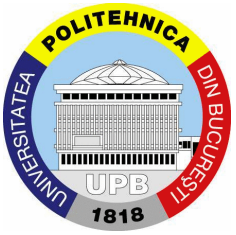

Universitatea "POLITEHNICA" București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Teză de doctorat

Analiza și modelarea predictivă a performanței rețelei de achiziție de date a detectorului ATLAS

Analysis and predictive modeling of the performance of the ATLAS TDAQ network

Doctorand **ing. Lucian LEAHU**

Conducător științific prof. dr. ing. Dan Alexandru STOICHESCU

prof. dr. ing. Vasile BUZULOIU

2013

Abstract

After almost twenty years of research, development and installation, the Large Hadron Collider (LHC) accelerator at CERN produced its first collisions in 2008, planning to run until the end of 2012. ATLAS (A Torroidal LHC ApparatuS) is the biggest experiment built and operated on the LHC ring. Being a general purpose detector, it studies a wide range of physics aspects, out of which the search for the "God particle" - Higgs boson - is its most significant mission. In 2012 ATLAS already recorded collisions data, called *events*, which were, with a big probability, candidates for proving the existence of this particle. Capturing this type of "interesting" events is the task of the ATLAS detector, however filtering them from the huge amount of data being generated is the purpose of the Trigger and Data Acquisition system (TDAQ).

ATLAS TDAQ is implemented as a three layer filter, reducing in real-time the rates of the events (1.6 Mbytes big) down to a level which can be written to mass storage: from 40 MHz (64 Tbytes/s) to 200 Hz (320 Mbytes/s). This real-time selection is performed using dedicated hardware in the first level and large farms of computers in the next two levels, interconnected by a dedicated high speed Ethernet network. The efficiency of the TDAQ system is given by the continuity of the events flow and by its capability of sustaining the design rates. Level-2 is a key filtering system because it applies physics algorithms to a 100 KHz rate of events and clears 97% of the read-out buffer space, critical to the flow continuity. The high rate of event processing in Level-2, involving data analysis and data transport over the network, places strict requirements on the total processing time. A contributing factor to this is the network delay and loss, for which requirements weren't strictly established from the design stage.

We set upper limit requirements for the network delay and loss as being the total Level-2 processing time. This is obtained by employing a mathematical queuing model on the Level-2 system. One of the limitations of this model is that it cannot differentiate the delay and loss caused by the network only. In order to overcome this, we introduce an approach applicable to any communicating systems, hence Level-2 network as well, incorporating both loss and delay into a central concept named *quality attenuation* (ΔQ). For data networks we propose a component-wise view of ΔQ allowing us to perform topological compositions which, for the Ethernet case, are easily applicable. We show how this method of finding contributors to the overall ΔQ is a lightweight, cheap and non-intrusive technique, applicable in system's operational phase.

We obtain on one hand performance indicators for entire network paths and for individual network devices, called *Structural Delay* and on the other hand a prediction on how the Level-2 network's ΔQ scales with the load placed on the system. For this we quantified the degree of correlation of the traffic pattern placed on the network by the TDAQ software. The ΔQ dependency on the load will then serve as a requirement trade-off space between the network and the software generating a type of traffic pattern.

Acknowledgments

At the beginning and continuously during the work for this thesis I was told that a PhD is something that pushes the knowledge further. My attempt in achieving that wouldn't have been possible if it wasn't for the help, guidance, mentoring and patience received from great people during these years.

I would like to dedicate this thesis to the regretted Professor Vasile Buzuloiu who guided my very first steps in career and was my supervisor for many years until he sadly disappeared. His young spirit, energy, determination and remarkable mind were and still are of great inspiration for me. I would like to thank Professor Dan Alexandru Stoichescu for accepting the role of supervisor for the last stage of my thesis and for his advices and final reviews of my work.

I am especially grateful to Dr. Neil Davies for mentoring me during all these years and leading me into an exciting domain he and his team at PNSol pioneered. Although remotely, our long and numerous discussions gave shape to this thesis and modeled my way of perceiving performance, networks, engineering and mathematics.

My friends and former colleagues at CERN played a key role in the birth and evolution of this thesis. During four years I had the chance to work and learn in a very challenging and special environment, mainly driven by the philosophy of managing people of Mr. Brian Martin. I would like to express my sincere gratitude to him for all the effort invested in this work and, I need to mention, for the patience. Many of the practical aspects of this thesis were discussed and refined with him.

I would also like to thank Silvia, Stefan, Wainer, Eukeni for their thorough reviews, technical debates and practical help in the last phases of the thesis. I had memorable moments with them and with Adina, Matei, Lavinia, Marius, Laura, Mihai during my work and years spent at CERN. I want to mention and thank Mihai Ciuc from the LAPI laboratory in Bucharest for his help and guidance on both theoretical and administrative sides of the thesis.

I am equally grateful to my family and friends for their continuous support and encouragment.

A big "Thank you!" to all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 CERN

The second decade of the twenty-first century witnessed a huge accomplishment in particle physics science through the implementation and successful operation of the largest particle accelerator in the world: the LHC (*Large Hadron Collider*) [37] hosted at CERN [8]. It is the result of almost twenty years of international collaboration which involved research, development and construction of the 27 km long accelerator ring and of four major detectors accommodated by it.

The history of CERN goes back to 1949 when the French physicist Louis de Broglie put the first official proposal for the creation of an European collaboration laboratory, followed in 1952 by the foundation of a provisional body Council from which CERN takes its name: ***C**onseil **E**uropéen pour la **R**echerche **N**ucléaire.* In 1954 the Council was dissolved and, although its original name of CERN was retained, it was replaced by a different consortium called the *European Organization for Nuclear Research*. Nowadays, as the current understanding of matter goes beyond the level of nucleus, CERN is more frequently referred to as the *European Laboratory for Particle Physics.*

1.1.1 Physics

Keeping its main and initially stated mission to *"provide for collaboration among European States in nuclear research of a pure scientific and fundamental character"* CERN's central activity of today is to exploit the LHC and its experiments for studying high energy proton-proton collisions. Its greater purpose is to find the missing pieces in the current description of the fundamental structure of matter, called the *Standard Model.*

Physics theories dating from the past century have converged into an understanding of the Universe as being made from twelve fundamental particles and four fundamental forces: *gravity, electromagnetism,* the *weak nuclear force* and the *strong nuclear force*. The Standard Model was developed in the seventies and consolidates the theories on these twelve particles and three

of the forces. Although it is able to predict and explain many phenomenons and experimental results it does not capture the entire picture. It omits one fundamental force - *gravity* and it didn't experimentally confirmed the existence of a particle called *the Higgs boson* [36] which would explain the origin of particle mass.

The formalization of the electromagnetism took place in the 1860s and a century later theoretical links between electromagnetic and the weak force were developed, confirmed by the discovery of W and Z particles at CERN. This discovery was possible only under high-energy conditions such as produced by a particle accelerator. Further experiments showed that the effect of the strong force becomes weaker with the energy increase, hence at some point the electromagnetic, weak and strong forces are likely to be the same, i.e. they are unified. The theories say that the energies required in this case are extremely high and were reached only at the very beginning of the Universe: 10^{-34}s after the Big Bang. By extrapolating this behavior physicists look at the possibility of adding gravity to the mix, thereby unifying all the fundamental forces into a single "super force".

Energies high enough to test these ideas directly are impossible to be created at present, however the effects of the grand unification could be studied at lower energies. Using the idea of *Supersymmetry* (often abbreviated *SUSY*), which states that for each known particle there is a supersymmetric partner (superpartner), the detection of one particle could be done through the detection of its superpartner. This is possible at the energy scale of tera-electronvolts (TeV) which is currently achievable at the LHC.

1.1.2 The Large Hadron Collider

The LHC is the largest and most powerful particle accelerator built to date aiming to produce the physics required energy scale by proton-proton collisions. It consists of a 27 km ring of 9300 superconducting magnets accommodated in the tunnel of the former *LEP (Large Electron-Positron Collider)* accelerator, located 100 m underground and crossing the France-Switzerland border in Geneva area.

An accelerator's provided energy is directly related to its size and in the case of a circular one its energy is dependent on the radius: 4.3 km for LHC. By design the LHC accelerates two counter-rotating beams of protons of 7 TeV each, developing at collision an energy of 14 TeV[1]. The energy density and temperature provided by LHC are similar to those that existed a few moments after the Big Bang.

Obtaining this energy requires bringing and maintaining the particles at a speed close to the speed of light. This is achieved by the use of powerful electromagnetic systems and vacuum cavities[2] which will operate at extremely low temperatures: $4.5\,K$ (-268.7^oC) for the cavities and $1.9\,K$ (-271.3^oC)[3] for the magnets. These low temperatures are required by the LHC mag-

[1] 1TeV is the approximate energy of a flying mosquito, LHC however focuses this energy into a 10^{13} times smaller space

[2] eight per beam

[3] lower than the temperature of outer space (2.7 K or –270.5°C)

nets as they are made from niobium-titanium cables which become superconducting (conduct electricity without resistance) below a temperature of $10\,K$ $(-263.2C)$. The electrical current flowing through the magnets is 11850 A to create the magnetic field of 8.33 T, required to maintaining a circular path for the beams.

In order for collisions to occur the counter-rotating beams are intersecting in four locations called experiments. LHC hosts four major experiments, each built around a particle detector machine, as pictured in Figure 1.1:

1. *ATLAS* – A Toroidal LHC ApparatuS
2. *ALICE* – A Large Ion Collider Experiment
3. *CMS* – The Compact Muon Solenoid
4. *LHCb* – Large Hadron Collider beauty experiment

Each detector studies the results of particle interactions by specific means and purposes.

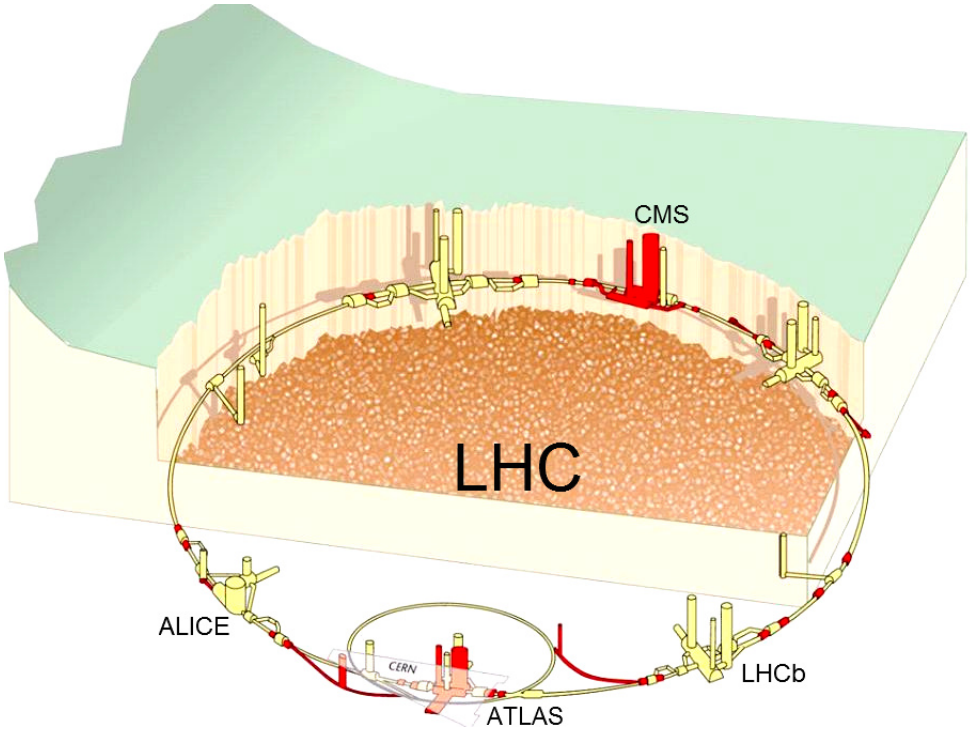

Figure 1.1: The LHC and its experiments

ATLAS and CMS are general purpose detectors and will investigate similar aspects, but having two independently designed detectors is required for cross-confirmations of new discoveries. ALICE and LHCb are specialized detectors for analyzing specific phenomena.

The LHC has been operated in 2010 and 2011 with collisions at half of the design energy, i.e. 3.5 TeV protons and in 2012 the proton energy has been increased to 4 TeV. It is expected that either the full or almost the full nominal proton energy can be reached after the long shutdown foreseen for 2013 and 2014. The maximum nominal instantaneous luminosity for proton-proton collisions was initially (i.e. in 2008) $10^{27} cm^{-2} s^{-1}$and has increased with time up to a maximum value of $6.7\,10^{33} cm^{-2} s^{-1}$in May 2012, being the closest to the design luminosity ($10^{34} cm^{-2} s^{-1}$).

The screen-shot in Figure 1.2 shows a typical run profile with stable beams ($B1$ and $B2$) reaching energies of maximum 4 TeV at the beginning of the run and linearly decreasing until 2.7 TeV at the end of the run. This also displays the state of the detectors on the LHC ring as the beams will be delivered only when all four are in STANDBY mode, hence ready for collisions.

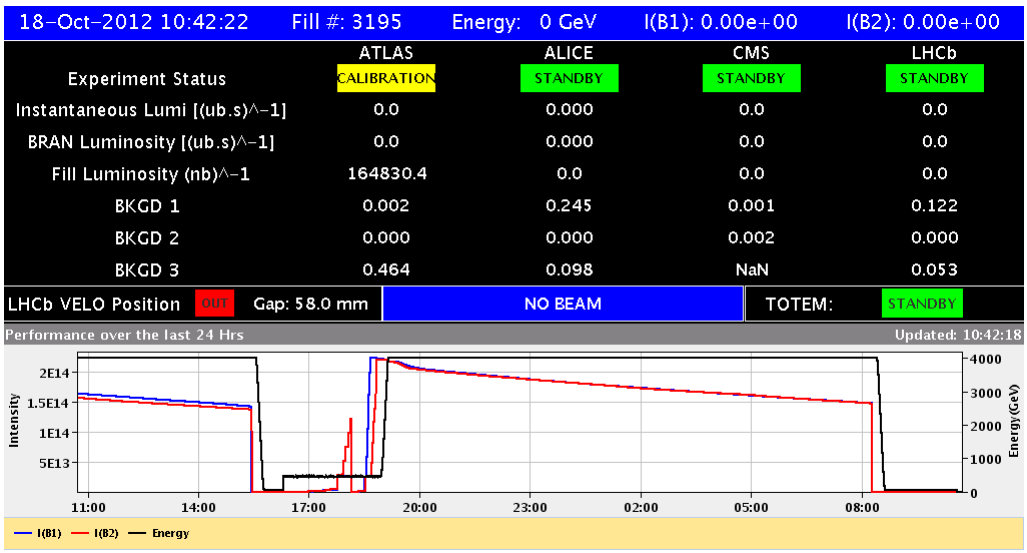

Figure 1.2: LHC run example

1.1.3 Operational costs

The LHC and its experiments are a big investment in knowledge expansion both from an intellectual and financial point of view. On top of the fixed cost demanded by the entire system in the design and implementation phases there is the operational cost of running the LHC. This encompasses human resources, infrastructure and most importantly power consumption.

A large portion of the electrical power is required to keep the temperatures in the LHC and the detectors at nominal values, that is, for cryogenic systems. The total power consumption for LHC is around 120 MWH (230 MWH for all CERN), which corresponds more or less to the power consumption for households in the Canton (State) of Geneva[4].

[4]assuming an average of 270 working days for the accelerator the estimated yearly energy consumption of the LHC is about 800 000 MWH

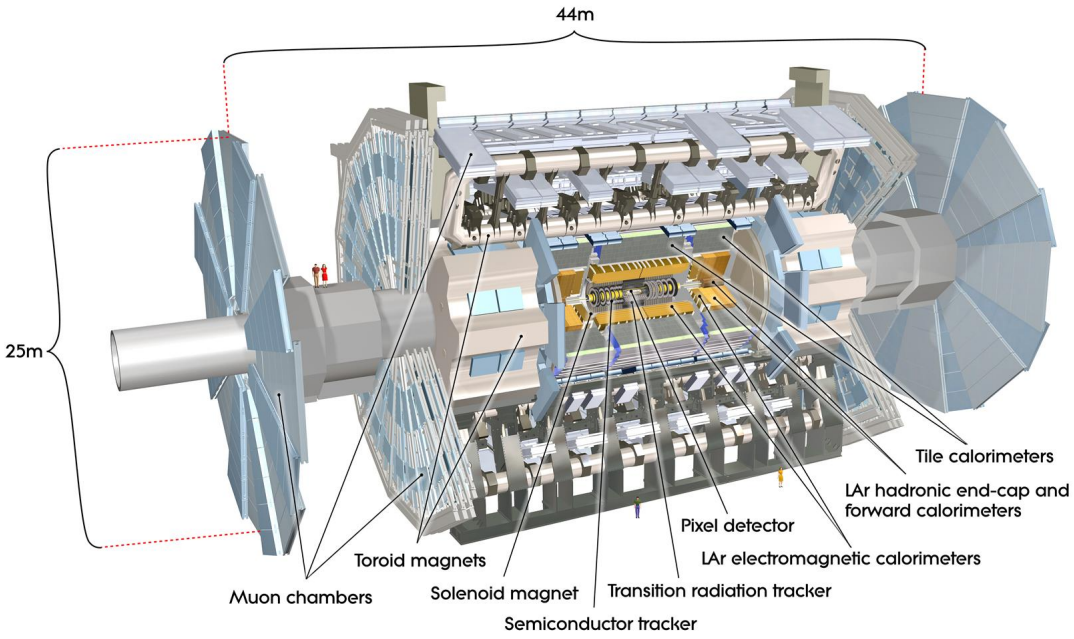

Figure 1.3: The ATLAS Detector

1.2 The ATLAS Detector

ATLAS [5] is one of the four major experiments accommodated at CERN and placed on the LHC ring. Started in 1992, the ATLAS detector has reached its final installation stages synchronously with the completion of the LHC accelerator, in September 2008. ATLAS is capable of investigating a wide range of physics due to it's general purpose construction, it's main goals are nonetheless the search for the Higgs boson, extra dimensions and particles that could make up the dark matter.

1.2.1 Architecture and event detection

ATLAS is accomplishing its objectives through four different sub-detectors which capture the momentum and energy of particles. Figure 1.3 displays the cylindrical architecture of the ATLAS detector in which the sub-detecting systems are placed on different layers. A collision takes place in the center of the detector. In this way the generated sub-particles will traverse and leave their "signatures" in all sub-detectors, from the innermost to the outermost layer:

Tracking chamber $\rightarrow$ *Electromagnetic calorimeter* $\rightarrow$ *Hadron calorimeter* $\rightarrow$ *Muon chamber*

The physics term for the whole set of traces belonging to a single proton-proton collision is an *event* and the data recorded by the detector is referred to as *event data.*

The first collisions at low energies were detected in 2009. At the beginning of 2010, proton-proton collisions were successfully performed at 7 TeV and currently ATLAS is taking data from collisions at high luminosity levels. In 2012 a number of events were considered as candidates for the identification of the Higgs boson. Figure 1.4 shows a three-dimensional view of one of the Higgs candidate events with two electrons & two muons.

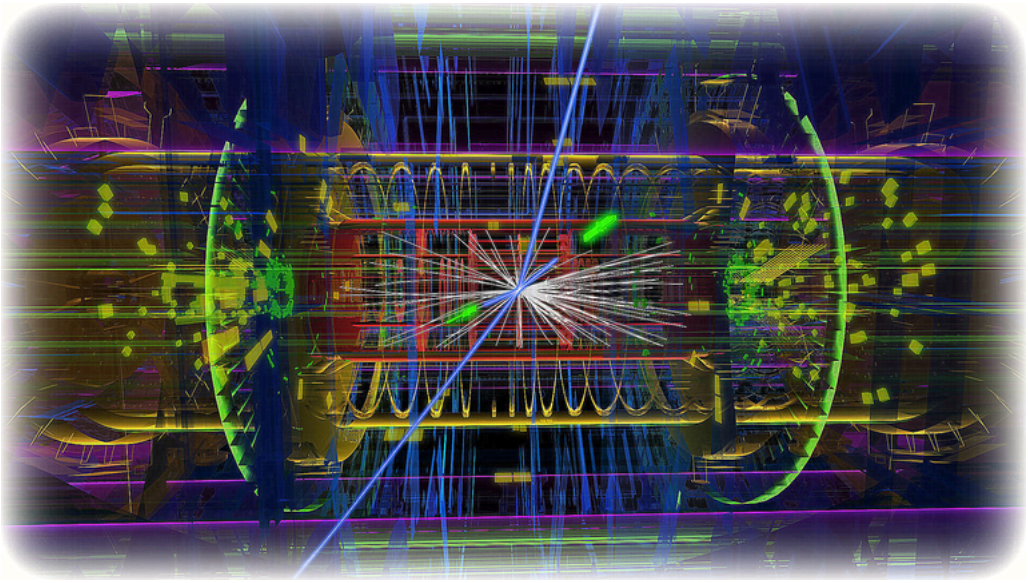

Figure 1.4: Collision detected in ATLAS - Higgs candidate event

The multilayer detector translates the huge amount of information generated by the proton-proton collisions into digital data. This amount is given by two main factors: the number of "sensors" (of the order of 10^8) spread across its entire volume - about half that of the Notre Dame Cathedral in Paris, measuring 25 meters high and 45 meters long - and the frequency at which collisions occur. The result is a 1.6 Mbytes event data occurring each $25\,ns$ (40 MHz frequency), resulting in a total data output rate of 64 Tbyte/s. Dealing with the huge amount of data is the mission of another sub-system in the ATLAS experiment which will be introduced next.

1.2.2 Trigger and Data Acquisition System

One of the main goals ATLAS hopes to achieve is finding the Higgs boson. The associated *Higgs field* would explain why some particles have mass and others don't by the fact that mass is a consequence of the interactions with this *field*. This new particle, postulated theoretically by Peter Higgs in 1964, is expected to occur in extremely rare situations. It occurs for a very short period of time, as it rapidly decays into other particles, as a result of other particle's decay. The only way to detect its presence is to look for its predicted decay products and identify those that

can't be accounted for by other processes, i.e. to capture them and to deliver them for offline analysis and validation.

The frequency at which events holding useful information occur is extremely low ($\sim 10^{-14}\%$ of the total number of events, i.e. $\sim 4 \cdot 10^{-5} Hz$). Given the huge amount of data generated by the detector - technically impossible to be retained, the biggest challenge for the *Trigger and Data Acquisition (TDAQ)* system becomes the efficiency in selecting those rare events and discarding the rest. The TDAQ system can thus be viewed as a real-time digital filter responsible with physics data selection and delivery.

This difficult task is accomplished at present by the use of three filtering levels: Level-1, Level-2 and Level-3. Each of them selects "interesting" events from the ones provided by the previous level through the use of specific algorithms applied on the event data.

The final selection and analysis will be performed offline at CERN but also remotely by the contributing universities and laboratories around the globe, hence the selected events must be written to mass storage for later use. As depicted in Figure 1.5, the input for Level-1 is the data coming directly from the detector, whereas the output for Level-3 is the mass storage - also known as the Tier-0 center at CERN.

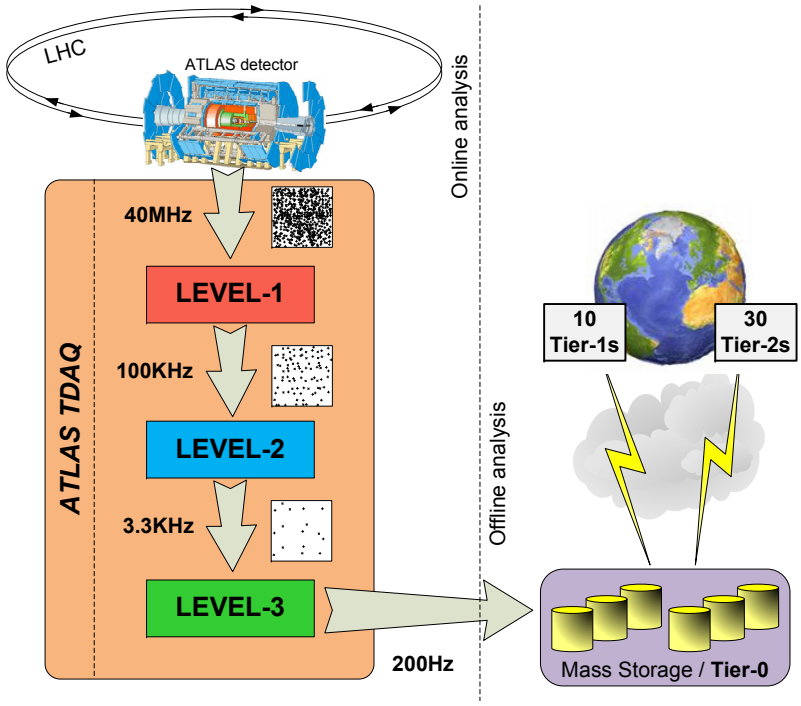

Figure 1.5: ATLAS TDAQ Diagram (rates as per design [24])

From this point the data is shipped to 10 Tier-1 and 30 Tier-2 centers distributed worldwide over long haul connections. The Computational Model for this activity which involves the hardware and software resources, the Grid paradigm and the network connections is thoroughly described in [23].

The ATLAS TDAQ system contains all the elements needed to both filter the events and to sustain the data flow from the detector to mass storage: dedicated hardware, commodity hardware (industrial PCs [19]), network resources, software packages, algorithms. The level-wise separation of the system is done depending on which type of on-line analysis/filtering an event is passed through. The higher the Level, the higher the complexity of the physics algorithms and the lower the rate of events flowing through that Level, as suggested in Figure 1.5. The rates of events are expressed as maximum values and are used as design criteria for the entire ATLAS Detector [24].

The online filtering of events is accomplished by three trigger levels[5]. Next we will summarize the design requirements for each of them but also the real measurements taken from the operational phase of the ATLAS detector:

- Level-1 - is built using dedicated hardware for filtering and buffering. By design it receives a 40 MHz rate of events and delivers 100 KHz to the next level. In the operational phase Level-1 received 40MHz and delivered a 60-65 KHz output rate.
- Level-2 - is built using commodity PCs for more complex algorithms which use partial event data. By design it receives 100 KHz from Level-1 and delivers 3.3 KHz to the next level. In the operational phase it received 60-65 KHz and delivered a 5-6 KHz output rate.
- Level-3 - is built using commodity PCs for the most complex algorithms which use full event data. By design it receives 3.3 KHz from Level-2 and delivers 300 Hz to mass storage. In the operational phase it received 5-6 KHz and delivered a 500-600 Hz output rate.

1.3 Performance in TDAQ

Although data rates between filtering levels have changed with respect to the design, most of the ATLAS TDAQ logic and interactions required for the selection process remain as described in [24]. In order to efficiently perform its job while collisions take place a filtering level requires a continuous flow of data from the upstream level and from other elements in the system, on request.

A thorough analysis of these aspects will be done later in the thesis, at this moment we only want to emphasize that TDAQ is by design a highly coupled and inter-dependent system. This characteristic is amplified in the higher levels (2 and 3) by sharing the same infrastructure: the interconnecting network, the processing power and/or memory. In such a system a design flaw can wholly disrupt it either as a complete failure or as a decrease of event processing rates in all parts of the system. A complex monitoring system has been put in place surveying the detector specific hardware, the computer farm, the network, the algorithms and many more. This is done

[5]The name *trigger* comes from the fact that a certain Level is activated - *triggered* - only by an event who has reached it and requires processing

for the purpose of understanding any potential disruption or for giving confidence that the system performs as designed.

At each level the TDAQ system's performance criteria are related to the rates of delivering decisions on events to the next element in the TDAQ chain. These rates depend on how fast and reliable individual event data analysis is performed. Analyzing an event involves: obtaining the required event data, processing it, potentially executing these two steps multiple times and finally forwarding the result. This process is characterized by a cyclic behavior and will basically produce, by its repetitive nature, the rates discussed above.

In the context of the high operational costs (mentioned in 1.1.3) for running the LHC and the ATLAS experiment, the cost per detected interesting event is also substantial. It clearly becomes a major factor in constraining the efficiency of the TDAQ system which is directly proportional to the rate of interesting events delivered by it. Furthermore, given the low probability of an event to be the one which, after offline analysis, is the evidence of new discoveries, the TDAQ system has to minimize the risk of missing one such event due to bad performance: losing packets in the network, discarding events due to buffer over-flow, hardware failures etc. All of these are related to the aforementioned reliability and speed of executing the cycles needed for processing events.

1.4 Contributions and outline

In this section we identify the main contributions of the current work in the effort of understanding the performance of communicating systems in general, the ATLAS TDAQ in particular. We will then guide the reader through the thesis by presenting its outline.

1.4.1 Thesis contributions

The very high and complex demand placed on the TDAQ system requires a deep understanding of the *dependencies* between its components. A lot of effort was invested in establishing the required performance in terms of event rates and data bandwidth e.g. in [11, 56, 24]. We are doing an analysis of the TDAQ system focused on the flow control and back-pressure mechanisms which act as strong couplings between sub-systems, e.g. filtering levels. We identify a compelling aspect in the system (the event buffers) which will be a consideration point for the rest of the work. The Level-2 trigger was mathematically modeled with respect to the aspect identified above because Level-2 is responsible for clearing a large portion of these buffers. This was done by applying a queuing theory model called M/G/c/c [1] which gave us an indication on the allowed time for Level-2 analysis.

The Level-2 analysis involves running algorithms and transferring data over the network, but the focus of this thesis is the data network component of the TDAQ system. The mathematical model is not able to separate the contribution of the network to the overall Level-2 analysis time. Due to this and other limitations we continue by introducing the *observational model* and the concept of *quality attenuation* as a method of characterizing network performance.

We also proposed a generic design flow approach in which the quality attenuation is able to assist the design stages. Although TDAQ reached the final phase, i.e. the operational one, this approach can be used to confirm that TDAQ is meeting its goals or to (re)define them. Furthermore, it can be considered as a design approach for the envisaged upgrade of the TDAQ system.

We then showed how the observational model works for data networks and how to apply it in the case of ATLAS TDAQ. Based on this model we identified three components for the quality attenuation in the data networks case, which can be treated as independent random variables. We continued by presenting a methodology of obtaining these components for networks and network devices alone based on statistical measurements. This methodology is cheap, non-intrusive and easy to implement as it uses commodity devices (PCs) and standard Linux tools, hence it can be used in a system in all the design stages, including the operational phase.

Finally we obtain a link between the load of the network and its performance by studying the variability of the quality attenuation. Based on this we are able to obtain the traffic pattern correlation factor as the relative deviation from the Poisson pattern and show that the traffic inside the TDAQ network is more correlated (bursty) than Poisson. We then make an extrapolation of the network quality attenuation for an increased load while keeping the same traffic pattern. We also indicate that from a networking point of view a constraint on the quality attenuation becomes a requirement for the applications placing traffic on that network, i.e. the ATLAS TDAQ software in our case.

1.4.2 Outline

Chapter 2 contains an analysis of the TDAQ's read-out component and its filtering levels emphasizing the buffering and flow control mechanisms implemented in all sections. The flow control is designed in such a way that it can propagate upstream up to the detector level. Based on this we will identify a main coupling factor in the TDAQ system, which has the potential of causing event discards in the detector specific read-out part, to be the buffers holding the event data.

Because Level-2 is the place where 97% of those buffers are cleared, in Chapter 3 we apply a mathematical model on the Level-2 in order to see what are is the probability of saturating it and thus block the process of freeing the mentioned buffers. We will show that the M/G/c/c queuing model is the closest to the real system and mathematically tractable in the same time. Although assumptions and simplifications will be made in order to fit this model, the results obtained by simulating it indicate the scale of the allowed Level-2 processing time. The sensitivity of this value to the number of Level-2 processors and with the probability of saturation will be obtained.

The contribution of the network delay to the total Level-2 processing time is not extractable from the mathematical model introduced in Chapter 3. The focus of this thesis is on the network component of the TDAQ system, therefore a different approach is needed. In Chapter 4 we will introduce the observational model which is applicable to any communicating system - including data networks. A central concept in this approach is the quality attenuation which captures both the delay and the loss in a network. For the specific case of data networks we identify here a decomposition of the quality attenuation in three independent random variables and show a method

of obtaining them.

Chapter 5 shows how this method is employed for the ATLAS TDAQ network - an Ethernet based network. We perform the component-wise decomposition of the quality attenuation for the Level-2 network and use a differential approach for obtaining the performance indicators for individual network devices. We then show how one of the components obtained for the Level-2 network is an indicator for the traffic pattern from the point of view of correlation factor and how we can use it to predict network behavior as a function of its load.

Chapter 6 concludes the thesis work emphasizing its contributing aspects, identifies future research steps and suggests further applications of the concepts and methodologies introduced in the thesis.

Chapter 2

Buffering and Flow-Control in ATLAS TDAQ

This chapter introduces the ATLAS TDAQ multilevel system, designed to filter and move the events from the detector's read out system towards the mass storage facilities. It emphasizes the techniques employed as back-pressure mechanisms, allowing the system to temporary accommodate rates larger than the design specifies.

2.1 Context

ATLAS, the biggest experiment hosted at CERN started to collect data from proton-proton collisions generated inside the Large Hadron Collider. Although many cutting-edge technical challenges have been overcome, there is still a need to understand and quantify the *performance* of its data acquisition system (TDAQ) in order to both optimize its current use and guide the next steps of improvements and upgrades.

Predictions of different metrics for performance are currently based on *"paper models"* ([24] - Appendix A) and *Discrete Event Simulation (DES)* models [32, 21]. The "paper model" is a static model of TDAQ, used for estimating:

- the processing requirements, expressed as the required number of computers in different parts of the system
- the rates of relevant data passing through the system, computed for two scenarios: low intensity and design intensity of the proton beams

The results obtained are given only as average values or as ratios between average and maximum (or minimum) values. The model assumes fixed values for the CPU load in computers (77%) and the network links load (60%). There is no knowledge of what happens when these values are reaching saturation, or what is the response to random fluctuations of the rates inside the system.

The other approach was to create a DES model of the ATLAS TDAQ system. The shortcoming of this approach is that the DES paradigm is capable of sampling only discrete loading points of the system, rendering it a descriptive (interpolative), but not predictive (extrapolative) approach. Using the model to extrapolate loading points close to saturation is risky in DES, as there is no knowledge of how close to congestion the resources of the system are. The philosophy of a DES approach is to create a computerized model of the system. It aims to "copy" the functionality and implementation details of the system, effort which can become overwhelming for a complex system like ATLAS TDAQ.

A system's saturation conditions are normally managed through the implementation of flow-control mechanisms whereas the fluctuations of rates inside the system are smoothed out through the use of buffers. These create a coupling between the elements of the system which is difficult to predict using a paper or a DES model. We will study these aspects in the current chapter keeping the values for rates as they are specified in the design document [24].

2.2 Level-1 Trigger

In ATLAS, collisions occur with a constant frequency of 40 MHz, controlled by a global clock signal. Each $25\ ns$, as result of these collisions, particles decay into sub-particles leaving an instantaneous splash of traces in the detector. As soon as the millions of "sensors" inside the ATLAS detector finish converting these traces into digital values, an event is ready to be processed by the first Trigger Level.

Level-1 Trigger's high level functionality is to reduce the rate of interesting events from 40 MHz to 100 KHz. This selection is based on *trigger menu*s, which are sets of conditions combined in a Boolean logic returning an ACCEPT/REJECT decision. Only a very small fraction of the event data is verified against the trigger menu (i.e. if it passes or not the thresholds specified in the conditions) for each event in order for a decision to be made [13].

During the latency period introduced by these operations, the continuous flow of events has to be buffered in a large number of *pipeline memories* (of the order of 10^8). As their cost is proportional to the depth of these memories, the delay allowed for Level-1 to make a decision of whether to *accept* or *reject* an event has been limited to $2.5\ \mu s$. This small latency is achievable by using dedicated hardware and purpose-built processors inside Level-1.

For each accepted event, the Level-1 Trigger is designed to:

1. send a *L1A* (Level-1 Accept) signal via the *TTC(Timing, Trigger and Control)* system [60] towards the front-end electronics and towards the *readout drivers (RODs)*, which causes the data from the pipeline memories to be pushed into the *derandomizers* [25]. These are buffers meant to fulfill multiple tasks:

    - to average out the high instantaneous data rate at the output of the pipeline memories in order to match the available input bandwidth of the next elements in the chain: the RODs .

- to temporally align the event fragments arriving with different delays from parts of the detector, due to the different distances they have to travel.

2. provide information, called *Region of Interest (RoI)*, to the next level to guide the data analysis performed there. The RoI information is sent to the RoI Builder (RoIB) [50] which packs this data into RoI records. The size of this type of record does not exceed 512 bytes. Knowing the maximum rate of Level-l 1 accepts (100KHz), the maximum cross sectional throughput generated by the RoI records becomes 51.2 MB/s.

Figure 2.1 illustrates the flow of data from the detector through all the buffers in the readout system, as well as the Level-1 roles and placement in the ATLAS TDAQ system. Sizes of the involved buffers, the number of the links between elements and the cross-sectional required bandwidth are also presented.

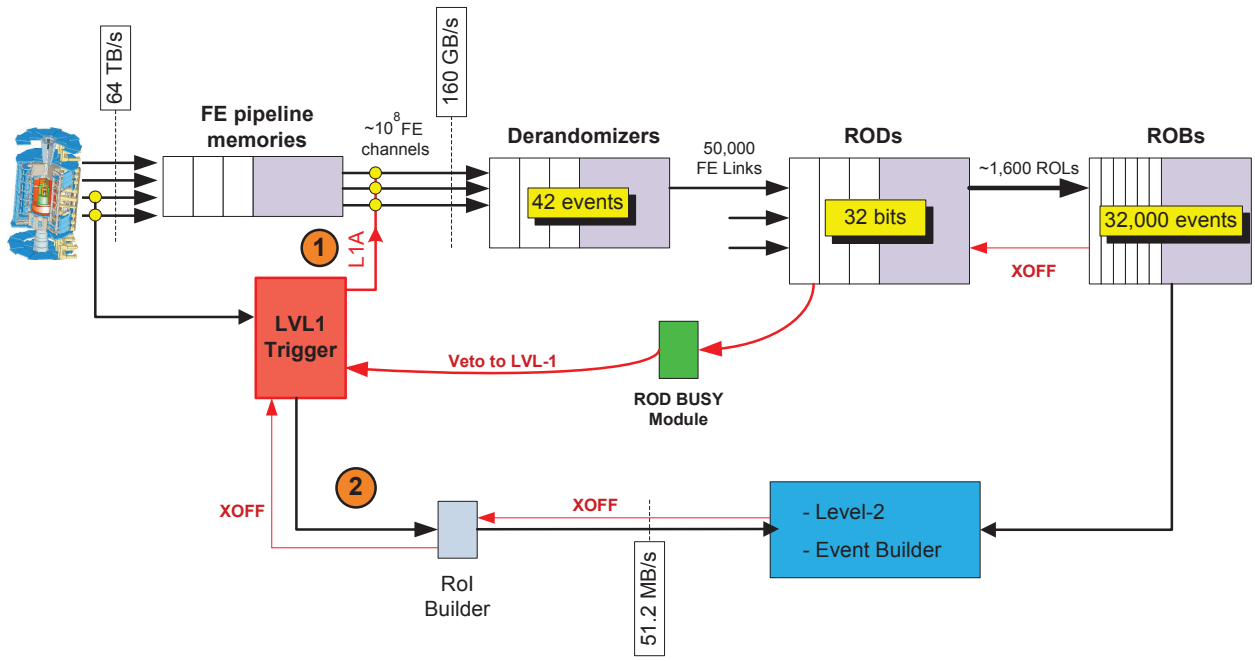

Figure 2.1: Buffering and Flow Control in the Level-1 Trigger and TDAQ readout system

The RODs act both as multiplexers (from $\sim 50,000$ Front End Links to $\sim 1,600$ Read Out Links) and as an interface between the sub-detectors and the TDAQ system - the output of the RODs are homogeneous, following the S-LINK specifications ([55]), although the RODs are implemented differently in the sub-detectors. RODs also perform data processing for the purpose of reducing the event size.

From the RODs data is transported over the Read-Out Links (ROLs) into the 1583 Read-Out Buffers (ROBs), which are the last dedicated pieces of hardware from the ATLAS detector. The RODs add control words indicating the begin and the end of the fragment and a check-sum to

Detection system	Sub-detector	No. of ROLs	No. of ROSs	Data per L1 accept (kB)
Inner Detector	Pixels	132	12	60
	SCT	90	8	110
	TRT	192	18	307
Calorimetry	LAr	762	68	576
	Tile	65	8	48
Muon Spectrometer	MDT	204	16	154
	CSC	16	2	10
LVL1	Calorimeter	54	7	28
	Muon RPC	32	4	12
	Muon TGC	24	2	6
	MUCTPI	1	1	0.1
	CTP	1	1	0.2
Forward detectors	BCM	3	1	1
	LUCID	1	1	1
	ALFA	2	1	1
	ZDC	4	1	1
Total		**1583**	151	**1239**

Table 2.1: Number of ROLs, ROS computers and event data

each fragment. These are discarded by the ROBs, after checking for errors. They are mounted on so-called ROBIN boards having the standard PCI connector, hosted in standard PCs called *Read-Out Systems* (ROSs) [22]. The numbers of ROLs and ROSs per sub-detector are summarized in Table 2.1.

2.2.1 Flow Control

The readout chain illustrated above contains many levels of buffering and information can be lost at any of these stages if buffers become saturated. In order to avoid uncontrolled loss of data, different flow control strategies are adopted:

- *systematic dead-time* in Level-1([13], Section 20.2). For four bunch crossings after each Level-1 Accept no other accept is issued. This limitation is imposed by the front-end systems hardware. The minimum interval between two Level-1 Accepts is thus 5 x 25ns = 125ns. This happens because on receipt of a Level-1 Accept most of the sub-detector front-end systems (the FE pipeline memories in Figure 2.1) are sending data for several consecutive bunch crossings. The number of these bunch crossings varies and the maximum value is five, characteristic to only one sub-detector.

- Level-1 limitation on the number of accepted events in a given interval. If this limit is

reached, *dead-time veto* is introduced until the required interval passes, inhibiting all possible Level-1 Accepts during that time. The purpose is to avoid derandomizers overflow. An example of usage is 8 events allowed in a $80\mu s$ time window, i.e. an instantaneous average of 1 event in a $10\mu s$ time window, equivalent to the 100KHz maximum design rate. The two values, number of accepted events and time interval, are parameters for the Level-1 Trigger.

- *Vetoing* a Level-1 decision with an external signal to prevent saturation of the RODs. An event that has passed the Level-1 criteria for being accepted will be vetoed as rejected if at least one of the RODs occupancy has exceeded a certain threshold.

The first two strategies have an impact on the Level-1 accept rates calculated in two ways: instantaneously and over the interval given as parameter for the dead-time veto. The limit for these numbers is given in Table 2.2.

Item	**Rate**	**Comments**
Proton-proton collisions	40 MHz	25 ns interval - given by the LHC global clock signal
Max. instantaneous Level-1 Accepts	8 MHz	125 ns interval - limited by the systematic dead-time
Level-1 Accepts	100 KHz	given by the dead-time veto - max. accepted events in a given interval

Table 2.2: Level-1 rates

The extreme case when all the events could be accepted by the Level-1 is presented in Figure 2.2. The purpose of this example is to better explain the effect of the first two strategies on the outcome of Level-1 Accepts in this particular case.

One can notice that even if Level-1 Accepts could be issued at each 25 ns, the systematic dead-time spaces out the possible accepted events 125 ns apart. Furthermore, even if 8 of these consecutive accepts could occur, the dead-time veto inhibits the rest of them until the $80\mu s$ interval elapses. The average rate of events calculated over the $80\mu s$ interval is 8/$80\mu s$ = 100 KHz.

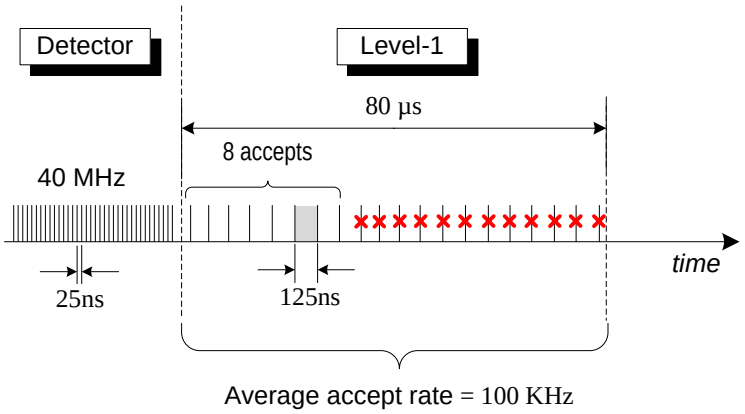

Figure 2.2: Flow Control effect for an extreme case in Level-1 Trigger

Furthermore, we will quantify the overall effect of the systematic dead-time by calculating the probability of missing interesting events in the Level-1 Trigger. At 40 MHz input rate in LVL1 and at 100 KHz acceptance rate, we can calculate the probability of an event being accepted:

$$P_{accept} = \frac{100KHz}{40MHz} = 2.5 \times 10^{-3} = 0.25\% \tag{2.1}$$

and then the probability of an event being rejected:

$$P_{reject} = 1 - 2.5 \times 10^{-3} = 99.75\% \tag{2.2}$$

Next, the probability that four consecutive events are rejected is:

$$P_{reject-4} = (1 - 2.5 \times 10^{-3})^4 = 99\% \tag{2.3}$$

Knowing that four events are deliberately missed, we want to calculate the probability that at least one of these four events is interesting for LVL1, i.e. the probability of accepting one of these four events:

$$P_{accept-1} = 1 - P_{reject-4} = 1\% \tag{2.4}$$

Following these calculations we can state that 1% of the interesting events for LVL1 will be missed due to the systematic dead-time mechanism.

All the mechanisms introduced above force the Level-1 to limit the amount of accepts issued by the first filtering level, meaning that they are limiting the Level-1 accept rate and perform arrival pattern shaping.

2.2.2 ROB Resource

The first two mechanisms can be considered pro-active measures because they are meant to prevent system overload. Only the third strategy is a reactive approach, being applied as response to a saturated system. When back-pressure is issued by the RODs the flow control mechanism is carried out by the use of veto signals to the Level-1. Vetoing doesn't take into account the physics measurements on events and the fact that a Level-1 Accept was issued for them. In this situation, the chances that good and rare events are discarded at this level increase.

From the RODs the next elements in the flow of event data are the ROBs, which are buffers 32,000 events deep. This number is a result of paging ROB's 64 Mbytes memory in pages large enough to accommodate the maximum event fragment in each ROB, which is 1.6 Kbytes, hence the page size was set to 2 Kbytes. Knowing the Level-1 rate we can calculate the time required to fill the ROBs: 320 ms for 100 KHz and 426 ms for 75 KHz Level-1 rate.

ROBs also have the possibility to send back-pressure signals to the RODs. The S-LINK protocol employs XON-XOFF signaling to prevent buffer overflow. Assertion of XOFF by the ROB results in the ROD to stop outputting data and to raise its BUSY signal, which may halt the Level-1 Trigger.

ROBs form a demarcation line between Level-1 and the following levels: Level-2 and Event Builder, introduced in the next sections. Based on their positioning in the system and the back-pressure they can generate upstream towards the RODs and consequently towards Level-1, they become a critical point in terms of buffering in ATLAS TDAQ. The availability of space in the ROBs is thus a *key resource* for the performance of the system. For this reason the time needed by the system to clear an event from the ROBs becomes the measure for the service time used in the models presented next.

2.3 Level-2 Trigger

Following a Level-1 accept an event is moved from the detector into the ROBs on the path described earlier and illustrated in Figure 2.1. Simultaneously, the Level-1 Trigger passes *Region-of-Interest* (RoI) information to the Level-2 on a dedicated path, using the RoI Builder. The RoI represents a set of coordinates in the detector's space based on the energy deposits in the calorimeters and muon track segments found.

The RoI instructs the Level-2 what fraction of event data to analyze for a decision to be made. As a result, the Level-2 will interrogate the ROBs for fetching only an estimated $\sim 2\%$ of the full event data when algorithms will run. However, the RoI guided Level-2 data requests are not uniformly distributed towards the readout systems. The ROSs most in demand are referred to as "hot". The amount of requested data differs also from event to event. In Figure 2.3 we present the data volume requested by Level-2 for two classes of events obtained from real collisions.

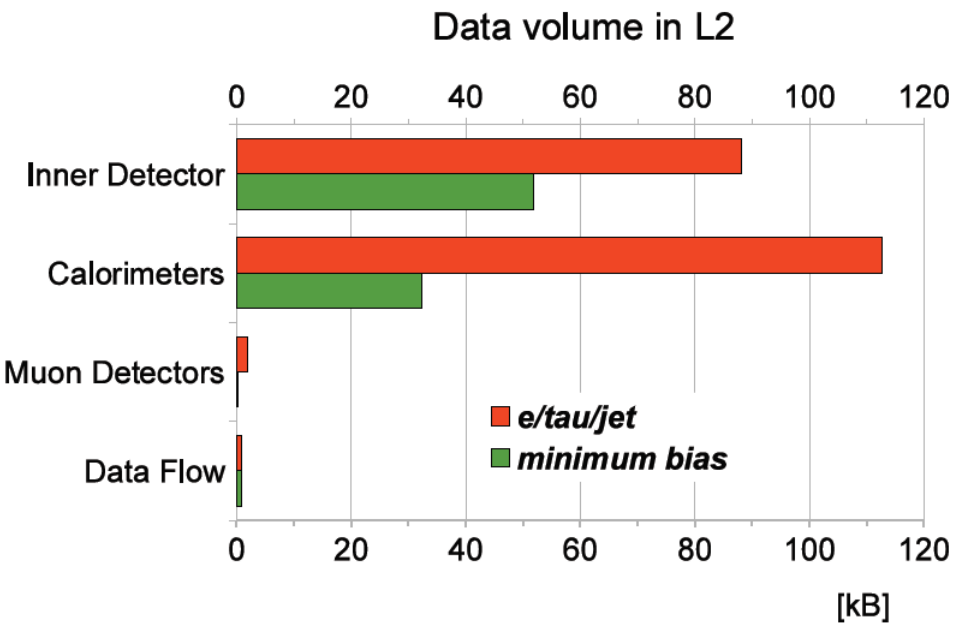

Figure 2.3: Data volume requested by Level-2

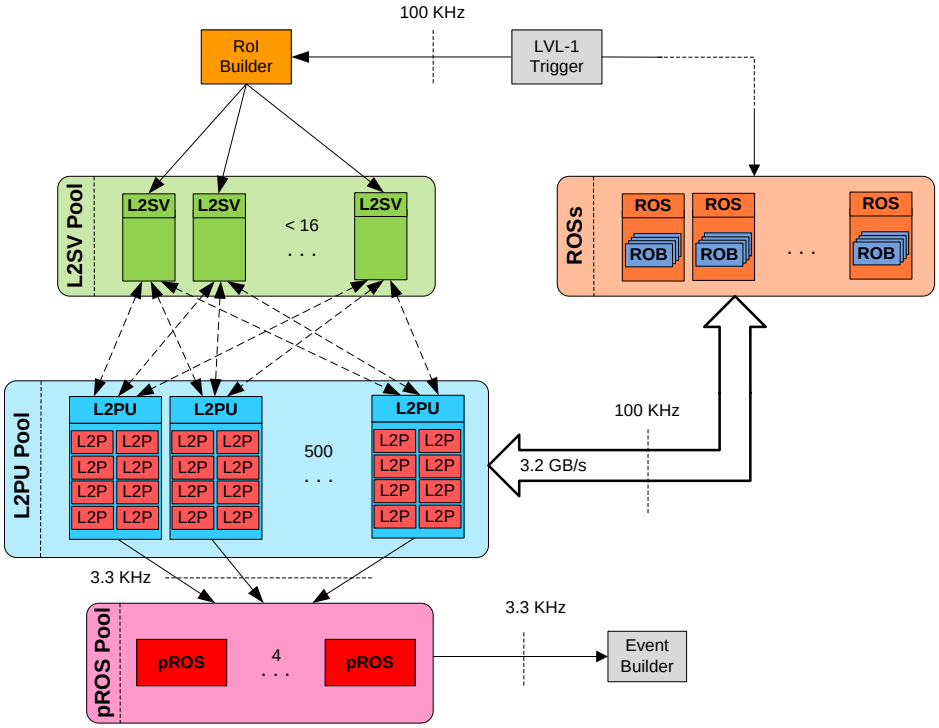

Figure 2.4: Level-2 Trigger components

The input rate of RoIs into the Level-2 is equal to the output rate of the Level-1 Trigger, i.e. 100 KHz by design. At this frequency, the RoI mechanism provides a substantial reduction in required carrying capacity, which is crucial knowing that a single event is 1.6 Mbytes. Using full event data at 100 KHz would imply a required bandwidth of 160 GB/s only in Level-2, while with the RoI mechanism reduces it to $\sim$3.2 GB/s.

The Level-2 Trigger is designed to make an on-line selection of events, reducing the rate from 100 KHz to 3 KHz. Figure 2.4 depicts the components which are used to implement this trigger level:

- up to 16 *Level-2 Supervisors* (L2SVs) computers, number limited by the output ports of the RoI Builder. The L2SVs receive RoI data for events from the RoI Builder in a round-robin manner. Based on a load balancing algorithm an L2SV dispatches its assigned event to one of the processing unit under its coordination and waits for an *accept* or a *reject* decision for it. Regardless of the decision obtained, the L2SV forwards it to the next sub-system, the DFM (2.4), in groups of 100.

  A unique subset of the L2PUs is assigned to each L2SV. The list is initially ordered such that successive entries are for L2PUs running on different nodes. The list is used to assign events to the L2PUs: for each event the RoI information received from the RoI Builder is sent to the first L2PU in the list, after which the entry is removed. When the L2SV receives the L2 decision information, the identifier of the L2PU that sent it is added to the end of the list, making a new processing slot available.

- approximately 500 *Level-2 Processing Units* (L2PUs) - 8 x CPU core computers[1]. An L2PU runs a Level-2 Process (L2P) on each of its CPU core[2], which gives us the total number of L2Ps to be 4000 [3]. An L2P receives RoI data corresponding to an event with the task of deciding whether this event is still *interesting* or not. Guided by the RoIs, the L2P starts an iterative process in which it interrogates data from only a small fraction of the total number of ROBs[4].

  The minimum data granularity is a ROB; requests usually span several ROS PCs and amounts in ranges from a few to 50 KB per event. An iteration consists of three steps, each involving a certain type of resource being used, as illustrated in Figure 2.5:

[1]The number of CPUs per computer increased with time since the design stage. The Level-2 cluster is heterogeneous comprising also 12 x CPU computers.

[2]Or hyper-thread, as newer computers have this option. For each processor core that is physically present, the operating system addresses two virtual or logical cores

[3]This number is around 7000-8000 at present due to the new generations of computers introduced in the system and due to their heterogeneity

[4]In the operational phase a different type of Level-2 request was added: *L2 Etmiss*. L2 requests only to the ROBs specified in the request, whereas *L2 Etmiss* requests are forwarded to all ROBs in the ROS PC. These data contain sums of energy deposits calculated in the calorimeter RODs. Only these energy sums, 6 words for each ROB, are passed to the L2PU sending an L2 Etmiss request and are used for the second-level missing energy trigger. This trigger is in use since the beginning of 2012 and runs at a rate of about 10 kHz.

1. L2PU sends a request for event data to a number of ROSs. Resource used: the Level-2 network (upstream).

   (a) ROSs reply with partial event data. Resource used: the Level-2 network (downstream).

   (b) L2PU runs algorithms on the received data and decides if it needs to go on the next iteration or the event is *rejected*. Resource used: L2PU CPU.

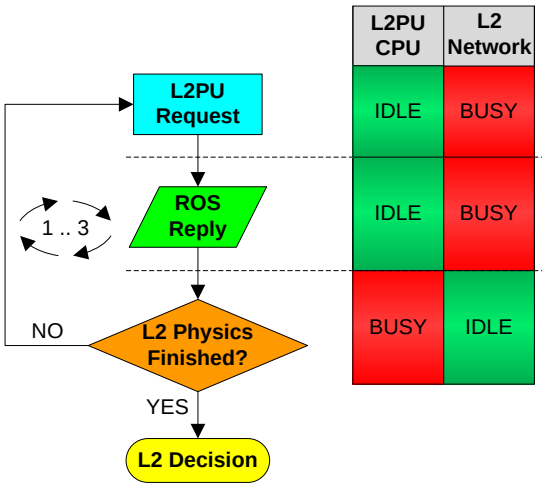

Figure 2.5: Level-2 Iterations

The maximum number of iterations is 3 (see [24] - Appendix A, Section 2.1) and an event is considered *accepted* if it passes all the required steps. Finally, the L2PU sends the result back to the L2SV, but if the decision is an accept, the result of the Level-2 analysis goes to the pseudo ROS.

- up to 4 *pseudo ROS* (pROS)[5] computers. For an accepted event, the pROS receives detailed record of the L2 analysis from the Level-2 processors and keeps it until the data is requested by the next level. At a latter stage this information will be used, the pROS being seen as a readout system component. The L2 Result contains the L1 Result and details of the L2 decision process in the form of a ROD fragment. The computational effort consumed at this level is not wasted in this way.

  In terms of performance, the pROS pool has to sustain both an input and output rate of 3.3 KHz of L2 analysis records, i.e. the Level-2 accept rate.

[5]currently this type of node is referred to as *L2 Result Handler (L2RH)* and their number is set to three - configurable

In terms of buffering the L2PU is allowed to queue a certain number of events, a programmable parameter and currently is set to 2. With respect to the flow-control the L2SV has a central role in handling it in the Level-2 Trigger. If an L2SV doesn't have an available L2PU to send a task to it issues back-pressure to the RoI Builder by asserting XOFF on the S-LINK connection. This can cause the RoI Builder to relay an XOFF further to the L1 system when it receives a fragment that can not be accepted due to the builder's output FIFO being full or bus transfers being blocked.

A key role of the Level-2 Trigger emerges from its main mission: to reduce the rate of events from 100KHz to 3.3KHz, i.e. a rejection factor[6] of 30. Approximately 97% of the events have to be discarded at this level which involves clearing them from the ROBs. Although the clearance signal is sent at the next level, Level-2 is the place where the decision is made.

The time spent in processing an event and eventually reaching the reject decision roughly gives the time an event has to stay in the ROBs. The distribution of the time an event stays in the ROBs is the emergent property of the entire operation of the Level-2 Trigger that has to be characterized. Taking into consideration the effect of the space in the ROBs on the Level-1 functionality, we identify this to be the main coupling factor of the Level-1 and Level-2 Triggers.

2.4 Event Builder and Level-3 Trigger

The last filtering level performed on-line, the Level-3 Trigger, is the first place in the system where analysis is applied on full event data, reason for which this level is also referred as the *Event Filter (EF)*. The component which constructs the full events and provides them to the Event Filter is called the *Event Builder (EB)* [34]. An illustration of what elements reside inside these sub-systems and the flow of data passing through them is given in Figure 2.6.

The Event Builder receives the LVL2 decisions and, based on the type of these decisions, it issues clearing of the events from the ROBs - in case of a *reject*, or it assembles the full event data - in case of an *accept*. The most resource consuming operation is the latter, due to the LVL2 accept rate which generates a total of $3.3KHz \times 1.6MB \backsimeq 5.3GB/s$[7] transfer rate from the Read-Out System to EB. These tasks are accomplished by two types of elements in the Event Builder:

- DFM - *Data Flow Manager* - receives the LVL2 decisions in blocks of 100. A rejected event is added by the DFM into a pool of events to be *cleared*. When this pool reaches a certain number of entries (100) or a preset time interval has passed, the DFM sends a multicast [35] message to all the ROSs and pROSs (2.3) informing them that the hosted ROBs (or buffers in case of a pROS) can clear the corresponding event fragments from their memory.

  For an accepted event the DFM chooses one SFI to which it assigns the job of collecting the full event data and from which it expects a confirmation when the assembly of data is

[6] The numbers obtained from a running system with real collision data are different from the design: the rate had to be reduced from 60-65 KHz to 5-6KHz, i.e. a rejection factor of 10-13

[7] during data taking a rate of ~6 KHz was observed, hence the data rate for assembling events was 9.6 GB/s

finished. The assembled events for which a confirmation has been received by the DFM enter in the same pool of events to be cleared. Estimations indicate that one DFM suffices to perform all the required tasks.

From the point of view of redundancy, there is a pool of 12 DFM computers available in the system, but the DFM application can be run on only a single computer at a given time. In case the active DFM fails, the ATLAS data taking software has an online recovery mechanism in order for a different computer to become the active DFM.

- SFI - Sub Farm Interface. When an event is assigned to it by the DFM, the SFI sends requests for data to all the ROSs and the pROS, in a sequential manner. Since the next elements in the chain could potentially request input from very many ROS sources there is the risk that all the responses would concurrently converge at the egress port of the network core, create congestion and possible data loss. To avoid this the SFI employs traffic shaping using a credit based flow control technique. There is a parameter called *the number of outstanding requests* representing the number of randomly chosen ROSs to which requests are sent in a batch by the SFI. Currently this value is set to 15. After all the chosen ROSs finish replying with data, the same operation is performed on a different set of ROSs, until all the ROSs send their data to the SFI.

  After all event fragments are received a confirmation is signaled to the DFM, called an *End of Event (EoE)* message. The assembled events are buffered inside the SFI for a short interval until they are requested by the Event Filter. The estimated number of SFIs required in the system is 100 and currently there are 90 nodes installed.

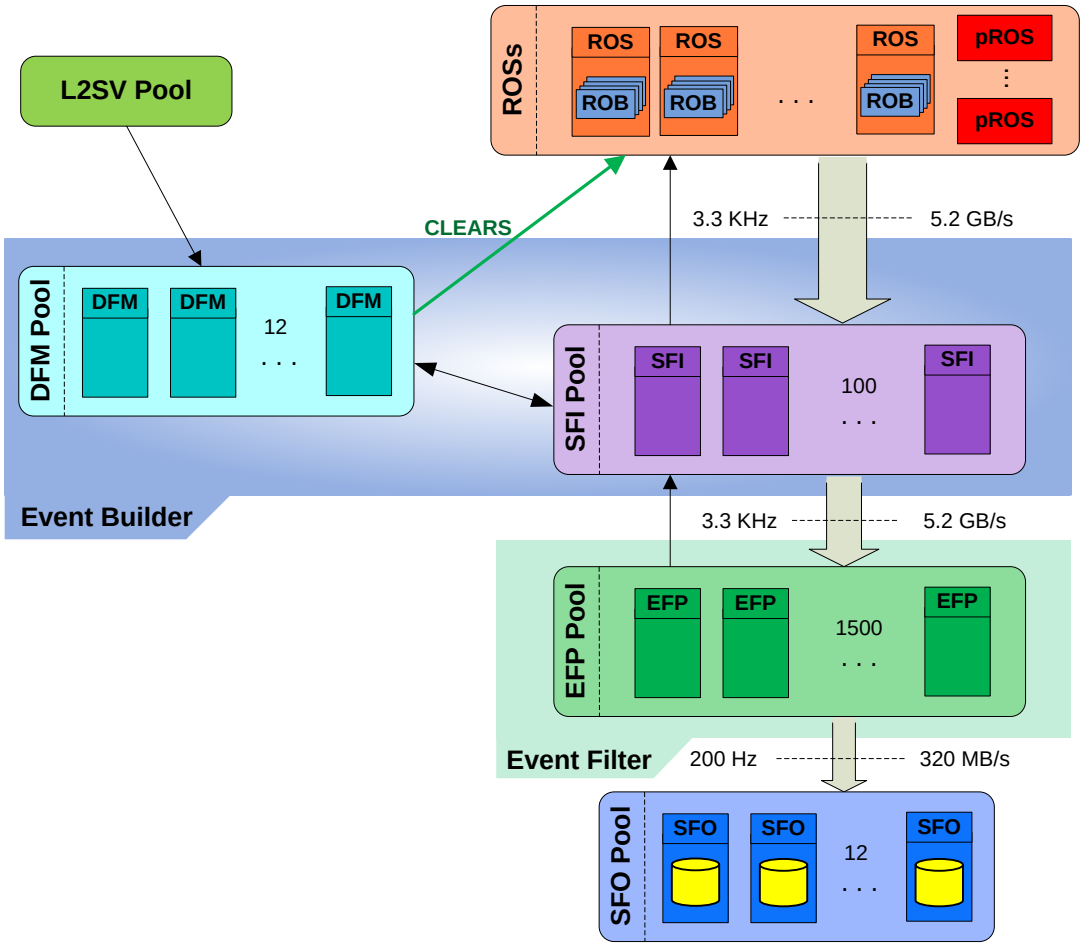

Figure 2.6: Event Builder and Event Filter

At this point, full events validated by the Level-2 Trigger are held inside SFIs, waiting for the last filtering level to collect them. The Level-3 Trigger, known as the *Event Filter (EF)* performs this selection step before the events are sent to the mass storage. Its purpose is to further reduce the amount of data written on disks, from a rate of events of 3.3 KHz, as it is received from LVL2, down to 200 Hz. In this case, the data written on storage is around 320 Mbytes/s[8].

The Event Filter is composed of ~800 EF nodes each running an *Event Filter Dataflow - EFD* process and a number of *Event Filter Processor Units (EFPUs)* equal to the number of CPU cores. The SFIs are balanced to serve an equal number of EF nodes. The EFD is a multi-threaded application and deals with data flow operations while the EFPUs deal with running the filtering algorithms. In this way the Event Filter ensures a decoupling between the event processing and data flow operations.

The total number of EFPUs is configurable and is around 6000-8000. After the Event Filter

[8]in data taking conditions at an observed rate of 600 Hz the data rate was 960 Mbytes/s

accepts an event its corresponding data is sent to 5 SFO (Sub-Farm Output) machines. An EFD acts as a client while the SFI and SFO act as servers because the EFD is either requesting data from the SFIs or requesting one of the SFOs to accept event data.

2.4.1 Flow Control

The flow control mechanisms at this level is controlled by the DFM which may send back-pressure messages to the L2SV in case the Event Building cannot proceed. This may happen if all the SFIs are busy or the Event Filter and/or the data recording cannot keep up with the rate. An SFI can be busy because either the maximum number of outstanding assignments by the DFM (typically 10) has been reached or the number of events not yet dispatched to the EF is at the upper limit (typically 100).

An EFD is able to slow down the event request rate proportionally to an internal SharedHeap memory occupancy and hence provides an upstream back-pressure mechanism, i.e. towards the SFI. The increase in SharedHeap occupancy can be caused by two things: EFPU processing is not coping with the incoming rate and/or the SFOs are not receiving events fast enough. The latter case is due to either the network bandwidth or writing to disk. The SFOs are equipped with a total storage space of 30 Tbytes, designed to store events for maximum 24 hours in case of lost communication with the mass storage, but also to average out the bursts of events towards the mass storage.

2.5 ATLAS TDAQ Data Network

The task of the ATLAS TDAQ system ends here, the events will be available from now on for offline analysis and remote distribution to a much wider "audience" all around the globe. In order to provide a continuous flow of interesting events during an entire run - lasting up to 20 hours - the TDAQ system has to sustain the designed rates of events sent between its elements. In this context, one of the key components of the TDAQ system is the *network* which interconnects the $\backsim$3000 computers, more precisely the *data network* carrying all the information related to event data and physics algorithms.

The ATLAS TDAQ holistic network [57] is a multipurpose, multilevel and hierarchical network, incorporating two independent networks:

- the *Control Network* - responsible with carrying traffic related to: network booted operating systems, file servers access, maintenance operations, monitoring information. This network can also be used for in-band management of the network devices.
- the *Data Network* - supports all the communication required by the exchange of messages and the transport of data between: Read-Out System, Level-2, Event Builder, Level-3, as described above.

These networks are physically separated by the use of different network devices and links. Interference between them is thus avoided, allowing for an independent study and performance analysis. Due to the distinctive type of information transported - data produced by proton-proton collisions - the Data Network in the ATLAS TDAQ system is required to deliver real-time, lossless and large bandwidth traffic and consequently falls under the scope of this work.

The Data Network carries all the messages between the elements belonging to the readout system, Level-2, Event Builder and Level-3 (Event Filter) over a high speed Ethernet network [17] build out of commodity devices. There are incontestable motivations for why Ethernet was chosen as standard over the other options, e.g. ATM: exponential increase of the supported speeds, multi-vendor, pervasive, low price per port, support for Virtual LANs [56, 17, 14]. In addition, at the time the standards were analyzed, Ethernet was introducing Gigabit over standard CAT-5 copper cables and TenGigabit over optical fiber, which made Ethernet suitable to offer the bandwidth required by ATLAS TDAQ.

Because of different requirements in terms of latency and loss, the Data Network is physically separated into:

- the *DataFlow Network* - the network interconnecting the Read-Out System, the Level-2 and the Event Builder. It transfers a large amount of data with a latency limited by the time the events are allowed to stay inside the ROBs.
- the *BackEnd Network* - the network which distributes the fully built events from the SFIs (Event Builder) to the EFs (Level-3) and finally to the SFOs. Given the low rate of events at this level and the localized data for an event on a single PC, the performance requirements of this network are less demanding.

The current version of the TDAQ data network involves two classes of network devices (switches) mapped on two hierarchical layers: the core network belonging to the core layer, and the concentration network belonging to the aggregation layer. In Figure 2.7 the aggregation layer is represented by the ROS Concentrators and the Level-2 and Level-3 Concentrators - while the core network is made up of four chassis switch-routers. The core of the DataFlow network consists of two redundant chassis switches placed on different VLANs, hence different IP subnets as depicted in Figure 2.7. In case of one chassis switch failure this architecture offers full connectivity for the Read-Out System on the redundant links but only half of the connectivity for the Level-2 processors (i.e. odd or even racks of computers).

In this context, the L2PU and EF computers are denoted using the same name: *Exchangeable Processing Units* (XPUs) because of their capability of being designated to run either Level-2 or Level-3 software. By simply using addresses from different classes and multiple VLANs on the same physical connection, the applications - regardless of their type - are able to communicate properly with the rest of the system. This explains the overlapping area between the BackEnd and the DataFlow networks consisting of the Level-2 and Level-3 aggregation switches in 2.7.

The network described and depicted above introduces an inevitable quantity of delay and loss in the ATLAS TDAQ system. They represent a part of the amount of the loss and delay a certain

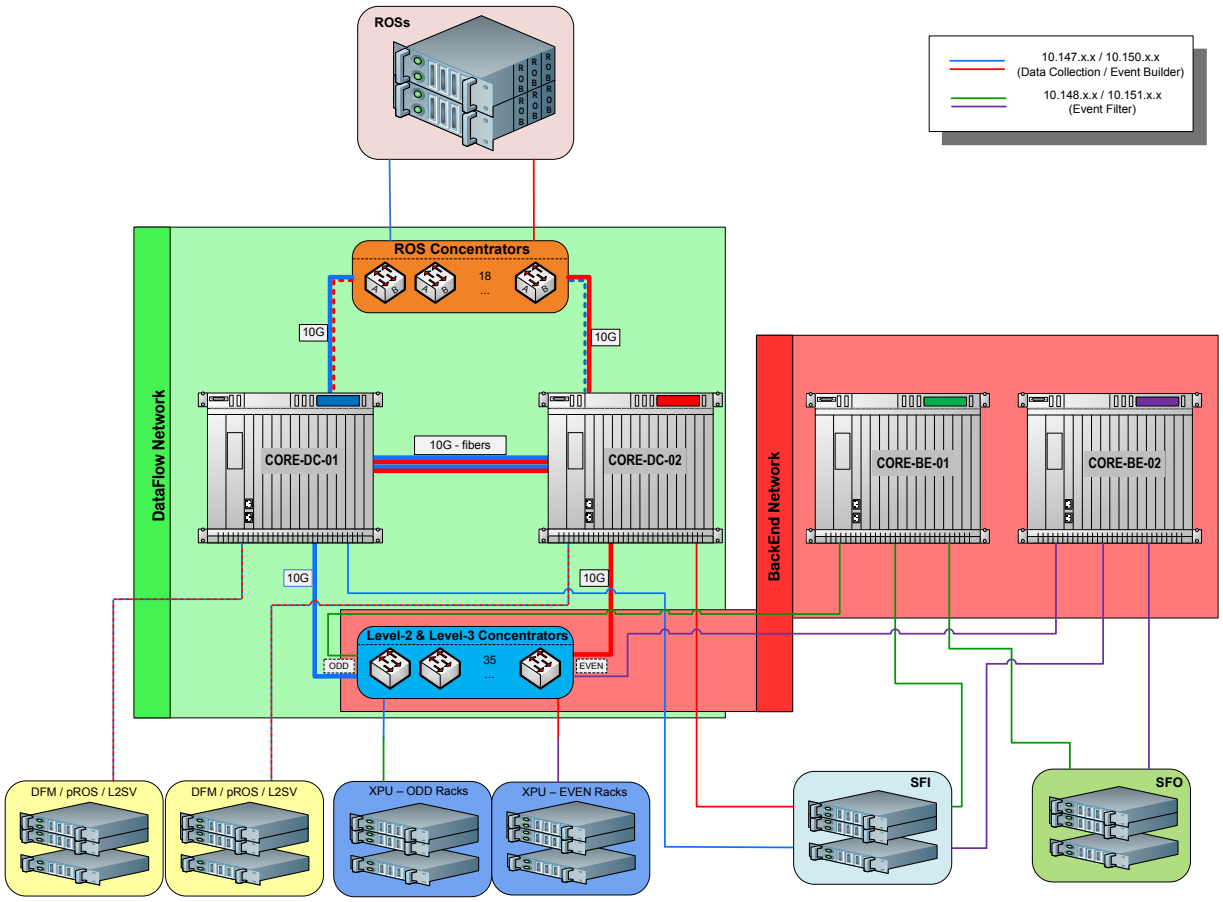

Figure 2.7: The ATLAS TDAQ hierarchical network

trigger level is allowed to introduce. As an example the ATLAS TDAQ Level-2 Trigger intensively uses the network by dispatching events to L2PUs, requesting and receiving data from the Read-Out System in multiple iterations and finally sending back the decisions. The potential increased delay introduced by the network is reflected on the increased LVL2 total processing time. The loss of packets can cause incomplete events to occur and for these a forced decision is made after a certain time-out, increasing too the processing time. Adding to this the limited latency allowed in the Level-2 Trigger it becomes obvious why the emergent network performance has to be well understood and pro-actively managed.

2.6 Performance coupling

Under different data taking conditions and configurations, the rates of events may vary abruptly. The implemented buffering allows variations of rates for short period of time (seconds) compared to the long run time (hours), hence the system has to have a mechanism to self regulate. This is achieved through the complex flow control mechanisms implemented in all the data flow

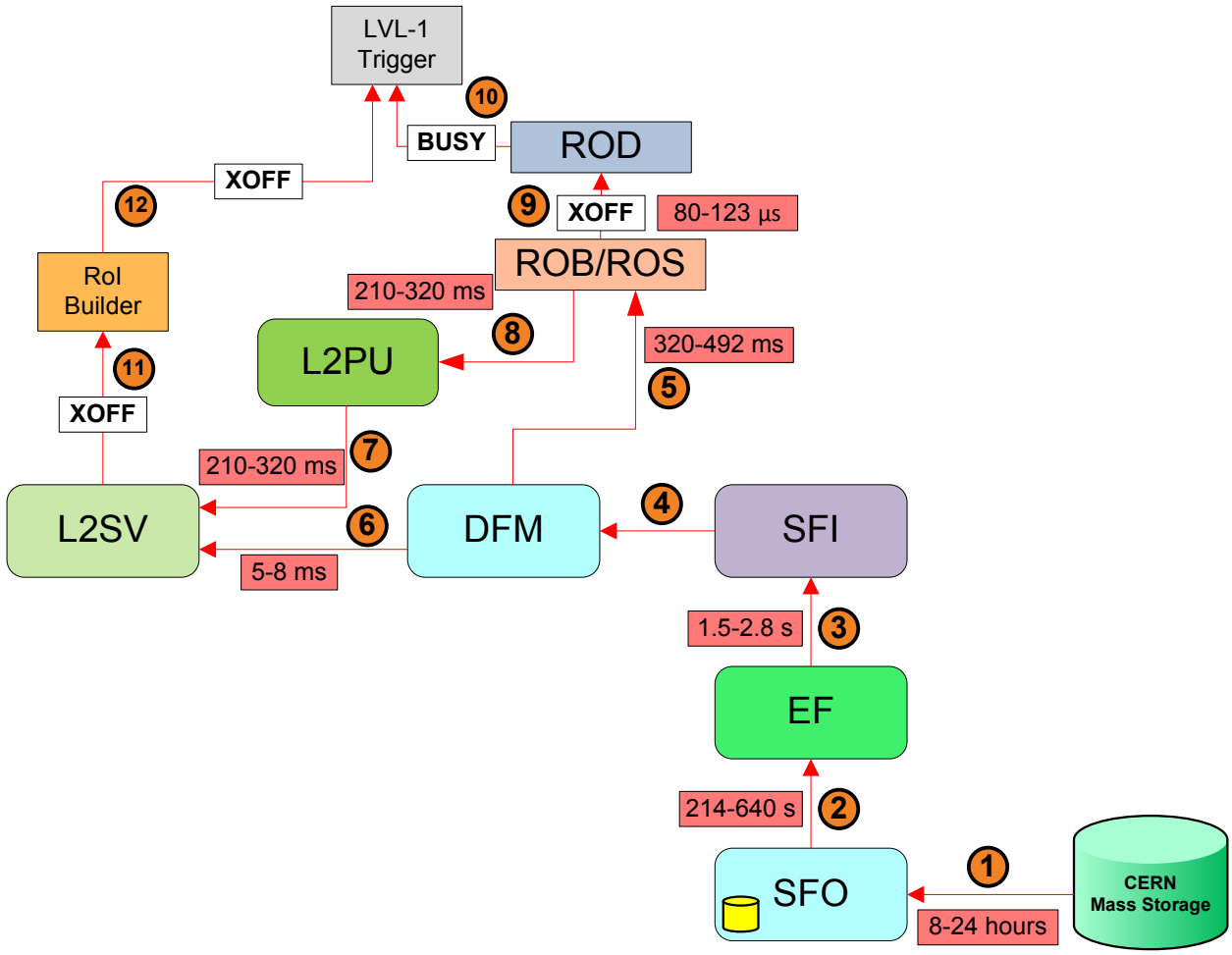

Figure 2.8: Flow control points in ATLAS TDAQ

components, pinpointed in Figure 2.8[9].

When we say flow control we include both direct and indirect effects on the upstream components. We classified them indicating which of the points in diagram 2.8 belongs to which categories:

- flow control signals: XOFF on the S-LINK connections or dedicated BUSY signals - hardware level - {9, 10, 11, 12} in the diagram
- erroneous or busy connection links - operating system level - {1, 2, 3, 4, 5, 6, 7, 8} in the diagram
- I/O (disk) busy status - operating system level - {1, 2} in the diagram

[9]As the flow-control is normally an upstream oriented mechanism we intuitively numbered the points in the reverse direction

- back-pressure from a job by not clearing a buffer fast enough - application level - {5, 9, 10} in the diagram

- back-pressure from a job by not returning a confirmation of job completion or a reply fast enough - application level - {2, 3, 4, 7, 8} in the diagram

- back-pressure from a job by not requesting data fast enough - application level - {2, 3, 4} on the diagram

The rates of events flowing between the elements depicted in Figure 2.8 depend on the performance of each individual segment but also on the performance of the rest of the system in two directions. The downstream one is the normal flow of data which represents the workload fed to that particular segment: e.g. the rate Level-1 accepted events gives the amount of work served to the next level and so on. The upstream one manifests under the form of inhibiting factors who can diminish the amount of work performed within a particular segment: e.g. if Level-2 becomes saturated the events will be queued inside it until the Level-1 is eventually affected by the RoIB who is issuing XOFF messages.

We will detail each point in Figure 2.8. We will also mention, where possible, the time allowed by the system - typically by the upstream elements - for the causing element to completely malfunction without affecting the flow of events[10]:

1. This back-pressure occurs if there are link failures between the SFOs and CERN mass storage system or if the latter exhibits application limitations (e.g. writing to disk). The immediate effect is that the SFOs are forced to store events locally. The flow of events is disturbed when the SFOs local disk space (detailed in Section 2.4) is exhausted, meaning an allowed time of 8-24 hours.

2. This back-pressure manifests itself if the SFOs are not requesting or receiving events fast enough. This results in an increase of the SharedHeap utilization on the EF nodes which automatically causes the Event Filter to decrease the rate of event processing rate. The size of the SharedHeap is 256 MB, the event size is 1.6 MB, hence at a rate of 200-600Hz events not sent to SFOs the allowed time for completely malfunctioning SFOs is 214-640 seconds.

3. This back-pressure occurs if the EFDs (introduced in Section 2.4) are not requesting, receiving or confirming to the SFIs the completed event data transfer fast enough. This causes the SFIs not to clear the full events stored in local buffers. At a rate of 3.3-6 KHz Level-2 accepted events, 1.6 MB event size and 90 SFIs the amount of data entering a SFI is 57-106 MB/s. The time allowed here depends on the configurable number of events not yet dispatched to the EF, currently 100 per SFI, hence the time until all SFIs will get full and won't be able to accept any assignments is 1.5-2.8 seconds.

[10] considering two scenarios for data rates: design and maximum (as observed during the runs)

4. In case the SFIs are not able to accept any work assigned by the DFM for any reason, this back-pressure mechanism is not affecting DFM's functionality, but it triggers 5 and 6 in Figure 2.8.

5. This back-pressure manifests itself by filling the ROBs. The reason is that the ROB CLEAR messages do not reach the ROSs due to communication problem or because they are not sent by the DFM. The CLEAR messages are sent in packets reaching all the ROSs. A single faulty link can cause a global problem because a single full ROB can trigger back-pressure mechanism 9 in Figure 2.8. The maximum time allowed for a DFM to not communicate CLEARs to ROSs is the time required to fill the empty ROBs, which is 320-492 ms.

6. The back-pressure towards the L2SVs occurs if the Event Building cannot proceed, i.e. there is no SFI available. The time allowed here is the time it takes to fill the 5 L2SVs buffers for decisions to be sent to DFMs, typically 100 large. This buffers are filled at Level-1 rate, hence the allowed time is 5-8 ms.

7. The back-pressure from the L2PUs towards the L2SVs occurs when an L2PU is not processing events fast enough and consequently its queue of 2 events becomes full or there is a fault on the link between the two elements. In this case, the particular L2PU will no longer be considered an available resource by the L2SV. The time allowed in this back-pressure scenario is the time to fill all the L2PUs queues and processing room (2+1 events for ~7000 L2PUs) at the Level-1 accept rate, i.e. 210-320 ms.

8. If the ROSs do not reply with ROB data to the L2PUs fast enough the Level-2 processing is queued on the latter. Given the small queue on the L2PU, which is 2 events, the time allowed for a ROS to answer to it with data is the time to fill a L2PU queue and processing room (2+1 events) at the Level-1 accept rate divided by the number of L2PUs, which results in the same value as for point 7 in Figure 2.8: 210-320 ms.

9. This back-pressure is implemented in the S-LINK protocol as an XOFF signal and occurs when a ROB cannot receive additional data. The ROD DSP has an input buffer where it is possible to store up to 16 events. The BUSY (10 in Figure 2.8) is raised towards the Level-1 Trigger when the input buffer has more than 8 events and is freed when it has less. The maximum time allowed by the system is the time required to reach the ROD threshold (8 events) at Level-1 accept rate: 80-123 μs.

10. The BUSY signal immediately vetoes any Level-1 accept (see subsection 2.2.1).

11. This back-pressure occurs when the L2SV doesn't have an L2PU available for event processing and it sends XOFF to the RoI Builder on the S-LINK connection.

12. The RoIB will relay an XOFF to the L1 system when it receives a fragment that can not be accepted due to the builder's output FIFO being full or bus transfers being blocked. The allowed times for both 11 and 12 in Figure 2.8 are extremely small due to the limited amount of space in the output FIFO for the S-LINKs (512 words, 32 bits each) [55].

From the back-pressure mechanisms detailed above we draw the conclusion that the ATLAS TDAQ is a tightly coupled system in which, in a long run scenario, performance in one part has the potential of influencing the whole system. Furthermore, the TDAQ implementation allows for tuning the rates of events in all three trigger levels by changing the filtering thresholds. This leads to the necessity of quantifying the effect on the rest of the system especially from the perspective of bottlenecks.

The back-pressure directions converge towards the Level-1 Trigger by means of the ROBs and the RoI Builder, both directly connected to the Level-2 filter. Because of this and due to the stringent delay requirements in terms of Level-2 processing, of the order of hundreds of milliseconds, this thesis will focus on the Level-2 subsystem, mainly on its data network performance analysis.

Chapter 3

Mathematical model for Level-2

This chapter presents a mathematical model applied to the Level-2 trigger system. Due to its complexity and essential role in clearing a large portion of the events buffers, this level is of interest to us from the perspective of its performance. We will present the results of the mathematical model from two perspectives: their usefulness and their limitation with respect to the data network, thus justifying the need for a different approach.

3.1 Context

An undesirable situation for ATLAS TDAQ is the incapability of processing events although they are successfully generated by the detector. This can happen in a trivial situation: the system is saturated and the rate of processing events is lower than their arrival rate. In this case, the feedback mechanisms immediately signal the Level-1 to inhibit accepted events, increasing thus the rate of discarded events and the chances that interesting events are dropped. This is accomplished by employing a veto mechanism driven by the flow control signals (see subsection 2.2.1).

3.1.1 Resources

The Level-2 trigger is an extremely important level because it is the point where $\sim$97% of the events are filtered out and will consequently be cleared from the ROBs. The functionality of Level-2 has to be aimed at minimizing the probability of ROBs filling up (for a given Level-1 acceptance rate) and thus avoiding the back-pressure mechanism mentioned above. Hence, the time needed for Level-2 to process events has to be properly controlled, as it is directly linked with the clearing time from the ROBs.

Figure 3.1 shows that the Level-2 trigger functionality follows a probabilistic tree until a decision is made. We are concerned by two types of resources being used: L2PU CPU and L2 Network, both accounting for a certain time budget. An individual Level-2 process is essentially a serial

process: when the BUSY periods are added together, they form the total Level-2 processing time. One can notice that out of these, the network BUSY periods have the potential of being significant, as they present in two out of three states in each Level-2 iteration.

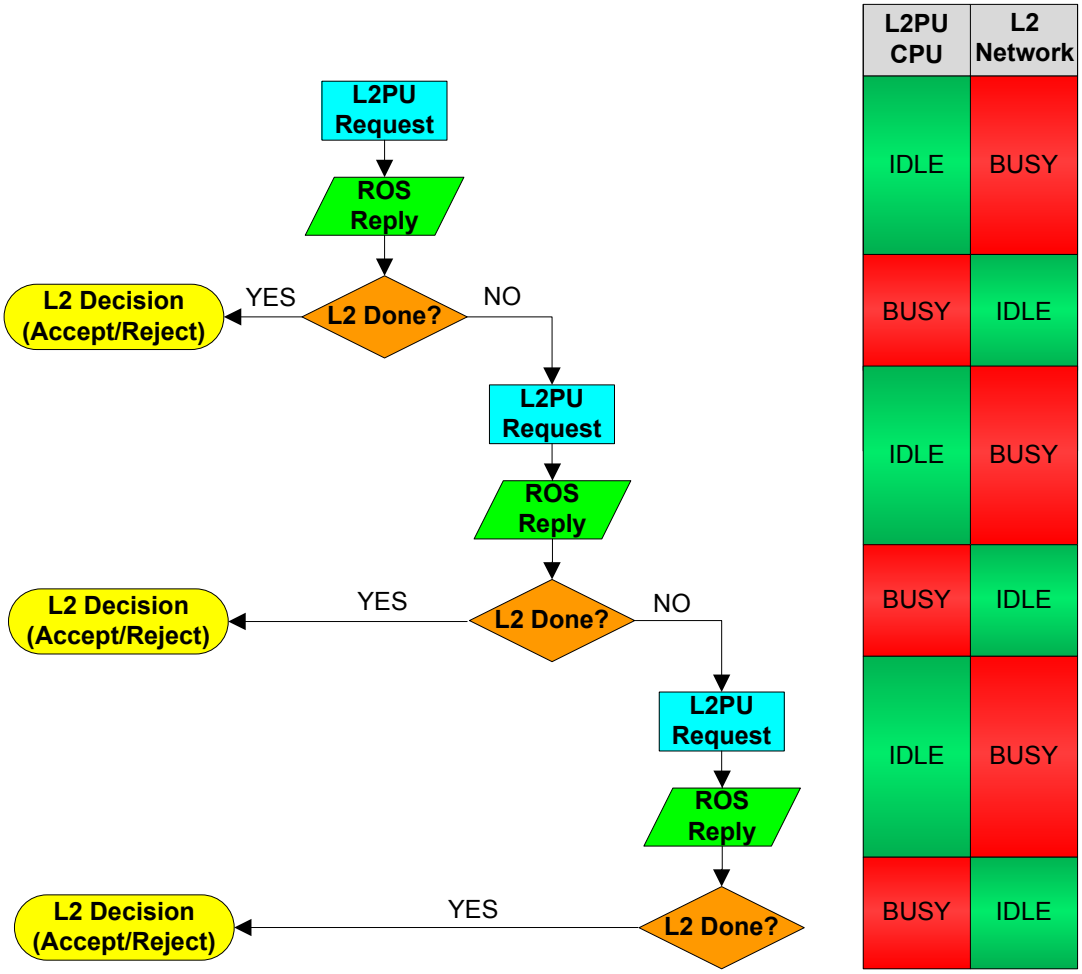

Figure 3.1: Level-2 Probabilistic tree and resource consumption

3.2 Approach

The goal of this chapter is to introduce a mathematical approach for quantifying the average event processing time in Level-2, as a function of the number of L2 processors available and the probability of the system being full. The approach is based on a **Queuing Theory model**, chosen to properly fit the architecture and the functionality of the Level-2 subsystem.

The corresponding mathematical terms for the mentioned parameters which we will use from now on are:

- the *service time* for the event processing time in Level-2
- the *server* for the L2 processor
- the *blocking probability* for the probability of system being full

Knowing that the service time in Level-2 comprises the *transmission delay* and the *computing delay*, the mathematical approach we introduce returns important indications:

- the maximum bound of the accumulated network delay, i.e. the aggregated network delay has to be less than the indicated value of the service time.
- the sensitivity of the service time to the number of processors, indicating for example the maximum allowed number of computers that can go offline without exceeding a certain value for the service time for an imposed blocking probability.
- the sensitivity of the blocking probability to the service time. The formulas used later in the analytical model state that zero probability of system being full requires an infinite number of processors or zero service time, unfeasible in both cases.

3.3 Assumptions

In order to choose the correct model for the Level-2 but also a mathematically tractable one, the queuing model applied on the Level-2 trigger requires a good knowledge of the system and some assumptions to be made.

The input for a Queuing Theory [2][1] consists of: *arrival pattern* of customers into the queue, *service time distribution* (the time needed to serve a customer), *number of servers* and *queuing discipline* (e.g. first come first served, last come first served, random order).

The working assumptions for the Level-2 system refer to the type of these characteristics:

- the distribution of the inter-arrival time is exponential
- the service time distribution is general
- the number of servers equals the number of Level-2 processes available for handling events. The important assumption here is that the servers have no queue in front of them, i.e. if all the servers are busy, the next event which arrives will be dropped.

Furthermore, in order for the model to be analytically tractable, we abstract away from the presence and role of the Level-2 Supervisors. In the real system the L2SVs receive Level-1 events in a round robin manner from the RoIB (see 2.3) and an L2SV dispatches its assigned events to only a pre-allocated pool of Level-2 Processors. The Level-2 system is thus made up of a number of parallel queuing systems, much more difficult to be captured in a mathematical model. We thus need to simplify the model and consider that the events are dispatched directly to one of the available Level-2 Processors.

3.3.1 Exponential distribution in Level-2

The first assumption concerns the events arrival distribution into the Level-2. We will approximate the inter-arrival time distribution for the events entering the Level-2 with an *exponential distribution*[1]. The logical steps we will follow to justify this approximation are:

1. identify the source of events entering Level-2 as being the Level-1 accepts
2. show that the Level-1 process is a sequence of Bernoulli trials characterized by the probability p_{accept} of accepting an event
3. explain that the probability of having an accepted event after *n* consecutive trials follows a *geometric distribution*, characterized by *n* and p_{accept}
4. explain that for large $n \gg 1$ and small $p_{accept} \ll 1$ the geometric distribution becomes its continuous analogue: *the exponential distribution*
5. show that the conditions above are met in Level-1, hence the distribution of Level-1 accepts is exponential in the limit.

Step 1

As explained in sections 2.2 and 2.3, events accepted by the Level-1 trigger are sent into the 1600 ROBs over dedicated paths. Simultaneously, on a separate path, Level-1 instructs the RoI builder (RoIB) to construct the RoI records for the accepted events, which are immediately sent to the Level-2. What Level-2 receives from Level-1 and identifies as an event[2] is the RoI record only, hence the arrival distribution we are interested in is the arrival pattern of RoI records.

The RoIB is a dedicated hardware and requires $2\,\mu s$ to build the RoI record from the moment when all the event fragments are available. The distribution of events entering Level-2 is not changed since they left Level-1, because they are only delayed with a fixed interval. Therefore, the distribution of events entering Level-2 is the distribution of accepted events produced by the Level-1 trigger.

Step 2

On average, from a constant rate of 40 MHz events generated by the ATLAS detector, only 100 KHz are accepted by Level-1. The output of this filtering level is thus a sequences of Bernoulli choices occurring at 40 MHz, with two probabilities (see Figure 3.2):

- $p_{accept} = \frac{100KHz}{40MHz} = 0.25\% = p$

[1] Also known as "negative exponential distribution"

[2] We will use the term "event" for a RoI record because it contains part of the raw event data and the Level-1 identifier for that event

- $p_{reject} = 1 - p_{accept} = 99.75\% = q$

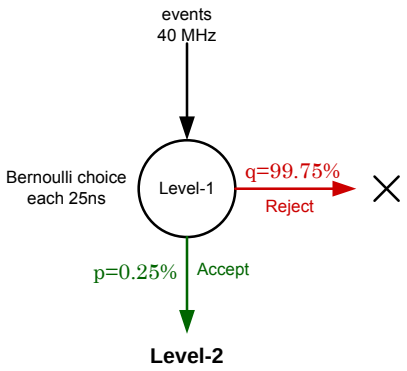

Figure 3.2: Bernoulli choice in Level-1

Step 3

Having a sequence of Bernoulli choices, the probability that the first success (i.e. accept in our case) occurs after *n* failed trials (i.e. rejects in our case) is:

$$(1-p)^n p \tag{3.1}$$

where *p* is the probability of success at each trial. If *N* is the number of failures preceding the first success, also known as the waiting time to the first success, then *N* has the *geometric distribution*:

$$P\{N = n\} = (1-p)^n p \tag{3.2}$$

In the case of the Level-1 trigger, this distribution characterizes the number of rejects the Level-1 issues before an accept occurs, with $p = p_{accept}$.

Step 4

Continuing the Bernoulli choices discussion, the mean number of successes in *n* trials is np. If *n* trials are conducted in a given interval *x*, the success rate, i.e. the number of successes per unit time is:

$$\lambda = \frac{np}{x} \tag{3.3}$$

For very small probabilities $p \ll 1$ and large $n \gg 1$, in such a way that λ remains constant, we have ([16]):

$$\lim_{\substack{p\to 0\\ n\to\infty}}(1-p)^n = \lim_{n\to\infty}\left(1-\frac{\lambda x}{n}\right)^n = e^{-\lambda x} \tag{3.4}$$

Equation 3.4 shows that the geometric distribution approaches the exponential distribution at the limit as the trials are taken infinitesimally close together, each with very small probability of success. The average number of successes per unit time has to remain constant.

Step 5

The conditions enumerated above are met by the Level-1 trigger because: $p = p_{accept} = 0.25\% = 0.0025 \ll 1$, *n* is given by the rate collision rate in the detector and the average rate of accepts is 100KHz.

In these conditions, we are able to state that the Level-1 accepts occur at exponentially distributed intervals, with the rate $\lambda = \frac{1\,event}{10\,\mu s}$.

3.3.2 Other queuing characteristics

The rest of the characteristics based on which the queuing model is chosen are:

- *service time* - processing an event in Level-2 can involve a number of iterations on the cycle: *ask for data* $\rightarrow$ *receive data* $\rightarrow$ *process data* until a decision is made. The distribution of the event processing time is difficult to describe using a well-known distribution, reason for which we consider the service time in Level-2 to have a *general distribution*.
- *number of servers* - the number of L2PU computers is 500, each with 8 Level-2 processes (L2P) running in separate threads. The number of servers is thus 4000.
- *queuing discipline* - we consider a simplified version of the L2PU process where there is no queue attached to it. The queuing discipline is irrelevant in this case.

3.4 M/G/c/c queuing model

In the extended Kendall's notation [2, 1] a queuing model is written like $A/B/C/K/N/D$, where:

- A is the arrival process

- B is the service time distribution
- C is the number of servers
- K is the number of places in the system
- N is the calling population ("customers")
- D is the queue discipline

In our case we use a simplified notation $A/B/C/C$ because: the number of places in the system equals the number of servers (C=K), the events are generated uninterruptedly for a long period of time (N can be considered ∞) and D is irrelevant, as there is no queue in front of each server.

Given the premises above, the queuing model becomes the $M/G/c/c$ model, which is detailed in [1] and depicted in Figure 3.3. M stands for exponential/Memory-less/Markovian distribution, G stands for general distribution and c represents the number of servers/places available in the system.

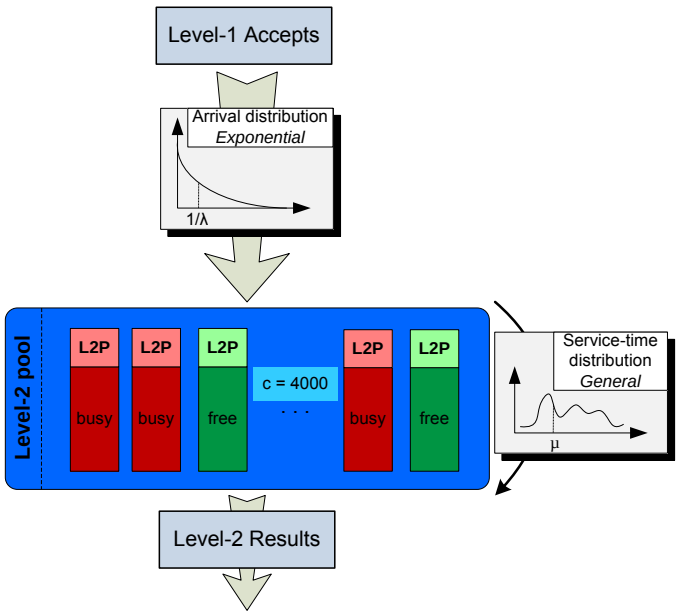

Figure 3.3: M/G/c/c queuing model for Level-2

In this model, events arrive at exponentially distributed intervals with mean $1/\lambda = 10\mu s$, equivalent to stating that events arrive according to a Poisson process [16] with rate $\lambda = 1\, event/10\mu s$. The L2Ps event processing times (i.e. service times) are independent and identically distributed with a general distribution characterized by mean μ. By design, the number of available L2Ps is $c = 4000$ [3]. When a new event arrives, it immediately goes into service if there is an L2P

[3] In the operational phase this number is ~7000, but for consistency with the design the computations were performed with 4000 L2Ps

available, otherwise it will be dropped from the system. We want to obtain the probability of an event being dropped, called *blocking probability*, as a function of the mean service time, μ and the number of servers *c*.

The situation of an event being rejected from the system occurs when all the places in the system are occupied at the arrival time. This can be expressed as the state <*system is full*> in which the customer finds the system on its arrival. For Poisson arrival processes, it holds an interesting property, called PASTA (see explanation in Appendix A.1) which states: *the probability of occurrence of the particular state seen by an event is equal to the blocking probability the system is actually in, calculated over a long period of time.*

It is shown that the probability of an event finding *n* customers in the Level-2 system is:

$$p_n = \frac{\rho^n/n!}{\sum_{k=0}^{c} \rho^k/k!}, \quad n = 0, 1, ..., c \tag{3.5}$$

where $\rho = \lambda\mu$ is called *the occupation rate* or *server utilization*, representing the fraction of time the system is busy. For a stable system, the condition $\rho < 1$ needs to be satisfied.

Consequently, the blocking probability $B(c, \rho)$, defined as the probability of an arriving event finding all the L2Ps busy, is given by:

$$B(c, \rho) = p_c = \frac{\rho^c/c!}{\sum_{k=0}^{c} \rho^k/k!} \tag{3.6}$$

After some transformations (see [1], section 11.3), relation 3.6 can be written as a stable recursion:

$$B(c, \rho) = \frac{\rho B(c-1, \rho)}{c + \rho B(c-1, \rho)} \tag{3.7}$$

with $B(0, \rho) = p_0 = 1$, allowing for a simpler numerical computation for large values of *c*.

3.5 Results

The relation 3.7 gives us a three-dimensional trade-off space between: the number of servers (c), the occupation rate (ρ) and the blocking probability (p_c). In Level-2 "language" these variables translate into: the number of Level-2 processes, the average event processing time (encompassing the network delay) as a function of the arrival rate and the probability that the system is full.

3.5.1 Dependencies and model scaling

In order to quantify and understand the dependencies between these parameters, we applied formula 3.7 for different pairs of *(c, arrival rate)* and over a range of blocking probabilities. We know that by design the arrival rate in Level-2 is 100 KHz and the number of available L2Ps is 4000. The rate of events assigned for a single L2P is thus 25Hz.

Keeping this factor constant, we started the computation from a scaled-down *(c, arrival rate)* pair and then continued over exponentially increasing pair values up to the design ones: (4000, 100KHz). The goal was to understand how the queuing model works in small, medium and large systems and if the same scaling stands for the values of interest: average service time and blocking probability.

The dependency between the service time and the blocking probability, for a given number of Level-2 processes and the corresponding arrival rate is presented in Figure 3.4. We can observe that the bigger the number of servers/associated arrival rate, the greater the average service time allowed for a fixed (e.g. 1%) blocking probability. This tells us that, from a scaling perspective, the greatest influence on the average service time is attributed to the number of servers and not on the arrival rate. Table 3.1 summarizes the results from Figure 3.4:

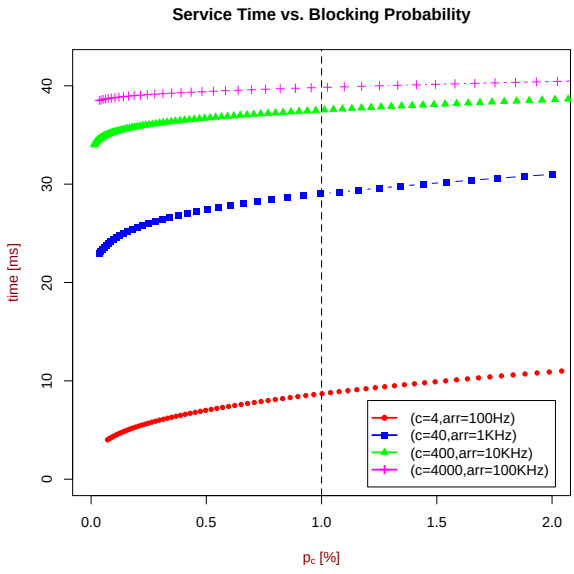

Figure 3.4: Service time vs. Blocking probability - different *(c, arrival rate)*

We can also notice the asymptotic trend of the plots towards zero, a confirmation of the the mathematical formulas, which show us that 0% blocking probability is never reached for positive values of service time.

Arrival rate	No. of servers	Average service time ($p_c = 1\%$)
100 Hz	4	8.5 ms
1 KHz	40	29 ms
10 KHz	400	37.5 ms
100 KHz	4000	40 ms

Table 3.1: Queuing model results

The next thing we need to examine is the sensitivity of the average service time to the number of servers available in the system when the arrival rate has a fixed value of 100 KHz. Figure 3.5 indicates the fact that for a fixed blocking probability (i.e. 1%) the values for the average service time grow linearly with the number of servers available.

As a consequence, in case of Level-2 the mathematical model gives us the impact of losing a certain number L2Ps on the probability of saturation and on the average Level-2 processing time. For 1% blocking probability, the variation is:

$$\frac{\triangle \mu}{\Delta c} = \frac{1\,ms}{100\,L2Ps} \tag{3.8}$$

which means for example that for each 200 L2Ps lost (i.e. 25 physical computers) the allowed average service time in Level-2 decreases with 2 ms.

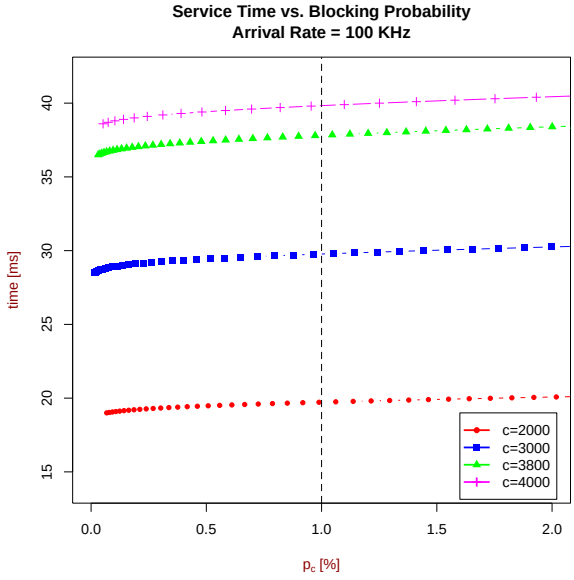

Figure 3.5: Service time sensitivity with the number of servers for 100KHz rate

If we extend the range of probabilities to reach much smaller values, we obtain the plots in Fig-

ure 3.6, where the probability axis is logarithmic. It can be observed that the linear behavior explained before stands for smaller probabilities too. Furthermore, we can compute the sensitivity of the service time to the blocking probability: for $c = 4000$, a decrease of p_c with 7 orders of magnitudes, i.e. from 1% to $10^{-7}\%$ introduces a decrease of $\sim$4 ms on the service time.

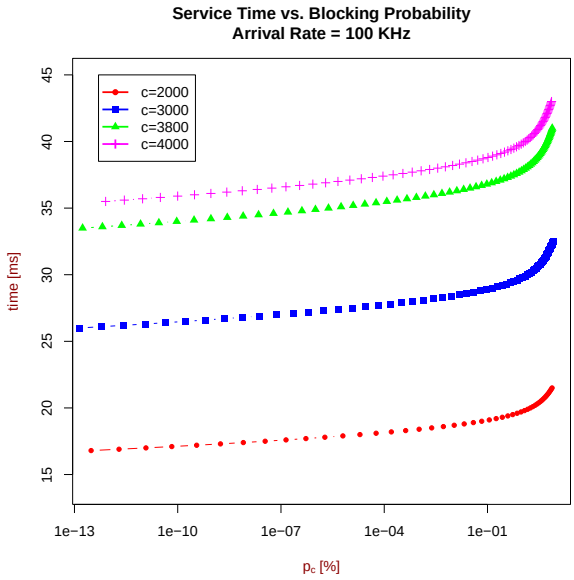

Figure 3.6: Service time/probability sensitivity with the number of servers for 100KHz rate

Although simplified and purely mathematical, the model employed above returns also an order of magnitude for the upper limit of the cumulative network delay allowed in Level-2. For an imposed probability of saturation this value is of order of tens of microseconds (40 ms for 1%, 36 ms for $10^{-7}\%$). This happens hypothetically when physics algorithms take zero processing time in Level-2 and the total service time is composed only by the communication (network) delay.

3.5.2 Particular case study - 1/2 of the system

We can apply the results above for a particular case which is of special interest from the networking point of view: losing one of the DataFlow core switches. In subsection 2.5 it is mentioned that the DataFlow network is redundant through the deployment of two core switches, visible in Figure 2.7. However, this redundant solution protects the entire Read-Out System, but only half of the Level-2 system.

In the scenario with one core switch failing, the traffic from all the ROS computers will be transported through the working switch to one half of the XPUs, whereas the other half of the Level-2 system will lose connectivity. Consequently, all the Level-2 processing tasks will be taken over by the half of the computers which remained connected. Using the mathematical

model, we want to quantify the effects of this particular situation, where $c = 2000$, on the average service time and on the blocking probability.

Figure 3.7 details the dependency between the service time and blocking probability for $c = 2000$, with probabilities ranging from $10^{-13}\%$ to 10% and for 100KHz input rate. The probability axis has logarithmic scale.

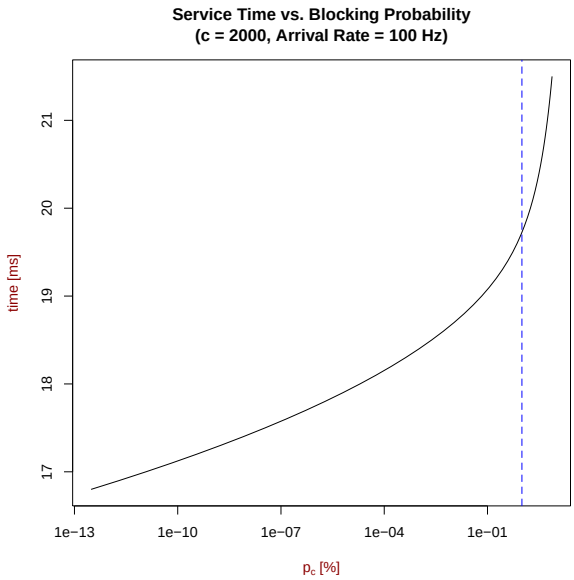

Figure 3.7: Service time vs. Blocking probability

We can observe that the relation 3.8 is met for 1% blocking probability, the service time in this case being obtained from the average service time for 4000 servers and deducting the amount for 2000 of them:

$$40ms - 2000 \times \frac{\triangle\mu}{\Delta c} = 40ms - 2000 \times \frac{1ms}{100} = 20ms \tag{3.9}$$

which is confirmed by Figure 3.7. Also, in the "half of the system" case, the sensitivity of the service time to the blocking probability is smaller than in the case with a full system: a decrease of p_c with 7 order of magnitudes, i.e. from 1% to $10^{-7}\%$ introduces a decrease of $\sim$2 ms on the service time, compared to $\sim$4 ms, as it was in the full system case.

The conclusion we can draw from the results given by the mathematical model is that all the performance metrics scale with the same factors for server number decrease, 1/2 in our case.

3.6 Fidelity analysis

In the previous section we presented a mathematical model for a simplified version of the Level-2 system. The simplifications and assumptions made were required by the process of fitting a computable mathematical model. We have shown that the model has a solution for the performance measures of interest, i.e. the blocking probability and the average service time, which allowed us to understand the relationship between key Level-2 aspects:

- **average time** spent in the Level-2 by an event, directly linked with the time needed to clear ROB space. The Level-2 network delay contributes to this amount of time.
- **probability of saturation** of the Level-2, seen as the probability of an event finding all the L2Ps busy. The higher the average time spent in Level-2 by events, the larger the probability of system saturation.
- **number of Level-2 Processors**.

Next we will summarize the assumptions which we exploited in the model and we will analyze the fidelity of the employed mathematical model to a closer to reality one. The complexity of the Level-2 system is high compared to what a mathematical model can capture. Trying to bring the model closer to the real system complicates the mathematics underneath to a level which does not allow the computation of any performance measure.

However, for some of the changes in the mathematical model we will quantify the absolute difference which they would introduce. We show in this way that the utilized model offers the best compromise between accurate results and computational effort.

3.6.1 M/G/c/c and M/G/c/K models

One important simplification that we employed when creating the model is the zero waiting room in the Level-2 system. In reality, each L2P application incorporates a waiting queue for events, of programmable size. This parameter is called *l2puQueueSize* [31], its value being typically set to 2. The waiting room in the Level-2 system is thus $2 \times number\,of\,L2Ps$, which, added to the actual processing room, gives a total number of places in the system (the K parameter in Kendall's notation) of $3 \times number\,of\,L2Ps = 12,000$.

The M/G/c/c queuing model we exploited is a particular case of a complex problem: the M/G/c/K model, which would apply better in our situation (i.e. M/G/4000/12000 system). The problem is that although extensive effort has been put into solving the performance measures for this queuing system, the solutions can be found only numerically and for small values of *c* (up to 10) (see [30] for example). Furthermore, when trying to compute the blocking probability for this kind of system, the variability of the service time is required, adding another level of complexity to the problem.

3.6.1.1 Error in the blocking probability

Given the constraints mentioned above, the M/G/c/c model occurs to be the closest to reality and mathematically tractable model. In order to quantify its "closeness" to reality we want to understand the amount of probability mass by which we are wrong, when calculating the blocking probability in the M/G/c/c case, compared to the M/G/c/K case.

Our approach uses two queuing models which are mathematically solvable: M/M/1/K and M/M/1. The first is the model of an one-server system with a queue equal to *K*, while the latter is also known as the M/M/1/∞, i.e. implies one-server system with an infinite queue. The difference in probability mass between these two models will give us a good indication on the scale of the same difference between the M/G/c/c and M/G/c/K models.

The absolute difference between probabilities is given by equation A.3. Figure 3.8 depicts the dependency of this difference, called the absolute error, on the number of places available in the system (*n* - corresponding to K from the notations above) and on the loading factor (ρ). For a system close to saturation, ρ is close, but smaller than 1.

We can notice that the absolute error between these two queuing models is extremely small for *n=4000*, even when $\rho \to 1$: its order of magnitude varies between $10^{-180}\%$ and $10^{-14}\%$.

Figure 3.8 contains also the graphical representation of equations A.4 and A.5. They represent the differentials of the absolute error with respect to *n* and ρ respectively. We can observe that the decrease rate of the absolute error (as *n* grows and the loading factor decreases) is also extremely small, more pronounced than the absolute error itself.

There are no mathematical means to calculate the blocking probability error between the used model (M/G/c/c) and the more suitable model (M/G/c/K). We can only infer from the values obtained above that the absolute error of interest is also extremely small. This gives us a scale of the error between the M/G/c/c model and the model closer to reality: M/G/c/K.

3.6.2 Abstraction

Another simplification introduced by the mathematical model refers to the Level-2 supervisors. As mentioned in Section 2.3, a Level-2 supervisor has the role of dispatching the Level-2 processing tasks to the Level-2 processors found under its coordination, based on a load balancing algorithm.

The mathematical model would have become far more complex in this situation. Hence, we abstracted away from the Level-2 supervisors in our model, considering a general distribution for the Level-2 processing time. Thanks to the exponential arrival into the Level-2 and to the PASTA property, the model takes into consideration only the average value of the event processing time and not the mechanisms underneath.

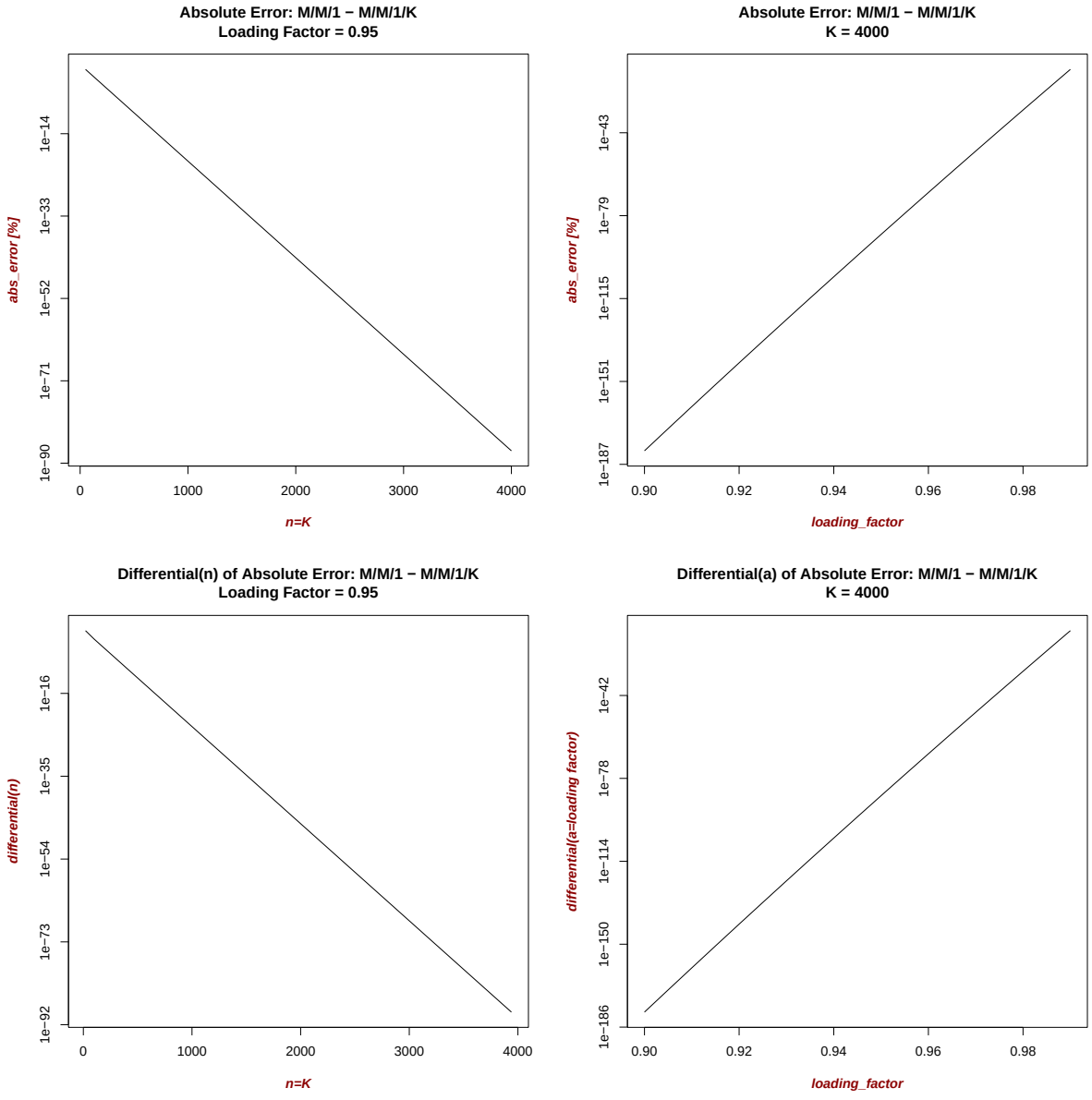

Figure 3.8: Absolute error and its derivative between M/M/1 and M/M/1/K

3.7 Conclusions

The queuing theory model employed for the Level-2 system is able to give us important information about the Level-2 system performance such as the probability of saturation and the average L2 processing time. The queuing theory is used to model the state of the resource that's being contended, the Level-2 processing room in our case. From here, the performance measures mentioned above can be extracted.

One of the limitations of this approach is that the information regarding the performance of the DataFlow network - a component of the Level-2 - is, however, limited. From the results in Section 3.5 we can obtain the upper bound on the accumulated network delay assuming that the processing time is negligible, which, in the current implementation is not the case. Another limitation comes from the simplifications and assumptions considered when the M/G/c/c model was chosen and the difficulty of using a different, more complex mathematical model. One aspect which the mathematical model can not capture is the dependency of the network performance on the network load.

In the next chapter we will introduce a different approach, applicable to any communicating systems, therefore to Level-2 as well, which is able to extract performance measures for the data network and consequently guide its performance analysis. This new approach also addresses the relationship between load and performance.

Chapter 4

Observational model

4.1 Introduction

The real time requirements imposed on the ATLAS TDAQ system, especially on the Level-2 system generate an acute need to comprehend the performance of its data network infrastructure. A complex and detailed network monitoring system [39, 53, 6] has been put in place and is currently exploited in ATLAS for post-analysis of performance and real-time fault detection.

The performance indicators are obtained with respect to the requirements imposed on the data network in the design stage. These refer to the amount of traffic on all the links, as well as lost, discarded and erroneous packets. Although requirements in terms of delays can be extracted from the event rates, they concern only the applications, not the data network. Consequently, the current performance monitoring system doesn't capture the amount of delay the network introduces during the system run.

The reason is that the network delay aspect in the context of the ATLAS TDAQ was/is considered to be of minor importance in terms of impact on the TDAQ performance. It is considered that a high speed network, located in a single geographical location produces delays with one order of magnitude lower than the allowed application delays. Likewise, due to the buffering and allowed latencies detailed in Section 2.6 the system is considered to be highly tolerant to latency variations.

As introduced in [33], we consider that the amount of time an application spends while waiting for data transport needs to be properly understood and its evolution with the load of the system needs to be evaluated. This is required in order to have a predictive model which is able to describe how the system performs in conditions close to saturation. Applications do not care about how data is transported, they care only about how promptly and reliably they receive it. In this chapter we propose an end-to-end description of a communication system from the delay and reliability perspective aiming to apply it in the end to the ATLAS TDAQ network.

4.1.1 General context

Data networks have witnessed a significant growth with respect to both the geographical expansion and the number of incorporated technologies. Internet is the main driving factor for this process as the end-user's demand to access the Internet grew continuously in terms of "how much", "where" and "when".

Along with these, a key aspect which the end-user perceives - by means of applications - is the response time from the communicating peer. We refer to this as the Quality of Experience (QoE) and it becomes a performance metric when comparing two network access experiences. If the first aspects: bandwidth, time-of-the-day and location access are usually known beforehand from e.g. a contract with an ISP (Internet Service Provider), the response time is an instantaneous measure and is an emergent property of the network itself. Given the number of possible factors influencing the response time, a uniform approach in describing the network performance is thus needed.

This description must also incorporate the relationship between the performance and the system load in order to understand the network behaviour in saturation conditions.

4.1.2 Etymology

The term "observational" that we use for our framework comes from the process algebra paradigm for modeling communicating systems. It states that a system is defined by its outputs and inputs only - the black-box philosophy, therefore a comparison of two system is based only on observing the inputs and outputs. Two systems are behaving the same if they preserve the externally observable behaviour, e.g. they produce the same output in response to identical input.

This approach is formalized in the *observational bisimilarity* theory in [40, 41] and provides a framework for constructing and comparing different models, at different levels of abstraction:

- strong equivalence
- *observational* equivalence
- *observational* congruence

Our work is based on an abstraction level which relies only on observation and does not delve into low-level technical or implementation details, hence we name the framework the *observational model.*

4.2 Approach

For a system which relies on interactive communication between its elements there is a fundamental cycle of behavior which characterizes it. Examples are the communication protocols,

which have phases exhibiting cycles of behavior, e.g. request-response protocols. Furthermore, viewing a system as a set of resources, during a cycle the system has to perform is a "walk-through" a subset of those resources. The cycles, due to their repetitive nature, constrain the emergent performance of the communication system.

4.2.1 Performance

Performance in this context is a statistical[1] measure for answering questions like:

- How quickly you get round that cycle?
- How predictably you get round that cycle?
- What is the probability of getting round the cycle before an event/timer occurs?

In order to answer these questions, we decompose a cycle into so-called interactions which can be tasks and/or operations, including here potential resources utilization. Using a resource involves: gaining access to it, using it, releasing it, all operations which influence the executed task. The tasks can also be concurrently performed, involving the existence of shared resources, which consequently results in a non-deterministic response time from these resources.

4.2.2 ΔQ

Performing a complete cycle is thus equivalent to "passing through" or "experiencing" all those interactions. From the point of view of the cyclic process two things can happen when an interaction takes place:

- the process is *delayed* with a time T.
- the process *loses* all or part of its elements, e.g. bits of information. Although normally applications recover from lost information by implementing re-transmission protocols, from the point of view of the interaction we refer to this as *failure*.

Ideally, a perfect interaction is one which has $T = 0$ and the probability of losing something inside it also 0. We define the **Quality Attenuation (ΔQ)** attached to an interaction to be the deviation from this ideal situation, i.e. a positive value for T and a binary value for failure (YES/NO). Given the repetitive nature of cycles, ΔQ becomes a statistical measure incorporating both the *distribution of the execution time* and the *probability of failure*.

If we look at a higher level, in order for an application exploiting the system to work the communication between two specific peers (which can include more interactions) in the system needs

[1] We use "statistical" to express the fact that we don't deal with scalar variables when talking about: performance, delay, time. These are random variables characterized by distributions

a certain *quality*, i.e. a bound on loss and delay characteristics. ΔQ is a method to manage and capture this bound.

Moreover, the use of ΔQ for evaluating the effect of a network element also extends to the characterization of bandwidth. This can be achieved looking at the effect of the arrival rate exceeding the service rate (see [2, 1] for an explication of queuing theory terms). If this happens for a longer period than the buffering allows then packet loss will ensue - in line with the observational paradigm: excessive offered load results is an observed change in ΔQ.

4.2.2.1 Mathematical support

In the context of communicating systems the probability of failure is fundamentally a time constraint on the execution time. From an observational point of view a system which will never produce a result - a real failure - and a system which produces the results with an excessive delay are indistinguishable. If the response expected from an interaction exceeded a time limit the executed interaction is considered FAILED, otherwise a SUCCESS. This allows us to introduce next a mathematical support for ΔQ in order to encompass both the execution time and the failure to execute: the *improper CDF*[2] .

First of all, we mention that a "proper" CDF for a random variable X is given by a function F_X defined as:

$$F_X(x) = P(X \leq x),\ x \in (-\infty, +\infty) \tag{4.1}$$

with two properties:

$$\lim_{x \to -\infty} F_X(x) = 0 \tag{4.2}$$

$$\lim_{x \to +\infty} F_X(x) = 1 \tag{4.3}$$

A ΔQ is defined as an improper CDF and it differs from a "proper" one in that it is defined only on the interval $(-\infty,\ T_f)$ and that

$$F_X(T_f) = p_f,\ p_f \leq 1 \tag{4.4}$$

as depicted in Figure 4.1. An improper CDF is thus not required to meet the property in relation 4.3, allowing an amount of probability of failure: $1 - p_f$. T_f is called the failure time and represents the time constraint placed on the execution time - a time-out.

The performance of an entire cycle is thus given by the total accumulated ΔQ. In order to compose the ΔQ back from the individual interactions we need to make use of *distribution convolution* as we would do for the normal CDFs. For improper CDFs we can consider the rest of the

[2]Cumulative Density Function

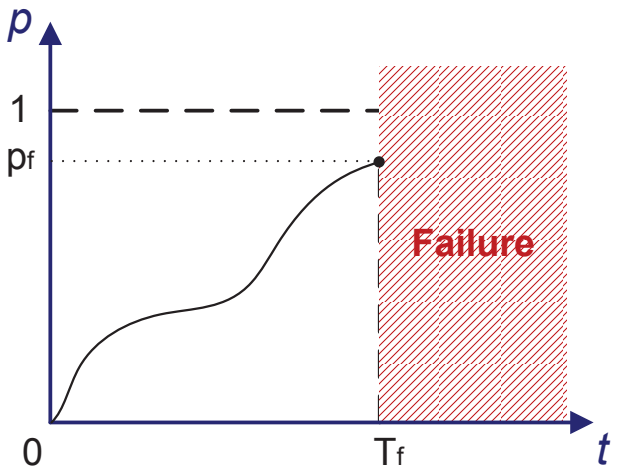

Figure 4.1: Improper CDF exemplified

probability mass to be towards $+\infty$, thus enabling the use of convolution for these distributions in the interval of interest.

4.2.3 Design

Before moving on with utilizing the ΔQ in a formal framework we need to introduce an approach for describing the design process for a communicating system as a whole. We will show how the ΔQ concept is also able to assist the system design stages from establishing the requirements to validating the system, or the other way around, as we will see for ATLAS TDAQ network.

We propose a system design flow terminology which can be expressed by the acronym **AREA** (**A**spiration $\rightarrow$ **R**equirement $\rightarrow$ **E**xecution/**E**nsurance $\rightarrow$ **A**ssurance) depicted in Figure 4.2, where:

- *Aspiration* is a goal or a desired set of system behaviors, expressed informally e.g. "I want a web page to load in less than 2 seconds".
- *Requirement* represents that which is required by the entire set of computational and communicating tasks - together with their associated cycles - to achieve the aspiration. It is the process of reification of an aspiration using formal terms, in our case improper CDFs. An aspiration is thus translated at this stage into a set of requirements on ΔQ distributions. They will serve as criteria when performing the actual implementation.
- *Execution/Ensurance* is the implementation stage in a system design process. In an execution environment, where the desired loading conditions must be simulated or synthesized, ΔQs of interest are measured and are compared with the requirements. If they are not met, changes to the system are made until we ensure that all the outcomes' ΔQs meet the requirements. An iterative loop can occur between R and E steps due to the possibility that

certain requirements are not achievable, hence they need to be adjusted. This feedback can be propagated back to the Aspiration step where the goals can be adjusted as well.

- *Assurance* is the validation stage of the system in a production/commercial environment, i.e. quality assurance. We measure the ΔQs of interest and assure that they meet the requirements.

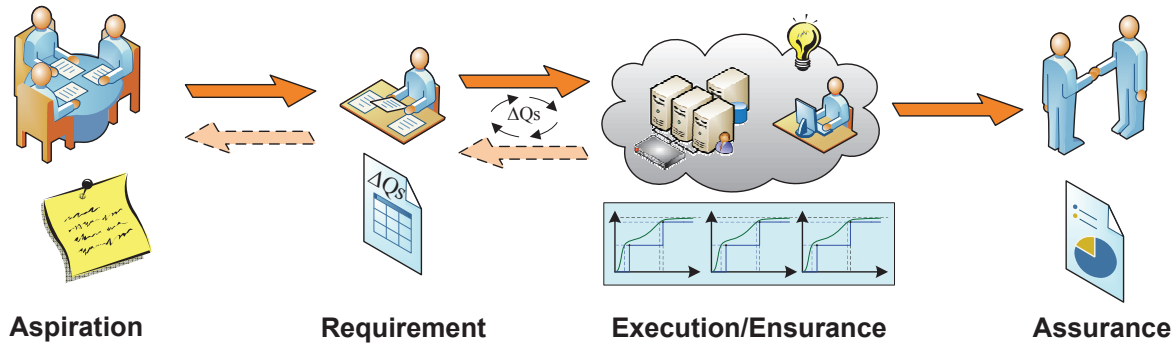

Figure 4.2: Design flow stages

In the particular case of the ATLAS TDAQ network the first three steps were performed considering the delay and loss as minimal impact for a design criteria and considering only average values like bandwidth. The design value was set to 60% based on average values since there weren't any means of predicting the effects of instantaneous collisions at the design time. Given the operational stage the system is in at the moment this work is carried on, we can only perform the last step, the Assurance, which in this context can be considered more appropriately an *Analysis* step. As a result we can deduce backwards what the application (the TDAQ software) should expect from the network in terms of latencies.

We will introduce next a formal way of manipulating the ΔQs in order to accomplish the AREA flow mentioned above.

4.3 Concepts

The decisions which need to be made in the last two design stages are guided by the ΔQ (measured and required). This is possible because we defined ΔQ with mathematical support which allows us to use it as a *comparison metric*.

In order to do that, we need to measure it in a formalized framework, characterized by a set of concepts:

- an *outcome* is a task the system has to perform, to which performance measures are attached: time to complete and resource consumption. Outcomes can be viewed at different levels of abstraction, i.e. they can be decomposed or grouped together.

Figure 4.4 depicts a sequential scenario for executing a task which requires: access to the network, computation and writing to a disk. Each of these steps can be viewed as individual outcomes or they can be encompassed by one global outcome. We refer to this technique as *abstraction*.

- *observables* are time-aware indicators in the system that are triggered by a particular task. We use observables to capture when a task has started or ended in order to measure its execution time. The way of choosing the observables reflects the level of abstraction the corresponding outcome is captured at.

- the ΔQ *(Quality Attenuation)*, introduced earlier is the measure for an outcome.

- the *load on a resource* expresses the instantaneous demand on a resource or on a set of resources by all the tasks in the system. It directly influences the outcome's ΔQ by delivering a response time dependent on the load.

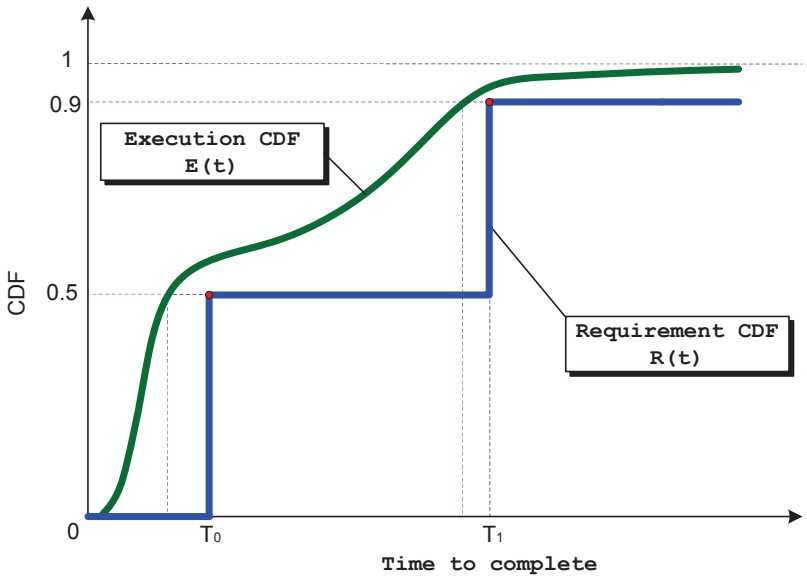

Figure 4.3: Improper CDF as a metric

The formalization introduced earlier allows us to define the goals a system has to achieve and the method for establishing if they are met or not. We introduce two concepts for ΔQ relative to the Requirement and Execution/Ensurance stages in the system design flow:

- a requirement ΔQ (RΔQ) is the desired ΔQ, i.e. the design criteria

- an execution ΔQ (EΔQ) is the ΔQ which is measured when the system is put into operation.

A system is considered to have delivered the required performance if all the outcomes have better EΔQs than RΔQs. Figure 4.3 exemplifies how the improper CDF for ΔQ can be used as a metric for a given task. The required execution time for the exemplified task is expressed by the RΔQ's CDF, stating that:

- 50% of the time the task has to finish earlier than T0
- 90% of the time the task has to finish earlier than T1

The outcome meets the imposed requirements on the ΔQ if its execution CDF is always larger than its requirement CDF. Mathematically, the execution meets its requirement if the relation 4.5 is satisfied:

$$E(t) \geq R(t), \ \vee t > 0 \tag{4.5}$$

4.3.1 Abstraction

An example for the observational model introduced above is illustrated in Figure 4.4. Consider a system which relies on three main tasks: transmit data over the network, perform some computation and store data on the disk. One can choose to have one global outcome encompassing the networking, CPU and disk as one resource. In this case the aggregated response from all of the resources defines the global ΔQ. Alternatively one can choose to split the system tasks and associate each of them an outcome and ΔQ.

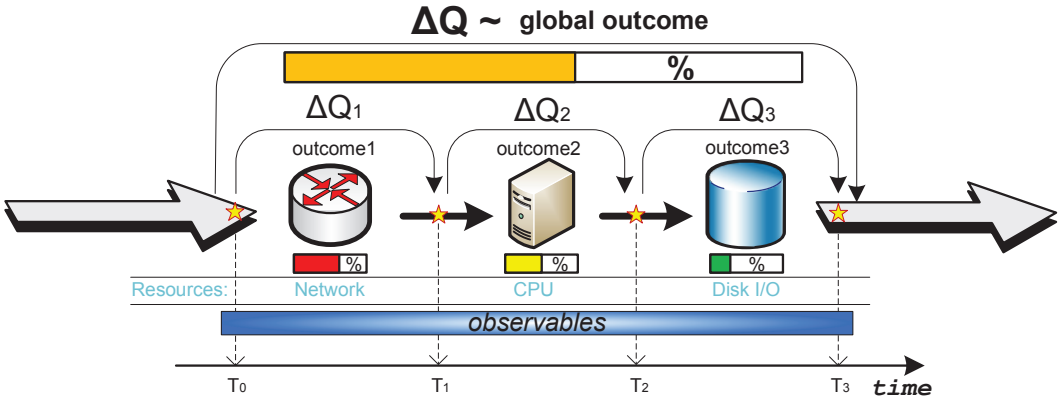

Figure 4.4: Abstraction exemplified in observational model

The observation points, i.e. observables are the elements which define the abstraction level of an outcome. Two scenarios are possible in terms of abstraction levels:

- consider the T_0 and T_3 observables and obtain one global outcome. The task triggers the T_0 observable and may eventually trigger the T_3 observable. The ΔQ is the quality attenuation suffered by the task on its path between the two observables, denoted as:

$$\triangle Q = T_0 \rightsquigarrow T_3 \tag{4.6}$$

The network, CPU and disk resources are grouped together into a single "virtual" resource whose load is a combination of those three.

- look at each $T_{0,..,3}$ observables and obtain three outcomes. In this case we have three associated ΔQs:

$$\triangle Q_{i+1} = T_i \rightsquigarrow T_{i+1},\ i \in \{0, 1, 2\} \tag{4.7}$$

and each outcome is associated with an individual resource consumption.

4.3.2 Resource utilization

In addition to the performance indicators mentioned until now, loss and delay, we need to take into account the resource consumption. This is important because requirements can be placed on resource utilization too, we therefore need a method of inferring on the resources utilization which occurs in the system.

The classical approaches explicitly model the resources in order to extract their consumption. Extensive work was carried on in this direction, e.g.: QoS-based Resource Allocation Model (Q-RAM) in [52], scheduling framework for the CPU resource allocation in [12], real-time scheduling theory with different priority schemes in [49, 38]. All these models involve modeling all the access mechanisms and all the contention mechanisms specific to a shared environment, thus becoming unmanageable for large systems in which contention for resources occurs with a high frequency.

When we address the resource utilization aspect, we consider that they are *stochastically shared* between a numbers of "consumers". The philosophy of sharing implies a competition for resource, which, from the perspective of the process trying to access it, can be done in three ways:

- preempting the process that currently uses the resource, implying that all other processes must wait.
- every other process can preempt it
- it fairly shares the resources with the rest of the processes

Regardless of the type of competition for resources, we consider that the amount of contention (i.e. load) a resource is dealing with is reflected in the ΔQ associated to the outcomes using those resources. When a task uses a set of resources, the obtained ΔQ (through its statistical descriptors: mean and variance) is a function of how loaded those resources are. From the point of view of a particular task, we can abstract away from the rest of the tasks in the system by looking only at the load they put on the resources of interest.

The coupling of the resource load and the outcome's ΔQ allows a top-down view of a system. This means that one can start from the set of outcomes a system has to fulfill and derive the set of constraints on resource consumption, i.e. the set of necessary properties of a system in order to fulfill the imposed outcomes. Having this knowledge, one can also prioritize access to resources in an apprehensive manner.

4.4 $\triangle$Q in data networks

We will show how the concepts introduced above apply to a particular type of system, a data network. Specific to these systems are the resources involved, the type of performed tasks, the type of observables and the structure of the ΔQ.

4.4.1 Particularities

Data networks are meant to serve a purely shared environment, where a large numbers of users access them, making a suitable case study for our methodology of extracting performance measures from complex systems. Furthermore, specific to data networks is the component-wise structure of the ΔQ which will help us understand the relationship between ΔQ and the load of the network.

4.4.1.1 Resources

A data network, regardless of its type (Ethernet, ATM, PPP, Token Ring etc), exhibits two fundamental types of resources:

- the transmission medium: copper, optical, radio waves
- the devices handling the flow of data: repeaters, hubs, bridges, switches, routers

These resources have different access mechanisms (CSMA/CD, token passing), transporting speeds (10M, 100M, 1G, 10G, 100G), technologies, protocols and architectures (user plane, control plane), parameters which create a first classification of resources. Furthermore, in a shared resource environment these two types of resources have different physical and logical mechanisms for managing the contention for them, becoming additional criteria for an extended classification of resources and consequently of the data networks.

4.4.1.2 Tasks and observables

The task a data network has to fulfill in most of the cases is to transport packets between two points of the network. The associated observables are in this case the time-stamped traces captured on the network interfaces defining the end points.

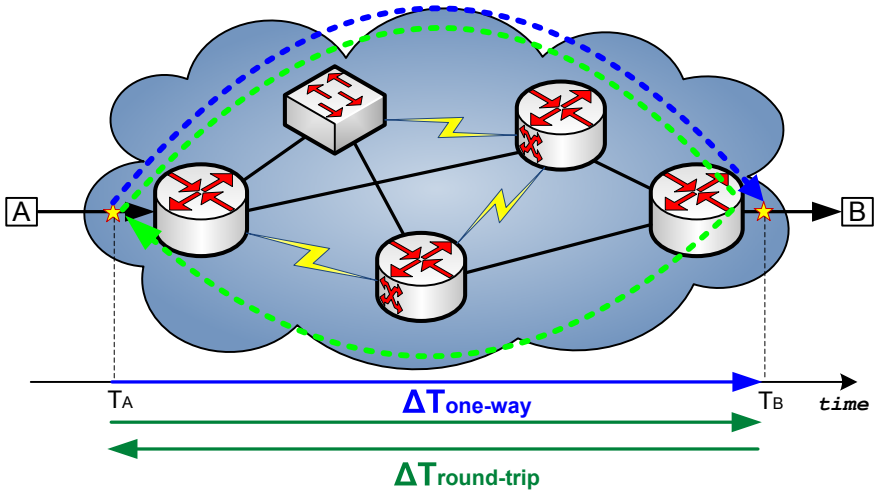

Figure 4.5: Tasks and observables in data networks

Measuring the difference between the two values yields the delay a packet suffers from source to destination, and hence the ΔQ for this system. The failure of a task in the data network context is expressed by an "over-the-threshold" delay, signaled as a packet loss.

There are two possibilities in measuring ΔQ, depicted in Figure 4.5:

- consider only one observation point and measure the ΔQ for the cycle, e.g. A and $\triangle T_{round-trip}$ respectively in Figure 4.5. In practice this is called the *Round-Trip Time (RTT).*
- consider two or more observation points and measure the ΔQ for each segment, e.g. A, B and $\triangle T_{one-way}$ respectively in Figure 4.5. This measurement is called the *One-Way Delay (OWD).*

From a technical perspective, capturing traces in different points of a network requires the use of either the use of expensive equipments like traffic analyzers or commodity PCs with dedicated software (e.g. [20, 59, 62]). The second method is easier and cheaper to be implemented. Measuring RTT is simpler because both observables are using the same clock for time-stamping. For OWD however, it requires very precise clock synchronization between the capturing devices or the implementation of a clock adjustment algorithm.

4.4.2 Components

Each network element - a resource in the system - adds delay and potential loss to the served packets. The ΔQ introduced in Section 4.2.2 is a function of: physical topology, logical topology, packet size, the load of the network, time of day and other factors.

Based on these factors ΔQ can be split into components which fall into two distinct categories: *immutable* (can't be influenced, they depend on the system structure) and *mutable* (manageable,

they depend on factors such as the loading of the system). This classification stems from a limitation of the queuing theory, theory which is the grounds for our analysis and predictions in Chapter 5. Queueing models cannot properly model a common characteristic of data networks: fixed delays. This type of delays always occur due to propagation delays, regardless whether they occur in copper cables, fiber optics or inside electronic components, hence it cannot be avoided. We thus separate the fixed delays from the rest in order to be able to apply the queueing theory on the remaining delay.

There are different approaches in separating the delay and loss into components, e.g. in [9] four major contributors were considered: propagation delay, transmission delay, nodal processing delay and queuing delay. We consider three basis components which build up the ΔQ to be sufficient for our model:

1. **G** - *immutable* - is the contribution of the physical topology, being constant for a given network path. It represents the delay introduced by the network on a "zero length" packet, consisting of: transmission medium delay and the delay introduced by the access to the medium. G comes from *Geographical* or *Given*.

2. **S** - *immutable* - is the contribution of the mechanisms for processing packets, being constant for a given logical topology and packet size. It consists of serialization and deserialization delays, inter-frame gap and any OSI Layer 2 [3] introduced overheads.

   The purpose of S is to compare the differential delay and loss caused by different packet sizes. This is why, as we will see in Chapter 5, for the particular type of networks analyzed in this thesis we move the inter-frame gap and the preamble - both fixed values causing a fixed delay - into G.

   For a given network type, topology and configuration, S is dependent only packet *size*, hence its name.

3. **V** - *mutable* - is the contribution of the contention for resources and in the case of data networks it is caused by the queueing mechanisms. V is indirectly dependent on the network load, but also on other factors such as the time of the day, or the total delay. In the same time, V is independent on the size and network topology. V comes from *Variable*.

These components can be viewed as continuous random variables each characterized by a Probability Density Function (PDF). The decomposition in only three components - as opposed to four components in other approaches - was done with the purpose of obtaining statistically independent components, i.e. any two random variables from the [G,S,V] tuple are *independent*.

We use here the result which states that for a function f of two random variables with

$$f(x, y) = x + y \tag{4.8}$$

its PDF is the convolution of the individual PDFs for independent random variables, as demonstrated in [61].

Hence, for a given network path, the ΔQ is the convolution of the distributions above, being a function of the size and load:

$$\triangle Q(size, load) = G \otimes S(size) \otimes V(load) \tag{4.9}$$

Furthermore, we denote the convolution in relation 4.10 to be the **Structural Delay (SD)** for a network path, incorporating the immutable components only. The Structural Delay can be seen as the ΔQ obtained when the system is idle, i.e. the load is zero, hence being a function of packet size:

$$SD(size) = G \otimes S(size) = \triangle Q|(load = 0) \tag{4.10}$$

Conceptually, the SD for a network is a property which can be used as a comparison metric. Regardless of what type of applications are using the network, this contribution to the overall ΔQ cannot be changed, unless structural changes are made: configurations, connections. Given the wide range of configuration parameters network devices are offering, we consider them fixed and we incorporate them within the SD. A different set of values for the configurable parameters will generate another SD value for the network.

The most important ΔQ component for the analysis performed in this thesis is the one which can be managed and traded within the system: V. Using relations 4.9 and 4.10, we can obtain V by "subtracting[3]" the structural delay from the ΔQ, denoted as:

$$V(load) = \triangle Q \ominus SD \tag{4.11}$$

V is a function of the load, representing the ΔQ introduced by the contention generated inside the network, i.e. the queuing mechanisms. In this context we use V to infer on the queues occupancy in the system, on the traffic pattern and on the load of the network.

4.4.3 Properties

The ΔQ introduced by a network exhibits two important properties:

1. *Conservation.* ΔQ, as a function of the network elements, is monotonically increasing from a topological point of view. The ΔQ can only be accumulated, not diminished and the ΔQ introduced by an element is always a positive value. The operation of "adding" ΔQs is a composition of distributions.

2. *Topology-based convolution.* The components of the ΔQ can be convolved based on topology. Having a system made by A and B, depicted in Figure 4.6, it is possible to calculate

[3]mathematically, "subtracting" two known distributions involves a form of deconvolution. In mathematics, deconvolution is an algorithm-based process used to reverse the effects of convolution on recorded data.

the component-wise convolution for G, S and V for a fixed load and packet size. The aggregate distributions G_{AB}, S_{AB} and V_{AB} can be obtained by:

$$G_{AB} = G_A \otimes G_B \tag{4.12}$$

$$S_{AB} = S_A \otimes S_B \tag{4.13}$$

$$V_{AB} = V_A \otimes V_B \tag{4.14}$$

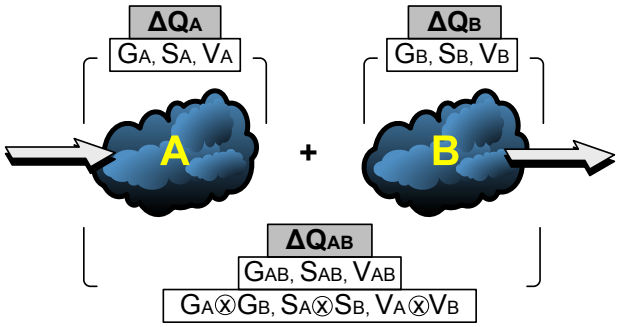

Figure 4.6: Topology-based convolution

The composition of the two variables can thus be calculated by convolution. In [61] it is mentioned that the sum of two random variables ε, η produces a new random variable ζ with the PDF given by:

$$\omega_\zeta(z) = \int_{-\infty}^{+\infty} \omega_{\varepsilon\eta}(z-y, y)dy \tag{4.15}$$

For independent random variables the following can be applied:

$$\omega_{\varepsilon\eta}(x, y) = \omega_\varepsilon(x)\omega_\eta(y) \tag{4.16}$$

Relation 4.15 becomes:

$$\omega_\zeta(z) = \int_{-\infty}^{+\infty} \omega_\varepsilon(z-y)\omega_\eta(y)dy = (\omega_\varepsilon * \omega_\eta)(z) \tag{4.17}$$

which represents the convolution between the ε and η random variables.

Next, using relation 4.9 we obtain the aggregate ΔQ:

$$\triangle Q_{AB} = G_{AB} \otimes S_{AB} \otimes V_{AB} \tag{4.18}$$

However, a common mistake in practice is to convolve the topological ΔQ's directly:

$$\triangle Q_{AB} = \triangle Q_A \otimes \triangle Q_B \tag{4.19}$$

An example for this mistake could be measuring the delay of each sub-system and adding them in order to obtain the end-to-end delay. The relations 4.18 and 4.19 do not yield the same result because of the possible dependencies between ΔQ's. In conclusion, *only piece-wise convolution of the components gives the end-to-end correct answer for* ΔQ.

.

4.4.4 Ethernet case study

In practice, for Ethernet[4] networks, given the G, S and frame size characteristics exposed next, the convolution operation can be simplified.

4.4.4.1 G component

In the case of full-duplex Ethernet G is a constant for a fixed physical topology due to the medium access technology which no longer relies on CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol. The latter introduced a random time to access the medium in case of collisions[5]. The continuous distribution can be thus reduced to a Dirac delta function shifted with the value of G.

We mention in Section 5.3.2 that specific to Ethernet are the preamble and Inter-Frame Gap which is constant for each frame, regardless of its size. We incorporate these two, the Ethernet header, the checksum - all fixed quantities, into G.

4.4.4.2 S component

The Ethernet frame (called generically packet[6]) sizes can take values from a finite set of discrete numbers, from 84 to 1542 bytes as shown in Section 5.3.2. Furthermore, S is dependent on the packet size: the larger the packet, the larger its additional contribution to the ΔQ. This dependency can be approximated by a linear model for the entire range of possible packet sizes, as we will see being considered in Section 4.4.5:

[4]we consider the full-duplex standard, since it became the commonly used one since 1997 as shown in Section 5.3.1

[5]another example is the Asynchronous Transfer Mode (ATM): the medium access method is cell-based, introducing thus a random delay until data is put on the wire

[6]Not related to the OSI [3] model layer denominations, unless specified

$$S(size) = s \times size \tag{4.20}$$

The working assumption that the delay component is linearly dependent on the packet size is utilized for other network performance studies, e.g. in [9]. Furthermore, in [43] it is shown that the minimum packet transit time through a router is linearly dependent on the packet size. Assuming that the linearity holds for other network devices we can consider that the linear relation is valid for a multi-hop network path. This assumption is confirmed by the measurements performed in Chapter 5.

Knowing that the G component of the minimum packet delay is constant, what remains - S - is consequently a linear function of packet size, as shown by relation 4.20. This relation expresses the fact that S is a linear function of a random variable *s*.

4.4.4.3 V component

In order to obtain V we need to apply relation 4.11. Knowing that, for Ethernet, the G and S are constants, for each packet size V becomes a transformation of random variables:

$$V(load) = \triangle Q(size, load) - [G + S(size)] \tag{4.21}$$

In terms of distributions, as mentioned in [61], page 23, the linear transformation of a random variable $\eta = a\xi + b$ creates a new random variable with the distribution ω_η following equation 4.22:

$$\omega_\eta(y) = \frac{1}{|a|}\omega_\xi\left(\frac{y-b}{a}\right) \tag{4.22}$$

which, for $a = 1$ and $b = -[G + S(size)]$ becomes:

$$\omega_\eta(y) = \omega_\xi\left(y + [G + S(size)]\right) \tag{4.23}$$

meaning that the distribution of V for a fixed size is the distribution of ΔQ shifted with a constant value: $G + S(size)$.

4.4.4.4 Convolution

In conclusion, for Ethernet, S can be represented by a delay-per-byte value which is a random variable taking only a constant value for Ethernet networks. Its distribution can thus be reduced to a shifted Dirac delta function. In Appendix B.1 we show that the convolution of two Dirac delta distributions, shifted one with value *a*, one the other one with value *b*, results in a Dirac delta distribution shifted with value $a+b$. The convolution of two constants can thus be expressed

by their arithmetic sum. Consequently, for topological convolution, the global G and S are the *arithmetic sum* of the G and S for the individual segments.

Relations 4.12 and 4.13 become in the case of Ethernet:

$$G_{AB} = G_A + G_B \tag{4.24}$$

$$S_{AB} = S_A + S_B \tag{4.25}$$

For V, the topological convolution can still be obtained using the generic convolution in relation 4.14.

Concerning the Structural Delay, for a fixed packet size the convolution in relation 4.10 becomes:

$$SD(size) = G + S(size) \tag{4.26}$$

4.4.5 Methodology for obtaining G, S, V

Next we will show a method of extracting the G, S and V characteristics for a path through a data network. This method uses a sufficiently large number of ΔQ measurements between the observables which define the path under analysis. As we will see, "sufficient" means a certain number of samples which is able to differentiate a statistical cloud from which trends and demarcation lines can be extracted. In our case this number is of the order of thousands.

Measuring the ΔQ for a network path involves capturing the delay between two observation points for different packet sizes and at randomly chosen and de-correlated moments in time. In practice, for an Ethernet network we will use:

- size: randomly chosen from interval [50,1508] bytes, uniformly distributed, to which the Ethernet specific information is added. This is explained in Section 5.3.2.
- time: randomly chosen and the inter-sample interval following a *negative exponential distribution*. The negative exponential distribution was chosen due to the PASTA property explained in [51].
- samples: 4000

The result will be a scattered plot with points characterized by *[size; delay]* tuples, exemplified in Figure 4.7. As depicted here, for the same size there is variability in the time values, variability which is dependent on the system load.

Given the large number of measurements, the minimum values for the same size, from the scattered plot, are obtained when the loading of the network was close to zero, i.e. $V \approx 0$. The

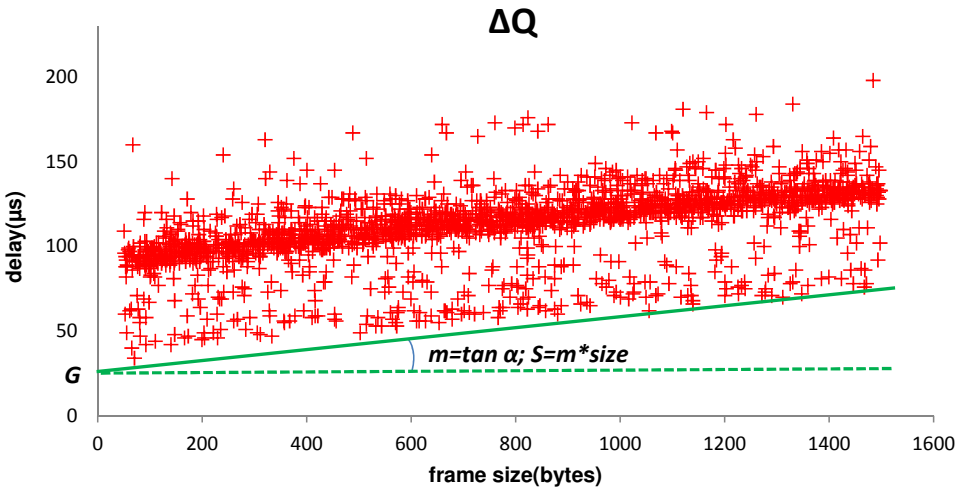

Figure 4.7: Obtaining G, S and V

statistical model for the points representing minimum values can be approximated by a linear model:

$$y = y_0 + mx \tag{4.27}$$

where:

- y_0 is the delay introduced by the network on a zero bytes packet, i.e. the intersection between the regression line through the minimum values and the Y axis. For a zero length packet, there are no serialization, de-serialization delays, inter-frame gaps and no OSI Layer-2 ([3]) overheads, as no encapsulation takes place. Hence, y_0 represents the G component and is measured in time units.

- m is the additional delay dependent on the packet size. Hence, m represents the linear coefficient for S component - where $S(size) = m \times size$ - and is measured in *seconds/byte*. In the rest of the work we will simplify S's notation to express only the non-deterministic component, i.e. the linear coefficient *m*.

- the quantity under the drawn line represents the Structural Delay.

The V component represents the distribution of delay for each packet size, purely dependent on the load of the system. V is thus obtained by normalizing the scattered plot, i.e. deducting the SD from the plot using equation 4.21. Figure 4.8 displays an example of a measured V, showing its independence on the frame size.

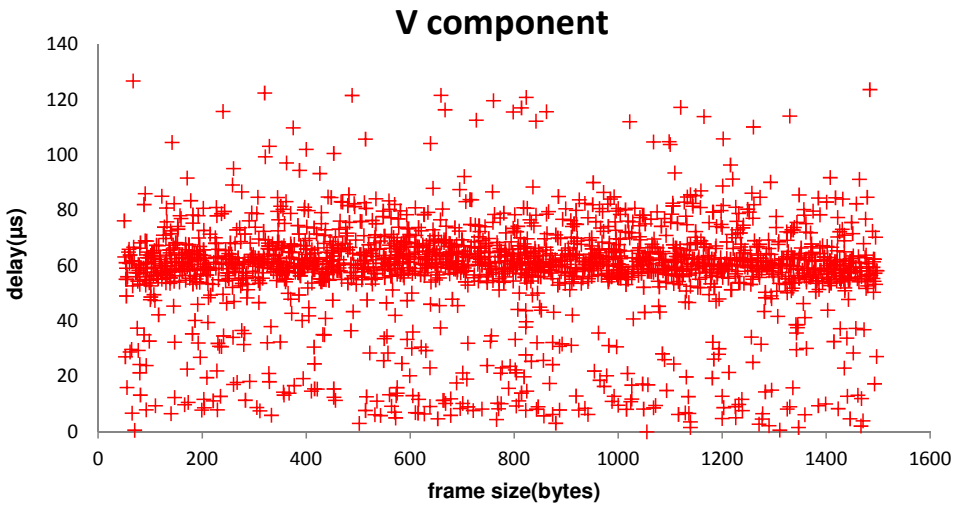

Figure 4.8: V independence on frame size

4.5 Conclusions

The approach we introduced in this chapter targets the definition and description of a communicating system performance, the purpose being to study the performance of the ATLAS TDAQ data network. We showed that in the context of the cyclic behavior these systems exhibit, performance is an emergent property of repeatedly running a set of *cycles*.

In the case of communication protocols overlaid on top of data networks a cycle can be decomposed down to the one-way data transport of an element of the protocol (e.g. a SYN-ACK for TCP as explained in [3, 47]). A result of the task of transporting data is the fact that an element is delayed and possibly lost. We thus introduced the concept of *quality attenuation* (ΔQ) for the purpose of capturing both aspects: *delay* and *loss*, by means of a mathematical support (improper CDF).

We then proposed a design flow approach called AREA which incorporates the steps required to produce a system from the informal requirements gathering until placing it into operation. We showed how the ΔQ is able to guide this process by acting as a metric when making design decisions: compare the requirement ΔQ with the execution ΔQ.

The observational paradigm, which has its origins in Stochastic Process Algebra (observational bisimilarity), gives us the possibility of measuring ΔQ by abstracting away from the implementation details of a system and by looking only at observation points, called *observables*. Choosing particular observables is equivalent to choosing a particular abstraction level, hence the same concepts work for e.g. a cross continental network as they do for a home network.

We then focused on the particular case of data networks for which ΔQ can be de-composed into three contributors: G, S and V. These components basically capture the factors which contribute

to the total ΔQ, and, at the same time they are independent and manifest some properties for topological convolution. We exemplified how these contributors work and how they are obtained for Ethernet data networks which, as we will see, is the adopted standard for ATLAS TDAQ network.

In the next chapter we will use the concepts introduced above for doing a performance analysis on the ATLAS TDAQ DataFlow network. We will show how the G and S components can be an alternative for assessing network devices performance characteristics. We will then indicate how V offers valuable information for extracting traffic pattern properties and for predicting network behaviour.

Chapter 5

Performance in ATLAS TDAQ Network

5.1 Context

In section 2.5 we introduced the ATLAS TDAQ holistic network as a key component of the TDAQ system. Its purposes range from managing devices to commuting physics data between front-end detectors and processing nodes. The latter is the most important task from the perspective of the ATLAS experiment's main goals and is the most demanding in terms of network performance requirements.

The network in charge of transporting physics related data is called the *Data Network* and encompasses the *DataFlow* and *BackEnd Networks*. Based on the Level-2 role exposed in Section 2.3 and due to the limitation of the mathematical approach exercised in Chapter 3, we will use the Level-2 network (a component of the DataFlow network) as a proof of concept for the observational model introduced in Chapter 4.

5.2 Chapter goals

In this chapter we will quantify the delay and loss in the Level-2 introduced only by the data network and how it varies with the workload placed on the network by the ATLAS TDAQ software. We will obtain three categories of results:

- network devices performance measures - usable as comparison metric
- traffic pattern measures as distances from the deterministic or Poisson pattern - extracted from the ΔQ results for different loading factors of the network
- predictive models for network behavior under increased loads - needed for the next upgrade phase of the ATLAS TDAQ system

In order to apply the concepts introduced above on the ATLAS TDAQ network, an understanding of the supported technologies and the structure of the network is required. We will identify the points in the system where network traffic can be captured and decide on the observables we will use to measure the Level-2 network's ΔQ.

5.3 Ethernet standard

The network of the ATLAS TDAQ system (described in Section 2.5) is an Ethernet based network, standard which "won" the competition with other standards like ATM for multiple reasons: incontestable market penetration, multi-vendor support, exponential evolution of speed (10M $\rightarrow$ 100M $\rightarrow$ 1G $\rightarrow$ 10G $\rightarrow$ 100G) and low cost per port [18]. All these facilitated extensive research and testing of multi-vendor Ethernet network equipment in order to identify those which best meet the ATLAS TDAQ requirements (see [58]).

Ethernet is a widely known standard in computer networks which has become the "de facto" standard in Local Area Networks. Extensive documentation describing Ethernet is available, hence what we want to mention here are the aspects which influence the performance analysis methodology when applied to this type of networks. The key aspect we are following is the delay introduced by all the components and mechanisms specific to Ethernet.

5.3.1 Evolution

Ethernet was invented in 1973 in Xerox PARC laboratories to address the communication between hundreds of computer located in the same building and the world's first laser printer. The name Ethernet comes from its initial description as a communication technology across different "ethers": cable, telephone, radio waves. Although the 10 Mbps Ethernet running on coaxial cable was developed in 1978, its standardization occurred only in 1983 under the name IEEE 802.3 [15]. After this moment Ethernet witnessed an exponential growth in terms of speed and pervasiveness.

Table 5.1 summarizes the key milestones in its evolution in order to understand in which historical point the ATLAS TDAQ network was designed and what technologies were available. One of the key moment in the evolution was the standardization of FastEthernet full-duplex, which changed the way access to the medium is performed, removing the highly non-deterministic half-duplex access.

At the time this network was designed (2002-2005) the highest standardized speed for Ethernet was 10 Gigabit over optical fibers. Consequently, the connectivity is implemented using 1 Gigabit copper links and 10 Gigabit optical fibers. The current upgrade stage considers 100 Gigabit optical fibers and 10 Gigabit copper connections, which is a big step forward in terms of decreasing costs, increasing performance and simplifying the network architecture.

Year	Milestone	Organization/Company
1973	Ethernet was invented	Xerox
1976	2.94 Mbps Ethernet network was built	Xerox
1978	X-wire was developed: 10 Mbps Ethernet	Xerox
1980	DIX Ethernet released the 10 Mbps standard	DIX
1981	Started building Ethernet components	Interlan, Bridge Communications
1983	802.3 specification formally approved	IEEE
1995	FastEthernet was standardized: 100 Mbps	IEEE
1997	FastEthernet full-duplex was standardized	IEEE
1998	1 Gigabit Ethernet was released	IEEE
2002	10 Gigabit Ethernet was released	IEEE
2010	802.3ba standardized: 40/100 Gigabit	IEEE
2011	802.3bg standardized: 40/100 Gigabit	IEEE

Table 5.1: Ethernet evolution

5.3.1.1 Physical layer

The initial Ethernet implementations used coaxial cables to interconnect devices. A newer form of Ethernet became more popular as it used twisted pair wiring instead of coaxial cable, followed by the development of optical fibers. Medium evolution with the Ethernet speed as well as the time required to place a bit on the wire for each medium type is mentioned in Table 5.2.

The propagation delay introduced by the media type is:

- between 4.32 and 5.64 ns/m for copper. This results from the propagation speed which ranges from 0.59c to 0.77c, where $c \simeq 3 \times 10^8 m/s$ is the speed of light.

- 5 ns/m for optical fiber. This results from the propagation speed in silica glass: $v = c/n = 2 \times 10^8 m/s$, where *n* is the refractive index and is usually around 1.5.

We will consider these values for the purpose of quantifying the delay introduced by the network connections. In the case of ATLAS TDAQ network, the distances range from few meters to hundreds of meters.

Medium	Speed	Delay/bit
10BASE-5 10BASE-2 10BASE-T	10M	100ns
100BASE-T4 100BASE-TX 100BASE-FX	100M	10ns
1000BASE-LX 1000BASE-SX 1000BASE-CX 1000BASE-T	1G	1ns
10GBASE-SR10 10GBASE-LR 10GBASE-LRM 10GBASE-CX4	10G	0.1ns
100GBASE-SR10	100G	0.01ns

Table 5.2: Evolution of medium speed and delay/bit

5.3.2 Ethernet Frame

The IEEE 802.3 Ethernet frame structure is detailed in [56, section 3.2]. Of interest for our measurements is the mechanism for sending an Ethernet frame on the wire and the range of allowed frame sizes.

Before the actual data frame a preamble is sent, containing seven bytes each containing the sequence 10101010 followed by one byte called *Start Frame Delimiter (SFD)* containing the sequence 10101011. After the data frame a minimum *Inter-Frame Gap (IFG)* of 96 bits (12 bytes) is sent. Each Ethernet frame is thus accompanied by 20 additional bytes.

Instance	Size contributors	Total size interval (bytes)
Ethernet (no VLAN)	preamble + SFD + Ethernet header + Payload + IFG	[84, 1538]
Ethernet (VLAN)	preamble + SFD + Ethernet header (VLAN) + Payload + IFG	[84, 1542]
ΔQ measurement (VLAN)	Payload	[42, 1500]

Table 5.3: Ethernet frame sizes

Moreover, in the case of the TDAQ network the traffic separation is implemented using VLAN tagging (standard 802.1q [14]) which adds 4 bytes to each Ethernet frame. The minimum size

however is decreased with 4 bytes, hence the frame size allowed interval becomes [64, 1522] bytes. Adding to this the preamble, the SFD and the IFG we obtain the complete Ethernet frame sizes with VLAN ranging from 84 to 1542 bytes.

With respect to the G, S and V components of the ΔQ we incorporate the preamble, the SFD, the IFG, the Ethernet header (18 bytes - including the VLAN tag) and the checksum into G because it's a fixed overhead for the Ethernet implementation of a data network. S will thus be a measure dependent on a size ranging from 42 to 1500 bytes. Table 5.3 summarizes the Ethernet frame sizes in all the situations exposed above.

5.4 Structure

5.4.1 Network elements

The current architecture of the data network consists of two hierarchical layers:

- the **access/concentrator** layer, implemented using small switches, called *Pizza-box*
- the **core/central** layer, implemented using *Chassis based* switches.

The connectivity between these layers is done through 10G optical uplinks or 2 x 1G copper trunks. The ATLAS TDAQ Data Network block diagram emphasizing the main sub-systems is depicted in Figure 5.1:

- the DataFlow Network, comprising of:
  - the ROS Concentrators: pizza-box switches, typically 24 x 1G copper ports and 2 x 10G optical ports
  - the Level-2 & Level-3 (XPU) Concentrators: pizza-box switches, typically 48 x 1G copper ports and 2 x 10G optical ports
  - the Data Core switches: chassis based switches
- the BackEnd Network, incorporating:
  - the Level-2 & Level-3 (XPU) Concentrators, shared with the DataFlow Network
  - the BackEnd Core switches: chassis based switches

5.4.2 Connectivity & redundancy

From the point of view of the latency introduced by the medium, it is important to mention the cable lengths in ATLAS TDAQ network, because each meter of copper or optical fiber accounts for approximately $5\,ns$ delay, as detailed in Section 5.3.1.1. The lengths of the links between:

- ROS / XPU computers and their concentrator switches range from $1\,m$ to $10\,m$.
- DFM / pROS / L2SV / SFI /SFO computers and the core switches range from $8\,m$ to $15\,m$.
- XPU concentrators and core switches range from $8\,m$ to $30\,m$.
- ROS concentrators and the core switches range from $150\,m$ to $200\,m$. These long distances are due to underground hosting (closer to the ATLAS detector) of the ROS computers together with their concentrator layer while the rest of the acquisition system is hosted at the surface.
- the concentrator switches are $10\,m$.

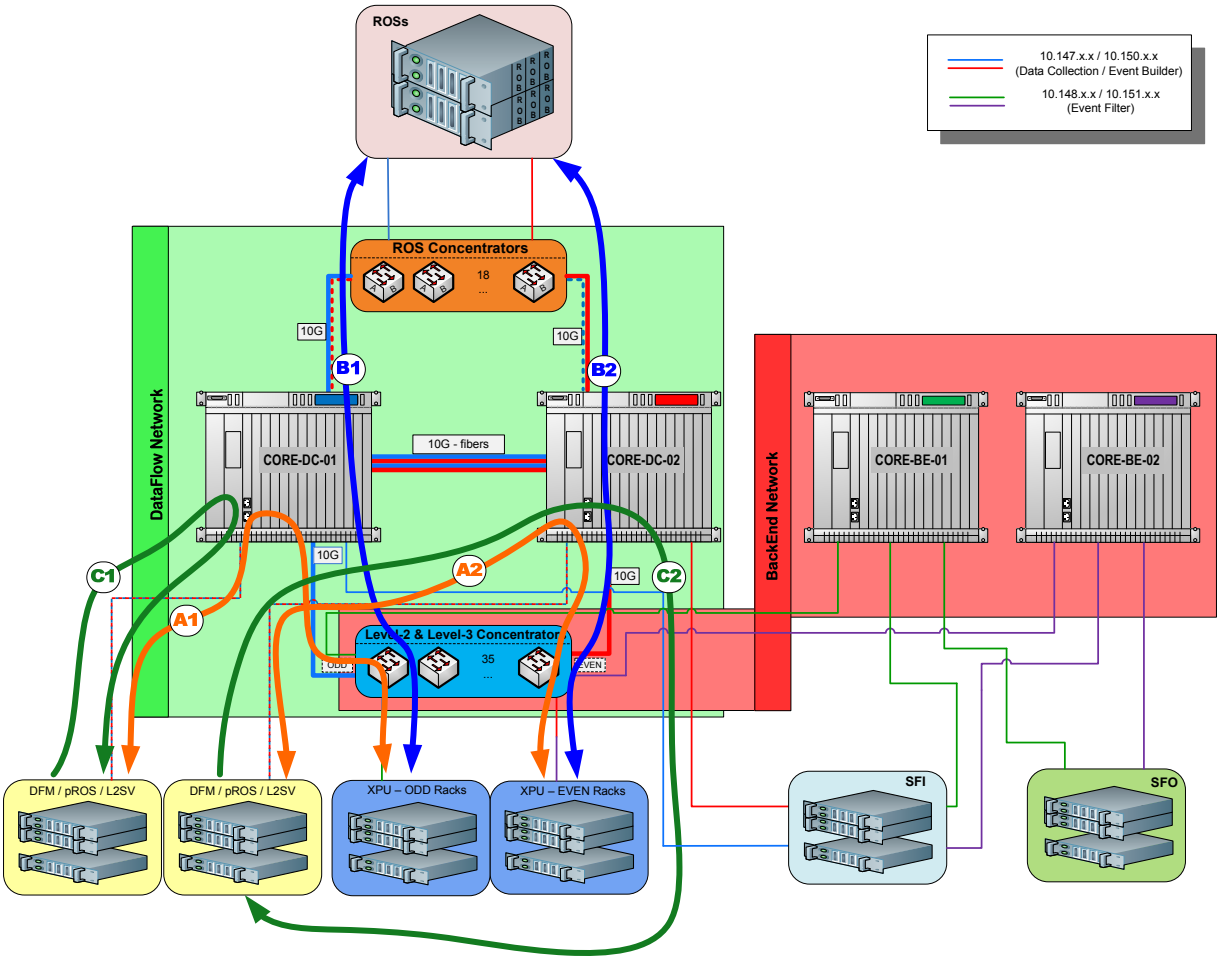

Figure 5.1: ATLAS TDAQ Network - Level-2 traffic paths

These values indicate that the propagation delays introduced by the cables are of the order of few nanoseconds ($\mathcal{O}(10^{-9})\,s$) up to one microsecond ($\mathcal{O}(10^{-6})\,s$).

The network has been designed to offer redundant connectivity to the ROS computers in case one of the DataFlow core switches or ROS concentrator switches fails. The redundancy is achieved by connecting each ROS with two links going to different ROS concentrator switches (A, B) and by inter-connecting the DataFlow core switches with four 10G fiber links. The classes of IP addresses carried by each link are visible in Figure 5.1.

5.4.3 The Level-2 network paths

In the Level-2 system the communication concerning physics data and algorithms takes place between (see Figure 5.1 for paths notation):

- *L2SVs* and *L2PUs*, both ways - A1/A2
- *L2PUs* and *ROSs*, both ways - B1/B2
- *L2PU* and *pROS*, one way - A1/A2
- *L2SVs* and *DFMs*, one way - C1/C2

Figure 5.1 indicates that all this traffic is over the DataFlow network. The devices supporting it are: the Level-2 concentrator switches, the Core-DC switches and the ROS concentrator switches.

The most demanding traffic from the sequence above is the L2PUs↔ROSs traffic. It is generated by the physics algorithms running on L2PUs which iteratively ask for event data from the ROBs until a decision is made on the event of interest (see Section 2.3). The estimated bandwidth required on this communication path is ∼3.2 GB/s by design[1].

Moreover, the total Level-2 processing time, involving up to three L2PU↔ROS iterations, is of the order of tens of milliseconds, as shown by the mathematical model in Chapter 3, Table 3.1. This is the only network delay requirement we currently have and is in fact an upper bound on the L2PU-ROS delay.

5.5 Performance assessment at design time

Prior to the implementation phase of the current network, extensive research and testing of multi-vendor Ethernet network equipment was carried on in the ATLAS TDAQ Networking group. This effort was performed in order to identify the network equipment who best meets the ATLAS TDAQ requirements mentioned in [58] by looking at performance indicators such as throughput, loss, latency across the device and its sensitivity with the packet size.

[1] ~2 GB/s for a 65 KHz observed Level-1 rate to which we add ~100MB/s, i.e. the L2 Etmiss data explained in Section 2.3

The method involved the use of dedicated network traffic generators (called GETB) which could be installed as PCI boards, providing accurate time-stamping for the boards installed on the same computer. The design of the GETB and the procedures for testing the devices are thoroughly described in [11, 10].

The testing of the devices was performed in laboratory conditions with traffic injected by the GETB. Poisson was chosen as traffic pattern, i.e. exponential inter-packet gap (IPG) in order to simulate the real traffic conditions. Poisson traffic is widely used for analytical models as a hypothesis when the number of sources generating the traffic is large enough. The frame sizes were chosen with respect to the recommendation in [7] but also a few additional values, the complete set being {64, 65, 135, 512, 1027 and 1518} bytes.

The advantages of this testing approach are: very precise delay measurements, automatic testing capabilities, flexible traffic patterns, 1 Gigabit line speed and PCI standard compliant. Based on the results of the tests performed with the GETB the current network devices were acquired.

Disadvantages of this approach were: the costs given by the design of a dedicated board, the set-up of the testing environment which required a big number of GETB boards and the difficulty of using them in the operational phase for performing the Assurance step introduced in Section 4.2.3.

In the next sections we will expose the differences concerning the performance analysis between the GETB and the Observational Model and where the two can become complementary.

5.6 Observational model implementation

This section presents the implementation details of the observational model introduced in Chapter 4 on the ATLAS TDAQ Network: network paths, observation points, tools to measure ΔQ and the tools to analyze the obtained results. The measurements were performed on the network in an operational phase, i.e. with load created by real traffic from physics experiments.

5.6.1 Measurement

We employed the One-Way Delay (OWD) approach (detailed in Section 4.4.1.2) for ΔQ measurements because the traffic in ATLAS TDAQ is highly asymmetric. The L2PU-ROS traffic for example is dominated by the physics data traveling from the ROS to L2PU. Considering that traffic asymmetry also occurs heavily in the Internet traffic (see [44]), going for the OWD approach makes this method applicable in a much wider context.

Measuring the OWD however requires a very precise clock synchronization between the observation points, in our case PCs running Linux Operating System (CERN compiled version named: *Scientific Linux for CERN - SLC* [54]). On these nodes the clock synchronization is achieved using the *Network Time Protocol (NTP) [42].* For the scale of delay values we are measuring NTP is not accurate enough, yielding offset values of the order of hundreds of microseconds

(10^{-4} seconds). To overcome this issue we use a clock correction algorithm detailed in Appendix B.2.

5.6.1.1 Generating probes

For generating frames with a certain size - called *probes* - we used the *ping* Linux tool [45] which accepts as argument the size of the *Internet Control Message Protocol (ICMP)* [29] packet payload. For example, in order to obtain an Ethernet frame of 84 bytes on the wire we give ping the argument:

$$s = 84 - Ethernet_{Header} - IP_{Header} - ICMP_{Header} = 84 - (20 + 18) - 20 - 8 = 18 \quad (5.1)$$

For an A to B communication path *ping* is executed on A, having B's IP address as argument for destination.

A typical measurement run involves an automated execution of the methodology explained in Section 4.4.5. We used the random number generators from Python [48] to obtain the uniform distribution for sizes and negative exponential distribution for inter-probe gap. Appendix B.3 provides details on the generation of pseudo-random numbers. The number of samples in a run is 4000 and the average time interval between two consecutive probes is 25 ms.

5.6.1.2 Capturing observables

We use *tcpdump* [59] Linux tool as packet capturing tool ran on both end-nodes simultaneously. The time-stamping required by the delay computation is offered by tcpdump itself. The timing information in tcpdump is obtained from the *ioctl SIOCGSTAMP* system call which allows getting the time the packet was received by the network interface card and not by the application layer. The results are saved in the *pcap* file format provided by the *libpcap [59]* library in Linux - a widely used format for packet captures.

At the operating system level, accurate time-stamping is supported by the use of *Time Stamp Counter (TSC)* register[2]. In addition, recent generation of processors are capable of reading this register at a constant rate given by the processor's maximum rate[3]. The currently installed processors are running at 2.4 GHz, hence the reads are performed at intervals as small as ~0.42 nanoseconds.

[2]the Linux kernel configuration parameter is called CONFIG_X86_TSC and is enabled for all the computers used as observation points

[3]the setting is called *constant_tsc* and for Linux it can be found in */proc/cpuinfo*

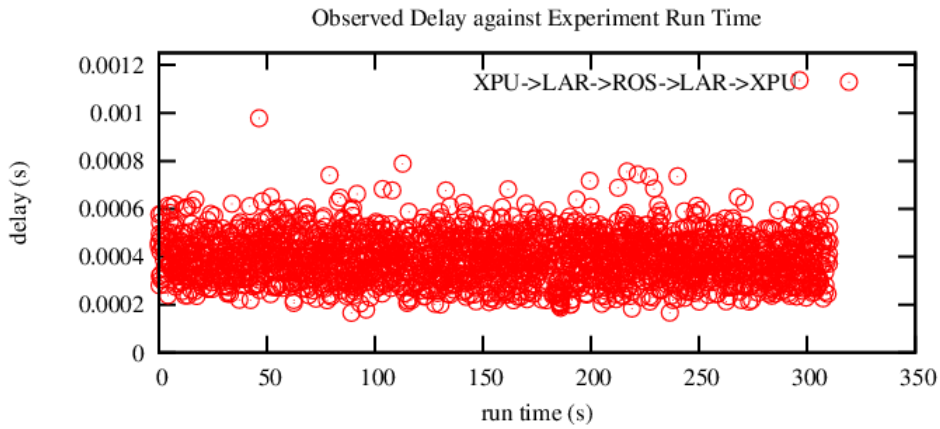

Figure 5.2: Same clock rate on the end-nodes

5.6.2 Computation and source of errors

The obtained captures are then processed by a tool written in Haskell [28] and provided by PNSol [46]. This tool computes the delay and loss percentage for a measurement run between configured observables. From the entire set of samples it then obtains the ΔQ and its components: G, S and V for separately for forward and return directions of the measured path.

The One-Way Delay is computed within bounded error limits by employing a clock correction algorithm detailed in Appendix B.2. The algorithm is based on the big number of samples offered to it and assuming the linear model mentioned in Section 4.4.5 (relation 4.27).

An important source for errors still remains the clock synchronization because during a measurement run the ambient conditions (i.e. temperature) change and so do the clock frequencies on the two end-nodes. A typical measurement run is of the order of minutes, time which we discovered is enough for the end-nodes clocks to drift apart. This affects the delay measurements because we use local clocks to time-stamp the packets at each end-node. Between two nodes the clock has an *offset* which is corrected by the algorithm mentioned earlier, but also a variable *clock rate.*

Looking at the measured delays as time series we spotted linear and non-linear variations of the clock rate. Ideally, the two end-node's clocks should have the same rate, case in which the time series looks like the one depicted in Figure 5.2. In practice, many measurements suffered from clock rate variations, i.e. clock drift between the end-nodes.

Figure 5.3 depicts some of the encountered situations: positive and negative linear and non-linear. Of particular interest is the bottom left plot which captures the moment when a data taking run starts and the clocks start drifting. One of the likely cause of this is the change in temperature inside the computer's chassis due to CPU intensive operations, change which can be up to 40 degrees Celsius. Oscillator's frequency, which gives the clock rate is known to be dependent on the temperature. Furthermore, the ATLAS TDAQ computers come from different manufacturers and different generations, hence oscillators come in a wide variety.

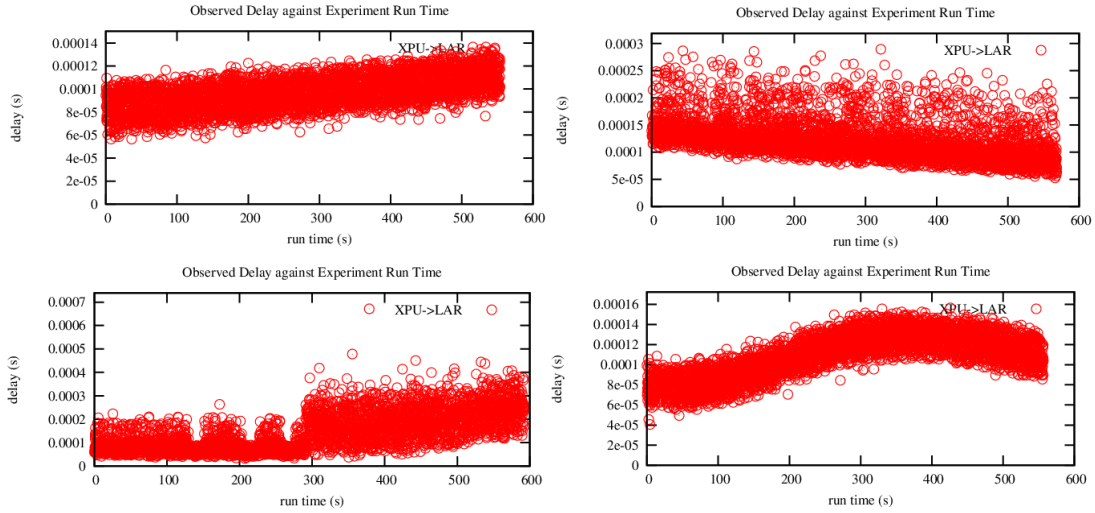

Figure 5.3: Clock drift detected between end-nodes

As seen in Figure 5.3, a variation of tens of microseconds is possible during a run. In a statistically independent manner, it affects the minimum delay points based on which G and S are calculated using the methodology described in 4.4.5. In order to cancel this effect we perform many measurements sessions discarding 10% of the results which are furthest from the mean and considering the average values for G and S.

5.7 Structural Delay

The Structural Delay (SD) of a network is given by the convolution between G and S components, as defined in relation 4.26. G and S can be extracted from ΔQ by employing the methodology described in Section 4.4.5. The SD characteristic can be used as a comparison metric between two networks from the *immutable* properties point of view, e.g. access to the medium, medium propagation delays, serialization/de-serialization delays, switching speed.

Due to the topological composition property of G and S expressed in the equations 4.24 and 4.25, we obtain G and S for segments of the network by arithmetic subtraction, provided that any two measurements from the set $\{G_A, G_B, G_{AB}\}$ are known. This can be achieved by measuring two network paths which contain a common segment.

Then we can apply relation 4.26 to calculate SD for individual network segments. The finest granularity for the network segments from the point of view of the observational model implementation exposed in Section 5.6 is given by the network devices.

We will use different network paths in order to obtain SD for a single network device. Then we will apply SD metric for comparing two core switches installed in the ATLAS TDAQ network in different implementation phases. They are made by different manufacturers, the newest one

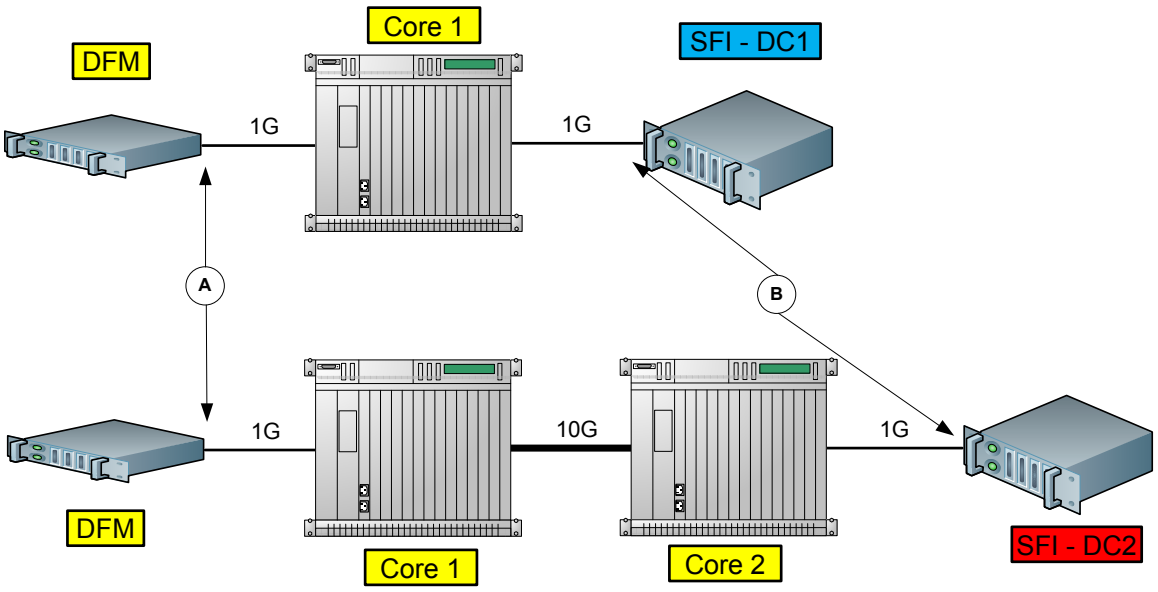

Figure 5.4: Network paths for core switch type 1

being a candidate for the coming upgrade of the network, therefore we will see if the newer device performs better or worse.

5.7.1 Differential paths approach

We will identify pairs of observation points, i.e. network interfaces in the ATLAS TDAQ network which communicate between them on paths sharing certain network segments. By measuring G and S for these paths we are able to extract the SD property for the " network difference".

We applied this method for the purpose of obtaining the SD characteristics for all the network devices in the ATLAS TDAQ network, depicted in Figure 5.1: two types of core switches (first type represented by: CORE-DC-01/CORE-DC-02/CORE-BE-01 and second type by: CORE-BE-02) and one type of concentrator switches (Level-2/Level-3/ROS).

5.7.1.1 Core switch

For the first core switch type we identified the paths in Figure 5.4 between DFM and SFI computers. The difference between the two paths is given by a core switch plus an optical fiber few meters long. The paths separation was possible due to the dual network configuration of the SFI nodes (known as DC1 and DC2) and due to the configuration of DFM interface with VLAN logical interfaces, having DC1 and DC2 networks overlaid on the same physical interface.

Probe packets were generated on the DFM node and the observables at both end points (noted with A and B) were recorded. The measurement consisted in more than 50 runs, each com-

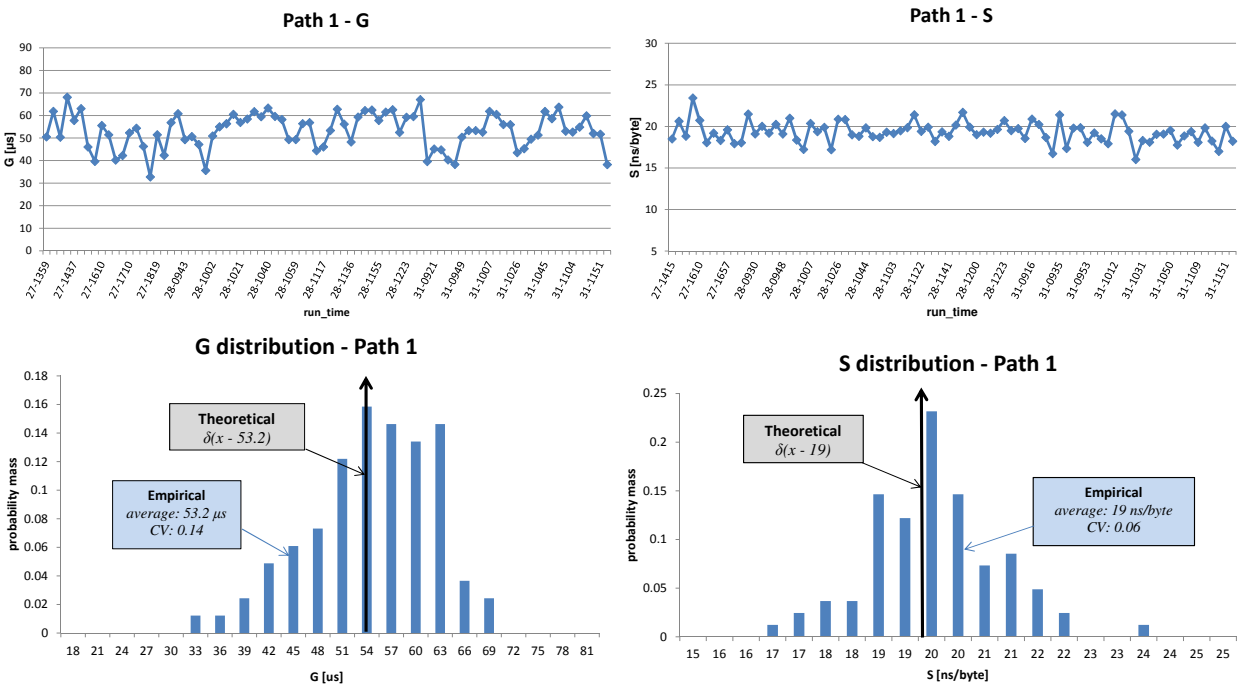

Figure 5.5: G and S for the first path

prising 4000 probe packets, as detailed above in Section 5.6.1. No loss occurred during our measurements.

In Figure 5.5 we present the G and S obtained for the first path, comprising only one core switch. The plots at the top show the G and S evolution with run time (following the pattern *day_time*), indicating that they are stationary but not perfectly constant. We stated that, theoretically, G and S are constants for a network, however this measurement is prone to errors for reasons detailed in Section 5.6.2.

Plots at the bottom show the corresponding distributions obtained through measurements (empirical) and the theoretical ones (Dirac delta functions shifted with the empirical average values). We can notice that the coefficients of variation (CV) are small: 0.14 for G and 0.06 for S, meaning that the errors introduced by measurement have a small magnitude on the two variables.

In Figure 5.6 we present the similar results for the second path, comprising two core switches. As expected we observe an increase in both G and S, due to the longer network path.

Table 5.4 summarizes the G and S values and shows the differences between the two paths. The increase in G and S for this longer path is assigned to the second core switch and to the optical fiber connecting the two core switches. The inter-core fiber is a few meters long. The propagation delay is 5 ns/m for a 10G fiber, thus yielding a G for the fiber of the order of tens of nanoseconds ($\mathcal{O}(10^{-8})\ s$) , which can be ignored when working with a microseconds scale ($\mathcal{O}(10^{-6})\ s$).

The Structural Delay for a core switch is:

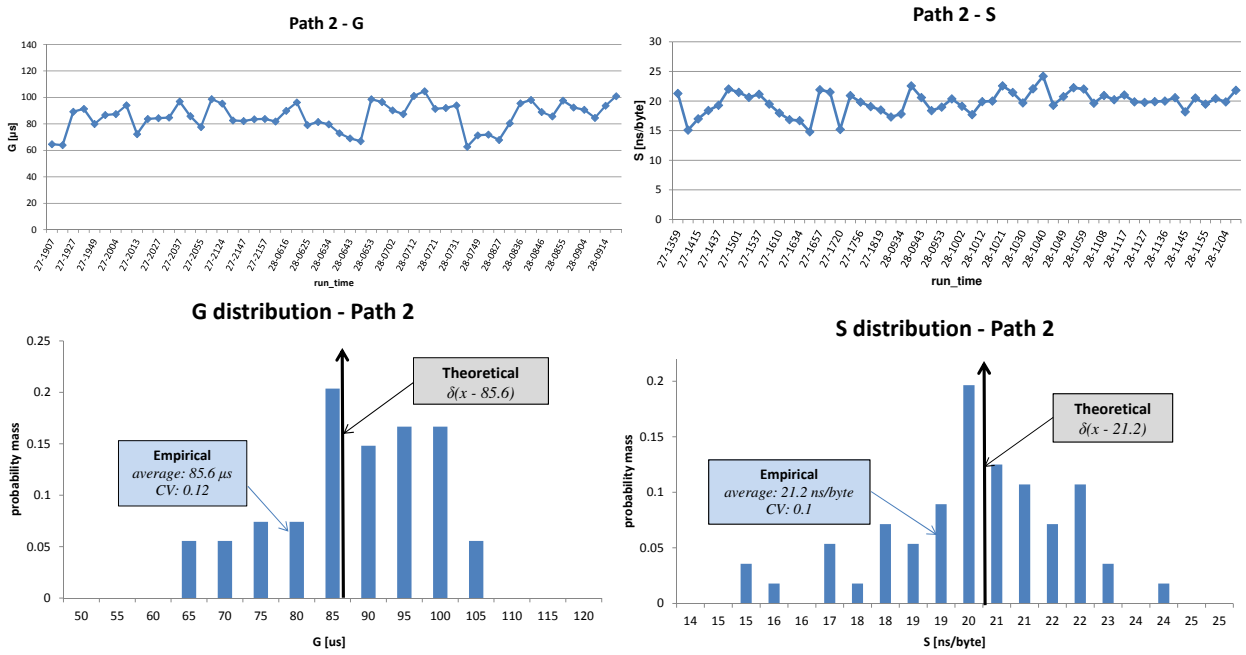

Figure 5.6: G and S for the second path

$$SD(size) = 32.4 + 2.2 \times 10^{-3} \times size[bytes]\ \mu s \tag{5.2}$$

Network segment	G [μs]	S[ns/byte]
Path 1	53.2	19
Path 2	85.6	21.2
Path 2 - Path 1 Core type 1	32.4	2.2

Table 5.4: G and S values with differences between paths

A verification of this result with the values obtained by the performance assessment techniques mentioned in Section 5.5 cannot be made too precisely because the latter are calculated as average values while we look only at minimum values. Figure 5.7 displays the latency results obtained using this alternative method, mentioning that each point in the scattered plot is an average value. We observe that a linear model approximating the minimum values in this plot would generate a G between 20 and 30 μs, compared to 32 μs what we obtained.

The same exercise for S yields values 5-6 times larger than what we obtained. We consider a few of explanations for this difference. One is the fact that we compare two type of results: averages against minimums. The second is the extremely precise scale of measurements required for S, which is of the order of nanoseconds. For our measurements we use a clock correction

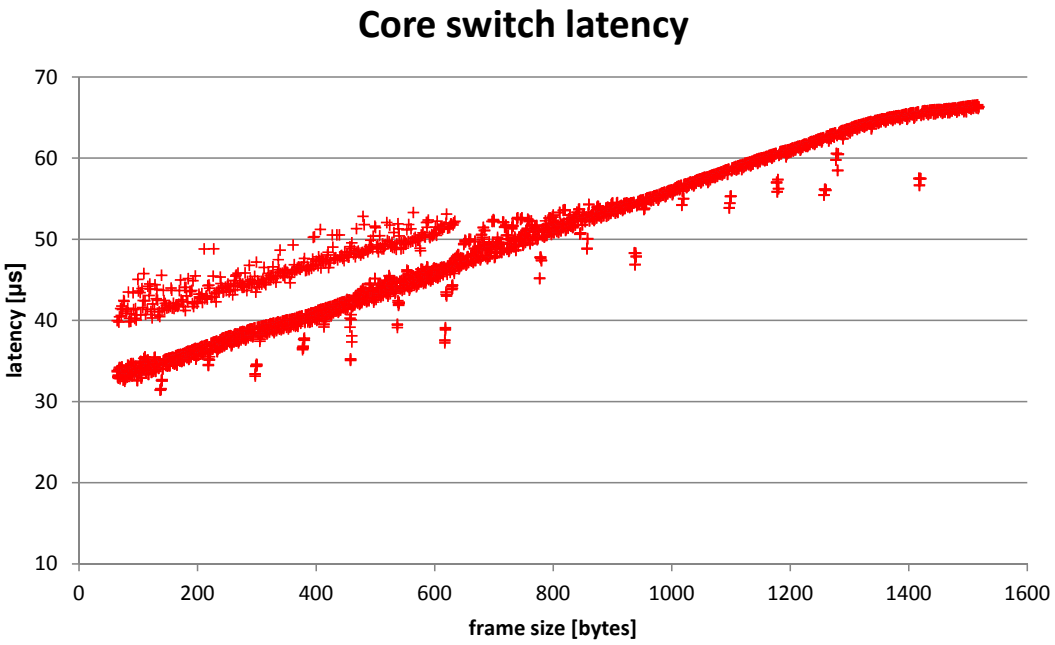

Figure 5.7: Core switch latency measured using a different method

algorithm working as an offset compensation between the two observation points, but as we will see in Section 5.6.2 the clocks also drift in time. These drifts can cause errors of the order of microseconds, one order of magnitude higher than the measurement scale for S.

The last possible explanation is that the performance measurements performed in the design stage were executed on an equipment not completely identical with the one measured in operation (i.e. the same architecture but a lower port density, hence a different switching fabric - see [56, section 3.3] for architectural details). We additionaly mention the different software and firmware versions which have undergone several upgrades since the lab testing was performed. This aspect is very likely to have improved the performance of the operational network device compared to the one that was tested - and this is what we are observing.

5.7.1.2 Core switches - different models

Another possible application of the SD metric is to compare two or more switch types. This is important when purchase decisions have to be made in the design stage.

For comparing two core switch models, one installed in the network at a later stage, we identified the paths in Figure 5.8: EF1 and EF2.

Applying the same methodology as above, we obtained the results in Figure 5.9 for the EF1 path and in Figure 5.10 for the EF2 path. The differences between the two paths - Δ_G and Δ_S - yield the differences between the two core switch types. Moreover, knowing the SD for the first switch type (relation 5.2) we can deduct the SD for the second type:

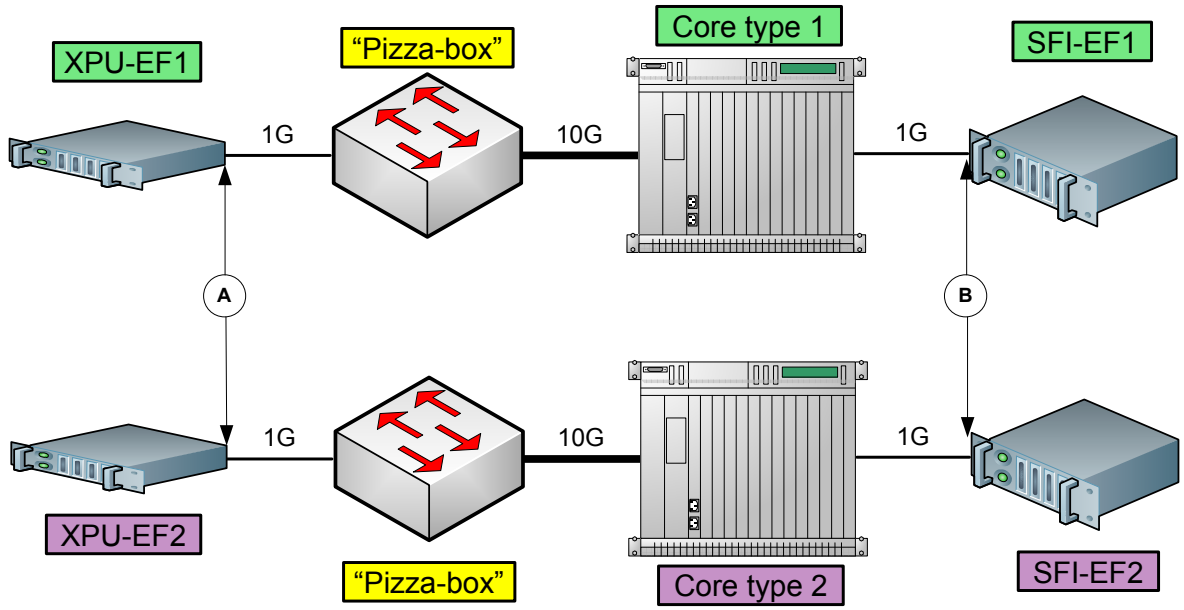

Figure 5.8: Network paths differential for core switch types 1 & 2

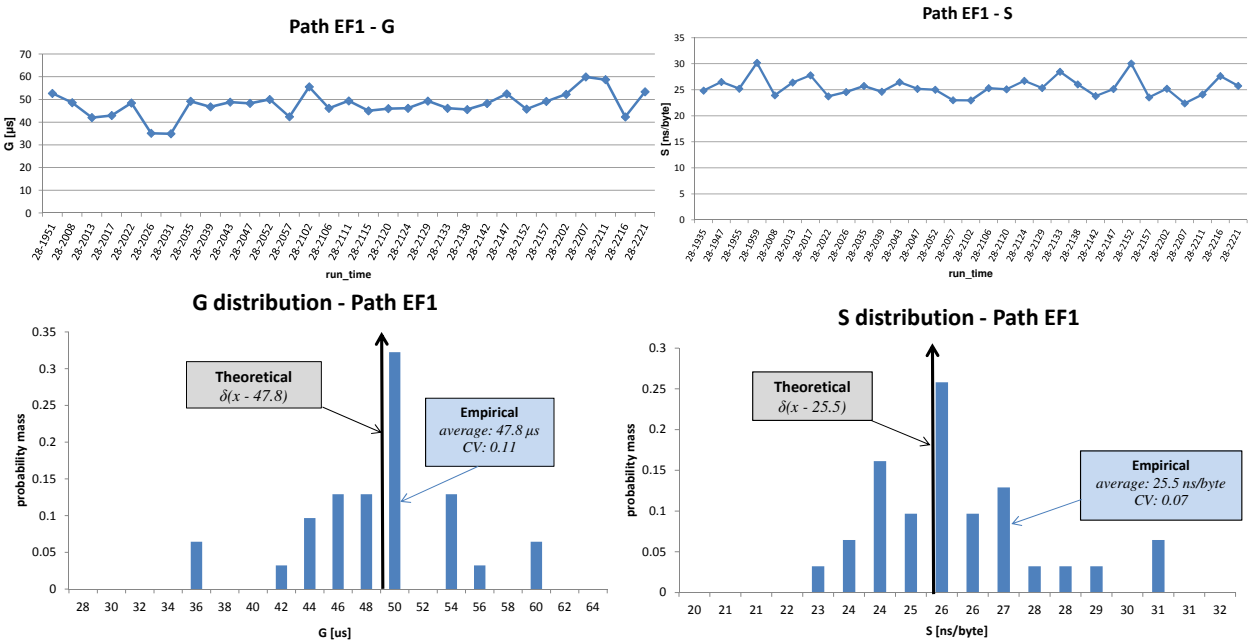

Figure 5.9: G and S for the EF1 path

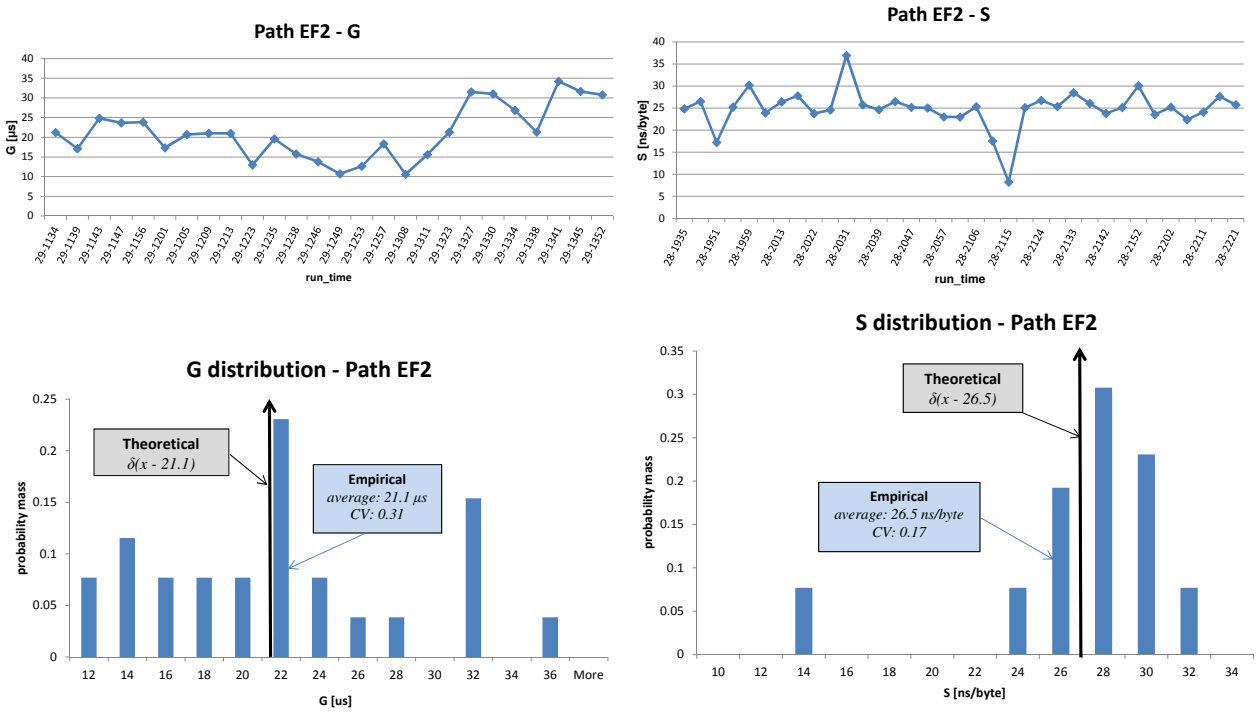

Figure 5.10: G and S for the EF2 path

$$G_{type2} = G_{type1} - \Delta_G = 5.7\mu s \tag{5.3}$$

$$S_{type2} = S_{type1} - \Delta S = 3.2\frac{ns}{byte} \tag{5.4}$$

$$SD(size) = 5.7 + 3.2 \times 10^{-3} \times size[bytes]\ \mu s \tag{5.5}$$

The results are summarized in Table 5.5. The conclusion is that the newer switch type has a better G, but a slightly worse S that the first switch type. Figure 5.11 shows the difference in the delay introduced by the SD for the two switch types computed for the entire range of frame sizes. Although S is slightly better for core type 1 the big difference in G is not canceled even for large frames, hence for the entire range of Ethernet frame sizes the second switch is better from the performance point of view. For the maximum frame size, the core type 2 is still faster with 25.2 μs than core type 1.

From the point of view of the envisaged upgrade, our performance assessment validates the decision to move to a new generation of switches in the core part of the network.

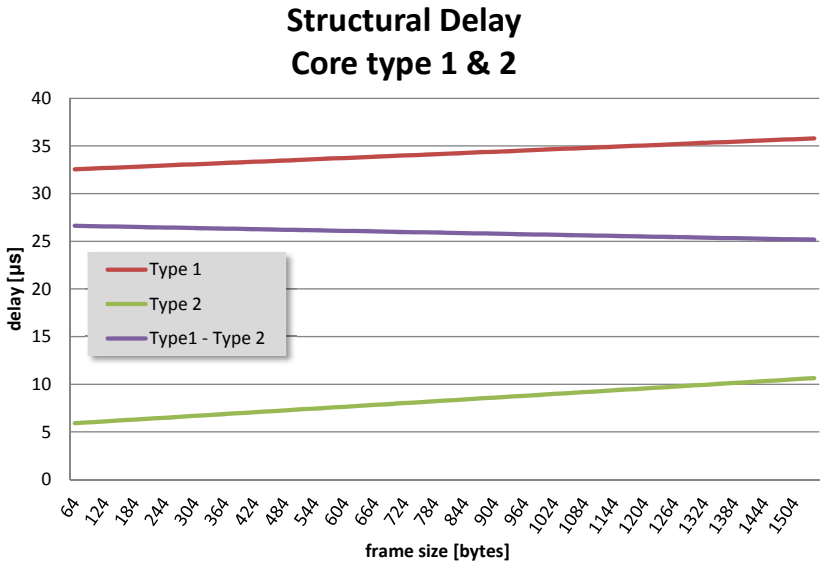

Figure 5.11: Structural Delay with the frame size

Network segment	G [μs]	S[ns/byte]
Path 1	47.8	25.5
Path 2	21.1	26.5
Path 1 - Path 2 (δ_G, δ_S)	26.7	-1.0
Switch type 2 (G_{type2}, S_{type2})	5.7	3.2

Table 5.5: G and S values - switch type 2

5.7.1.3 Concentrator switch

The same type of switch has been installed to perform traffic concentration for: ROSs, Level-2 and Level-3 computers. Since the purpose of this PhD is the description of the Level-2 network, we need to extract the performance characteristics for all its elements.

In order to "subtract" a concentrator switch, we identified two paths in the same network (DC1) depicted in Figure 5.12.

The differential contains also an optical fiber which in this case is ten meters long. Given the propagation delay of 5 ns/m we obtain a G for the fiber of the order of 50 ns which can still be ignored when working with a microseconds scale ($\mathcal{O}(10^{-6})\ s$).

Given the similarity with the differential approach to obtain the SD for core switches we present only the summary results in Table 5.6. The SD for a concentrator switch is:

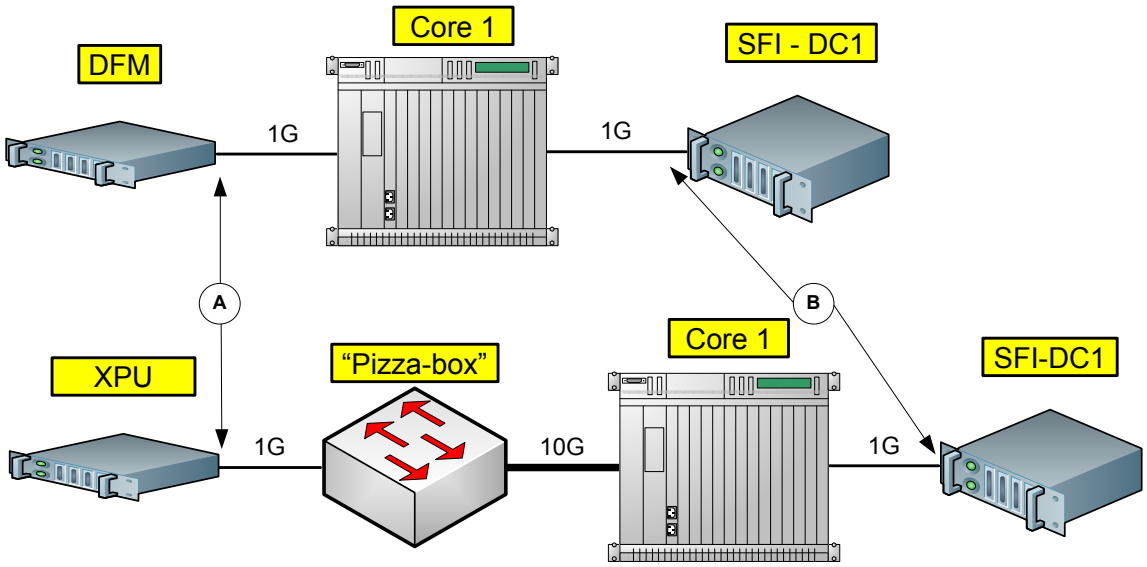

Figure 5.12: Differential network paths for concentrator switch

$$SD(size) = 8.8 + 7.2 \times 10^{-3} \times size[bytes]\ \mu s \tag{5.6}$$

Network segment	G [μs]	S[ns/byte]
Path 1	51.2	18.8
Path 2	60	26
Concentrator switch	8.8	7.2

Table 5.6: G and S values for concentrator switches

The measurements obtained through the alternative method presented in Section 5.5, keeping the same observation as above concerning the differences to our method, indicate the following values: $G = 6.6\mu s$ and $S = 7ns/byte$. We consider these values to be close to our results in acceptable ranges given the differences in approach (averages vs. minimums) but also given the next aspect which could account for the larger G we obtained. These results are for latencies between modules of the concentrator switch in two slightly different situations:

- for the alternative method between two copper ports placed on different switch modules
- for our method between one copper port and one optical fiber port (see Figure 5.12).

5.7.2 Level-2 network

We obtained until now the structural delay characteristic for all the switches involved in the Level-2 network. The most demanding traffic in this network is caused by the L2PU (XPU) - ROS communication. As it can be seen in Figure 5.1 this path is made up by two concentrator switches and one core switch plus the communication links. Figure 5.13 shows this path with the SD characteristics for each network device.

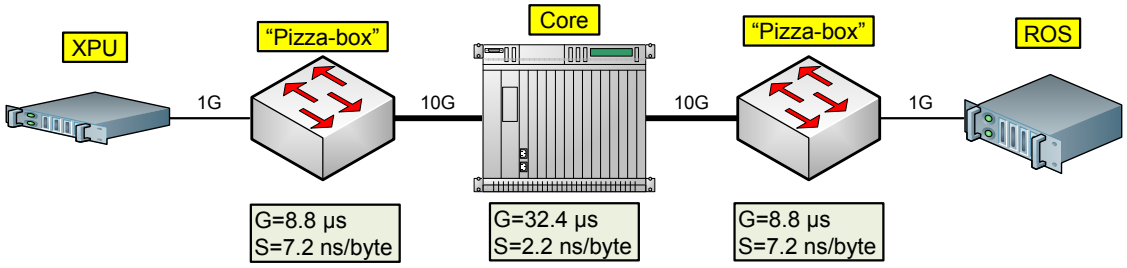

Figure 5.13: Level-2 network path: XPU-ROS

Knowing that topological convolution works in a component-wise manner, we can obtain the SD for the entire path. Adding the G for the links, which in this case are of order of hundreds of meters[4], yielding a few microseconds, we obtain:

$$G = 2 \times G_{concentrator} + G_{core} + G_{links} = 51\mu s \tag{5.7}$$

$$S = 2 \times S_{concentrator} + S_{core} = 16.6\frac{ns}{byte} \tag{5.8}$$

$$SD(size) = 51 + 16.6 \times 10^{-3} \times size[bytes]\ \mu s \tag{5.9}$$

In our scenario we were also able to measure the end-to-end path directly. We are thus able to cross-check the results calculated in a component-wise manner with those directly measured. The obtained SD for the XPU - ROS path was:

$$SD(size) = 51 + 20 \times 10^{-3} \times size[bytes]\ \mu s \tag{5.10}$$

meaning a perfect match for G and a difference of $3.4\,ns/byte$ for S. The error between the two approaches is: 0% for G and 19.7% for S. The topology convolution is thus verified for G while for S we consider the bigger error to be caused by the measurement scale factor mentioned in Section 5.6.

[4]due to the ROS computers which are underground in the detector cavern

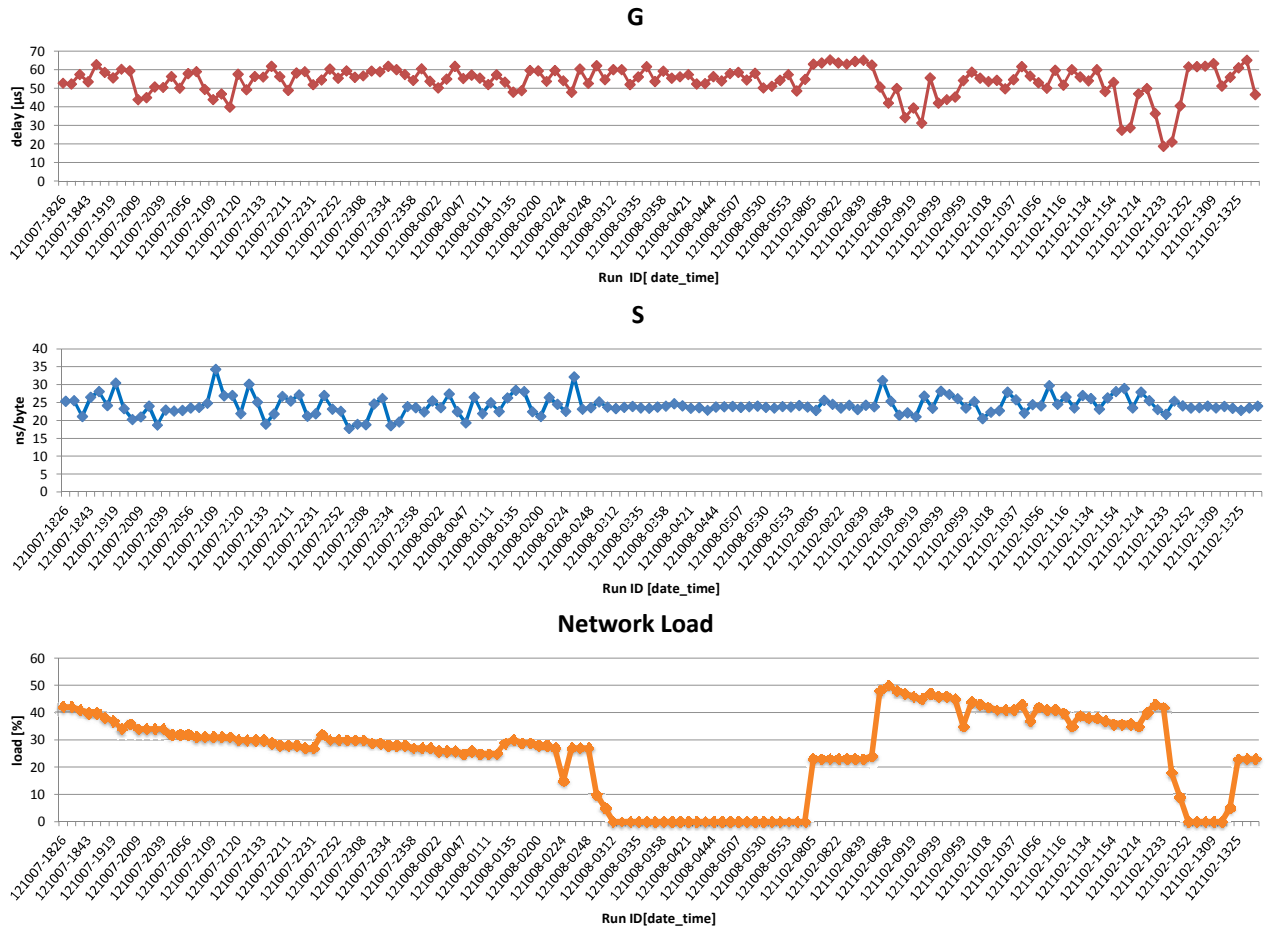

Figure 5.14: G, S and the network load

5.7.3 Non-dependency on load

When we introduced the structural delay concept in Section 4.4.2, we stated that the G and S components are dependent on "structural" factors leaving the network load factor to V. We implicitly stated that the G and S do not depend on the load. We experimentally proved that this statement is true measuring G and S on the ROS-XPU path at different loading factors.

Figure 5.14 shows the evolution of G and S with measurement run time along with the network load extracted for the moments when measurements took place. One can notice that the G and S trend was not affected by the network load variations, the only effect being on their variance. The explanation is that, as system gets more loaded, the precision of the measurements decreases due to the time-stamping of the observables, aspect analyzed in Section 5.6.2.

5.8 V - load dependent performance indicator

We mentioned in Section 4.4.2 that one important reason for decomposing ΔQ into immutable and mutable components was to separate the fixed contribution to the ΔQ from the rest. The remainder, called V, is a random variable on which queueing theory models can be applied because it contains no fixed delays - the latter are not properly modeled by the queueing theory. The reason why we need these theoretical models is because V is the effect of the queueing inside a data network. Applying the theoretical formulas for V will allow us to obtain:

- an estimate of the queue occupancy on different loading conditions
- the level of statistical correlation of the input traffic
- the network delay growth with the potential increase of the network load in the same traffic pattern conditions

Using this information we are able to infer on traffic pattern structure and predict its influence on the amount of delay the network will induce for an increased load.

In the particular case of the Level-2 network this translates into:

- the size of the ROS output buffers and of the XPU concentrator switches buffers
- the coefficient of variation for the traffic generated by the ROS applications
- the ROS-XPU delay to be expected in case it is planned to increase the utilization on the network links (e.g. by increasing the event rates).

5.8.1 Network traffic dependency

5.8.1.1 Contention, patterns and queueing

The resources that are competed for in data networks are the physical layer transmission capacities. In the case of Ethernet, Table 5.2 shows the time required by the different medium types to generate a bit of information. For example, a 1 Gigabit link requires 1 ns/bit, hence:

$$T_{1G} = 8 \times 1542 \times 10^{-9} s = 12.336 \times 10^{-6} s \tag{5.11}$$

i.e. ~12 μs for a full size Ethernet frame. In case information is transmitted faster than a physical layer can accept, queueing occurs right before being sent on the medium.

In the example before, if an application sends full size Ethernet frames on a 1 Gigabit link at intervals smaller than 12 μs, the network interface card will buffer them until the link becomes available again. The subsequent packets will thus suffer additional delay only due to waiting

in the queue. In queueing theory the time required to serve a "customer" - packet in our case - is called the *service time*. We introduce a term for denoting the service time for the largest packet size: the *Packet Service Time (PST)*. This basically represents the delay suffered by a packet of maximum size allowed by Ethernet when going through the network. For 1 Gigabit it is $12.336\mu s$, for 10 Gigabit $1.2336\mu s$ and can be used as a worst case scenario (or the most conservative delay) when the distribution of packet sizes is not known.

The fraction of the time the network medium was busy transmitting something is called the *network load*. It is measured as a percentage of the link capacity and is calculated as the number of transmitted bytes divided by the maximum number of bytes the link can actually transmit. This metric is calculated as an *average* value, typically over 30 seconds or more, hence it does not capture the instantaneous load, e.g. how bursty the traffic really is. We refer to this as the *traffic* or *arrival pattern*. There is a strong dependency - proved mathematically e.g. in [2] - between the network load, traffic pattern and the time waiting in the queue for a given service time. Furthermore, if the traffic pattern is constant, what remains is the dependency between the queueing time and the load.

We stated earlier that the G and S components are not dependent on the load, while V is. Furthermore, V is not dependent on packet size, hence it has stationary statistical properties on the entire range of sizes. As explained in Section 4.4.5 the method for obtaining V implies the removal from the ΔQ of the immutable aspects, including the dependency on size. For a given load, the measured V is thus a distribution characterized by mean, variance and higher-order moments if necessary.

5.8.1.2 XPU-ROS path

The most demanding communication in the Level-2 system is the ROS-L2PU communication. The Level-2 algorithms require event data stored in the ROSs in up to three iterations until a decision is made, as detailed in Section 2.3. The allowed Level-2 execution time per event - including both communication and processing - in order to avoid events filling up the ROBs is dependent on the number of Level-2 processors available (see Section 3.5) and is of the order of tens of milliseconds ($\mathcal{O}(10^{-2})\,s$). We will quantify what is the fraction of this value caused by the network and how it increases with the network load.

The infrastructure supporting this communication is depicted in Figure 5.13. ΔQ measurements were performed in the manner described in Section 5.6.1 having as observation points the network interfaces on the XPU and ROS computers. No lost packets were detected during the measurements. The tools which compute the measurements automatically extract the mean, the standard deviation and the improper CDF for V with thirteen percentiles: 10th, 20th, 25th, 50th, 60th, 70th, 75th, 80th, 90th, 95th, 99th, 99.5th and 99.9th. The number D associated with a percentile P indicates the delay value for which P percent of the total samples have their delay less than or equal to D. For example, the maximum delay measured for the "fastest" 25% of the samples is the value of the 25th percentile.

Figure 5.15 presents the results for V measured on the aforementioned path, on the direction

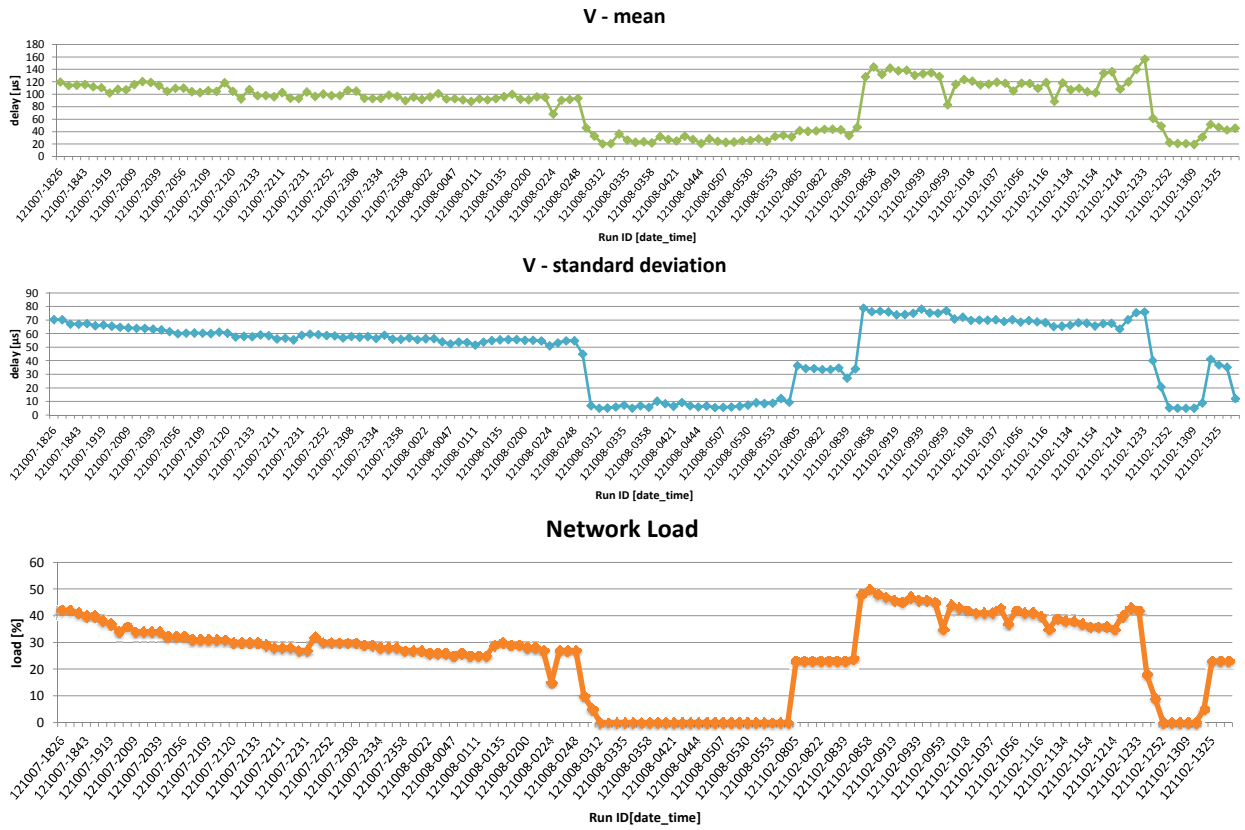

Figure 5.15: V(mean and standard deviation) evolution with the network load

ROS→XPU, knowing that this is the data flow of interest. In Figure 5.16 we normalized each plot to 1 in order to fit all three on a single graph and to better show both descriptors for V closely follow the evolution of the load.

However, during the zero load periods the mean and standard deviation are indeed smaller, but not zero. This accounts for the errors in measurements which we have to deal with, but also with other possible aspects in the functioning of the switches which we are abstracting away from in our analysis: OS scheduling and other randomness in the network devices. Also, the network is idle from the point of view of a monitoring tool which averages values over 30 seconds, while our probe packets are sent as a Poisson process with an average of 25 ms. It is thus likely to capture instantaneous data transfers generated by network protocols, e.g. Spanning Tree Protocol used for loop prevention.

Figure 5.17 plots V coefficient of variation obtained for the same measurements. The *coefficient of variation (CV)* is a dimensionless number representing the fraction of the standard deviation from the mean. We can observe that CV reflects the significant decrease of the standard deviation for zero load periods by taking values in the range $[0.2, 0.3]$. Furthermore, we see that for the ranges of loadings we were able to capture, i.e. $[0\%, 50\%]$ CV is less than 1, which is the value for the exponential distribution (characterizing Poisson processes). The distributions with CV<1

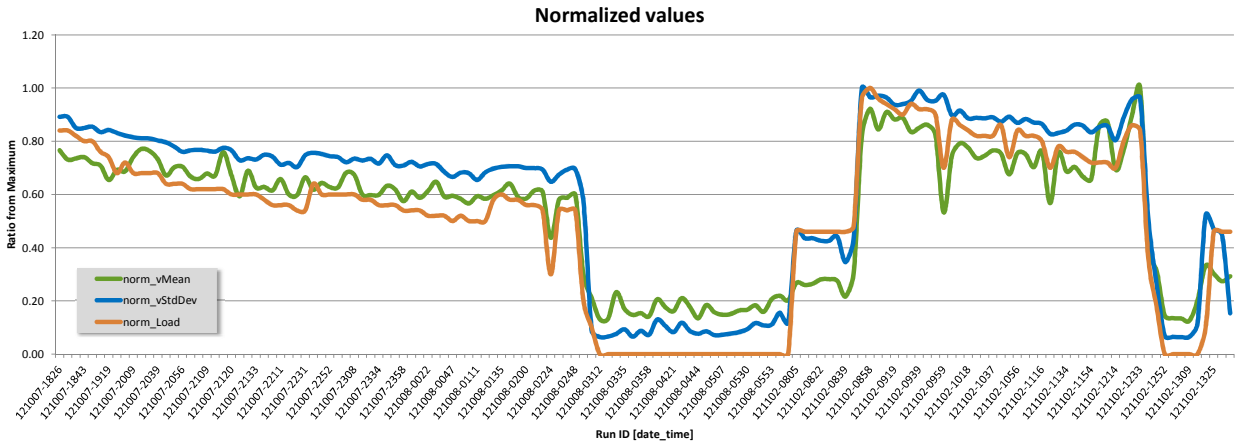

Figure 5.16: V moments evolution with the network load - normalized values

are considered to be low variance.

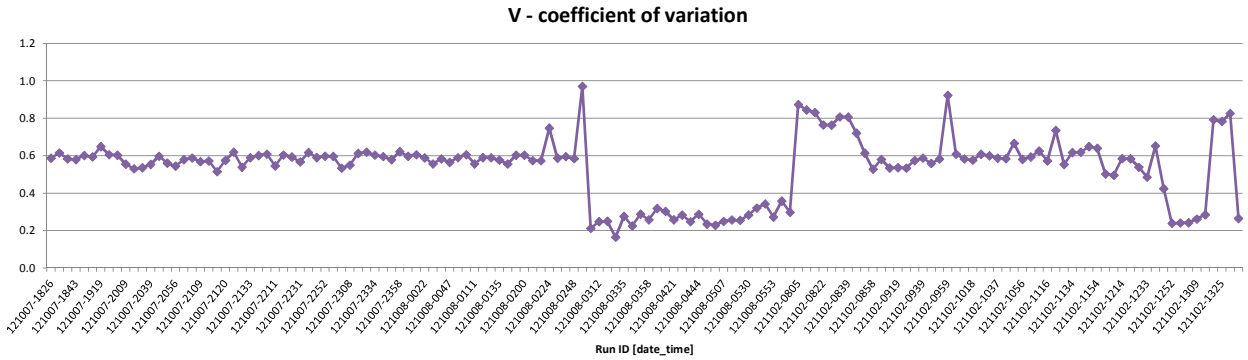

Figure 5.17: Coefficient of variation for V

The dependency of the mean and CV with the load is plotted in Figure 5.18, indicating two things. One is that the increase in V with load follows a linear trend for the entire range of load values, which indicates the potential increase for larger network loads. The second is that CV shows an increase from 0% to 25% load but becomes stable for loads between 25% and 50%, meaning that the standard deviation of V grows with the same amounts as the mean of V does. The conclusion is that the V random variable has a low variance distribution, regardless of the network load.

For our network segment of interest (XPU↔ROS), we showed the dependency of V on the network load. However, since V is the result of waiting in the queues and their occupancy depends also on the arrival traffic pattern, V offers additionally a valuable insight in what type of traffic the TDAQ software is generating. Knowing V we are able to quantify the queues occupancy and the shape of the traffic as a relative distance from the Poisson traffic, metric which is widely used by network theoreticians.

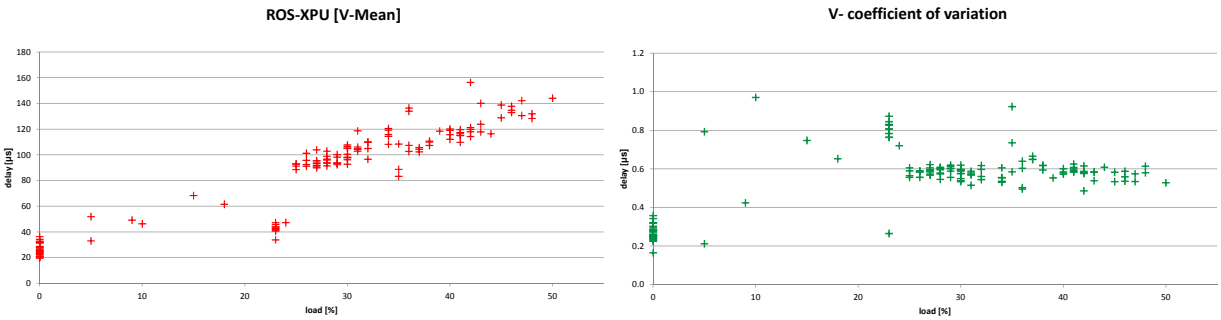

Figure 5.18: V (Mean and CV) vs. load

5.8.2 Queueing models

We introduce the notations we will use in the following sections, characteristic to the queueing theory models thoroughly described in [2]. We already applied Kendall's extended notation in Chapter 3 for capturing the Level-2 system. In case of the underlying network, we use the simplified notation A/B/C with:

- A - the arrival process - instantiated as: M for Markovian/Memoryless/Poisson arrival or G for general arrival. The associated distribution is for the time between two consecutive arrivals. A Poisson arrival is characterized by an exponential distribution of the inter-arrival times. The inter-arrival times in queueing theory is defined to be the times between the beginning of two consecutive packet transmission, hence they include packet transmission times. It must not be mistaken with the inter-packet time (or gap)[5], widely used in networking, which defines the times between the beginning of a packet transmission and the end of the transmission of the previous packet.

- B - the service time - instantiated as: M, D or G. The associated distribution is for the time required by the system to serve a packet. D is the case for deterministic service time, e.g. constant.

- C - the number of servers. A network segment is described by C=1.

The simplified model implies an infinite number of places in the system, i.e. large enough buffers in networking terms[6]. We thus have a limited set of models we will work with, a combination of: {M,G}/{M,D,G}/1.

In the case of Ethernet the service time is dependent on the medium speed and the frame size. For a given medium type the service time distribution is the distribution of the frame size measured and scaled in time units, hence we consider it to be G (general). For the 1 and 10 Gigabit medium the correspondence is:

[5] commonly abbreviated as IPG

[6] this assumption is supported by the extremely low rate of lost packets in the ATLAS TDAQ data network

$$T_{1G-size} = size[bytes] \times 8 \left[\frac{ns}{byte}\right] \tag{5.12}$$

$$T_{10G-size} = size[bytes] \times 0.8 \left[\frac{ns}{byte}\right] \tag{5.13}$$

We also mention the next set of notations:

- λ - the mean arrival rate of customers into the system. $\tau = \frac{1}{\lambda}$ is the mean inter-arrival time.
- μ - the mean service rate per server. In our case, for a single server, it represents the service rate of the network itself. $s = \frac{1}{\mu}$ is the mean service time.
- ρ - the server utilization, in our case the load of the network. $\rho = \frac{\lambda}{\mu}$
- C_X^2 - squared coefficient of variation of a positive random variable X. $C_X^2 = \frac{Var[X]}{E[X]^2}$
- W_q - expected steady state a customer spends in the queue.
- W_s - expected customer service time.
- W - expected steady state a customer spends in the system. $W = W_q + W_s$
- L_q - expected steady state number of customers in the queue
- L - expected steady state number of customers in the system. In our case $L = L_q + 1$

The **G/G/1** queueing model is the most generic one, with general arrival distribution and service time, hence applicable to our network. This model cannot be described mathematically by an equation between the elements above, but only by inequalities ([2, section 5.5]). One of them is for a particular case for a system loaded close to saturation and it's called the *heavy traffic approximation*. It is not applicable in our case due to the loading factors as high as only 50% (Figure 5.15). The ones useful to us set the bounds for W_q as a function of load, arrival pattern, service time pattern and average service time:

$$W_q \leq \frac{\rho W_s}{(1-\rho)} \left\{\frac{C_s^2 + C_\tau^2(2-\rho)/\rho}{2}\right\} \tag{5.14}$$

$$W_q \geq max\left\{\frac{\rho W_s C_s^2}{2(1-\rho)} - \frac{W_s}{2}, 0\right\} \tag{5.15}$$

From inequality 5.14 we can obtain after some transformations the lower limit for the C_τ^2:

$$C_\tau^2 \geq \frac{1}{2-\rho}\left[\frac{2W_q(1-\rho)}{W_s} - \rho C_s^2\right] \tag{5.16}$$

Other models: **M/G/1**, **M/M/1** and **M/D/1** will be used as reference points in establishing how a different arrival pattern or a different service would make a change in the queue sizes and the allowed loading of the network.

5.8.3 Inferring on queue sizes

On the ROS-XPU path queueing occurs in the network interface cards on the ROSs and in the switch output buffers where multiple sources of traffic compete for the corresponding output link. These places are identified in Figure 5.19 and explained next in a top-down manner.

The traffic sent by the ROS application can oversubscribe the 1 Gigabit output link, hence the network card has to buffer the instantaneous load. The arrival pattern is given by only one ROS application per computer which has to respond with data to the L2P processes (hosted on the XPU) and to the SFIs (for Event Building). The service time in these points is given by the link speed and the frame size, as in relation 5.12.

The second queueing point is in the output buffers of the ROS concentrator switches. Up to 10 ROSs are sending data through one 10 Gigabit uplink to the core switch and, although the link is not oversubscribed from the point of view of the bandwidth, instantaneous loads can cause buffering to occur. The arrival pattern is the result of the aggregated traffic coming from 10 ROS applications. The service time is in this case specific to the 10 Gigabit line-speed and is given by relation 5.13, i.e. ten times faster than for a 1 Gigabit link.

The third queueing point is similar to the second one, the only difference being in the arrival pattern. This is the result of a mixture of Level-2 and Level-3 traffic, knowing that XPUs can be designated L2PUs or EFs.

The fourth point is defined by the XPU concentrator switch output links towards the XPU computers. In case the XPU is a L2PU type, the arrival traffic is coming from the ROSs with Level-2 event data, which, if sent simultaneously, generate bursts which in turn cause queueing. The service time is the one for a 1 Gigabit link.

The load for all the links is obtainable from the network monitoring tool and has a 30 seconds granularity. Figure 5.20 shows a typical loading profile during a data taking run for the links involved in the ROS-XPU path. The red plots indicate the input load on a port, while the blue ones indicate the output load. We can notice that the highest load occurs on the ROS link to the concentrator switch, ~50% and decreases to ~5% towards the XPU. The network load corresponds to the ρ in the queueing theory, hence we will use the normalized values: e.g. 0.5 for 50%, 0.05 for 5%.

The ROS-concentrator and concentrator-core links are the most utilized in the whole path, however a significant difference exists between the two. The amount of queueing in the first case is controlled by an application which places data on a 1 Gigabit link at a potentially non-regulated rate. Once the packets are sent on the wire they are reaching the concentrator switch at a rate limited by the line speed. The 10 Gigabit uplink is able to accommodate all the traffic coming from 1 Gigabit links. Queueing in the second case can occur only as a result of the switching

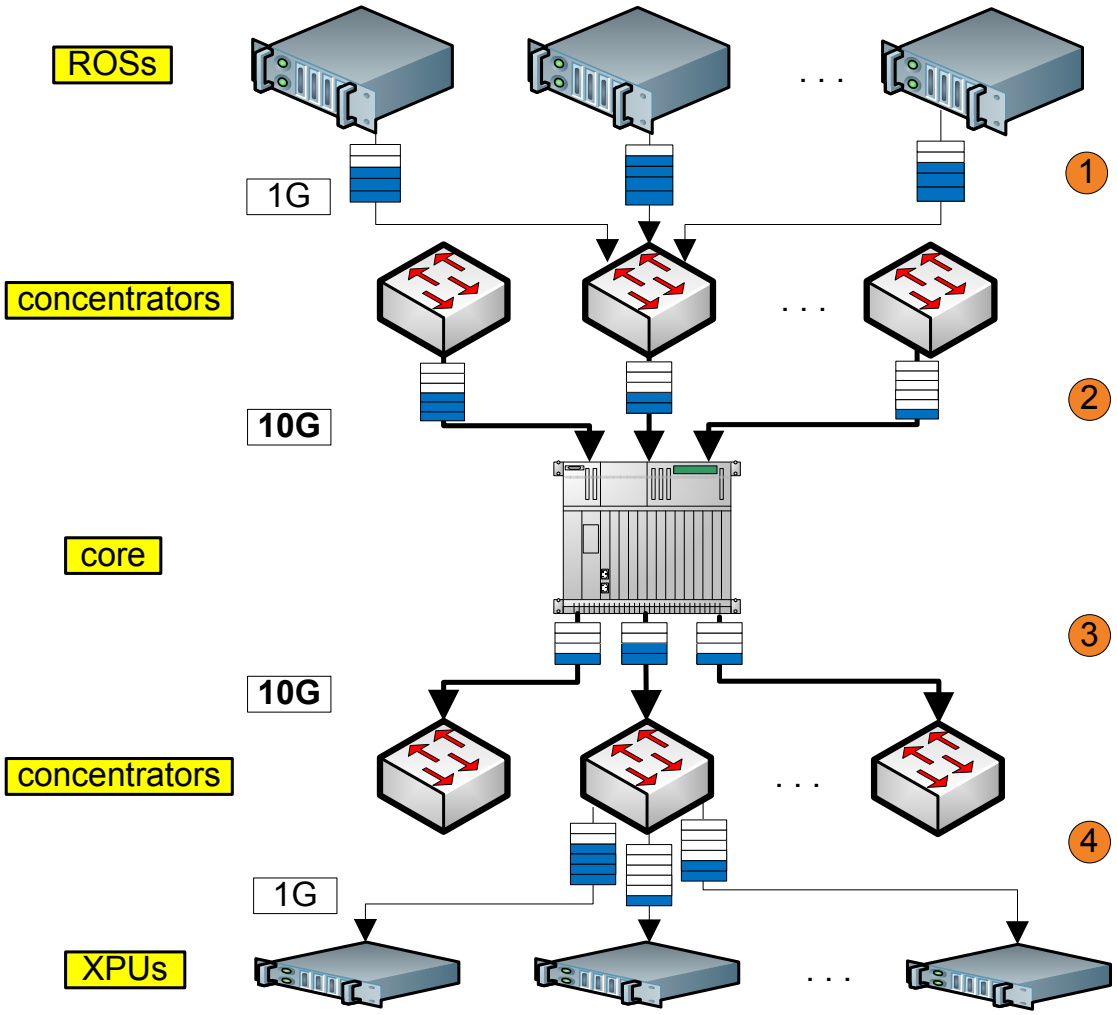

Figure 5.19: ROS-XPU queueing

mechanisms which could alter the rate of packets (i.e. group them together).

We thus consider the first queueing point to be the dominant one in the whole queueing chain. It is also the place where queueing occurs as a direct result of an application sending data over the network. Furthermore, the queueing models with one server and infinite queueing are mathematically tractable for only a queue in front of the server. Measuring the queue size in this point is thus of special interest for the queueing models relations which would allow us to extract ROS traffic patterns.

The approach we take makes use of the V component extractable from the ΔQ. We explained that V is a measure of how much queueing a packet suffers between two observation points. We also observed that for zero load the V component is not zero, which means that it captures also other random aspects inside the network devices when switching occurs. In order to cancel the latter we adopt a *differential* approach between V measured in two different loading conditions.

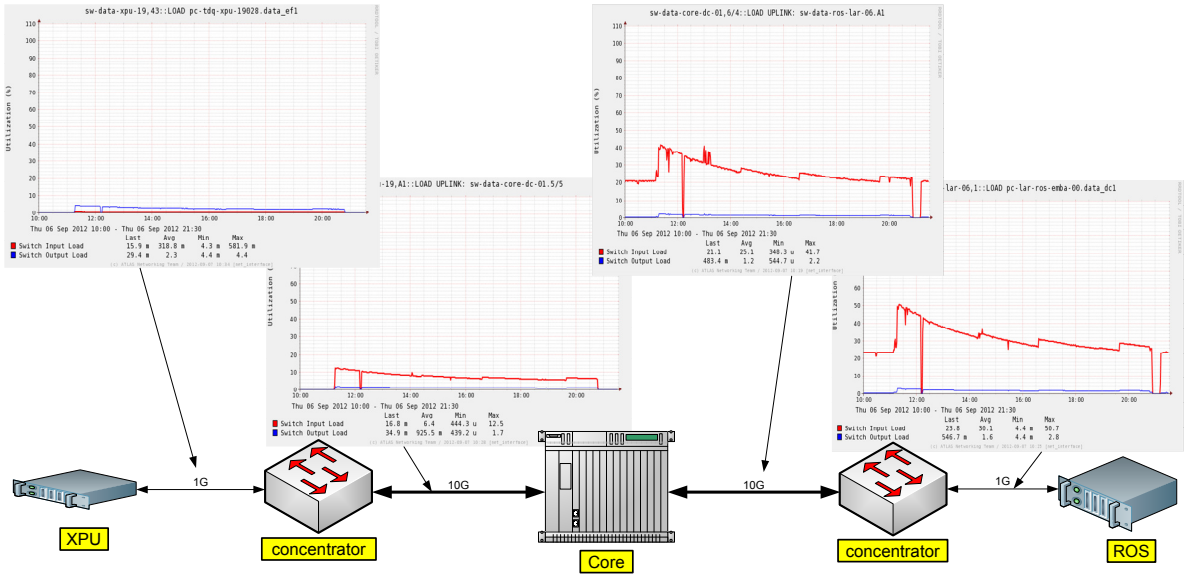

Figure 5.20: ROS-XPU network path loaded

The difference in V will thus indicate the additional queueing delay added by a certain amount of loading.

We performed the same type of ΔQ measurements as before, having as observation points two ROS computers connected in the same concentrator switch, as depicted in Figure 5.21. We sampled different loading factors of the ROS (A) to concentrator switch link, including the zero loading. The second link, from the switch to the ROS (B) carries "request for data" traffic coming from the L2PUs and from SFIs, which places a very low load on the network (see Figure 5.20).

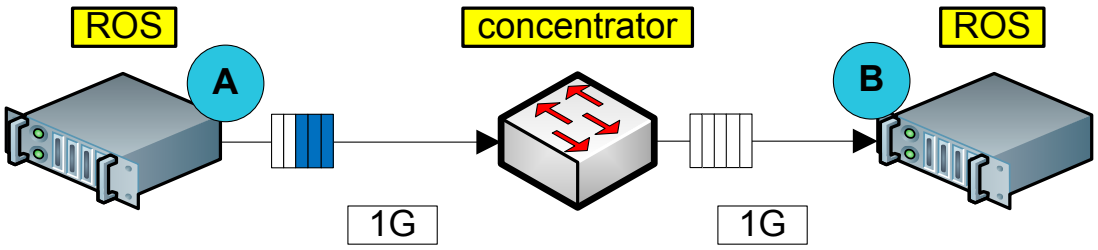

Figure 5.21: ROS-ROS path

Table 5.7 summarizes the V (mean and standard deviation) measured for three different loading conditions. The relative values are obtained by deducting the zero load values from the absolute ones.

The difference in the mean of V represents the W_q - the waiting time in the queue. If we would know the average service time W_s we could infer on the *size of the queue*:

ρ	V(mean) (μs)		V(std. dev.) (μs)	
	absolute	relative to zero load	absolute	relative to zero load
0	5.5	-	2.6	-
0.25	44	38.5	47	44.4
0.4	67.7	62.2	59	56.4

Table 5.7: V for different loading factors

$$L_q = L - 1 = \frac{W_q}{W_s} - 1 \tag{5.17}$$

For this purpose we captured more than 300,000 packets from real traffic (snapshot took on the ROS interface) which showed the following statistical characteristics for the packet size distribution: $average = 433.7\,bytes$ and $standard\,deviation = 834.5\,bytes$. Knowing that the service time is linearly dependent on the packet size (relation 5.12) we are able to calculate:

$$W_s = \frac{433.7 \times 8}{1000} = 3.46\,\mu s \tag{5.18}$$

$$C_s^2 = \frac{834.5^2}{433.7^2} = 3.7 \tag{5.19}$$

We can now estimate the size of the queue in both loading situations (i.e. how many packets are waiting in the ROS network buffers) to be:

ρ	L_q
0.25	10
0.4	17

Table 5.8: Queue sizes for different loading factors

5.8.4 Results from theoretical models

We considered the queueing model in Figure 5.21 to be the most generic one: G/G/1 because of the generic distributions for the arrival pattern and service time. We want to quantify the difference between our system and a theoretical one from the point of view of the arrival traffic.

Having obtained the queue sizes and the coefficient of variation of the service time C_s^2 we can insert these values into the formulas for three theoretical models which consider the arrival pattern to be Poisson: M/G/1, M/M/1 and M/D/1. We are thus able estimate the theoretical loading factors for these models and compare them with what we observed in the network monitoring tools.

For the M/G/1 queueing system the relation 5.20 (known as Pollaczek's formula) applies:

$$W_q = \frac{\rho W_s}{1-\rho}\left(\frac{1+C_s^2}{2}\right) \tag{5.20}$$

Knowing $L_q = \lambda W_q$ and $\rho = \lambda W_s$ it follows that

$$L_q = \frac{\rho^2}{1-\rho}\left(\frac{1+C_s^2}{2}\right) \tag{5.21}$$

from which we can obtain the load ρ:

$$\rho = \frac{\sqrt{L_q^2 + 2L_q\left(1+C_s^2\right)} - L_q}{1+C_s^2} \tag{5.22}$$

For the M/M/1 system the service time is described by a Poisson distribution characterized by $C_s^2 = 1$. ρ becomes

$$\rho = \frac{\sqrt{L_q^2 + 4L_q} - L_q}{2} \tag{5.23}$$

For the M/D/1 system the service time follows a constant distribution characterized by $C_s^2 = 0$, ρ becoming

$$\rho = \sqrt{L_q^2 + 2L_q} - L_q \tag{5.24}$$

Computing the load for all the theoretical models above and for the two queue sizes we estimated in Section 5.8.3 we obtain the results in Table 5.9. We notice a big difference between the observed load and the theoretical loads. These results say that the queues sizes we calculated from observation can be obtained by some much higher loading factors if Poisson traffic is used.

Queue size	ρ			
	observed (G/G/1)	M/G/1	M/M/1	M/D/1
10	0.25	0.83	0.91	0.95
17	0.40	0.89	0.94	0.97

Table 5.9: Loading factors for theoretical models

Applying relation 5.21 on the observed loading factors but for the M/G/1 model, we obtain an average queue size extremely smaller compared to the measured ones (i.e. smaller than 1), which confirms that the big difference in queue sizes is made by the *arrival pattern* into the queueing system.

By comparing the results obtained from measurement with the theoretical ones we identified the cause for the large queues - and consequently for the large V values - to be the type of traffic the ROS application places on the network. We continue our analysis by showing how far is the real traffic from the theoretical Poisson traffic.

5.8.5 Traffic patterns

5.8.5.1 Comparison to Poisson process

The statistical property of the arrival patterns which can be used as a metric when comparing them with respect to the burstiness is the coefficient of variation of the inter-arrival times: C_τ. It is worth mentioning two aspects regarding the arrival pattern:

1. In order to isolate the effect of packet lengths on the arrival pattern, a comparison was carried on in [26] between the autocorrelation coefficients of inter-arrival times and those of inter-packet times, as defined in Section 5.8.2. The outcome of this comparison was that no noticeable difference was observed between the two, hence no significant correlation was caused by the packet lengths.

2. A different approach in characterizing the arrival process variability was addressed in [27] making use of the indices of dispersion for intervals and for counts. It can assist the comparison of two traffic patterns being applied to different traffic types generated by computer workstations. Although more complex than the simple coefficient of variation, we consider the latter to be sufficient when a comparison is made to a theoretical pattern, i.e. Poisson.

An arrival pattern is considered bursty if the inter-arrival time distribution shows greater variability than for a Poisson process, i.e. $C_\tau > 1$. Inequality 5.16 sets the lower limit for the coefficient of variation - in its squared version - as a function of the load, waiting time in the queue, waiting time in service and the coefficient of variation of the service time. We can thus calculate this limit for the two loading situations we captured on the ROS-ROS traffic and which we summarized in Table 5.10.

ρ	$C_\tau \geq$
0.25	2.85
0.4	3.43

Table 5.10: Coefficient of variation (arrival traffic) for two loading factors

These values reveal a very bursty traffic, two-three times more bursty than the Poisson process, a characteristic which increases with the load. We will investigate next the causes for this by looking at real traffic samples captured on the ROS interface.

5.8.5.2 Real traffic captures

We captured approximately 360,000 TCP packets on one of the ROS data interfaces during data taking using the *tcpdump* Linux tool. The tool was configured to capture all the traffic having as source the ROS computer. The load of the ROS network was ~40%, i.e. $\rho = 0.4$. We were thus able to observe two things regarding the traffic placed on the network by the ROS: the packet size variation as received by the network card and the packet inter-arrival time distribution.

We identified two types of packets: small packets which fit an Ethernet frame (marked by the *tcpdump* tool with sizes less than or equal to 1514[7]) and large packets which require *fragmentation.* Fragmentation is the process by which large blocks of data are split at a certain OSI layer in smaller pieces in order to fit the underlying layer's *Maximum Transmission Unit (MTU).* This process can be performed inside the operating system or inside the network card for the purpose of saving CPU processing power. For a TCP segment, in case the operating system passes on this responsibility to the network card, the process is called *TCP offload* (detailed in e.g. [4]).

5.8.5.3 Packets not fragmented

We will analyze next the arrival pattern component caused by the first category of packets. Figure 5.22 shows the distribution of the inter-arrival times for packets which fit an Ethernet frame, hence not requiring fragmentation. The total number of these packets is ~320,000, representing ~88% of the total captured packets.

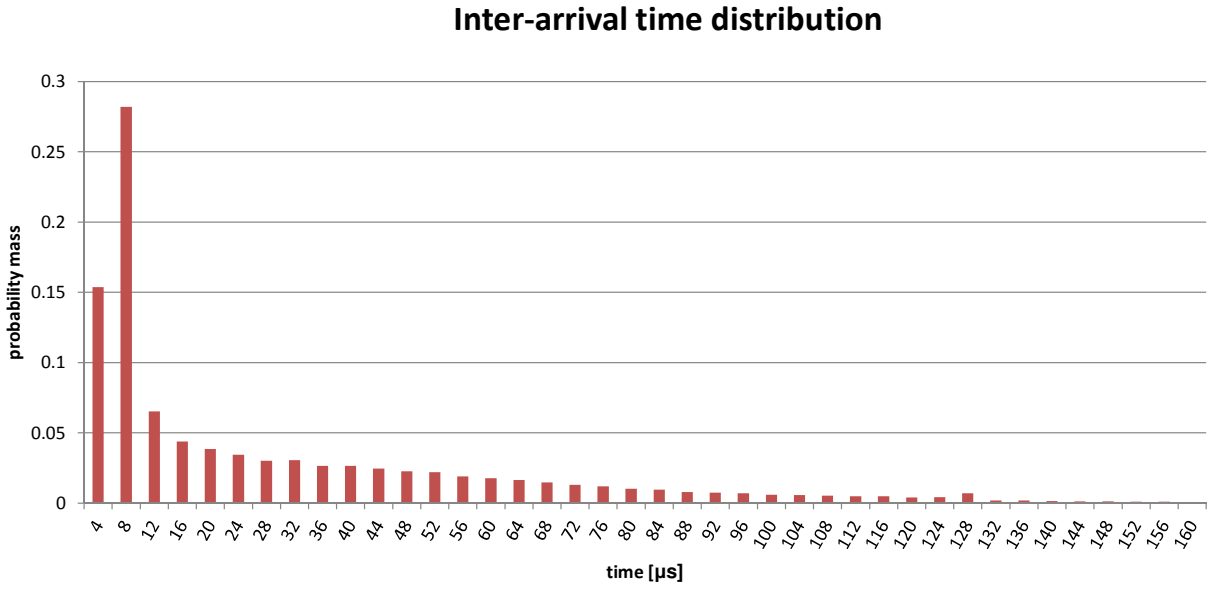

Figure 5.22: Packets not requiring fragmentation - inter-arrival time distribution

[7] 1514 is the maximum Ethernet frame size without the preamble, IFG and VLAN tag, as detailed in subsection 5.3.2

Property	Value
Mean	29.73 μs
Standard Deviation	37.66 μs
Coefficient of Variation	1.27
Count	321240

Table 5.11: Statistical properties of inter-arrival times of small packets

The statistical properties of the arrival pattern for the not fragmented packets are summarized in Table 5.11. The mean value for the inter-arrival times is larger than the *Packet Service Time* introduced in Section 5.8.1, which is 12.336μs. This means that the not fragmented packets arrive on average at intervals larger than the time 1 Gigabit line requires to process a full Ethernet frame. We can also observe that the coefficient of variation is larger than one but under the estimated value of 3.43 in Table 5.10. These aspects indicate that *there is a different contributor to the arrival pattern* which is the cause for the difference in the coefficient of variation, i.e. in the amount of burstiness.

5.8.5.4 Packets requiring fragmentation

We continue our investigation by looking at the second type of packets: the ones requiring fragmentation. Out of the total number of captured packets, ~40,000 (12%) are larger than the maximum Ethernet frame size. Figure 5.23 shows the distribution of the size of these packets. We can notice a large number of packets of size around 10,000 bytes, more precisely 10,274 bytes, being the result of the ROS application action of aggregating the event data fragments from all the hosted ROBs.

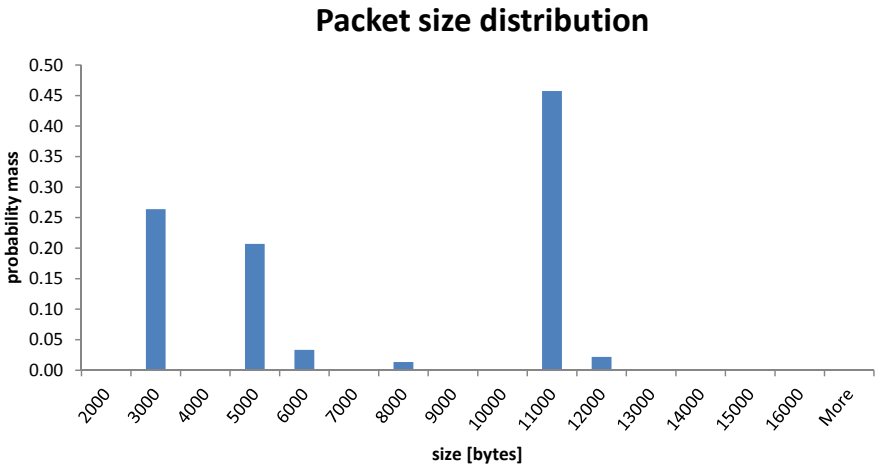

Figure 5.23: Packets requiring fragmentation - size distribution

All the computers in the ATLAS TDAQ system currently have the TCP offload feature enabled,

meaning that the processing required by the TCP protocol is performed on the network interface cards and not inside the operating system. For packets larger than the maximum allowed Ethernet frame the network interface cards deals also with their fragmentation.

Once a packet has been fragmented, all the resulting Ethernet frames are sent back-to-back on the 1 Gigabit wire, hence queueing occurs in the output buffers of the network interface cards. This creates an artificial effect on the arrival pattern by inserting groups of packets within the distribution plotted in Figure 5.22. We will quantify this effect mathematically by calculating the additional delay added by a queueing system when clustering is performed on the arrival packets. We will also simulate the modification of the arrival pattern - and implicitly of its coefficient of variation - introduced by the fragmentation.

5.8.5.5 Additional delay caused by bursts

A packet of size s is split into $n = \left[\frac{s}{1514}\right]$ packets[8] of size 1514 and one packet of size $s - n \times 1514$, hence a total group size of $n + 1$. In order to see the arrival pattern of the groups we plotted the inter-arrival times of the packets requiring fragmentation in Figure 5.24. Along with this we fitted a theoretical exponential distribution with the mean of the experimental results, i.e. $1/\lambda = 280\mu s$. We can observe that the empirical distribution follows closely the theoretical one within a limited tolerance.

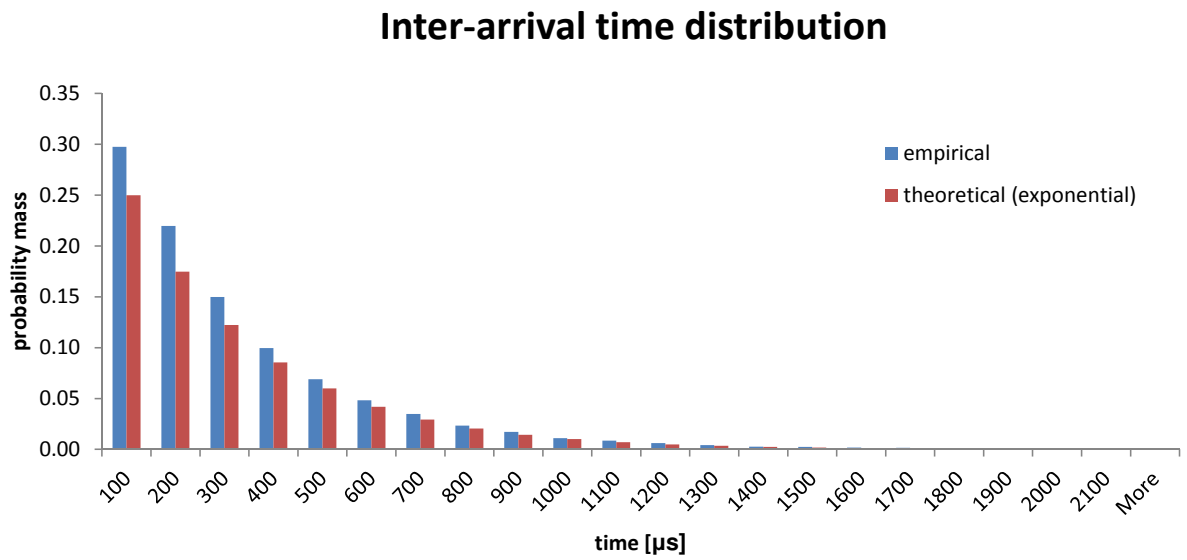

Figure 5.24: Packets requiring fragmentation - inter-arrival time distribution

Considering the arrival of groups to be a Poisson process we can apply a theoretical result mentioned in [1, section 10.4]. It states that the additional delay introduced by the clustering of arrivals, denoted W_G, is given by:

[8] [x] is the integer part of x

$$W_G = \frac{(E[G^2] - E[G])\, W_S}{2E[G](1-\rho)} \tag{5.25}$$

where:

- G - is the random variable characterizing the group sizes. In [61] it is shown that $\sigma_G^2 = Var[G] = E[G^2] - E[G]^2$.
- W_S - is the mean service time of 1 Gigabit Ethernet for the packets obtained after fragmentation. The average packet size is 1475 bytes.
- ρ - is the loading factor. $\rho = \lambda E[G] W_S$, where λ is the rate of the Poisson process characterizing the groups arrival.

The statistical properties of G obtained by measurement are summarized in Table 5.12.

Property	Value
Mean	4.77
Standard Deviation	2.42
Variance	5.86
Count	34,295
Sum	163,800

Table 5.12: Statistical properties of groups

Knowing also that

$$\lambda = \frac{1}{280}\mu s^{-1} = 3.571 \times 10^{-3}\mu s^{-1} \tag{5.26}$$

and

$$W_s = 8 \times 1475 \times 10^{-9} s = 11.8\mu s \tag{5.27}$$

we obtain the additional delay to be

$$W_G = 36.98\mu s \tag{5.28}$$

This value represents the additional time spent in the queue by packets arriving in groups described by a Poisson random variable G with rate λ compared to the situation when the same amount of packets would arrive according to a Poisson process with the rate $\lambda E[G]$.

In our case, at a 40% network load, i.e. $\rho = 0.4$, at which the real traffic captures took place, we observed a queueing delay of $120\mu s$ - represented by the mean value of V in Figure 5.15. We thus estimated that 30% ($36.98\mu s$) of this is caused by the clustering of packets, which is produced by the TCP offload mechanism enabled on the ATLAS TDAQ computers.

5.8.5.6 Additional burstiness

The grouping of packets at the network interface card due to the TCP offload mechanism does not allow us to capture the inter-arrival time distribution of the packets using the *tcpdump* tool. What we observe is only the inter-arrival time distribution of the packets at the Linux kernel level. We can however estimate this distribution as it is generated by the network interface card, i.e. after the fragmentation takes place.

As mentioned earlier, each packet requiring fragmentation becomes a group of packets sent back-to-back by the network card on the output buffer placed right before the 1 Gigabit link. For these packets we consider an inter-arrival time close to zero. The number of packets generated by the fragmentation can be calculated from the G random variable introduced above in Section 5.8.5.5. This number is given in the "Sum" row in Table 5.12: 163,800.

In order to obtain the number of packets sent back-to-back we have to subtract the number of groups (row "Count" in Table 5.12: 34,295), as they have measurable and larger inter-arrival times. In the end we obtain ~130,000 packets with inter-packet times close to zero. Estimating the arrival distribution as seen by the network card output buffers involves modifying the distribution shown in Figure 5.22 in two ways:

- adding the 130,000 number in the first bin characterizing the smallest inter-arrival times.
- overlapping the empirical distribution of the groups arrivals in Figure 5.24.

The result is depicted in Figure 5.25. The change in the tail of the distribution is caused by the inter-arrival times of the packets requiring fragmentation, exhibiting values larger than $100\mu s$

The polarization of the simulated values in the two extremes increases the dispersion of the arrival pattern. This accounts for the missing component in the coefficient of variation identified in Section 5.8.5.3 where we noticed a difference between the one measured for the packets not requiring fragmentation and the theoretically estimated value.

5.8.6 Conclusions on V

The investigation on the traffic pattern was triggered by the significant discrepancy between the theoretical results for queueing models and our measurements by means of V component of the ΔQ. In a network the mean value of V represents the average time of waiting in the queues - in the ATLAS TDAQ case the ROS output buffers. From V we deduced the size of the queue and we could use this value to show that the theoretical models indicate a much higher loading factor than the one measured if Poisson arrival traffic is to be used.

We thus identified the cause of this discrepancy to be the arrival pattern distribution. By looking at a short sample of real traffic we deducted a lower bound of the coefficient of variation of the arrival pattern which revealed a traffic between two and three times more bursty than the Poisson one. The examination of the real traffic indicated that one of the important causes for burstiness

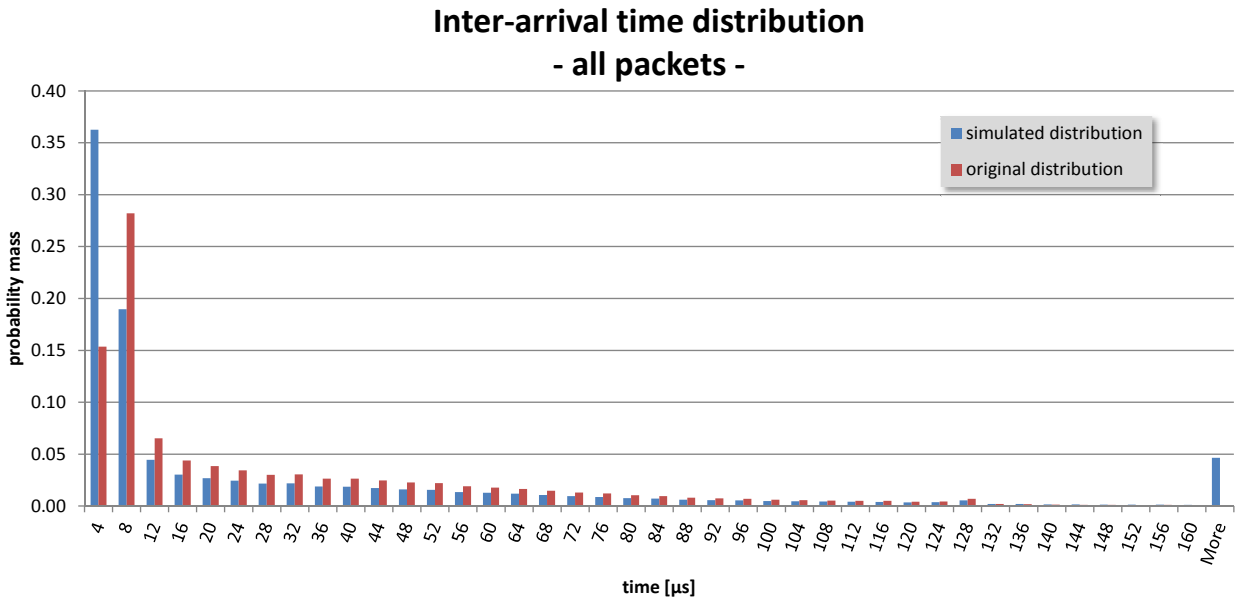

Figure 5.25: Simulated distribution after packets fragmentation

in the system is the TCP offload setting caused by large TCP fragments generated by the ROS application. We quantified the additional delay caused by this mechanism to be around 30% of the total delay suffered by a packet.

The traffic pattern was investigated using the ROS-ROS measurements as depicted in Figure 5.21 because it represents a single queue system, easy to model mathematically. However, the discoveries on the traffic pattern are affecting the performance of the entire ROS-XPU path, which is the main Level-2 communication path. Next we will predict network behaviour for our path of interest for increased loads, under the same arrival traffic patterns. A change in the arrival pattern changes the V caused by the ROS output buffers and, due to the topological convolution properties of ΔQ, affects the entire ROS-XPU path.

5.9 Predictive model

Measuring ΔQ of different network paths and under different loading conditions offered us cognitive insights into performance measurements of the ATLAS TDAQ network. We were able to apply the observational model with its particularities for an Ethernet network from which we identified two types of network performance characteristics: fixed (Structural Delay) and variable (dependent on running conditions and thus manageable). We will use their compositional properties to capture predictive aspects of the network behaviour.

5.9.1 Extrapolation of V with load

An important result mentioned in Section 5.8.1 refers to the dependency of V on the network load and traffic pattern. Moreover, the network load and the traffic pattern are dependent on each other. This dependency was captured in Table 5.10 by showing that the coefficient of variation of the arrival pattern increases with the load.

We also learned that the coefficient of variation of V is constant with the load, meaning that the standard deviation of V grows with the same amount as its mean does. At higher loads we thus have "wider" variations which translate into increased delays for packets. In order to describe this effect for the ROS-XPU path we extracted three percentiles from the measurements on V: 90th, 95th and 99th. We plotted them along with the mean against the network load in Figure 5.26 and fitted a linear trend for each of them.

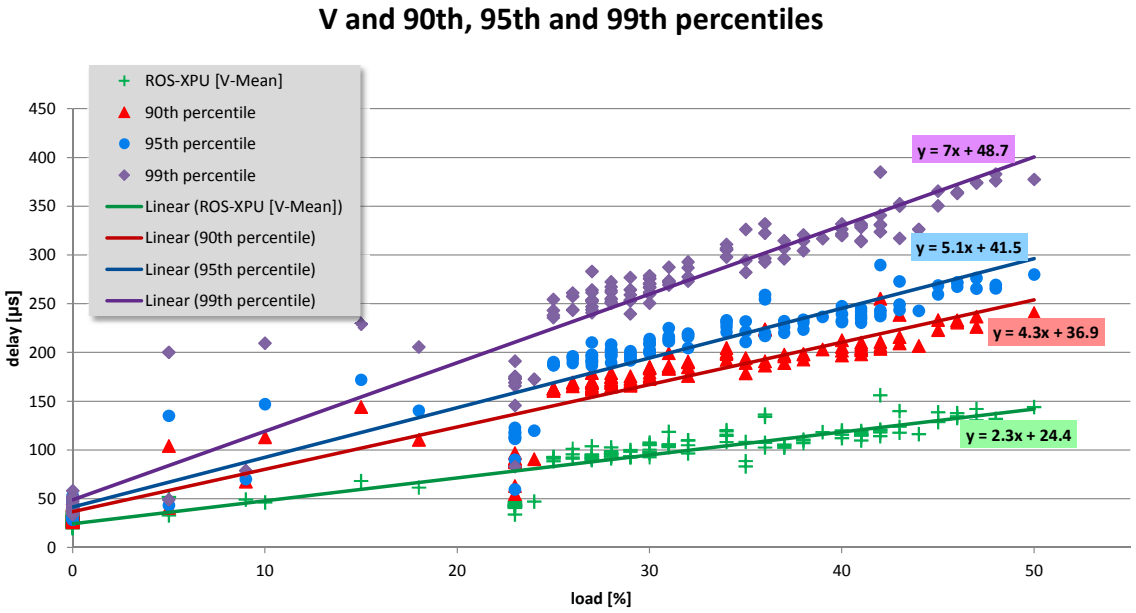

Figure 5.26: V mean and percentiles with load

This graph exposes the lower bounds of the V component of the overall delay for 10%, 5% and 1% of the packets. For the highest captured load we notice that 1% of the packets have their queueing delay larger than $400\mu s$.

5.9.1.1 Non-linearity

Using the linear trends obtained so far for network loads under 50%, one could extrapolate the network behaviour for increased loads. Figure 5.27 depicts the extrapolative area above 50% network load if linear trends are considered.

However, we showed in Section 5.8.5.5 that an important portion of the overall delay is caused

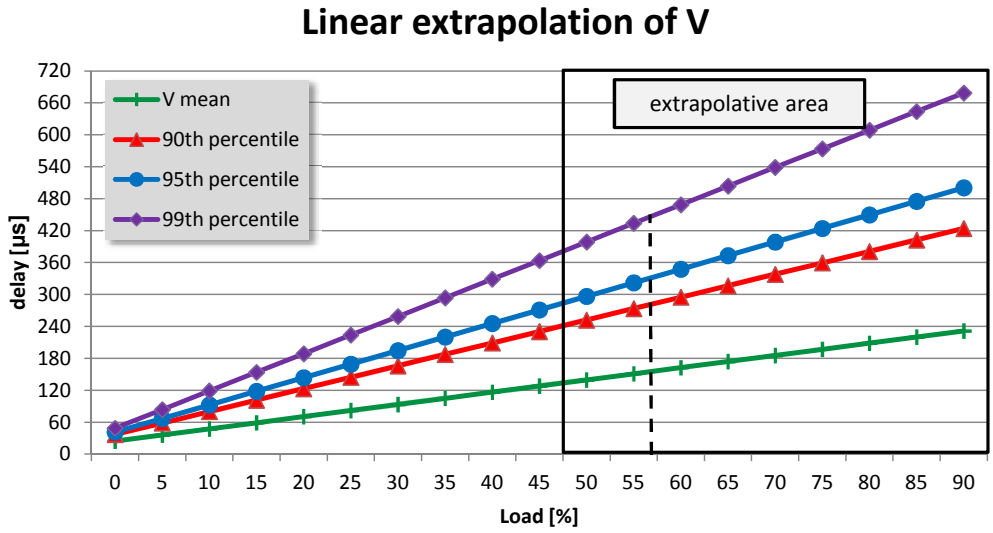

Figure 5.27: V mean and percentiles extrapolation with load

by the clustering of packets as result of the TCP fragmentation at the network interface card. We called this amount of delay to be W_G, i.e. the waiting in the queues due to *grouping*, and is expressed by relation 5.25. This relation reveals the fact that W_G is a function of load of the type:

$$W_G = \frac{k}{1-\rho} \tag{5.29}$$

where k captures the structure of the groups generated by fragmenting the large packets. For the ATLAS TDAQ type of traffic, we measured it to be:

$$k = \frac{(E[G^2] - E[G])\, W_S}{2E[G]} = 29.54\mu s \tag{5.30}$$

Relation 5.29 represents a non-linear dependency of the waiting time on the load, and is depicted in Figure 5.28. We can observe that for loads up to 50% the W_G can be approximated by a linear dependency on the load, confirming the measurements in Figure 5.26. For increased loads however, the non-linear contribution of the W_G becomes more dominant and hence the linear extrapolation in Figure 5.27 does not hold.

Instead, Figure 5.29 depicts the *non-linear extrapolation* of V considering the contribution of the W_G component obtained above. At design network load (60%) and higher the Level-2 network should expect the delays in Table 5.13 for the slowest 10%, 5% and 1% of the packets.

On top of these values we need to add the Structural Delay component in order to obtain the total ROS-XPU delay characteristic. The measured Structural Delay is given by relation 5.10, which for the extreme packet size cases they yield an additional delay of: $51\mu s$ and $81\mu s$. These values

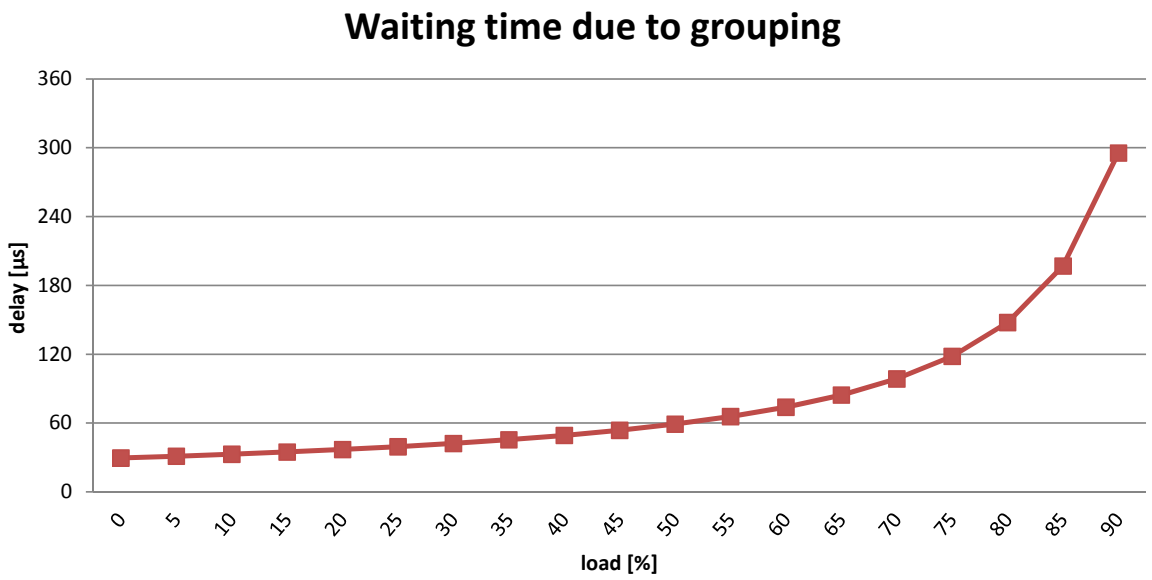

Figure 5.28: W_G dependency on load

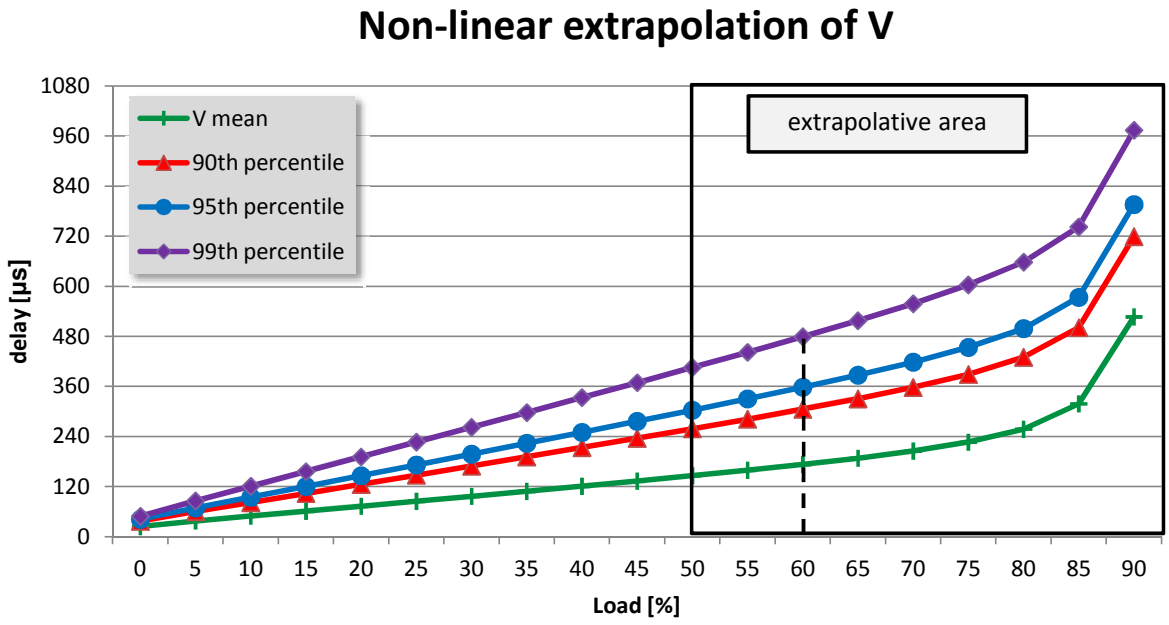

Figure 5.29: V mean and percentiles non-linear extrapolation with load

Load [%]	V mean [μs]	percentiles [μs]		
		90th	95th	99th
60	173	306	359	480
70	205	358	418	558
80	257	430	499	658
90	526	719	796	974

Table 5.13: V at increased loads

need to be inserted into the Level-2 overall processing delay in the network BUSY intervals depicted in Figure 3.1.

5.9.2 Level-2 delay

An L2 algorithm iteratively requires on average 2% of the full event data from the Read-Out System, meaning an average value of 32 KB. The amount of data ranges from a few to 50 KB per event. From the networking point of view this translates into a minimum 34 packets sent on the ROS-XPU path.

The L2 process has to wait for all the packets to be received in order to re-assemble the data and run the algorithm on it, process which is denoted as *last-to-finish synchronization*. This implies that the delay of the event data transfer from the L2 process perspective is given by the "slowest" received packet. One out of 30 packets, representing 3.3%, falls under the 95th percentile delay. Thus, the Level-2 processing depends on the ROS-XPU delay given by the values in Table 5.13 for the 95th percentile to which we add the Structural Delay: $51\mu s$ for smallest packet size and $81\mu s$ for the largest.

5.9.3 Other aspects

Many of the aspects discovered by looking at the ΔQ can be seen as means of predicting network behaviour in case different conditions are set.

One of them makes use of the topological composition of ΔQ. We showed in Section 5.7 how we can obtain the end-to-end Structural Delay by convolving this characteristic of each network element. We can thus predict the effect on the end-to-end performance by the replacement of a single network component without re-evaluating the entire network path. All we need to to is to measure its Structural delay and replace it in the end-to-end one.

Another aspect is related to the queue sizes. We have the means of inferring on queue sizes and hence dimension the system or change some settings (e.g. TCP offload) accordingly. The next upgrade of the ATLAS TDAQ network makes use of newly introduced 10 Gigabit copper links which will change the mean service time of the network by an order of 10. This will have a direct effect on the queueing delays and using the V component of the ΔQ we are able to capture that.

Equally, a change in the TDAQ software has a quantifiable effect on the arrival pattern into the network, hence the amount of delay to be expected can be extracted.

Chapter 6

Conclusions, original contributions and future steps

This final chapter summarizes the work carried on in this thesis in the context of the ATLAS TDAQ operational experiment, emphasizes the original contributions brought by the author in the domain of network performance and proposes ideas and future research steps based on the current work.

6.1 Conclusions

Particle physics reached an inflection point in its history potentially marking a big step in the discovery of new particles and of the missing parts of the current physics models. We are witnessing a moment in history when computer science, networking and physics are converging in the sense that there is theoretically enough processing power and speed of communication in the commodity equipments required to run high-energy physics algorithms in real time.

This great purpose placed an extremely high demand on the tools which were designed to accomplishing it, hence on the data acquisition system of the ATLAS experiment. Its design and implementation took almost twenty years in a epoch described by increasingly fast developments of communication and computing technologies. An example is the Ethernet standard. When ATLAS TDAQ network was designed 10 Gigabit Ethernet on optical fiber was only emerging, but by the time the system was operational the 40/100 Gigabit and 10 Gigabit on copper were already standardized. It thus became obvious that the ATLAS TDAQ system could perform faster, hence an upgrade is envisaged soon.

In this ever-changing context the current work addresses the need of understanding the performance of such a complex system. A method of modeling the ATLAS TDAQ system using a Discrete Events Simulator [32, 21] is not able to keep up with this evolution, while a pure mathematical description of the system offers only limited information. This thesis proposes a method of describing performance using high level statistical metrics which provides a manageable ab-

straction level for complex systems like ATLAS TDAQ. It also employs mathematical queueing models as reference points or as offering bounds on performance aspects, where applicable.

After having introduced the ATLAS TDAQ system we started with a description focused on the performance coupling and strong dependency between the system components, all putting performance constraints on the interconnecting data network. We emphasized the fact that the design and current network performance metrics relied on average values like bandwidth, because loss and delay were considered to have a limited impact on the global performance. Equally, loss and delay are instantaneous characteristics which were difficult to capture or predict at design stage. We addressed them considering that these aspects have an impact on the scalability of the system and also on its behaviour close to saturation. We also made use of a system in its operational phase, hence access to valuable real traffic samples.

We obtained some useful, yet limited results of a mathematical model applied on a critical subsystem called the Level-2 trigger, justifying the need for a different approach in case a more thorough performance description is desired. We incorporated this approach in the framework called *the observational model*, name coming from the stochastic process algebra paradigm which relies on observational congruency between systems. Applying this model for data networks, Ethernet in particular, we introduced a mathematically supported concept which encapsulates both loss and delay: *the quality attenuation* or ΔQ.

The method of obtaining ΔQ is a cheap, non-intrusive one and, although exposed to measurement errors due to the time-stamping process, a statistical approach in performing it is able to yield good results if enough samples are used. We exploited ΔQ - a component-wise metric - in multiple ways:

- as a comparison metric between network devices. We used a differential approach in order to obtain individual performance characteristics.
- as an end-to-end network description, using its piece-wise convolution properties.
- as a measure of the delay caused by queueing. A comparison with theoretical models lead us to investigating the traffic pattern structure.
- as a model able to capture predictive aspects in case different running conditions are set, e.g. increased network loading conditions

6.2 Original contributions

We can classify the original contributions of this thesis in two categories: *theoretical* (by defining novel theories, methods and concepts) and *practical* (by obtaining new results using already existing work or our introduced theories).

Theoretical

The first theoretical aspect consists of the mathematical description of the Level-2 system as a queueing model. We proved that the M/G/c/c model is the most accurate and yet mathematically tractable description of the second filtering level of the ATLAS TDAQ.

Following the conclusions on the pure mathematical model we introduced the observational model - a generic framework made up by a set of concepts: outcomes, tasks, observables and the central one: quality attenuation (ΔQ), aimed to describe performance of any communication system. ΔQ is mathematically sustainable by improper CDFs, concept which is able to capture loss, delay as well as constant values through the Dirac delta function.

For data networks we decomposed ΔQ into three basis, statistically independent random variables for which we presented their properties. The most important one is the compositional property, computable using the results for the convolution of random variables distributions. Furthermore, we defined a methodology to extract the ΔQ components from a large number of measurement samples using a linear statistical dependency of the delay on the packet size.

The mentioned observation model is able to assist a design flow process described by the acronym AREA (Aspiration $-$ Requirements $-$ Ensurance/Execution $-$ Assurance/Analysis), newly introduced in the thesis. It proposes a high level description of processes and interactions from the system design stage until putting it into an operational/commercial environment.

Practical

A new description of the ATLAS TDAQ system was introduced, being the result of synthesizing the buffering and back-pressure mechanisms inside the system. Consolidated on these aspects, we created a performance coupling description with allowed times for back-pressure mechanisms to occur, which are indicators of the allowed latencies per sub-systems. We showed that the Level-2 system has two mechanisms of issuing back-pressure towards the Level-1 trigger, hence towards the detector, rendering it a highly sensitive filtering level.

Applying a mathematical description to the Level-2, we were able to obtain and analyze a relationship between the number of Level-2 processors, the probability that the system is full and the time required by Level-2 to process an event. We applied this relationship for different configuration and requirements scenarios, obtaining for the current system structure a Level-2 processing time of the order of tens of milliseconds.

Measuring the ΔQ for the ATLAS TDAQ data network allowed us to initially achieve a performance assessment of its network devices by employing a differential approach on the network paths and by making use of the compositional properties of ΔQ. The measurement process itself - a statistical one - is applicable in an operational environment, i.e. when physics data taking was running, utilizing lightweight tools negligible from the point of view of resources consumed.

The most important set of results is related to the V component of the ΔQ. We showed how it captures the queueing delay experienced by a packet, being a component dependent on the load

of the network and on the traffic creating that load, i.e. its pattern. We thus had the grounds and the means to initiate and carry on investigations on queue sizes and on the ATLAS TDAQ traffic pattern being generated by the ROS application. We utilized V with theoretical queueing models results revealing a big difference between the ATLAS traffic and the Poisson one, hence quantifying its level of burstiness.

The ΔQ allowed us in the end to identify predictive aspects of the network performance, mainly the traffic delay shaping at increased loads than the ones captured, showing the non-linearity of the waiting time with the load. Based on this information we obtained the delay characteristics at the network design average load (60%) and we showed that the Level-2 events processing is dependent on a small fraction of packets, falling under the fraction of 5% most delayed.

6.3 Future research

The practical results introduced earlier are exploiting only a small portion of the potential the observational model exposes. First of all, this model can be employed more widely to e.g. Internet Service Providers, fixed or mobile operators, where the absolute values for loss and delay are higher, though the same principles apply.

In this thesis, we applied it in a very particular context - the ATLAS TDAQ - characterized by unique requirements, structure and running conditions. The latencies and loss inside this system are extremely low and the tolerance for larger delays is manageable. These results however can be used to study system scalability and vicinity to saturation, mainly in the context of the envisaged system upgrade.

As future research steps, with applicability to the ATLAS TDAQ system, we identify:

- integration with the network and system monitoring tools or with an expert system used for fault analysis by picking the key observables and capturing their moving time series trend. This would allow a proactive detection of application degradation as opposed to the current tools which offer averaged values with a lag larger than 5 minutes. A certain increase in the instantaneous delay or delay variation is a sign that data suffers increased queueing and alarms or actions can be triggered.

- application of this type of performance analysis to other parts of the ATLAS TDAQ network, like the Event Building, which is heavily utilized and with a different traffic profile than Level-2. Network load reaches peak values of 90% which can be used efficiently to cross-check our results for Level-2. Also, in the Event Building sub-system the traffic shaping mechanism could be adjusted in a more knowledgeable manner, having as direct feedback the change in the ΔQ.

- refinement of the measurement tools, with a correction for clock drift, not only clock offset. The challenge would be to fit a function through the time series plots, which, as we saw, are non-linear.

- implementation of different and more precise observation points, like optical taps on the network devices, yielding a complete image of the network performance. Although an additional cost, this solution would allow constant and out-of-band performance analysis, with the associated benefits.

Bibliography

[1] Ivo Adan and Jacques Resing. *Queueing Theory*. Department of Mathematics and Computing Science, Eindhoven University of Technology, February 2002.

[2] Arnold O. Allen. *Probability, Statistics, and Queuing Theory*. Academic Press, Inc., 1250 Sixth Avenue, San Diego, CA 92101, second edition, 1990.

[3] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, New Jersey 07458, fourth and international edition, 2003. ISBN 0-13-038488-7.

[4] Herbert H. Hum Jaidev P. Patwardhan Greg J. Regnier Annie P. Foong, Thomas R. Huff. Tcp performance re-visited. *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2003.

[5] ATLAS – A Toroidal LHC ApparatuS. `http://atlas.web.cern.ch/Atlas`.

[6] S M Batraneanu, D H Campora, B Martin, D O Savu, S N Stancu, and L Leahu. Advanced Visualization System for Monitoring the ATLAS TDAQ Network in real-time. In *18th IEEE Real-Time Conference 2012*, Berkeley, California, Jun 2012.

[7] S. Bradner and J. McQuaid. Benchmarking Terminology for Network Interconnect Devices. RFC2544, May 1999. `http://www.ietf.org/rfc/rfc2544.txt?number=2544`.

[8] CERN – European Organization for Nuclear Research. `http://public.web.cern.ch/public/`.

[9] Baek-Young Choi, Sue Moon, Zhi-Li Zhang, Konstantina Papagiannaki, and Christophe Diot. Analysis of Point-To-Point Packet Delay In an Operational Network. IEEE INFOCOM, 2004.

[10] Matei Ciobotaru, Stefan Stancu, Micheal LeVine, and Brian Martin. GETB, a Gigabit Ethernet Application Platform: its Use in the ATLAS TDAQ Network. In *Proc. IEEE Real Time Conference 2005*, Stockholm, Sweden, June 2005. in press.

[11] M.D. Ciobotaru. *Characterizing, managing and monitoring the networks for the ATLAS Data Acquisition System*. PhD thesis, Universitatea "Politehnica" Bucureşti, 2007.

[12] Hideyuki Tokuda Clifford W. Mercer, Stefan Savage. Processor capacity reserves for multimedia operating systems. 1994.

[13] ATLAS Collaboration. ATLAS Level-1 Trigger Technical Design Report. *CERN/LHCC/-98-014*, 1998.

[14] LAN MAN Standards Committee. IEEE Std 802.1Q – Virtual Bridged Local Area Networks. IEEE Computer Society.

[15] LAN MAN Standards Committee. IEEE Std 802.3 – Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. IEEE Computer Society.

[16] Robert B. Cooper. *Introduction to Queueing Theory*. North Holland, 2nd edition, 1981.

[17] R.W. Dobinson, M. Dobson, S. Haas, B. Martin, M. J. LeVine, and F. Saka. IEEE 802.3 Ethernet, Current Status and Future Prospects at the LHC. *CERN open-2000-311*, October 2000. `http://doc.cern.ch/archive/electronic/cern/preprints/open/open-2000-311.pdf`.

[18] R.W. Dobinson, S. Haas, and B. Martin. Ethernet for the ATLAS Second Level Trigger? March 1998.

[19] M Dobson, M Ciobotaru, E Ertorer, H Garitaonandia, L Leahu, M Leahu, I M Malciu, E Panikashvili, A Topurov, and G Ünel. The Architecture and Administration of the ATLAS Online Computing System. In *15th International Conference on Computing In High Energy and Nuclear Physics*, pages 114–119, Mumbai, India, Feb 2006.

[20] Ethereal. `http://www.ethereal.com/`.

[21] Golonka P., Korcyl K., Saka F. Modeling large Ethernet networks for the ATLAS high level trigger system using parameterized models of switches and nodes. *CERN-OPEN-2001-061*, June 2007.

[22] Green B., Misiejuk A., Strong J.A, Kieft G., Krause E., Kugel A., Mï¿½ller M., Yu M., Wasen J.v. ATLAS TDAQ/DCS ROS ROBIN Design Document (RDD). May 2004.

[23] ATLAS Computing Group. ATLAS Computing Technical Design Report. *CERN/LHCC/2005-022*, July 2005. `http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf`.

[24] ATLAS HLT/DAQ/DCS Group. ATLAS High-Level Trigger Data Acquisition and Controls Technical Design Report. *CERN/LHCC/2003-022*, October 2003.

[25] ATLAS TDAQ Steering Group. Trigger & Daq Interfaces With Front-end Systems: Requirement Document. *DAQ-NO-103*, JUN 1998.

[26] Ricardo Gusella. *The Characterization of Variability of Packet Arrivals in Networks of High-Performance Workstations*. PhD thesis, Computer Science Division (EECS), University of California, 1990.

[27] Riccardo Gusella. Characterizing the variability of arrival processes with indices of dispersion. TR-90-051, September 1990.

[28] Haskell. `http://www.haskell.org/haskellwiki/Haskell`.

[29] ICMP. `http://tools.ietf.org/html/rfc792/`.

[30] J. MacGregor Smith. *M/G/c/K blocking probability models and system performance*. Department of Mechanical and Industrial Engineering, University of Massachusetts, Marston Hall, Room 111B, Amherst, MA 01003, USA, September 2002.

[31] J. Schlereth. Level 2 Supervisor Design. *DC-046*, July 2002.

[32] Golonka P. et al Korcyl K., Cranfield R. Computer modeling the atlas trigger/daq system performance. *ATL-DAQ-2003-040*, May 2005.

[33] L Leahu, N Davies, and D A Stoichescu. Performance vectors for data networks obtained through statistical means. *The Scientific Bulletin, University POLITEHNICA of Bucharest*, Series C. In Press.

[34] G. Lehmann. *Data Acquisition and Event Building Studies for the ATLAS Experiment*. PhD thesis, Universität Bern, JUN 2000.

[35] Reqirements for Multicast Protocols. `http://www.faqs.org/rfcs/rfc1458.html`.

[36] The Higgs Boson. `http://www.exploratorium.edu/origins/cern/ideas/higgs.html`.

[37] The Large Hadron Collider Project. `http://lhc.web.cern.ch/lhc/general/gen_info.htm`.

[38] J. W. Layland; C. L. Liu. Scheduling algorithms for multiprogramming in a hard real time environment. *Journal of ACM*, 20:46–61, 1973.

[39] B. Martin, A. Al-Shabibi, S.M. Batraneanu, M.D. Ciobotaru, G.L. Darlea, L. Leahu, and S.N. Stancu. Advanced monitoring techniques for a large scale data processing network. *TERENA Newtorking Conference*, 2008.

[40] R. Milner. *A Calculus of Communicating Systems*, volume 92. Springer-Verlag, 1980.

[41] R. Milner. *Communication and Concurrency*. Number ISBN 0 13 115007 3 in PHI Series in Computer Science. Prentice Hall, 1989.

[42] NTP: the Network Time Protocol. `http://www.ntp.org`.

[43] Konstantina Papagiannaki, Sue Moon, Chuck Fraleigh, Patrick Thiran, and Christophe Diot. Measurement and Analysis of Single-Hop Delay on an IP Backbone Network. Proceedings of INFOCOM, 2002.

[44] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkley, April 1997.

[45] Ping Linux Manual. `http://linux.die.net/man/8/ping`.

[46] Predictable Network Solutions Ltd. `http://www.pnsol.com`.

[47] J. Postel. Transmission Control Protocol. RFC0793, September 1981. `http://www.ietf.org/rfc/rfc0793.txt?number=793`.

[48] The Python Programming Language. `http://www.python.org/`.

[49] R. Rajkumar. Synchronization in real-time systems: A priority inheritance approach. Technical report, Kluwer Academic Publishers, 1991.

[50] Robert Blair, John Dawson, Gary Drake, William Haberichter, James Schlereth, Jinlong Zhang. The ATLAS High Level Trigger Region of Interest Builder. *ATL-DAQ-PUB-2007-001*, November 2007.

[51] R.W.Wolff. *Poisson arrivals see time averages*. Opns. Res, 1982.

[52] R. Rajkumar; C. Lee; J. Lehoczky; D. Siewiorek. A resource allocation model for qos management. 1997.

[53] R Sjoen, S Stancu, M Ciobotaru, S M Batraneanu, L Leahu, B Martin, and A Al-Shabibi. Monitoring individual traffic flows within the ATLAS TDAQ network. In *17th International Conference on Computing in High Energy and Nuclear Physics*, volume 219, page 052013, Prague, Czech Republic, 2010.

[54] Scientific Linux @ CERN. `http://linux.web.cern.ch/linux/scientific.shtml`.

[55] CERN S-LINK homepage. `http://hsi.web.cern.ch/HSI/s-link/`.

[56] S. Stancu. *Networks for the ATLAS LHC Detector: Requirements, Design and Validation*. PhD thesis, Universitatea "Politehnica" Bucureşti, JUL 2005.

[57] S Stancu, M Ciobotaru, L Leahu, B Martin, and C Meirosu. Networks for ATLAS Trigger and Data Acquisition. In *15th International Conference on Computing In High Energy and Nuclear Physics*, pages 605–608, Mumbai, India, Feb 2006.

[58] Stefan Stancu, Matei Ciobotaru, and David Francis. Relevant features for dataflow switches, April 2005. `http://sstancu.home.cern.ch/sstancu/docs/sw_feat_noreq_v0-5.pdf`.

[59] Tcpdump. `http://www.tcpdump.org/`.

[60] TTC System. `http://ttc.web.cern.ch/TTC/`.

[61] M. Ciuc; C. Vertan. *Prelucrarea Statistica a Semnalelor*. Editura MatrixRom, Bucharest, 2005.

[62] Wireshark. `http://www.wireshark.org/`.

Appendix A

Queuing Theory

A.1 PASTA Property

In a queuing system, each newly arriving "customer" finds the system in a particular state - S. A possibility is to consider this state as given by the number of busy servers. A very interesting property, true only for Poisson arriving streams but for a general distribution of service time, is called PASTA (**P**oisson **A**rrivals **S**ee **T**ime **A**verages).

It states that customers which arrive in the system find on average the same situation as an external observer which looks at the system at arbitrary points in time. In terms of system states, this property says that the fraction of customers finding the system in state S is equal to the fraction of time the system is actually in that state S. Through the "eyes" of Poisson arriving customers, we can obtain mean values for the situation of the system itself, reason for which this property is also referred as ROP (**R**andom **O**bserver **P**roperty)

We mentioned that this property holds only for Poisson streams. Let's look at an example for deterministic arrivals: users come to work on a computer at the beginning of each hour and each of them always uses it for half an hour. The busy periods for the computer are illustrated in the top part of Figure A.1. Although the computer is busy for 50% of the total time, each arriving customer finds the computer idle, hence 0% of the time is loaded from its point of view.

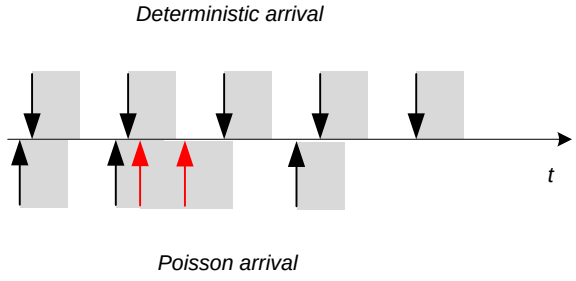

Figure A.1: Load seen by Deterministic and Poisson arrivals

Due to the nature of Poisson processes, the long-term probability of an user finding the computer busy increases, see the bottom part of Figure A.1. It is rigorously proved in [51] that this probability equals the fraction of time the system is really busy. The very basic idea is that a Poisson arrival process "scans" the state space of the system in such a way that for long term it reveals the correct proportion of states.

A.2 Blocking probabilities in M/M/1/K and M/M/1 systems

The blocking probability for a finite queuing system is the probability that there is no room in the system for an arriving customer. For the M/M/1/K system, the blocking probability is the probability of having *K=n* customers in the system:

$$P(N = n) = \frac{(1-\rho)\rho^n}{1-\rho^{n+1}} \tag{A.1}$$

where $\rho < 1$ is the loading factor, i.e. the arrival rate multiplied by the average service time.

For the M/M/1 queuing system there is no blocking probability because the queue size is infinite and there will always be room for another customer. In order to compare it with the M/M/1/K system, we can compute the probability of having more than *K=n* customers in the system.

$$P(N \geqslant n) = \rho^{n+1} \tag{A.2}$$

The probability of having more than *n* customers in a infinite queue system is equivalent to the probability of having a finite queue system, of size *n*, full. Therefore, the difference between relation A.2 and A.1 gives us the absolute error between the two models, with respect to the probability masses:

$$\varepsilon = \rho^{n+1} - \frac{(1-\rho)\rho^n}{1-\rho^{n+1}} \tag{A.3}$$

The absolute error depends on two variables: *n* and ρ. The differentials with respect to each of the variables are:

$$\frac{\delta\varepsilon}{\delta n} = \frac{\rho^n \ln\rho(2\rho - \rho^{n+2} - 1)}{(1-\rho^{n+1})^2} \tag{A.4}$$

$$\frac{\delta\varepsilon}{\delta\rho} = \frac{\rho^{n+1}}{(1-\rho^{n+1})^2}\left[2(n+1)\rho^{2n+3} - (n+1)\rho^{2(n+1)} - 2(n+1)\rho^{n+2} - \rho^{n+1} + 2(n+1)\rho - n\right] \tag{A.5}$$

Appendix B

Support for Observational Model

B.1 Dirac delta functions convolution

A Dirac delta function δ is defined by:

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{B.1}$$

A delta function shifted with a is consequently defined by:

$$\delta(x-a) = \begin{cases} +\infty, & x = a \\ 0, & x \neq 0 \end{cases} \tag{B.2}$$

Relation B.2 defines the distribution associated to a random variable taking value a only. For the particular case of G component introduced in Section 4.4.2, if G is a constant G_A for a particular network segment, its PDF is defined by:

$$f_A(x) = \delta(x - G_A) \tag{B.3}$$

Topologically convolving two network segments which have their G constant is reduced to the convolution of e.g. f_A and f_B:

$$f_{AB} = f_A \otimes f_B \tag{B.4}$$

defined by:

$$f_{AB}(x) = \int_{-\infty}^{+\infty} f_A(\tau) f_B(x-\tau) d\tau \tag{B.5}$$

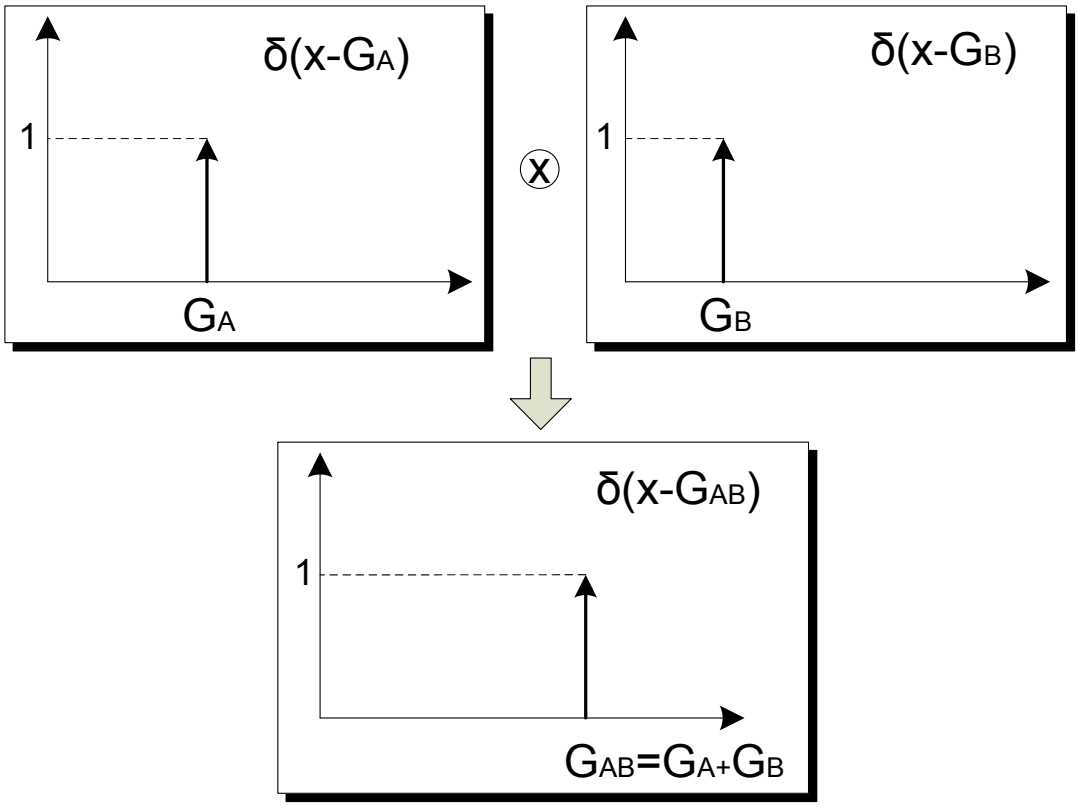

Figure B.1: Delta functions convolution exemplified for G_A and G_B

Knowing that $f_A(\tau)$ and $f_B(\tau)$ take a non-zero value, i.e. 1, only when $\tau = G_A$ and $\tau = G_B$ respectively, we obtain:

$$f_{AB}(x) = f_B(x - G_A) = \begin{cases} +\infty, & x - G_A = G_B \\ 0, & x - G_A \neq G_B \end{cases} \tag{B.6}$$

equivalent to:

$$f_{AB}(x) = \begin{cases} +\infty, & x = G_A + G_B \\ 0, & x \neq G_A + G_B \end{cases} \tag{B.7}$$

This proves that convolving the distributions of two constants results in the distribution of the sum of those constants, as presented in Figure B.1 for G_A and G_B, meaning that:

$$G_{AB} = G_A + G_B \tag{B.8}$$

B.2 Clock synchronization

When measuring the delay of a packet sent from A to B we use the local clocks for time-stamping. These exhibit skewness due to an independent relativity to an absolute clock. A typical measurement of the delay is illustrated in Figure B.2 for both directions, where:

- t_i is the relative clock at A when sample i was sent
- $t_i + \triangle$ is the relative clock at B when sample i was sent. $\triangle$ is thus the skew between A and B at a given moment in time.
- δ_i is the *real* delay suffered by the sample in transit.

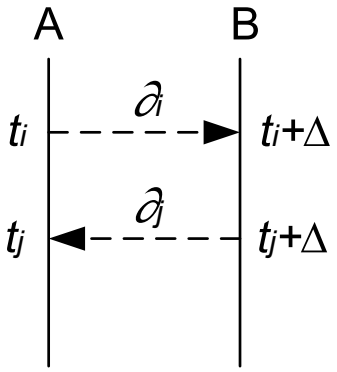

Figure B.2: Measuring the delay with clock skewness

The *observed* transit time $\overrightarrow{AB_i}$ for i^{th} packet is:

$$T_{\overrightarrow{AB_i}} = (t_i + \triangle) - t_i + \delta_i + \chi_i = \triangle + \delta_i + \chi_i \tag{B.9}$$

while in the reverse direction for j^{th} packet is:

$$T_{\overrightarrow{BA_j}} = t_j - (t_j + \triangle) + \delta_j + \chi_j = -\triangle + \delta_j + \chi_j \tag{B.10}$$

where χ is a random variable capturing the randomness from the rest of the transit.

We present an algorithm to calculate the skew $\triangle$ using a large number of samples and under the following assumptions:

- the clock skew is varying slowly as compared with the sampling rate
- the end-to-end delay is a linear function of some quantization or the packet length, i.e.:

$$\delta = c + m \times length \tag{B.11}$$

or

$$\delta = c + m \times quantization_{length} \tag{B.12}$$

The strategy is to sample sufficiently often the random component χ in order to capture moments when it yields "zero" values. We also consider that in the absence of any other information the minimum delay is symmetric, i.e. equal in both directions.

We gather a collection of One-Way Delay values for both the directions:$\overrightarrow{AB}$ and $\overrightarrow{BA}$, each consisting of a population of

$$\triangle + \delta_i(size) + \chi_i$$

for $\overrightarrow{AB}$ and

$$-\triangle + \delta_i(size) + \chi_i$$

for $\overrightarrow{BA}$ respectively.

On each set of data we apply a procedure which we name it Fit, for $\overrightarrow{AB}$ consisting of the following steps:

- arrange the population by the packet size and obtain an array of

$$\{pktsize,\, [\triangle + \delta_i(size) + \chi_i | size = pktsize]\}$$

- construct the sample population

$$\{pktsize,\, min\, [\triangle + \delta_i(size) + \chi_i | size = pktsize]\}$$

- under the assumption that we will sample a $\chi_i = 0$, we obtain a population for analysis of

$$\{(pktsize,\, \triangle + \delta_i(pktsize)\}$$

- we use linear regression to fit a line through this population:

$$y = y_0 + m \times x$$

  with the parameter $y_0 = \triangle + c$ and m from the equation B.12.

Running this procedure for both directions, we obtain y_0 in both cases:

$$Fit(\overrightarrow{AB}) \rightarrow y_{0AB} = \triangle + c$$

$$Fit(\overrightarrow{BA}) \rightarrow y_{0BA} = -\triangle + c$$

From here we can calculate:

$$\triangle = \frac{y_{0AB} - y_{0BA}}{2}$$

$$c = \frac{y_{0AB} + y_{0BA}}{2}$$

$\triangle$ represents the clock skew and c represents the G component of the ΔQ introduced in Chapter 4.

B.3 Python random number generators

Python provides *Pseudo-Random Number Generators (PRNGs)* in a module called *random.* We will show the generator accuracy for two distributions: *uniform* and *exponential.*

B.3.1 Uniform distribution

A random number with uniform distribution in the interval $[a, b]$ is generated by the function:

```
random.uniform(a,b)
```

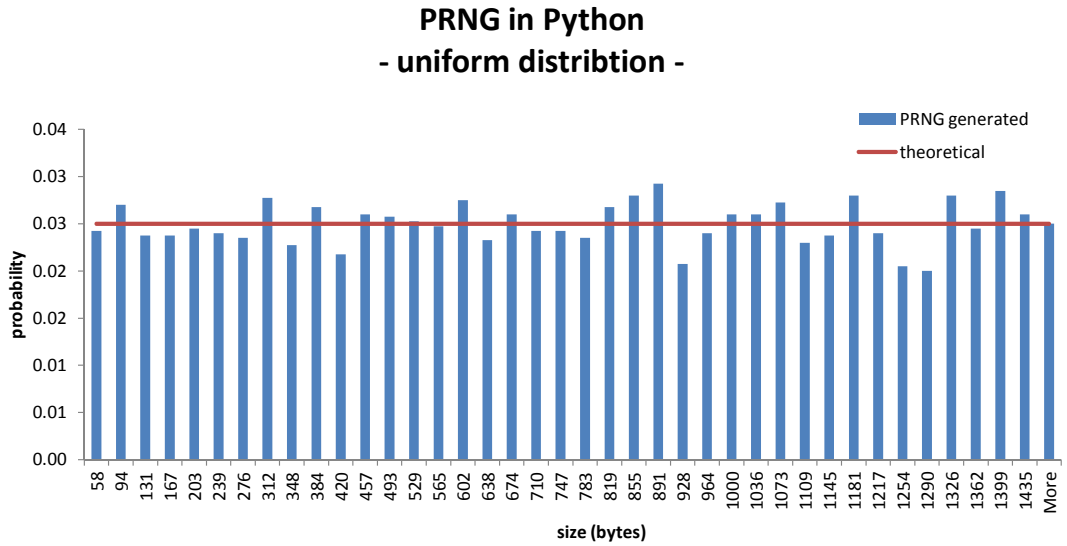

Figure B.3: PRNG in Python - uniform distribution

Figure B.3 presents the Python's PRNG results for uniform distribution in the interval [36, 1514]. These number were used as probe frame sizes when performing the methodology described in Section 4.4.5. We can notice the small deviation from the theoretical distribution, acceptable for our purposes.

B.3.2 Exponential distribution

A random number following an exponential distribution with parameter λ is generated in Python by the function:

```
random.expovariate(lambd)
```

Figure B.4 presents the Python's PRNG results for exponential distribution having a $\lambda = 0.4\,ms^{-1}$. These number were used as inter-probe intervals when performing the methodology described in Section 4.4.5.

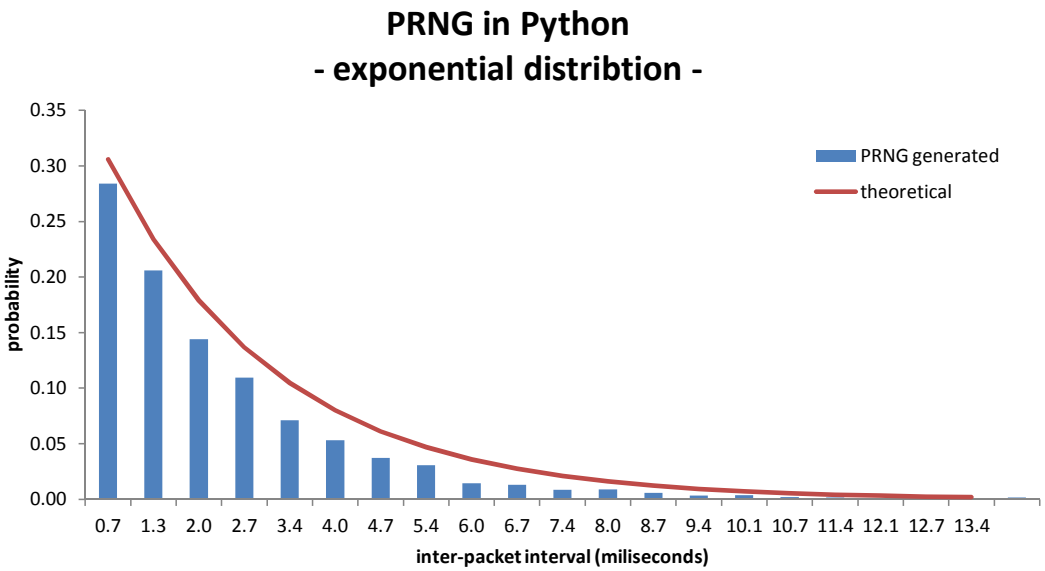

Figure B.4: PRNG in Python - exponential distribution