SLAC–PUB–7989
November, 1998

Network Performance Testing for the BaBar Event Builder

T.J. Pavel, D. Millsom, M.E. Huffer, C.T. Day
For the BaBar Computing Group[1]

Abstract. We present an overview of the design of event building in the BaBar Online, based upon TCP/IP and commodity networking technology. BaBar is a high-rate experiment to study CP violation in asymmetric e^+e^- collisions. In order to validate the event-builder design, an extensive program was undertaken to test the TCP performance delivered by various machine types with both ATM OC-3 and Fast Ethernet networks. The buffering characteristics of several candidate switches were examined and found to be generally adequate for our purposes. We highlight the results of this testing and present some of the more significant findings.

Introduction

The BaBar experiment is a fairly typical high energy physics detector that will operate at the PEP-II asymmetric e^+e^- storage ring at SLAC to study *CP* violation in the *B* meson system. The design of the experiment is decribed in detail in [1] and an overview of the data acquisition (DAQ) architecture is presented in [2]. The unusual aspect of the BaBar DAQ is the high sustained rate of events it seeks to support. This requires the use of a partial-reconstruction software trigger (L3 trigger) for reduction of the data set to a manageable rate. Even with the L3 trigger, BaBar will record some 10^9 events per year (resulting in 30 TB/year of raw data alone).

The design parameters for the DAQ system are a modest event size ($\sim$32 kB after feature extraction processing in the front-end CPUs), a trigger rate of 2 kHz delivering events from the front-end CPUs into the L3-trigger farm, and a reduction factor of 20 in the L3 trigger, giving a data storage rate of 100 Hz. The event building is to be done over an IP network as the event is delivered from the front-end to the L3-trigger node. This gives a requirement of an average aggregate load of 64 MB/s on the front-end network. For reasonable multiplicities of machines (*e.g.* above 10), this performance should be readily deliverable on 100 Mbps-class networks.

Event Builder Overview

Some basic decisions in the event builder design were made early on. First, it was decided to base the design on top of the TCP/IP protocol stack, using a commercial data-link technology. Originally, the likely candidate technology was FDDI, but by 1996 it became apparent that the only meaningful options were Fast Ethernet or ATM OC-3. Furthermore, it was decided that trying to perform all of the L3 trigger within a single large SMP node was too risky. Although such a single-node architecture would simplify many things, we wanted the ability to scale our system up by a large factor if machine luminosity or backgrounds or processing time were to increase. SMP machines with more than 30–60 processors become prohibitively expensive, while a farm architecture can scale up as large as the network switching fabric can support. We expect network switches to provide much more cost-effective scaling than SMP machines.

[1)]This work was supported in part by Department of Energy contract DE–AC03–76SF00515.

Presented at the International Conference on Computing in High-Energy Physics (CHEP98), Chicago, IL, August 30, to September 4, 1998

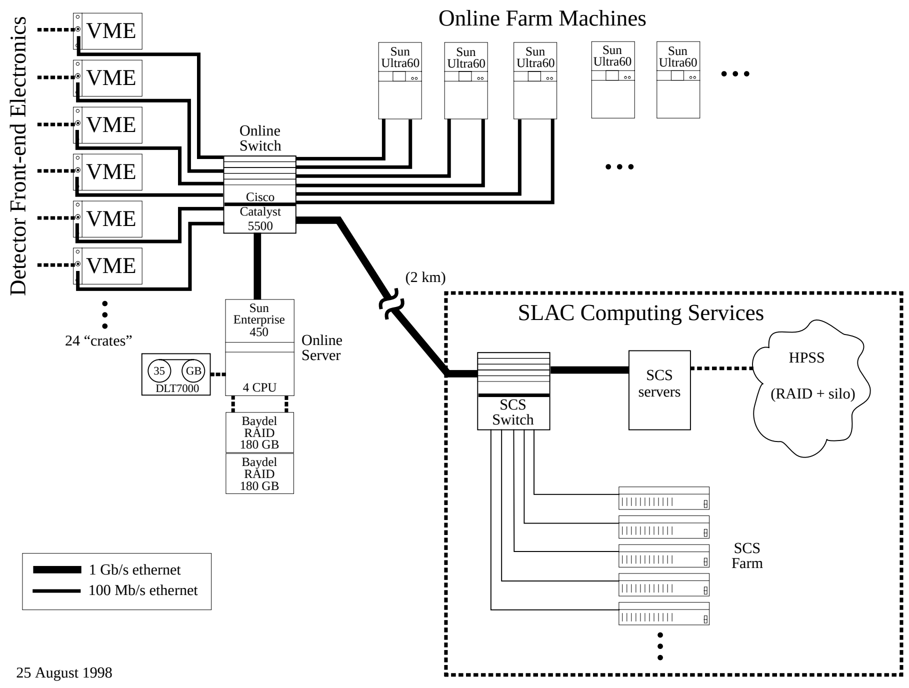

Figure 1: Diagram of the BaBar data acquisition and logging system.

When the construction phase of BaBar began, we felt the need to validate this IP-based event builder design before proceeding with too much of the development work. Therefore, we undertook a program of studying the main performance aspects of our design in a small-scale test lab. This paper presents the main results of that investigation.

At the present time, BaBar development has progressed and many hardware decisions have been made. The current hardware design is shown in Fig. 1, reflecting our design decisions: Fast Ethernet, the Cisco Catalyst 5500, Sun Ultra 60s, and Motorola MVME2306 VME processors. The major design parameters referenced later are the number of senders ($\sim$22) and the number of receivers ($\sim$15). This gives a requirement for the throughput of each link at about 4 MB/s. In addition, we are currently evaluating UDP v. TCP for the underlying transport in the event builder. All of the early studies focussed on TCP because that was easier to model and has better dynamical properties. However, it is felt that a UDP implementation is a better match to the datagram-style service provided by the event builder. So long as one can supply a system of flow-control to prevent overflowing of buffers, it should be the case that UDP throughputs exceed TCP throughputs for all operating systems.

Tools and Methodology

In order to validate the design of the BaBar event builder, we focussed on two main aspects and constructed small-scale experiments to investigate each. The first aspect was to understand the TCP throughput performance achievable between any pair of machines. The other main aspect was to understand the collective effects of the whole system, which would generally manifest themselves in the behavior of the switch. These are described in the following sections.

Table 1: Machines used in the testing

Label	Model	Hardware	Software	Network interface
Alpha	DEC 500/266	AXP 21164/266 MHz	DEC Unix 4.0B	DEC tulip FE ATMworks351 OC3
DEC UW	DEC UW/533	2 * AXP 21164/533 MHz	DEC Unix 4.0D	DEC tulip FE
Ultra2	Ultra2	2 * Ultra/167MHz	Solaris 2.5.1	Sun hme FE Sun ba OC3
Ultra60	Ultra60	2 * Ultra2/360 MHz	Solaris 2.6	Sun hme FE
Solaris/P6	PC	2 * PPro/200 MHz (512kB L2 cache)	Solaris 2.5.1	SMC/DEC FE Interphase OC3
Linux/P6	PC	PPro/200 MHz (256kB L2 cache)	Linux 2.0.35	SMC/DEC FE
IBM 43P	IBM 43P	PowerPC 604/100 MHz	AIX 4.2	3Com 905 FE
IBM F50	IBM F50	2 * PPC 604e/166 MHz	AIX 4.2	3Com 905 FE
VxWorks	Motorola MVME2306	PPC 604e/300 MHz	VxWorks 5.3.1	DEC tulip FE Radstone OC3

For the individual system performance studies, we relied heavily on the **netperf** suite of tools written by Rick Jones at HP.[2] These tools allow for the systematic measurement of many aspects of network performance. We concentrated exclusively on TCP throughput, as that seemed most relevant for event builder performance, and we sought to map out the TCP throughput as a function of the record size passed to the `write()` system call. In all the results that follow, we show the results of 60 second runs of the netperf TCP_STREAM test. Each datapoint was repeated up to 10 times until a 95% confidence interval is reached. Points that did not converge in 10 trials are shown with their correspondingly larger errors.

The set of machines tested and their configurations are shown in Table 1. For the most part, we sought to use evenly matched machines from the different vendors. At the time we undertook this study (Fall 1996), this generally meant 200 MHz/128 MB machines. We tended to use network interface cards (NICs) from the original workstation vendors. For the Intel PCs, we selected cards based on the DEC 21140 "tulip" chip. Of course, our results represent ideal conditions, with no significant additional load upon either the machines or the network switch.

Fast Ethernet Measurements

Because of the wider availability of systems with Fast Ethernet, we performed initial studies with Fast Ethernet and Unix workstations. This demonstrated many interesting effects (discussed below) and taught us which parameters were the most relevant for performance.

Effect of Sender

The most consistent factor in observed TCP throughput, particularly at the lower record sizes, turns out to be the nature of the OS kernel on the sending side. This makes sense, since for record sizes less than the MTU (1500 bytes), efficient use of the medium depends on how well the sending kernel can collect up the individual records into packets. Furthermore, at very small record sizes, the cost of a kernel entry (typically 2–3 μsec on the Unix systems) can dominate performance.

Figure 2 shows a range of different sending systems, all communicating with a Digital Unix system as the receiver. Results are comparable for other receiving machines (see below). One can immediately see

[2]See `http://www.netperf.org/netperf/NetperfPage.html`

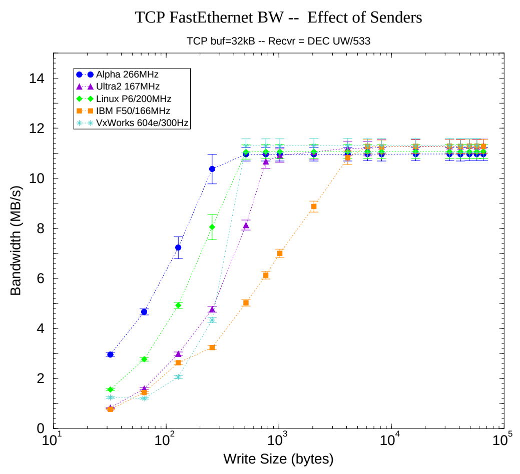

Figure 2: TCP throughput as a function of write-record size for a variety of sending systems going to a Digital Unix receiving system.

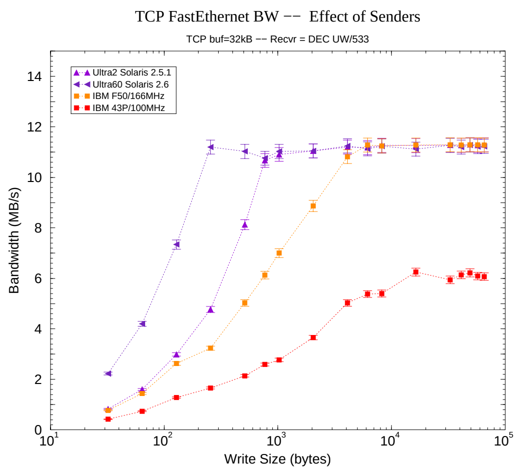

Figure 3: TCP throughput as a function of write-record size for two pairs of similar systems. The triangles represent similar UltraSPARC hardware running two different releases of Solaris, while the squares represent two different hardware platforms running the same release of AIX.

that Digital Unix gives the best sending performance, while AIX has the worst. However, all systems share a similar shape to their throughput curves. For record sizes below about 1 kB, performance is generally much less than the full 100 Mbps line speed. At around 128 B to 1 kB, the throughput rises quickly and reaches a plateau where it is limited by the line speed of the underlying network medium. There are some slight variations in performance achieved in the plateau region, but these differences are rather small.

The differences in sender performance can be due to hardware or to software. Figure 3 shows some additional variation in sender performance. The triangles show the difference between Solaris 2.5.1 and Solaris 2.6, running on similar hardware. The improvement in Solaris 2.6 comes from the promotion of the sockets API into a true system-call interface from an emulation library written on top of TLI. On the other hand, the square points demonstrate the difference between AIX 4.2 running on two different machines. The 43P machine was the only system tested that was unable to achieve Fast Ethernet line speed. The reason for this is demonstrated in Fig. 4. The total memory bandwidth (read + write) of the 43P machine is only 60 MB/s, and this seems to be insufficient for the number of data copies executed in its TCP stack. Other 43P systems (even the most recent 333 MHz CPU) have identical memory bandwidth. Furthermore, this memory speed is only 20% of the speed of the highest machine in Fig. 4, indicating that there is a wide variation in memory speeds available in systems of similar vintage.

Effect of Receiver

Contrary to the sender effects above, the influence of the receiving machine is much less pronounced, except in the extreme case of the IBM 43P machine, where memory bandwidth is insufficient to maintain more than half of line-speed. This can be seen in Fig. 5. Memory bandwidth is even more important on the the receiving side than on the sending side, since the receiving system typically has an extra copy to do (unless its ethernet driver can DMA directly into the mbuf area).

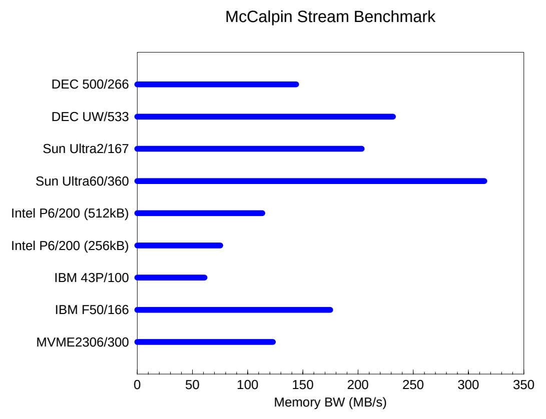

Figure 4: Memory bandwidth as measured by the McCalpin Stream benchmark [3] for a variety of systems. Note that the bandwidth reported here is the sum of read and write bandwidths ($= 2\times$ the maximum copy rate).

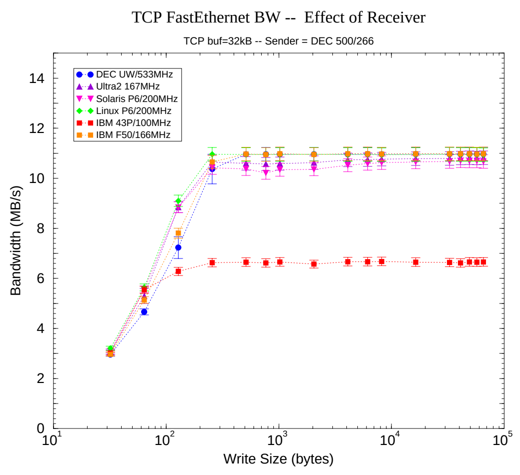

Figure 5: TCP throughput of a Digital Unix system sending to a variety of receiving hosts. With the exception of the IBM 43P receiver, all other systems show little variation in performance.

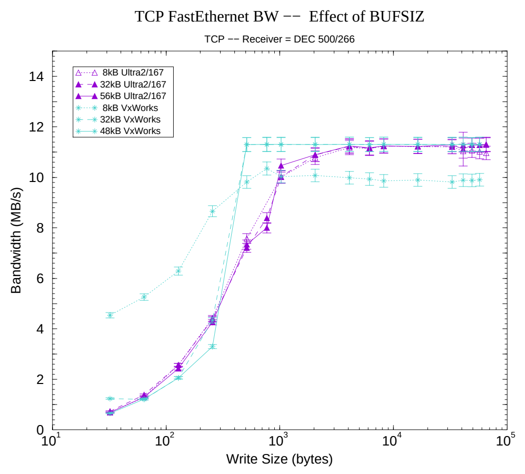

Figure 6: Effect of the TCP window size (equal to the in-kernel socket buffer size) on performance.

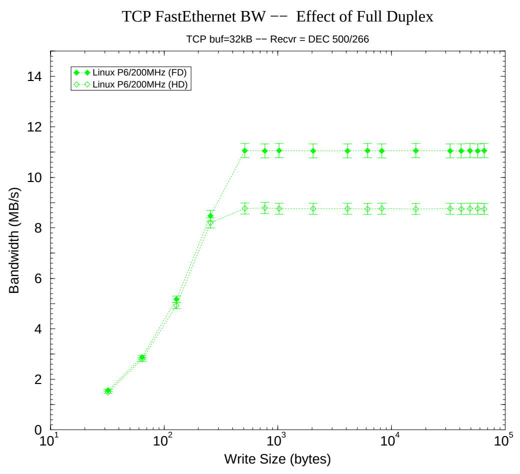

Figure 7: Effect of full-duplex versus half-duplex ethernet interfaces.

Effect of BUFSIZ

The TCP window size (`SO_SNDBUF` and `SO_RCVBUF` socket options) is well-known to be an important factor in TCP performance on WANs [4]. However, it turns out to be less critical on LANs because of the low latencies involved. At 100 Mbps, a typical round-trip latency of 0.2 ms would only fill a 2.5 kB window, indicating that even a relatively small TCP window of 4 or 8 kB is likely to be sufficient for acceptable performance.

Experimentally, we observed that window size ranges of 8 kB to 56 kB made very little difference in performance on some systems (*e.g.* Digital Unix or Solaris), but made significant effects on others (AIX and VxWorks). Figure 6 shows three buffer sizes for Solaris and for VxWorks, with VxWorks showing unusual performance in the 8 kB case. The throughput is actually higher with the 8 kB buffers for small write records, but is worse by about 10% for large records.

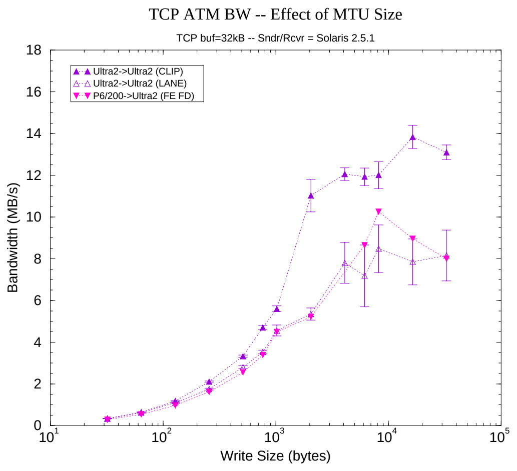

Figure 8: Effect of MTU size.

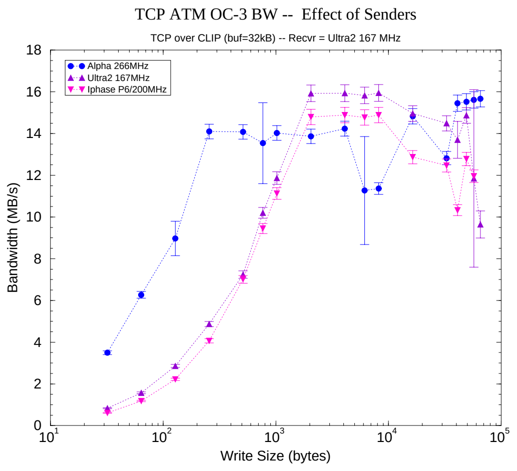

Figure 9: Effect of sender on ATM throughput.

Effect of Full-Duplex/Half-Duplex

One factor with large performance implication is whether the Fast Ethernet interface is operated in half-duplex or full-duplex mode. On several systems in which we used the same hardware in both modes, we observed full-duplex achieving throughputs of 11 MB/s, while half-duplex interfaces reached a maximum of less than 9 MB/s. This effect can be seen in Fig. 7. We attribute the effect to ethernet collisions between the almost-constant stream of data packets leaving the sender and acknowledgment packets travelling in the opposite direction.

ATM OC-3 Measurements

The other major candidate technology was ATM running over OC-3 SONET (155 Mbps base signalling rate). From the base rate of OC-3 one must subtract the ATM cell header overhead (5 bytes out of 53 or 9.4%). This still results in a 40% advantage (140 Mbps v. 99.5 Mbps) for ATM over Fast Ethernet in the theoretical maximum bandwidth available for an IP application.

There are two main protocols for IP on top of ATM, classical RFC1577 IP (CLIP) and LAN emulation (LANE). Of the two, CLIP is conceptually simpler and more performant. LANE emulates ethernet networks down to the link layer. This means that 6-byte MAC addresses, broadcasts, and multicasts are all emulated on top of ATM virtual circuits. Furthermore, it means that the ethernet MTU of 1500 bytes is retained as the maximum packet size in LANE networks (as opposed to an MTU of 9 kB in CLIP networks). Figure 8 shows the pronounced effect of this MTU size on TCP throughput using a 167 MHz UltraSPARC CPU (the difference may be smaller with faster CPUs). In Fig. 8 the LANE throughput is no larger than that of Fast Ethernet.

For these reasons, we focussed on CLIP for our ATM investigations. We tested three different ATM drivers for Unix systems (indicated in Table 1) with a Cisco LS1010 ATM switch. We also sought to test an ATM interface for VxWorks, but we were unable to make that driver work with our ATM switch. The throughput curves for the different systems are shown in Fig. 9. These shapes are very similar to those in Fig. 2, indicating that the OS kernel dominates behavior in the region of low record sizes. The plateau region for CLIP reaches 15–16 MB/s, as one would expect from the ATM line speed.

Nevertheless, we felt that this extra line speed did not justify the difficulties in coaxing ATM drivers to work. In our limited experience, we had difficulty with 3 of the 4 ATM drivers tested, contrasting with 2 out of 6 Fast Ethernet drivers which had problems. This difficulty is doubtless due to the extra complexity involved in ATM drivers. Even with CLIP, there are many more layers of software involved in

Table 2: Buffering parameters of several candidate switches.

Switch Model	Buffering
Cisco Cat 5500 (Ethernet)	32 kB input/160 kB output per port
Cisco LS1010 (ATM)	3.0 MB (payload) shared (32-port switch)
Fore ASX-200BX (ATM)	624 kB (payload) per 4 OC3 ports

an ATM driver than with a Fast Ethernet driver. Furthermore, ATM drivers are much harder to obtain for niche operating systems like VxWorks. In contrast, Fast Ethernet is rapidly becoming a technology that is delivered standard with all systems (*e.g.* on the motherboard). Hence, the software support disparity between the two technologies can be expected to grow wider.

In addition to the driver consideration, there is the disadvantage of extra cost to the ATM solution. Based on prices available in early 1998, we estimated the cost of Fast Ethernet at $400 per switch port and $50–$400 per NIC for the hosts (depending on the vendor). For ATM, we estimated the cost at $600 per switch port and $600 per NIC. This gives Fast Ethernet a price advantage of roughly a factor of two. Although secondary to the concern of building a functional system, this price advantage cannot be ignored.

Switch Performance

Demonstrating good performance for a single sender/receiver pair solves only part of the event builder requirements. There is the additional hurdle of demonstrating that the aggregate performance of a collection of N senders and M receivers will be able to maintain desired performance. In the case of very uniformly-distributed traffic (or very long time scales of averaging), there is little concern. The LAN switches we considered have backplanes that support 1–3 Gbps of traffic, while the BABAR design point is an aggregate of only 512 Mbps. However, network traffic in general is known to be self-similar [5], and event-builder traffic in particular might have timing structure that causes it to be quite non-uniformly distributed.

In an extreme case, if each of N sending nodes decided simultaneously to send data to the same receiver node, we would have $N \times 100$ Mbps of traffic entering the switch and only a single 100 Mbps channel of data draining that load. This data is buffered by the switch, but such buffers are obviously finite, and therefore the switch can only buffer up such bursts for short time scales. In particular, the relation is:

$$\mathrm{BUF} = (N-1) \times \ell \times \Delta t,$$

where BUF is the per-port buffering available, N is the number of sending nodes, ℓ is the link speed, and Δt is the maximum length of such a burst. Substituting realistic numbers from the BABAR design gives a maximum Δt of 80 μsec of simultaneous sending (or 8 kB per sender per burst).

An analysis of several potential switches shows some variation in the design of the buffering strategies, but a rough parity in the amount of buffering available per port. This information is detailed in Table 2. We found ATM switches generally to be non-blocking, while Fast Ethernet switches typically provide more ports than their backplanes can support. The per-port buffer is generally around 150 kB.

In our preliminary investigations [6], we identifed several ways to prevent the overflow of switch buffers using sender-side flow control in ATM. These included using the Available Bit Rate (ABR) traffic class (available in UNI 4.0) and setting up rate-limited permanent virtual circuits (PVCs) for each sender/receiver pair. The latter would limit each virtual circuit to $135/N$ Mbps, or about 6 Mbps in the BABAR case; nevertheless, aggregate performance would be $6 \times M$ or $\sim$90 Mbps for each sender.

Subsequent investigations revealed that one can perform similar flow control with the TCP window size. The TCP protocol guarantees that no more than window-size bytes are in flight between any sender/receiver pair. By setting the window size sufficiently low, one can definitively protect the switch buffers from overflowing. For the BABAR case of $N \simeq 22$, this means using a window size of 8 kB, which has been shown to deliver adequate performance.

Our investigations also showed that even when the sum of all window sizes exceeds the switch port buffer, the TCP dynamics conspires to make packet loss fairly minimal. This should not be too suprising, since LAN switches are a form of "store and forward" network not unlike WANs with routers, where TCP has been optimized to minimize the packet loss due to congestion. The TCP slow-start/congestion-avoidance algorithm allows senders to send packets only as fast as they receive acknowledgments back from the receiver [7]. When a packet is lost, the sender must back off to the last successful slow-start threshold. Assuming that packets from all senders have an equal probability of being passed by the switch, this drives the system stochastically into a roughly uniform division of the available bandwidth. Our experimental study showed a loss of only 41 out of 60000 packets (0.07%) with one receiver getting packets from 4 senders at a window size of 64 kB each.[3]

Conclusions

In order to validate the IP-based event builder design for BABAR, we undertook a program of investigating TCP performance over a small LAN, using both Fast Ethernet and ATM OC-3 networks. The results showed that 100 Mbps-class performance is relatively easy to achieve, provided that the systems on the network have a requisite level of memory bandwidth. Additionally, there are many interesting effects in the shape of the performance curve by OS type, particularly on the sender side.

At the current time, ATM networks work reasonably well and the candidate OC-3 technology does have a 40% line-speed advantage over Fast Ethernet. However, we found that Fast Ethernet is cheaper by a factor of two and significantly better supported in most operating systems. This drove our decision to use Fast Ethernet in the BABAR DAQ system.

In investigating the behavior of switch buffering under heavy load, we found that LANs with switches behave much like WANs, and that TCP dynamics is well-tuned for this environment. One can throttle TCP connections using the window size parameter. This should be sufficient to avoid packet loss in the BABAR environment, but even where it is not, the dynamics of TCP tend to minimize packet loss and to distribute the bandwidth stochastically among the N senders.

We expect the performance of a UDP-based event buildier to be at least as good as the TCP results, presuming a successful mechanism for flow control in the event builder. This is the direction we are currently pursuing for the BABAR event builder, but the TCP option is available as a fallback.

References

1. BABAR Technical Design Report, SLAC-R-457 (1995).
   See also `http://www.slac.stanford.edu/BFROOT/doc/TDR/`.
2. I. Scott *et al.*, "The BABAR Data Acquisition System," CHEP 98 #19, submitted to these proceedings.
3. John D. McCalpin, "STREAM: Sustainable Memory Bandwidth in Recent and Current High Performance Computers," a continually updated technical report, `http://www.cs.virginia.edu/stream/`.
   See also `http://reality.sgi.com/mccalpin/papers/bandwidth/bandwidth.html`.
4. W. Richard Stevens, *TCP/IP Illustrated, Vol. 1*, Addison-Wesley (1994) pp. 282–291.
5. W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, **2** (Feb. 1994) pp. 1–15.
6. C.T. Day, internal BABAR note.
7. V. Paxson, "Automated Packet Trace Analysis of TCP Implementations," *ACM SIGCOMM Comp. Comm. Rev.*, **27** (Oct. 1997) pp. 167–179. See also `ftp://ftp.ee.lbl.gov/papers/vp-tcpanly-sigcomm97.ps.Z`.

[3] This analysis performed using **tcptrace** by Shawn Ostermann of Ohio University. See `http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html`