Experiences Deploying a 10 Gigabit Ethernet Computing Environment to Support Regional Computational Science

Jason Cope[1], Theron Voran[1], Matthew Woitaszek[1], Adam Boggs[1,2], Sean McCreary[1,2], Michael Oberg[2], and Henry M. Tufo[1,2]
{copej, voran, woitasze}@colorado.edu
{boggs, seanm, oberg, tufo}@ucar.edu

[1] University of Colorado, Boulder, CO
[2] National Center for Atmospheric Research, Boulder, CO

Abstract. With the advent of Grid computing technology and the continued improvements to high-performance network infrastructure, computational science in distributed computing environments has become an essential research platform for scientists that require access to distributed computational resources, scientific data archives, or Grid-enabled scientific instruments. The development of these environments is challenging because the allocation of staff, funding, and resource usage between the collaborating resource providers and users is a difficult problem to address. Our solution is to leverage the recent advances in 10 Gigabit Ethernet network infrastructure and Grid computing technology to deploy a cost effective, distributed computing environment. This paper focuses on the use of these high-performance network products, including 10 Gigabit Ethernet products from Myricom and Force10 Networks, as an integration tool and the potential consequences of deploying this infrastructure in a legacy computing environment.

1 Introduction

Today's computational scientists often develop complex workflows in their investigations that span multiple sites and cross organizational borders. The simulation data, as well as the resources for computation and visualization, are often geographically split. For example, Earth scientists may require access to historical climate data hosted at the National Center for Atmospheric Research (NCAR) in Boulder, CO and the geographically separated computational resources provided by the TeraGrid. Grid computing and federated storage technologies seek to make data and computational resources available to scientists regardless of location. The development of this technology coincides with the deployment of specialized wide-area networks (WANs) such as the Teragrid and National LambdaRail, which provide high-speed dedicated links between sites.

One of the barriers for utilization of these networks is the equipment for connecting to these networks. Traditionally, such equipment is quite costly (such as SONET). The advent of 10 Gigabit Ethernet (10GbE) provides the ability to connect via host adapters, reducing cost and complexity. Combined with the appropriate software infrastructure (GridFTP[9], etc.), 10GbE technology provides an opportunity for large-scale collaboration between institutions of moderate size at reasonable cost.

In this paper, we present our evaluation of 10GbE network equipment intended for use in a Grid-enabled or federated computing environment. We deploy a testbed between NCAR and the University of Colorado at Boulder (CU) consisting of legacy computing systems and 10GbE network equipment from Myricom and Force10 Networks. Our evaluation focuses on the performance of this deployment for network transports and communication libraries, data transfer tools, and network security tools operating within our legacy computing environment.

2 Background

Ethernet has closely followed the latest networking trends, frequently reinventing itself to remain relevant in the rapidly-changing world of high-performance networking. Ethernet is ubiquitous; it is a familiar and well-understood technology in the world of high-performance computing and numerous examples of its use can be found from the first Beowulf cluster [15] to systems on the current Top500 list [8].

The relatively recent development of 10GbE technology brings Ethernet to the WAN arena, providing a means of connecting individual hosts to high-speed WAN environments. Before the advent of 10GbE, most WANs were constructed using synchronous optical networking (SONET), which is much more costly and complicated in terms of the equipment required. The ability to connect commodity hosts at 10Gb/s speeds makes 10GbE a cost-effective solution for cluster interconnects, SAN connectivity, and coupling institutions over long distances.

The Myricom Myri-10G card we evaluate here joins a herd of other host adapters including the Intel PRO/10GbE LR, the Chelsio T110, the Neterion NXB 10G line, and the LeWiz Talon3220. Of these, the Intel and the Chelsio cards are based on the PCI-X bus. The rest use PCI-Express.

The Myri-10G card operates in either 10-Gigabit Ethernet and 10-Gigabit Myrinet mode (MX). It uses an 8x PCI-Express slot, providing up to 2 GBytes/sec in each direction, full-duplex. The card is based on the Lanai Z8E chip and has 2MB local memory. Myricom purposely did not include a full TCP Offload Engine (TOE) on the card, instead providing several stateless offloads for interrupt coalescing, IP and TCP checksuming, TCP segmentation, receive-side scaling, large receive offload, and multicast filtering [3]. The Myri-10G card operates with 10GBase-SR, 10GBase-LR, and 10GBase-ER optics. We used 10GBase-LR optics for the work in this paper.

3 Previous Work

Work related to our findings includes performance evaluations of 10-Gigabit Ethernet adapters and experiments building infrastructure around such adapters.

Most of the previous 10GbE performance evaluation was performed with the Intel PRO/10GbE LR PCI-X adapter, most likely because it was the first 10GbE host-based adapter to become available. This card operates on the PCI-X bus, which has a peak bandwidth of 8.5 Gb/s. In [10], Feng et al. found the card to have a top throughput of 7.2 Gb/s and a l2 μs latency, after some tuning and tweaking. Some of the tuning included increasing the PCI-X burst transfer size, switching from a SMP to a uniprocessor kernel, increasing the TCP window size, and increasing the maximum transfer unit (MTU) size.

In [13], Hurwitz and Feng also evaluated the performance of the Intel 10GbE adapter in MPI and SAN configurations to judge it as a cluster interconnect. They achieved latencies of 13-22 μs using LAM-MPI in a directly connected configuration. Some of their tuning included setting the message size where LAM-MPI switches from short to long-message protocols to 2MB, and also set the kernel TCP window to 16MB. They ran both TCP and MPI-over-TCP tests, and found that in some cases MPI actually outperformed TCP, due to TCP optimizations in LAM-MPI. This lead to their conclusion that optimization in all layers is important for performance.

In [12], Hurwitz and Feng also evaluated the Chelsio T110 10GbE adapter, which is another PCI-X host adapter with a full TOE onboard. With an MTU of 1500 bytes, this adapter showed a throughput of 7.6 Gb/s and a latency of 8.9 μs with TOE enabled, vs 5 Gb/s 10.37 μs without TOE. However, with a 9000-byte MTU the throughput without TOE increased to 7.2 Gb/s, and did not affect the throughput with TOE. The authors concluded that this is because the TOE handles the overhead of segmenting the data into MTU sized chunks, so with TOE enabled the actual MTU size does not greatly affect the host performance.

10GbE is an attractive technology because it provides the ability for geographically disjoint networks to be joined with native Ethernet. Several sites are investigating this possibility. In [14] the authors built a trans-European WAN using native 10GbE as the physical layer (PHY). They successfully tested the compatibility of 10GbE with SONET/SDH transponders, and OC-192 tributary interfaces, sending native Ethernet frames over each. The measured 5.4 Gb/s throughput with TCP, using the Intel 10GbE adapter. Their findings also highlighted the need for more intelligent congestion control algorithms in the presence of multiple streams on long-haul networks.

A distributed supercomputer project in the Netherlands, named DAS-3, now uses Myri-10G cards in a homogeneous WAN consisting of 4 clusters at 4 universities [1, 11]. It is not just a conventional computational grid, but also a platform for researching overlay networks on optical links.

4 Deployment Analysis

Since the cost to deploy and maintain this network is of critical importance, we present an evaluation of the challenges and experiences we encountered during the deployment of the CU-NCAR testbed. We categorize these experiences based on system level features including operating system kernel usage, device driver and network interface configuration, network switching configuration, and security considerations.

4.1 NCAR/CU Collaborative Computing Environment

We deployed a collaborative computing environment between the systems at NCAR and CU. The NCAR Research Systems Evaluation Team (ReSET) provides massively parallel computing infrastructure for use by computational and atmospheric scientists from CU, NCAR, the TeraGrid, and other collaborators. The Computational Science Center (CSC) at CU provides high-throughput computing capabilities on a variety of computer architectures for computational scientists from CU, NCAR, and elsewhere. Since users typically employ resources from both sites and we anticipate the deployment of a similar distributed computing environment between multiple computing sites in the near future, it is natural to evaluate such a deployment using resources provided by the CSC and ReSET.

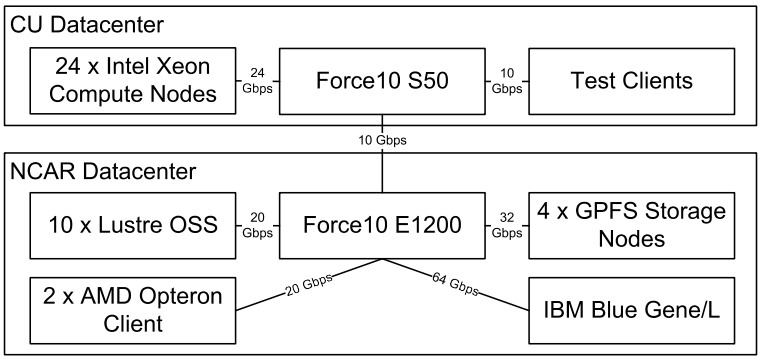


Fig. 1. The NCAR and CU collaborative distributed computing environment

Our initial deployment of this network connects the ReSET and CSC resources through a single 10Gbps link (see Fig. 1). Each site contains a unique set of resources for various computational science activities. The CSC provides a 128-processor Intel Xeon cluster, a 56-processor IBM PowerPC 970 cluster, approximately 10TB of storage, and several large-memory systems suitable for data analysis. The CSC computing infrastructure is ideal for scientists who wish to

develop their applications on a variety of computer architectures, execute embarrassingly parallel applications (non-network intensive) that do not require large numbers of processors, or execute many high-throughput applications in a single experiment. The ReSET computing environment provides resources for the execution of massively parallel applications. This environment includes massively parallel computational systems (e.g., an IBM Blue Gene/L rack) and several hundred terabytes of data storage through an IBM GPFS cluster and a CFS Lustre cluster.

The ReSET computing infrastructure utilizes a Force10 E1200 to connect all of the computing resources. The E1200 is a fully non-blocking switch and our configuration currently contains 96 x 1Gbps and 4 x 10Gbps ports. The CSC computing infrastructure is connected through several switches. This evaluation focuses on the CSC computing resources connected through a Force10 S50. The S50 is a fully non-blocking switch and contains 48 x 1Gbps and 2 x 10Gbps ports.

Most of the legacy computing equipment supported in both the ReSET and CSC computing environments was originally intended to operate a 1Gbps network fabric. These systems use a variety of Broadcom or Intel 1Gbps network interfaces. Several systems can adequately support 10Gbps network interfaces and we have added Myricom Myri-10G Ethernet cards to these systems. Myricom has reported that these network interfaces are capable of near line rate performance in several new systems and are capable of attaining several Gbps in legacy systems. Our tests examine the achievable performance in this environment.

4.2 Local Test Configuration

We performed our evaluation on a subset of the legacy infrastructure deployed in the collaborative environment. The configuration of the systems we used is described in Table 1. The available systems provide a broad array of hosts to evaluate the Myri-10G NICs as well as the various software components we anticipate using. The available hosts use single, dual, or quad core processors. Each host also uses a different motherboard and chipset, which are key factors for the performance of the NICs.

The software configuration for our hosts was nearly homogeneous. All hosts in our evaluation ran the SuSE SLES9 operating system. Each host also used the Linux 2.6.18 kernel. Compared to previous kernel releases we tested, including the stock SuSE kernel (v. 2.6.5) and the 2.6.16 - 17 kernels, this kernel provided the best performance for our hosts and includes an older release of the Myri-10G Ethernet driver. TCP auto-tuning was enabled in these hosts. We used v. 1.1.0 of the Myri-10G Ethernet driver for our Ethernet performance evaluations, v. 1.2.0e of the MX-10G driver, and v. 1.2.1pre2 of the MX-OE driver.

Deployment and configuration of these hosts is straightforward. The open-source Ethernet driver is loaded at runtime as a kernel module. Various network tuning parameters, such as the interrupt coalescing interval and MTU, can be set through common network configuration tools such as ethtool and ifconfig.

Processor	**Motherboard**	**Chipset**	**DMA Speed (Mbit/s)** read/write/bidir	**Write Combining**	**Interrupt Type**
Intel Xeon EM64T (2 x 3.4 GHz)	Supermicro X6DH8	Intel E7520	9600/12840/16392	disbled	MSI
AMD Opteron 270 (4 x 2.0 GHz)	Supermicro H8DCE	nVidia nForce 2200 / 2050	10824/12904/2808	enabled	MSI
AMD Opteron 2212 (2 x 2.2 GHz)	Supermicro H8DMR-82	nVidia MCP55	9176/11360/12448	enabled	MSI
Intel Quad-Core Xeon (8 x 3.6 GHz)	Supermicro X7DB8	Intel 5000P	12784/12904/24824	enabled	MSI
AMD Opteron (2 x 2.6 GHz)	Tyan Thunder K8WE S2895	nVidia nForce 2200 / 2050	11008/10984/19216	enabled	MSI

Table 1. Test hosts used in the evaluation.

The MX-10G driver includes several scripts to load the driver and adjust tuning parameters. The most significant tuning parameter we changed during our evaluation was the interrupt coalescing interval. For the older hosts in our configuration, we increased the interrupt coalescing interval from the default 25μs to 125μs in order to achieve higher throughput. We used the suggested Myricom TCP window sizes of 16MB maximum buffer size for the send and receive buffer, an 87380-byte default receive buffer size, and a 64KB default send buffer size. Table 2 shows the performance through a crossover link between all hosts in our environment.

5 Performance Evaluation

For our evaluation of the performance of the CU-NCAR testbed, we executed a suite of system and application level benchmarks using the systems described in Table 1. As previously mentioned, we are most interested in achieving the highest possible performance from the deployed infrastructure by integrating legacy hardware with the high-speed network. This network will have multiple uses in a production computing environment, including high-speed data transfers, distributed memory application communication, and general network application usage. Therefore, our evaluation considers a wide range of hardware and application level benchmarks to execute in a diverse and often heterogeneous computing environment.

5.1 Local Network Protocol Performance Evaluations

Various network protocols were evaluated for this environment using legacy and specialized systems in point-to-point and switched configurations. We evaluated the two network protocol stacks of interest in our evaluation: TCP and MX-10G. We used Netperf[6] to evaluate the raw performance (throughput and latency) of

the local network infrastructure and protocol stacks. We also used the Intel MPI Benchmarks (IMB) [2] to measure MPI performance for possible application in wide-area environments. The test hosts included Intel Xeon and AMD Opteron servers, described in Table 1.

The protocol of most interest for this evaluation is TCP since most of the software we intend to use in our collaborative environment uses TCP. This software includes MPI applications that utilize MPICH or various data transfer tools such as GridFTP. We also measured UDP performance for comparison's sake[3].

Additionally, we evaluated the performance of the three drivers available for these NICs: the Myricom 10G Ethernet driver (MYRI10GE), the Myricom MX-10G driver (MX-10G), and MX Over Ethernet (MX-OE). The Ethernet emulation mode of the MX-10G driver was used to evaluate the TCP performance of this driver. The MX-OE driver was necessary to run MX over our Ethernet switches. We evaluated the throughput performance and latency of the Myricom 10G NICs with the appropriate drivers using Netperf for both TCP and UDP. Our TCP and UDP throughput evaluations were performed by varying the size of the application buffer to maximize the throughput of Netperf.

Fig. 2(a) illustrates the TCP performance of the NIC and drivers with varying application buffer sizes for the sending host, leaving the TCP window to be auto-adjusted by the Linux kernel[4]. The optimal buffer size for all MTU and driver configurations was found to be between 16KB and 256KB. The throughput for the MYRI10GE driver peaked at 9726.55 Mbit/s with a 64 KB buffer size and 9000-byte MTU, after which the throughput gradually drops. We believe the drop is due to memory copies between kernel and user space. In contrast, the MX driver does not show such a large drop in throughput, though it peaks at 8685.33 Mbit/s. From reviewing the source, the MX driver utilizes zero-copy functions and a default interrupt-coalescing interval of 10μs to achieve lower latency. This means the driver services the data more quickly, but will service less data at once, and spends more time servicing interrupts as well. We also tested the MYRI10GE driver with a lower interrupt coalescing interval, and were not able to get rid of the drop. Therefore, we believe that zero-copy calls in the MX-IP driver are responsible for its smooth throughput curve in the face of increasing application buffer sizes.

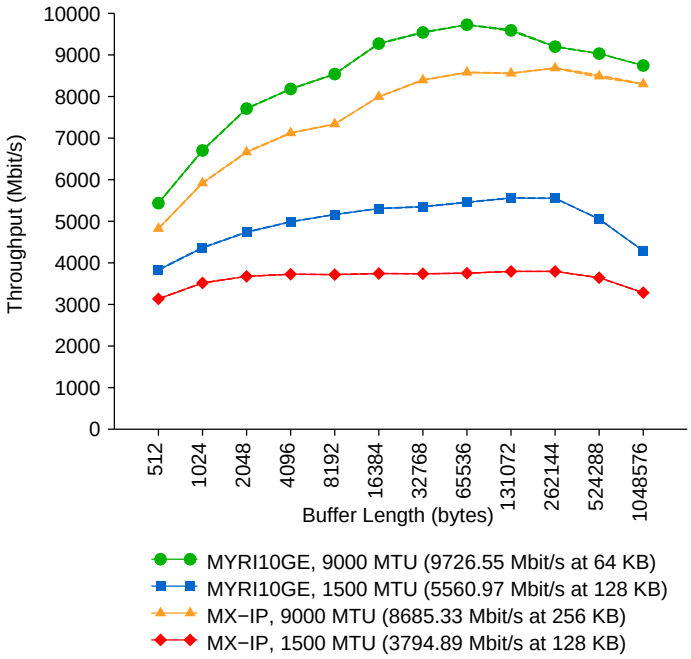
Fig. 3 shows the performance of the two drivers for 9000 MTU size through the S50 switch compared with a subset of the tests through the crossover link.

5.2 MPI Performance Evaluations

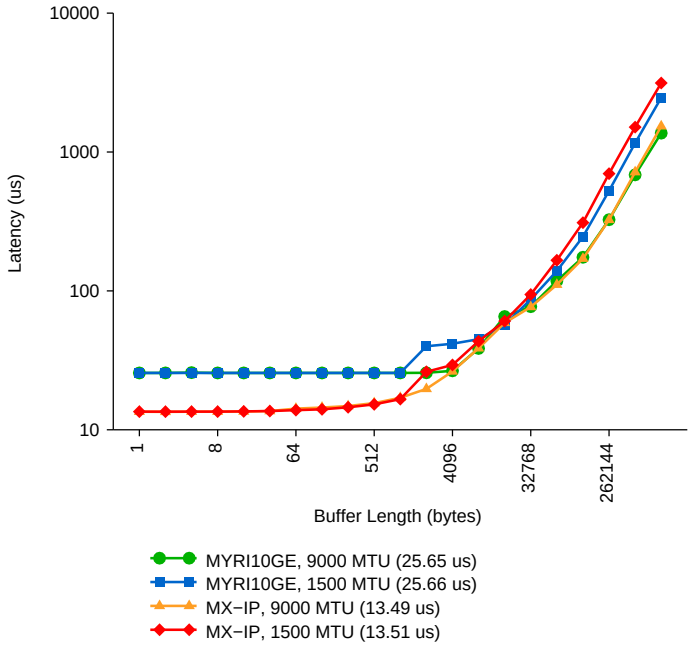
We also evaluated the performance of MPI-based communication using the Myricom 10G NICs and the various communication protocols supported for these NICs. Using the Intel MPI Benchmark suite, we measured the transmission

[3] UDP results are in Appendix A

[4] We also experimented with manually setting the TCP window size and calculating the bandwidth-delay product, which showed that the kernel was properly adjusting the TCP window size.

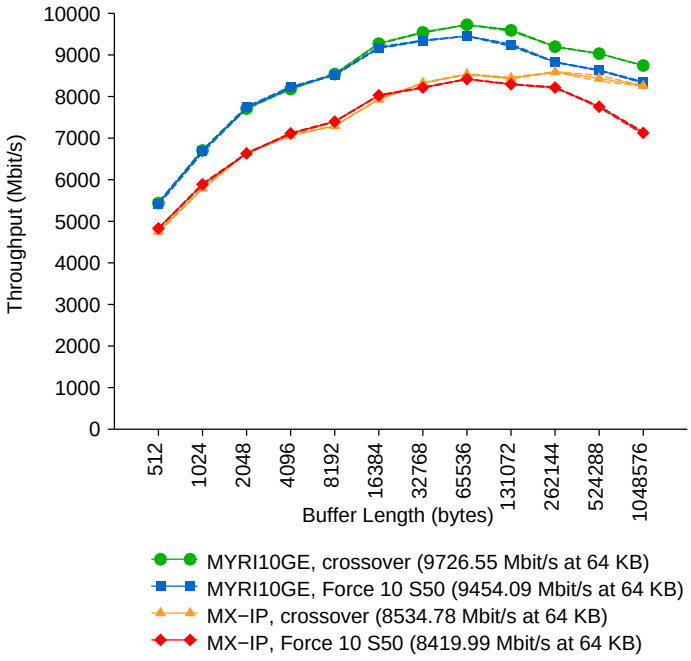


(a)

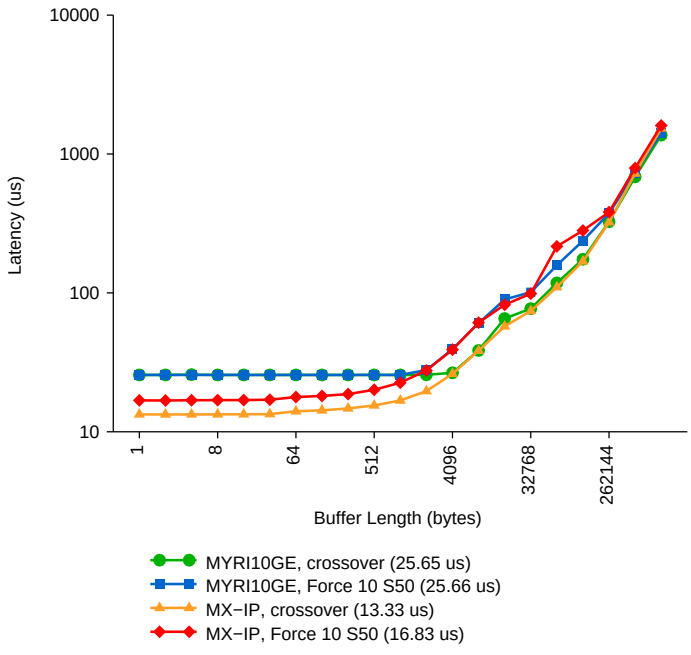


(b)

Fig. 2. Streaming TCP throughput with Netperf's TCP_STREAM test 2(a) and TCP latency with Netperf's TCP_RR test 2(b) for variable application buffer sizes over a crossover link, for Myricom's 10G Ethernet driver (MYRI10GE) and Myricom's MX driver with Ethernet emulation (MX-IP) for 1500 and 9000 MTU sizes. The TCP window size was auto-adjusted by the Linux kernel.



(a)



(b)

Fig. 3. Streaming TCP throughput with Netperf's TCP_STREAM test 3(a) and TCP latency with Netperf's TCP_RR test 3(b) for various application buffer sizes in crossover and switched configurations for Myricom's 10G Ethernet driver (MYRI10GE) and Myricom's MX driver with Ethernet emulation (MX-IP) for 9000 MTU.

	Receiving Host			
Sending Host	Xeon EM64T	Opteron 270	Opteron 2212	Quad-Core Xeon
Intel Xeon EM64T		7027.4	6923.9	6898.1
AMD Opteron 270	6182.3		6294.2	6218.9
AMD Opteron 2212	7830.6	7710.9		8065.6
Intel Quad-Core Xeon	7958.7	8336.3	7879.1	

Table 2. The throughput measured between the hosts in our test environment (in Mbit/s).

latency and throughput for a variety of message sizes for uni-directional communication patterns (using the IMB PingPong test) and bi-directional communication patterns (using the IMB PingPing test[5]). We evaluated these NICs with Myricom's MX communication protocol (MX-10G) and MPICH distribution (MPICH-MX) and a vanilla distribution of MPICH using the Myri-10G Ethernet driver(MYRI10GE+MPICH). All tests were performed over the crossover link, using a 9000-byte MTU (Fig. 4 and 9).

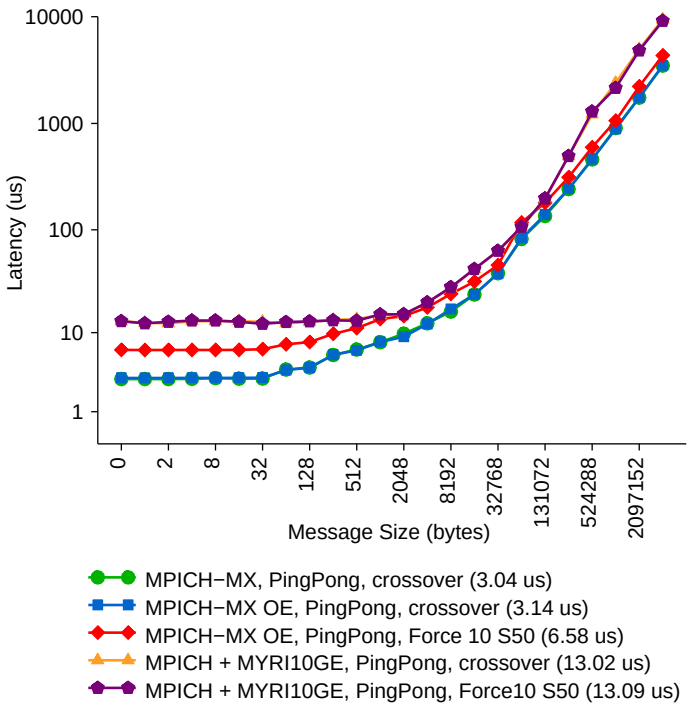
Not surprisingly, we observed the best MPI performance when using the crossover link, where the MX-10G driver and Myricom's MPICH-MX MPI distribution which showed a latency of 3.04 μs and throughput of 9637.04 Mbit/s at the 4MB message size. The MX-over-Ethernet (MX OE/MX-IP) mode showed similar performance results. The S50 switch added about 3.5 μs to the latency, and dropped 1600 Mbit/s from the throughput. The effect of the switch was negligible when using the card in TCP mode (MYRI10GE driver), providing a latency of 13 μs and bandwidth of 5280 Mbit/s. Again, we believe that the MX driver is using zero-copy techniques to provide greater bandwidth at large message sizes. This is discussed more in section 6.

5.3 WAN Network Protocol Performance Evaluations

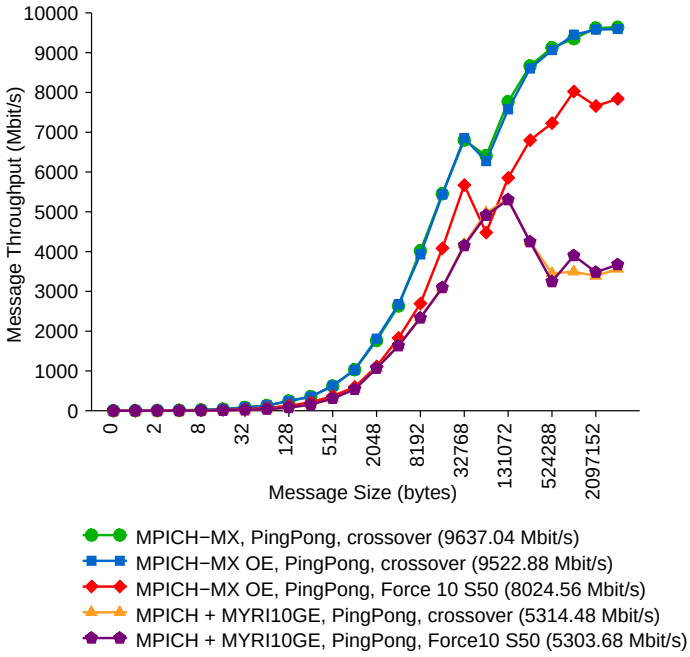
In order to validate the link between the CU and NCAR computing resources, we evaluated the performance of the Myri-10G NICs and various applications between the two sites. During this evaluation, we focused on TCP performance and used Netperf to measure the throughput and latency of the link. At CU, we used the Intel Quad-Core system as the test host, connected to the Force10 S50. The second 10G port on the S50 is connected to the Boulder Research and Administrative Network (BRAN). The NCAR test host is a dual CPU Opteron (model 252) system and is connected to a 10G port on the Force10 E1200 router at NCAR. This router is also connected to BRAN.

In Fig. 5(a), we measured the throughput and latency of the link. The test hosts were capable of sustaining 9820.12 Mbit/s using a 9000 byte MTU. Similar to the local-area tests, we observed a drop in throughput for large application buffer sizes. As with our previous tests, we believe that this drop is related to

[5] Results for the PingPing tests are in Appendix B



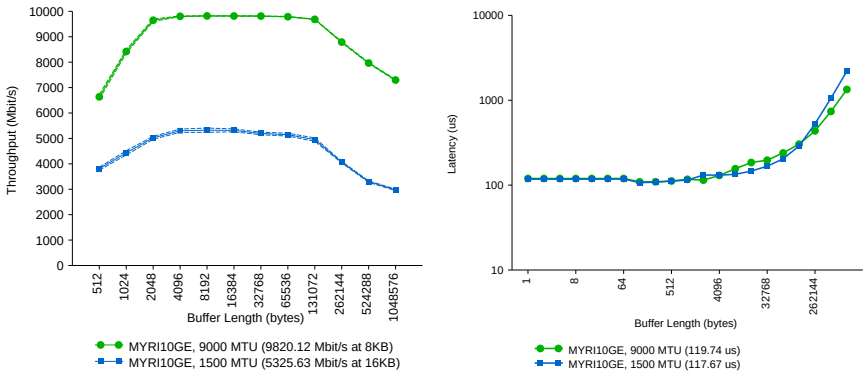
(a)



(b)

Fig. 4. MPI message latency 4(a) and throughput 4(b) measured with the Intel MPI Benchmark PingPong tests for Myricom's MPICH-MX (MPICH-MX) and MPICH using the Myri-10G Ethernet driver (MPICH+MYRI10GE) with a 9000-byte MTU.

the large memory copies Netperf executes to move data from user memory and into a kernel memory buffer. We describe our investigation of this behavior in Section 6. The link latency is over an order of magnitude greater than what we observed in the local-area testing. This behavior was expected due to the increase in distance between the two sites and the additional network infrastructure along the link.



(a) (b)

Fig. 5. Streaming TCP performance for various application buffer sizes over the WAN link between CU and NCAR. This includes the throughput 5(a) and the latency 5(b) of the NICs measured with Netperf's TCP_STREAM and TCP_RR tests for Myricom's 10G Ethernet driver (MYRI10GE) for 9000 MTU.

5.4 Bulk Data Transfer Evaluations

We anticipate the main use of the network will be for the bulk transfer of data between sites through the use of file transfer utilities. An evaluation of common data transfer tools for various data archives was performed to evaluate the network's performance for these operations and to identify the appropriate tools needed to execute these data transfers. Through this evaluation, we identified optimal tuning parameters for various file transfer operations and determined the appropriate tools to use for common use cases.

We evaluated the network infrastructure with the ubiquitous OpenSSH scp and sftp tools, Pittsburgh Supercomputing Center's (PSC) High-Performance Networking patch for the OpenSSH toolset (hpn-scp)[4], Sandia National Laboratory's mpscp[5], and the Globus Toolkit's GridFTP[9]. All of these tools provide different data transfer features and stress different aspects of the network infrastructure. Since the OpenSSH tools are standard features in Linux distributions, evaluation of these tools on the network infrastructure will provide a baseline performance analysis for data transfers using the Myricom NICs on Linux

based hosts. PSC's OpenSSH patch provides buffer tuning options and the ability to disable cryptographic ciphers (the *none* cipher). mpscp provides parallel stream support while using sshd authentication. GridFTP provides parallel data transfer streams and buffer tuning optimizations for Grid-enabled resources.

		File Size (MB)			
Tool	**Setting**	1	10	100	1000
GridFTP	(p = 1)	56.0	550.4	1576.8	1954.4
	(p = 2)	53.6	565.6	2293.0	3368.8
	(p = 4)	52.8	489.6	2304.0	3531.2
mpscp	(w = 1)	91.2	780.8	3472.8	5261.6
	(w = 2)	89.6	807.2	4137.6	6972.0
	(w = 4)	87.36	801.6	4582.0	8656.8
OpenSSH scp	(cipher = arcfour)	80.8	284.0	654.4	699.2
HPN scp	(cipher = arcfour)	10.1	51.3	85.8	728.8
	(cipher = none)	76.8	457.6	917.6	1018.4
OpenSSH sftp	(cipher = arcfour)	80.8	388.0	628.8	668.8
HPN sftp	(cipher = arcfour)	76.8	343.2	636.8	699.2
	(cipher = none)	62.4	367.52	832.8	951.2

Table 3. Results of various file transfer operations (in Mbit/s).

As can be seen from Table 3, the Myricom NICs are effective at transferring data across the CU-NCAR testbed. GridFTP and mpscp provided the best overall performance because the data channels of each server are unencrypted and both tools employ parallel data channels. Using two parallel transfer streams ($p = 4$) provided the best GridFTP performance while mpscp exhibited the best transfer performance with four streams ($w = 4$). The performance of scp did not noticeably improve when patched with PSC's HPN patch with the *arcfour* cipher, but provided a noticeable improvement using the *null* cipher and an unencrypted data stream. Similar to the scp results, sftp performance improved when enabling the *null* cipher.

5.5 Firewall Performance

Our final evaluation concerns scalable, high-performance network monitoring and security tools on 10 Gigabit network infrastructure. We have traditionally used firewalls to secure our network perimeters, limit access to our network hosts based on the traffic source, limit outgoing traffic based on destination, or to limit access to network services running on specific ports. We have used packet filtering tools, such as iptables, to create these firewalls and we noticed in our initial evaluations that they can noticeably limit network throughput. Prior evaluations have also noted the decrease in throughput for iptables when using large rule sets due to its linear rule processing behavior [7]. Stateful firewalls, such as the

connection tracking tools provided by ip_conntrack, offer more complex packet filtering tools that check details of individual network connections. Alternative tools, such as nf-HiPAC, offer scalable and high-performance packet filtering. nf-HiPAC provides a tree-based packet classifier.

We compared the scalability and performance of iptables and nf-HiPAC using the Myri-10G NICs. The Intel Xeon Quad-Core system at CU was used as a firewalled network host and the AMD Opteron 252 system at NCAR was used as a network client. Network traffic was generated by a Netperf client deployed on the AMD host that communicated with a Netperf server on the Intel host. We evaluated the performance of iptables, ip_conntrack, and nf-HiPAC for several rule set sizes.

		Number of Rules						
Tool	**Configuration**	0	100	500	1000	5000	10000	20000
iptables	(stateless)	9323.5	9337.4	9540.0	9244.9	7799.8	4183.1	903.4
	(stateful)	9467.8	9575.2	9576.3	9414.5	7759.9	4274.7	1145.3
nf-HiPAC		9635.6	9526.5	9618.9	9532.7	9582.7	9620.3	9478.0

Table 4. The performance impact that iptables has on throughput for various rule set sizes, measured with Netperf (in Mbit/s).

		Number of Rules						
Tool	**Configuration**	0	100	500	1000	5000	10000	20000
iptables	(stateless)	487.9	496.8	483.0	496.8	560.6	691.6	1745.0
	(stateful)	488.6	482.8	479.9	488.3	544.0	711.4	1462.7
nf-HiPAC		482.0	476.8	470.5	481.6	474.0	467.7	476.8

Table 5. The performance impacts that iptables has on latency for various rule set sizes, measured with Netperf (RTT in μs).

The results from this evaluation are detailed in Tables 4 and 5. Simply loading the packet filtering modules into the kernel reduced the throughput between the hosts by 300 Mbit/s. We also noticed that loading the ip_conntrack module into the kernel further degraded the outgoing throughput of a host more than the incoming throughput by approximately 1 Gbit/s. Both the iptables and nf-HiPAC kernel modules added over 250 μs to the latency just by their loading. Beyond 10,000 rules the iptables tools degraded the throughput to 4.5 Gbit/s or less, with an RTT latency as high as 1745.0 μs. The performance of nf-HiPAC scaled well for the larger rule sets that we evaluated, maintaining a nearly constant throughput of 9.5 Gbit/s and an RTT latency of 460-485 μs.

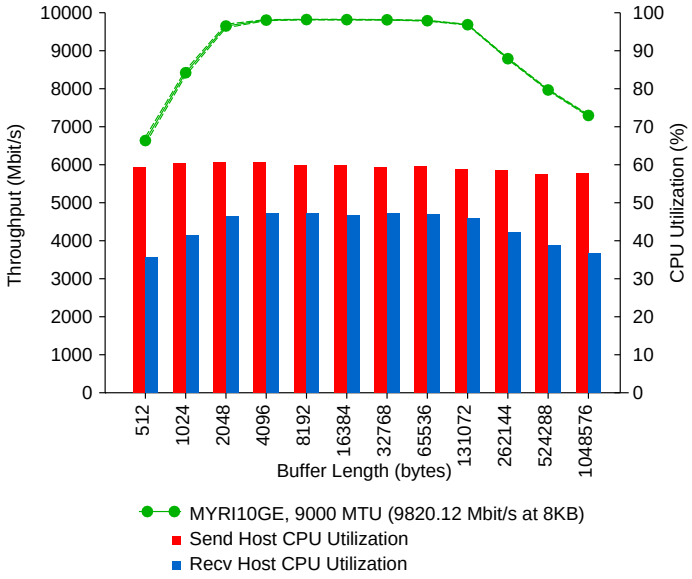
From these results, it is evident that iptables and ip_conntrack do not scale for large rule sets. With small rule sets, we can support packet filtering tools on our hosts without significantly impacting the network throughput. For larger rule sets, more efficient tools such as nf-HiPAC should be considered.

6 Application Tuning

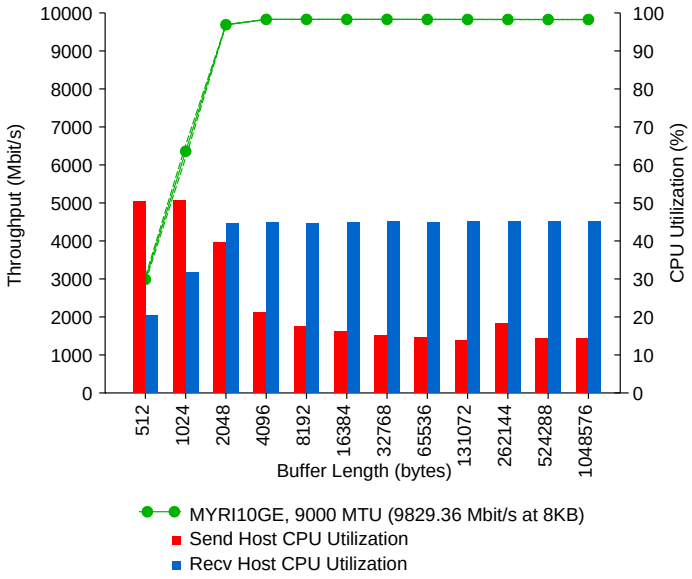
During the network protocol and bulk data transfer evaluations, we found that several of the applications did not meet our performance expectations. The most troubling problem that we identified was the decrease in throughput for large application buffer or block sizes. This behavior affected several of the applications, including Netperf, MPICH, and mpscp. Identifying the causes of this bottleneck and correcting it are of critical importance, so that the NICs can be fully utilized for a broad range of use cases.

We identified that the performance decrease of the applications was related to large application buffer or block sizes and the saturation of the sending client CPU. As the application buffers increase, more CPU resources are required to copy the buffer from the user memory space to the kernel memory space. In 6(a), the sending host is saturating one CPU with the Netperf client and 20% of the other CPU with the Myri10GE interrupt handler. When sending data between the two hosts in the TCP_STREAM test, the Netperf client invokes send() on a buffer in user memory space, the kernel copies the buffer from user space into kernel space, and the Myri10GE driver copies the kernel buffer to the NIC. To reduce the CPU overhead caused by the memory copies on the sending host, a zero-copy operation can be used. Zero-copy reduces the number of memory copies by passing references to the data instead of the buffer. The Linux sendfile() system call implements a zero-copy network operation. The Netperf TCP_SENDFILE test reads a portion of a file into the Linux buffer cache (kernel memory) and the kernel passes references of the buffer to the NIC. This avoids repeated copy operations from user to kernel memory and the extra memory copy between the kernel buffers. In 6(b), the sending host CPU utilization decrease to approximately 20% for application buffers greater than 4KB. As the application buffer size increases, the throughput is sustained and does not decrease for larger buffer sizes like the send() system call.

Since zero-copy operations can reduce the CPU load and sustain throughput, we analyzed the source of several of our network applications. We could not identify the use of sendfile() in any of the bulk data transfer tools we evaluated. Since we could sustain network throughput with the sendfile() system call with Netperf, we evaluated zero-copy operations for one of the applications we evaluated. We modified mpscp to support the sendfile() system call. When transferring files between hosts with mpscp, the sender reads a block from the file (which requires a copy of the file data from kernel to user memory), sends the data to the receiver (which requires a copy of the block from user memory back to kernel memory and a copy of the kernel buffer to the NIC), the remote host reads the data from the socket (which requires a copy of the data from the



(a)



(b)

Fig. 6. Streaming TCP throughput and CPU utilization for variable application buffer sizes with `send()` 6(a) and with `sendfile()` 6(b) measured with Netperf's TCP_STREAM test for Myricom's 10G Ethernet driver (MYRI10GE) for 9000 MTU.

NIC to a kernel buffer and then a copy from the kernel buffer to user memory), and the remote host writes the data to the filesystem (which requires the copy of the data from user memory back to kernel memory). Using sendfile() will reduce the number of memory copies on the sending host. Using sendfile() in mpscp requires that the file be read into the buffer cache and then references to the memory passed to NIC. Fig. 7 illustrates the performance of mpscp using the send() and sendfile() system calls. The sendfile() implementation of mpscp doubles the performance of the send() implementation and sustains 10 Gbps of throughput for large buffer sizes. When using a single thread, the sendfile() performance degrades for buffer sizes larger than 1MB. While zero-copy operations were implemented on the sending host, they were not implemented on the receiver. The decrease in performance is most likely caused by the receiving hosts CPUs saturating on the memory copy operations from kernel to user memory. Other zero-copy operations may be required to sustain the throughput performance for large block sizes. Linux does not currently offer a receive variant of sendfile(), but a version may be created through the use of several zero-copy calls recently added to the Linux kernel (splice(), vmsplice(), and tee()). Other drawbacks to sendfile() include being limited to using a file or memory mapped buffer as input, as well as portability concerns.

7 Conclusions

Our tests show that choosing the right hardware (chipset, processor, etc.) and the right software configuration are crucial to achieving the best possible performance for a given application with the Myri-10G NICs. Table 2 shows the throughput measured between our systems using the Myri-10G NICs. Although these are all fairly new systems, there is still a large variance in performance among them.

The bulk data transfer tests show that the not all applications are able to utilize the full bandwidth available. In fact, most of the stock applications could not be coaxed anywhere near the available bandwidth, showing the need for specialized applications such as mpscp and GridFTP. We found that adding zero-copy support to mpscp increased the throughput of the application in the CU-NCAR testbed by decreasing the load on the sending host. Our traditional host-based network filter, netfilter/iptables, was not able to support high-throughput network traffic for large rule set sizes. An optimized packet filter, nf-HiPAC, was capable of sustaing high-throughput and low latency data streams with an interface consistent with netfilter.

Overall the Myri-10G NICs along with the Force10 switches and routers provide the necessary infrastructure to facilitate regional computational science in a WAN environment. While improvements to software infrastructure were necessary to support intensive network operations, the addition of zero-copy through the Linux sendfile() system call provided a noticeable improvement without significant modification to the software. Other zero-copy improvements may be required to sustain throughput for the receiving host.

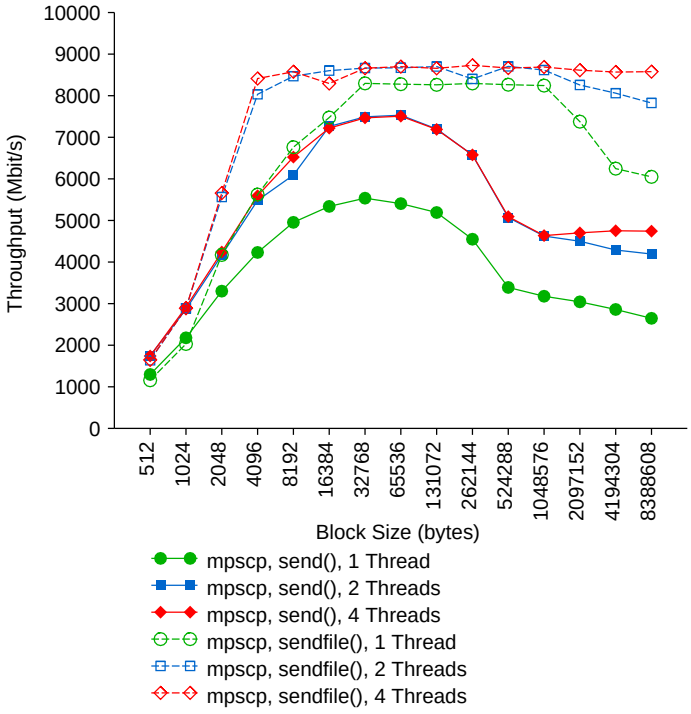


Fig. 7. mpscp performance when using the send() and sendfile() Linux system call. mpscp scp was evaluated for multiple threads and block sizes.

8 Acknowledgments

University of Colorado computer time was provided by equipment purchased under DOE SciDAC Grant #DE-FG02-04ER63870, NSF ARI Grant #CDA-9601817, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program. We would like to thank Myricom and Force10 Networks for loaning equipment and providing technical support for this research. We would also like to thank the individuals who provided to us invaluable assistance during this project: Paul Hyder of the National Oceanic and Atmospheric Administration, Mike Forbes, David Woods, and Vince Biondollio from the University of Colorado Information Technology Services, and David Mitchell from the National Center for Atmospheric Research Network Engineering & Telecommunications Section.

References

1. Distributed ASCI supercomputer 3 (das-3). http://www.cs.vu.nl/das3/.
2. Intel mpi benchmarks. http://www.intel.com/software/products/cluster /clustertoolkit/features.htm#mpi_benchmarks.
3. Myri-10g 10-gigabit ethernet solutions. http://www.myri.com/Myri-10G/10gbe_solutions.html.
4. High performance ssh/scp - hpn-ssh. http://www.psc.edunetworkingprojectshpn-ssh/, 2007.
5. Mpscp high-performance data transfer utility. http://www.sandia.govMPSCPmpscp_design.htm, 2007.
6. Netperf: A network performance benchmark. http://netperf.org/netperf, 2007.
7. nf-hipac: High performance firewall for linux netfilter performance tests. http://www.hipac.orgperformance_testsresults.html, 2007.
8. Top500 supercomputer sites. http://www.top500.org, 2007.
9. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in highperformance computational grid environments, 2001.
10. W. chun Feng, J. G. Hurwitz, H. Newman, S. Ravot, R. L. Cottrell, O. Martin, F. Coccetti, C. Jin, X. D. Wei, and S. Low. Optimizing 10-gigabit ethernet for networks of workstations, clusters, and grids: A case study. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 50, Washington, DC, USA, 2003. IEEE Computer Society.
11. H. E. B. et al. The distributed ASCI supercomputer project. *Operating Systems Review*, 34(4):76–96, 2000.
12. W. Feng, P. Balaji, C. Baron, L. N. Bhuyan, and D. K. Panda. Performance characterization of a 10-gigabit ethernet toe. *hoti*, 0:58–63, 2005.
13. J. Hurwitz and W.-C. Feng. Analyzing mpi performance over 10-gigabit ethernet. *J. Parallel Distrib. Comput.*, 65(10):1253–1260, 2005.
14. C. Meirosu, P. Golonka, A. Hirstius, S. Stancu, B. Dobinson, E. Radius, A. Antony, F. Dijkstra, J. Blom, and C. de Laat. Native 10 gigabit ethernet experiments over long distances. *Future Gener. Comput. Syst.*, 21(4):457–468, 2005.

15. T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages I:11–14, Oconomowoc, WI, 1995.

Appendix

A UDP Protocol Results

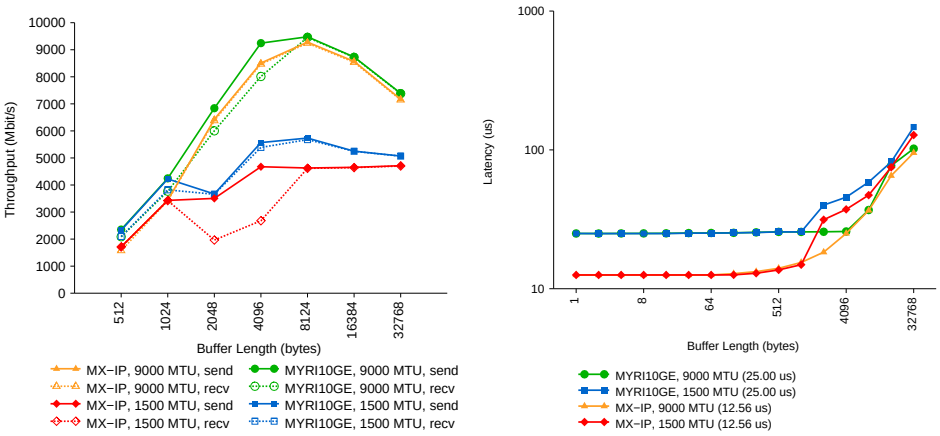


Fig. 8. Streaming UDP performance for various application buffer sizes. This includes the throughput for both the receiving and sending hosts 8(a) and the latency of the NICs 8(b) measured with Netperf UDP_STREAM and UDP_RR for Myricom's 10G Ethernet driver (MYRI10GE) and Myricom's MX driver with Ethernet emulation (MX-10G) for 1500 and 9000 MTU.

B MPI PingPing Results

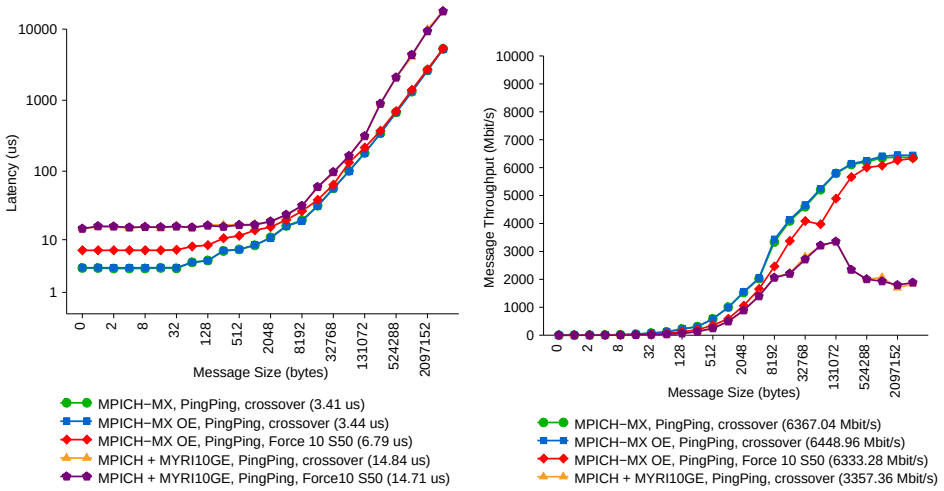


Fig. 9. MPI message latency 9(a) and throughput 9(b) measured with the Intel MPI Benchmark PingPing tests for Myricom's MPICH-MX (MPICH-MX) and MPICH using the Myri-10G Ethernet driver (MPICH+MYRI10GE) with a 9000-byte MTU.