Investigation of the Networking Performance of Remote Real-Time Computing Farms for ATLAS Trigger DAQ

B. Caron, R. Hughes-Jones, K. Korsyl, C. Meirosu, and J. L. Neilsen

***Abstract*—To test the feasibility of using remote farms to perform real-time event selection in the Trigger/Data Acquisition System for the ATLAS experiment at CERN, a Proof of Concept was set up during 2004. The behavior of the request-response protocol to move application data has been measured for remote farms connected with different Wide Area Networks including a dedicated lightpath, a Virtual Private Network, and the standard production network. The dynamics and effect of using TCP/IP as the transport protocol has also been investigated for this real-time application. These data are compared with conventional bulk data transfers and used to validate the observed performance of the online farms and estimate the effect of this traffic pattern on the Wide Area Network.**

***Index Terms*—Protocols, real time systems, TCP, wide area networks.**

I. Introduction

SEVERAL experiments, including ATLAS at the Large Hadron Collider (LHC) and D0 at Fermi Lab, have expressed interest in using remote computing farms for processing and analysing the information from particle collision events. Different architectures have been suggested, ranging from pseudo-real-time file transfer and subsequent remote processing, to the real-time requesting of individual events described here.

To test the possibility of using remote computing farms for real-time processing within the ATLAS experiment, a collaboration [1] was set up between members of ATLAS Trigger/DAQ, Canarie, DARENET, Netera, PSNC, UKERNA and Dante to demonstrate a Proof of Concept and measure end-to-end network performance. The testbed shown in Fig. 1 was centred at CERN and used three different types of wide area high-speed network infrastructures to link the remote sites:

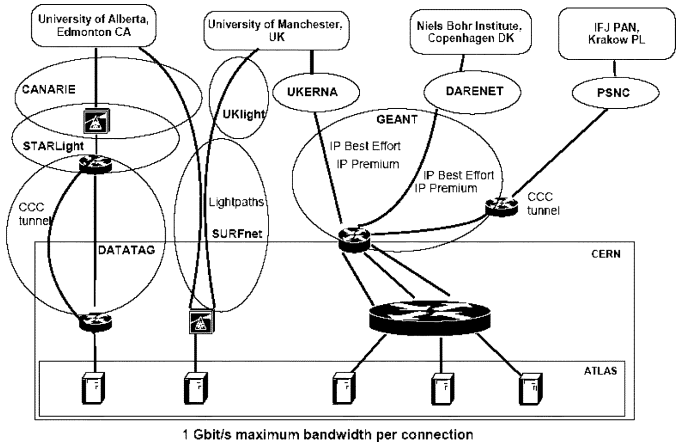

Fig. 1. The network configuration of the testbed.

- an end-to-end Ethernet over a Synchronous Digital Hierarchy, (SDH) lightpath[1] to the University of Alberta in Canada
- standard end-to-end Internet Protocol (IP) connectivity over the academic production network to the University of Manchester in the UK and the Niels Bohr Institute in Denmark
- a Virtual Private Network (VPN) composed of an MPLS tunnel over the GEANT network and an Ethernet VLAN over the Polish PIONIER network to IFJ PAN Krakow.

For potentially interesting particle interactions in ATLAS, the standard data acquisition system [2] records all the data fragments from the sub-detectors and builds them into full blocks of data (called events in the particle physics world) using code called the Sub-Farm Interface (SFI). These events are then fed to processors that form the Event Filter (EF) that analyse the detailed physics content of the event. Selected events are then recorded. As in the standard data acquisition system, the Proof of Concept used Transmission Control Protocol (TCP/IP) streams to move events from an SFI located at CERN to remote sites for Event Filter computation and then return the results. The remote EF processors used a request-response protocol to obtain the events from the SFI and return the result to the Sub-Farm Output (SFO). In the final experiment, a typical event will be ~1.5 Mbytes and EF computation times

Manuscript received June 13, 2005; revised April 24, 2006. This work was supported in part by the Particle Physics and Astronomy Research Council in the U.K., by the European Union under the IST-2001-33185 grant, and by the Polish Funding Agency under Grants 620/E-77/SPUB-M/CERN/P-03/DZ 110/2003-2005, and 620/E-77/SPB/5_PR UE/DZ 465/2002-2004.

B. Caron is with the University of Alberta, Edmonton, Canada.

R. Hughes-Jones is with the School of Physics and Astronomy, The University of Manchester, Manchester, M13 9PL, U.K. (e-mail: R.Hughes-Jones@manchester.ac.uk).

K. Korsyl is with IFJ PAN Krakow, Poland.

C. Meirosu is with CERN, Geneva, Switzerland, and with the "Politehnica" University of Bucuresti, Romania.

J. L. Neilsen is with the Niels Bohr Institute, Copenhagen, Denmark.

Color versions of Figs. 3-11 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2006.878895

[1]Here a lightpath means an end-to-end path over the network configured with a fixed and guaranteed bandwidth. The bandwidth of the path is not shared with other users and may be thought of as "circuit switched" rather than "packet switched" as in the conventional internet.

of the order of 1–2 s. To meet security requirements [3], the testbed at CERN used a private network. This required the use of Network Address Translation (NAT) systems to allow connections to be made to the remote farms over the production network. The remote farms connected using a lightpath or a VPN appeared as an extended local area network (LAN) and used the same IP network addresses as the systems at CERN.

Section II presents the methodologies used in testing the behavior of the network infrastructure, the Trigger/DAQ (TDAQ) application and its protocol. The results section begins by presenting the performance of the network components using User Datagram Protocol (UDP/IP). This is followed by and evaluation of the application request-response protocol over the testbed using TCP/IP as in the TDAQ application. Finally the operation of the real TDAQ application software over the testbed is presented.

A comprehensive discussion on potential scenarios using remote computing farms with the current ATLAS TDAQ software is reported elsewhere [4].

II. Methodology

A. End Hosts, NAT Systems, NICs and Networks

Even though server-quality PCs (the PCs used SuperMicro P4DP8 motherboards) were used for these tests, it was important to characterise the performance of the end systems, the NAT boxes, as well as the networks involved. The end hosts and NAT boxes should have sufficient CPU power, memory bus bandwidth and Input/Output (I/O) capability to cope with the expected network traffic, i.e., packets should not be dropped in the end hosts themselves. It was also important to establish the throughput and packet loss of the end-to-end network infrastructures used.

A methodology [5] for evaluating the end host network performance by using UDP packets to measure the latency, throughput, jitter, packet loss and the activity on the PCI/PCI_X buses was used to evaluate these PC systems. udpmon [6] was used to send streams of UDP packets spaced at regular, carefully controlled intervals between the server systems connected back to back. UDP/IP frames were chosen for the tests as they are processed in a similar manner to TCP/IP frames, but are not subject to the flow control and congestion avoidance algorithms defined in the TCP protocol and thus do not distort the base-level performance. The packet lengths given are those of the user payload.[2] The methodology was also applied to each of the networks tested to measure end-to-end performance and characterise packet loss.

B. The Request-Response Application Protocol

The data transfer protocol used in the TDAQ DataFlow application is presented in Fig. 2. The EF sends a small request to the SFI, which immediately responds by transferring an entire event. The EF processes the event data and, if the event is considered interesting from the physics point of view, requests a buffer for temporary storage at the SFO. The SFO grants the request (immediately, if enough space is available) and as soon as the EF receives this grant response, it starts transferring the event to the SFO. The work presented in this paper focused on determining the performance of the transfers over a long-distance network, so all the data was dummy and no physics analysis was performed.

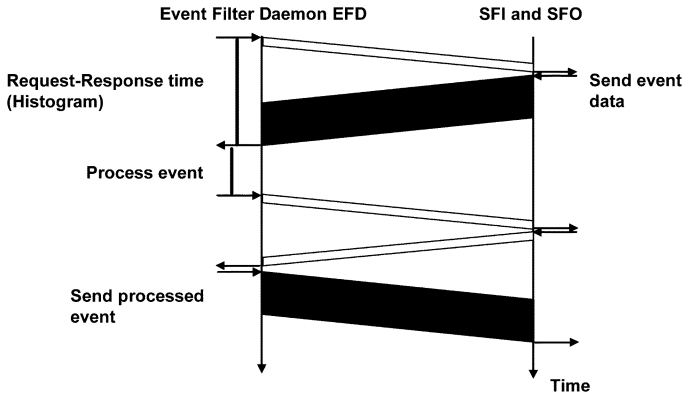

Fig. 2. Request-response protocol used in the ALTAS Event Filter Application.

The operation of the event request-response application protocol used in TDAQ was investigated with the test program tcpmon [7]. Tcpmon implemented a request-response protocol over a TCP connection to transfer data as shown in the top portion of Fig. 2. This program was instrumented in a similar way to udpmon and provided histograms of the round trip request-response times as well as a time series of these latencies.

C. TCP Stacks and Operation

Work on advanced network protocols implementing sender side modifications to TCP, has highly increased the bandwidth utilisation in long delay high bandwidth and multi-user network environments [8]. This allows a single stream of a modified TCP stack to transmit at rates that would otherwise require multiple streams of standard TCP. The measurements reported here were made on systems using RedHat Linux 9 with the 2.4.20 kernel, patched to allow choice of the active TCP stack algorithm [9] with no system reset required after changing the stack. The possible choices included standard TCP, High Speed TCP [13] and Scalable TCP [14]. The High Speed and Scalable stacks both make the reduction of the rate less severe when detecting packet loss, whilst increasing the transmission rate more aggressively than standard TCP during the recovery. Web100 [10] was used to instrument the TCP stacks.

III. Results and Discussion

A. Characterising the Network Testbed

Fig. 3 shows the UDP achieved throughput, packet loss and packet re-ordering for memory-to-memory tests performed between PCs at CERN and Manchester when passing through the NAT box. For packets greater that 1200 bytes the systems were capable of operating at line speed with no packet loss, but small packets suffered considerable losses. For UDP tests between CERN and Manchester made without the NAT box, packets were only lost when the length was 200 bytes or less and spacing of less than 5 μs. This, together with observations

[2]Allowing for 20 bytes of IP and 8 bytes of UDP headers, the maximum user payload for an Ethernet interface with a 1500 byte Maximum Transfer Unit (MTU) would be 1472 bytes.

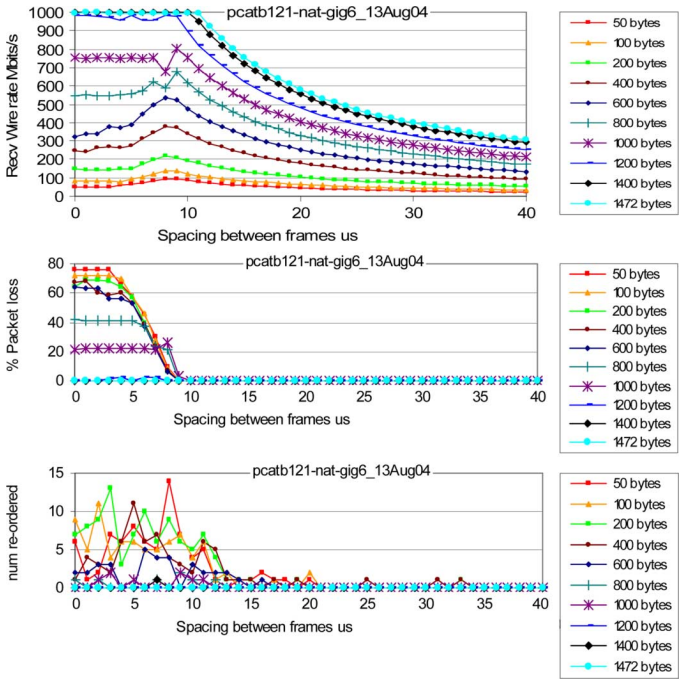

Fig. 3. Results of UDP traffic sent from Manchester to CERN through the NAT box.

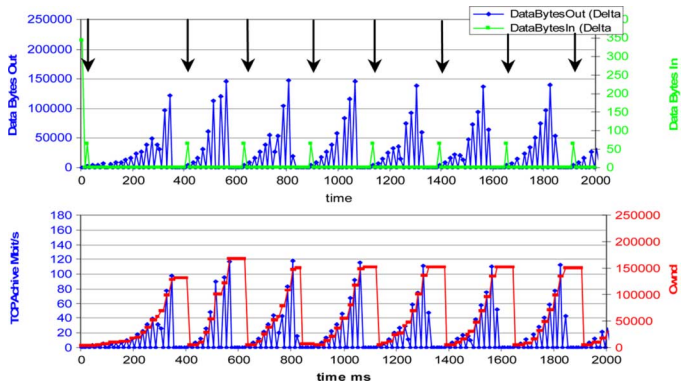

Fig. 4. Request-response traffic showing the effect of cwnd reduction after periods of no data transmission. Upper plot: the request is shown in green with an arrows, and the response in blue points Lower plot: The variation of cwnd in solid red line and the TCP throughput in blue points.

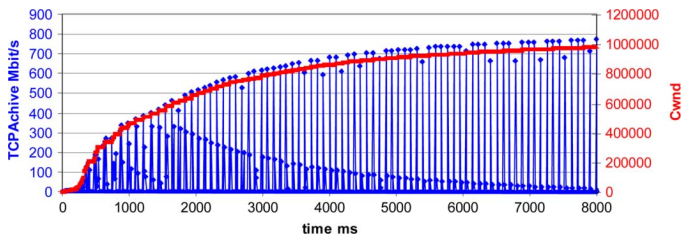

Fig. 5. The bandwidth (points) of the request-response traffic with the standard TCP congestion algorithm but with cwnd reduction turned off. Cwnd is shown as the solid red line.

of the number of packets transmitted and received in the end hosts, suggests that the NAT box was responsible for the losses observed in Fig. 3.

However, this packet loss did not affect the TCP flows used for the request-response application protocol as TCP tries to completely fill the packets when sending data and only uses small packets to acknowledge (ACK) the data received. These ACK packets are only sent for every other received data packet, which would give a spacing of $\sim$24 μs, where there is no observed loss for UDP traffic.

The packet re-ordering shown in the bottom graph in Fig. 3 was due to the multiple forwarding engines in the M160 Juniper core routers of the production network, and is a well known effect [12].

B. The Request-Response Protocol Over the WAN

After characterising the end to end links using UDPmon, tcpmon was used to investigate the performance of the request-response traffic. In these tests, the size of the TCP send and receive buffers were set to delay*bandwidth product for each link used. For all the tests reported in Sections III-B and III-C the MTU was set to 1500 bytes giving a maximum segment size (MSS) for the TCP data of 1448 bytes.

To understand the action of TCP, the internal parameters were sampled every 10 ms using the web100 kernel interface. Fig. 4 shows the behavior of the standard TCP stack for the first 2 seconds of the test when tcpmon was run between Manchester and CERN. This network path had a round trip time (RTT) of 21 ms. The upper plot shows the 64 byte requests in green, highlighted with arrows, followed by the 1 Mbyte response shown as blue points. A new request was sent 80 ms after each response was received. TCP is in its slow start phase where it exponentially increases the Congestion Window (cwnd) each RTT thus allowing more data to be sent each time. In this case it takes 8 round trip times or $\sim$160 ms to send the response. In the lower plot, the red solid line indicates the increasing behavior of the TCP Congestion Window as a function of time and the blue points show the corresponding increase in throughput.

After the 80 ms pause in sending data (which would occur while the remote CPU was analysing the event), the TCP stack dramatically reduces the congestion window; as proposed in RFC 2861 [15] where it is suggested that after inactivity in sending data, cwnd no longer reflects the current information about the state of the network and should be reduced. This means that for our TDAQ distributed application, TCP is always in the slow start phase limiting the request-response latency to 160 ms and the maximum throughput to $\sim$120 Mbit/s. Fig. 5 shows the case with the standard TCP congestion avoidance algorithm (AIMD), but with the cwnd reduction feature turned off. TCP slow start allows the congestion window to open, when it does, the response occurs in about 1–2 RTT giving a request-response latency of $\sim$40 ms and a maximum throughput of $\sim$800 Mbit/s.

Fig. 6 shows the results of a similar test on the Geneva-Edmonton connection, performed using the same TCP stack and the cwnd reduction turned off. Having an RTT of 150 ms, this was the longest connection of our testbed, giving a correspondingly longer TCP slow start period. The first 1 Mbyte event took about 1.8 seconds. Soon afterwards the TCP stack entered the congestion avoidance phase and the throughput grew slowly up to about 800 Mbit/s. The congestion window grows much

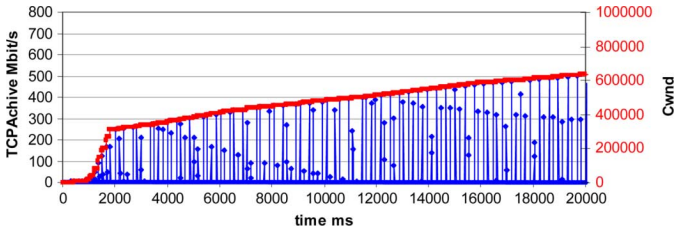

Fig. 6. Request-response transfers on the Geneva-Edmonton connection using the standard TCP congestion algorithm but with cwnd reduction turned off. The achievable bandwidth is shown by the blue points and Cwnd by the solid red line.

slower than that of the Manchester-CERN path because the RTT is much larger.

As the sending and receiving TCP buffers were set to the delay bandwidth product for these tests, when a request is received, and data is sent for transmission over the network, TCP is limited by cwnd, and only transmits data up to the current value of cwnd. With a 1 Gbit/s link this takes less than the 10 ms sample interval. To send more data, TCP has to wait for the ACKs to arrive one RTT later and then sends more data. This gives the "spiky" aspect observed in the plots. Closer examination of the data plotted in Fig. 5 for the Manchester-CERN path reveals that 3 RTT were required to send the 1 Mbyte of data up to ∼1.5 s into the test and then 2 RTT were needed, but the amount of data sent in the second RTT gradually decreased as the cwnd increased. For the CERN-Edmonton path, 3 RTT were required up to ∼11.5 s into the test then 2 RTT up to about 166 s and then the 1 Mbyte could be sent in one RTT.

Fig. 7 presents the time series plots of the individual request-response transaction latencies obtained on the Geneva-Manchester and Geneva-Edmonton connections, for a response of 1 MB in length. The delay is clearly dominated by the physical RTT of the respective connection, as demonstrated in the above discussion. For the Geneva-Manchester plot, the drop from 64 ms to 54 ms at 1.4 s into the test corresponds to the change from needing 3 RTT to send the data to only 2 RTT, and the curved decrease indicates that less and less time is required in the second RTT to send the data. The artefact at 8.1 s corresponds to a congestion event requiring the re-transmission of two packets. The alternate latencies of 29 ms and 42.5 ms from, 18.5 s onwards, correspond to the need to send a small amount of data during the second RTT for alternate transfers.

The step changes in the latency observed on the Geneva-Edmonton circuit at 10.8 and 166 s correspond to the need for 3, 2, and finally 1 RTT being required to send the response data, as discussed above.

C. Using the ATLAS DataFlow Application.

As one of the tests, the Atlas Online and DataFlow software [11] was configured at CERN to operate a three node EF farm in Manchester, while the SFI and SFO applications ran on the same node installed at CERN. Two of the EF nodes were connected through 100 Mbit/s Ethernet links, while the third node had a Gigabit Ethernet link to the same switch that was in turn connected to the wide area network. The ATLAS Online software allows for pre-defined nodes to be removed or added from/to

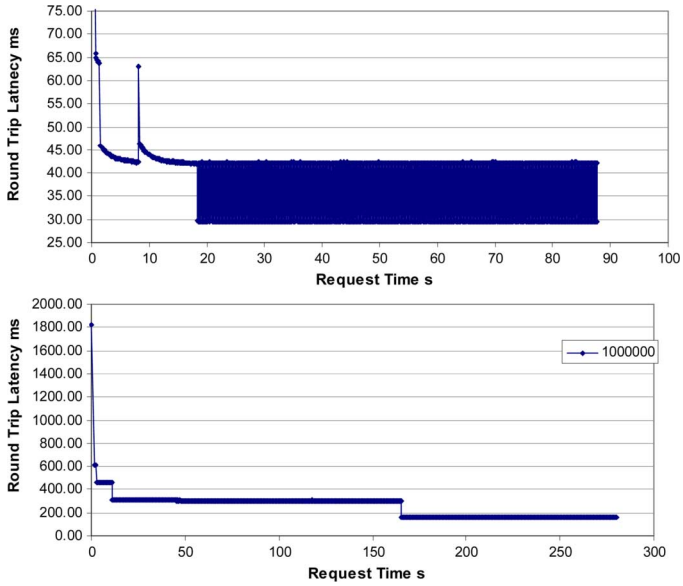

Fig. 7. The delay of the request-response transaction on the Geneva-Manchester (top plot) and Geneva-Edmonton (bottom plot) connections.

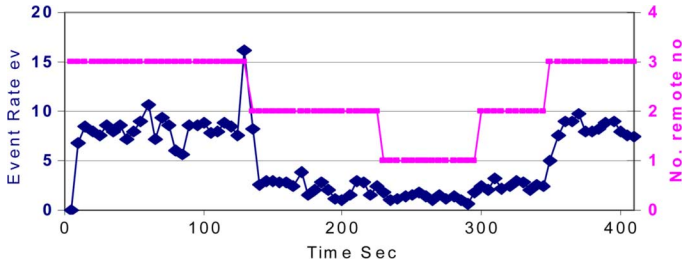

Fig. 8. Average event rate received by the SFO as a function of the time though the test.

the system configuration at runtime. The solid magenta line in Fig. 8 presents the number of active EFD nodes during a certain time interval and the diamond points show the average total rate of received events, as reported by the SFO.

At about 120 s into the test, the node with the Gigabit Ethernet connection was removed and at about 220 s another node was removed. At 300 s and 350 s the nodes were returned to the farm. The results show a rate of about 6 Hz, for the node connected through Gigabit Ethernet and 1–2 events/s for the 100 Mbit nodes. At this time the socket buffers of the application could not be set to the value of the delay∗bandwidth product as used in the other tests described above. The RTT was 20 ms and for the transferred event size of 1 MB the transfer time determined by tcpmon was 42.5 ms. Taking into account the SFI-EF-SFO communication, the time for returning an event to the SFO would be around 95 ms. giving an expected event rate of 10.5 Hz.

Tcpdump, a software tool that allows the parameters of individual packets to be recorded, was used to investigate the behavior of the application. Tcpdump was run on the SFI node at CERN. Fig. 9 and Fig. 10 show the data transfer from the SFI to the EF. Fig. 9 shows the classic TCP slow start period of the communication between the SFI and the EF, which took roughly 320 ms, and Fig. 10 shows the detailed packet dynamics as a function of time for the transfer of a 1 Mbyte response, taken

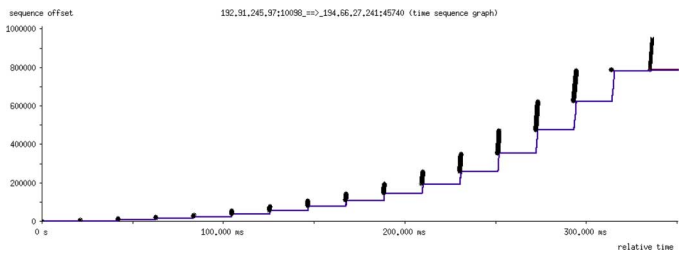

Fig. 9. Detailed view of the slow start phase of the SFI-EFD communication.

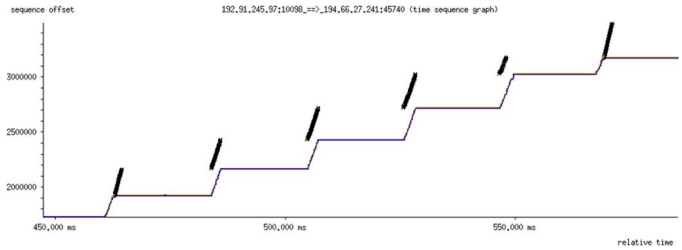

Fig. 10. Detailed view of the SFI-EF communication.

well after the slow start phase had ended. The almost vertical black lines are composed of small bars, each bar corresponding to the sending of a TCP packet of 1448 bytes. The green solid line represents the amount of transmitted data that has been acknowledged by the remote node, with the steps corresponding to the arrival of the acknowledgment packets received by TCP.

Fig. 10 shows that data are sent out in a small burst of packets sufficient to fill but not overflow the remote TCP buffer. As soon as the acknowledgments arrive, one RTT later, more data (represented by the black lines) are sent on the TCP connection. This plot demonstrates that the application could not set the buffer size of the receiving socket to the required value, thus TCP was unable to obtain the maximum transfer rate on the connection. It took about 115 ms to transmit one event, which meant a rate of $\sim$4 events per second for the overall system, much lower than the expected 10 events per second and closer to the measured 6 events per second.

The SFO application establishes an application level timeout for the interval when a complete event should arrive. This was fixed at a value characteristic for a local computing cluster environment. The counting starts after sending a positive answer to the EF's request for temporary storage space. Due to the TCP buffer limitations discussed above, sometimes the event cannot arrive within the configured timeout value. The application considered this to be an error and dropped the connection voluntarily, only to try re-enabling it immediately. Fig. 11 shows this behavior, when the dark lines terminate near the top of the plot. Unfortunately, the newly established connection will have to pass through the TCP three way handshake and the slow start phases before being able to transmit data at the highest rate, causing further delay. As TCP has its own timeout mechanisms and can inform the application if a link is terminated, it is clear that the use of application timeouts and the response to error conditions must be approached with some care.

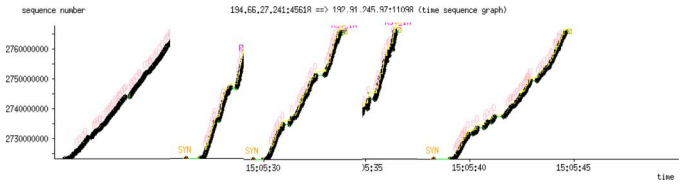

Fig. 11. tcpdump trace of the EFD-SFO connection.

IV. Conclusion

We investigated the behavior of an application designed for a computing cluster when used in a long-distance network scenario. The application consisted of a request-response protocol, running over a TCP/IP connection and designed for high-energy physics data analysis in the Trigger and Data Acquisition System of the future ATLAS experiment at CERN.

The dynamics of the TCP protocol strongly influence the performance achieved by the application. Due to the request-response nature of the protocol, it is not the TCP throughput but the round-trip time that determines the performance of the application. The transmission time of the first block of data sent after the opening of the connection is considerably increased by the TCP slow start. It is also essential to set the TCP buffer sizes to allow rapid transmission of the data. Linux TCP implementation-specific optimisations, like the automatic reduction of the current window size after a certain time of inactivity on the connection considerably reduced the performance of our application.

The request-response nature of the protocol under investigation makes this application different from standard high-energy physics applications deployed over long-distance networks, which usually involve the bulk transfer of large files. However, in addition to the high-energy physics field, our findings are relevant to applications that use iSCSI transfers over Internet remote database accesses or to interactive medical imaging applications that need to transfer large patient images in real time. These investigations are also relevant to simulations that use High Performance Computing facilities with a remote researcher requiring real-time visualisation to allow interactive computational steering using haptic input devices.

We believe that the advanced TCP stacks tested are effective for rapid recovery from packet loss and are stable. They could be built into the ATLAS TDAQ installation environment. We note during the manuscript process that the 2.6 kernels now have these features.

Acknowledgment

The authors would like to thank members of the ATLAS Online and DataFLow groups for their help in making this work possible. They would also like to thank E. Martelli and P. Moroni from CERN, as well as members of Canarie, DARENET, Netera, PSNC, UKERNA, and Dante for their help in configuring the National Research Networks and GEANT.

References

[1] R. Hughes-Jones, B. Caron, K. Korcyl, C. Meirosu, and A. Waananen, "A proof of concept demonstration of remote farms for real-time processing for ATLAS trigger/DAQ," *ATL TDAQ*, Jun. 2004.

[2] ATLAS High-Level Trigger, Data Acquisition and Controls, Technical Design Report CERN/LHCC/2003-022, Jul. 2003.
[3] U. Epting, Computing and Network Infrastructure for Controls (CNIC) Policy [Online]. Available: http://wg-cnic.web.cern.ch/wg-cnic
[4] C. Bee, "On the potential use of remote computing farms in the ATLAS TDAQ system," presented at the 14th IEEE Real Time Conf., Stockholm, Sweden, Jun. 2005.
[5] R. Hughes-Jones, P. Clarke, and S. Dallison, "Performance of 1 and 10 gigabit Ethernet cards with server quality motherboards," *Future Generation Comput. Syst.*, vol. 21, no. 4, pp. 469–488, 2005.
[6] UDPmon: a Tool for Investigating Network Performance. [Online]. Available: http://www.hep.man.ac.uk/~rich/net
[7] TCPmon: Test program Instrumenting the Request-Response Protocol. [Online]. Available: http://www.hep.man.ac.uk/~rich/net/tcpmon
[8] H. Bullot, R. L. Cottrell, and R. Hughes-Jones, "Evaluation of advanced TCP stacks on fast long-distance production networks," *J. Grid Computing*, vol. 1, no. 4, pp. 345–359, 2003.
[9] The 2.4.20 kernel was from http://www.kernel.org/patched with Yee Ting Li's patch 20040119_linux-2.4.20-altAIMD-0.3-web100-2.3.3_sacks.patch [Online]. Available: http://www.hep.man.ac.uk/~rich/SC2004/patches
[10] Web100 Project home page. [Online]. Available: http://www.web100.org/
[11] A. dos Anjos, S. Armstrong, J. T. M. Baines, H. P. Beck, C. P. Bee, and M. Biglietti *et al.*, "Deployment of the ATLAS high level trigger," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 4, pp. 2144–2149, Aug. 2006.
[12] Final Report On Network Infrastructure And Services, DataGrid WP7, Deliverable D7.4, EU DataGrid Document DataGrid-07-D7-4-0206-2.0.doc, Jan. 26, 2004.
[13] S. Floyd, "HighSpeed TCP for large congestion windows," RFC 3649, Dec. 2003.
[14] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, Apr. 2003.
[15] M. Handley, J. Padhye, and S. Floyd, "TCP congestion window validation," RFC 2861, Jun. 2000.