

Communication Performance over a Gigabit Ethernet Network. *

Paul A. Farrell[†] Hong Ong[‡]

Department of Mathematics and Computer Science,
Kent State University, Kent, OH 44242.

[†] farrell@mcs.kent.edu [‡] hong@mcs.kent.edu

Abstract

Cluster computing imposes heavy demands on the communication network. Gigabit Ethernet technology can provide the required bandwidth to meet these demands. However, it has also shifted the communication bottleneck from network media to protocol processing. In this paper, we present an overview of Gigabit Ethernet technology and study the end-to-end Gigabit Ethernet communication bandwidth and latency. Performance graphs are collected using NetPipe which clearly show the performance characteristics of TCP/IP over Gigabit Ethernet. These indicate the impact of a number of factors such as processor speeds, network adaptors, versions of the Linux Kernel and device drivers, and TCP/IP tuning on the performance of Gigabit Ethernet between two Pentium II/350 PCs. Among the important conclusions are the marked superiority of the 2.1.121 and later development kernels and 2.2.x production kernels of Linux and that the ability to increase the MTU beyond the Ethernet standard of 1500 could significantly enhance the throughput attainable.

1 Introduction

Cluster computing, such as that possible with Beowulf class clusters, offers great potential for increasing the amount of computing power and communication resources available to large scale applications. It is likely that the combined computational power of a cluster of powerful PCs connected to a high speed network may exceed a stand-alone high performance supercomputer.

Running large scale parallel applications on a cluster imposes heavy demands on the communication network. Therefore, in the early 1970's, one of the design goals for cluster computing was to limit the amount of communication between hosts. However, due to the features of some applications, a certain degree of communication between hosts may be required. As a result, the performance bottleneck of the network severely limited the potential of cluster

computing. Recent emerging high speed networks such as Asynchronous Transfer Mode (ATM), Fibre Channel (FC), and Gigabit Ethernet change the situation somewhat.

These high speed networks offer raw bandwidth ranges from 100 megabits per second (Mbps) to 1 gigabit per second (Gbps) satisfying the communication needs of many parallel applications. However, the maximum achievable bandwidth at the application level is still far away from the theoretical peak bandwidth. This major roadblock to achieving high speed cluster communication is caused by the time required for the interaction between software and hardware components.

In this paper, we discuss the communication performance attainable with a PC cluster connected by a Gigabit Ethernet network. Gigabit Ethernet is the third generation of Ethernet technology and offers raw bandwidth of 1 Gbps. The focus of this work is to discuss the Gigabit Ethernet technology, to evaluate and analyze the end-to-end communication latency and achievable bandwidth, and to monitor the effects of software and hardware components on the overall network performance.

This paper is organized as follows. Section 2 gives an overview of Gigabit Ethernet technology. In Section 3, we describe the hardware and software test environment. In Section 4, the end-to-end TCP communication characteristics are presented. Finally, conclusions and a discussion of future work are presented in Section 5.

2 An Overview of Gigabit Ethernet Technology

Gigabit Ethernet, also known as IEEE Standard 802.3z, is the latest Ethernet technology. Like Ethernet, Gigabit Ethernet is a media access control (MAC) and physical-layer (PHY) technology. It offers one gigabit per second (1 Gbps) raw bandwidth which is 10 times faster than fast Ethernet and 100 times the speed of regular Ethernet. In order to achieve 1 Gbps, Gigabit Ethernet uses a modified version of the ANSI X3T11 Fibre Channel standard physical layer (FC-0). To remain backward compatible with existing Ethernet technologies, Gigabit Ethernet uses the same IEEE 802.3 Ethernet frame format, and a compatible

*This work was supported in part by NSF CDA 9617541, NSF ASC 9720221, by the OBR Investment Fund Ohio Communication and Computing ATM Research Network (OCARnet), and through the Ohio Board of Regents Computer Science Enhancement Initiative

full or half duplex carrier sense multiple access/ collision detection (CSMA/CD) scheme scaled to gigabit speeds.

Like its predecessor, Gigabit Ethernet operates in either half-duplex or full-duplex mode. In full-duplex mode, frames travel in both directions simultaneously over two channels on the same connection for an aggregate bandwidth of twice that of half-duplex mode. Full duplex networks are very efficient since data can be sent and received simultaneously. However, full-duplex transmission can be used for point-to-point connections only. Since full-duplex connections cannot be shared, collisions are eliminated. This setup eliminates most of the need for the CSMA/CD access control mechanism because there is no need to determine whether the connection is already being used.

When Gigabit Ethernet operates in full duplex mode, it uses buffers to store incoming and outgoing data frames until the MAC layer has time to pass them higher up the legacy protocol stacks. During heavy traffic transmissions, the buffers may fill up with data faster than the MAC can process them. When this occurs, the MAC layer prevents the upper layers from sending until the buffer has room to store more frames; otherwise, frames would be lost due to insufficient buffer space.

In the event that the receive buffers approach their maximum capacity, a high water mark interrupts the MAC control of the receiving node and sends a signal to the sending node instructing it to halt packet transmission for a specified period of time until the buffer can catch up. The sending node stops packet transmission until the time interval is past or until it receives a new packet from the receiving node with a time interval of zero. It then resumes packets transmission. The high water mark ensures that enough buffer capacity remains to give the MAC time to inform the other devices to shut down the flow of data before the buffer capacity overflows. Similarly, there is a low water mark to notify the MAC control when there is enough open capacity in the buffer to restart the flow of incoming data.

Full-duplex transmission can be deployed between ports on two switches, a workstation and a switch port, or between two workstations. Full-duplex connections cannot be used for shared-port connections, such as a repeater or hub port that connects multiple workstations. Gigabit Ethernet is most effective when running in the full-duplex, point-to-point mode where full bandwidth is dedicated between the two end-nodes. Full-duplex operation is ideal for backbones and high-speed server or router links.

For half-duplex operation, Gigabit Ethernet will use the enhanced CSMA/CD access method. With CSMA/CD, the same channel can only transmit or receive at one time. A collision results when a frame sent from one end of the network collides with another frame. Timing becomes critical if and when a collision occurs. If a collision occurs

during the transmission of a frame, the MAC will stop transmitting and retransmit the frame when the transmission medium is clear. If the collision occurs after a packet has been sent, then the packet is lost since the MAC has already discarded the frame and started to prepare for the next frame for transmission. In all cases, the rest of the network must wait for the collision to dissipate before any other devices can transmit.

In half-duplex mode, Gigabit Ethernet's performance is degraded. This is because Gigabit Ethernet uses the CSMA/CD protocol which is sensitive to frame length. The standard slot time for Ethernet frames is not long enough to run a 200-meter cable when passing 64-byte frames at gigabit speed. In order to accommodate the timing problems experienced with CSMA/CD when scaling half-duplex Ethernet to gigabit speed, slot time has been extended to guarantee at least a 512-byte slot time using a technique called *carrier extension*. The frame size is not changed; only the timing is extended.

Half-duplex operation is intended for shared multistation LANs, where two or more end nodes share a single port. Most switches enable users to select half-duplex or full-duplex operation on a port-by-port basis, allowing users to migrate from shared links to point-to-point, full-duplex links when they are ready.

Gigabit Ethernet will eventually operate over a variety of cabling types. Initially, the Gigabit Ethernet specification supports multi-mode and single-mode optical fiber, and short haul copper cabling. Fiber is ideal for connectivity between switches and between a switch and high-speed server because it can be extended to greater length than copper before signal attenuation becomes unacceptable and it is more reliable than copper. In June 1999, the Gigabit Ethernet standard was extended to incorporate category 5 unshielded twisted-pair (UTP) copper media. The first switches and network interface cards (NICs) using category 5 UTP became available at the end of 1999.

2.1 Enhanced CSMA/CD

The MAC layer of Gigabit Ethernet uses the same CSMA/CD protocol as defined in IEEE 802.3. As a result, the maximum network diameter used to connect nodes is limited by the CSMA/CD protocol.

IEEE 802.3 (10BaseT) defined the original CSMA/CD mechanism. This scheme ensures that all nodes are granted access to a physical media on a first come, first serve basis. The maximum network diameter in 10BaseT is limited to 2000 m. This distance limitation is due to the relationship between the time (also known as slot time) required to transmit a minimum frame of 64 bytes and the ability to detect a collision (a limit known as propagation delay). When a collision occurs, the MAC layer detects it and sends a halt signal to cause the transmitting nodes to stop transmitting

and enter a backoff phase prior to retrying transmission.

When the IEEE defined 802.3u (100BaseT) in 1994, it maintained the Ethernet framing format and raised the speed limit to 100Mbit/s. As the bit rate increases, the time needed to transmit a frame is reduced by a factor of 10. This implies that the network diameter is decreased from 2000 m for 10BaseT to 200 m for 100BaseT.

Since IEEE 802.3z represents another tenfold increase in bit rate as compared to 100BaseT, the network diameter is further reduced by another factor of 10. But, a network diameter of 20 m is clearly too short for most network configurations and is thus impractical. In addition, this distance is even less if delays in active components such as repeaters are considered. Moreover, with today's silicon technology, it is not yet feasible for vendors of repeater chips operating with a 25 MHz clock to scale up to operate with a 250 MHz clock.

As a consequence, the IEEE 802.3z working committee redefined the MAC layer for Gigabit Ethernet and introduced a mechanism that will preserve the 200 m collision domain of 100BaseT. This is necessary because two nodes, which are 200 m apart, will not be able to detect a collision when both simultaneously transmit a 64 byte-frame at gigabit speed. This inability to detect collisions will eventually lead to network instability.

The mechanism to preserve the 200 m network diameter is known as *carrier Extension*. Carrier extension, developed by Sun Microsystems Inc. (Mountain View, California), is a way of maintaining IEEE 802.3 minimum and maximum frame size with a meaningful network diameter. The resultant mechanism leaves the CSMA/CD algorithm unchanged. Carrier extension increases the slot time to 512 bytes rather than the 64 bytes defined in IEEE 802.3. If the frame is shorter than 512 bytes, then it is transmitted in a 512 byte window and the transmitted frame is padded with carrier extension symbols.

Upon receipt of a frame carrying carrier extension symbols, the entire extended frame is considered for collision and dropped if necessary. However, the Frame Check Sequence (FCS) is calculated only on the original (without extension symbols) frame. The extension symbols are removed before the FCS is checked by the receiver. So, the logical link control layer is not even aware of the carrier extension.

Carrier Extension wastes bandwidth. For example, a small packet of 64 bytes will have 448 padding bytes of carrier extension symbols. This clearly results in low throughput and an increased collision rate which may increase the number of lost frames. In fact, for a large number of small packets, the Gigabit Ethernet throughput is only marginally better than 100BaseT.

To gain back some of the performance lost due to car-

rier extension, NBase Communication (Chatsworth, California) proposed a solution known as packet bursting. It is essentially a modification to the carrier extension procedure. The idea is to transmit a burst of frames every time the first frame has successfully passed the collision window of 512 bytes. Carrier extension is only applied to the first frame in a burst. This essentially averages the wasted time in the carrier extension symbols over the few frames that are transmitted. Packet bursting substantially increases the throughput and does not change the dynamics of the CSMA/CD algorithm. It only slightly modified the existing MAC definition.

3 Testing Environment and Network Parameters

This section describes the testing environment for the communication benchmarking that we conducted. First, we will present the hardware and software environment. Next, we present how TCP parameters are tuned to accommodate gigabit speeds and discuss some performance issues related to TCP.

3.1 Hardware

The experiments conducted to evaluate the communication latency and throughput were performed on a cluster of two Pentium II PCs running at 350MHz with 64MB 100MHz (PC100) SD-RAM. The PCs are connected back to back via Packet Engine GNIC-II Gigabit Ethernet adaptors installed in the 33MHz PCI slot. The machines are also connected together through an SVEC 5 port 10/100 Mbps autosensing/autoswitching hub via 3Com PCI 10/100 Mbps Ethernet Cards (3c905B). The machines are isolated from other network traffic. This is important for the accuracy of the tests. In later tests, we swapped the GNIC-II adaptors for Alteon ACEnic adaptors. In addition, several tests were performed using a Celeron processor running at 300 MHz with 66 MHz bus overclocked to 100MHz thus giving an effective processor speed of 450MHz.

3.2 Software

The machines run RedHat Linux. Several versions of the Linux kernel were installed and tested. The kernel version used for benchmarking will be indicated later in the performance graphs.

The software program used to test the communication performance was NetPipe (version 2.1) [1]. NetPipe is a network performance analysis tool developed at Ames Laboratory. It provides accurate and useful information to reveal the network performance for each different block size. This program can be obtained from <ftp://ftp.scl.ameslab.gov/pub/netpipe>. NetPipe increases the transfer block size from a single

byte to large blocks until transmission time exceeds 1 second. Specifically, for each block size c , three measurements are taken for block sizes $c-p$ bytes, c bytes and $c+p$ bytes, where p is a perturbation parameter with a default value of 3. This allows examination of block sizes that are possibly slightly smaller or larger than an internal network buffer. NetPipe clearly shows the overhead associated with different protocol layers, in particular TCP. NetPipe was also slightly modified locally to replace the read/write system calls with send/recv system calls. This improved the robustness of the code with experimental drivers.

3.3 Network Parameters

Since TCP was originally engineered to provide a general transport protocol, it is not by default optimized for streams of data coming in and out of the system at high transmission rates (e.g 1Gbps). The RFC on TCP/IP Extensions for High Performance (RFC 1323 [9]) defines a set of TCP parameters to improve performance over large $bandwidth \times delay$ paths and provide reliable operations over high speed paths. Systems that need to comply with RFC 1323 can be configured in the following ways:

- Systems must use Path MTU Discovery specified in RFC 1191. It allows the largest possible packet size to be set, rather than the default of 512 bytes.
- The host systems must support RFC1323 "Large Windows" extensions to TCP. This RFC defines a set of TCP extensions to improve performance over large $bandwidth \times delay$ paths and to provide reliable operation over high-speed paths.
- The host system must support large enough socket buffers for reading and writing data to the network. Without RFC1323 "Large Windows", TCP/IP does not allow applications to buffer more than 64KB in the network, which is inadequate for almost all high speed paths.
- The application must set its send and receive socket buffer sizes (at both ends) to at least the $bandwidth \times delay$ product of the link.

In addition, TCP may experience poor performance when multiple packets are lost from one window of data. With the limited information available from cumulative acknowledgments, a TCP sender can only learn about a single lost packet per round trip time. An aggressive sender could choose to retransmit packets early, but such retransmitted segments may have already been successfully received. A Selective Acknowledgment (SACK) mechanism, combined with a selective repeat retransmission policy, can help to overcome these limitations. The receiving TCP sends back SACK packets to the sender informing the sender of data that has been received. The

sender can then retransmit only the missing data segments. RFC 2018, TCP Selective Acknowledgments (SACK), is in the process of being standardized. Further information on SACKs could be obtained from [10] <http://www.aciri.org/floyd/sacks.html>.

Linux kernel 2.0.x supports path MTU discovery but neither RFC1323 nor SACKs are supported. For Linux kernel 2.1.x and later, the TCP/IP suite has been upgraded to meet the extensions specified in RFC 1323, SACKs, and path MTU discovery. Two socket buffers sizes, 64KB and 128KB, were used for benchmarking. Note that the maximum socket buffer size is 64KB ($2 \times 32KB$ where 32KB is the default socket size) in Linux kernel 2.0.x and 128KB ($2 \times 64KB$) in Linux 2.1.x and later. In addition to RFC 1323, we also set the TCP_NODELAY option to disable the "Nagle" effect, which delays transmission of small packets, so that we do not get overestimates of performance.

4 Performance Results

For the purpose of this paper, we are interested in how network adaptors and processor speeds affect gigabit throughput, how TCP/IP performs on Gigabit Ethernet networks, what is the impact that different versions of the Linux kernel and device drivers have on Gigabit Ethernet throughput, and what is the maximum attainable throughput and minimum latency that can be achieved for our system configuration.

The throughput graph is plotted using throughput versus transfer block size. Throughput is reported in megabits per second (Mbps) and block size is reported in bytes since they are the common measurements used among vendors. The throughput graph clearly shows the throughput for each transfer block size and the maximum attainable throughput. The throughput graph combined with application specified requirements will help programmers to decide what block size to use for transmission in order to maximize the achievable bandwidth.

Another important network performance measure is latency. In NetPipe, the latency is determined from the signature graph. This graph is plotted using throughput per second versus total transfer time elapsed in the test. The network latency coincides with the time of the first data point on the graph.

In the remainder of this section, we present a detailed investigation of Gigabit Ethernet performance on the testbed described earlier.

4.1 Kernel Comparison

The graphs in Figure 1 show the performance of GNIC-II Gigabit Ethernet adaptors for different versions of the Linux kernel. In these tests, we used the Hamachi v0.07 driver for the GNIC-II and the socket size is set to 64KB. This is because the maximum socket size in kernel 2.0.35 is only 64KB and the Hamachi v0.07 was the only driver

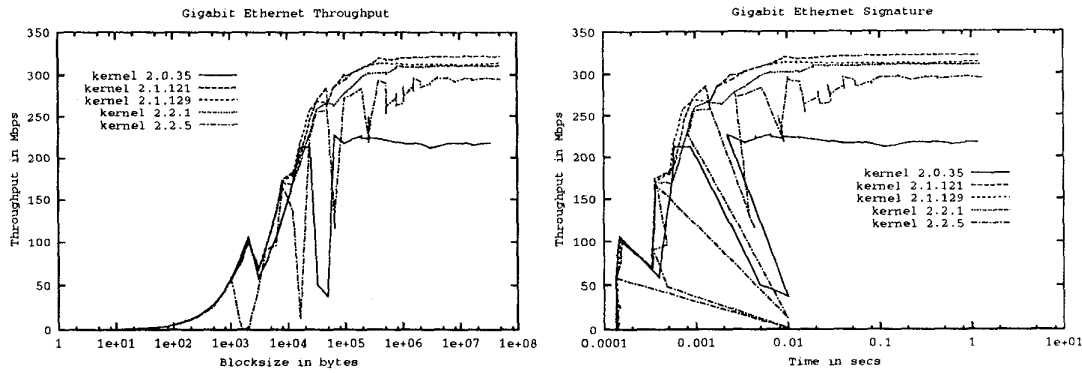


Figure 1: Kernel Progression: Throughput and Signature Graphs

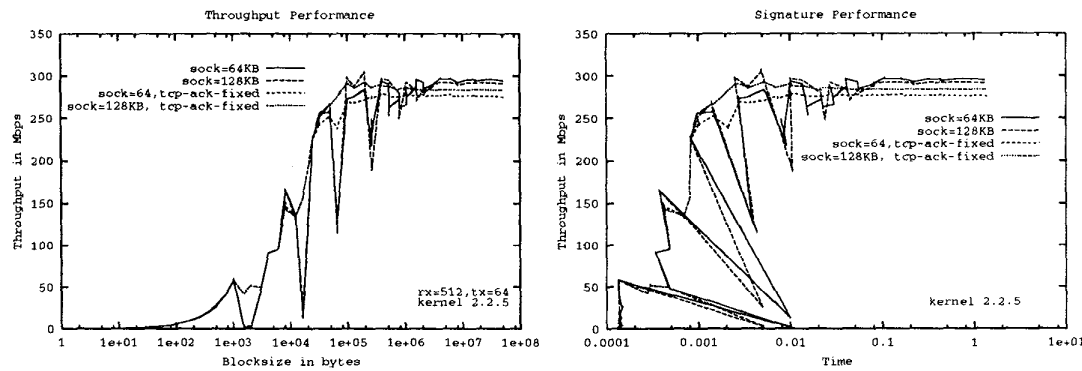


Figure 2: Effects of Fixing the TCP-ACK Problem

available at that time. We have modified the Hamachi v0.07's descriptor length to 32 for the send list and 64 for the receive list instead the default size of 16 and 32.

In kernel 2.0.35, the maximum attainable throughput was approximately 226 Mbps for message size of 191KB. However, from the figure, we also see that there is a 80% drop in performance for transfer block size ranges from 32KB to 64KB. This probably related to the maximum default TCP socket buffer size of 32KB. We see that this problem is eliminated when the maximum default TCP socket buffer size is increased to 64KB in Linux kernel 2.1.x.

Using kernel 2.1.x and kernel 2.2.1, the curves are free of severe drop out. The maximum attainable throughput for these graphs is approximately 310 – 322 Mbps for transfer block sizes greater than 1MB.

The sudden drop of performance (roughly 20-30 Mbps) at 3KB is consistent across kernel 2.0.x, kernel 2.1.x, and kernel 2.2.1. This performance anomaly is related to the size of MTU (1500). As we will see later, this performance drop will be shifted to the right as we increase the MTU

size. However, the cause of this anomaly is yet to be determined. Regardless of which kernel versions is used, the latency is roughly 135 – 140 μ seconds.

Linux kernel 2.2.5 has many severe drops in the performance curves. These are caused by Linux kernel delayed TCP acknowledgments even when the TCP_NODELAY flag was set. Using Andrea Arcangeli's proposal ¹, we modified the Linux TCP suite and were able to get smoother throughput graphs although the peak throughput was marginally reduced. Figure 2 shows the throughput and signature graphs of the modified TCP driver.

4.2 Processor Comparison

Three types of processors were tested to see how processor speed effects the GNIC-II Gigabit Ethernet throughput and latency performance.

In Figure 3, we see that faster processors can attain higher throughput for large transfer block sizes, that is blocks greater than 1MB. The maximum attainable

¹A detailed discussion of TCP acknowledgment problem can be found in <http://linux.ucs.indiana.edu/hypemal/linux/kernel/9903.0/0208.html> and <http://www.icase.edu/coral/LinuxTCP.html>

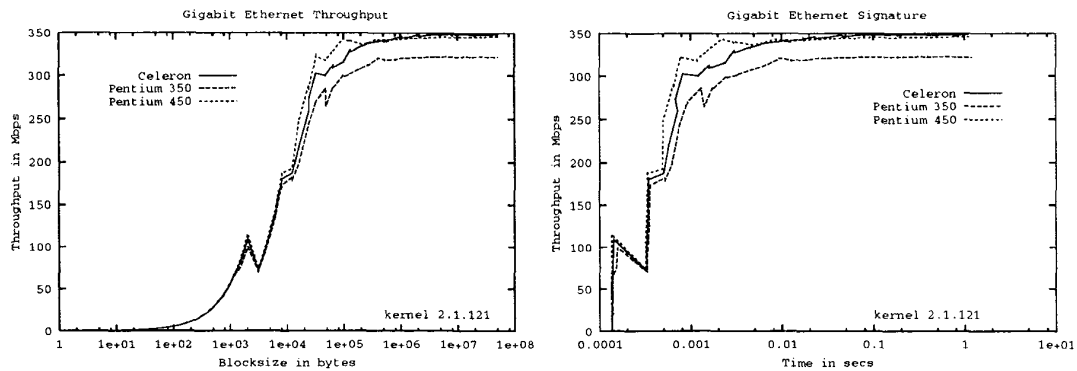


Figure 3: Processor Comparison: Throughput and Signature Graphs

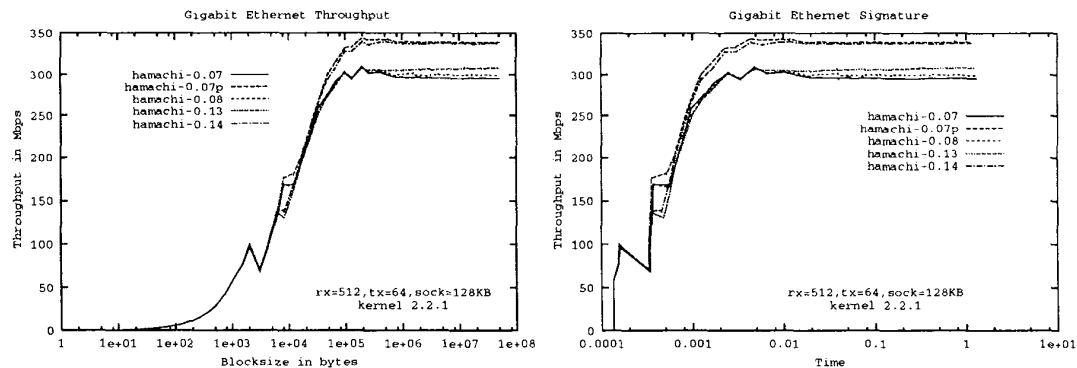


Figure 4: Hamachi Driver Comparison with TCP Socket Buffer of 128KB

throughput is approximately 348 Mbps for the Celeron and the Pentium II 450 and 320 Mbps for the Pentium II 350. This is largely due to the fact that faster processors can process the protocol stacks and calculate TCP checksums faster than the slower processors. Note that we are not obtaining the 29% increase in throughput to 411 Mbps which one would expect for a 450 MHz processor, if the processor speed were the only or dominant factor in attainable throughput. For transfer block sizes less than 4KB, the performance is approximately the same for all the processors tested. This is because the latency, approximately 139μ seconds, is dominating the throughput for these smaller transfers.

The experiment clearly shows that processor speed is a factor in Gigabit Ethernet network performance.

4.3 Driver Comparison

In this section, we show results of experiments to determine how different versions of the Hamachi drivers affect performance. Figure 4 plots the throughput performance results using socket buffer size of 128KB. The corresponding signature graph is also plotted.

Hamachi driver versions v0.07 and v0.08 were written by Donald Becker. Hamachi v0.13, v0.14, and v0.07p are all based on Donald Becker's Hamachi driver. Hamachi v0.13 and v0.14 were written by Eric Kasten. Hamachi v0.07p is written by Pete Wyckoff. In general, the later versions were created to enhance stability and robustness rather than to increase throughput. However versions v0.14, and v0.07p support hardware checksumming on the receiving side.

Hamachi drivers, which support hardware checksumming, have better performance as evidenced by the graphs. For example, incorporating hardware checksumming in v0.07 raises the peak throughput from 280 Mbps for v0.07 to 320 Mbps for v0.07p. The latency, however, has remained consistent throughout all Hamachi drivers.

4.4 MTU Comparison

To preserve compatibility with 10 Mbps and 100 Mbps Ethernet, the Gigabit Ethernet standard still limits the Maximum Transmission Unit (MTU), that is the maximum frame size that can be transmitted, to 1500 bytes. Standards bodies are reluctant to change this since, among other

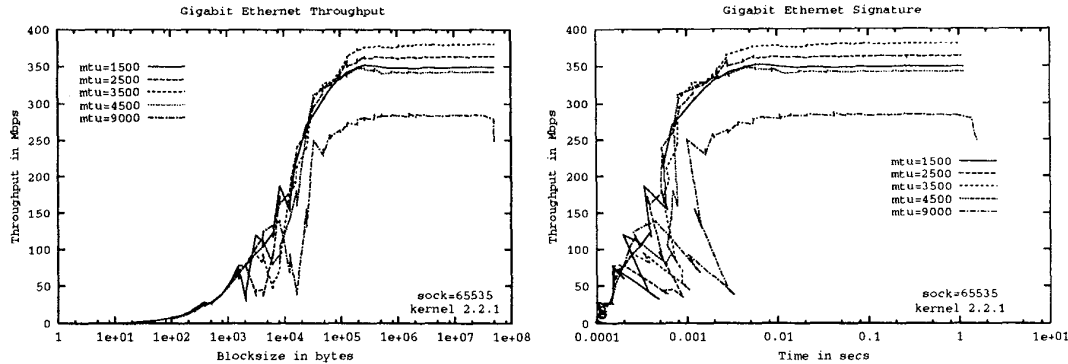


Figure 5: MTU Comparison with TCP Socket Buffer of 64KB

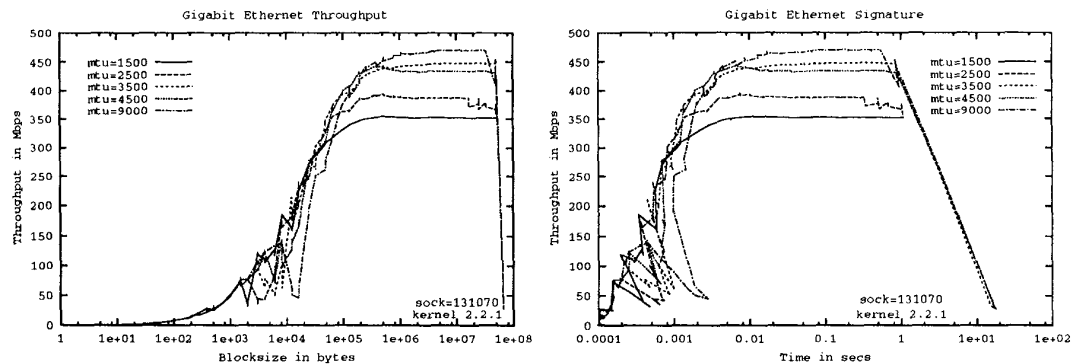


Figure 6: MTU Comparison with TCP Socket Buffer of 128KB

issues, they wish to avoid the complications in specifying how larger frames transitioning from networks with MTU greater than 1500 to ones with MTU of 1500 should be handled. This would be a fairly widespread transition if Gigabit Ethernet supported MTUs greater than 1500, since the slower Ethernet standards do not. One of the common uses for Gigabit Ethernet is expected to be in aggregating switches which take multiple 100 Mbps input streams from workstations and output to other Gigabit Ethernet switches on a gigabit link. An efficient implementation of Gigabit Ethernet with MTU greater than 1500 bytes would probably require switches to resegment Ethernet frames greater than 1500 bytes and recompute the checksums. This would add to the cost of switches.

It is to be expected that high speed networks such as Gigabit Ethernet would benefit from an MTU larger than 1500. This is an effect which has been noted for other high speed networks, such as FDDI, ATM [12] and Fibre Channel. In addition to improving the throughput, one would expect that a larger MTU would also reduce the load on the CPU by reducing the number of frames, which would

need to be processed for large message sizes. As a result of these factors, some companies, notably Alteon, have enhanced the Gigabit Ethernet functionality by adding a facility to support MTUs and hence frame sizes greater than 1500 bytes. Alteon coined the name Jumbo Frames for this functionality, and their network interface cards (NICs) and switches support Jumbo Frames of up to 9000 bytes.

To confirm these hypotheses about performance, we tested the Alteon ACEnic adaptor, using different MTU sizes. Figure 5 and Figure 6 show the throughput using socket buffer sizes of 64KB and 128KB, respectively.

There are three interesting observations from these figures. The maximum achievable throughput is approximately 470 Mbps which is obtained using 128KB socket buffer size and MTU equal to 9000 bytes. However, for socket buffer size equal to 64KB, the maximum achievable throughput is only 380 Mbps and the optimal MTU is equal to 3500 bytes rather than 9000 bytes. These results confirm the hypothesis that large socket buffers and large MTU will give the better throughput. However, for socket buffer size less than or equal to 64KB, the maxi-

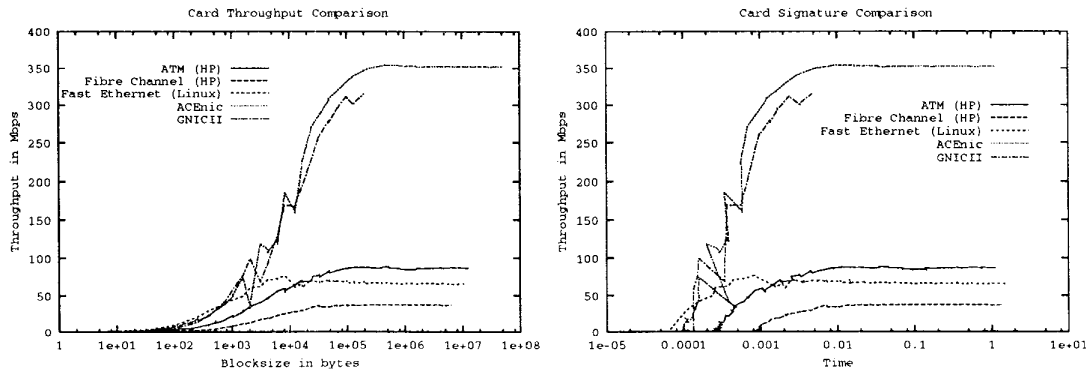


Figure 7: Comparison of High Speed Networks

mum MTU should only be set to roughly 3500 bytes. In fact, for MTU greater than 3500 bytes, sending messages of size greater than 1MB using 64KB socket buffers will result in decreased performance.

The second observation is the effect of MTU on the anomaly mentioned earlier. We observed that the drop at 3KB for 1500 bytes MTU starts to shift as we increase the MTU size regardless of socket buffer size.

The third observation is the performance drop for block size greater than 64MB. This is not a surprise at all since our machines are only equipped with 64MB of memory and thus we are seeing the bandwidth limitation of accessing virtual memory on disk rather than the limitation of network throughput. However, this observation further confirms that processor speed and memory will become network bottlenecks in a cluster connected via Gigabit Ethernet since it will be faster to access remote data through the network rather than local data on disk.

Figure 5 and Figure 6 also show the signature graphs for socket buffer sizes equal to 64KB and 128KB respectively. The latency for a single byte ranges from 177μ to 248μ seconds for a 64KB socket buffer and from 169μ to 249μ seconds for a 128KB socket buffer.

4.5 High Speed Networks Comparison

In this section, we will compare the performance of various high speed network technologies. Figure 7 shows the comparison graphs. We remark that these are less scientifically rigorous results, since the tests of ATM and Fibre Channel were performed on Hewlett-Packard C180 processor with 128 MB of memory rather than on the PCs used for the Gigabit Ethernet tests.

From the figures, it is clear that the ACEnic adaptor has higher throughput and lower latency than the GNIC-II adaptor. In fact, ACEnic is about 11% higher in throughput and 15% lesser in latency than GNIC-II.

From these figures, it is also clear that Gigabit Ethernet

outperforms ATM, Fibre Channel and Fast Ethernet by an order of magnitude. However, Fast Ethernet has the lowest latency among these competing network technologies.

5 Conclusion

Communication performance is affected by a number of factors, including CPU speed, I/O speed, bus architecture, network adaptors, device drivers, and protocol stack processing. While most of these factors do not contribute significantly to the performance of slower networks, they begin to become factors in high speed networks.

We studied the emerging and promising high speed technology, Gigabit Ethernet. Gigabit Ethernet provides the bandwidth required to meet the demands of current and future applications. However, it has also shifted the communication bottleneck from network media to hardware and software components. It is critical to improve these components in order to achieve high speed transmission.

In this paper, a detailed study of the communication latency and throughput of Gigabit Ethernet was presented with emphasis on the TCP/IP protocol stack processing. Various Linux kernel and device drivers were monitored to track Gigabit Ethernet performance. It was concluded that Gigabit Ethernet performance was significantly enhanced by the improved TCP/IP implementations in the 2.1.121 and later Linux development kernels and the 2.2.x production kernels. The effect of processor speed on Gigabit Ethernet throughput was analyzed and it was shown that, although improved performance was obtained with faster processors, the improvement in performance was not proportional to the increase in processor speed. This leads to the conclusion that processor speed is not the major bottleneck in obtaining improved throughput.

The performance results of two network adaptors, ACEnic and GNIC-II, were also presented. In order to tune the TCP/IP protocol for high speed networks, we analyzed the effects of a number of TCP parameters and in-

licated the optimal values for our test environment. We also showed that the ability to increase the MTU beyond 1500, that is to transmit Jumbo Frames, could significantly enhance the throughput attainable. More surprisingly, we showed that the optimum choice of MTU was not always the largest. In order to make the largest MTU selectable the optimal choice, one also has to ensure that the TCP socket buffer size is large enough.

We intend to continue tracking Gigabit Ethernet performance for newer Linux kernels and device drivers. More interestingly, newer developments such as M-VIA, an implementation of the VI architecture for Linux, hold great promise of reducing latency significantly and increasing throughput to a lesser degree. Since the VI architecture defines a lower overhead standard interface between applications and the network interface card, we believe it could lessen the impact on high speed network performance of the overhead of protocol processing.

References

- [1] Q.O. Snell, A.R. Mikler, and J.L. Gustafson, "Net-PIPE: Network Protocol Independent Performance Evaluator", Ames Laboratory/ Scalable Computing Lab, Iowa State., 1997.
- [2] Stephen Saunders, "Data Communication Gigabit Ethernet Handbook", McGraw Hill, ISBN 0-07-057971-7, 1998.
- [3] S. Elbert, et al, C. Csanady, "Gigabit Ethernet and Low-Cost Supercomputing".
[http://www.scl.ameslab.gov
/Publications/Gigabit/tr5126.html](http://www.scl.ameslab.gov/Publications/Gigabit/tr5126.html)
- [4] Packet Engines, "Gigabit Ethernet Networking with Full-Duplex Repeater Technology", 1999.
<http://www.packetengines.com/>
- [5] Packet Engines, "Gigabit Ethernet vs. ATM Networks", 1999.
<http://www.packetengines.com/>
- [6] Gigabit Ethernet Alliance, "Gigabit Ethernet Over Copper", March, 1999.
<http://www.gigabit-ethernet.org/>
- [7] Gigabit Ethernet Alliance, "Gigabit Ethernet Overview", 1997.
<http://www.gigabit-ethernet.org/>
- [8] Joe Skorupa and George Prodan, "Battle of the Backbones: ATM vs. Gigabit Ethernet", Data Communications, April 1997,
[http://www.data.com/tutorials/
backbones.html](http://www.data.com/tutorials/backbones.html)
- [9] Jacobson, Braden, & Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.
- [10] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018 October 1996.
- [11] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, B. Volz , "Known TCP Implementation Problems", RFC 2525, March 1999.
- [12] P.A. Farrell, H. Ong, A. Ruttan, "Modeling liquid crystal structures using MPI on a workstation cluster", to appear in the Proc. of MWPP99.