# Quality of Service on Linux for the Atlas TDAQ Event Building Network

Y.Yasu[1], A. Manabe[1], Y. Nagasaka[2], Y. Hasegawa[3], M. Shimojima[4],
M. Nomachi[5], H.Fujii[1] and Y.Watase[1] on behalf of the Atlas Trigger/DAQ group
[1](High Energy Accelerator Research Organization, Japan)
[2](Hiroshima Institute of Technology, Japan)
[3](Shinshu University, Japan)
[4](Nagasaki Institute of Applied Science, Japan)
[5](Osaka University, Japan)

**Abstract**

Congestion control for packets sent on a network is important for DAQ systems that contain an event builder using switching network technologies. Quality of Service (QoS) is a technique for congestion control. Recent Linux releases provide QoS in the kernel to manage network traffic. We have analyzed the packet-loss and packet distribution for the event builder prototype of the Atlas TDAQ system. We used PC/Linux with Gigabit Ethernet network as the testbed. The result showed that QoS using CBQ and TBF eliminated packet loss on UDP/IP transfer while the UDP/IP transfer in best effort made lots of packet loss. The result also showed that the QoS overhead was small. We concluded that QoS on Linux performed efficiently in TCP/IP and UDP/IP and will have an important role of the Atlas TDAQ system.

Keywords: QoS, Linux, TCP, UDP, IP Multicast, CBQ, TBF, DAQ

## 1 Introduction

Traffic management of the data flow is an essential point to avoid congestion. QoS is one of the solutions to avoid the congestion, because it provides the bandwidth allocation in the network. Gigabit Ethernet technology, which is a major in the high performance network, however, has no functionality of the QoS while ATM has the functionality on a hardware level. Recently, link-sharing and resource management for packet network is intensively studied in computer science [3]. On Gigabit Ethernet with QoS, it will be expected to avoid the congestion of the event builder traffic.

From the view point of scalability for the event builder, IP multicast is attractive. However, TCP does not support multicast while UDP is a non-reliable protocol. There are many studies of IP multicast. A solution is to make protocols reliable in the application level. Another solution is to make it reliable in lower levels such as Data link layer. QoS on Linux is implemented below the IP layer and woven into the network layer.

We studied the DAQ/EF-1 Event Builder using Linux/Gigabit Ethernet [1]. We already had a preliminary result of QoS for the event builder [2].

## 2 Congestion Control and Bandwidth Allocation

The essential point of traffic management is packet scheduling, known as queueing method. Packet queueing disciplines are studied in techniques related to ATM in computer science. They are Class-Based Queueing (CBQ), Token Bucket Filter (TBF) and so on.

### 2.1 Traffic Management in Linux

For Linux, kernel 2.2.x and later support many kinds of QoS. The developer toolkit also supplies some commands for manipulating the parameters in the kernel. The typical command is called

`tc`. CBQ queueing discipline in Linux kernel supports hierarchical allocation of bandwidth for link sharing, limiting the packet rate, prioritizing and so on. The TBF queueing discipline in Linux kernel can also limit packet rate by using a token bucket algorithm. The traffic control in TCP/IP for Linux is done only for output queueing. For an example of a queueing discipline, one can define CBQ as root with a bandwidth of 1000Mbit/s. When a packet comes into the queueing discipline, the packet will be sent to a classifier called `u32`, which classifies the packet. The packet will come into a class called CBQ, which also assigns 1000Mbit/s as the bandwidth. After the packet passes through the class, a queueing discipline called TBF will limit the transfer speed. The transfer speed will be limited to 100Mbit/s according to the rate of the token even if packets arrive over the rate.

## 3 Performance Evaluation

Table 1: Configuration of PC500 system

| CPU | Chipset | Memory | PCIbus |
|---|---|---|---|
| PentiumIII/500MHz | 440GX | 100MHz/SDRAM/256MB | 32-bit/33MHz |
| Linux kernel version | gcc version | NIC | acenic driver version |
| 2.4.5 | egcs-2.91.66 | AceNIC(1MB) | 0.8 |

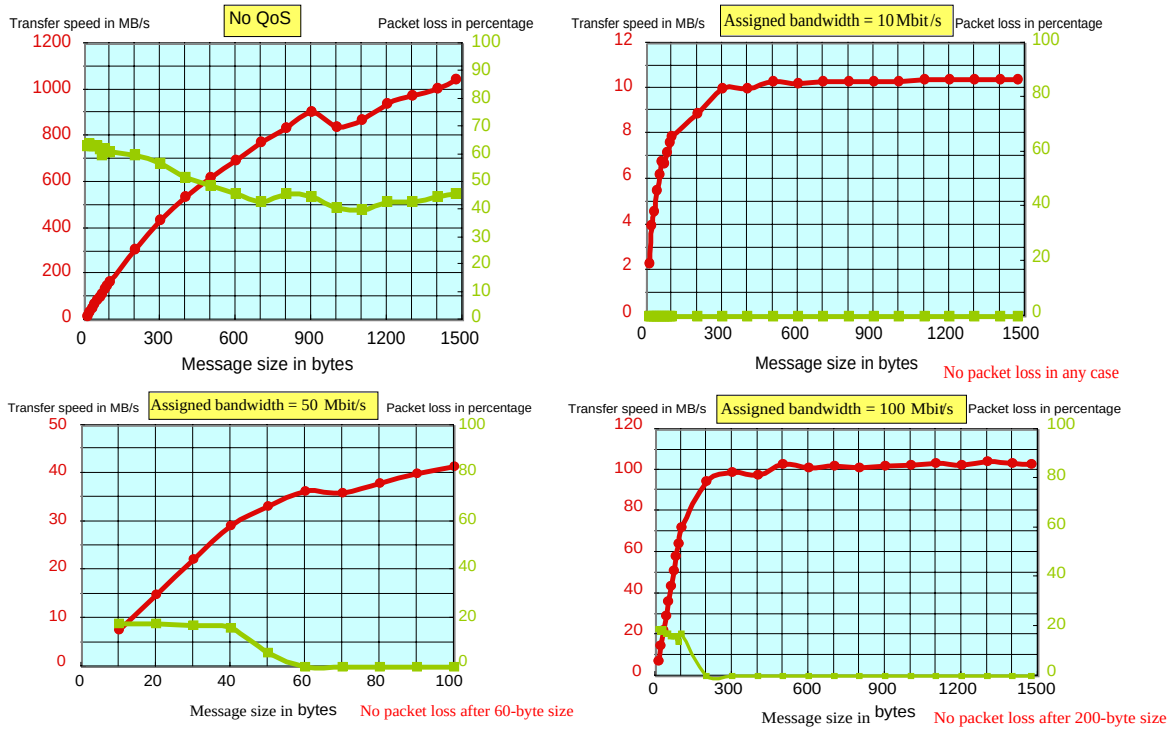


Figure 1: Packet loss on UDP/IP multicast transfer

### 3.1 Measurement

On the measurement of QoS, we measured and evaluated Packet loss for UDP/IP multicast transfer, TCP/IP transfer speed with/without QoS, CPU usage of QoS on TCP/IP transfer and packet distribution on TCP/IP transfer with/without QoS & regular trigger. We used `netperf` and `ttcp` utilities as the measurement tools. The Linux network driver for Alteon AceNIC was used. We enabled the Nagle algorithm for the measurement. The TCP buffer size was 500k bytes.

### 3.2 Setup

The configuration of the PCs is shown in Table 1. Four PCs of PC500 were used with a Gigabit Ethernet switch for the evaluation of QoS. The switch was a 3Com Super StackII Switch 9300, which has 12 Gigabit Ethernet ports.

### 3.3 Packet loss on UDP/IP multicast transfer

Figure 1 shows an analysis of packet loss on UDP/IP multicast transfer. We made a special tool in this measurement. The dark line in the Figure shows the transfer speed and the light line shows the percentage of packet loss. Without QoS, lots of the packet loss were observed. This depends on the message size and the percentage of packet loss was over 60 % when the message size was 10 bytes. On the other hand, no packet loss was observed when an expected bandwidth was assigned with 10 Mbit/s. When the bandwidth was assigned with 50 Mbit/s, packet loss occurred at small message sizes. When the assigned bandwidth was 100 Mbit/s, the message size that packet loss occurred became larger. We should manage the assigned bandwidth when we eliminate packet loss on UDP/IP transfer. It depends on the message size. We also observed that there was no packet loss at the sender while the receiver lost the packet.

### 3.4 Speed and CPU usage on TCP/IP transfer

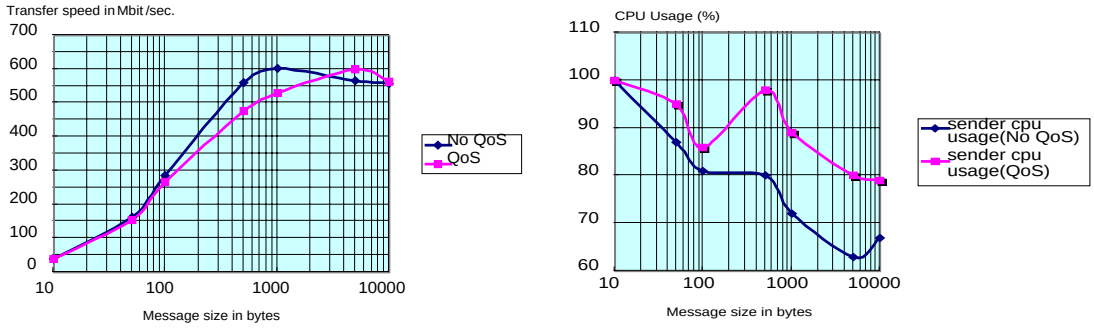


Figure 2: Speed and CPU usage on TCP/IP transfer

Figure 2 shows transfer speed and CPU usage. The assigned bandwidth was 1000 Mbit/s when QoS is applied. This means the transfer is not limited by the assigned bandwidth. The QoS algorithm uses more CPU time, compared with No QoS. The overhead of QoS algorithm or the efficiency of transfer depends on the message size. The transfer with QoS is very efficient at small message and large message. That is, QoS overhead is small. The CPU usage between the transfer with QoS and that without QoS was small at small message, but not at large message. 10 % more CPU was used at large message on the transfer with QoS.

### 3.5 Packet distribution on TCP/IP transfer

A pseudo trigger was provided for generating a regular trigger signal. The assigned bandwidth was 10 Mbit/s. The message size was 10 bytes. The `tcpdump` utility was used for sampling the data. We measured the packet distribution in 4 cases, the transfer with QoS, the transfer without QoS, the transfer with QoS in way to generate regular trigger and the transfer without QoS in the way. Figure 3 shows the packet distributions on TCP/IP transfer in those cases. The x axis is a time interval between sending packets, in seconds. The y axis is the number of packets. In first case, which is on the left side of the Figure, QoS managed to send packets at 10 msec, which is the Linux OS scheduling time. In the second case, packets were mostly sent in several 10 usec. In the third and fourth case, packets were mostly sent around 1 msec, which is the regular trigger rate, 1 kHz. The difference between the third and the fourth was not observed.

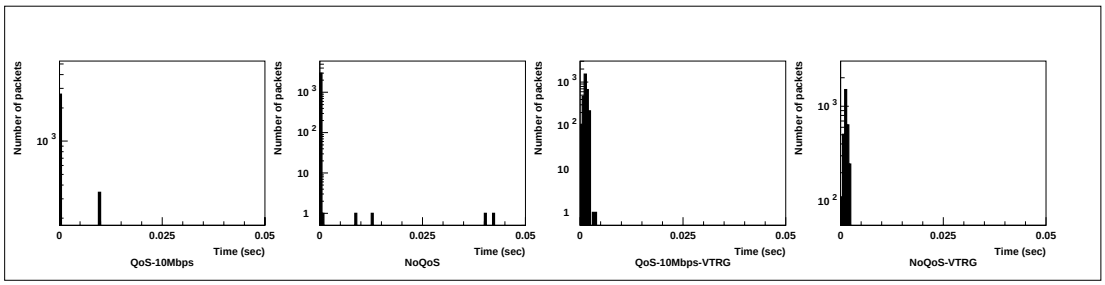


Figure 3: Packet distribution on TCP/IP transfer

## 4 Conclusion

We measured and evaluated Packet loss on UDP/IP multicast transfer, speed and CPU usage on TCP/IP transfer, and packet distribution on TCP/IP transfer with/without QoS & regular trigger. QoS could eliminate packet loss on UDP/IP multicast transfer. This shows the feasibility to solve the scalability of Atlas TDAQ event builder. QoS was efficiently performed on TCP/IP transfer. CPU usage of QoS on the transfer was small in comparison with that of No QoS on the transfer. The QoS makes the packet flow flat in unit of the scheduling time.

## 5 Acknowledgments

The authors wish to thank Prof. Takahiko Kondo at KEK for his support and encouragement. The authors also like to thank Dr. Hanspeter Beck and Dr. Reiner Hauser for checking our paper and their encouragement.

## References

[1] Y.Hasegawa, Y.Nagasaka, Y.Yasu, DAQ/EF-1 Event Builder system on Linux/Gigabit Ethernet, ATL-DAQ2000-008, March 2000

[2] Y.Yasu et al, Quality of Service on Gigabit Ethernet for Event Builder, the 3rd International Data Acquisition Workshop on Networked Data Acquisition Systems, Lyon, France, October 20, 2000

[3] S.Floyd and V.Jacobson, Link-sharing and Resource Management Models for Packet Networks, IEEE/ACM Transactions on Networking, Vol.3 No.4, August 1995