Data-flow Performance Optimisation on Unreliable Networks: the ATLAS Data-Acquisition Case

Tommaso Colombo on behalf of the ATLAS Collaboration

CERN, CH-1211 Geneva 23, Switzerland and Institut für Technische Informatik (ZITI), Universität Heidelberg, B6 26, 68131 Mannheim, Germany

E-mail: Tommaso.Colombo@cern.ch

Abstract. The ATLAS detector at CERN records proton-proton collisions delivered by the Large Hadron Collider (LHC). The ATLAS Trigger and Data-Acquisition (TDAQ) system identifies, selects, and stores interesting collision data. These are received from the detector readout electronics at an average rate of 100 kHz. The typical event data size is 1 to 2 MB. Overall, the ATLAS TDAQ system can be seen as a distributed software system executed on a farm of roughly 2000 commodity PCs. The worker nodes are interconnected by an Ethernet network that at the restart of the LHC in 2015 is expected to experience a sustained throughput of several 10 GB/s.

A particular type of challenge posed by this system, and by DAQ systems in general, is the inherently bursty nature of the data traffic from the readout buffers to the worker nodes. This can cause instantaneous network congestion and therefore performance degradation. The effect is particularly pronounced for unreliable network interconnections, such as Ethernet.

In this paper we report on the design of the data-flow software for the 2015–2018 data-taking period of the ATLAS experiment. This software will be responsible for transporting the data across the distributed Data-Acquisition system. We will focus on the strategies employed to manage the network congestion and therefore minimise the data-collection latency and maximise the system performance. We will discuss the results of systematic measurements performed on different types of networking hardware. These results highlight the causes of network congestion and the effects on the overall system performance.

1. Introduction: the ATLAS Trigger and Data-Acquisition system in 2015–2018

ATLAS [1] is one of the experiments installed at the Large Hadron Collider (LHC), CERN, Geneva, Switzerland. It observes proton-proton collision events delivered by the LHC at a design frequency of 40 MHz. Each event corresponds to a data size of 1 to 2 MB. The ATLAS Trigger and Data-Acquisition (TDAQ) system [2] is responsible for the selection of interesting events, reducing the initial frequency of 40 MHz to ~ 1 kHz of stored events. This requires an overall trigger rejection factor of the order of 10^4 against minimum bias events, while retaining the rare new physics processes, such as Higgs boson decays. The TDAQ system, outlined in Figure 1, is based on the combination of a hardware-based first-level trigger and a software-based High-Level Trigger.

The first-level trigger (Level-1) is a synchronous pipelined electronics system, with a guaranteed maximum latency of 2.5 µs. It selects events using coarse-grained data from the calorimeters and the muon spectrometers. It triggers the detector readout at a maximum rate of 100 kHz, imposing the necessary dead-times to protect the detector front-end buffers. As part

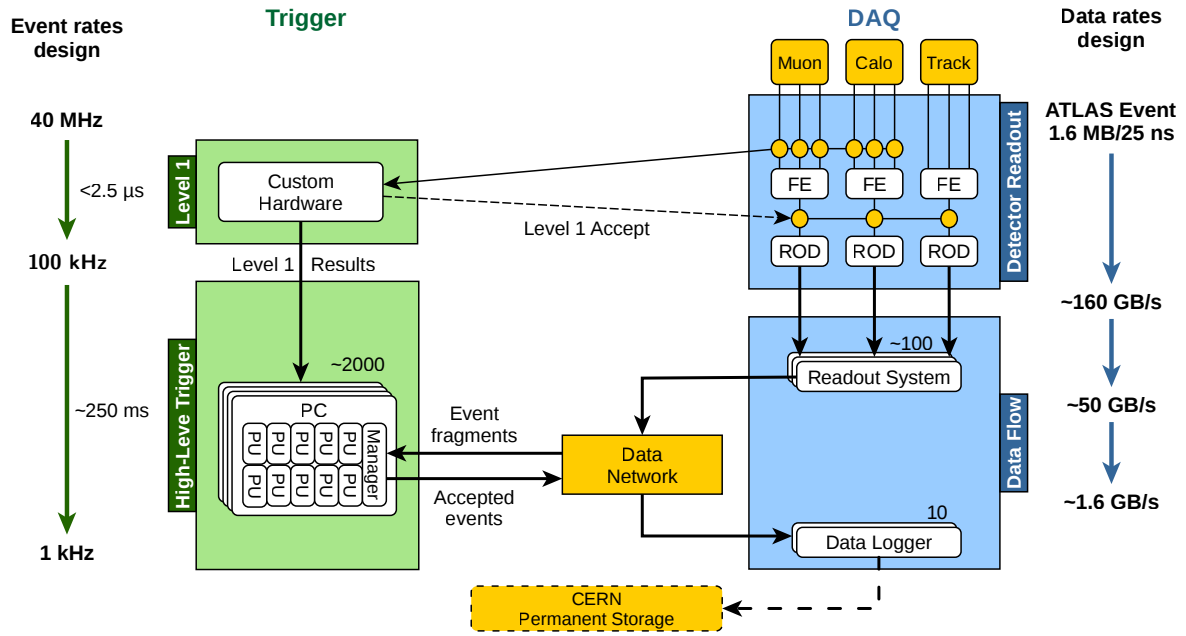

Figure 1. Outline of the ATLAS Trigger and Data-Acquisition system, stating the event and data rates foreseen for the 2015–2018 data-taking period.

of the selection process, it identifies a so-called region-of-interest (RoI): a part of the detector that recorded interesting signals that is used as a seed for the High-Level Trigger.

The High-Level Trigger (HLT) is a distributed software system running on around 2000 PCs interconnected by an Ethernet network. When an event is accepted by the Level-1, its data fragments (1860 fragments of variable sizes around 1 kB) are distributed, using custom optical links, to hardware buffers in the Readout System (ROS) nodes. At the same time, region-of-interest information is assigned to one of the HLT Processing Units. Each HLT worker node hosts one HLT Processing Unit per CPU core. Using the region-of-interest as a starting point, the Processing Unit incrementally retrieves and analyses event fragments, until a decision can be taken. The event can be rejected even without analysing all its fragments, thus limiting the fraction of data to be retrieved. Fragments of rejected events are deleted from the ROS buffers, while accepted events are transferred to one of the Data Logger nodes for storage. The rate of events that can be accepted is mostly determined by the availability of offline computing and storage resources, and is foreseen to be around 1 kHz for the 2015–2018 data taking period.

The physical layout of the Data-Acquisition and High-Level Trigger system is represented in Figure 2. The core of the system consists of two large network routers with a maximum capacity of several hundred 10GbE ports. ROS nodes are directly connected to both core routers with 4 10GbE links. HLT nodes are organised in racks of at most 40 nodes. Each node in a rack is connected to an aggregation switch with a GbE link. The rack switches have 10 GbE links to both core routers. For the rationale behind this design, please refer to [3].

2. Performance challenges: network congestion

The latency of the data transfer between the ROS and the HLT Processing Units (i.e. the data collection latency) is critical to the performance of the whole system: each of the HLT Processing Units operates exclusively on an event assigned to it, with the HLT selection proceeding iteratively starting from the region-of-interest identified by the Level-1, collecting data fragments incrementally as needed. A Processing Unit is blocked while it waits for the data fragments to

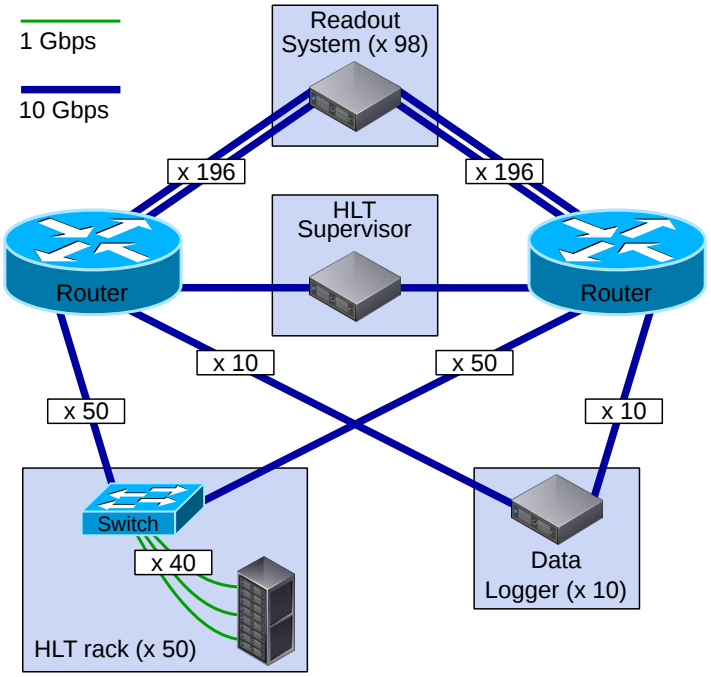

Figure 2. Network architecture of the ATLAS Data-Acquisition and High-Level Trigger system.

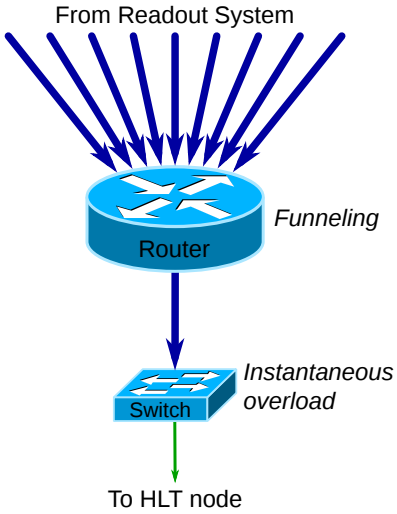

Figure 3. Visualisation of the network congestion.

be collected, which means that the data collection latency effectively translates into lost CPU time.

Data transfers from the ROS are naturally bursty. Event fragments are striped over all the ROS nodes, since each node buffers data from a specific region of the detector. A single HLT Processing Unit usually requests fragments from multiple ROS nodes at the same time. As the fragments are already available in the ROS buffers and are sent as soon as the request reaches the ROS, many nodes will start sending fragments at the same time to the same destination, thus creating instantaneous network congestion (see Figure 3).

As mentioned in Section 1, the network technology used throughout the HLT system is

Ethernet. The messaging among applications is based on the TCP protocol. Ethernet provides an "unreliable" transport for network packets. In particular, packets can be dropped in response to a variety of conditions, including network buffer overflows caused by congestion. TCP provides reliable connections by detecting dropped packets and retransmitting them. However, this comes at a price: TCP has both an active packet drop detection mechanism, in which the receiver detects the lost packet and causes the sender to react very quickly, and a passive mechanism, which is based on relatively long timeouts ($\geq$ 200 ms) on the sender side [4]. Unfortunately, in cases of network congestion, the passive mechanism plays a much more important part than the active mechanism. Hence, network congestion can radically impact the data collection latency and therefore the overall performance of the system.

3. Preventing packet drops: a client-side traffic shaping algorithm

Smoothing the rate of data requests generated by a HLT node can alleviate the network congestion by controlling the maximum size of the traffic burst from the ROS. Obviously such a smoothing mechanism imposes a trade-off: excessive smoothing can increase the data collection time by unnecessarily delaying the requests for data, whereas insufficient smoothing will not eliminate packet drops.

For the measurements presented here, a credit-based traffic shaping algorithm was used. Its basic rules are as follows.

- Each HLT node has a fixed number of credits available.
- Each data request to a ROS node uses as many credits as the number of fragments it asks for.
- Each response returns the credits used by the corresponding request.
- If all available credits are used, further requests are blocked until the necessary credits become available.

Since all event fragments are similar in size, the number of fragments in a request gives a rough estimation of the size of the corresponding response. Therefore, this algorithm effectively limits the maximum burst size of data transfers directed to the same HLT node.

4. Measuring the effects of congestion

Detecting buffer overflows in an Ethernet network is a relatively simple task: most switches publish cumulative counts of dropped packets per port and the Linux kernel provides the total number of packet retransmissions that have occurred in a TCP connection. However, the relationship between the dropped packets count and the overall application performance is rather complex, especially when using a protocol with congestion control features, such as TCP. The parameter space to investigate is very large and the monitoring information (e.g. instantaneous buffer occupancies) scarce or unavailable. Therefore, generating synthetic traffic patterns, which can be tightly controlled and are known in advance, proves instrumental in investigating the performance profile of the network. The tests described in the following paragraphs employ this approach.

The key application performance metric in the ATLAS HLT case is the total data collection latency per event: as mentioned in Section 2, data collection directly translates to an inefficiency source for the HLT farm.

4.1. Test set-up

The tests were performed using some of the hardware available in the ATLAS TDAQ system. At the time of these tests, the TDAQ infrastructure was still under consolidation. Hence the test set-up is slightly different from the one described in Section 1. In particular:

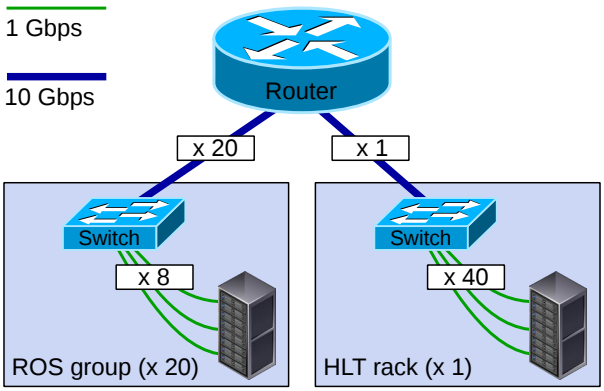

Figure 4. Test set-up.

- ROS nodes were not directly connected with 10GbE links to the core routers; they were instead connected with GbE links to an intermediate aggregation switch with a 10GbE uplink to the core (the intermediate switch was kept under-subscribed so that no congestion could appear there).
- HLT racks are only connected to one of the two core routers.

Given these constraints, the following test set-up, shown in Figure 4, was chosen:

- 20 ROS groups:
  - Each group consists of 8 nodes connected to one switch (160 total).
  - 12 event fragments of 1.1 kB are served by each node (1920 total fragments, 2112 kB full event size).
- 1 HLT rack:
  - It consists of 39 PCs connected to one switch.
  - Each PC hosts 24 HLT Processing Units (936 total).
- The uplinks from the switches connect to a single core router.

This configuration was chosen because it provides with a realistic model of the expected network congestion phenomena in the final system topology. In particular, the congestion at the rack-level switch is well represented.

The core router uses input-buffering with a peculiar implementation of switch-level virtual output queueing: input ports are grouped in modules of 8 ports at most and each module maintains multiple, distinct queues to every output port on the router. Buffer space is abundant: 1.5 GB per module, shared among the queues.

The top-of-rack switches use output-buffering. Two different models were tested:

- a switch with small (600 kB), per-port dedicated buffers
- a switch with a small (a few MB) buffer, shared among all ports
- a switch with a large (1.25 GB) buffer, shared among all ports

The results obtained with such a set-up should scale reliably with the higher number of HLT racks in the complete system: the network congestion phenomena individually affect the output buffers of the network ports that are actually in use.

4.2. Simple test scenario: full event building

A very simple traffic pattern is the so-called full event building:

- Events are assigned to HLT Processing Units at a constant rate.
- The Processing Units immediately collect all fragments of assigned events.

It should be noted that this pattern is the harshest in terms of generated traffic bursts: since the data corresponding to an event is collected all at once, the maximum burst size corresponds to the event size.

The assignment rate parameter for this test was selected with the goal of utilising roughly 90% of the available bandwidth of the HLT rack uplink: at the chosen 500 Hz assignment rate, the total rack input bandwidth was ~ 1.1 GB/s.

The results of the measurements are shown in Figure 5. The total data collection time per event (shown in the plots on the left) is influenced both by the network conditions and by the traffic shaping mechanism: with few traffic shaping credits available, the large data-collection time is due to collection inefficiency because the HLT nodes cannot fully utilise the network bandwidth; with many traffic shaping credits the network congestion determines the performance.

It is possible to estimate the minimum possible value for the data collection latency to use as a reference point. For the sake of simplicity, the time it takes for data requests to reach the ROS nodes and the time it takes for the ROS to prepare the data can be neglected[1]. The dominant component is given by the so-called packetisation delay of the data, i.e. the amount of time it takes to transmit the data on the slowest link in the path. In this set-up, this corresponds to the amount of time it takes to transmit a full event on the GbE link from the rack-level switch to the HLT node, which is ~ 18 ms.

When the switch with small per-port buffers is used, each buffer can be though of as belonging to the HLT node connected to its switch port. As the traffic bursts directed to an HLT node get bigger than the size of the buffer, packets are dropped: the maximum and average round trip-times (shown in the plots on the right) increase steeply and the data collection time is accordingly influenced. Since fixed-size fragments were used for these tests, the maximum number of credits assigned to each HLT node translates directly to the size of the maximum allowed traffic burst (e.g. 200 credits, with a 1.1 kB fragment size, translate to a burst size of 220 kB plus protocol overhead). As expected, the minimum data collection latency results from a configuration with 500 credits (i.e. a maximum burst size of 550 kB plus protocol overhead) in which the buffers are used almost up to their maximum capacity, but are never overflowed.

The results using the switch with a small shared buffer is less easily interpreted: the burst size is not the only relevant parameter. The shared buffer is bigger than the ~ 2 MB full event size, so, in principle, for a single burst no overflows can occur. However, since the buffer is shared, the dynamics of the system must be considered: the shared buffer is filled with event data by a 10GbE input and emptied by multiple GbE outputs, handling one event each. Therefore, multiple traffic bursts, corresponding to different events targeting different outputs, will overlap in the shared buffer, requiring a buffering capacity that is several times higher than the maximum burst size. As a consequence, also in this case the minimum data collection latency results from a configuration with a maximum burst size reduced using traffic shaping.

5. Summary

A particular challenge to the performance of the ATLAS TDAQ system is the inherently bursty nature of the data traffic from the ROS buffers to the HLT nodes. On the TDAQ Ethernet network, this can lead to packet drops which cause very high data collection latencies, an inefficiency source for the system. Traffic shaping techniques can mitigate these phenomena.

[1] Measurements show that both time intervals are smaller than 0.5 ms.

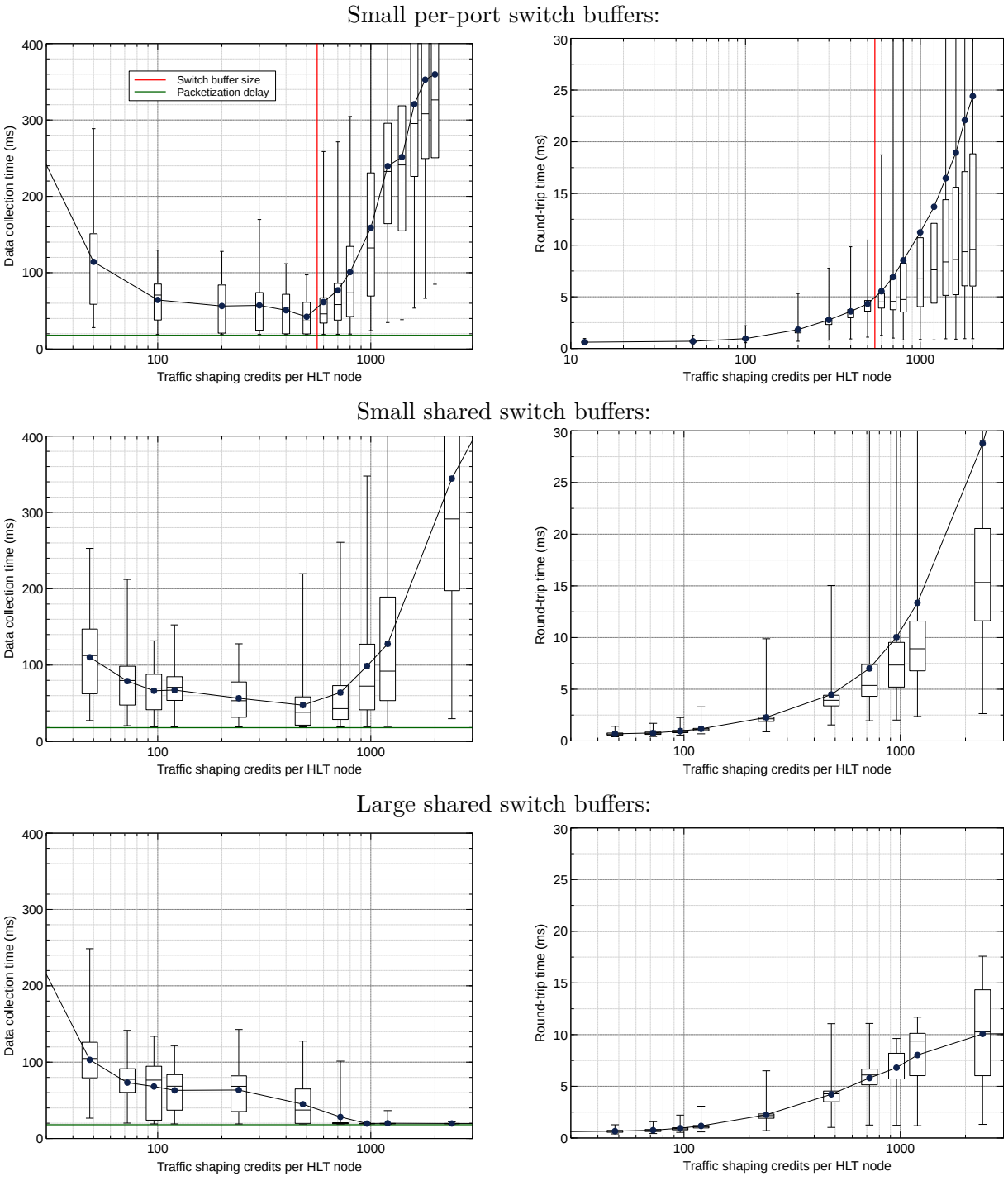

Figure 5. Data-Collection time and round-trip time as a function of the number of traffic shaping credits for the three different switch models under study. The test conditions are detailed in the text. The bullets represent the average values. The horizontal box lines represent the first quartile, the median, and the third quartile. The box whiskers represent the first and the 99th percentile.

As shown in this paper, the buffering space available in the network device plays a crucial role. When a switch with large buffers is used, packets are never dropped, even without traffic shaping: the data collection latency can get close to the minimum value allowed by the network topology. If a switch with buffers too small to handle the bursts is used, traffic shaping is effective in reducing the data collection latency by preventing packet drops. However, even in the best case, the average data collection latency does not reach its minimum possible value.

As a significant price premium is placed on network devices with large buffers, optimising the system performance becomes a balancing exercise among the cost of a better performing network, the cost of developing better traffic shaping techniques, and the cost of the inefficiencies introduced by network congestion.

6. Further investigations

The behaviour of an Ethernet network subject to congestion can be studied by injecting controlled traffic patterns and monitoring the relevant application performance metrics. However, gaining a deeper understanding of the congestion effects on a commercial system is a daunting task, as many useful metrics, such as queue occupancies, are either unavailable or too coarse. An alternative solution is the development of simulation models, which allow deeper inspection. Currently, models of the ATLAS TDAQ system are being developed, with the goal of enabling further investigations without having to deal with the limitations of testing on a production system. Once the models are proven capable of reproducing the known behaviour of the system, their results will be used to guide the optimisation of the network usage patterns.

References

[1] ATLAS Collaboration 2008 *JINST* **3** S08003

[2] ATLAS Collaboration 2003 *ATLAS High-Level Trigger, Data-Acquisition and Controls Technical Design Report* ATLAS-TDR-016 CERN-LHCC-2003-022 CERN

[3] Pozo Astigarraga M E *in these proceedings*

[4] Allman M, Paxson V, and Stevens W *TCP Congestion Control* RFC 2581 IETF