

The use of TCP/IP for Real-time Messages in ATLAS Trigger/DAQ

ATLAS Data Collection Note 62

R. E. Hughes-Jones¹

1. Introduction

This paper discusses the behaviour of the TCP/IP protocol when used for exchanging data and control messages in the ATLAS Trigger/DAQ environment. The ATLAS Trigger/DAQ is considered as a real-time application; and details of the application level protocol, such as the packet contents, are not considered here. Although the exact details of the network communications within ATLAS Trigger/DAQ are still being discussed, the general functionality expected from the network is sufficient for this analysis. Some indications of where this protocol would be useful and where it may be inappropriate are presented.

Appendix A gives a brief or “micro” introduction to the parts of the TCP/IP protocol that are relevant to Atlas Trigger/DAQ.

¹ The University of Manchester

2. Messages in the Trigger/DAQ Candidate Architecture

The most important messages exchanged in the Network based Architecture discussed in [1] are shown in Figure 1, and are fully described in [3]. At the application level, the messages exchanged make up a Request-Response protocol, e.g. when an SFI sends a message requesting data from a RoB in the ROS, the message returned is either the data requested, or just an empty fragment consisting only header information. In the case of a transmission failure, there will be no response.

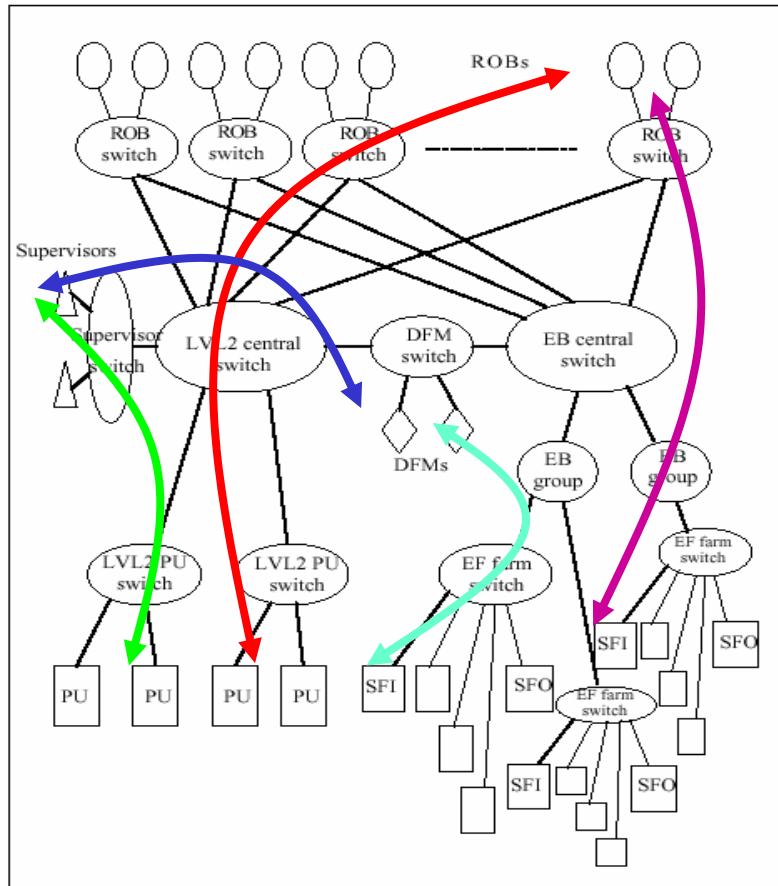


Figure 1 Message exchange in the Candidate Trigger/DAQ Architecture

The messages shown in Figure 1 are described in the following paragraphs. The rates and bandwidths being consistent, have been taken from table 8-1 and A-10 of the TDR.

- **Supervisor to Level2 processing Unit.** The Supervisor (L2SV) provides each L2PU with LVL1 and RoI information about the event that it should process next and after processing the L2PU returns the accept/reject result of the trigger calculation. These exchanges have a low rate of ~500 Hz per processor.
- **L2PU to ROB.** These messages request specific RoI sections of the event data which is returned from the selected RoBs in the ROS and need to be of low latency.

- **L2SV to DFM.** The accept/reject decision from the Level 2 processors are grouped by the Supervisor and passed to the Data Flow Manager. The rate is low and expected to be ~ 2kHz between individual Supervisors and DMF processors.
- **DFM to the ROS.** Transmission of the grouped “event clear” messages could use multicast provided that the ROS could allow for lost packets.
- **DFM to SFI.** These messages tell the individual SFI which of the events accepted by level2 they are to collect and pass to the Event Farm. The rate between a given DMF and an individual SFI is expected to be low ~ 20Hz.
- **SFI to RoBs.** Although data must be collected from each RoB for every accepted event, a given SFI will only request data from each RoB at ~20Hz (the event rate of the SFI).
- **SFI to Event Farm Processors.** The SFI will send the built event, typically 2-3 Mbytes to the EF processors at a rate of < 20 Hz.

Currently there are several options for the configuration of the ROS:

The Switch based ROS

- a. Data requests go direct to the RoB and the requested data is sent directly back to the requestor (L2PU or SFI) over the network. The ROS provides monitoring and deals with operational matters.
- b. The request goes to the ROS or RoB controller and it is then fanned out to the RoBs over a network. The reply to the Level 2 processor goes back directly over the network.
- c. The request goes to the ROS or RoB controller and it is then fanned out to RoBs over a network. The reply to the processor goes first to the RoB controller and then on to the L2PU. In terms of the request-response on the network this is the same as the bus based ROS.

The Bus based RoS

The request goes to the ROS or RoB controller and it is then fanned out to RoBs over a local bus. The reply to the processor goes to the RoB controller and then on to the processor over the network.

Taken from the TDR (Just to help you to get switch vs. bus-based in-line):

- The bus-based ROS — the ROS PC contains three ROBins, each connected to its own PCIbus and each having four ROL inputs to four ROBs, and one PCI output. This output is connected to the ROS’s PCI-buses, which implement the RRM function. Requests, coming from LVL2 or the EB, for event fragments in the ROB, are handled directly by the ROS PC which aggregates the event fragments over its PCI-buses before dispatching them to the LVL2 or the EB. Gigabit Ethernet interfaces connect the ROS to the LVL2 and EB networks.
- The switch-based ROS — the ROS PC houses typically 10 ROBins, each having four ROL inputs to four ROBs, and one Gigabit Ethernet output. The RRM function is implemented typically using a 10 x 4 Gigabit Ethernet switch, which concentrates the ROBin outputs directly into four Gigabit Ethernet outputs: two for the LVL2 network and two for the EB network¹. The switch-based ROS is understood to include the switch as well as the PC. The ROS PC itself, does not play any role in the process of transferring data between the ROBs, and the LVL2 and EB. It is solely responsible via its PCI-bus for the physical housing of the ROBins, their initialisation and control, and some monitoring functions.

3. TCP/IP and Trigger/DAQ

3.1. Properties of the Transfers

The general properties of the ATLAS trigger/DAQ network transfers discussed above may be summarised as follows:

- Many logical links are involved between the components shown in Figure 1.
- These links will remain alive for a long time – many days! So they are not like Web transfers nor transactions nor file transfers.
- The application protocol is mainly Request-Response – typically the request will fit into 1 packet and generally the response will be 1-2 packets long. Or in the case of the Bus based ROS, the reply would have about 8 packets – corresponding to the number of RoBs controlled by the ROS.
- There are no continuous high rate flows i.e. there is no bulk streaming.

3.2. Implications

The time taken by the 3-way handshake used when a TCP link is created is unimportant as the links last for many days. This is an issue for Web access and transaction processing.

The behaviour of the TCP slow-start algorithm is also not very important as

- a. The RoB/ROS – L2PU and event building messages are on a LAN and the round trip time, rtt , is small $\sim 50 - 100 \mu s$.
- b. The data fragments are small $\sim 1-2$ packets. Note there would be an issue for the Bus based ROS, or a ROS design where the RoB data are collected and then sent over the network. If there had been a previous timeout (due to a lost packet for example) Slow start would be invoked which means that more ACKs have to be sent and the transfer takes correspondingly longer.

Limitation of the transmission bandwidth due to TCP entering the congestion avoidance phase also has limited impact as the data fragments are small. Normally when TCP detects congestion, it halves the TCP congestion window, $cwnd$. However, the data responses are only 1-2 packets long, so reducing $cwnd$ is not an issue.

There is much current research into new transport protocols that to improve the behaviour of standard TCP, for example HighSpeed TCP [6] and Scalable TCP [7]. This work is aimed at improving throughput recovery after TCP has detected congestion, and is more appropriate to high bandwidth long rtt WAN links rather than hosts on a LAN.

On the positive side, TCP will take care of any packets lost in the network.

However each TCP connection requires some resources to be allocated, even though “pooled” buffering may be used for the data. This will require careful consideration when considering scalability of the system.

As discussed below, TCP itself will send acknowledgements for both the request and the response packets. As the Trigger/DAQ application. Is based on Request-response, the exchange of the extra ACK information is not needed from the application point of view.

4. Trigger/DAQ Use Cases and the TCP/IP Protocol

The following sections attempt to illustrate the packet flow when TCP/IP is used as the network transport. Each section gives a diagram to show the way the Trigger/DAQ applications would interact with the protocol stack and the physical network, with the time line running down the page. There is then a discussion of what would happen if packets were lost or corrupted.

TCP will send an ACK protocol packet to acknowledge the data it has received; this control information may be included in a data packet if one is being returned to the sender, this is often called piggyback ACKs. If there is no data to send to the sending host, a separate packet will be created.

Another important aspect is the average time between sending subsequent requests to a remote host. The sending TCP stack detects a lost packet by either receiving duplicate ACKs for subsequent packets send or by a timeout expiring if no ACK is received. Tests on a LAN, See Section **, observed that ACKs could occur up to ~35 ms after a packet was sent, indicating that the TCP timeout was greater than this. As only a small amount of data is being sent and if the average time between sending requests is large, TCP will tend to rely on timeouts to detect packet loss.

4.1. Event Building: SFI requesting data directly from a RoB

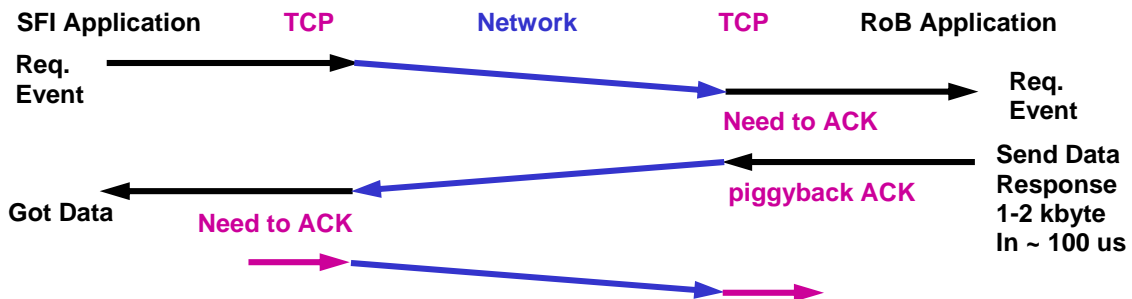


Figure 2 The packets involved when a SFI requests data directly from a RoB

When the RoB node receives a request for the event data from the SFI, it can locate and return the data in ~100µs, so the TCP stack in the RoB node will piggyback the ACK on the data. However, each SFI processes ~40events/s, so on average it will only send a request to the same RoB every ~66 ms. This means that the TCP stack in the SFI will create and send an ACK as shown in Figure 2. Thus compared to using UDP/IP or Raw frames, the load on the network has increased: the number of packets out of the SFI and the number into the RoBs has doubled and the total number of packets on the network has increased by 3/2. This will involve extra CPU power on the SFI and RoB to process the ACKs. The extra packets will lead to additional buffer occupancies inside the layered

switching network, making congestion and longer latencies in the switches more likely to occur.

4.1.1. Assume a SFI request is lost

TCP wont timeout and re-try sending the request for ~ 35 ms – a long time compared with the rtt.

4.1.2. Assume a RoB Response is lost

The sending SFI wont get the ACK so the SFI will timeout and re-send the request. But the RoB wont get its ACK for the data it sent to the SFI, so the RoB TCP will think about timing out and re-try. **Both** ends re-try, sending more messages into the network which is inefficient to say the least and a potential cause for network trouble i.e. additional packet loss invoked inside the switches!!

4.1.3. Assume an ACK from the SFI is lost

TCP in the RoB will timeout and resend its data with the piggybacked ACK. **Thus the RoB resends data that is not wanted** – the SFI has used it already! In general this timeout will occur before the next request from the SFI (average separation ~66ms).

4.2. Event Building: SFI requests to ROS, with data returned directly from a RoB.

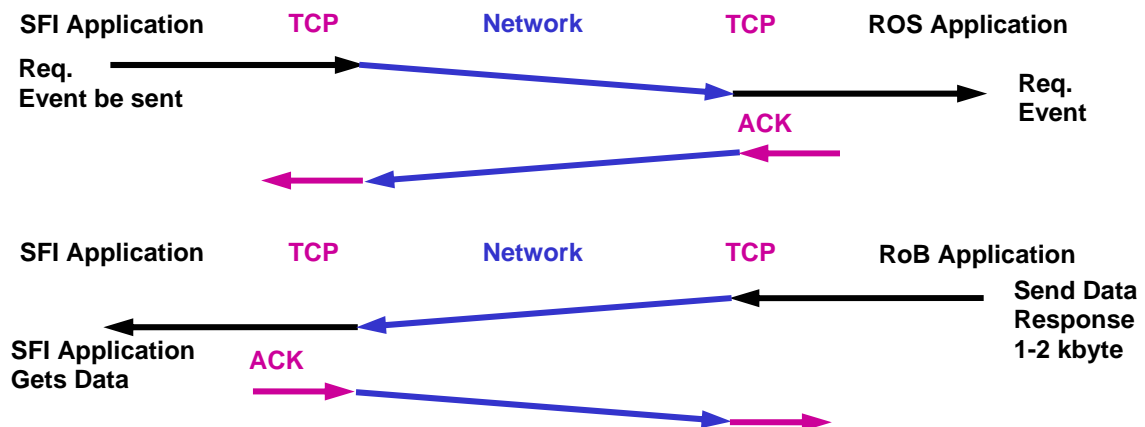


Figure 3 The packets involved when a SFI sends requests to the ROS and data is returned directly from a RoB

Figure 3 shows that in this case there are two separate TCP links: one between the SFI and the ROS and another between the RoB and the SFI to return the data. Both of these links require their own ACKs and ACK processing. This doubles the number of frames on the network for event building, and the ACKs will require processing power on the SFI, the ROS and the RoB.

Further I/O and processing is required for the communication between the ROS and the RoB. This is not shown in Figure 3.

As above, each SFI processes ~40 events/s, so on average it will only send a request to the same ROS every ~66 ms, which is in excess of the TCP timeout.

4.2.1. Assume a SFI request or a RoB response is lost

The relevant TCP wont timeout and re-try sending the request for ~ 35 ms – a long time compared with the rtt and the processing times.

4.2.2. Assume an ACK is lost

TCP in the SFI or the RoB will timeout and resend its data with the piggybacked ACK. **Thus data is resent that is no longer wanted** – the SFI (or the RoB) has used it already! In general this timeout will occur before the next request (average separation ~66ms).

4.3. Level2: L2PU requesting data directly from a RoB

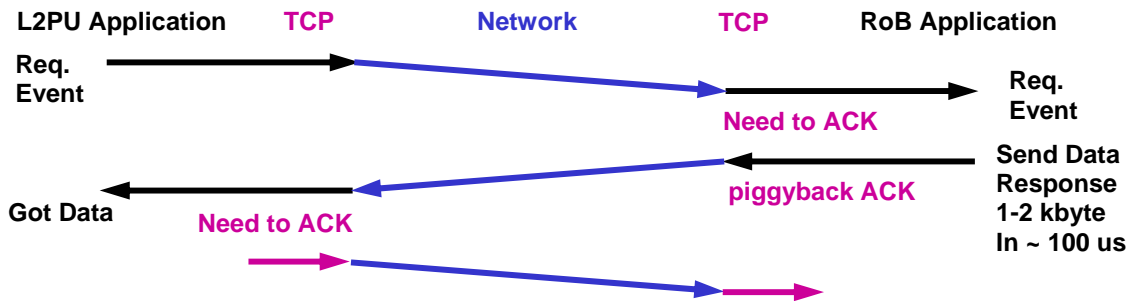


Figure 4 The packets involved when a Level2 Processor requests data directly from a RoB

When the RoB node receives a request for the event data from the L2PU, it can locate and return the data in ~100µs (the same as for event building), so the stack in the RoB node will piggyback the ACK on the data.

Individual L2PU – RoB request rates vary depending on the trigger menu, but taking the High Luminosity trigger as an example, a Calorimeter RoB will have a request rate of ~6kHz from one L2PU – requests 1 every ~ 62 ms. On average the L2PU will accesses 20-40 RoBs per event This means that the TCP stack in the L2PU will create and send a separate ACK packet as shown in Figure 4. Thus compared to using UDP/IP or Raw frames, the load on the network has increased: with the number of packets out of the L2PU and the number into the RoB having doubled. As with the SFI-RoB case, the total number of packets on the network has increased by 3/2. This will involve extra CPU power on the L2PU and RoB to process the ACKs

4.3.1. Assume a L2PU request is lost

TCP wont timeout and re-try sending the request for ~ 35 ms – a long time compared with the rtt and the ~10ms average processing time of the L2PU.

TCP also blocks all other communications between that L2PU and that ROB until the lost packet has been received. Thus if other events in that L2PU require data from the same RoB, these worker threads may stall as well. This does not seem to be optimum.

4.3.2. Assume a RoB Response is lost

The sending L2PU won't get the ACK so L2PU will timeout and re-send the request – which is what may be required. But the RoB won't get its ACK for the data it has just sent to the L2PU, so the RoB TCP will think about timing out and re-try. **Both** ends re-try which is inefficient !!

4.3.3. Assume an ACK from the L2PU is lost

TCP in the RoB will timeout and resend its data with the piggybacked ACK. **Thus the RoB resends data that is not wanted** – the L2PU has used it already and may even be several events further on! In general this timeout will occur before the next request from the SFI (average separation ~66ms).

5. Investigations of TCP on a LAN

To verify the behaviour of the TCP protocol, tcpdump and tcptrace were used to record and display the packets sent between two PCs connected back-to-back. The 2.4.19 Linux kernel was used with the standard TCP stack. A packet was sent to request a “data” response after a given delay; the requesting node then waited before repeating the request. Both the delay before giving the response and the time to wait between requests could be varied. Figure 5 shows how TCP transported the request and the response. As expected from the discussion above, with the delay to issue the response being only $66 \mu\text{s}$, the ACK of this request was piggybacked on the response. As the next request was made 66 ms later, the TCP stack in the requesting node generated an extra packet about 35 ms after receiving the response to acknowledge the 1400 byte response.

These tests confirmed that TCP behaved as expected from the analysis presented in this paper.

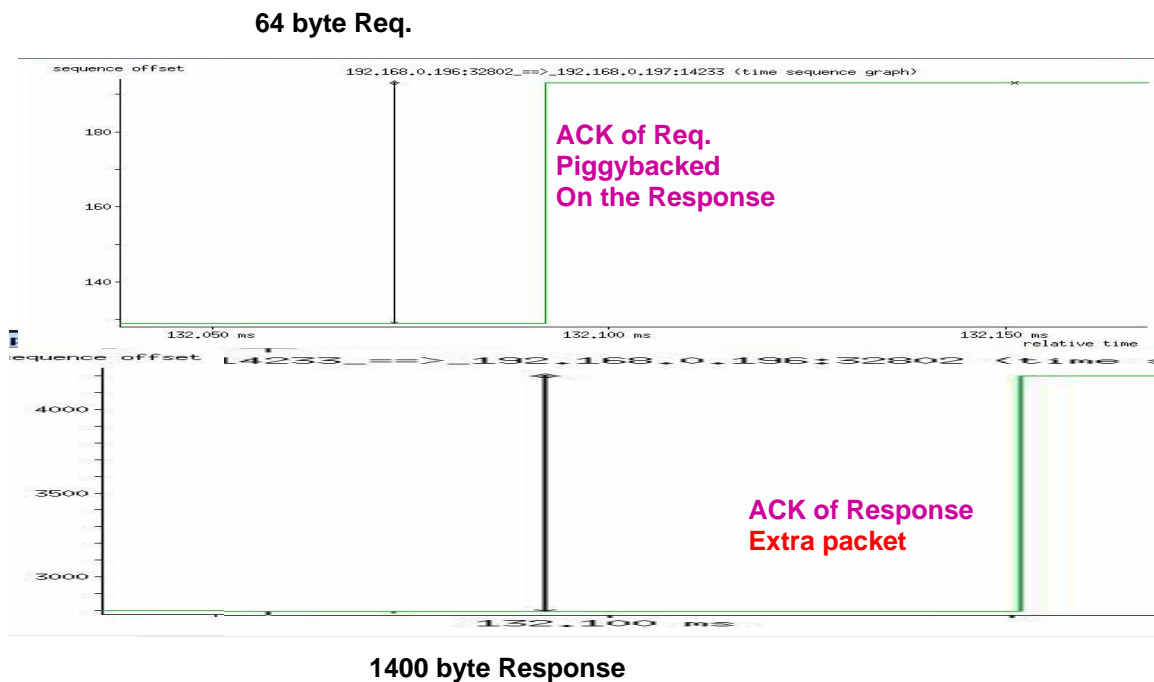


Figure 5 Time series plot of the information exchanged by the TCP protocol for the Request and its Response.

6. Summary and Conclusions

TCP/IP does packet loss recovery which is useful but TCP/IP (or the implementation) sometimes does things behind your back which may not be helpful in the real-time environment. An Example is dropping packets when internal queues become full and treating this like congestion on the network.

Under some conditions described above which in our real-time environment are not pathological, TCP will re-send data when it is no longer wanted.

TCP/IP uses timeouts to detect lost data just like the Trigger/DAQ applications do now for UDP/IP or Raw frames but the timing is much cruder. In effect, the Trigger/DAQ high performance applications loose fine control of the network transfers and through this the timing of the thread operations.

But most important, the TCP/IP ACKs will generate extra traffic on already loaded links.

For these reasons UDP/IP or Raw frames would be preferable for the high rate transfers:

- L2PU to ROB.
- SFI to RoBs.

TCP/IP is useful for the low rate “control” message exchanges which would include:

- between L2SV and L2PU.
- between L2SV and DFM.
- Between DFM and SFI.

There seems little point in increasing the complexity of these application message exchanges by using UDP/IP and taking care of any packet loss at the application level.

TCP/IP also useful to stream the 2-3MByte events from SFI to EF for local and remote farms. The anticipated rate is of these messages is 40 Hz.

TCP/IP is clearly useful for sending data across the academic network infrastructure for processing at remote laboratories. An example of this would be the possibility of using a remote Event Filter farm.

7. Acknowledgements

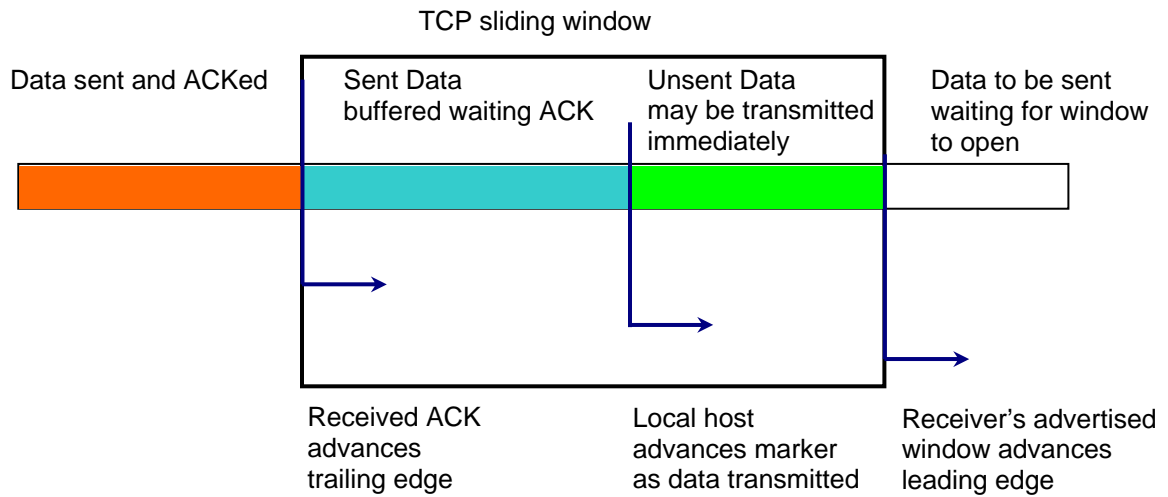
The author would like to thank the ATLAS Trigger/DAQ community for information and help and in particular Bob Dobinson, Hanspeter Beck and Dave Francis for most useful discussions, comments and encouragement.

8. References

- [1] “ATLAS-TDAQ: A Network based Architecture”, DataCollection note 59,
- [2] “The Trigger DAQ TDR” http://atlas-proj-hltdaqdcs-tdr.web.cern.ch/atlas-proj-hltdaqdcs-tdr/draft_fin_r6/PDF/TDR.pdf
- [3] High Level Description of , ATLAS DF-012, June 2002.
- [4] W.R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.
- [5] G.R. Wright and W.R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*, Addison-Wesley, 1995.
- [6] The HighSpeed TCP stack and some results are available from <http://www.hep.man.ac.uk/u/garethf/hstcp/>
- [7] A discussion of the Scalable TCP stack and the patch are available from <http://www-lce.eng.cam.ac.uk/%7Ectk21/scalable/>
- [8]

9. Appendix A :A micro Introduction to TCP/IP

TCP was designed for reliable data transfer over slow, unreliable Wide Area Networks.



- c.) the congestion window, *cwnd*, will be set to 1 only allowing 1 data packet to be sent until an ACK is received. As ACKs are received, the amount of data