity the total rate expected is 500 (630) Hz for the nominal thresholds j180, 3j75 and 4j55 (j290, 3j130 and 4j90) used at LVL1. In order to obtain a total HLT jet output rate of 25 Hz, the nominal thresholds for the single-, three- and four-jet triggers have to be raised to the following values at low luminosity: j360, 3j150 and 4j100. Similarly, for design luminosity, the nominal thresholds have to be increased compared to LVL1 to achieve an output rate of 25 Hz. Jets with lower thresholds will be accepted by prescaling of these events. Except for the case of b-jet tagging (see below), the thresholds for the jet triggers at LVL1 could be increased.

8.6.6 b-Jet Tagging

For a single b-jet efficiency of 60% (70%), the rejection against u-jets obtained with a LVL2 algorithm using the pixel system alone is about 15 (9). If this algorithm is required to provide subsequent analyses (offline or EF) with an unbiased jet sample in the region $\varepsilon_b < 70\%$, working points for the trigger algorithm with a lower rejection (e.g. $R_u \sim 3$ and $\varepsilon_b^{LVL2} \sim 85\%$) should be used. Using this low-rejection working point, the b-tagging has been applied to multi-jet events. It has been found that the acceptance for final states containing multiple b-jets could be im-

proved by lowering the standard p_T^{jet} thresholds at LVL1 (at the price, however, of having a significant increase in the LVL1 jet rate and to a lesser extent also in the LVL2 jet rate) and including b-jet tagging at LVL2 to reduce the rate.

8.6.7 B-Physics

With respect to previous analyses (which were based on modified offline code), a comparable performance has now been obtained for the $B_d \rightarrow \pi^+\pi^-$ and $B \rightarrow D_s(\phi(K^+K^-)\pi)\pi$ selections using algorithms suitable for the LVL2 trigger. In the case of the $B \rightarrow J/\psi(e^+e^-)X$ channel, a LVL2 rate 2.8 times higher than that reported in Ref. [8-2] is currently obtained using the same selection cuts (for 6% higher efficiency). It is possible to obtain performance approaching that reported in Ref. [8-2] using different selection cuts, details of which are given in Ref. [8-38].

In the following, the rates expected after the HLT are given normalized to a LVL2 inclusive $p_T > 6$ GeV muon rate of 9 kHz. As described in Section 8.6.3, this might be reduced further to about 4.3 kHz. Selections for three representative channels have been studied: $B_d \rightarrow \pi^+\pi^-$, B decays including the subsequent decay $D_s \rightarrow \phi(K^+K^-)\pi$, and B decays including the subsequent decay $J/\psi \rightarrow e^+e^-$. For these channels, a total rate of 88 Hz is expected after the HLT. This corresponds to a rejection factor of 260 with respect to LVL1 and a factor of 100 with respect to the LVL2 standalone inclusive muon selection (a factor of 60 with respect to the matched muon selection). The rejection with respect to LVL1 is shared between LVL2 (a factor of about 20) and the EF (a factor of about 13).

For these three selections, the expected yield from signal events [$B_d \rightarrow \pi^+\pi^-$, $B_d \rightarrow J/\psi(e^+e^-)\,K^0_s(\pi^+\pi^-)$ and $B_s \rightarrow D_s\pi \rightarrow \phi(K^+K^-)\pi$] passing the offline selection criteria [8-2] (including the LVL1 and LVL2 trigger conditions) corresponds to a rate of the order of 10^{-3} Hz.

8.6.8 HLT Output

In the following, an estimate of the overall HLT rate due to genuine single and double high-p_T objects is given, without taking into account the rate from fake contributions. As discussed in the previous subsections, in most cases the fraction of the rate due to fake contributions is of the order of 50%. A further reduction of these fake components could be achieved; however, the related increase in system resources used (e.g. CPU time) and the reduced signal efficiency have to be taken into account. For the selections presented in the previous sections, further improvements in the system performance (e.g. the algorithm latency) are possible.

At low luminosity, in a very conservative approach, only the following trigger objects are considered (the physics process contributing dominantly to the rate is indicated in brackets): μ20 ($W \rightarrow \mu\nu$), 2μ10 ($Z \rightarrow \mu\mu$), e20 ($W \rightarrow e\nu$), 2e15 ($Z \rightarrow ee$), γ40, j360, 3j150 and 4j100. The sum of the expected rates amounts to 65 Hz, without applying additional criteria such as b-jet tagging. The expected contribution from B-physics (for the three selections described above) is 90 Hz, giving a lower limit of 155 Hz on the total HLT rate at low luminosity. At design luminosity, with the same restrictions as before, the high-p_T objects considered are the following: μ20 ($W \rightarrow \mu\nu$), 2μ10 ($Z \rightarrow \mu\mu$), e30 ($W \rightarrow e\nu$), 2e20 ($Z \rightarrow ee$), γ60, j580, 3j260 and 4j150. The sum of these rates amounts in this case to 240 Hz.

In both cases, the value obtained (which excludes the contribution from fakes and from prescaled triggers as well as from monitor and calibration triggers) is somewhat larger than the assumed number of about 100 Hz to be written to mass storage. Further reduction of the rate due

to genuine high-p_T objects can be obtained by either increasing the p_T threshold or by requiring a more exclusive selection. For the latter, more specific physics processes would have to be targeted at the HLT selection level, and not only at the classification stage.

8.7 References

8-1 *Physics requirements for the ATLAS high-level trigger,* ATLAS internal note, ATL–DAQ–2000–033 (2000)

8-2 *ATLAS detector and physics performance technical design report*, CERN–LHCC / 99–14 / 15 (1999)

8-3 *ATLAS trigger performance status report,* CERN–LHCC / 98–15 (1998)

8-4 *ATLAS first level trigger technical design report*, CERN–LHCC / 98–14 (1998)

8-5 *Selection of high-p_T electromagnetic clusters by the level-2 trigger of ATLAS,* ATLAS internal note, ATL–DAQ–2000–002 (2000)

8-6 *First implementation of calorimeter FEX algorithms in the LVL2 reference software*, ATLAS internal note, ATL–DAQ–2000–020 (2000)

8-7 *Photon identification with the ATLAS detector,* ATLAS internal note, ATL–PHYS–99–016 (1999)

8-8 *Identification of high p_T electrons by the second level trigger of ATLAS,* ATLAS internal note, ATL–DAQ–2000–003 (2000)

8-9 *High-p_T level-2 trigger algorithm for TRT detector in ATRIG*, ATLAS internal note, ATL–DAQ–2000–043 (2000)

8-10 *Performance of a LVL2 trigger feature extraction algorithm for the precision tracker,* ATLAS internal note, ATL–DAQ–99–013 (1999)

8-11 *Global pattern recognition in the TRT for the ATLAS LVL2 trigger,* ATLAS internal note, ATL–DAQ–98–120 (1998)

8-12 *Performance studies for electron and photon selection at the event filter,* ATLAS internal note, ATL–DAQ–2000–007 (2000)

8-13 *1997 ATLAS jet production*, ATLAS internal note, ATL–PHYS–97–102 (1997)

8-14 *Electron/jet separation with the ATLAS detector,* ATLAS internal note, ATL–PHYS–99–015 (1999)

8-15 *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)

8-16 *Description of global pattern recognition program (xKalman),* ATLAS internal note, ATL–INDET–97–165 (1997)

8-17 *Further studies and optimization of the level-2 trigger electron/photon FEX algorithm*, ATLAS internal note, ATL–DAQ–2000–042 (2000)

8-18 *Isolation of electrons and photons on the second level trigger,* ATLAS internal note, ATL–DAQ–2000–026 (2000)

8-19 *IPATREC: inner detector pattern-recognition and track-fitting,* ATLAS internal note, ATL–SOFT–94–009 (1994)

8-20 *Muonbox: a full 3D tracking programme for muon reconstruction in the ATLAS spectrometer*, ATLAS internal note, ATL–MUON–97–198 (1997)

8-21 *A muon trigger algorithm for level-2 feature extraction*, ATLAS internal note, ATL–DAQ–2000–036 (2000)

8-22 *A method for a level-2 muon trigger for ATLAS*, ATLAS internal note, ATLAS–DAQ–99–03 (1999)

8-23 *DICE manual version 0.10*, ATLAS internal note, ATL–SOFT–95–011 (1995)

8-24 A. Ferrari et al., Z. Phys. **C70** (1996) 413

8-25 J. Ranft, *DPMJET version II.3 and II.4: sampling of hadron–hadron, hadron–nucleus and nucleus–nucleus interactions at cosmic ray energies, according to the Dual Parton Model*, INFN–AE–97–45 (1997)

8-26 T. Sjöstrand, *Pythia 5.7 and JETSET 7.4 physics and manual*, CERN–TH.7112 / 93 (1993)

8-27 *Combined muon reconstruction at level-2*, ATLAS internal note, ATL–DAQ–2000–037 (2000)

8-28 *Missing transverse momentum reconstruction in ATLAS*, ATLAS internal note, ATL–PHYS–96–080, (1996)

8-29 *Event filter rates for the etmiss+jets trigger at low luminosity*, ATLAS internal note, ATL–DAQ–2000–016 (2000)

8-30 *ATLFAST 1.0 a package for particle level analysis*, ATLAS internal notes, ATL–PHYS–96–079 (1996) and ATL–PHYS–98–131 (1998)

8-31 *Level-1 rates for triggers using the ETmiss signature*, ATLAS internal note, ATL–DAQ–99–011 (1999)

8-32 R. Brun et al., *GEANT3*, CERN / DD / EE / 84–1 (1996)

8-33 *Performance studies of jets in the high level trigger*, ATLAS internal note, ATL–DAQ–2000–015 (2000)

8-34 *Jet reconstruction at the second level trigger*, ATLAS internal note, ATL–DAQ–98–115 (1998)

8-35 *Jet finder library; version 1.0*, ATLAS internal note, ATLAS–SOFT–98–038 (1998)

8-36 *b-tagging event selection for the ATLAS high-level trigger*, ATLAS internal note, ATL–DAQ–2000–023 (2000)

8-37 *PixTrig: a track finding algorithm for LVL2*, ATLAS internal note, ATL–DAQ–2000–025 (2000)

8-38 *B-physics event selection for the ATLAS high-level trigger*, ATLAS internal note, ATL–DAQ–2000–031 (2000)

8-39 P. Billoir, Nucl. Instrum. Methods **225** (1984) 352;
P. Billoir and S. Qian, Nucl. Instrum. Methods **A294** (1990) 219;
P. Billoir and S. Qian, Nucl. Instrum. Methods **A295** (1990) 492

8-40 *Pattern recognition in the TRT for the ATLAS B-physics trigger*, ATLAS internal note, ATL–DAQ–99–007 (1999)

8-41 *LVL2 full TRT scan feature extraction algorithm for B physics performed on the hybrid FPGA/CPU processor system ATLANTIS: measurement results*, ATLAS internal note, ATL–DAQ–2000–012 (2000)

9 Architecture Proposal

9.1 Introduction

This chapter presents an architecture proposal for the ATLAS HLT / DAQ / DCS system. It provides a common architectural framework in which different implementation alternatives can be studied up to the Technical Design Report, at present foreseen for June 2001. The proposal is based on the work done by the HLT / DAQ / DCS community since the ATLAS Technical Proposal, published in April 1994 [9-1]. The previous chapters and associated back-up documents contain detailed descriptions, conclusions and results of this work. These chapters also detail the evolution of each project since the ATLAS Technical Proposal (TP), in terms of measurements made, conclusions reached and technology advances. The conclusions are the result of technical reasoning, and, where possible, this reasoning has been supported by prototype implementations and measurements performed within the appropriate project. Where measurements resulting from implementations have not been possible, results from simulation (modelling) studies have been used, for example to extrapolate from small-scale prototypes to full size-systems.

Section 9.2 gives the main requirements and an overview of the proposed HLT / DAQ / DCS architecture, introducing its main systems and subsystems and their associated boundaries and interfaces. Examples of how these systems and subsystems collaborate to provide the principle functionality of the system are also presented. In addition, event-data monitoring and partitioning are briefly described in the context of the proposed architecture. Sections 9.3 to 9.7 then go on to present a more detailed decomposition of the systems and subsystems introduced in Section 9.2. A summary of the chapter is given in Section 9.8, and some conclusions are presented in Section 9.9.

9.2 Architecture Overview

9.2.1 Major HLT/DAQ/DCS Requirements

The major requirements of the HLT / DAQ / DCS system are detailed below.

The architecture shall be:

- Able to select the events required for the analysis of all physics channels of interest with high efficiency.
- The same for all physics channels (i.e. RoI-guided high-p_T physics, B-physics, etc.).
- Able to accept the LVL1 Trigger accept rate of 75 kHz (upgradable to 100 kHz), and to produce an output rate acceptable to the offline (100–200 Hz).
- Tolerant to uncertainties in our current knowledge of rates, data volumes, algorithms, rejection factors and selection strategies.

- Able to allow event rejection to be done at different levels [i.e. LVL2 Trigger or Event Filter (EF)] in order to optimize the efficiency of the selection as a function of the particular physics channel.
- Able to import and allow the use of offline algorithms and other software facilities (e.g. the offline event display) online at the level of the EF.
- Able to allow the permanent running of a minimal subset of the system [e.g. gas, HV and safety aspects of the Detector Control System (DCS)] autonomously in order to ensure the safety of the entire detector (e.g. during shutdown periods).
- Highly flexible and adaptable to future possible evolution of the LHC machine (mode of operation, backgrounds, luminosity, etc.) and to new physics signatures.
- Able to evolve and take maximum advantage of technological advances over a period of ~ 15 years.
- Able to facilitate the use of commodity/commercial equipment and to benefit from industrial standards, where possible.
- Able to also coherently support DAQ development for the final ATLAS experiment, test beam, subdetector production testing systems, the ROD crate.
- Able to support system partitioning for the purposes of detector commissioning, calibration, testing and debugging.

9.2.2 Strategy

The proposed architecture is based on the following strategy that has been studied in the DAQ/EF -1 Project and the LVL2 Pilot Project:

- On receipt of a LVL1 Trigger accept signal, data are sent in parallel from the RODs of the detector subsystems to readout buffers.
- In parallel with the above, the LVL1 Trigger provides information for use in the LVL2 Trigger selection via a separate data path, e.g. potentially interesting features [i.e. Region of Interests (RoIs)], global information such as energy sums and the trigger type.
- Making use of the information provided by the LVL1 Trigger, the LVL2 Trigger selectively accesses subdetector data. Access is done, at least initially, only for regions of the detector flagged by LVL1 as being of interest, and for the detectors that were used in the LVL1 Trigger. Data from other detectors are accessed subsequently, only for RoIs that have been validated. Events may be rejected as soon as it is established that they do not satisfy the selection criteria.
- For events that are rejected by the LVL2 Trigger, the data are cleared from the readout buffers.
- For events that are retained by the LVL2 Trigger, the process of event building is initiated, collecting all relevant subdetector data for each event to a single location.
- Following event building, the EF analyses the assembled event data, making a final selection of events to be retained for offline analysis.
- For events that are accepted by the EF, the data are sent to mass storage.

In the description of the strategy given above, many details are omitted for reasons of simplicity, and only the case of standard physics-event selection is addressed. More details are given in the discussion later in this section.

The architecture described in the following sections supports the strategy described above. It is presented using the Unified Modelling Language (UML) [9-2]. This gives a consistent description of the full HLT / DAQ / DCS system. The architecture at a high level follows closely that presented in the ATLAS TP, a major difference being the use of sequential processing in the LVL2 selection process.

9.2.3 High-Level Functional Specification

The proposed architecture covers: High-Level Triggers (HLTs), Data Acquisition (DAQ) and Detector Control System (DCS).

The High Level Triggers provide the elements necessary to reduce the LVL1 Trigger accept rate of 75 kHz (upgradable to 100 kHz) to something of the order of 100–200 Hz.

The Data Acquisition system provides everything necessary to move the detector data from the output of the RODs to mass storage. It also contains the tools needed to initialize, partition and control the experiment data-taking, as well as to monitor the event data and the performance of the systems and subsystems of the experiment. For normal physics running, when the HLTs are used to select the events to go to mass storage, the DAQ system is responsible for serving event data to the HLTs.

The HLT is based on logically separate LVL2 and EF selection stages. This is a deliberate strategy at this point in time, in order to allow for maximum flexibility in the architecture and the event selection sequence. The concept of distributing the selection process between LVL2 and EF is a vital element in the architecture, which allows it to be flexible to changes (luminosity, detector knowledge, background conditions, etc.).

For LVL2, the RoI concept distinguishes between primary and secondary RoIs. Studies have shown that after processing only the primary RoIs, a rejection of about 95% of LVL1 accepted events is achieved. The use of secondary RoIs would allow one to further reduce the rate (by doing more exclusive selections) and to provide additional seeds for the EF processing. Furthermore, sequential selection allows the early rejection of events, minimizing the resources used. The increased demands from B-physics (with respect to the ATLAS TP) can be dealt with using a different approach (beyond the RoI scheme) for subsequent LVL2 processing steps after the confirmation of a low-p_T muon. The LVL2 processing shall use comparatively simple, fast and optimized event selection algorithms to minimize its average latency. Where appropriate, in addition to conventional processors, FPGA co-processors may be used to aid achieving the required performance.

The EF provides final online event selection to achieve an additional rate reduction (even for inclusive triggers, see Chapter 8). To avoid the development, implementation and maintenance of EF specific event selection algorithms, the EF shall maximize the deployment of algorithms developed and implemented by the Offline software system. Results from the LVL2 selection shall be made available to the EF, in the same functional manner as LVL1 and subdetector data. These results may be used to confirm the LVL2 decision; as a starting ‘seed’ for EF processing; to select the appropriate EF algorithms to use for a given event. The full LVL2 output will also be required for quality-control checks of the LVL2 results by the EF. So as to capitalize on the event

reconstruction performed by the event selection process, the EF will also play an important role as a facility for subdetector calibration analysis and monitoring.

Given the recent and continuing advances in networking technology and its deployment, the EF could be geographically distributed, e.g. at the experiment, in the CERN computer centre, or even off the CERN-site [9-3], thus allowing the EF to use standard offline resources as they become available in geographically dispersed institutes. Even in the case that available networking does not provide adequate bandwidth and reliability, specific types of events may still be steered to predefined off-CERN site resources, e.g. for subdetector calibration.

The DCS provides the functionality of: overall supervision and monitoring of the subdetectors; acquisition, analysis and treatment of non-physics-event data e.g. temperatures and voltages; recording of non-physics-event data. DCS has implicit connections to: all subdetectors, including LVL1; elements of HLT/DAQ which require its services; the LHC machine. In addition, it has an explicit dependence on the DAQ system for the purposes of control and information exchange.

9.2.4 Overview of Systems and Subsystems

The proposed HLT/DAQ/DCS architecture is presented in Figure 9-1. It comprises a suite of systems, namely: *DataFlow*, *LVL2 Selection*, *EventFilter*, *Online Software* and *DCS*. The arrows in the figure indicate the direction of dependence between the different systems and subsystems and not the direction of data exchange. The source of the arrow indicates the dependant.[1] Each system contains two or more subsystems, also shown in the figure. The subsystems and the software dependencies between them (e.g. between the Run Control and DataBase, in the Online Software) are introduced later in this section, and described in detail in the subsequent sections. Systems not included in this architecture proposal are shown in dotted lines (e.g. LVL1 Trigger).

The DAQ system comprises the DataFlow and Online Software systems. They collaborate, to meet the data acquisition requirements of normal data-taking and to serve event data to the HLT (more specifically, LVL2 DataFlow serves data to the LVL2 Selection, and EF I/O to the EventFilter). In addition to its function of serving data to the LVL2 Selection, the LVL2 DataFlow subsystem contains components specific to the operation of the LVL2 Trigger. The DataFlow and Online Software systems also collaborate, optionally with the EventFilter (depending on run requirements), to implement the data acquisition and analysis of runs where the LVL2 Trigger element of the HLT is not involved (e.g. cosmic-ray runs and dedicated detector calibration runs). In this case, the LVL2 DataFlow subsystem (see Figure 9-1) has no role to play.

HLT functionality is provided by the LVL2 Selection and EventFilter systems. Data are served to them by the DataFlow system.

The DataFlow system provides the functionality of receiving and buffering data from the subdetector RODs [9-4], distribution of events and event fragments to the HLTs and the sending of events to mass storage. It contains four subsystems: *ReadOut*, *LVL2 DataFlow*, *EventBuilding* and *EF I/O*. The ReadOut Subsystem (ROS) provides the receiving and buffering of subdetector data. The LVL2 DataFlow provides the distribution of the selected event data to the LVL2 Selection

1. As an example, referring to Figure 9-1, the LVL2 DataFlow implements an interface (a set of functions). The ProcessingUnit subsystem depends on this interface. A change in the interface means a change to the ProcessingUnit subsystem. The direction of the data exchange can be specified in the definition of the interface.

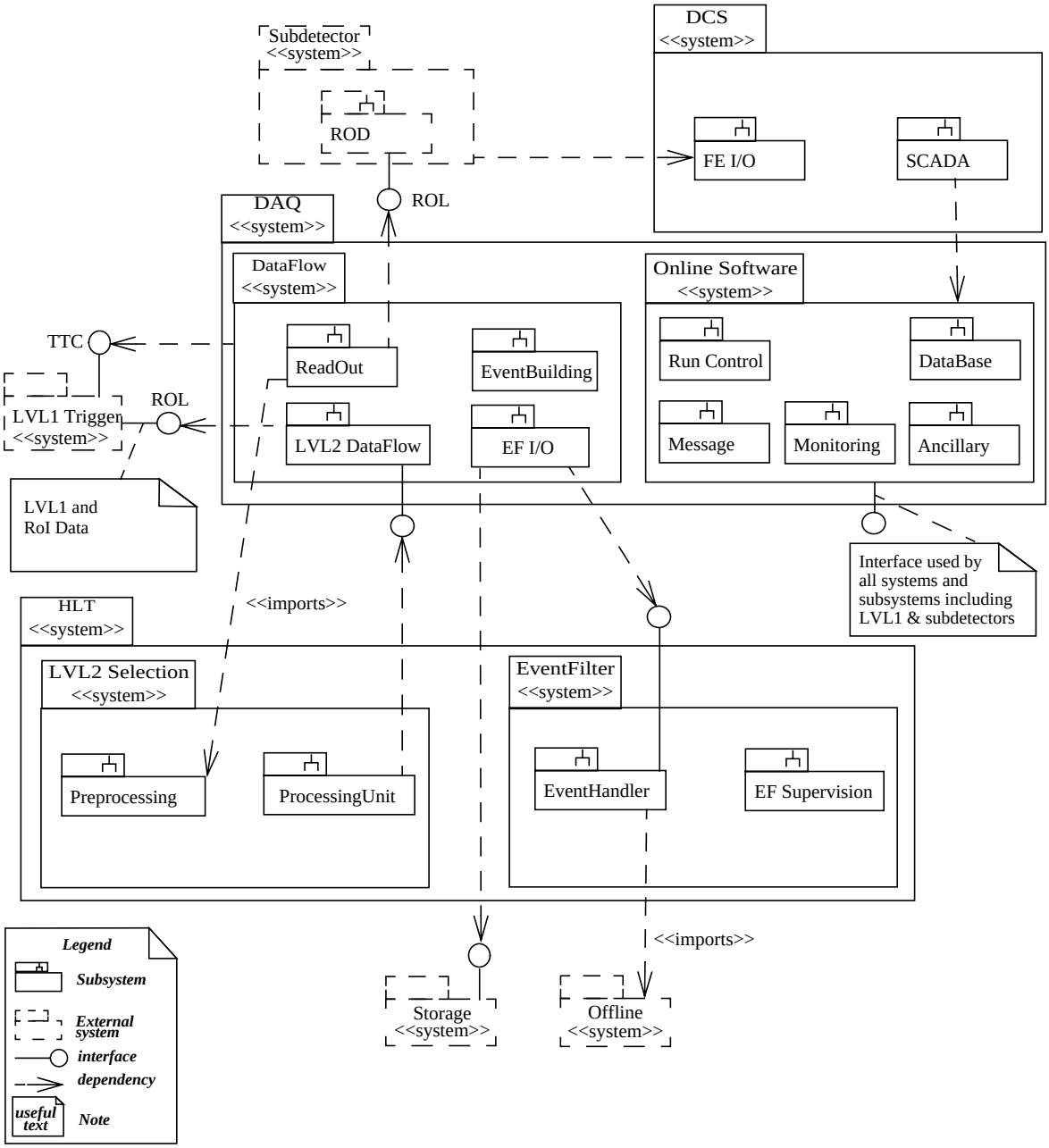

Figure 9-1 Diagram showing the main systems and subsystems of the proposed HLT/DAQ/DCS architecture.

system. The EventBuilding provides the merging of event fragments into complete events and the EF I/O provides the distribution of events to the EventFilter system and the sending of events to mass storage. The dependence of the LVL2 DataFlow on the LVL1 Trigger represents the transfer of RoI information from the latter to the former. The dependence of the DataFlow on the LVL1 Trigger represents the triggering of the ROS and EventBuilding in the absence of a LVL2 Selection (e.g. during a dedicated calibration run). The ROS also provides a sample of the event fragments it has handled, to the Monitoring subsystem (see Online Software system). This is the first stage in the HLT/DAQ/DCS architecture, where subdetector groups are able to monitor the performance of their detector (note that detectors will also perform monitoring in their individual ROD crates). Equivalent functionality is provided by the EF I/O and allows access to

complete events for detector monitoring purposes if the EventFilter is not present. The DataFlow subsystems are further described in Section 9.3.

The Online Software system is responsible for the configuration and control (in collaboration with the DCS) of the detector and HLT/DAQ/DCS systems. To this end it provides configuration, including the management of detector and DAQ partitions; run control; distributed information management; monitoring infrastructure; graphical user interfaces for the purpose of control and configuration. It contains five subsystems: *Run Control*, *Message*, *DataBase*, *Monitoring* and *Ancillary*. It is implicitly understood to have connections to all subdetector systems and other HLT/DAQ/DCS systems. The Online Software system architecture is given in Section 9.4.

The first stage of HLT event selection is provided by the LVL2 Selection. It contains two subsystems: *ProcessingUnit* and *Preprocessing*. The former receives RoI data from LVL1 via the LVL2 DataFlow and steers the processing of each event, requesting and processing selected event data based on the RoI mechanism. The Preprocessing provides the preparation of selected data prior to processing (e.g. zero suppression for the liquid argon calorimeter). The Preprocessing functionality is developed as part of the LVL2 Selection system and may be implemented as a component of the ROS, as shown by the dependency of the ROS on the Preprocessing subsystems. Further details of the LVL2 Selection system architecture are given in Section 9.5.

The EventFilter system provides the final stage of the HLT event selection. It contains two subsystems: *EventHandler* and *EF Supervision*. The EventHandler subsystem provides the functionality of event selection and other ancillary tasks, i.e. physics monitoring, detector monitoring and calibration analysis. The EF Supervision subsystem provides the configuration, control and operational monitoring of the EventHandler. The EventHandler receives built events from, and returns accepted events to the EF I/O (for transfer to mass storage). It imports as far as possible, filtering algorithms and their implementation from the Offline system. The EventFilter system architecture is detailed in Section 9.6.

The DCS comprises two component subsystems, namely: Supervisory Controls And Data Acquisition (SCADA) and Front-End Input/Output (FE I/O). The DCS architecture is described in Section 9.7.

9.2.5 System and Subsystem Interactions

9.2.5.1 Overview

Having introduced the principal systems and subsystems of the architecture above, this section goes on to describe how they interact in a normal physics run and a calibration run. To do this, three collaborations are described: *LVL2 selection*, *LVL2 selected event* and *Calibration event*. A collaboration defines and describes the systems, subsystems and the messages exchanged between them to realize a functionality of the HLT/DAQ/DCS system.

The LVL2 selection collaboration starts with the LVL1 accept decision, the LVL2 selected event collaboration starts when the LVL2 Selection decision is obtained. The Calibration event collaboration is relevant for handling events in standalone operation, e.g. dedicated calibration runs, test beam, subdetector debugging and commissioning.

Prior to any data-taking session (physics run or calibration run), the desired data-taking partition is defined using the Online Software system. After its definition, the data-taking partition

may be configured, used and controlled by the Online Software system functionality. The following description applies to the defined data-taking partition.

The event triggering is performed by an external trigger system (e.g. LVL1 in a physics run, specific detector hardware for a dedicated calibration run) and distributed to subdetector systems by the TTC system (see Chapter 3). The generated trigger initiates the sending of ROD event fragments from the subdetector's RODs, via the ReadOut Links (ROLs), to the associated set of ROSs. In a physics run, the RoIs and associated LVL1 data are transmitted to the LVL2 DataFlow from the LVL1 Trigger system in parallel with the transmission of ROD event fragments to the ROSs. The subsequent steps then depend on the type of data-taking session in question. For a physics run there follows the LVL2 selection collaboration followed by a LVL2 selected event collaboration. In the case of a calibration run there follows a Calibration event collaboration.

The following subsections describe the LVL2 selection, LVL2 selected event and Calibration event collaborations. The figures (UML collaboration diagrams) show the collaborating systems and subsystems and the communications between them. Each communication has a sequence number, shown in the associated figure, which indicates the order in which the communication occurs. The arrow, associated with each communication, points towards the receiving system or subsystem.

9.2.5.2 LVL2 Selection Collaboration

The LVL2 selection collaboration is shown in Figure 9-2. The collaboration consists of the ProcessingUnit, LVL2 DataFlow and ROS subsystems and the LVL1 Trigger system. The following list briefly explains the communication flow in Figure 9-2. The precondition for this collaboration is that there has been a LVL1 accept.

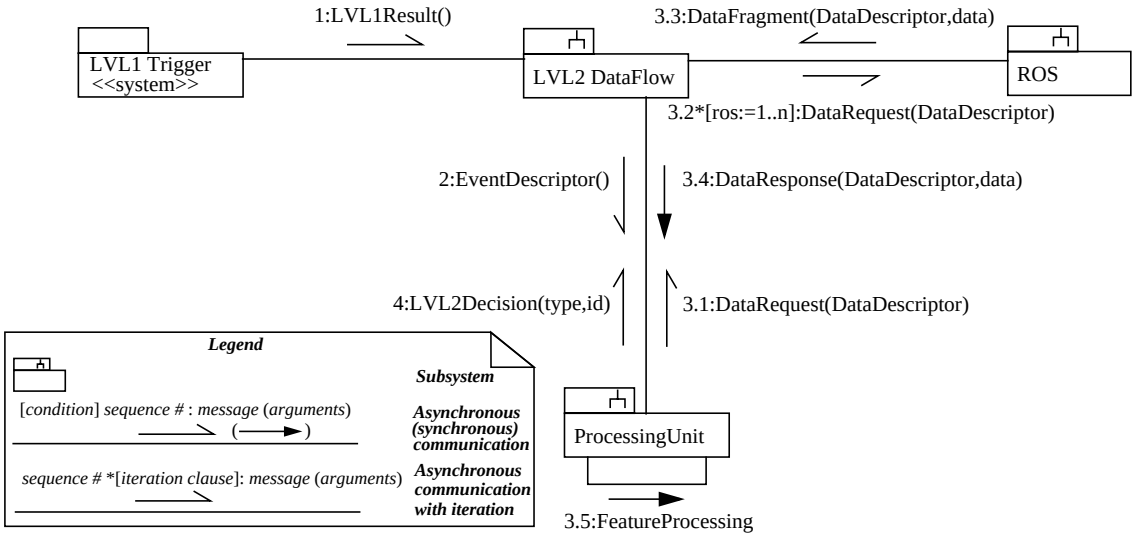

Figure 9-2 LVL2 selection collaboration.

1. The LVL1 result is communicated to the LVL2 DataFlow subsystem. The result consists of an event identifier, RoI and other LVL1 data (see Chapter 4).
2. The LVL2 DataFlow communicates an event descriptor, e.g. the event identifier and RoI η–ϕ range, to the ProcessingUnit subsystem.

3. The communications indicated by the sequence numbers '3.x' identify a sequence of related communications. In this case it indicates the processing of a feature in the event, associated with the communicated event descriptor, by the ProcessingUnit subsystem.

   3.1 The ProcessingUnit requests data from the LVL2 DataFlow. The parameter DataDescriptor identifies the event, subdetectors and detector region from where data is being requested.

   3.2 The LVL2 DataFlow asks all relevant ROSs, as indicated by '*[ros:=1..*n*]', for their data in the detector region defined in the data descriptor.

   3.3 All ROSs that have been asked by the LVL2 DataFlow, communicate the requested data to the LVL2 DataFlow. The latter is responsible for collecting all relevant data from all relevant ROSs in the data-taking partition and providing the resulting, formatted data record to the ProcessingUnit.

   3.4 The single data record is communicated to the ProcessingUnit.

   3.5 The ProcessingUnit performs feature processing on the communicated data record. If additional feature processing is required in the LVL2 processing, the sequence 3.1 to 3.5 is repeated.

4. On completion of the LVL2 processing, the ProcessingUnit subsystem communicates the LVL2 Selection decision to the LVL2 DataFlow. The event identifier and the type of decision are communicated as parameters. The decision type indicates that the event should be rejected or accepted[1] for further analysis.

9.2.5.3 LVL2 Selected Event Collaboration

The LVL2 selected event collaboration is shown in Figure 9-3. The collaboration consists of the LVL2 DataFlow, ROS, EventBuilding, EF I/O and EventHandler subsystems. The following list briefly explains the communication flow in Figure 9-3.

The precondition for this collaboration is that the LVL2 selection, as described above, has been completed and the resulting decision has been communicated to the LVL2 DataFlow by the ProcessingUnit, sequence number 4. The sequence numbers in this collaboration diagram continue from where those in Figure 9-2 finished.

5. LVL2 DataFlow communicates the LVL2 decision to all ROSs in the data-taking partition and, if the type parameter indicates that the event is to be accepted, in parallel to the EventBuilding. This parallelism is denoted by the sequence numbers 5a and 5b. The notation '[type:=accept]' on communication 5b denotes a 'condition' for sending the LVL2 decision to the EventBuilding and should be read as 'if type equals accept'. The LVL2 DataFlow may group decisions prior to communicating them to the ROSs and EventBuilding. In this case an array of event identifiers is communicated in the parameter list. For LVL2 decisions of type reject, the ROSs 'remove' the associated event fragments from their internal data structures and the collaboration sequence finishes.

1. An event could be accepted for two reasons: it has passed the selection criteria and should be analysed by the EventFilter; or it has failed the selection criteria but should be retained for further analysis, e.g. for trigger studies, as explained in Section 9.5.1.

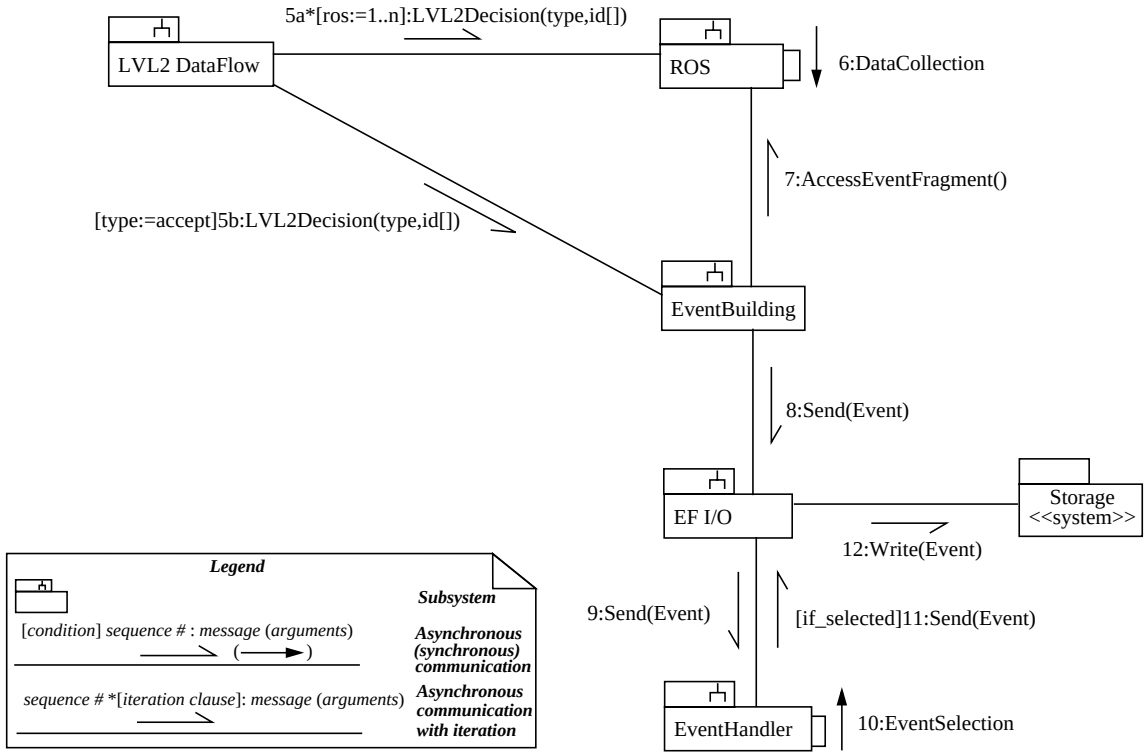

Figure 9-3 LVL2 selected event collaboration.

6. If the LVL2 decision is of type accept, the collaboration continues with this step. The ROS collects all event fragments into a single event fragment ready for access by the Event-Building.
7. The EventBuilding accesses the event fragment in all ROSs and merges them into a single event fragment.
8. In this communication, the EventBuilding communicates a built event to the EF I/O. The EF I/O builds and adds the event header to the event.
9. EF I/O supplies a full event to the EventHandler.
10. The EventHandler performs event selection and/or monitoring and/or calibration processing on the event depending on the event type.
11. If the event is selected by the EventHandler, as indicated by the condition '[if selected]', the event is communicated to the EF I/O. At this point the data in the communicated event may depend on the event type: it may be a physics event with the results of the EventHandler processing; only the reconstructed event, i.e. the raw information removed.
12. EF I/O communicates the event to the Storage system.

9.2.5.4 Calibration Event Collaboration

The Calibration event collaboration is shown in Figure 9-4. It only differs from that of the LVL2 selected event collaboration in that the LVL2 DataFlow and ProcessingUnit play no role. The LVL2 DataFlow is replaced with a subdetector Trigger system as the source of the trigger decision. In the case of a normal physics event or an event related to the LHC beam structure,[1] this

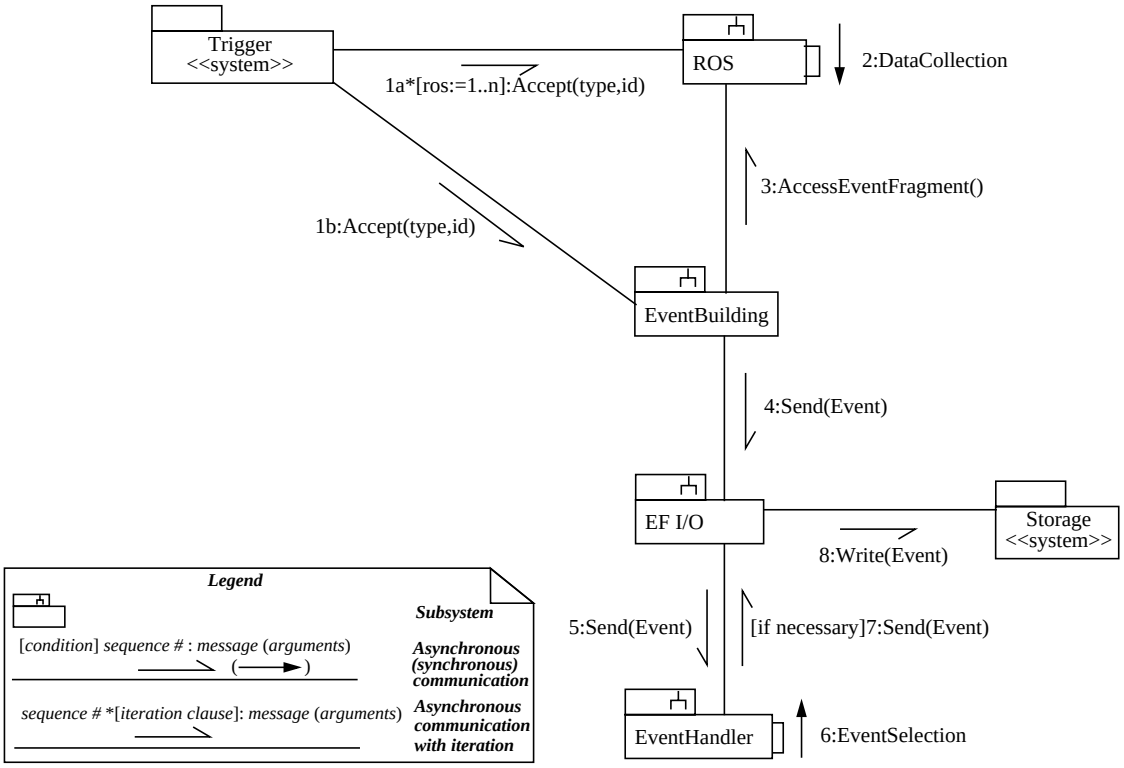

Figure 9-4 Calibration event collaboration.

Trigger system is the LVL1 Trigger. Otherwise it is a specific subdetector trigger system. The Trigger system communicates an accept to all concerned ROSs and the EventBuilding subsystem, in parallel. The parallelism in sending the accept to the ROSs and the EB is shown by the sequence numbers 1a and 1b. The parallel communication of the accept to all ROSs is shown by the condition '*[ros:=1..*n*]' were n is the number of ROSs in the data taking-partition. The subsequent communications are equivalent to those described in Section 9.2.5.4.

9.2.6 Monitoring Aspects

This subsection briefly describes the monitoring functionality offered by the proposed architecture at the level of the EventFilter and DataFlow systems.

The monitoring done at the level of the EventFilter system is characterized by the analysis of complete event data using the most up to date calibration and alignment files. The EventFilter provides monitoring tasks with the framework for control and event access. It also provides the functionality required to control them. The results of the monitoring tasks could be made available to the user directly by the EventFilter, or via the Online Software's Monitoring subsystem. Events can be directed to specific parts of the EventFilter depending on their event type.

In addition to monitoring in the EventFilter, the proposed architecture also allows monitoring tasks to receive event fragments as well as complete events from two stages of the DataFlow: the ROS and the EF I/O. The monitoring tasks use an interface of the Monitoring subsystem to

1. For example, a trigger generated in the LHC beam bunch gap for monitoring or calibration purposes.

access these events or event fragments, and the DataBase subsystem for the storing or retrieving of monitoring results. The Monitoring subsystem in turn uses interfaces of the ROS and the EF I/O to access event fragments or events from the DataFlow, see Figure 9-5. The monitoring performed at this level works with event fragments and pertains more to detector and HLT/DAQ status. It should be noted that detectors will also run monitoring tasks at the ROD-crate level.

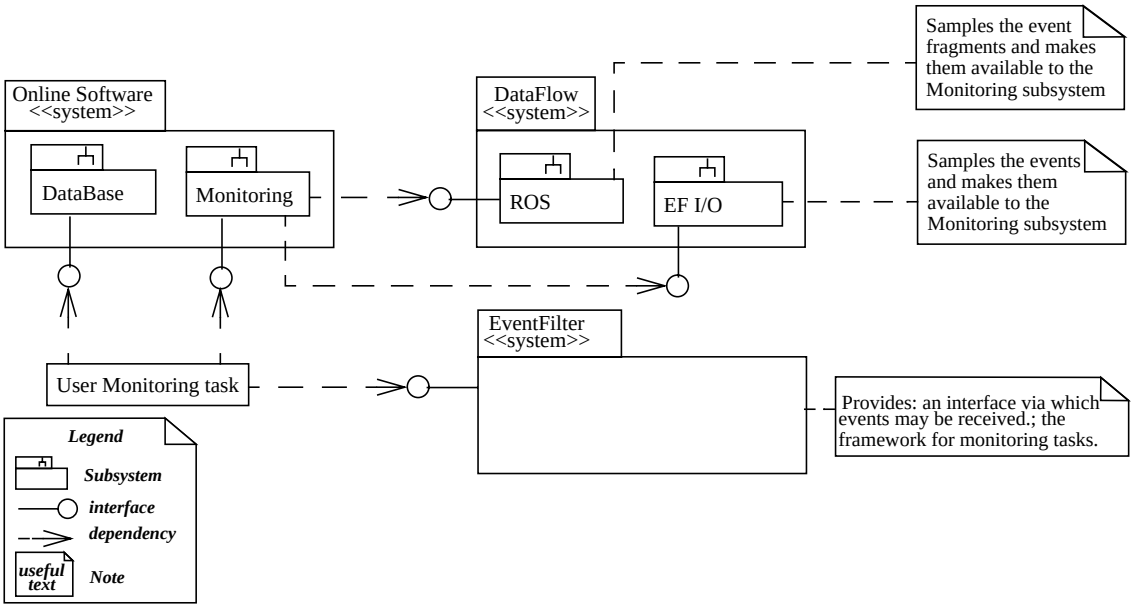

Figure 9-5 Overview of monitoring functionality in the proposed architecture.

The many aspects of the control, coordination and distribution of online monitoring facilities for the experiment need further study, as noted in Chapter 10.

9.2.7 System Partitioning Requirements

This section defines the concept of system partitions and partitioning, and notes a few important requirements, issues and limitations.

During operation of the ATLAS experiment, there will often be a requirement, particularly during detector commissioning and debugging, to use one set of elements of the detector (including possibly DAQ system elements) independently from, and concurrently with another set. Partitioning is defined as the logical and/or physical separation of the experiment into mutually exclusive subsets (partitions) of these elements. An element is an ATLAS system, subsystem or a part of a subsystem, e.g. a subdetector, a ROS or a part of a subdetector.

Certain elements used in the DAQ may possibly be used in many different ways, depending on the details of the configuration of a given partition. For example, a partition consisting of just one subdetector may require customized software for the EventFilter, software different to that used during physics data-taking. This shows the need for the possibility to associate software with hardware on a per-partition basis: a partition should therefore describe the software resources needed to run the DAQ with that partition. A partition is therefore defined as a set of

(hardware) elements corresponding to a certain configuration, and any associated (software) resources needed.

The total number of partitions used for a data-taking session is defined by the number of TTC systems, as each partition will require an independent TTC system and associated dead-time handling facilities for partition triggering. The number of TTC systems currently foreseen in ATLAS is forty. The HLT/DAQ/DCS must therefore be able to support up to forty concurrent data-taking sessions to comply to the partitioning requirements of the experiment's systems. Elements being used in a particular partition cannot be concurrently in use in another partition, e.g. the muon barrel subdetector cannot be used for normal data taking and concurrently for testing.

The issue of system partitioning needs to be studied in detail and has been noted in Chapter 10.

9.2.8 Boundaries and Interfaces

In this section, the boundaries and interfaces between the various subsystems shown in Figure 9-1 and introduced in Section 9.2.4 are described.

1. ROD – ROS: The boundary between these two subsystems has been defined [9-4] as the ROL. The interface to this boundary is the API required by the ROS to receive ROD event fragments via the ROL. The ROD event fragments shall be formatted according to the description given in Ref. [9-5].
2. ROS – EventBuilding: The EventBuilding merges event fragments into full events. To this end, it retrieves, from one or more ROSs, the required event fragments. The boundary between these two subsystems is therefore defined as a buffer containing the event fragments buffered in the ROS. The EventBuilding uses an API to this buffer to retrieve and remove event fragments from this buffer.
3. ROS – LVL2 DataFlow: The LVL2 DataFlow merges the RoI data from one or more ROSs into a single data structure to be communicated to the LVL2 Selection. For this functionality the boundary and interface between the ROS and LVL2 DataFlow is the same as that defined between the ROS and EventBuilding, see above. In addition, the LVL2 DataFlow communicates the LVL2 Selection decision to the ROSs. For this purpose the boundary is defined as a messaging system and its associated API. The details of the messaging system and its associated API are to be defined.
4. LVL2 DataFlow – EventBuilding: For events accepted by the LVL2 Selection, the LVL2 DataFlow communicates to the EventBuilding an event descriptor.[1] Therefore the boundary between these two subsystems is a messaging system and associated API. The details of the messaging system and its associated API are to be defined.
5. LVL1 – LVL2 DataFlow: The purpose of this boundary and interface is to communicate the LVL1 Trigger information (including RoI details) to the LVL2 DataFlow. The boundary and interface between these two systems (defined in Chapter 4 and described in more detail in Ref. [9-6]) is several ROLs and the appropriate API.
6. LVL2 DataFlow – ProcessingUnit: A boundary and interface is required between these two subsystems for the communication of the LVL1 Trigger information (including RoI details) from the LVL2 DataFlow to the ProcessingUnit; the communication of the LVL2

1. For example, trigger type and event ID.

Selection decision to the LVL2 DataFlow; the requesting and receiving of the RoI data to and from the LVL2 DataFlow by the ProcessingUnit. The boundaries and interfaces for these are messaging systems and their associated APIs, the details of which are to be defined.

7. EventBuilding – EF I/O: The EventBuilding communicates events to the EF I/O via the boundary and buffer. The boundary and interface are a buffer and an API, respectively. The EF I/O communicates to the EventBuilding, via the API, the location address where events may be placed within the buffer. The API also allows the EF I/O to query the status of the EventBuilding.
8. EF I/O – EventFilter: The EF I/O communicates events to the EventFilter. The boundary is a buffer from which the EventFilter retrieves events. The interface allows the EF I/O to query the readiness of the EventFilter to receive events and the communication of the location of events within the buffer to the EventFilter.
9. Online Software – 'All Systems': The Online Software exchanges commands, information and database contents with all the other systems for the purposes of control, event data and HLT/DAQ operational monitoring, and configuration. The Online Software interacts with the SCADA system of DCS to ensure consistent control of the overall experiment.
10. DCS – 'All Systems': The DCS exchanges commands, information and database contents with all the other systems for the purposes of configuration, control, non-event data and detector hardware monitoring. It interfaces directly to the appropriate elements of ATLAS systems via the FE I/O subsystem.The SCADA subsystem interfaces to the Online Software system to ensure that the latter performs consistent control of the overall experiment.

9.3 The DataFlow System

The DataFlow system provides the functionality of: receiving and buffering data from the subdetector RODs; building of complete events from individual event fragments; distribution of events and event fragments to the HLTs; the sending of events to final storage. It contains four subsystems: *ReadOut*, *LVL2 DataFlow*, *EventBuilding* and *EF I/O*, which are shown in Figure 9-6 together with the dependencies between them. These dependencies have been described in Section 9.2.8. The functions of these subsystems were briefly introduced in Section 9.2.4. This section now presents each subsystem in detail.

9.3.1 The ReadOut Subsystem

9.3.1.1 Introduction

The ROS is the subsystem of the DataFlow which provides subdetector data reception, buffering and distribution. Section 9.3.1.3 presents a baseline architecture for the ROS based on the conclusions from the DAQ/EF-1 ReadOut Crate (ROC) and the Pilot-Project ROB Complex studies, see Section 9.3.1.2. The architecture described is incomplete in that some model views are missing, however the fundamental concepts are presented.

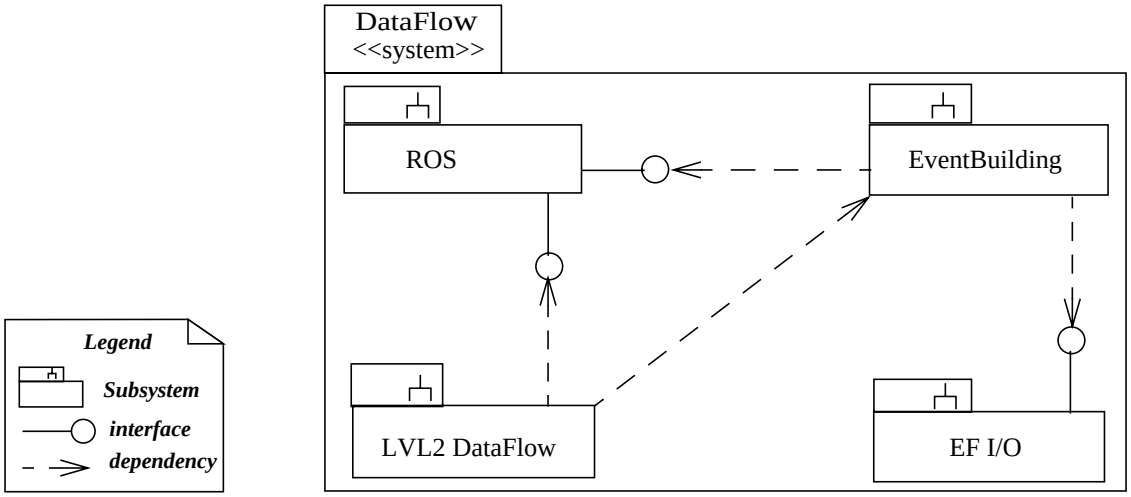

Figure 9-6 A package diagram showing the DataFlow subsystems and their dependencies.

9.3.1.2 Main Conclusions from the ROC and ROB Complex Studies

The following lists the main conclusions of the work done in the DAQ/EF-1 project on the ROC and in the LVL2 Pilot Project on the ROB Complex. More details and explanation of terms can be found in Chapters 5 and 6. The final subsection discusses these conclusions.

9.3.1.2.1 ROC Conclusions

- Cornerstones of the DAQ-Unit design are Tasks, message passing and I/O Modules (IOMs).
- The Local DAQ (LDAQ) subsystem, providing all the common DAQ functions which are not related to moving event data, allows the possibility of operating parts of the DataFlow (e.g. a single ROC) in isolation for testing and debugging purposes.
- The LDAQ has proved to be a resource-consuming task in terms of CPU time, memory and networking. This justifies the model of separating it from the DAQ-Unit.
- A ReadOut Crate and SubFarm Crate based on the DAQ/EF-1 design and the deployment on Commercial Off-The-Shelf (COTS) products, capable of meeting the ATLAS requirements, is within reach.
- The use of COTS products has decreased the resources required during the design and deployment.
- Based on today's technologies and the potential requirement of more than a single ReadOut Link per ReadOut Buffer, the ROBIN implemented on a mezzanine card has shown to be an important concept, if the ReadOut Buffer input requirements are to be met.

9.3.1.2.2 ROB Complex Conclusions

- Requirements can be met with current technology.
- A compact design of the ROBIN is achievable, though non-commodity off-the-shelf components may be necessary. Several workable design options have been shown to exist.

- There are advantages to grouping buffers in a ROB Complex. Depending on the evolution of technology, the grouping factor could be from one to a few tens. The proposed architecture must be flexible in this respect.
- Preprocessing gives the important possibility to reduce the data volume to be transmitted to LVL2 processors. On-the-fly preprocessing has been demonstrated using spare processor capacity (FPGA, conventional processors).

9.3.1.2.3 Discussion

The emphasis in the DAQ/EF-1 project has been on the required functionality and design, while the Pilot Project has focused on how the data-flow performance requirements can be satisfied.

The above conclusions refer both to *commercial* and *commodity* products. The latter is a subset of the former and refers specifically to products available from general purpose suppliers, e.g. PCs, while *commercial* also includes specialist products, e.g. VMEbus. Both sets of conclusions are consistent with the statement that Commercial Off the Shelf Products may meet the performance requirements. The exclusive use of commodity products does not allow the performance requirements to be met, particularly in the area of input to the ROS and preprocessing.

Both projects have identified and recognized the concept of a ROBIN as being an important element in achieving the ROS performance requirements. The conclusions of the two projects lead to the architecture described in the next section.

9.3.1.3 ROS Architecture

9.3.1.3.1 Overview

This section presents the proposed architecture for the ROS. It is based on the conclusions outlined above and the description of the boundaries and interfaces between the ROS and other DataFlow subsystems and HLT/DAQ/DCS systems given in Section 9.2.8.

9.3.1.3.2 Major Use Cases

The *users* of the ROS are the following: Subdetector ROD, Subdetector Trigger, LVL2 DataFlow, EventBuilding and Online Software, as shown in Figure 9-7. The following sections describe what the ROS must do for each of these users.

- The Subdetector ROD

  The ROS must receive and buffer ROD event fragments from one or more RODs. These fragments are sent at the LVL1 accept rate of 75 kHz (upgradable to 100 kHz). For each LVL1 accept there is one ROD fragment per ROD and the size of these fragments may be up to 1800 bytes on average. Therefore a ROD requires that the ROS can receive and buffer data at up to 135 Mbyte/s (upgradable[1] to 160 Mbyte/s). The ROD–ROS boundary is the ROL, see Section 9.2.8. The control of the flow of data between the ROD and the ROS is ensured by the ROL.

1. For a LVL1 accept rate of 100 kHz, the RODs shall reduce the average event fragment size to 1600 byte.

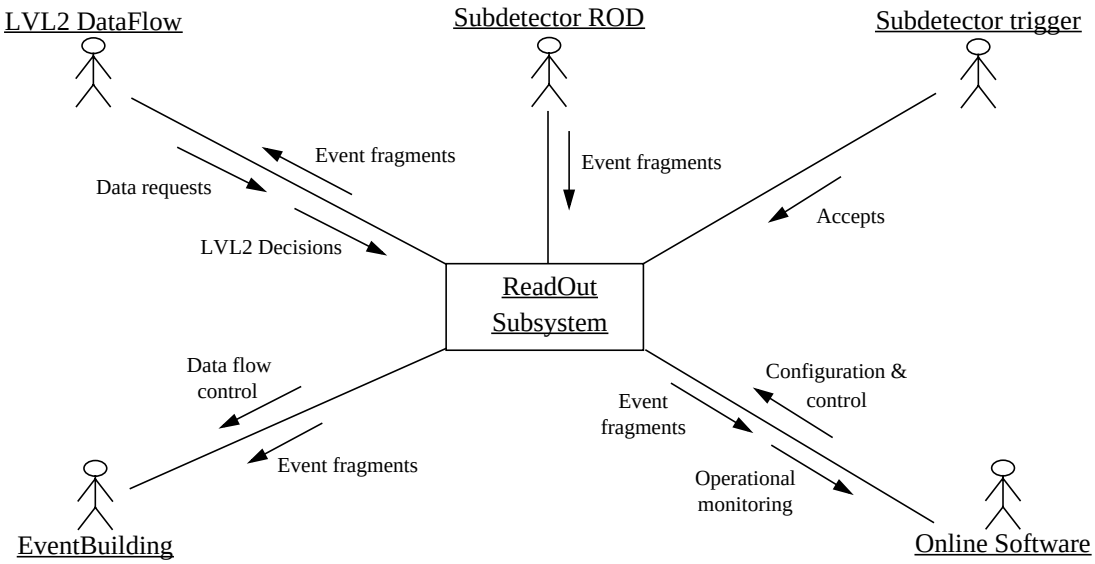

Figure 9-7 The ReadOut subsystem context diagram.

- LVL2 DataFlow

  The interface between LVL2 DataFlow and the ROS provides the LVL2 DataFlow with access to one or more event fragments. The interface allows event fragments to be located in the ROS buffer(s) and / or copied from the buffer(s). The rate at which the LVL2 DataFlow accesses a ROS may be up to 15–20 kHz.

  The ROS receives the LVL2 Selection decision (reject or accept event). For each reject decision the ROS removes event fragments from its buffer(s). The LVL2 DataFlow provides the ROS with the event identifier of event fragments to be rejected or accepted.

  Events may be rejected by the LVL2 Selection, at up to a rate of 100 kHz, and accepted events may have a rate of a few kHz. These rates depend upon the apportioning of processing between the LVL2 Selection and the EventFilter, as well as the running conditions.

- EventBuilding

  For each event accepted by the LVL2 Selection, the ROS collects all event fragments associated with the event identifier and trigger type into a single event fragment[1] and provides the EventBuilding with access to this event fragment, see Section 9.2.8.

- Online Software

  The ROS is configured, controlled and has its operation monitored by the Online Software system. During the configuration phase it receives and uses parameters sent by the Online Software system that define its mode of operation. The ROS provides the Online Software system with status and error information, enabling its operation to be monitored. In addition, the ROS provides the Online Software system with a sample of event fragments for the purposes of subdetector monitoring.

1. The terminology used here is that described in Ref. [9-5]: *ROS event fragment* is used instead of *ROC event fragment* for clarity in this document.

- Subdetector trigger

  It should be possible to trigger actions on an event in the ROS by a trigger system which is specific to a subdetector. This functionality is required in configurations which do not use the central LVL1 Trigger or the LVL2 Selection systems in the configuration. For example, in configurations used for test beam, subdetector commissioning, production testing and dedicated calibration runs.

  The subdetector trigger systems will distribute the trigger signal using the TTC system, (see Chapter 3). The ROS should receive the TTC information associated with the partition of which it is a member.

9.3.1.3.3 Main Functional Elements

The high-level designs of the ROC and the ROB Complex have a degree of commonality: both designs are based on asynchronous I/O (e.g. event fragment input and requests for data); ROB event fragments are stored and accessed (locate, delete, retrieve) by an EventManager; the elements in each of these designs communicate by some form of message exchange.

In addition to the elements mentioned above, the ROC design identifies the concept of a Task. Tasks are connected to one or more asynchronous I/O channels and are scheduled by a Scheduler. Examples of I/O channels are input from a ROL, output from the ROC and communication between Tasks. The ROC design also has a component which provides: configuration, control, operational monitoring, interfacing to the online control software and the sampling of event fragments for monitoring purposes.

The ROB Complex design identifies: a ROBIN for receiving and buffering ROD fragments; a network interface for receiving data requests, trigger decisions and sending data replies; a ROB Controller for managing transfers of data from a ROBIN to a network interface and of requests and decisions from the network interface to ROBINs. Preprocessing of event fragments can be done in the ROBINs and/or in the ROB Controller.

Based on the above points, and the major use cases presented in the previous section, this section now presents the main functional elements of the ROS.

The ROS is composed of two packages:[1] a general *I/O Module (IOM)*, and a *LocalController*, see Figure 9-8. The IOM contains the *DataHandler* package. The DataHandler provides all the functionality related to the handling of event fragments: reception, buffering and distribution. The LocalController implements the interface to the Online Software system, see Section 9.2.8, and therefore provides the functionality required for configuration, control and operational monitoring within a ROS. In collaboration with the IOM, it also communicates samples of the event fragments to the Online Software system for the purpose of subdetector monitoring. The ROS is the first stage of the proposed HLT/DAQ/DCS architecture where subdetector data can be accessed for monitoring.

The IOM provides a well-defined ROS functionality associated with the interfacing of the ROS to an external system or subsystem. In Figure 9-8 it can be seen that the ROS accesses four external systems and subsystems (excluding the Online Software system); correspondingly, four specialized IOMs are derived from the general IOM:

1. A collection of related elements.

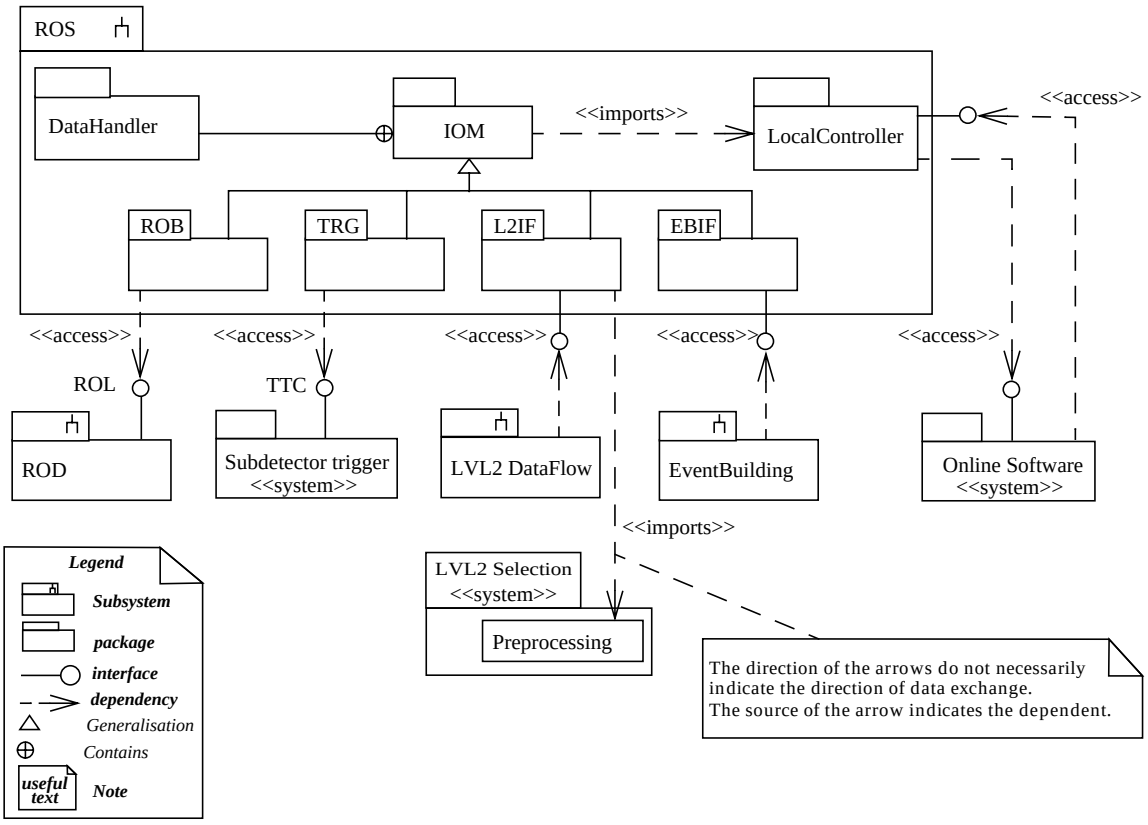

Figure 9-8 Diagram showing the ReadOut subsystem, its three packages and their relationship to other HLT/DAQ systems and subsystems.

- The ReadOut Buffer (ROB) is connected to the external I/O with one or more subdetector RODs. It provides the functionality to receive and buffer one or more ROD event fragments and provides other IOMs access to these event fragments.
- The EventBuilding InterFace (EBIF) is connected to the external I/O with the EventBuilding subsystem and provides the EventBuilding with access to event fragments within the ROS.
- The LVL2 DataFlow InterFace (L2IF) is connected to the external I/O with the LVL2 DataFlow. It provides the same functionality as the EBIF as well as the preprocessing of event fragments and the reception of decisions from the LVL2 Selection. The dependence, labelled imports in Figure 9-8, shows that preprocessing is provided by the Preprocessing element of the LVL2 Selection system.
- The TRG IOM is connected to the external I/O with a Subdetector trigger system. It receives and distributes trigger information within the ROS in the case where the LVL2 Selection is inactive.

Though each derivative of an IOM provides a different and well-defined functionality, IOMs have a common design with respect to the handling of event data, configuration, control, operational monitoring and interfacing to the Online Software system. The common design is provided by the DataHandler and LocalController.

The main functional elements of the DataHandler are shown in Figure 9-9. The central element is a Task.[1] It is connected to one or more asynchronous I/O channels: external input or output,

e.g. input from a ROD; internal communications, i.e. communications between Tasks. A Task provides all the functionality necessary to handle its associated I/O channels, e.g. receive ROD event fragments from a ROL and put them in a buffer. A *Scheduler* is responsible for scheduling[1] one or more Tasks.

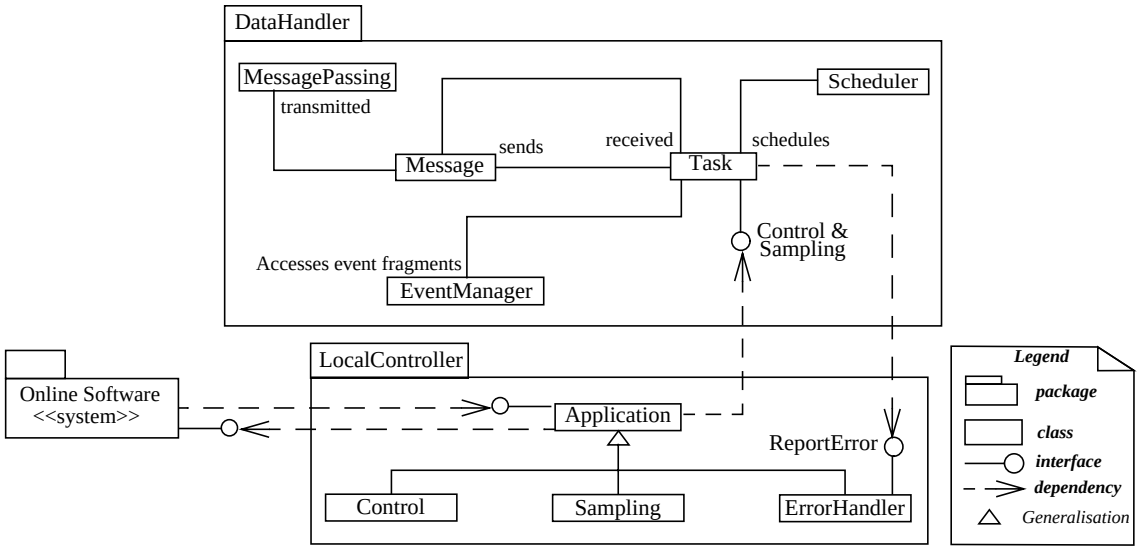

Figure 9-9 Diagram showing the main functional elements of the DataHandler and LocalController.

Figure 9-9 also shows that a *Task* sends and receives a *Message* and that a Message is transmitted by *MessagePassing*.[2] The latter provides a unique interface between all possible Tasks. For the purposes of configuration, control and operational monitoring, Task implements an interface (labelled Control in the figure) which is used by the LocalController. Additionally, a Task uses an interface implemented by the LocalController (labelled ReportError in the figure) for the purpose of reporting error information. A Task may need to receive, buffer and access (i.e. locate, delete and copy) event fragments. This functionality, buffering and access is provided by an *EventManager*.

The main functional elements of the LocalController are also shown in Figure 9-9. It consists of three specific elements: Control, Sampling and ErrorHandling. These elements are derived from a generic element, *Application*. The latter, and therefore its derivatives, implements an interface which is used by the Online Software for the purpose of configuration, control and for providing the Online Software with a sample of event fragments. The Application uses an interface of the Online System for asynchronous communications, e.g. error reporting. The Control element provides all the functionality associated with run control: It provides the interface to the Run Control subsystem (of the Online Software system); it implements a finite-state machine and maps it to that of the Run Control subsystem; it controls all Tasks within a ROS for the purposes of run control; it provides the functionality required to control a ROS in the absence of the Run Control subsystem, e.g. for detector commissioning and standalone use. The ErrorHandler element receives asynchronous error messages from Tasks within the ROS and communicates

1. The term *Task* does not, at this stage, imply lightweight threads or heavyweight threads (processes).
1. Exactly how Tasks are scheduled is an implementation issue.
2. Communication may be via shared memory or data, message passing or operating system primitives.

them to the Online Software system. The sampling element collaborates with the DataHandler to communicate event fragments from the ROS to the Online Software system.

In summary, the main functional elements described above collaborate to provide the ROS functionality described in Section 9.3.1.3.2. The Tasks of an IOM collaborate to provide a specific functionality of the ROS. One or more IOMs then collaborate to provide the full ROS functionality and a number of ROSs are used to receive and buffer subdetector event data.

As ideas change and the detailed design evolves, the IOM design allows derivatives of the specific IOMs (e.g. ROB) to be defined and designed re-using, where appropriate, the designs of existing Tasks. The proposed architecture provides a framework in which different implementation alternatives can be studied up to the TDR.

9.3.1.3.4 Possible Deployment

Given that final decisions do not have to be made at the present time, it is important that the proposed ROS architecture be open, i.e. it should not target a specific hardware implementation. In the ROC and the ROB Complex work, different deployments have been studied. The main functional elements, described above, are consistent with the diverse deployment possibilities currently being considered: complete use of commercial products, i.e. VMEbus system; use of commodity products, i.e. PCs; mixture of commercial and / or commodity products and proprietary components. The use of proprietary components may be required to meet the demanding input requirements from the subdetector RODs and the LVL2 Selection preprocessing requirements.

A common conclusion of both the ROC and ROB Complex work has been the identification of a ROBIN as being important to the design and implementation. The ROBIN provides a part of the ROB functionality: It receives and buffers ROD fragments from a ROL and marshals access (location, retrieval and deletion) to ROB fragments. One or more ROBINs may be integrated in a ROB. In the proposed architecture of the ROS, the ROBIN is achieved by the combination of one or more Tasks and an EventManager per ROL. Other Tasks within the ROS access event fragments in a ROBIN via an API, e.g. a Task servicing Data requests from the LVL2 DataFlow accesses event fragments within the ROBIN via this API.

Each of the IOMs described above could be implemented as a single process and subsequently one or more deployed per processor, depending on the number of processors in the system. Alternatively, one or more of the IOMs described above could be merged to produce a single process, i.e. the L2IF could be merged with one or more ROBs. The resulting single process could be deployed on a processor instead of two, thus avoiding the potential overheads involved in context switching. The optimal merging and deployment should be extensively studied with implementations and / or modelling. Similarly, the number of IOMs of a specific flavour, e.g. the number of L2IFs per ROS, should also be studied.

9.3.2 LVL2 DataFlow Subsystem

9.3.2.1 Introduction

The LVL2 Dataflow is the subsystem responsible for handling the flow of events[1] into and out of the LVL2 Selection system and the flow of detector data requested by this system. For each

LVL1 accepted event, information fragments are assembled from the different components of LVL1. The event is then passed to the LVL2 Selection system and when the LVL2 decision is returned, the decision is transmitted to the ROS and in the case of accepted events, also to the EventBuilding. During the LVL2 Selection processing of each event, each request for detector data (e.g. all data within a given RoI and detector) is received by the LVL2 DataFlow. The latter then distributes requests to all of the appropriate ROSs and builds a single block of data which is returned to the LVL2 Selection. The LVL2 DataFlow subsystem has to be capable of handling events at the maximum LVL1 Trigger rate of 75 kHz (upgradable to 100 kHz).

9.3.2.2 Main Conclusions from the Pilot Project

Within the Pilot Project, prototype Supervisor and RoI Builder (RoIB) components were implemented and used to study the event control aspects of the LVL2 Dataflow (as described in Section 6.5). The concept used is for the RoIB to receive the LVL1 information fragments, assemble them into a single record and pass this to a selected processor in a small farm of Supervisor processors. A prototype RoIB has been built using FPGAs in a highly parallel architecture. It was concluded that:

- Such a design, using custom hardware for the most demanding tasks, reduced the demands on the other Supervisor components, so that they can be implemented using standard processors.
- Such an RoIB plus a small Supervisor farm can satisfy the requirements for the LVL2 Trigger, i.e. a rate of up to 100 kHz.

Within the Reference Software and the Testbeds, the servicing of data requests was handled by the ROB proxy in the LVL2 processing node, together with the request proxy in the ROB Complex emulator and the network components involved in the communication between them. The ROB proxy used a technology-independent message passing interface for access across the network to the request proxy in the ROB Complex emulator. The ROB proxy and associated tools were responsible for the supply, conversion, formatting and preparation of data which was delivered in the form of objects as required by the algorithms. The Reference Software has been run in many configurations, using three network technologies (ATM, Fast and Gigabit Ethernet, and SCI) and over MPI on a 96-node commercial cluster. Processors in Testbeds using this architecture have demonstrated:

- Data collection within an RoI from a single detector at an acceptable rate.
- A three-step sequential selection strategy with prototype algorithms running on a multi-node testbed with the processor requesting simulated event data from ROB emulator nodes.
- The validity of the request/response based architecture.

For the network technologies it was concluded that:

- ATM and Ethernet, as commodity items, are particularly interesting candidates for ATLAS.
- Using optimized drivers and discarding the TCP/IP stack gave the high-node I/O data rates required.

1. *Flow of events* refers to the flow of EventDescriptors and the corresponding returned LVL2 Decisions, not the detector data for the events.

- In both of these technologies the cost of a large network (~ 1000 ports), including switches and host adapters, appears to be consistent with the ATLAS cost estimates.

9.3.2.3 LVL2 DataFlow Architecture

9.3.2.3.1 Major Use Cases

The *users* of the LVL2 DataFlow are the following: LVL1 Trigger system, LVL2 Selection system, ROS, EB subsystem and Online Software system, as shown in Figure 9-10. The following sections describe what the LVL2 DataFlow must do for each of these users.

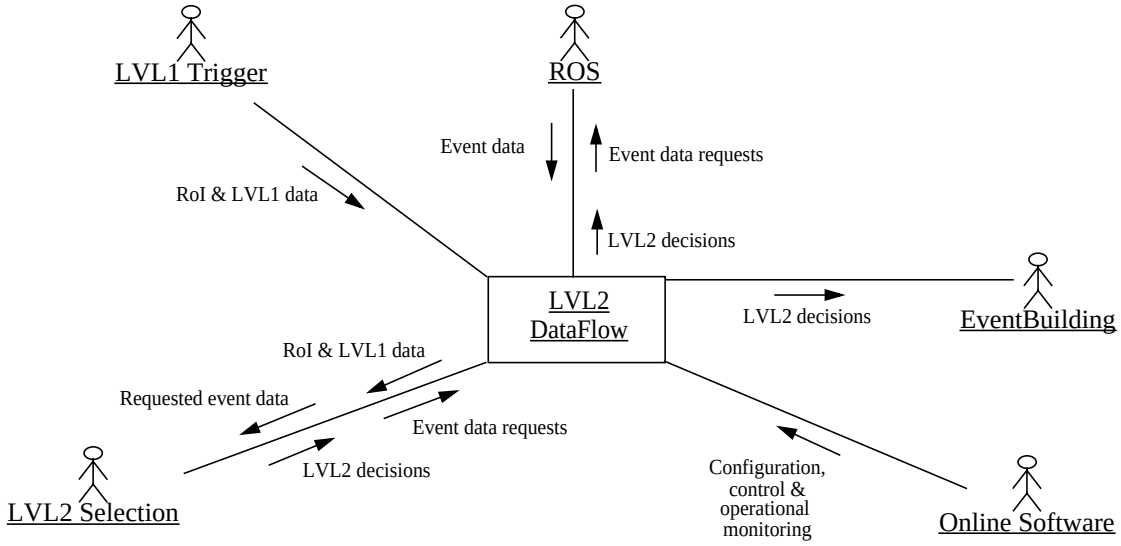

Figure 9-10 LVL2 DataFlow subsystem context diagram.

- The LVL1 Trigger

  The LVL1 Trigger is composed of the central trigger processor, muon trigger and calorimeter trigger. For each LVL1 accept the different parts of the LVL1 Trigger send information fragments over ReadOut Links (ROL) to the LVL2 DataFlow. The fragments, which contain the event identifier, information about the RoIs and other LVL1 data, must be assembled into a single EventDescriptor record for each event for later use by the LVL2 Selection system.

- The LVL2 Selection

  The LVL2 DataFlow sends the EventDescriptor record to the LVL2 Selection system and waits for the LVL2Decision to be returned. The LVL2 DataFlow is also responsible for load-balancing within the LVL2 Selection system. Thus it assigns each event to be processed to a specific ProcessingUnit within the LVL2 Selection system. The assignment algorithm can take into account such factors as the number (and type) of events currently outstanding on each ProcessingUnit. This also allows some monitoring of the LVL2 Selec-

tion by noting ProcessingUnits which have failed to return a LVL2Decision within a time-out period from the sending of the EventDescriptor.

In addition, during the processing of an event by the LVL2 Selection, each time that detector data are required the LVL2 DataFlow receives a request with a DataDescriptor which includes the event id, the detector id and appropriate RoI information. The LVL2 DataFlow has to collect the required data and return it to the LVL2 Selection as a single record, suitably formatted for the selection algorithms.

- The ROS

  Whenever the LVL2 DataFlow receives a request for detector data, from the LVL2 Selection, it determines which ROSs need to be sent requests for data, send these requests and wait for the data to be returned. It assembles the data fragments into a single suitably formatted block to be returned to the LVL2 Selection.

  When the LVL2 DataFlow receives a LVL2Decision it sends this to all of the ROSs. The ROSs use this information, for example, to delete rejected events. For performance reasons these decisions will probably be grouped.

- The EB

  When the LVL2 DataFlow receives a LVL2Decision, which is an accept, it is communicated to the EB for further action.

- The Online Software

  The LVL2 DataFlow communicates with the Online Software for initialisation and configuration parameters, run control, error reporting and monitoring.

9.3.2.3.2 Main Functional Elements

The functional elements of the LVL2 DataFlow are the RoIB, Supervisor and Selective Data Collection (SDC). These and their external connections are shown in Figure 9-11. Note that the Steering and Feature Extraction are parts of the ProcessingUnit, described in the LVL2 Selection (Section 9.5) below.

- Region of Interest Builder (RoIB)

  The RoIB is the architectural element that, on receipt of a LVL1 Trigger accept, collects RoI fragments from the LVL1 system and assembles them into an RoI record, which it transmits to a Supervisor processor.

  The RoIB deals with the following system elements and messages:

  - LVL1 Trigger

    External system composed of the central trigger processor, muon trigger and calorimeter trigger.

  - Supervisor Processor

    One of an assemblage of fast processors comprising the Supervisor.

  - RoI fragment

    An item transmitted by the LVL1 Trigger components containing information relating to features in the event or other information required by LVL2. The LVL1Result is the complete set of RoI fragments for a LVL1 accept.

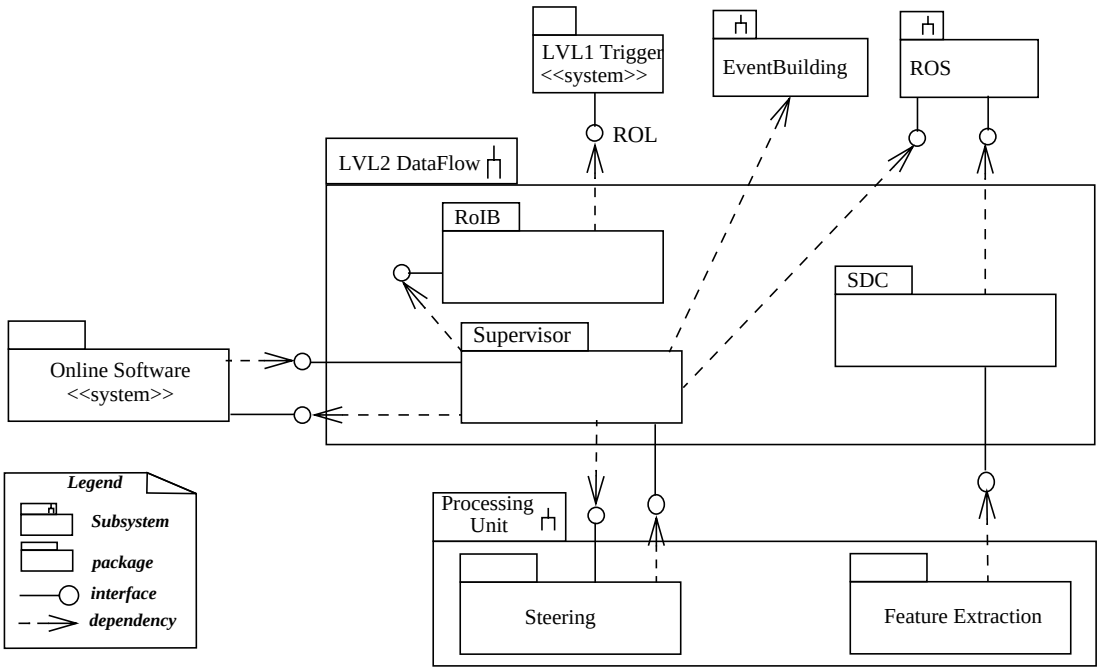

Figure 9-11 Class diagram showing an organisation of the LVL2 DataFlow.
SDC = Selective Data Collection

- RoI record

  Output record generated by the RoIB containing RoI information for an event, for use in the LVL2 Selection process. It contains the complete LVL1Result.

- Supervisor

  This is the architectural element that receives the RoI record from the RoIB, selects a ProcessingUnit to make the LVL2 Selection, transmits an EventDescriptor (with the RoI record) to a Steering process within that ProcessingUnit and awaits the results of its analysis. After the LVL2 Selection is complete, the Supervisor issues appropriate instructions to the ROSs to clear the event or transfer it to the EventBuilding. It has the further function of keeping track of available LVL2 processors, handling error conditions and gathering monitoring information.

  The following system elements are affected by the supervisor: ROS, RoIB, EventBuilding, LVL2 Selection (executed in a farm of fast processors) and Online software.

- Selective Data Collection (SDC)

  This architectural component is responsible for obtaining data from the ROS for LVL2 processing, both when using RoI guidance and for complete detector data collection (for full scans).

  Details of a possible internal structure for the SDC are shown in Figure 9-12. When Feature Extraction inside the ProcessingUnit requires subdetector data it issues a DataRequest (1), with a DataDescriptor giving the event identifier, the subdetector and the part of the subdetector (e.g. RoI position and size, or complete detector) required. The request is forwarded (1.1) by the DataCollector to the ROS Proxy. For RoI data requests the identifiers of the ROSs where the data should be located are determined by accessing (2, 2.1) a ROS ID Look-Up-Table (LUT). This matches the η–ϕ window of the re-

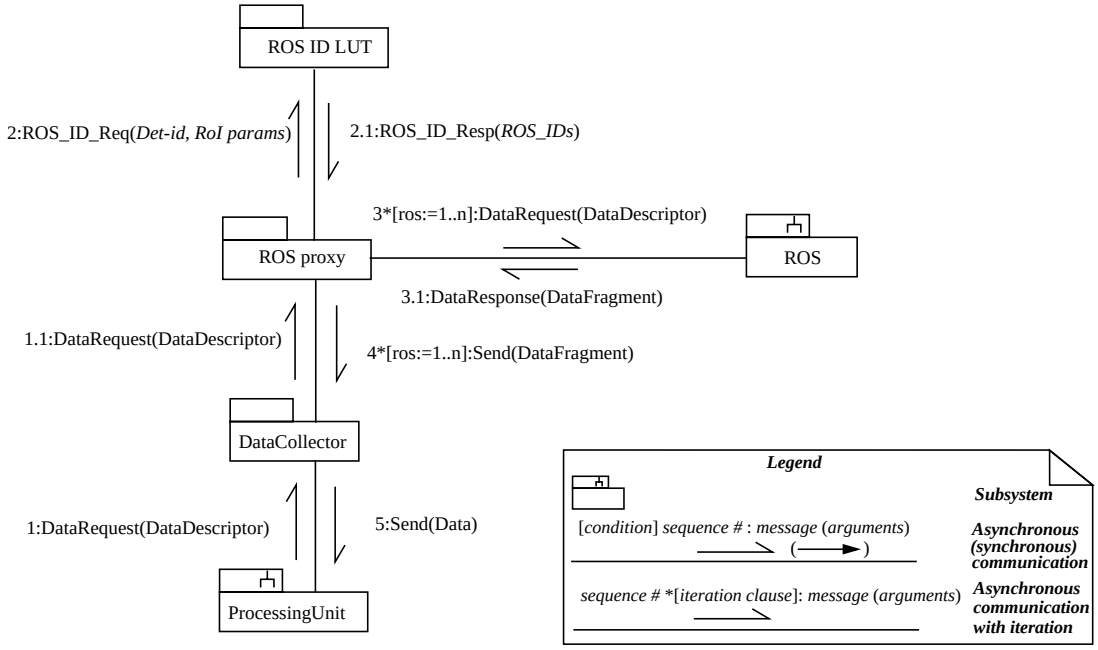

Figure 9-12 Collaboration diagram showing an organization of the Selective Data Collection. Here the Selective Data Collection is decomposed into a Data Collector, a ROS proxy and ROS ID LUT.

quest against a detector-dependent list of the known η–ϕ coverage of the ROSs. For each overlap of windows a request (DataRequest) is sent to the appropriate ROS. After all requests (3) have been made, responses (3.1:DataResponse), each with a DataFragment from a single ROS, are awaited and built into a reply list in the DataCollector (4). When all information has been received, the elements of the reply list are merged and re-formatted by the DataCollector into a single Data record. The Data record is then passed back to the ProcessingUnit (5). If any ROSs do not return data within a timeout period, then null records are inserted in the reply list and the information is flagged as being incomplete.

For the full-scan case, the procedure is very similar, except that requests are non-selectively sent to all ROSs serving data of a particular detector. A reply list is built and contents merged and re-formatted as for the RoI case, but of course the size of the data record is now significantly larger.

9.3.2.3.3 Possible Deployment

The implementation details of the RoIB and Supervisor remain open, but the isolation of the custom high-frequency components to the RoIB promises a cost-effective solution. The type of processors and organization for the Supervisor remains an open option for study.

The implementation of the SDC includes a data gathering and formatting function local to the LVL2 processor performing the Feature Extraction (this could be in the processor itself or may be delegated to an intelligent network interface card attached to the processor); the network software and hardware; an interface local to each ROS.

The communications between the large number of ROSs and the many processors used for the LVL2 Selection require low latencies and high message rates. This can be provided by commodity network hardware, but currently only by using optimized drivers. A single network would

provide a simple solution for all communications between the Supervisor, the LVL2 processors and the ROSs.

9.3.3 EventBuilding Subsystem

9.3.3.1 Introduction

The EventBuilding is the HLT/DAQ/DCS architecture element which is responsible, in response to a signal from a trigger element, for collecting all the event fragments relevant to a bunch crossing into a complete event at a single location. This may be qualified differently according to the context in which the EB is used, as exemplified by the Use Cases in Section 9.3.3.2.

This section defines the EB from a high-level, architectural, point of view. It builds on the results of both the DAQ/EF-1 project and the LVL2 Pilot project, to provide a common architectural framework in which different implementation alternatives can be studied as necessary.

In general, two patterns of interaction, coordinated by a DataFlow Manager (DFM), between EventBuilding sources and destinations can be identified, depending on which initiates a protocol transaction. In this respect the EB protocol may be one of two types.

- Source-initiated transaction: An EB source is asked to send an event fragment to a destination; the source obtains from the DFM, the identity of the destination and transfers the fragment to it.
- Destination-initiated transaction: An EB destination is assigned an event by the DFM and subsequently asks the relevant sources to transfer their event fragments to it.

To make this section complete, we recall the main requirements of the final ATLAS system as defined in the ATLAS TP, summarized in Table 9-1.

Table 9-1 Full event builder parameters.

Fragment size / source[a]	1–15 kbyte	Number of sources[a.]	100–1500
LVL2 Accept rate	1–5 kHz	Number of destinations	100–200
Data rate/source[b]	1–75 Mbyte/s		
Total data rate	1–8 Gbyte/s	Number of nodes	200–1700

a. Assumes 1–15 ROBs/source and 1 kbyte/ROB
b. Assumes data from all sources

9.3.3.2 Main Conclusions from DAQ/EF -1 and the Pilot Project

9.3.3.2.1 The DAQ/EF -1 Project

The EB has been extensively studied [9-7] in the DAQ/EF-1 project. A high-level design for the EB element [9-8] has been produced, which identifies the following components: source and destination processes, the Data Flow Manager (DFM), the switching network, and the Local

DAQ (LDAQ) which provides local control and operations monitoring. Based on the design and implementations, the results and conclusions are summarized in Chapter 5. The following principal points should be underlined:

- A source-initiated protocol has been chosen and studied to define the interaction between sources, destinations and DFM. This choice is justified by the following considerations: it is the continuation of the protocol for the flow of event data; events can be sent to any destination; and a push-like protocol is amenable to a simpler implementation.
- The design has been successfully implemented into an event builder capable of running on top of several switching-network technologies. The design and implementations are believed to be excellent candidates for the initial definition of the event building in the continuation of the ATLAS HLT/DAQ/DCS project.
- Ethernet and ATM networking technologies are the principal candidates for the switching-network implementation.

9.3.3.2.2 The Pilot Project

Based on advances made by the RD31 collaboration [9-9], investigations of event building have been pursued in the Pilot Project as an extension of the LVL2 work. The same software and testbeds were used to study alternatively LVL2 and event-building aspects, as well as the integration of LVL2 and event builder data flow over a common physical network. The results and conclusions from this work are summarized in Chapter 6. The main points are noted here:

- Destination-initiated transactions have been selected and studied. These can be used for any type of data collection (gathering data from a subset of ROSs or from all ROSs), e.g. selective data collection, event building in multiple steps/stages, and full event building.
- The concepts demonstrated offer a basis for convergence of LVL2 and event builder towards the same data-flow protocol, common data collection software and possibly an identical hardware implementation.
- The principle of using a common physical network for LVL2 traffic, event building and protocol messages has been demonstrated on a small-scale ATM testbed using dedicated software. Computer simulation models also studied the possibility of combining LVL2 and event-building traffic on the same network.

9.3.3.3 EventBuilding Architecture

9.3.3.3.1 Overview

In this section we define the proposed architecture for the EventBuilding subsystem, based on the material presented in the previous sections and chapters. The functional elements are identified and defined first, then their interaction to realize the process of event building is defined. Boundaries and interfaces with the rest of the HLT/DAQ/DCS system were defined in Section 9.2.8.

9.3.3.3.2 Context and Use Cases

The process of event building deals with the following HLT/DAQ/DCS system elements:

- Event: all the data read out from the detector and corresponding to the same bunch crossing (same event identifier).
- Event fragment: a fraction of an event corresponding to the data from one or more ROSs.
- Trigger: an element external to the EB and signalling to it which events have to be built.
- ROS: the subsystem element providing the input event fragments.
- EF I/O: the subsystem element where the event fragments are collected into a complete event.

The context diagram shown in Figure 9-13 depicts how the EB interacts with other systems and subsystems. In addition to the transfer of data, a control protocol is shown between ROS, EB and EF I/O to indicate that the process of building an event implies some form of supervision. Interaction with the Online Software element is also indicated, for control, configuration, and operational monitoring purposes.

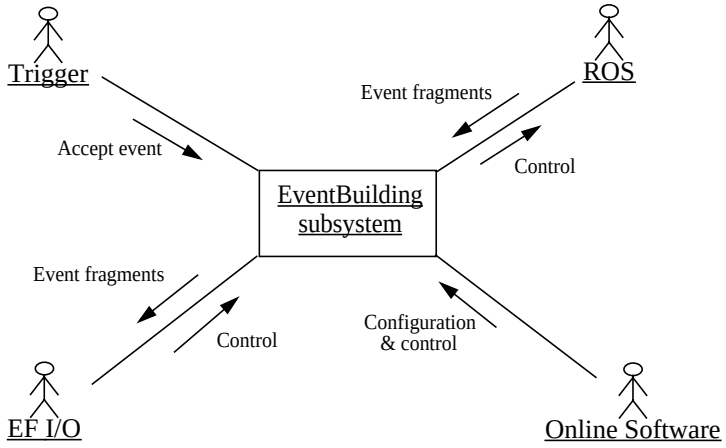

Figure 9-13 EventBuilding context diagram.

To illustrate the interaction of the EB with other system elements, we present a few major Use Cases:

1. *Build a complete event out of all event fragments*. The Trigger system signals to the EB that the event with identifier ID has to be built using all the ROSs. The EventBuilding assigns a destination DST in an EF I/O, where the event will be built. The EventBuilding notifies all the ROS that event ID has to be built at destination DST. Each source sends its fragment to the destination DST. DST assembles all the fragments into a full event.
2. *Build an event out of a subset of the event fragments (depending on trigger type)*. The trigger system signals to the EB that the event with identifier ID has to be built according to a trigger type which involves a subset of the detector. The EventBuilding assigns a destination DST where the event will be built, then continues as case (1) above with the exception that only a subset of the ROS participates in the event.
3. *Build events of a trigger type into a subset of the EF I/Os*. This case is the same as (2) above with the exception that the destination EF I/O belongs to the subset of EF I/Os which have been associated with that trigger type at configuration time.
4. *Partitioning*. An EB deals with a subset (partition, see Section 9.2.7) of the ROS and EF I/Os; for example, it includes all the ROS from a single detector and a number of

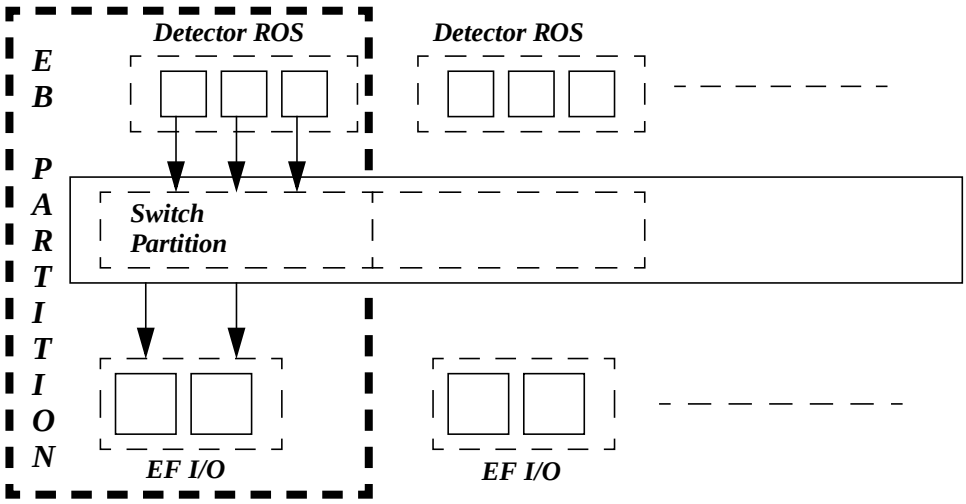

Figure 9-14 EventBuilding and Partitions.

EF I/Os. event building proceeds concurrently and independently within each different partition, with independent trigger sources. Figure 9-14 sketches an example of EB partitions.

9.3.3.3.3 Main Functional Elements

The EB consists of the following functional elements: source, destination, DFM, physical network, LocalController and network manager. Each of these is briefly described below.

- Source

  The Source is a component of the ROS (specifically the EBIF). It provides the following functionality: initialization of the network interface, sending of event fragments to a destination, and associated error handling. The association of an event ID (eID) of an event fragment in the source, and the destination ID (dID) of the destination, is obtained from the DFM (or possibly from the destination directly). The source uses the eID to access the event fragment and, together with the associated dID, initializes the network interface. The latter subsequently transfers the event fragment. When transmission has terminated (or a transmission error occurs) the network interface notifies the source. The source subsequently performs any further actions necessary, and, in case of error, may re-send the event fragment.

- Destination

  The Destination is a component of the EF I/O. It provides the following functionality: initialization of the network interface, event-fragment ordering, buffer management, and associated error handling. The destination understands the concept of an event and that events comprise a set of event fragments; the set may by identified by one or more attributes [e.g. eID, event type, and source ID (sID)]. The destination provides the network interface with the parameters necessary to perform the event-fragment transfer from the sources. When a transfer has terminated (or an error occurs) the network interface notifies the destination. The destination then notifies the DFM that the transfer from the source sID, has been completed (successfully or otherwise), and subsequently performs any fur-

ther actions necessary. The destination must provide the functionality to cope with the concurrent transfer of more than one set of event fragments, i.e. more than one event.

- Data Flow Manager

  The DFM is the component of the EB which ensures the correct flow of event fragments between sources and destinations. It defines the high-level protocol and provides the mechanisms to implement it. The DFM assigns dIDs to event fragments and traces their reception at their destinations. The dID assignment policy is based on the fact that event fragments are grouped into a set, and on the choice of partitioning (see below), sources and destinations.

  The assignment policy must also be able to select a specific dID from a possible set of dIDs. The association of a set of dIDs to a set of sIDs to form a partition of the EB is static and performed at initialization, but the final selection of a particular dID within the set for a given event is done at the time of event building.

  The DFM must record which sources have transferred event fragments to a destination. When all sources (statically selected at initialization) have transferred their event fragments, the DFM informs the appropriate destination that the event is complete. The DFM may also inform a destination that an event is incomplete when it deems that one or more sources, based on a predefined criterion (e.g. time-out), has not completed the transfer of an event fragment. The DFM is, however, neither responsible for the initialization, management or monitoring of the switching network nor for the handling of network congestion or network errors. These are the tasks of the network manager.

  An EB partition is defined as a given mapping between preselected sets of sources and destinations. Partitions are static (at the run level), non-intersecting and defined by certain common attributes. The DFM uses these attributes in association with the event ID to assign destinations. Possible common attributes to a set of event fragments include:

  a. A list of sources defined by a set of sIDs (not necessarily a continuous range).

  b. A range of ROSs defined by a set of ROS IDs.

  c. The event type.

  The selective use of these attributes by the DFM allows event building based on specific event types within a given partition; selective event building according to data source (e.g. all sources of data from a given subdetector); collective event building.

- LocalController

  The LocalController is the element providing configuration, control and operational monitoring of the behaviour of the EB. It also provides the interface to the Online Software system. See Section 9.3.1.3.3.

- Physical Network

  The Physical Network is the physical medium for the transfer of control and data messages. It interfaces to the other EB elements via a specific network API. We note that control messages (e.g. notification to a source of a destination assignment) and data transfers (fragments) may be transported by the same or by different networks.

- Network Manager

  The Network Manager is the component providing the initialization and monitoring of the physical network.

9.3.3.3.4 The EventBuilding Process

This section defines how the EventBuilding functional elements interact to provide the required functionality. Different patterns of interaction may be defined between the functional elements. This section considers only the source-initiated protocol transaction. Note this is in contrast to the protocol for the Selective Data Collection, described in Section 9.3.2. There, a destination-initiated protocol is well matched to the sequential strategy; avoids a high-rate system for distributing the RoI data requests (there is at least one request per LVL1 accept) across all ROSs; distributes the protocol load across all of the LVL2 processors, by only involving the processor requiring data and those ROSs which have to supply data in each request.

The model consists of objects (sources, destinations, DFM) and a high-level protocol. The transfer mechanism is provided by the inter-connection network and may be independent of the control mechanisms. The relationships between the objects are described in more detail in Section 5.2.3.The time-ordered sequence of the high-level protocol is depicted in Figure 9-15.

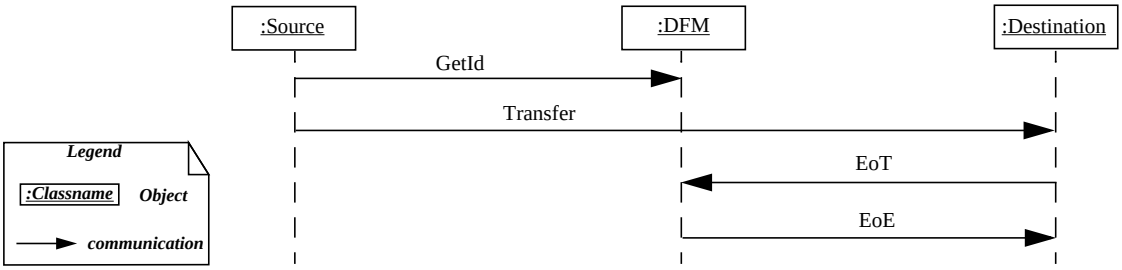

Figure 9-15 The time-ordered sequence of the high-level protocol for source-initiated transactions.

The set of mechanisms defining the high-level protocol is: *GetId*, *Busy/$\overline{Busy}$*, *End-of-Transfer* (EoT), *End-Of-Event* (EoE) and *Transfer*. The source uses a *GetId* mechanism with the DFM, to associate a dID with an event ID. The event fragment is then transferred. The destination notifies the DFM, via the EoT mechanism, when the event fragment has been received. The EoE mechanism is used by the DFM to inform the destination that the complete event has been transferred to the destination from all the sources. In the case where a destination becomes busy, a *Busy/$\overline{Busy}$* (B/$\overline{B}$) mechanism (not shown in Figure 9-15) is used between the destination and the DFM to halt the flow of new events to the destination. This mechanism is also used to resume the flow of event fragments.

9.3.3.3.5 Possible Deployment

The Source and Destination are designed and implemented as integral parts of the EventBuilding system. However, they are deployed as components of the ROS and the EF I/O subsystems respectively. There will be one or more Sources per ROS depending on the number of EBIFs in a ROS, see Section 9.3.1.3.4. Similarly for the EF I/O, see Section 9.3.4. The detailed design of the Source and Destination is tightly coupled, for reasons of performance, to the choice of the network interfacing. At this stage no choice has to be made and the detailed design of the Source and Destination must remain open to different network options, e.g. simple commodity interfaces or high-performance intelligent interface cards, protocol options.

As described in the previous subsections, the DFM is the component of the EB which ensures the correct flow of event fragments between Sources and Destinations. It may be deployed across one or more nodes: a functionality of the DFM may be deployed on the same node as a

Source, e.g. GetId; another functionality on the same node as the Destination, e.g. EoT; and the remaining functionality on one or more other nodes. The choice is a performance optimization issue. The DFM is also a component of the EB subsystem which is aware of partitions. The detailed design of the DFM must evolve with partitioning and allow different deployments to be studied with the aim of identifying the optimal deployment for performance and cost.

9.3.4 EF I/O Subsystem

The EF I/O subsystem provides the functionality to receive events fragments (i.e. it contains the EB destination), pass assembled events to the EventFilter and send EventFilter selected events to mass storage.

- Event fragments are received in the EF I/O and corresponding headers [9-5] are created at the subdetector level, and globally.
- Built events are sent to the EventFilter. No routing functionality is required here: communication is based on a one-to-one correspondence between the EF I/O instantiation and the EventFilter subset (single processor, cluster of processors or SMP machine).
- Selected events are sent to mass storage. The implementation of this functionality should be strongly coupled to the EventFilter, especially in the case of a remote location in a distributed environment. However, the presence of the EventFilter shall not be mandatory for the access to mass storage (e.g. for debugging purposes or unprocessed calibration/alignment data), hence the logical separation of this functionality from the EventFilter.
- The EF I/O must be configured, controlled and monitored by the Online Software system.

Figure 9-16 shows the context diagram of the EF I/O subsystem.

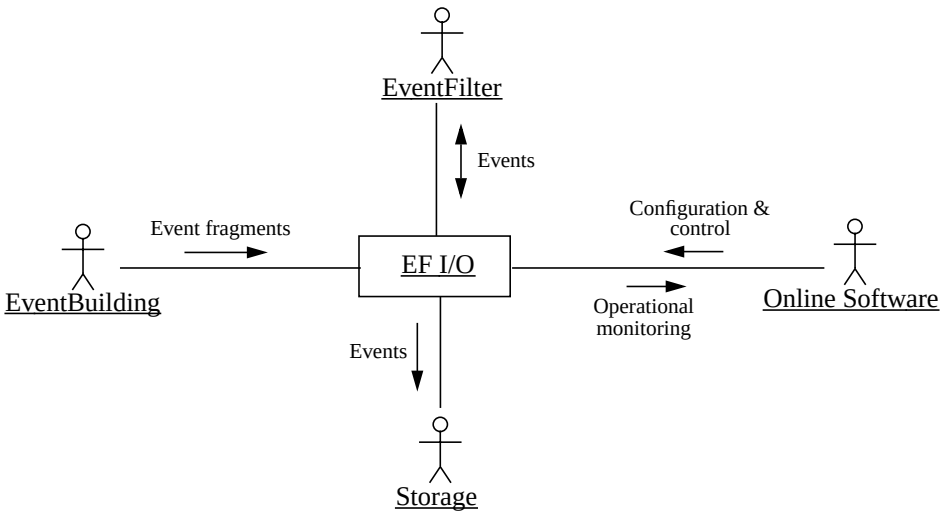

Figure 9-16 EF I/O context diagram.

The input functionality providing the interaction with the EB has been described in Section 9.3.3.3.3. Because many uncertainties remain on the nature of the Storage system, the

design of the output functionality is left open. The interface with the EventFilter is performed by APIs implemented by EventFilter components which are described in Section 9.6.

9.4 Online Software System

The Online Software System is responsible for overall experiment control, including run control, configuration of the HLT/DAQ system and management of data-taking partitions. The Online Software also includes the online-monitoring infrastructure and graphical user interfaces used for control and configuration, and the means for handling distributed information management including database management and tools.

The scope of the Online Software system also includes recommending solutions for the online computing infrastructure and implementation of the ATLAS control room (i.e. the Experiment Control Centre), including choice and configuration of servers and local area networks as well as workstations for experiment and detector subsystem control.

9.4.1 Main Conclusions from DAQ/EF -1 and Pilot Project

9.4.1.1 DAQ/EF -1 Conclusions

Here we give the principal conclusions of the Back-End software subproject. More detailed discussions can be found in Chapter 5 and in Ref. [9-10].

- A set of important software components covering control and configuration needs of the HLT/DAQ system have been identified and their requirements clearly stated.
- The identified software components have been implemented and can be used as a basis for future activities.
- The Back-End software has been successfully used to control, configure and monitor the DataFlow subsystem in various configurations during test-lab activities.
- The component model has proved very effective in organizing the development of software in a large, distributed collaboration.
- Mainstream software technologies such as C++, Java and Corba are suitable for control and configuration tasks in the HLT/DAQ system.
- The use of a well-defined software process has helped guide the development and improve the overall quality of the resulting software.
- Frequent, official releases of the Back-End software on a small set of supported platforms using CD-ROMs have proved to be an efficient means of distribution and allowed the users to become familiar with the Back-End software.
- The Software Release Tool (SRT) developed by the ATLAS Offline software group has successfully be used to manage a software project of 800,000 lines of code on multiple platforms over a period of more than 2 years.

9.4.1.2 Pilot Project Conclusions

- Components of the Back-End software of the DAQ/EF-1 project should be used to replace equivalent components of the Reference Software. Note, that this part of the Reference Software is already largely based on the high-level designs of the Back-End software.

9.4.2 Interfaces with other Subsystems

The Online Software has interfaces with the other systems and subsystems of the HLT/DAQ/DCS system and with the subdetectors, as shown in Figure 9-17. The concept of a LocalController is used as the interface point with other subsystems and is capable of bi-directional communication for exchanging information, messages and commands. The LocalController (Figure 9-17) is based on a framework providing access to Online Software facilities and is customized for each subsystem according to their specific needs. For the DataFlow, interfaces are required for the ROS, EventBuilding, LVL2 DataFlow and EF I/O. For the LVL2 Selection, the LocalController is included in the Framework of the ProcessingUnit. In the case of the EventFilter, the interface is with the EF Supervision. A LocalController will be developed in conjunction with the detector groups at the ROD crate level to provide an interface to subdetector ROD crates. The Online Software interacts with the SCADA system of DCS to ensure consistent control of the overall experiment. The details of this interface are described in Chapter 7.

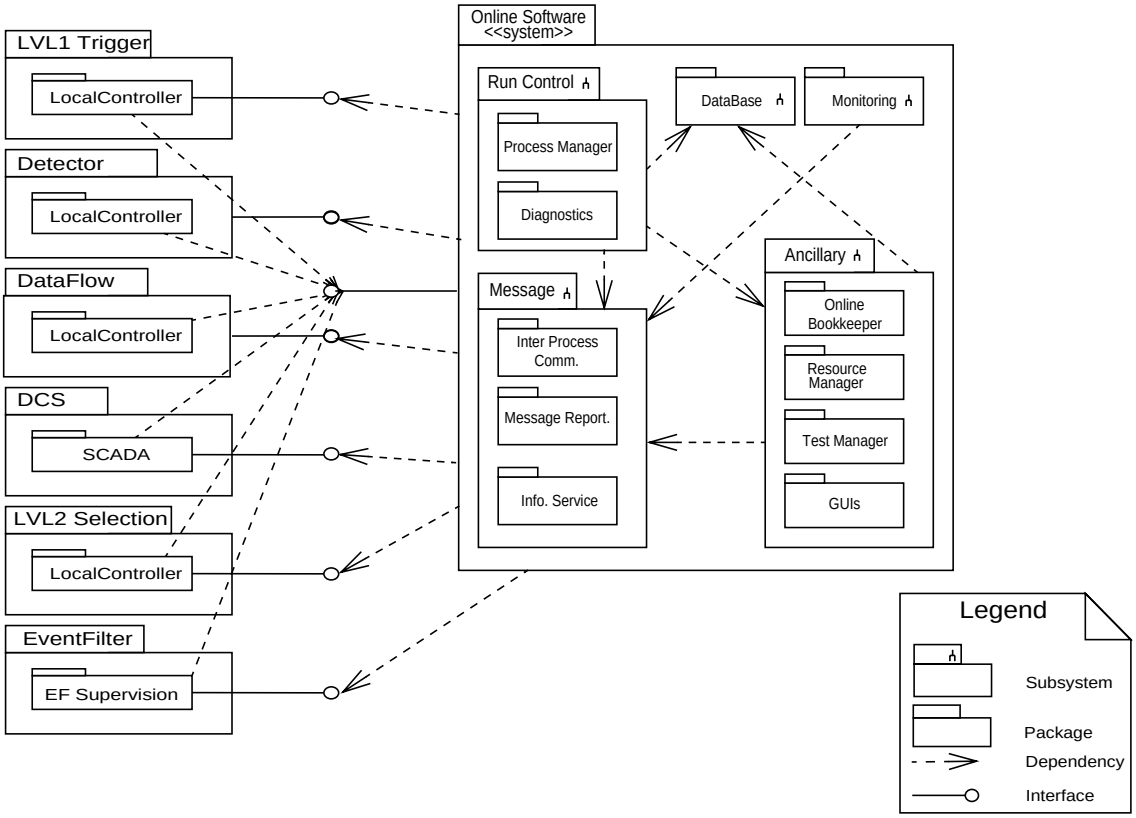

Figure 9-17 Online Software within the HLT/DAQ architecture.

9.4.3 Online Software Architecture

9.4.3.1 Overview

The Online Software will be based on the design of the Back-End subsystem of the DAQ/EF -1 project. The architecture will be taken from the Back-End component model as described in Chapter 5 and shown below in Figure 9-17.

To facilitate the integration with other subsystems, LocalController skeleton, based on the Back-End core components (see Chapter 5) will be included in the Online Software. Such a framework will enable the Online Software to communicate with subsystem-specific software for specialized control, configuration and monitoring tasks (e.g. SCADA for DCS). Appropriate use of the Back-End core components will replace the equivalent modules of the Pilot Project Reference Software (see Section 6.2.2).

9.4.3.2 Main Functional Elements

Figure 9-17 shown the main functional elements of the Online Software system. Their function is briefly explained below. More details can be found in Chapter 5.

9.4.3.2.1 Run Control

The run control component controls the data-taking activities by coordinating the operations of the HLT/DAQ/DCS systems and subsystems, other online-software components and external systems. It has user interfaces for the shift operators to control and supervise the data-taking session, and software interfaces with the HLT/DAQ/DCS subsystems and other online-software components. It contains functions for global process management and diagnostics.

9.4.3.2.2 DataBase

A DAQ system needs a large number of parameters to describe its system architecture, hardware and software components, running modes and the system running status. The system's operational parameters are stored in databases to allow maximum configurational flexibility.

9.4.3.2.3 Message

The aim of the Message Reporting System (MRS) is to provide a facility which allows all software components in the ATLAS HLT/DAQ/DCS system and related subsystems to report error messages. The MRS performs the transport, filtering and routing of messages. The Information Service (IS) provides an information exchange facility for software components of the system. Information (defined by the supplier) from many sources can be categorized and made available to requesting applications asynchronously or on demand. The inter-process communication function provides a facility for processes to inter-communicate in a flexible manner.

9.4.3.2.4 Monitoring

The monitoring subsystem is responsible for providing the means to control detector and physics monitoring applications and display their results (note that this does include DCS monitoring functions).

9.4.3.2.5 Ancillary

This subsystem provides the many ancillary functions required by any DAQ system. See Chapter 5 for more details.

9.5 LVL2 Selection System

9.5.1 Introduction

The LVL2 Selection is the system which for each LVL1 Accept, uses selected event data to decide whether the event should be deleted or retained for event building and further analysis. It receives from the LVL2 DataFlow a single message with information from the LVL1 Trigger system including details of the RoIs found. It can request detector data from the LVL2 DataFlow as needed for the LVL2 Selection task. Once the decision for an event has been made the LVL2 Selection returns it to the LVL2 DataFlow. Possible subsequent actions include:

1. Delete the event as the LVL2 criteria are not met.
2. Pass the event to EventBuilding because:
    a. LVL2 criteria are met, or
    b. Event should be retained for diagnosis / monitoring (based on some predefined criteria, e.g. a prescaling factor) even though the LVL2 criteria are not met, or
    c. LVL2 unable to complete the decision.

The system has to be capable of running at the maximum LVL1 Trigger rate of 75 kHz (upgradable to 100 kHz). The rate at which it requests events to be passed to EventBuilding should be of the order of 1 kHz (although depending on the boundary between LVL2 and EF this could change). The average latency for the LVL2 decision should be kept to a few milliseconds, but decision times of hundreds of milliseconds are acceptable for a small fraction of the events. The LVL2 Selection will necessarily be situated near to the DataFlow system, owing to its frequent low-latency data-access requirements.

9.5.2 Main Conclusions from the Pilot Project

The Reference Software has been run in many configurations and has been shown to scale to systems of up to ~100 nodes. The tests indicate that the required component performance can be obtained with commodity hardware (PCs) and OS software (such as Linux). The request / response-based architecture has been validated. Processors in testbeds using the Reference Software have demonstrated a three-step sequential selection strategy with prototype algorithms

running on a multi-node testbed with the processor requesting simulated event data from ROB emulator nodes.

The ATLANTIS system was integrated into the Reference Software and a testbed, appearing to the rest of the system as a standard node. This work indicates how FPGA co-processors could be included with standard processors in a transparent way, offering significant performance improvements for suitable compute-intensive algorithms and hence a reduction in the size of the processor farms required.

Models of a full-scale system indicate that: the high-p_T LVL1 Triggers may be handled by similar numbers (100–200) of 1000 MIPS processors at both high and low luminosity; the maximum data volume for LVL2 through the network is 20–40 Gbit/s for low luminosity (including the Inner-Detector full-scan for B-physics) and 10–25 Gbit/s for high luminosity; the Inner-Detector full scan increases the processing power and network bandwidth required in the LVL2 processors — current estimates give a total of ~ 700–800 processors without coprocessors or ~ 400–500 processors with FPGA co-processors. The additional computing power required for the B-physics processing at low luminosity provides a safety margin for the high-p_T triggers that cover the rest of the physics programme.

9.5.3 LVL2 Selection Architecture

9.5.3.1 Major Use Case

The Supervisor within the LVL2 DataFlow allocates the LVL2 Selection task for each event to a ProcessingUnit within the LVL2 farm, and sends the event identifier and the LVL1 information (LVL1 Trigger type, location of RoIs, etc.) to that ProcessingUnit. The subsequent processing of the event within LVL2 is divided into three functional blocks: Steering, Feature Extraction and Preprocessing. The first two are performed within the ProcessingUnit. However, for efficiency reasons the Preprocessing, which is essentially data preparation for the Feature Extraction, is performed within each ROS which supplies data.

When the LVL2 processing requires some detector data it sends a request to the SDC element of the LVL2 DataFlow. The request would typically be for the data within a given RoI and given detector. The SDC gathers together all of the data fragments for the request and returns the data in a single block.

Typically, the first step consists of the confirmation of the LVL1 RoIs using the calorimeter or muon detectors. Thus, for example, for a LVL1 electron trigger, the LVL2 Selection would request calorimeter data for the primary RoI and check whether the finer detail available to LVL2 confirmed the LVL1 Trigger. If this condition is satisfied data would be requested for the same RoI from the Inner Detector to check if a matching high-p_T track confirmed the trigger.

Once the confirmation of the LVL1 Trigger has been completed, further analysis may be performed, for example the secondary RoIs or, in the case of B-physics, an unguided track search in the Inner Detector, followed by further analysis, in the calorimeter and muon systems, of the track candidates found. A full-scan track search in the Inner Detector requires all of the data for the event from a whole subdetector. Such requests, however, involve considerable use of resources, both for the data transfer and the execution of the algorithm, and would be at a much lower rate than those for a single RoI.

If at any step none of the hypotheses associated with the LVL1 Trigger are satisfied the event is rejected. As soon as an event has been rejected or has met all of the LVL2 selection criteria, the ProcessingUnit makes the LVL2 decision. Normally if the event has been rejected it will be marked for deletion; however, sometimes for diagnostic or monitoring reasons it will be marked for retention and passing to the EventBuilding. If the event has met the LVL2 selection criteria it is always passed to the EventBuilding.

Finally the decision is passed back to the Supervisor in the LVL2 Dataflow. If the decision is that the event should be passed to the EventBuilding, the LVL2 processor also has to make the details of the LVL2 processing available to the EventBuilding so that these can be passed to the EventFilter. (The mechanism to do this is not yet decided as it depends on various implementation options of the farms and networks; for example, the data could be passed to a suitable buffer which can be read by EventBuilding, or the EventBuilding might request it directly from the LVL2 processor.)

9.5.3.2 Data Elements

The following data elements pass between the ProcessingUnit of the LVL2 Selection and the LVL2 DataFlow (see Figures 9-2 and 9-18):

- The *EventDescriptor*, which contains the *LVL1Result* formatted into a single record, plus any additional information required from the Supervisor. The *LVL1Result* which originates from the LVL1 Trigger, is received by the RoIB, and is passed from the DataFlow by the Supervisor. It provides the event identifier and a list of RoI information (LVL1 Trigger type, transverse energy and RoI location) for each event.
- *DataRequests* emitted by the ProcessingUnit to request event data required by the algorithms. These requests are distributed to ROSs as necessary by the SDC.
- *DataResponse* the data collected from ROSs, reformatted, possibly preprocessed and returned as a single block to the Feature Extraction by the SDC.
- The *LVL2Decision* contains the conclusion of the suite of algorithms for each event which is returned to the Supervisor. If the Event is to be retained for event building a summary of the intermediate processing results is stored as supplementary information for the EventFilter.

9.5.3.3 Functional Elements

Figure 9-18 shows the global view of the LVL2 Selection system, which contains the ProcessingUnit, with Framework, Steering and Feature Extraction functions and Preprocessing.

- Framework

  In each ProcessingUnit there is a processing Framework to hold the different components together. The principal functions of the Framework are:

  - Initialization of the ProcessingUnit, setting up the required connections to the DataFlow interface and configuration of all the components.
  - Communication with the Online Software for initialization and configuration parameters, run control, error reporting and monitoring.
  - Creation and scheduling of a certain number of worker threads which each implement a Steering and Feature Extraction chain for one event. Each ProcessingUnit

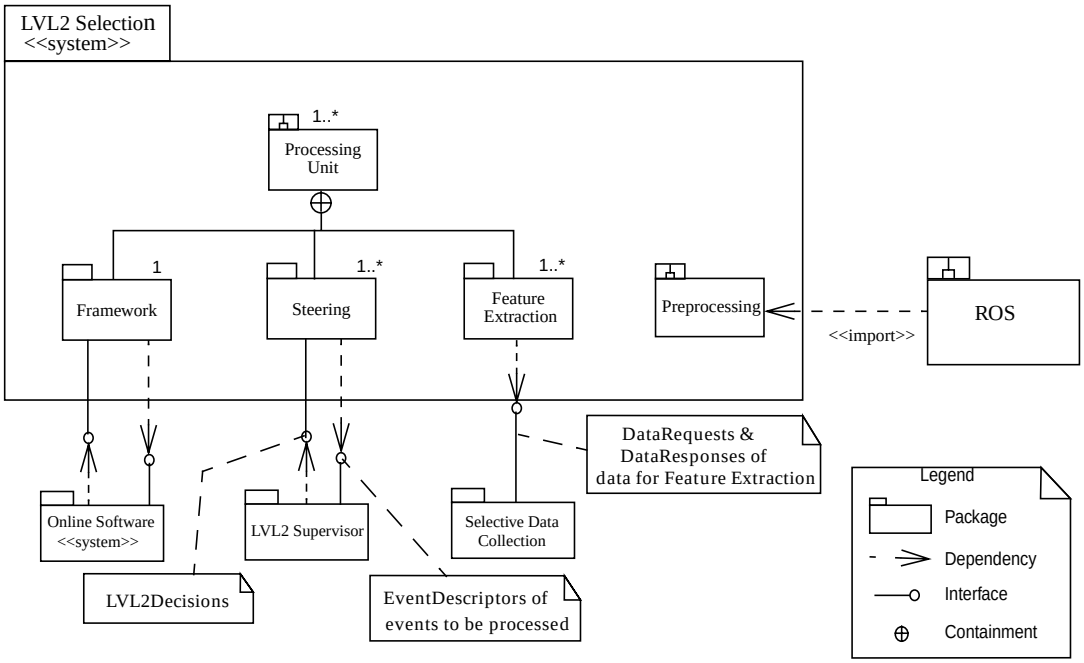

Figure 9-18 LVL2 Selection package diagram.

treats several events in parallel, the number being configurable and determined by the I/O latency (a consequence of the request/response strategy) and the number of CPUs.

- Ensuring system integrity, e.g. handling excessively long execution times of algorithms.

- Steering

  The Steering implements the RoI guided LVL2 Trigger decision process using the LVL1Result as input and producing the LVL2Decision. It is driven by a menu which contains several sequential steps, each with a set of selection criteria. For each step the necessary Feature Extraction algorithms (see the separate package described below) are called. Each Feature Extraction algorithm produces Features (typically clusters or tracks), within the region specified, e.g. by a LVL1 RoI, which are returned to the Steering. The Steering uses the Features and combinations of them to check against the selection criteria as specified in the menu. If at any given step the selection criteria are not satisfied the event is rejected. When the Steering has completed the decision, it produces the LVL2Decision which is returned to the Supervisor. In addition for events to be retained for event building a summary of the details of the decision process is made available for inclusion in the event building.

- Feature Extraction

  The bulk of the processing is done by Feature Extraction algorithms. To ensure flexibility in the overall HLT approach these algorithms must comply with certain rules established in collaboration with the Physics and Event Selection Algorithm (PESA) group:

  - Standardized Objects contain input and output data to facilitate interchange between algorithms, both within and between the LVL2 and the EventFilter systems (note also the compatibility with offline code). The Framework has to take care of the necessary conversion and construction of the objects.

- Algorithms must be self-contained (i.e. no references to libraries such as HBOOK or CERNLIB, or equivalent class libraries) and thread-safe.

An additional advantage of this approach is that algorithms can be developed, tested and evaluated in an offline environment with the usual third-party support libraries.

Though compatible with the EventFilter and offline algorithms, it is expected that LVL2 selection algorithms are developed independently and especially optimized for execution speed. The current set of algorithms studied in an online environment comprises EM, JET and τ calorimeter clustering; RoI-guided (high luminosity) and full-scan (low luminosity) TRT tracking; and Hough transform or Kalman-filter based Pixel and SCT tracking.

- Preprocessing

  To reduce the bandwidth requirements of the network connection between the ROS and the LVL2 Selection and to facilitate the LVL2 processing, certain types of preprocessing are foreseen within the ROS. These include data selection, e.g. removing any data from the ROS outside the RoI; data combination, such as returning energy sums from the calorimeter; reformatting, such as compacting the TRT data; simple processing, such as finding hit clusters in the SCT; zero suppression, e.g. for Liquid Argon Calorimeter. Although the preprocessing is not performed in the LVL2 ProcessingUnits, these tasks are part of the LVL2 Selection procedure and must be prepared as part of the LVL2 algorithms.

9.5.3.4 Possible Deployment

A simple implementation would be to run a single ProcessingUnit on each processor within a farm of LVL2 processors. The Framework could be implemented as a collection of threads, with *Mutexes* and *Condition variables* used for synchronization and access to shared Objects. It should allow asynchronous I/O (which permits overlap of data I/O with processing) and support multi-CPU processors (e.g. SMP systems). Each Steering could correspond to a thread supported by the Framework, and Feature Extraction could be performed either in the same or a separate thread. Performance enhancement of compute-intensive Feature Extraction algorithms could be achieved using FPGA *co-processors,* e.g. hardware-assisted TRT tracking. Multiple CPU machines and SMPs might also be used.

9.6 EventFilter System

9.6.1 Introduction

The EventFilter system forms part of the High Level Triggers (HLTs), and has three principal tasks: to further reduce the event rate accepted by the LVL2 Selection; to monitor the global detector performance and the physics quality of the data; to run global detector calibration and iterative alignment procedures.

One of the major issues of the EventFilter architecture is to provide the required rejection factor using a total CPU power inside the funding limits. This amount of CPU power is not precisely known and will depend strongly on the requirements on selection quality, as well as on the sharing of the selection task between LVL2 Selection and EventFilter. In addition, data-taking operations will require that additional tasks (express analysis, monitoring, calibration, etc.) be

performed. These requirements will evolve during ATLAS's lifetime, therefore putting a large premium from the outset on architectural flexibility and scalability in order to accommodate this evolution. Processor technology will also evolve at a rate which can be estimated from the road maps of the computing industry, although uncertainties remain.

In Section 9.6.2 the main conclusions from the EF studies in the DAQ/EF -1 project are presented. These conclusions form the basis for the EventFilter architecture described in Section 9.6.3.

9.6.2 Main Conclusions from DAQ/EF -1 Project

The following sections present the major conclusions of the EF prototype work relevant to the architecture. A detailed description of the work, measurements and conclusions can be found in Ref. [9-11].

- Platform Independence

  Modularity and portability have been achieved by clearly separating the generic part of the design from the communication aspects, the latter being dependent on the underlying hardware. The clear definition and specification of the boundary and interface between the DataFlow and the Distributor played a major role in minimizing the coupling between the DataFlow and EF, and thus facilitating the integration of these systems. The design and implementation based on object-oriented techniques has also greatly facilitated the work.

- Data Redundancy and Robustness

  The use of a back-up buffer has proved to be possible and judicious to avoid event loss due to errors during event processing.

- Data Communication Mechanisms

  The design has proved to be flexible enough to allow different models of data movement within the EventHandler. Data movement has been studied by copying events between nodes in a distributed system (PC prototype) and shared memory using SMP machines.

  The client–server protocol used between the processing task and the Distributor also implements efficient load balancing across the EventHandlers in a subfarm. Moreover, it does not require any intervention of a supervisor (data-flow manager), thereby improving the robustness of the system.

- Throughput and Scalability

  The throughput achieved by the present prototypes has been shown to scale. Projecting their measured throughput to some 100 EventHandlers, for a LVL2 output rate of 1 kHz, events of 1–2 Mbyte in size, and processing times of the order of 1 s (estimations made in the ATLAS TP), one is close to satisfying the ATLAS requirements.

  EventHandlers can easily be added, or increased in size to improve the overall performance.

  The SMP prototype has shown that, through the use of kernel threads, scaling up to several tens of processors and hundreds of processing tasks is possible.

- EventHandler Configuration and Monitoring

  The configuration and operational monitoring of large processing farms is a major challenge. The use of tools such as the Java mobile agents has proved to be promising in terms

of its adaptability, portability and scalability. The same techniques are being actively investigated for similar applications in CERN IT division.

- General Conclusions

  The concept of independent EventHandlers, or subfarms, connected to different output ports of the Event Builder has been studied in the context of the DAQ/EF -1 project. This approach is interesting since it is naturally scalable as long as the tasks performed in the EventHandlers are CPU bound. It reduces the number of nodes on the EventBuilding.

  The possibility of a mixed-technology (both at the hardware and operating-system levels) EF must be envisaged through the lifetime of the experiment, and then implies that the EF Supervision must be independent of the technology choice, as must be the largest possible percentage of the internal data-flow software. An architecture based on subfarms considerably eases this supervision and allows different technologies for the EF, e.g. PC farm and/or SMP machines, to co-exist in different EventHandlers. It is also fully compatible with the use of the GRID concept [9-3] inside the EF.

  Another conceivable approach emphasises the commonalities between the LVL2 farm and the EF, by considering the possibility of a single farm architecture for the two selection stages. In the rest of this section, we consider only the solution where the LVL2 and EF farms are separately implemented. However, the single-farm option is not rejected at this stage and is mentioned in Section 9.5. It should be further studied (see Chapter 10).

9.6.3 EventFilter Architecture

9.6.3.1 Major Use Cases

Figure 9-19 shows the context diagram of the EventFilter. It interacts with the other systems in the following way:

- Receives fully assembled events from the EF I/O and communicates selected events to the mass storage via the EF I/O. The data communicated back to the EF I/O will depend on the type of event analysed and may contain only the results of the EventFilter selection.
- Provides the subdetectors with the means of performing event-data monitoring and calibration.
- Receives configuration and control commands from the Online Software system and provides status information for the purpose of operational monitoring.
- Accesses calibration, alignment and geometry information from the databases.
- Provides updated information to the databases after having processed calibration specific event data.
- Uses algorithms developed and implemented by the Offline software group.

9.6.3.2 Main Functional Elements

Figure 9-20 shows the global view of the EventFilter system, containing the EventHandler and EF Supervision subsystems. The latter implements the interface to the Online Software system and also uses the interface implemented by the Online Software system.

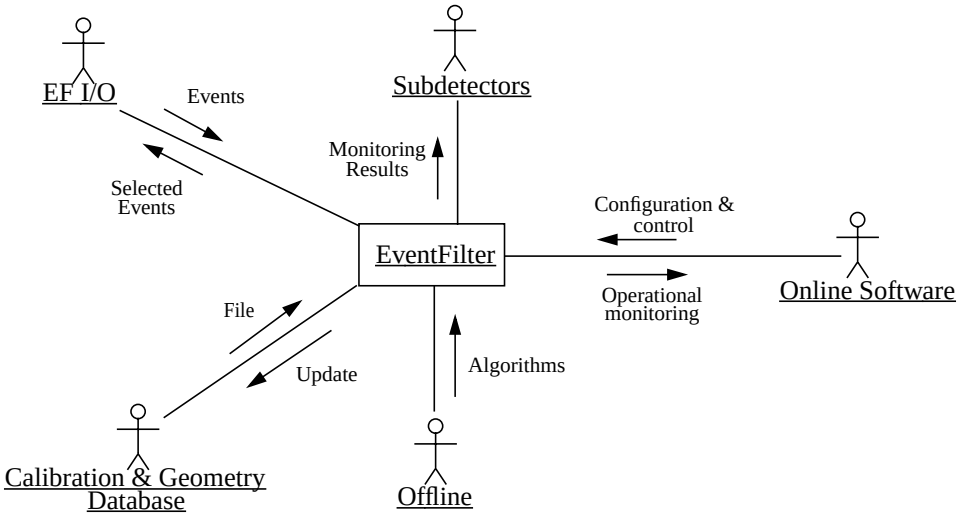

Figure 9-19 EventFilter context diagram.

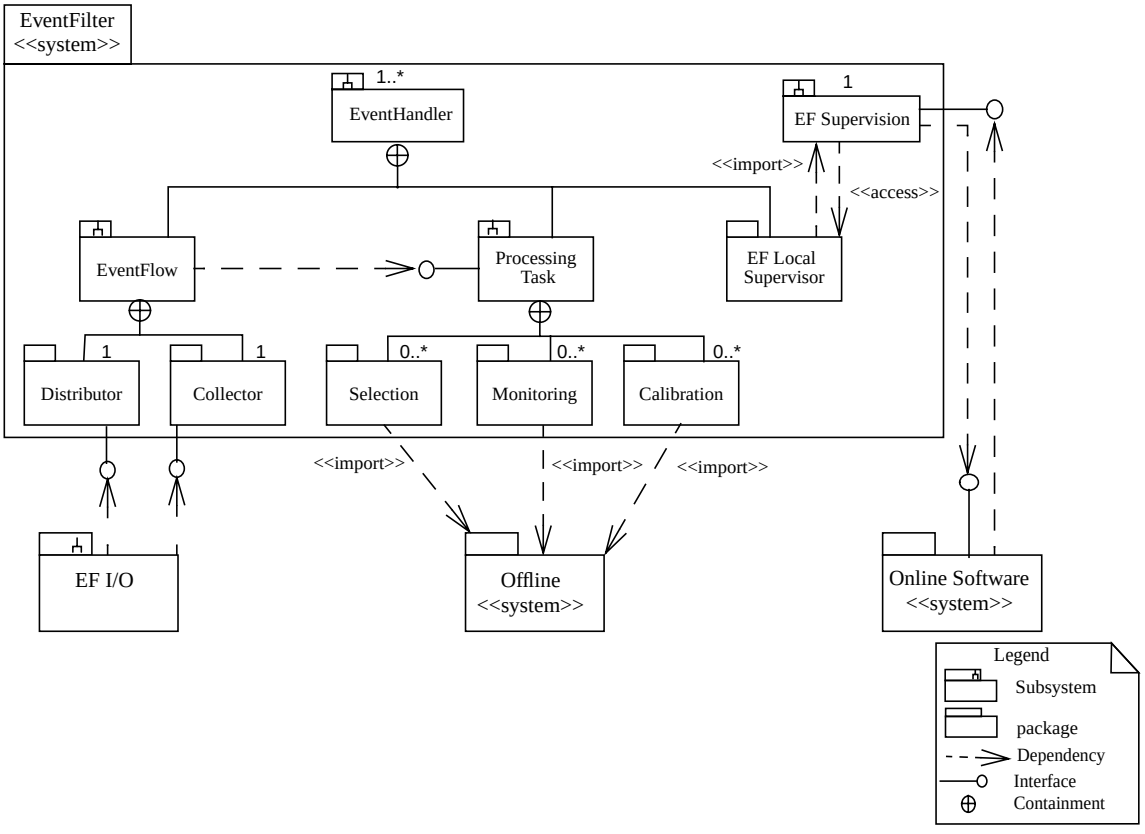

Figure 9-20 EventFilter package diagram.

The EventFilter is factorized in several independent EventHandlers, which provide the computing power necessary to accomplish the various tasks of the system. No restriction is made on the possible implementations (PC subfarm, SMP machine) or hardware architecture (single or separate farms for LVL2 and EventFilter). This factorization provides an easy way to implement flexibility and scalability. It is the responsibility of the EB subsystem to ensure the load balanc-

ing among the different EventHandlers of the EventFilter, by keeping account of the relative occupancy of its destinations and assigning new events accordingly.

The EventHandler is responsible for moving events through the different steps of the processing operation. The EF Supervision[1] controls and monitors the operation of the whole EventFilter. The EventHandler contains the EventFlow and the Processing Task subsystems. They also contain an EF Local Supervisor component which imports into the EventHandler the functionalities of the EF Supervision.

The EventFlow contains one Distributor and one Collector. The Distributor receives events from the EF I/O (see Figure 9-20) and communicates them to the different processing tasks of the EventHandler via an interface implemented by the Processing Task which isolates the different possible technologies of the EventHandlers. The Distributor distributes events according to event type. It is also keeps a safe copy of the event throughout the lifetime of an event in the EventHandler. The Processing Tasks implement the selection criteria as well as other ancillary tasks such as event-data monitoring or calibration analysis. They work independently on the basis of one event per processing task. The Collector sends the selected events back to the EF I/O (again via an API).

9.7 DCS

The DCS is described in detail in Chapter 7. In this section, its main architectural features are presented and put into the context of the whole HLT/DAQ/DCS architecture.

9.7.1 Introduction

The DCS system has two contrasting aspects: It is an integral part of the proposed HLT/DAQ/DCS architecture but, is required to be a technically and operationally separate system.

The reasons for the technical and operational separation of DCS are outlined below:

- The DCS is not involved at all in physics event-data handling. These are collected, transported, analysed and stored by the HLT and DAQ systems.
- The DCS must be capable of standalone operation during the commissioning of the subdetectors and during periods of no data-taking, i.e. when other HLT/DAQ systems and subsystems are not necessarily available.
- One of the two subsystems of DCS, the SCADA (see Figure 9-1), will be based on an integrated commercial software package. This package will be selected and deployed in collaboration with the other LHC experiments, in the context of the Joint Controls and Operations Project (JCOP), [9-12].

The integration of the DCS with the HLT/DAQ systems must provide:

1. The supervision issues referred to here concern overall farm initialization, control, monitoring and error handling, and are fundamentally different from those discussed in Section 9.3.2, which concern event-by-event control.

- A coherent environment in which the ATLAS experiment can be safely and efficiently operated. This calls for an extensive collaboration between the DCS and the Online Software system, in particular the Run Control subsystem.
- A well-defined interface either to the DCS or to the DAQ for each of the systems external to the HLT/DAQ/DCS. This requires the capability of seamless and efficient exchange of data between the DAQ and DCS. The DCS connects to external systems in the following ways:
    - All data and logging information from the DCS, required by the Offline system, is transmitted via the Online Software system to mass storage.
    - All interaction needed by ATLAS with the LHC accelerator is performed by the DCS.

The main consequence is that the DCS has essentially only one logical connection to the HLT/DAQ, namely to the Online Software, as shown in Figure 9-1, for control, configuration and operational monitoring. Apart from this, the DCS supervises the hardware of HLT and DAQ systems (e.g. electronics crates and racks), as it does for the subdetector hardware.

9.7.2 Main conclusions from prototype work

The experience from the LEP experiments and the analysis of the ATLAS experiment have shown that the detector is organized in a strict hierarchy and that very little interaction is needed horizontally. The DCS architecture has to be capable of mapping to this hierarchy and allowing efficient vertical interaction. The distributed and subdetector organization of the ATLAS experiment suggests to structure DCS into a federated supervisor system and a dedicated Front-end I/O system.

Evaluations of SCADA systems done within JCOP have proven that such packages, which are developed for the control of industrial installations, are usable for the supervision of the ATLAS detector, provided they are *open*, i.e.:

- The user has access to the information held internally in the SCADA system.
- There is the possibility to connect to specialised hardware.
- Interfaces exist to external software packages.

Due to the topology and size of the ATLAS detector, the Front-end I/O system has to be highly distributed. A fieldbus of the order of a hundred metres in length with ‘intelligent’ nodes has been found to provide a suitable solution. Prototypes of I/O modules which are based on the CAN fieldbus have successfully been built.

9.7.3 DCS Architecture

9.7.3.1 Major Use Cases

The “users” of DCS are shown in the context diagram in Figure 9-21. They are the Online Software, the Subdetectors, the LHC accelerator, the CERN infrastructure and the operator.

- The Online Software sends commands to configure and control the subdetectors. The DCS reports back the success or the failure of the execution of the commands. It also informs the Online Software about asynchronous changes of the operational status of the detector. Data to and from an ATLAS-wide database (e.g. configuration data and measurement results) are also transmitted via this path.
- The Subdetectors receive detailed commands from the DCS, e.g. to bring them in the desired state or to execute special procedures like calibration. The DCS receives both the results of these interactions and all the operational parameters of the subdetectors.
- The LHC accelerator and ATLAS exchange information in both directions via the DCS. This allows the operational status to be known and also the synchronization of procedures related to beam injection or dumping and ramping high voltages.
- The CERN infrastructure services send their operational status to the DCS. This enables the DCS to initiate procedures such as an orderly shutdown of part of the detector, if necessary.

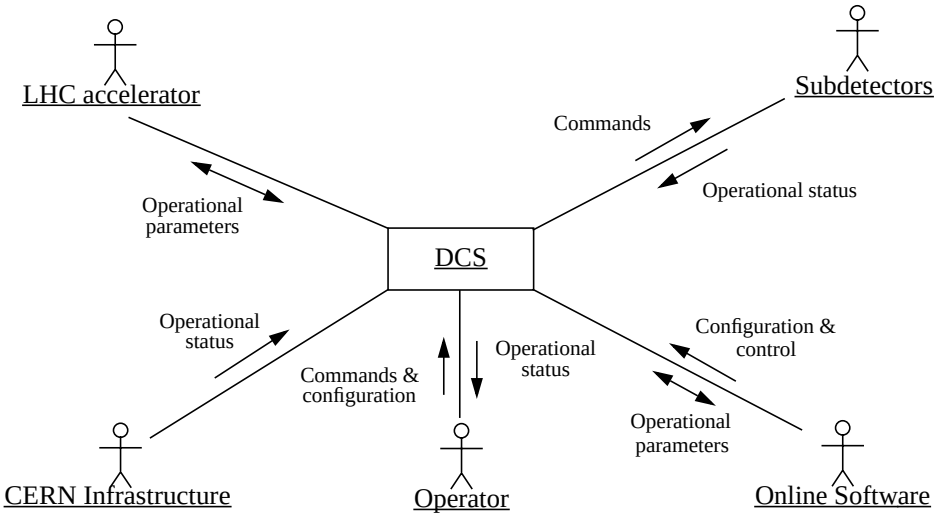

Figure 9-21 DCS context diagram.

9.7.3.2 Main Functional Elements

The DCS is composed of two subsystems: SCADA and Front-end I/O. Their internal structure is shown in Figure 9-22. The central functions of the SCADA, such as alarm handling and logging of data, are situated in the Management package, which also holds the complete knowledge of the state of the detector. These data are received from the Front-end I/O system via the Front-end Interface package, which comprises hardware drivers and software communication tools. Dedicated user-supplied control applications which analyse and combine these data, and which may initiate actions, reside in the Processing Layer. All connections to both the operator and the external systems are made via the User Interface Layer.

The Front-end I/O subsystem gathers the data from the subdetector hardware and performs actions. It consists of two subsystems: Device I/O and Complex Front-end Systems (CFSs). The Device I/O encompasses simple sensors and actuators and implements an interface, i.e. CanOpen which is used by the Front-end Interface package of SCADA, for the purposes of operational monitoring and control. The Device I/O subsystem may be either general-purpose

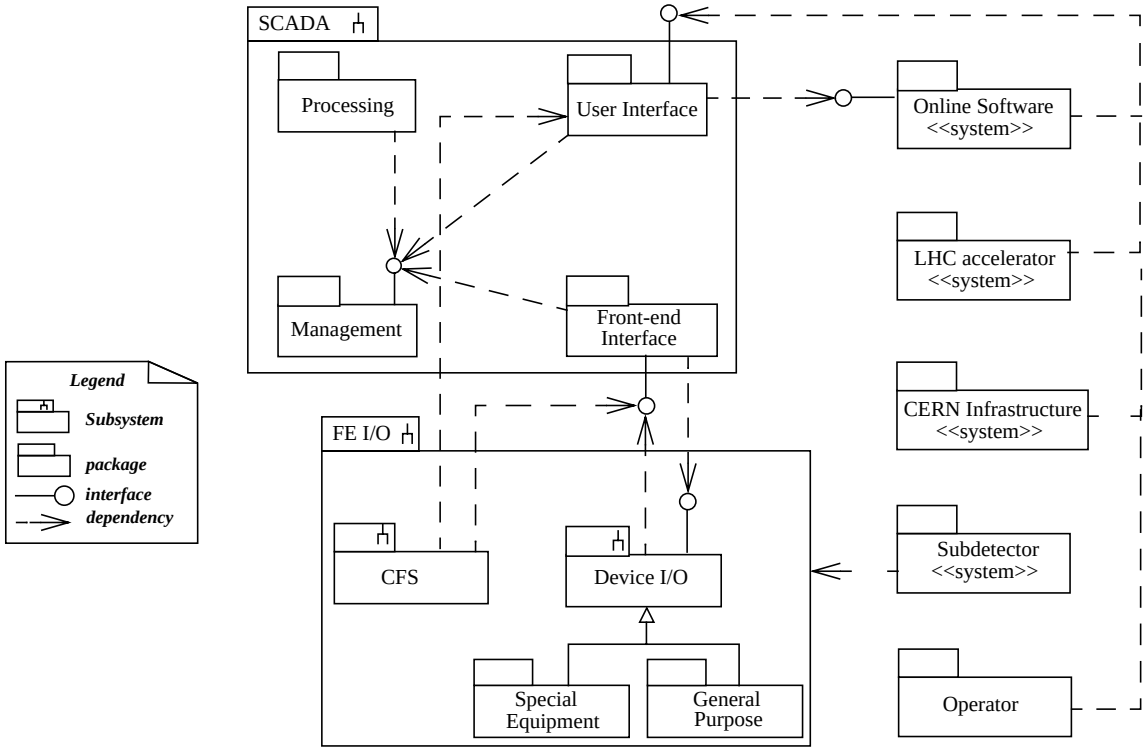

Figure 9-22 Package diagram showing the DCS subsystems and their main functional elements.

modules or special (dedicated) nodes for specific subsystems. CFSs range from small stand-alone control units e.g. based on Programmable Logic Controllers (PLC) to high-performance computer systems based on specialized processors, dedicated to the analysis of large volumes of raw data and extract summary information for use by the SCADA subsystem.

9.7.3.3 Possible Deployment

Modern SCADA packages usually run on PC hardware. In a first stage, all functions can be performed in a single PC, which allows standalone control of (part of) a subdetector. This is called a Local Control Station (LCS). In a later stage, a layer above the LCS will be constructed with the SCADA tools, which implements the supervisor level of the whole detector. This also includes the connection to external systems.

The Front-end I/O system will make intense use of the commercial, standardized CAN Fieldbus. General-purpose nodes will continue to be developed taking into account the necessary functions and the packaging needs of the subdetectors. Subdetector groups may develop dedicated CAN nodes, which will be embedded in their front-end electronics. The market will be continuously followed up in order to identify products which meet the DCS requirements.

9.8 Summary

This section summarizes the main elements of the architecture proposal, as described in the present chapter. The proposal should form the basis for the next phase of the HLT/DAQ/DCS project up to the TDR. It introduces and explains design ideas in many areas, with the level of

detail varying from system to system depending on our current knowledge and understanding. We have deliberately avoided addressing detailed implementation issues in this chapter. A current assessment of work to be done in the next phase up to the TDR, based on the architecture presented in this chapter, is given in Chapter 10.

The major requirements of the architecture have been identified, based upon the work of the DAQ/EF -1 Project, the LVL2 Pilot Project, the DCS Project, the Detector Interface Group and the PESA group.

The overall organization of the architecture has been presented, identifying its key systems, namely: DataFlow, LVL2 Selection, EventFilter, Online Software and DCS, and their component subsystems, boundaries and interfaces. The HLT system comprises the LVL2 Selection and EventFilter systems, while the DAQ system comprises the DataFlow and Online Software systems. Data is served to the LVL2 Selection and the EventFilter by the DataFlow system. Below we recall the principal functions of these systems and their subsystems.

- DataFlow

  Provides the functionality of receiving and buffering data from the subdetectors ReadOut Drivers, merging and distribution of events and event fragments to the High Level Triggers, and the sending of events to final storage.

  - ReadOut

    The ROS provides the receiving and buffering of subdetector data from the ReadOut Drivers over the ReadOut Links. It may execute preprocessing on data before transmission to the LVL2 Selection in response to a request. It provides partial event sampling for subsequent monitoring of detector data.

  - LVL2 Dataflow

    The LVL2 DataFlow receives the primary RoI information from the LVL1 Trigger and formats it; allocates an event to a processor; provides the distribution of the selected event data to the LVL2 Selection system following requests from the latter; and transmits the LVL2 decision to the ReadOut and EventBuilding.

  - EventBuilding

    The EventBuilding merges selected event fragments into complete events.

  - EF I/O

    The EF I/O contains the destination in which events are built by the EB. It provides the distribution of events to the EventFilter system, the reception of accepted events from it and the subsequent transmission of events to mass storage.

- LVL2 Selection

  The LVL2 Selection receives information from the LVL1 Trigger for each accepted event and applies optimized event selection algorithms to reduce the rate of events passed for full event building. The algorithms use detector data generally limited to RoIs; data are requested from the LVL2 Dataflow as and when required.

  - ProcessingUnit

    The ProcessingUnit is the computing resource responsible for the execution of the LVL2 selection algorithms. It is subdivided into a Framework for general support, plus Steering and Feature Extraction to execute the algorithms.

- Preprocessing

  Preprocessing provides data preparation for the LVL2 algorithms. It is performed in the ROS to obtain improved efficiency in the use of resources, especially network bandwidth.

- EventFilter

  The EventFilter receives merged fragments of events accepted by the LVL2 Selection and filters them by applying complex physics algorithms imported from the Offline. It provides a platform for running user-defined global monitoring (detector and physics) and calibration algorithms where data from one or more detectors are required.

  - EventHandler

    The EventHandler is the computing resource responsible for the execution of selection, monitoring and calibration algorithms.

  - EF Supervision

    The EF Supervision is responsible for the initialization, control and monitoring of the EventHandlers.

- Online Software

  The Online Software is responsible for the configuration and control of the detector and HLT/DAQ systems. It provides run control; configuration, including the management of detector and DAQ partitions; distributed information management; monitoring software infrastructure; graphical user interfaces for the purpose of control and configuration.

  - Run Control

    Responsible for the overall initialisation and control of all other ATLAS systems and subsystems. Interfaces to shift operators.

  - Message

    Responsible for collating, filtering, displaying, notifying and logging messages (typically error messages) emanating from all systems and subsystems. It is also responsible for receiving and making globally available data and status information concerned with the operation of the DAQ system.

  - Monitoring

    Responsible for providing the means to control detector and physics monitoring applications and display their results (note that this does include DCS monitoring functions).

  - DataBase

    Control, implementation and access to/from databases associated with the operation of the data acquisition system.

  - Ancillary

    Ancillary functions required for the operation of the Online Software system and required by shift operators

- DCS

  The DCS is responsible for the coherent and safe operation of the ATLAS detectors and associated systems, and for the interfacing with the LHC accelerator and the services of

the CERN infrastructure. It deals principally with non-physics event data which are time-stamped.

- SCADA

  Reads / writes data from / to FE I / O; processes, displays, and archives this data; implements control procedures; interacts with operator; handles commands, messages and alarms; connects to systems external to ATLAS.

- FE I / O

  Reads out and digitizes signals from detector hardware; preprocesses these data; implements low-evel control procedures; drives actuators.

The proposed architecture is highly distributed, with computing resources located both close to and far from the experiment. The location of these resources reflects their function. Table 9-2 shows the various computing resources in the HLT / DAQ / DCS system as a function of their geographical location, noting in each case the function of the resource and indications of its possible implementation.

Table 9-2 Geographical location of computing resources, and their function.

Possible Geographical Location	ROD Crate VME Processors & Custom boards	ROS COTS & Custom boards	LVL2 Selection PCs, SMPs & Co-processors	EventFilter PCs, SMPs
ATLAS pit	Partial Monitoring Hard Real Time User processes Calibration			
Surface Buildings		Partial Monitoring Hard Real Time	RoI processing Trigger algorithms Hard Real Time	Event selection Filter algorithms Calibration Full Monitoring Soft Real Time
CERN site				As above
Off CERN				As above No Real Time

Computing Resource

9.9 Conclusions

A proposal for the HLT / DAQ / DCS architecture of ATLAS has been presented in this chapter, and briefly summarized in Section 9.8. The architecture is based on the PESA conclusions and on the DAQ / EF -1 Project, LVL2 Pilot Project and DCS designs, and carries forward the principal design features. Many areas of the designs have been implemented in prototypes, as report-

ed in previous chapters. This proposal should serve as a basis on which to build the activities in the HLT / DAQ / DCS community in the next phase up to the TDR, currently planned for June 2001. Based on the proposed architecture and centred around the principal systems and subsystems described in the present chapter, a set of topics which should now be studied further and explored is given in Chapter 10.

9.10 References

9-1 *ATLAS technical proposal*, CERN / LHCC / 94–43 (1994)

9-2 J. Rumbaugh, I. Jacobson, G. Booch, *The unified modeling language reference manual*, Addison–Wesley (1998);
J. Rumbaugh, I. Jacobson, G. Booch, *The unified modeling language user guide*, Addison–Wesley (1998)

9-3 *High energy physics data grid initiative*, `http://grid.web.cern.ch/grid/`

9-4 *Trigger & DAQ interfaces with front-end systems: requirement document version 2.0*, ATLAS internal note, ATL–DAQ–98–103 (1998)

9-5 *The event format in the ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–98–129 (1998)

9-6 *Specification of the LVL1 / LVL2 trigger interface*, ATLAS internal note, ATL–DAQ–99–015 (1999)

9-7 *Summary document of the event building studies in the DAQ/EF -1 project*, ATLAS internal note, ATL–DAQ–2000–048 (2000)

9-8 *A logical model for event building in DAQ -1*, ATLAS internal note, ATL–DAQ–98–112 (1998)

9-9 *RD-31 status report, NEBULAS, a high performance data driven event building architecture based on asynchronous self routing packet switching networks*, CERN / DRDC / 95–14, DRDC / P36 (1995)

9-10 *Back-end summary document*, ATLAS internal note, ATL–DAQ–2000–001 (2000)

9-11 *Event filter summary document*, ATLAS internal note, ATL–DAQ–2000–005 (2000)

9-12 *Joint controls project (JCOP)*, `http://itowww.cern.ch/jcop`

10 Future Work

10.1 Introduction

This chapter indicates work in various areas that it is considered will have to be performed in the period up to the Technical Design Report (TDR) (at present scheduled for June 2001). At the time of writing the present document, a new organizational structure for the ATLAS Trigger, DAQ and DCS project is being set up. A detailed work plan (programme of work, schedule, milestones, resources), covering the period up to the submission of the TDR, will be prepared as soon as possible by the incoming Trigger, DAQ and DCS management, who will be responsible for the execution of the work in the context of this new organization.

The work performed in the last few years has addressed issues related to the LVL2 trigger, to DAQ/EF and to DCS in complementary and largely, separate projects. A result of this work is the logical architecture presented in Chapter 9, supported by the various implementation studies described in Chapters 5–7. A proof of principle has been established in many of the critical areas of the HLT/DAQ/DCS system. However, as described in the following, more work is required in order to select an implementation, make a high-level design, and to demonstrate convincingly its viability.

As discussed in Chapter 9, the HLT/DAQ/DCS system can be factorized into the following functional components:

- DataFlow (including ReadOut Subsystem, LVL2 DataFlow, EventBuilding (EB), EF I/O).
- Online Software.
- LVL2 Selection.
- EventFilter (EF).
- DCS.

A variety of implementation alternatives should be evaluated in which the functions of the LVL2 Selection and EF, and of the LVL2 DataFlow and EB are integrated to a greater or lesser extent, as discussed in Section 10.2. More work is required on the individual components, as discussed in Section 10.3. In addition, work is required in a number of general areas related to the overall system, as described in Section 10.4. This includes collecting more detailed requirements (from physics and the detectors), following issues related to interfaces with external systems, and system modelling and prototyping.

10.2 Implementation Alternatives

Following the DAQ/EF -1 project and the LVL2 Pilot Project, a range of implementation scenarios exist for the logical architecture described in Chapter 9. The principal differences concern the degree of integration between the LVL2 and DAQ/EF components.

The LVL2 implementation studied in the Pilot Project is potentially capable of performing also some of the functions of the DAQ and EF, namely full event building and subsequent event selection. It is therefore possible to envisage a scheme in which a single farm performs both the

LVL2 and EF processing. Even if there are separate farms for the two functions, one could envisage using common hardware and/or software components, allowing flexibility for moving computing resources between LVL2 and EF. On the other hand, there are benefits from separately optimizing the components for the LVL2 and EF tasks, as discussed in Chapter 9. A number of alternatives for passing the results of the LVL2 processing to the EF can be envisaged.

There are a variety of alternatives for overall event supervision. A single, integrated supervisor, managing event movement throughout the data-flow system, is a possibility. However, separate supervision systems can also be envisaged for LVL2 and for the EB, provided there is a means of communication between them, as discussed in previous chapters.

Another issue concerns the use of data-movement networks. In the case of separate processor farms for LVL2 and the EF, one can consider using a single physical network between the ROS and the two processor farms, or using separate networks. This concerns both the data-collection traffic and the associated control traffic. In fact, the more general issue of how to organize the network(s) for data movement, event-by-event control and other types of control needs to be addressed. An associated issue concerns the data-collection protocols for LVL2 and for DAQ/EF; they could be the same, but there may be benefits from separately optimizing the protocols for the LVL2 and DAQ/EF applications.

The work programme to be performed up to submission of the TDR should include studies of the issues discussed above.

10.3 Further Work on Components

In the following, issues that should be addressed relating to the various functional components described in Chapter 9 are identified. The LVL2 Selection and EF systems are discussed together in Section 10.3.3 in view of common issues, integration and networking aspects, that have to be studied.

10.3.1 DataFlow System

Future work in the ReadOut, LVL2 DataFlow, EB and EF I/O subsystems is addressed in this section. After discussing studies that are specific to the individual subsystems, we identify issues that need to be addressed in common.

10.3.1.1 The ReadOut Subsystem

There have been a number of activities in the area of the ROS during the last four years as discussed in Chapters 5 and 6, leading to the architecture presented in Section 9.3.1. In the next phase, the proposed ROS architecture should be developed in a single, coherent activity. Studies should converge on one or perhaps two complementary implementations to be evaluated in detail for the TDR.

Different solutions suggested for the hardware (see Section 9.3.1.3.4) must be compared, and the most appropriate ones retained. The expected evolution of relevant technology and Commercial Off The Shelf (COTS) products should be taken into account. Possible areas of hardware development are the use of co-processors for pre-processing (e.g. for data compression or

re-formatting) and the coherent evolution of today's implementations of the ROBIN (see Section 9.3.1.3). In addition, the use of *intelligent* network interfaces[1] for output to the LVL2 DataFlow and EB subsystems should be studied. Given the extensive hardware studies performed in the DAQ/EF-1 project and the LVL2 Pilot Project, and the limited time-scale to the TDR, future work should give high priority to the evolution of the design and implementation at the system level, building on the work that has already been done. New ROBIN hardware developments should only be undertaken after careful analysis of the potential benefits.

Related to the hardware is the software of the ROS. A suggestion for the interaction between components of the ROS architecture has already been described [10-1]. This design has been developed mainly within the DAQ/EF-1 project, and further discussion is necessary including the full community before the interaction model can be finalized. The ROS software design should then be implemented on existing ROS hardware.

An area that requires further study is implementation of pre-processing algorithms in the ROS which can, for example, perform data compression and re-formatting before data are transferred to the LVL2 DataFlow.

Another important issue is the connection of the ROS to the LVL2 DataFlow and EB subsystems. This connection is mainly driven by the requirements of the latter two subsystems; studies of the ROS will have to be flexible enough to accommodate different possible implementations in the next phase. Modelling will be used as one of the tools to help in the choice of detailed designs and implementations of the ROS. A common component is the local data collection that can be applied at the same time to HLT and DAQ requirements. The connection with other components of the HLT/DAQ can be handled by the DataFlow software, unless it is decided that the ROS will be a closed system interacting only by messages. In that case it may be possible to develop a specific interface. This choice may be proposed and discussed.

The interaction between the ROS and the LVL1 Trigger must be more thoroughly investigated. Three examples of this are:

- The possible use in the ROS of LVL1 trigger signals provided directly via the TTC system.
- The possible generation of BUSY signals at the level of the ROS to introduce dead time (in addition to applying back-pressure on the RODs using the XON/XOFF protocol on the ROLs).
- The need to label events in the ROS with an event identifier with a longer period of uniqueness than the 24-bit L1ID provided by the TTC system.

Prototype ROSs should be included in the HLT/DAQ prototypes (see Section 10.4.6). It may be that existing prototype ROSs can be used for this purpose, at least in a first phase. The test beds will have to run the prototype trigger algorithms, and the prototype ROSs should be able to provide realistic data to them via the rest of the DataFlow system. It should be possible to obtain the data either from test-data generators or from events stored locally in the ROSs. In the longer term such facilities will be useful for test and diagnostic purposes.

1. Intelligent network interfaces, sometimes referred to as intelligent network interface cards (NICs), have significant embedded processing power which can be used to relieve the host processor of protocol processing. The processing power might also be used, for example, to handle some of the higher-level data collection tasks.

A prototype ROS should be evaluated at the test beam, with connections to one or more flavours of subdetector RODs, in order to validate the design and implementation. This will also help to ensure that evolution of the test-beam DAQ remains coherent with ongoing ROS developments.

The aspects of fault tolerance of the ROS have not been studied to date and should be considered in the next phase. Similarly, the requirement of direct output to mass storage from the ROS, e.g. in stand-alone operation during commissioning, should be addressed.

10.3.1.2 LVL2 DataFlow

Work in the LVL2 Pilot Project (see Section 6) has led to the LVL2 DataFlow architecture described in Section 9.3.2. Although many issues have already been addressed in detail, more work is needed on the interaction between the LVL2 DataFlow subsystem and the ROS, to better evaluate the requirements and the performance aspects of RoI data gathering. In this respect, two specific topics that should be addressed in further modelling and prototyping studies are:

- The capability to gather fragments from hundreds of ROSs at the required rate (5–10 kHz for the inner-detector full scan alone).
- The capability to transport simultaneously RoI-type data traffic and inner-detector data for the full scan.

The distribution of requests and the collection of data, which are currently implemented by the Reference Software, could possibly be delegated to *intelligent NICs* (Network Interface Cards). This possibility should be investigated.

Evaluation, in test beds, of the suitability of Fast and Gigabit Ethernet for the LVL2 DataFlow application should be continued, both with respect to larger configurations and to the evaluation of the related hardware (switches and network interfaces). The protocol to run on top of the switching network also has to be addressed, including aspects of fault tolerance and error recovery.

A first prototype of the RoI-Builder–Supervisor (RBS) combination has been demonstrated to work at a LVL1 Trigger accept rate of 100 kHz. Nevertheless, much work remains to be done before the TDR:

- Testing the RBS using real inputs from prototype elements of the LVL1 trigger.
- Further prototyping of the RoI Builder which has been redesigned to use the smaller 9U Eurocard format in place of the current 12U one, making various improvements (e.g provision for longer ROI fragments, optical-fibre S-link inputs).
- Investigations are under way to determine the feasibility and possible benefits of using a four or eight-processor SMP[1] as the Supervisor in place of the current complement of PCs.
- A detailed study is needed to verify the proper operation of the RBS under all kinds of running conditions expected during the experiment, including test and calibration runs.

1. Symmetric Multi-Processors, or SMPs, are multiple processor systems that use a shared memory and a single copy of the operating system. The work load is balanced between the processors by the operating system. SMPs typically have up to 16 processors.

10.3.1.3 EventBuilding and EF I/O

A number of issues need to be addressed to continue the study and the development of the EB subsystem described in Sections 5.2 and 9.3.3):

- The continuation of the Gigabit Ethernet studies for the EB application, both with respect to larger configurations and to the evaluation of the related hardware (switches and network interfaces). This should expand on the work performed so far (eight × eight set-up) [10-2], using a larger set-up and including protocols other than TCP/IP (e.g. Gamma). Although TCP/IP has all the functionality required (and more), it generally implies a cost in additional overhead. The effectiveness of TCP/IP on different hardware should be studied and compared to the alternative solutions.
- As indicated in Section 9.3.3, the process of event building — i.e. the way sources and destinations interact to build an event — can be implemented according to different patterns of interaction. The different options should be evaluated within the overall HLT/DAQ architecture, and the control protocol should then be selected accordingly. In addition, the alternatives of using a single network for both data transfer and control messages, or of using separate networks, one for the transfer of event data and the other for the control messages, should be compared.
- Another issue that should be studied is the interaction between the EB and the subdetector calibration-trigger systems. This is particularly relevant for the concurrent operation of multiple partitions, where each partition has its own trigger source and dead-time logic independent of the central LVL1 trigger system.
- Issues related to mass storage of event data need to be kept in mind, following developments in the ATLAS Offline-Software project, and taking into account developments outside of ATLAS (CERN IT division, other experiments, CERN computing review).

10.3.1.4 Common Issues, Integration and Networking

Studies should be made of an integrated data-flow system, combining the LVL2 architecture developed in the Pilot Project, the EB, EF I/O and EF architecture from the DAQ/EF-1 project, and the ROS architecture which results from work in both projects. The studies of full-scan data collection, mentioned in Section 10.3.1.2 above, should also be extended to investigate full event building using the same protocol for all types of data collection, building on initial studies on the ATM test bed (see Section 6.8.2).

A related issue that should be studied is the interaction between the LVL2 DataFlow and EB subsystems. This concerns the transmission of the LVL2 trigger decisions, which may be grouped to reduce the rate of messages, and also the provision of detailed results of LVL2 processing that should be included in the built event (for use in the EF and offline). Alternatives for integrating the LVL2 and EB event-supervision systems should be investigated.

Building on developments in the DataFlow subsystems of DAQ/EF-1 and the LVL2 Pilot Project Reference Software, an integrated data-flow software framework should be provided for prototyping in the next phase of the project. The framework should be sufficiently flexible to allow the implementation alternatives discussed in Section 10.2 to be evaluated. It should be integrated with the Online software (see Section 10.3.2). Future software developments should aim at convergence on the methodology, languages, tools and platforms where appropriate. More generally, software-engineering methods should be used for the DataFlow software, building in

particular on the expertise acquired in the Back-End system of DAQ/EF -1. This will aid the integration of the overall HLT/DAQ/DCS system.

In the longer term, simulated detector data should be provided in the ROD output data format for use in prototype studies. Further work is required in the definition of this format — including the represenation of the data from the detectors — and ROB mapping (see Section 10.4.2), and the efficient conversion of ROD data into (e.g. C++) data objects suitable as input to algorithms (see Section 10.4.2).

A general study is needed of possible error conditions throughout the DataFlow system, identifying how to detect, recover from and monitor occurances of the various conditions that may occur.

10.3.1.4.1 Networking

It is now clear that, on the time-scale of LHC, industry will be able to provide most if not all of the networking equipment required for the ATLAS HLT/DAQ/DCS system. This is not to say the networking issue is solved, but rather that the path ahead requires further work in close collaboration with industry. A clear technical challenge is to build large networks capable of meeting our needs in terms of throughput, latency and quality of service. A complementary challenge is to achieve application-to-application performance close to that of the physical network, while if possible maintaining a standard API (application programmers interface).

Work on networking topics is suggested in the following:

- Studies of performance and scalability issues involving the modelling of large networks composed of many cascaded switches, with realistic ATLAS traffic patterns. Issues of network congestion avoidance must be addressed, as well as link aggregation (trunking) and multicast propagation.
- An evaluation of current network equipment, including likely future trends and products. Tests on large networks of several hundred nodes should be carried out. This should serve to calibrate and check the computer models used to predict the performance of the final system (see Section 10.4.5). Suitable test beds should be identified and tools to carry out the tests developed.
- A global review of networking throughout the ATLAS experiment with the aim of achieving rationalization and coherence. This should be considered taking into account the relationship with on-site and off-site networking, including issues related to computational GRIDs.
- The tracking of industry standards for computer networking, both hardware and software.
- A price estimate for all networking aspects of the HLT/DAQ/DCS system.
- A mapping of the different network architecture options onto Ethernet and ATM technologies.
- The HLT/DAQ networking design should allow for future expansion if required.
- Account should be taken in the design of the HLT/DAQ networking to allow for future expansion if required.
- The need for protocols above OSI layer 2 and the role of TCP/IP. The role of both layer-2 and layer-3 switching.

- Fault tolerance requirements in the networking.
- The use of quality-of-service[1] (QoS) features.
- The delivery of high performance and QoS to the application, in particular the following:
    - Consideration of operating system issues, such as process scheduling and network driver optimization. To what extent can standard off-the-shelf components be used?
    - The adoption of a suitable network API.
    - The use of intelligent NICs for LVL2 DataFlow and EB.
    - The use of multiple NICs in the ROSs and processors.

10.3.2 Online Software

The future work for the Online Software should be based on developing further versions of the required software using the current Back-End subsystem of the DAQ/EF-1 project (see Section 5.3) as a starting point. Since the Online Software acts as *software glue* for holding all the HLT/DAQ subsystems together, an important part of the work will be to ensure that suitable interfaces with the other systems and subsystems are developed. The principal areas of activity will include the following:

- Integration with other systems

  Developing integrated prototypes with other HLT/DAQ/DCS subsystems, LVL1 and detectors in laboratory and test-beam environments is seen as being the most appropriate means of producing the necessary interfaces.

  The intention is to use successive versions of the software in test-beam activities to provide experience in a real-life test environment. This will also allow the people from the detector groups to use the software at first hand and become familiar with the HLT/DAQ system prior to final integration. Training will be arranged for detector groups on how to use the Online Software for their test-beam activities.

  Use of the software will generate invaluable feedback to ensure the design, architecture and reliability are appropriate for the final system. It will also provide an excellent opportunity to test the interfaces with the rest of the experiment, focus attention on a common goal, and encourage a solution based on practicality and technical merit. Such an approach will enable the people involved to become familiar with delivering working systems and means they will appreciate the effort involved in moving from a test laboratory to real-life usage before the final experiment start-up date.

- Organization

  As a means of simplifying the development of the Online Software in a large, distributed collaboration and ensuring that rapid feedback from the other subsystems is possible, it has been proposed within ATLAS to develop the Online Software within the framework of an open-source project. A proposal [10-3] for such a framework has been made and has

1. Quality of Service protocols are being developed for use on large networks to provide consistent predictable delivery services. They provide tools to manage the bandwidth and allow different classes of traffic (e.g. with different latency requirements) to be handled with differentiated policies within the network.

met with general acceptance in the ATLAS HLT/DAQ community and detector groups. If adopted, the structure of the open-source project will ensure that a defined software process is used to guide software development, and the component architecture and designs will be used for future extensions.

By providing all the features of an open-source project, including access to the source code, it will enable those people not currently involved in the development of the software, but who rely on its functionality (such as people working in detector groups on detector-specific readout software), to participate more easily in the project. Adopting an open-source approach can also help reduce the work of individual subsystems and detector groups by allowing them to profit from each other's developments.

The steps described in the proposal [10-3] for converting the Online Software into an open-source project represent incremental improvements to the current Back-End organization and software suite. Should the proposal for an open-source project be adopted, a new document giving more details of the planning, resources, organisation and issues will be produced, and approval will be sought from the appropriate ATLAS experiment/institute bodies. This document could then be used to drive the development of the project and produce regular releases of the software.

- Addition of new components

  The Online Software, as outlined in the proposed future organization of the HLT/DAQ project, will include responsibilities beyond those defined for the Back-End subsystem of the DAQ/EF-1 prototype system. Additional responsibilities that include software development will be treated as extensions of the component set. The required software will be developed according to the software process and in conjunction with the detector groups and other HLT/DAQ systems. This includes the monitoring infrastructure and DCS SCADA interface.

- Design evolution

  We will continue to refine the design and implementations of the software components to meet the evolving needs of the experiment, and monitor the industrial trends to ensure that the system remains based on mainstream technologies. We will also track the activities of other LHC experiments that are considering using the DCS SCADA system also for supervisory functions of DAQ such as run control. This will allow us to understand how much of the overall experiment control could be performed by the SCADA system.

  The final implementation of the Online Software will depend on feedback from the test-beam activity and will use the most appropriate software technologies available closer to the experiment start-up date (technologies to be chosen approximately two years before commissioning).

10.3.3 LVL2 Selection and EventFilter

Future work that is required in the LVL2 Selection and EF areas is discussed in this section. After discussing studies that are specific to each of LVL2 and the EF, we identify issues that need to be addressed in common.

10.3.3.1 LVL2 Selection

Extensive studies have already been made of the LVL2 Selection process, as described in Chapter 6, leading to the architecture described in Section 9.5. Nevertheless, a number of issues need to be addressed further for the LVL2-specific aspects of the event selection process as described below.

More work needs to be done running prototype LVL2 algorithms on test bed systems to evaluate their performance (e.g. processing latency and rate capability) in a realistic online environment. This should include a complete set of feature-extraction algorithms (so far only examples have been implemented in the test-beds), with realistic input data provided by sources emulating the ROS functions. It should also cover the steering algorithms, using more complete menus and sequences than in the work done so far. In addition it should cover further measurements evaluating the benefits of using co-processors for CPU-intensive operations. Work in this general area should build on and extend the close collaboration between the algorithm developers in the Physics and Event-Selection Algorithms (PESA) activity and those involved in implementation studies. As described in Section 10.4.4, it should be possible to port LVL2 algorithms between the online environment and the offline framework.

Further work is required to evaluate pre-processing algorithms that may be implemented at the level of the ROS.

Another issue to be addressed is an investigation of the detailed mechanisms to deal with pre-scaling, tagging, forced-accept of certain LVL1 trigger types, etc. This is particularly required to cover the case where the final LVL2 decision is made in the LVL2 Processing Unit, rather than the LVL2 Supervisor, as this simplifies the transfer of the detailed summary of the LVL2 processing to the EventFilter.

The design and implementation of the Reference Software should be reviewed and further development and optimization made. Examples of further work in this area are:

- Enhancements to evaluate online performance; including further reducing communications overheads by optimization of the code.
- Optimization of the process-scheduling strategy to minimize the impact of queuing effects on the latency.

10.3.3.2 EventFilter

In the following, a few guidelines are presented for the next development phase of the EF system, based on the architecture described in Section 9.6:

- Identify in collaboration with the PESA, detector and physics groups ancillary tasks (monitoring, calibration analysis, etc.) which will be performed in the EF. Study how these tasks can be intertwined with the filtering task in order to save computing resources without jeopardizing the filtering function. Study the steering, possibly dynamic, of the ancillary tasks and how results can be returned to the end users.
- Assess the interplay between the flow of physics events and non-physics events (calibration, monitoring, etc.) in the farm, and study partitioning of the EF subsystem.
- Formulate and study possible schemes to implement full detector and physics monitoring, and calibration facilities in the EF environment, in collaboration with the Online software.

- Study the means of accessing offline calibration and alignment databases (and the associated versioning), and the implication of making those data available to of the order of a thousand nodes, in particular for a widely distributed architecture.
- Monitor the development of the new ATLAS Object Oriented (OO) reconstruction software and ensure that the requirements of the EF in terms of architecture, performance and robustness are met.
- Study a multiprocess implementation of the EF on SMP boards and compare its performances and behaviour (in terms of load balancing, error handling, robustness, flexibility) with the multithread solution.
- Evaluate the impact of non-homogeneous hardware/software implementations on the performance and operability of the EF subsystem, and investigate tools to handle these conditions.
- Investigate possibilities for a geographically distributed EF system, including implications of the GRID project [10-4].

10.3.3.3 Common Issues, Integration and Processor Technology

An important aspect of the work should be to evaluate alternative implementations with different degrees of integration between the LVL2 and EF event-selection systems (see Section 10.2).

In the following are some specific points that should be studied in the case where independent processor farms (or exclusive sets of processors within a single farm) are used for the LVL2 and EF selection:

- Develop a means to make the LVL2 trigger-type available to the EB system prior to event building (to allow events to be directed to appropriate logical partitions for special processing).
- Develop the mechanism(s) for making the full results of LVL2 processing available to the EB and EF systems.
- Understand the interaction between the event supervision systems of LVL2, the EB and EF. As part of this, the degree of integration between the supervision systems should be studied.
- Study the distribution of latencies for each of the LVL2 and EF stages. Effects related to disk and database access should be considered in the case of the EF.

In the case where the LVL2 and EF functions are implemented on the same processors, the following issues should be addressed:

- Study the distribution of latencies when running EF algorithms, based on offline code, concurrently with LVL2 algorithms in the same processor. Effects related to disk and database access should be considered.
- The ability to partition the system for dedicated calibration running (see Section 9.6).
- The implementation of tasks of the EF other than event selection (see Section 9.6), including monitoring, online processing for calibration and alignment, and event display.

A number of general issues need to be addressed:

- Understand the details of the event supervision throughout the LVL2 and EF processing stages, including issues of error handling, monitoring, etc.
- Identify, in collaboration with the PESA group, a complete selection and classification strategy for the HLTs, in order to achieve the best possible rejection whilst keeping the highest physics discovery potential.
- Use the knowledge gained in the studies of the PESA group and pursue further investigations in order to establish the level of flexibility needed in the HLTs, and understand how to map it onto different hardware implementations (moving selection algorithms and functionalities).
- Evaluate alternative approaches for the selection of *B*-physics events, including the possibility of restricting the LVL2 task to confirming the low-p_T muon found at LVL1, as well as the approach studied in more detail until now which involves performing a full scan of the inner detector at LVL2.
- Study the configuration aspects of the processor systems (LVL2 Selection, EF), including operating system issues, and their correlation with other parts of HLT / DAQ (notably Online Software). Also study the issue of synchronizing changes in the Offline and EF software.
- Study management and system monitoring of the computing resources for LVL2 and EF, and the integration of the management software with the Online Software. ATLAS should make every effort to benefit from the expertise of other experiments (both current and future), of large industrial, commercial and military applications (e.g. ASCI [10-5]) and the knowledge of the IT division at CERN and similar organizations elsewhere, in the area of handling large computer farms.
- Study the issue of coherently configuring the HLT system, together with the LVL1 system, in terms of event-selection criteria and associated parameters.
- Study complete error-handling procedures capable of keeping the data loss at a very low level and of dealing with failures in the whole system (restart components, kill unwanted processes, etc.).

Selection of the best processor technologies for the LVL2 and EF systems will be important from the point of view of performance, cost and other aspects. In terms of general-purpose processor technology, the following activities should be pursued:

- Continue the comparative evaluation and tracking of multi-PC farms and large commercial SMP machines. Both systems have advantages and disadvantages in terms of performance, maintainability, long-term upgrade paths and cost (both initial investment and overall cost over a period of ~ 15 years).
- Track processor chip technology evolution (e.g. Intel, Power PC, Alpha, etc.).
- Track future developments of local buses (e.g. higher-speed PCI and Infiniband).
- Packaging of processors taking into account requirements of compactness and water cooling for equipment to be installed at the experiment.

A number of issues need to be studied further concerning the possible use of FPGA-based co-processors, building on work done in the LVL2 Pilot Project (see Section 6.7). Areas of work that have been identified include:

- Detailed analysis of results from the implementation of TRT track finding on the ATLANTIS prototype, providing a check-list for future implementations.

- Implementation studies for track finding in the SCT and pixel detectors.
- Further integration of the ATLANTIS prototype system (see Section 6.7) in test beds, and detailed measurements of performance enhancement in an integrated environment.

Note that the possible use of FPGA processors within the ROS is discussed in Section 10.3.1.1.

10.3.4 Detector Control System

The future work internal to the DCS is addressed in detail in Section 7.7. However, we mention here the aspects related to the integration with the HLT/DAQ which have to be studied.

The need of direct interaction of the DCS with the detector readout electronics, in particular the ROD, has to be studied. This is related to the question of the availability of the Online Software system in the ROD area.

The connection of the DCS to the individual Online Software subsystems needs to be defined. Using the SCADA API tool prototypes should be implemented and validated in applications with subdetectors. Special emphasis has to be given to the overall experiment control during normal operations, i.e. the correlation between the DAQ Run Control and the DCS.

10.4 Further Work at System Level

10.4.1 Physics Requirements, and Event-Selection Algorithms and Strategy

In the future work, the links between online selection activities and physics studies should be reinforced. It has proven to be very useful to derive a classification strategy from the physics requirements, that in turn allows the definition of a list of selection criteria for the events to be accepted. The close co-operation between the physics groups and, in particular, the HLT groups, is therefore considered one of the essential ingredients for evolving further work in this direction.

The following list presents a number of issues that need to be addressed further in the areas of physics requirements, and event-selection algorithms and strategy:

- Widen the scope of some PESA activities to include studies of control channels, calibration and alignment triggers, and pre-scaled triggers, and determine the bandwidth needed. Modifications needed in the selection sequence and strategy in order to cope with changing external conditions with time (background level, luminosity, etc.) should also be addressed.
- Study LVL2 pre-processing algorithms.
- Study different scenarios for the sharing of the selection task between LVL2 and the EF, analysing the physics and system performance required and optimizing the global response of the system. This will allow one to understand the level of flexibility needed in the HLTs and how to map it onto different hardware implementations (moving selection algorithms and functionalities).
- Evaluate the impact of the knowledge about calibration, resolution, alignment, noise, fraction of dead channels, luminosity and magnetic-field data on the robustness of the se-

lection algorithms. Assess the tolerable limits as well as the precision and granularity needed at LVL2 and at EF for the different physics cases.

- Demonstrate the feasibility of moving algorithms from the offline suite to the HLTs, keeping a high selection efficiency while consuming only a limited amount of computing resources. The experience gained should be transferred into design documents that will help offline algorithm developers to shape correctly the new OO software for EF use.
- Assess more completely the suitability of FPGA-based co-processors for speeding up the execution of complex tracking algorithms at LVL2 (as already done in some *B*-physics cases). Compare the results in terms of execution time, efficiency and rejection with standard PC implementations.
- Continue development of a fast simulation program for trigger rate calculations (similar to fast simulation for physics analysis), with and without shower shape parametrization. This should be done in co-operation with the ongoing activities in the physics group.
- Develop a common framework for HLT selection, and, in particular, rationalize the situation at LVL2 where code is at present developed in a variety of frameworks, collecting the necessary requirements and building on the experience gained until now. This will be needed to define a coherent strategy for the online selection, and for implementing a complete LVL1 (simulation), LVL2 and EF selection software to pre-select events to be used in physics simulation studies.

Despite the progress made inside the PESA group, there are of course many activities that need to be completed or expanded in the future, in order to reach a comprehensive assessment of ATLAS online selection capabilities. In the following some items are listed as illustrative examples of future activities:

- Develop sample strategies for measuring trigger efficiencies, overlaps, etc.
- Extend the database of physics channels with corresponding dedicated menu items.
- Study, together with the physics groups, the possibility of assigning general quality flags to filtered events, in order to prioritize the offline processing.
- Make checks of the trigger performance with an updated detector layout.
- Demonstrate that the trigger algorithms give the required performance for data simulated with a non-ideal solenoid field.
- Study guidance of the EF by refined primary and secondary RoIs from LVL2.
- Study TRT LUT algorithm for LVL2 high-p_T electron trigger.
- Study possible EF high-p_T muon background rejection.
- Study muon isolation at LVL2 and EF.
- Extend to design luminosity studies of missing transverse energy using full simulation.
- Adapt LVL2 tracking code for tau selection.
- Study of tau selection at the EF level.
- Study the possible sharing of b-jet tagging algorithms between LVL2 and EF selection.
- Improve and refine the d_0 resolution and the fake track fraction of the b-jet tagging algorithm and study the effects of misalignment and of beam position stability.

- Extend the b-jet tagging studies to other physics channels and to the case of high luminosity.
- Extend the pixel scan to be able to incorporate SCT points.
- Develop code to extrapolate tracks from the SCT into the TRT.

10.4.2 Detector Requirements

The Detector Interface Group (DIG) has proved to be very useful, both for the DAQ/EF-1 project in particular and for the experiment in general. It has provided a forum for discussing issues of common interest between the detector and HLT/DAQ groups, particularly on DAQ and DAQ-detector readout-interface questions, see Chapter 3. This dialogue should continue in the next phase of the project. The group should identify and discuss issues and concerns of the detectors in the area of HLT/DAQ and see that they are transmitted to the HLT/DAQ group, and vice versa. The DIG should not be seen as the forum to solve problems, but more to identify, clarify and transmit them to the appropriate person or group. Here we present a non-exhaustive list of items which should be addressed by the DIG in the forthcoming development phase:

- ROD-Crate DAQ (see Chapter 3).

  The ROD-Crate DAQ will be based on an extension of the DAQ functionality of the DAQ/EF-1 ReadOut Crate.
- Use of DCS in the detector front-end electronics.
- Study the various use-case scenarios (calibration, monitoring, begin run configuration, stand-alone debugging, etc.) in detail in order to specify the DCS–ROD-crate interface.
- Busy and reset handling in the RODs.
- Database access from the ROD crates.
- Tracking and evolution of the DAQ–front-end interface definition [10-6].
- Refinements of the data format [10-7].
- Further definition of the detector data mapping in the ROBs [10-8].
- Further definition of the detector data content (e.g. data-compression schemes).
- Understand the distribution of detector monitoring requirements at the different levels (ROD crate, ROS, EF I/O, EF).
- System partitioning issues.

10.4.2.1 ReadOut Links

Future work to be addressed for the ReadOut Links (ROLs) includes the following:

- Fulfil all of the user requirements of the final ROL.
- Minimize the size, including investigation of two links on one PMC card.
- Decide on the chip-set that will be used in the final ROL.
- Assist designers in integrating link designs and layout onto ROD PCBs.
- Understand the 5V/3.3V/2.8V power-supply issues.

10.4.3 Test-Beam DAQ

The test-beam DAQ will be based on the DAQ/EF -1 system. Its future evolution and the overall ATLAS HLT/DAQ system development will be kept aligned. It must not become isolated and unconnected with the mainstream developments. A detailed schedule and work plan, together with responsibility and resource planning will be produced by the new Trigger, DAQ and DCS management.

10.4.4 External Interfaces

Future work relating to external interfaces concerns primarily the LVL1 trigger and the offline system, as discussed in the following. Note that the major issue of interfacing the HLT/DAQ/DCS system to the detectors is addressed separately in Section 10.4.2.

10.4.4.1 Interface to LVL1

Although a first step in defining the interface between the LVL1 trigger and the HLT/DAQ system has already been made, as described in Chapter 4 and in more detail in Ref. [10-9], much work remains to be done. Some specific points that require further attention are:

- Develop a strategy for coherently programming the selection criteria in LVL1 and the HLT.
- Define in more detail the configuration parameters that need to be shared between LVL1 and LVL2, and also the EF. For example, the LVL2 trigger has to know the value in GeV units for each of the six LVL1 muon p_T thresholds.
- Explore in detail how the LVL1 configuration information, together with data provided on an event-by-event basis to the RoI Builder, will be used in LVL2 and subsequently in the EF.
- Follow the evolution in the detailed design of the LVL1 trigger that may affect the data content and organization for LVL2.
- Follow developments in the design of the HLT/DAQ system as a whole that may have implications for the interface to LVL1.
- Take account of developments in the areas of the ROL and the data-format specification.
- Study the issues of error handling, monitoring, testing and diagnostics.
- Consider in more detail the treatment of calibration events within physics runs.

As mentioned in Section 10.3.1.2, it is planned to connect prototype elements of the LVL1 trigger with a prototype implementation of the RoI Builder.

10.4.4.2 Interface to the Offline System

The ATLAS offline software is now moving to a new OO framework [10-10]. The foremost benefit of moving to this framework is that it provides many of the features that we have already identified as requirements. It is also important that the trigger requirements have an influence on this software.

One of the main development streams for the future will be the provision of a complete chain of LVL1 (simulation), LVL2 and EF code, as pre-selection to the offline reconstruction. This suite will need to be *pluggable on-demand* into the general architecture and will be used by physicists who want to check the effects of trigger selection on their analysis.

In this context, many contact points with offline exist, that need to be monitored carefully in order to achieve the best possible integration of the trigger code in the new framework. Among the general requirements that online imposes on offline, one should mention coping with the online event format, in particular for EF use, and designing the algorithm interface (i.e. the classes of the transient store) in such a way as to be usable at various levels of online selection code, including LVL2. Related to this is the need to provide data in a suitable format for evaluation of the LVL2 and EF algorithms in a prototype online environment. Software will also have to be developed to translate the *online data* to the objects of the transient store that are input to the algorithms.

The other field where a big effort should go in the next phase is the study of the new OO software in the demanding EF environment. This should address questions related to the performance of the software, its robustness, and the use of external resources such as conditions databases and OODB storage mechanisms. Other items to be studied include the shaping of the algorithms (i.e. ordering the selection steps to allow the best possible rejection with minimal processing); the inclusion of parameters in the reconstruction code that can be adjusted to balance reconstruction performance against execution time (e.g. magnetic-field grid size, track-matching criteria, vertex identification, etc.); and the guidance in selected parts of the detector using the results of LVL2 or secondary RoIs indicated by LVL1.

10.4.5 Modelling

Much work has already been done to understand the implications of trigger menus, sequences and associated rates, and of data volumes per ROB and per RoI, using an Excel-based *paper model* of the ATLAS HLT/DAQ system. This model has been used to predict data and message rates at each step in the LVL2 processing, and to evaluate the number of processors and interfaces needed to support this flow. The model will be refined and updated to follow changes in the trigger strategy and our understanding of the parameters (rates, data volumes, component performance, etc.). It will continue to serve as a guide for more sophisticated modelling work and as a quick-and-easy test-bed for exploring the implications of implementation alternatives.

The more detailed models based on the SIMDAQ C++ code and the Ptolemy system will be needed to provide an in-depth picture of system performance throughout the HLT/DAQ system. The first steps have been taken in comparing these simulations with measurements from existing test-bed set-ups. This process will continue and be expanded as additional measurements are made. Measurements from the prototype programme will be used to tune the models and their parameters, and to ensure that the models are correct and predict reliably the performance of real systems of significant size.

A next step in the work is to use the models to understand and refine the system picture and to improve overall system performance by adjusting component behaviour. The discrete event simulations will allow us to both characterize expected system performance and to optimize it.

The above efforts are reasonably advanced already and will continue to improve our understanding of the full system and its relationship to existing test measurements. As the final architecture and component choices are made, the models will guide these choices, as well as track

them. In particular, modelling should be used to assess the merits of implementation alternatives with different degrees of integration between the LVL2 and EF functions. Ultimately, the models should be used to predict the performance of the full system.

Complementary to the global system-level modelling discussed above, an important activity will be detailed modelling of specific network technologies, using specialized tools where appropriate. Such detailed models, combined with measurements, should be used to develop more abstract models for use in the global system modelling.

Modelling of the ROS should also be performed, both in isolation and as an integrated component of the overall system.

10.4.6 Prototyping

An important aspect of the work up to the TDR will be a full-slice prototype implementation, integrating the DAQ, LVL2 and EF functions, as well as the DCS where appropriate. The prototype should be as flexible as possible to allow the implementation alternatives mentioned in Section 10.2 to be evaluated. It should build on the work done in the DAQ/EF -1 project and the LVL2 Pilot Project, as well as the DCS project, wherever possible.

Aims of the prototyping programme should include the following:

- Assess alternative implementations of the logical architecture.
- Study performance scalability within the limits of the size of the available test systems.
- Gain experience in the integration of the various elements of the architecture.
- Provide measurements for use in modelling (determine free parameters, check predictions, e.g. as a function of system size).
- Demonstrate the viability of the integrated HLT/DAQ/DCS architecture and its implementation.
- Demonstrate the use of realistic selection algorithms in the online environment.
- Evaluate the flexibility of the LVL2–EF boundary in terms of rate and physics.

As stated in Section 10.1, the detailed work plan for the period up to the TDR will be prepared as soon as possible. The following observations should be taken into account when the work plan for the prototype is developed.

Construction of the prototype system may require building prototype components where commercial solutions (COTS) are not available or are too expensive. Where appropriate and possible, hardware components from the LVL2 Pilot Project and/or DAQ/EF -1 systems should be re-used.

Existing software components and designs should be reused where appropriate and possible, although new software will have to be developed in certain areas.

An approach that has already been used successfully in the LVL2 Pilot Project and in the DAQ/EF -1 project is to implement a number of reduced-size *clones* of the prototype system (or parts of it). For development purposes, this allowed people to work in their home institutes, on modest-size systems, before making final measurements on larger systems. In addition to the

obvious advantages of reduced travel, this allowed parallel independent work to be performed. Such an approach should be considered for the future work.

Another important approach that was found to be valuable in the LVL2 Pilot Project (see Section 6.3) and in the DAQ/EF -1 project [10-11] was to make use of very large existing computer systems that are available to us. Such systems give the possibility to make specific detailed measurements and scalability studies. Again, this approach should be pursued as an ingredient of the future work programme.

10.5 References

10-1 *The design of the DAQ-unit in ATLAS DAQ/EF prototype -1*, ATLAS internal note, ATL–DAQ–2000–054 (2000)

10-2 *DAQ/EF -1 event builder system on Linux / Gigabit Ethernet*, ATLAS internal note, ATL–DAQ–2000–008 (2000)

10-3 R. Jones, *A proposal for converting the ATLAS DAQ back-end subsystem into an open source project*, Proc. Computing in High Energy Physics Conference, Padova, Italy, February 2000, `http://chep2000.pd.infn.it/paper/longpap-b389.pdf`

10-4 *High Energy Physics Data Grid Initiative*, `http://grid.web.cern.ch/grid`

10-5 *U.S. Government Advanced Strategic Computing Initiative (ASCI)*, `http://www.sandia.gov/ASCI/ASCI-links.htm`

10-6 *Trigger & DAQ interfaces with front-end systems: requirement document – Version 2.0*, Atlas internal note, ATL–DAQ–98–103 (1998)

10-7 *The event format in the ATLAS DAQ/EF prototype -1*, Atlas internal note, ATL–DAQ–98–129 (1998)

10-8 *Detector and read-out specification, and buffer–RoI relations, for level-2 studies*, ATLAS internal note, ATL–DAQ–99–014 (1999)

10-9 *Specification of the LVL1/LVL2 trigger interface*, ATLAS internal note, ATL–DAQ–99–015 (1999)

10-10 *Report from the architecture task force*, `http://www.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/atf/report/report_2.7.ps` (1999)

10-11 *Event filter summary document*, ATLAS internal note, ATL–DAQ–2000–005 (2000)

A Participating Institutes

The following is a list of the institutes which are participating in the ATLAS High-Level Triggers, Data Acquisition and Detector Control System project, together with the names of the physicists and senior engineers concerned. Short names are given for each institute; for the official name and location see the opening pages of this document.

Alberta
B. Caron, B.R. Davis, P. Green, J. Pinfold

Ankara
O. Cakir

Argonne
R. Blair, L. Price, E. May, J. Schlereth, G. Drake, J. Dawson

Barcelona
M. Bosman, M. Dosil, K. Karr, A. Pacheco

Bern
H.P. Beck, S. Kabana, G. Lehmann, R. Mommsen, K. Pretzl, A. Radu

Bucharest
E. Badescu, M. Caprini

CERN
G. Ambrosini, R. Bock, J.A.C. Bogaerts, M. Boosten,[1] D. Burckhart-Chromek, H. Burckhart, R. Dobinson, M. Dobson, N. Ellis, A. Fernandes, D. Francis, F. Giacomini, B.I. Hallgren, Y. Hasegawa,[2] R. Heeley, M. Joos, R. Jones, J. Lopez, L. Mapelli, B. Martin, R. McLaren, G. Mornacchi, M. Niculescu, J.O. Petersen, D. Prigent, J. Rochez, T. Shears, S. Tapprogge, L. Tremblet, G. Unel, H.C. Van Der Bij, F. Varela,[3] P. Werner

Copenhagen
H. Bertelsen, M. Dam, J.R. Hansen, A. Wäänänen

Cracow, Henryk Niewodniczanski Inst.
Z. Hajduk, W. Iwanski, K. Korcyl, J. Olszowska

Geneva
A. Clark, I. Efthymiopoulos, L. Moneta

Genova
M.Dameri, P. Morettini, F. Parodi

Innsbruck
B. Epp, V.M. Ghete, E. Kneringer, D. Kuhn, A. Nairz, M. Schaller, D. Schweiger

Istanbul
E. Arik, S. Cetin, T. Conka-Nurdan, A. Mailov, K. Nurdan

1. also at Technische Universiteit Eindhoven.
2. now at Department of Physics, Shinshu University.
3. also at University of Santiago de Compostela.

JINR
I. Alexandrov, V. Kotov, M. Mineev, V. Roumiantsev, V. Yambourenko

KEK
H. Fujii, A. Manabe, Y. Watase, Y. Yasu

Lancaster University
M. Smizanska

Lisbon
A. Amorim, A. Ribeiro

Liverpool
J. Lokier

London RHBNC
S. George, B. Green, D. Hutchcroft, J. Strong

London UCL
P. Clarke, R. Cranfield, G. Crone, P. Sherwood, S. Wheeler

Mainz
L. Köpke

Manchester
R. Hughes-Jones, S. Kolya, R. Marshall, D. Mercer

Mannheim
C. Hinkelbein, K. Kornmesser, A. Kugel, R. Männer, M. Müller, M. Sessler, H. Simmler, H. Singpiel

Marseille
C. Bee, P.Y. Duval, F. Etienne, E. Fede, C. Meessen, R. Nacash, Z. Qian, F. Touchard

Michigan State
M. Abolins, R. Brock, Y. Ermoline, B. Gonzalez Pineiro, R. Hauser, B.G. Pope

Moscow SU
N.V. Nikitin, F.K. Rizatdinova, S.Y. Sivoklokov, L.N. Smirnova

Nagasaki
Y. Nagasaka, Y. Tanaka

NIKHEF, Amsterdam
H. Boterenbrood, R.G.K. Hart, W.P.J. Heubers, P.P.M. Jansweijer, G.N.M. Kieft, R. Scholte[1], J.C. Vermeulen

Novosibirsk
I. Tikhonov

Pavia
A. Negri, G. Polesello, D.A. Scannicchio, V. Vercesi

1. also at University of Twente.

Petersburg NPI
V. Filimonov, S. Katunin, A. Kazarov, V. Khomoutnikov, S. Kolos, Y. Ryabov, I. Soloviev

Prague AS & CU
F. Hakl, M. Jirina

Protvino
S. Kopikov

RAL
J.T. Baines, A. Belias, D.R. Botterill, R.P. Middleton, F.J. Wickens

Rome I 'La Sapienza'
A. Di Mattia, S. Falciano, C. Luci, L. Luminari, F. Marzano, G. Mirabelli, A. Nisati, E. Pasqualucci, S. Robins, S. Veneziano, L. Zanello

Rome III 'Roma Tre'
C. Stanescu

Saclay
J. Bystricky, D. Calvet, J. Ernwein, O. Gachelin, T. Hansl-Kozanecka, M. Huet, P. Le Dû, I. Mandjavidze, M. Mur

Sheffield
C.N. Booth, M. Lehto

Technion, Israel Inst. of Technology
S. Dado, N. Lupu, S. Tarem, Y. Rozen

TRIUMF
M. Wielers

Tel Aviv
H. Abramowicz, E. Etzion

TUC Irvine
A.J. Lankford, M. Schernau

Udine
F. Scuri

Weizmann Institute of Science
E. Duchovni, E. Gross, A. Klier, M. Kupper, D. Lellouch, L. Levinson, G. Mikenberg

Wisconsin
S. Gonzalez, R. Jared, S. Qian, W. Wiedenmann, S.L. Wu, H. Zobernig

This document has been prepared with Release 5.5 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