EXPERIMENTAL ANALYSIS OF COMMUNICATIONS INTERFACES FOR HIGH-PERFORMANCE CLUSTERS

Alan George, Robert Todd, William Phipps
High-performance Computing and Simulation (HCS) Research Laboratory
Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida 32611-6200
E-mail: {george,todd,phipps}@hcs.ufl.edu

Keywords – computer networks, network interfaces, gigabit networks, cluster computing

ABSTRACT

This paper provides experimental data and analysis to quantify both the peak and sustained performance characteristics of four of the most promising new networking technologies for interconnecting machines in a high-performance cluster of workstations. These technologies include Scalable Coherent Interface (SCI), Myrinet, Fibre Channel System (FCS), and Asynchronous Transfer Mode (ATM).

Cluster networking performance experiments are conducted and measurements presented in terms of effective throughput and two-way latency using several different software-based communications protocol interfaces, including TCP/IP, UIUC Fast Messages, UCB Generic Active Messages, and a new multi-layer communications interface called SCALE Suite.

Finally, parallel and distributed processing experiments are conducted with these network architectures connecting UltraSPARC workstations, results are summarized and the basic scalability of the architectures is analyzed. Both parallel linear algebra and sorting programs are among the benchmarks, with a degree of parallelism that for some experiments reaches as high as several dozen.

1.0 INTRODUCTION

With the constantly increasing performance and decreasing costs associated with uniprocessor and multiprocessor workstations and personal computers, high-performance cluster (i.e. *hypercluster*) architectures and systems hold the potential to be a formidable platform for a variety of grand-challenge applications in computer simulation. However, in order for high-performance clusters to achieve their potential with parallel and distributed applications, substantial performance bottlenecks must be overcome in two critical areas. First, new technologies in the realm of gigabit computer networks must be effectively leveraged to support the cluster environment in a manner that supports lightweight, low-latency, and high-throughput communications. Secondly, new software protocol interfaces must be developed, and their use accepted and adopted, in order that the potential performance of the communications layer can be realized from the user application. Without significant advancements in cluster-communication architectures, interfaces, and system and user software, high-performance clusters will continue to be relegated to the limited realm of coarse-grained, embarrassingly-parallel algorithms and applications that marks their history.

Recently a number of relatively new *de facto* and *de jure* standards in local-area and system-area networks have been adopted and implemented by interconnect and networking vendors in the form of switches, links, and I/O-bus adapter interfaces for workstations and personal computers. These networking standards consist of well-known and lesser-known protocols, such as message-passing protocols including FCS, Myrinet, ATM, High Performance Parallel Interface (HIPPI), and just recently Gigabit Ethernet. In addition, distributed-shared-memory architectures are also now emerging and being leveraged to support workstation clustering, including byte-addressable networks such as Scalable Coherent Interface. While many of these networks support data rates in the Gb/s range, the ability to move application payloads at rates even remotely approaching these peaks, with latencies in the low microsecond range supporting finer grains of parallelism, is extremely challenging and far from readily attainable. The Scalable Cluster Architecture Latency-hiding Environment (SCALE) project at the *HCS Research Lab* attempts to capitalize on gigabit networks by utilizing a hybrid network approach.

In this paper, four sets of experiments are performed and their results presented for comparison. These experiments provide insight into some of the most promising high-performance networks and the software solutions that will drive the advances of tomorrow in interfacing and network technologies. Conclusions drawn from the experimental results are then presented along with possible avenues of future research.

2.0 EXPERIMENTS

To analyze the peak communication performance attainable for applications interacting in a cluster of workstations, a set of experiments has been constructed. The *Raw Performance Experiments* focus on the maximum

© 1998, HCS Research Lab. All Rights Reserved.

available performance with little overhead and a minimal protocol stack. The *TCP/IP Performance Experiments*, *Lightweight Protocol Experiments*, and *SCALE Experiments* test the suitability of several protocol stacks and the effect of the added overhead on overall performance as compared to the raw peak.

As results will indicate, since peak performance of the memory and I/O buses is typically not achieved in network transactions, the network adapter is often the hardware bottleneck in a cluster of workstations connected by a gigabit network system. One or more of four modes of transfer are employed in the architecture of network adapters for the interface between the workstation and the network. Figure 1 illustrates these transfer modes which include:

- *Direct-mapped* – The main processor pushes the data into the queued network output interface via a segment of virtually-mapped physical memory.
- *Large-buffer Mapped* – The main processor pushes the data into the network output buffer via a segment of mapped memory. Then, the network card's message processor transfers the message into the queued network output interface.
- *Small-buffer DMA* – The main processor pushes the data into kernel memory where the network card's message processor pulls the data via a DMA transaction into the queued network output interface for transmission.
- *Large-buffer DMA* – This mode follows the same path as the small-buffer DMA transactions with the exception that the network output buffer is large enough to contain a contiguous user message. A DMA engine is used to copy the message from the kernel buffer to the network output buffer. The message is then processed by the network processor and transmitted onto the network.

One approach to maximize performance is to eliminate needless copying of messages as they are moved from the user's application to the network. The direct-mapped style of data transfer allows the data to be moved directly to the network without any copying. By allowing a network and/or protocol stack to bypass some of these steps or possibly use idle cycles on the main processor to perform some of the copying, then overhead can be effectively reduced. The choice of data transfer mode affects the usability and performance of a network adapter.

The computers used to perform all of the network experiments consist of 200-MHz UltraSPARC-2 (U2) workstations. For the parallel processing experiments, as many as eight U2 workstations are used, in both single-CPU and dual-CPU SMP configuration, as well as up to eight 167-MHz UltraSPARC-1 (U1) workstations. All machines operate under Solaris 2.5.1.

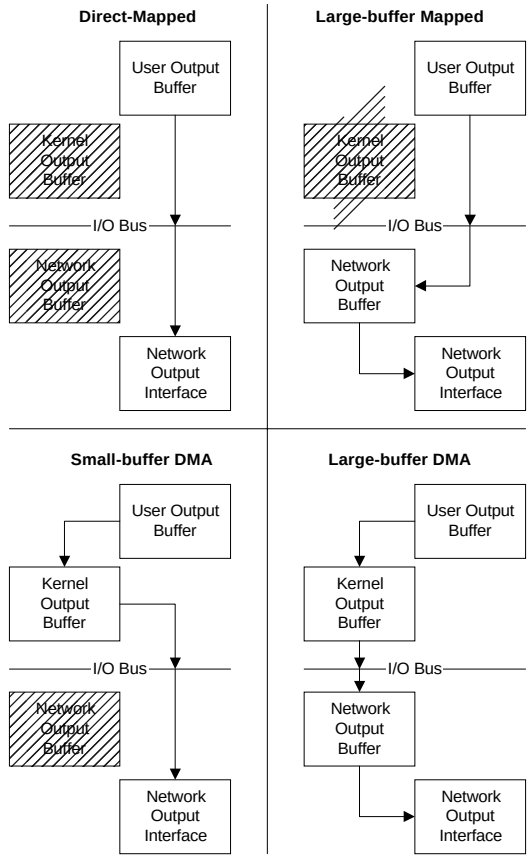

Figure 1. Four modes of transfer

This diagram shows four different modes of transfer for network adapter interfacing: direct-mapped transfers, large-buffer mapped transfers, small-buffer DMA transfers, and large-buffer DMA transfers.

The network testbeds used in these experiments include 1.6-Gbps/link SCI from Dolphin Interconnect Solutions, 1.28-Gbps/link Myrinet from Myricom, 1.0-Gbps/link FCS from Ancor Communications, and 155-Mbps/link ATM from FORE Systems, in switched-ring, star, ring, and star topologies respectively. The network interface cards and switches used are provided in Table 1.

Table 1. Network Adapter and Switch Specifications

Network	NIC	Switch
Myrinet	M2F-Sbus32	M2F-SW8 8-way switch
SCI	SCI/Sbus-2B	SW4-1 4-way switch
ATM	SBA-200E	ASX-200BX
FCS	FCS-1062	none used

© 1998, HCS Research Lab. All Rights Reserved.

Raw Performance Experiments

The raw performance experiments quantify the best-case performance of a network interface adapter without the interference and penalties of a software protocol. Figures 2 and 3 show the limitations of the memory bus and I/O bus of the U2 workstations. These measurements are the hard limitations that cannot be surpassed without a change in the architecture of the workstation.

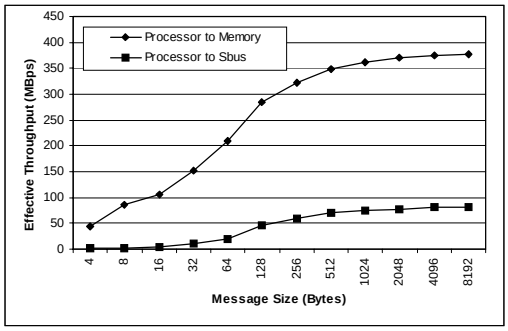

Figure 2. Processor to Memory and Processor to SBus One-Way (Read or Write) Throughput

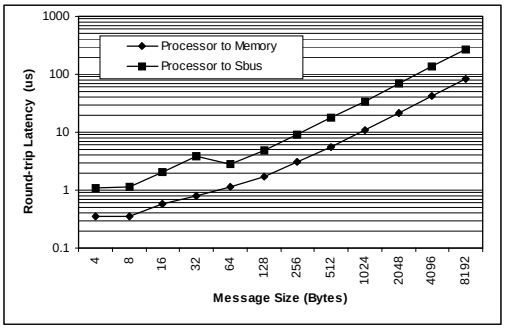

Figure 3. Processor to Memory and Processor to SBus Two-Way (Read then Write) Latency

The memory bus performance was measured by copying data from a user application into a DRAM bank on the memory bus. Memory bus bandwidth reaches performance levels of 376 MBps of one-way streaming throughput and round-trip latencies as low as 0.35 µs. Similarly, the SBus performance was measured by copying data from a user application into an SRAM bank on an SBus card. The SBus reaches a sustained one-way streaming throughput of about 83 MBps with round-trip latencies as low as 1 µs. Of course, the SBus does not possess the ability to cache transactions automatically, and hence will not keep pace with main memory for general memory access.

The suite of raw performance numbers shown in Figures 4 and 5 differ in the mode of transfer that they use due to the design of the adapter card. When using the SCI adapter, the SCI DMA uses the small-buffer DMA mode, while SCI shared-memory (SHM) uses the direct-mapped mode of transfer. The Myrinet cards have a Myrinet Control Program (MCP) that allows different modes of operation on the card, and thus the Myrinet Mapped tests use a large-buffer mapped mode of operation while the Myrinet DMA tests use a large-buffer DMA transfer. The FCS adapters have small network output buffers, and thus the transfers employ the small-buffer DMA mode of transfer. The raw drivers for Fibre Channel currently offer no better performance than the TCP/IP drivers, and are therefore excluded from these charts.

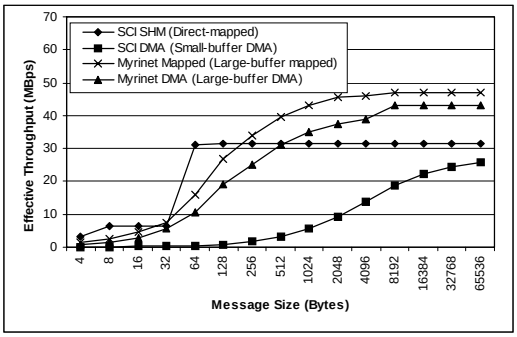

Figure 4. Raw Network One-Way (Send or Receive) Throughput for SCI and Myrinet

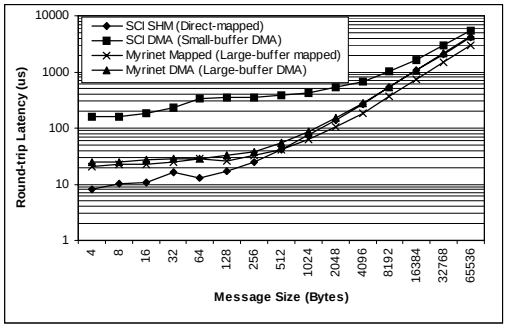

Figure 5. Raw Network Two-Way (Send then Receive) Latency for SCI and Myrinet

As can be seen from Figure 5, the direct-mapped network (SCI SHM) achieved the lowest round-trip latency of approximately 7.9 µs. The large-buffer mapped network (Myrinet mapped) was in second with a round-trip latency of 21 µs, and the large-buffer mapped Myrinet was able to achieve 27 µs. However, for streaming throughput, as shown in Figure 4, the large-buffer mapped mode of transfer performs best peaking at 48 MBps while the SCI SHM network peaks at 31 MBps. The reason for this behavior is because the main processor only allows a limited number of outstanding memory transactions which must be completed before the next message can be sent, keeping the SCI SHM network in an unpipelined mode. The large-buffer mapped mode allows the local buffer to be loaded at the speed of the I/O bus and allows the next message to be loaded as the current message is sent.

© 1998, HCS Research Lab. All Rights Reserved.

TCP/IP Performance Experiments

TCP/IP drivers typically perform all of the buffering between the user output buffer and the network output interface, given that an adapter supports all of the buffers shown. The general implementations of TCP/IP, even on high-performance networks, are a safe and simple implementation using the most DMA-oriented system of buffering possible to achieve process protection and low CPU utilization.

The following TCP/IP performance experiments were conducted using the *netperf* benchmarking utility (HP 1995). The socket size on both sender and receiver ports was set to 65536 bytes.

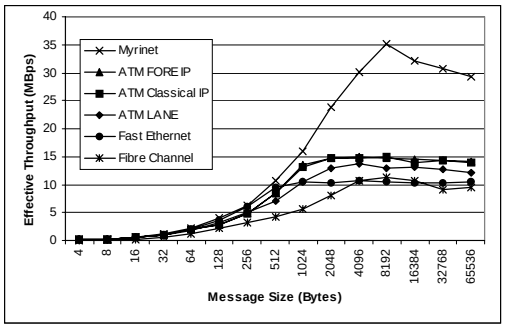

Figure 6. One-Way Effective Throughput with TCP/IP

The Myrinet cards, because of their large-buffer DMA mode of transfer under TCP/IP, achieve the best one-way streaming throughput performance of all the networks tested, attaining a maximum of 35 MBps. However, because TCP/IP implementations are inherently burdened by software bottlenecks, the latencies remain fairly consistent across all network platforms and hover between 0.5 and 1 ms for small message sizes. This latency is much too high to support medium- to fine-grain parallel and distributed algorithms.

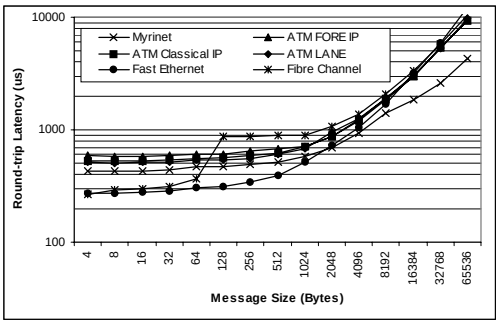

Figure 7. Round-trip Latency with TCP/IP

Lightweight Protocol Experiments

To take advantage of low-latency networks, the software communication protocol used on the network needs to be as lightweight as possible to reduce the amount of end processing. With currently available software, a slow thread library is matched with a large communication overhead so that neither seems to be relatively slow. Since the average latency of a system determines how well the performance will scale as more workstations are added, higher-latency software reduces the efficiency of the system and limits its scalability.

The communication protocol used for transporting data between workstations can severely affect performance. Traditional communication protocols, such as the TCP/IP stack, were built to supply a generic, reliable service over relatively slow and error-prone networks. Modern high-speed networks and dedicated workstation interconnects do not require the same amount of error protection and generic service.

With the desire to provide a parallel, message-driven environment using a lightweight transport, the *HCS Research Lab* chose to implement a protocol compliant with UCB's Generic Active Messages (GAM) specification (Mainwaring 1995) to maintain compatibility with other software. The HCS implementation, called SCALE Messages, provides the GAM function calls operating over generalized high-performance transport channels called SCALE Channels.

Because SCALE Channel concepts are designed to be universal and adaptable, implementations are and will be provided for a wide range of high-performance workstations and PCs, network interface adapters, and operating systems. Current emphases include UltraSPARC workstations operating under Solaris and Pentium-based workstations operating under Windows NT, Solaris x86, or Linux. All SCALE Channels offer reliable data transport with three run-time functions: read, write, and poll. These functions are non-blocking and are intended to be used with a high-speed thread package, also developed at the *HCS Research Lab*, called SCALE Threads (George et al. 1997), but can be used separately if needed.

An active message can be thought of as a lightweight remote procedure call. The SCALE Messages approach extends this concept to include remote thread generation and execution rather than a simple function call. The SCALE Messages implementation is modeled after the GAM 2.0 specification with extensions to support network independence and multithreaded execution. The University of Illinois at Urbana-Champaign has also implemented a messaging type interface similar to active messages called Fast Messages (FM) (Pakin et al. 1995). The main difference between the two specifications is that FM relies on pipelined communication and restricted handlers. Consequently, the FM implementation on Myrinet, while satisfying a different set of requirements, lags on this particular set of experiments.

SCALE Messages has been completed for SCI and is near completion for Myrinet. Figures 8 and 9 show the performance of SCALE Messages over SCI SHM where the

© 1998, HCS Research Lab. All Rights Reserved.

overhead of the SCALE Suite is only marginal in the larger message sizes. From a raw round-trip latency of just under 8 µs, the SCALE Messages layer provides a formatted and active message stream with a round-trip latency of only 23 µs.

To directly compare the performance of GAM, FM, and SCALE, the Myrinet network was selected (i.e. neither GAM nor FM have been ported and optimized for SCI). Using the same workstations and network testbeds, GAM and FM performance was measured. Although the Myrinet implementation of SCALE Channels is still preliminary, conservative estimates of its performance can be attained by taking the raw performance of Myrinet and adding to it the overhead of SCALE Channels and Messages measured over SCI. Indeed, such an estimate is conservative since further optimizations for pipelined communication with SCALE over Myrinet will likely lead to even better performance. Figures 10 and 11 compare the messaging performance of GAM, FM, and SCALE over Myrinet as well as the raw Myrinet peak.

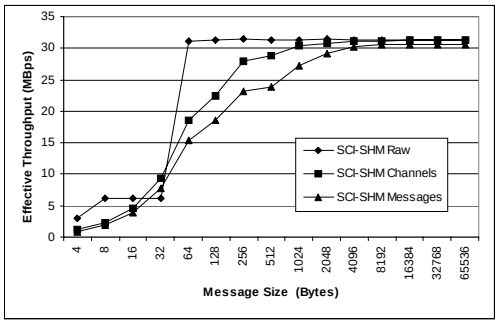

Figure 8. SCALE Messages, SCALE Channels and Raw Throughput over SCI

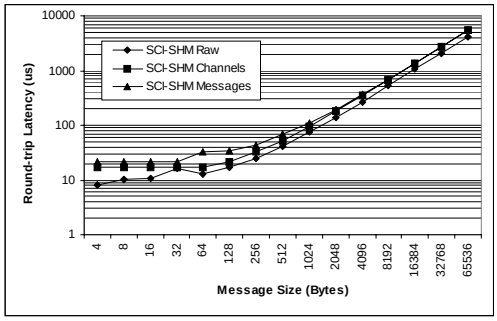

Figure 9. SCALE Messages, SCALE Channels, and Raw Round-trip Latencies over SCI

Although the SCALE Messages performance is based on an unoptimized version of SCALE Channels over Myrinet, the performance is near optimal with only a slight degradation over the raw performance. SCALE Channels and consequently SCALE Messages is able to gain a performance advantage over the GAM and FM protocols by two design techniques: user-level network control, and optimized block memory access. By exposing the network adapter's SRAM to the user as in the large-buffer mapped mode of transfer, an extraneous and costly copy through the kernel buffer is eliminated as well as the need for a DMA engine.

Traditionally, only DMA engines could issue 64-byte transactions which use the I/O bus in an efficient "burst" mode. With the introduction of the Visual Instruction Set (VIS) into the UltraSPARC line of processors, user applications can now issue 64-byte, uncached memory transactions. All main memory operations in the experiments take advantage of the optimal block move. By using the special block memory accesses on the mapped network adapter SRAM, SCALE Channels over Myrinet can sustain up to 48 MBps throughput between user applications. Both FM and GAM use the DMA engine for large transactions, which peaks at 36 MBps, and pay a performance penalty for the extra copy.

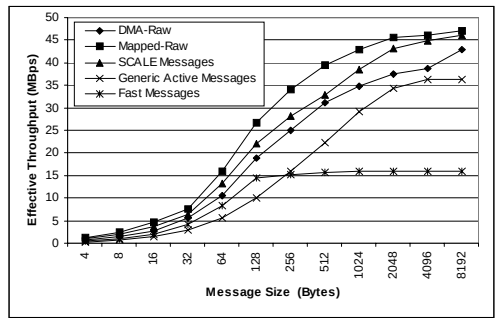

Figure 10. Throughput Comparison of Messaging Approaches over Myrinet

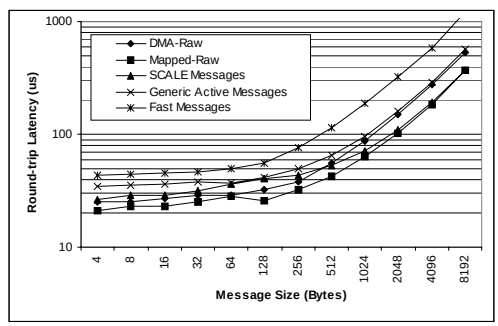

Figure 11. Round-trip Latency Comparison of Messaging Approaches over Myrinet

Parallel Processing Experiments

The two parallel algorithms used to analyze the performance of SCALE over a high-speed network (in this case SCI) are matrix multiplication and in-core sorting. The matrix multiplication simply consists of the classic $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ and is cache optimal such that the $\boldsymbol{B}$ matrix has already been transposed and minimizes cache misses. The in-core sort consists of a 100,000-element sort with 64-byte, 128-byte, and 256-byte elements. The data is initially distributed

© 1998, HCS Research Lab. All Rights Reserved.

among all nodes equally and the final sorted set remains distributed. Each node is responsible for a contiguous range of numbers in the overall sort. When a node initializes and reads the initial set of numbers it performs a "bucket" sort to distribute the correct numbers to the nodes that are responsible for that range. Once all numbers have been distributed to the correct nodes then a simple *qsort* is performed. This algorithm is similar to the Datamation sorting benchmark without the disk I/O (Anon et al. 1985). Dual-processor workstations were not used in the in-core sorting experiments since this benchmark was designed to show communication efficiency under random traffic while the dual-processor configuration only helps with computational demands.

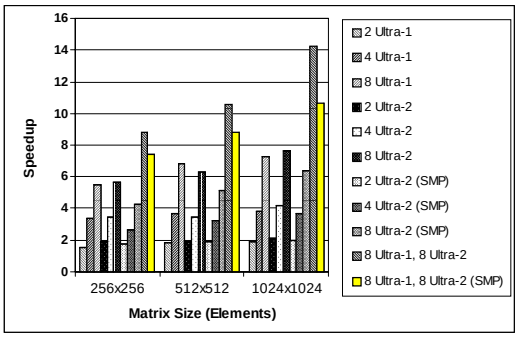

Figure 12. Matrix Multiplication Speedup

Figure 12 shows the results of the matrix multiplication experiment. The best speedup achieved was 14 using eight U1 workstations and eight single-CPU U2 workstations. For the larger matrix sizes near-linear speedup was achieved up to 16 processors. The 24-processor case (8 dual-CPU U2s and 8 U1s) exhibited slowdown compared to the 16-processor case because of the lack of optimal opportunistic load balancing across the heterogeneous network.

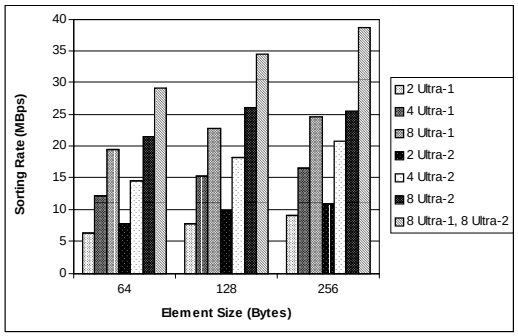

Figure 13. In-core Sorting Rate

Figure 13 shows the sorting rates of the in-core parallel sort algorithm for elements of different sizes. For a fixed data set, as the size of the cluster increases so does the sorting rate since each workstation has to sort proportionally less, although the network demand grows. For a fixed cluster size, as the size of the data set increases (by increasing the element size) so too does the sorting rate, although the percentage of total execution time attributed to *qsort* increases.

3.0 CONCLUSIONS

In this paper, four experiments have been performed to both analyze lightweight communication protocols for high-performance networks and show possible solutions. The memory and I/O bus bandwidths were illustrated along with the results of tests with raw interfaces to high-performance networks. The raw performance results are used to determine the upper-bound of performance from lightweight communication packages. The limits of the TCP/IP protocol stack were indicated on all networks under test. The GAM, FM, and SCALE lightweight communication protocols were then compared, with SCALE showing both a portable design methodology and superior performance. Parallel processing results were then introduced using the SCALE methodology including a matrix multiplication algorithm and an in-core sorting algorithm which achieved near-linear speedups in many cases. Future research plans include the optimization of SCALE Channels for other high-performance networks, workstations, and adapters.

The future of high-performance interconnects hinges on the performance of lightweight, yet robust communication protocols that take advantage of the hardware benefits while hiding the latencies that the interconnect incurs. We believe the SCALE Suite takes a step towards solving the software bottleneck by leveraging the performance benefits and limitations of high-performance interconnects and their interfaces.

REFERENCES

Anon et al. 1985. "A Measure of Transaction Processing Power." Datamation. V.31, no. 7: 112-118.

George, A.; Phipps, W., Todd, R., and Rosen, W. 1997. "Multithreading and Lightweight Communication Protocol Enhancements for SCI-based SCALE Systems," *Proceedings of the 7th International Workshop on SCI* (Santa Clara, CA. Mar. 24-27). 5-16.

Hewlett-Packard Information Networks Division, "Netperf: A Network Performance Benchmark, Version 2.0", February 15, 1995.

Mainwaring, A.M. 1995. "Active Message Applications Programming Interface and Communication Subsystem Organization," *Draft Technical Report*. Computer Science Division, University of California, Berkeley.

Pakin, S., Lauria, M., and Chien, A. 1995. "High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet", *Proceedings of Supercomputing '95* (San Diego, CA. Dec. 3-8). IEEE Computer Society Press, San Diego, CA, 1528-1557.