The SCALE Lightweight, Multithreaded, and Heterogeneous Network Interfacing Suite

Alan George, William Phipps, Robert Todd, and Matthew Chidester
High-performance Computing and Simulation Research Laboratory
Department of Electrical and Computer Engineering, University of Florida

Whereas the potential of high-performance networks promises to provide gigabit-per-second throughput to the desktop with sub-microsecond latencies between networked applications, limits in internal computer software and hardware make this promise far from reality. Using a modular scheme, the components of the SCALE software suite exploit network interfaces and processor architectures to their fullest, providing a network-independent abstraction to applications while at the same time emphasizing a lightweight software interface that supports the most promising high-speed networks including SCI, ATM, Gigabit Ethernet, Myrinet, Fibre Channel, and HiPPI.

The world of computers is one of fast-paced change and rapid technological advancement. As transistor widths shrink from micrometers to nanometers, processors are rapidly reaching clock speeds in the gigahertz range instead of megahertz, and memories are approaching densities measurable in gigabits rather than megabits. Similarly, high-speed networks with bit rates measured in Gb/s instead of Mb/s and latencies in microseconds instead of milliseconds are rapidly moving from the research lab to the real world.

With the advent of such low-latency, high-throughput networks, optimization of the network interface — both the software and hardware components — becomes important in order to realize the full potential of the network. Since a data packet may pass through many layers of software and hardware to reach the network fabric, all layers of the interface must be examined and optimized. In the age of 10 Mb/s Ethernet, the software delays were small compared to the hardware transmission time. However, as network speeds enter the Gb/s range, the overhead of system software and hardware becomes the dominant factor.

In this article, we present an approach to identifying and addressing these design challenges for high-performance network interfaces — the Scalable Cluster Architecture Latency-hiding Environment (SCALE). The SCALE approach is compared with other methods for supporting new network interfaces in terms of functionality and performance, and experimental system results are provided. The SCALE Software Suite is comprised of three major components: SCALE Messages, SCALE Threads, and SCALE

* The high-speed networking experimental testbed consists of symmetric multiprocessor workstations with dual 200 MHz UltraSPARC processors, a 32-bit SBus I/O bus operating at 25 MHz, 256 MB of main memory, 1 MB of L2 cache, running the Solaris 2.5.1 operating system, and interconnected to one another by a wide variety of high-speed networks.
Channels. Each component attacks one or more network-interface issues at the lowest level, exploiting hardware architecture where possible to achieve the highest performance.

**HIGH-PERFORMANCE MESSAGING LAYERS**

A messaging layer is a software subsystem that attempts to hide the details of network communication with semantics that are more natural to an application programmer. Message-passing conventions (e.g. send/receive) are common in current network applications. The vast majority of applications use Sockets, a message-passing interface which uses TCP/IP as the communication protocol.

**Sockets and TCP/IP**

Since TCP/IP is so widely used across the Internet, it seems reasonable to reuse the Sockets-TCP/IP interface for high-speed local networks to maintain software compatibility. This approach has serious drawbacks, most notably the inherent performance penalties. For instance, the TCP/IP error- and flow-control services, which provide error-free communications, in some cases duplicate services provided by high-end networks. Also, TCP/IP is handled by the operating system, which adds delays associated with context switching and copying data into and out of the kernel. In addition, message passing may not be the best communication paradigm for many applications since unexpected messages are difficult to handle.

Because of the performance limitations of Sockets-TCP/IP and the functionality limitations of message passing, researchers have developed new messaging protocols based on message handling. Message handling refers to a technique where each message includes information on how to process it at the receiving workstation. In addition to providing message-handling protocols, the next generation of messaging software adds support for lightweight communication protocols and robust networks.

**Fast Messages**

Developed by the University of Illinois at Urbana-Champaign (UIUC), Fast Messages (FM) supplies error-free communications without using the TCP/IP protocol or the operating system kernel and supports Myricom’s Myrinet, a 1.28 Gb/s switch-based network. An FM transaction begins by opening a communication channel between two workstations. Data is then pushed into the channel in arbitrary-sized
segments. The novel approach of FM is the ability to append more data to an outgoing message. At the receiving side, the workstation strips out the header information, which includes an index to the function to execute with the incoming data. FM is able to pipeline incoming messages by not requiring that the entire message arrives before executing the handler function.

FM offers a messaging layer that does not require operating system accesses and that directly controls the network adapter for better performance. FM strictly limits the execution time of message handlers and the types of functions they can use, and thus its message-handling functionality is best suited for small transactions, such as parallel process coordination. Additional limitations include the lack of support for multiple network protocols and interfaces.

**Generic Active Messages**

The University of California at Berkeley (UCB) created a message-handling library called Generic Active Messages (GAM), also developed for Myricom’s Myrinet adapters. GAM extends the message-handling paradigm by formalizing the programming interface and the structure of the communication subsystem beneath. In GAM, endpoints are responsible for sending and handling messages. Each message has information indicating which handler function to execute at a certain endpoint. Handler functions can accept up to eight integer arguments and also a bulk data transfer. Bulk data transfers are used to pass large amounts of information from one workstation to another with high throughput.

Sending messages in GAM is similar to remote procedure calls, but the handlers have limitations. For a receiving endpoint to detect and handle a new message, the process must call the GAM poll function. Handlers cannot block and must execute quickly since new messages are detected and handled by the thread that executes the poll command. Long-running handlers would block the polling thread. Also, access to the endpoint is serialized, allowing only one message handler to execute at a time.

GAM provides a flexible programming environment, but it is limited due to the lack of multithreaded and concurrent execution capability, pipelined message handling, and support for other network protocols and interfaces.
SCALE Suite

Because of the limitations of other messaging layers and the need for an optimal, system-wide approach to high-performance communications, the High-performance Computing and Simulation (HCS) Research Laboratory at the University of Florida developed the SCALE Suite. The SCALE Suite integrates a lightweight messaging layer, a fast thread library, and an abstracted network channel interface into a cohesive, multi-layer, portable library that provides maximum performance and flexibility to the user application and allows access to multiple high-performance networks.

SCALE Messages (SM) is an implementation of the GAM specification with extensions to support unrestricted handlers, concurrent network access, and most importantly, multithreading. An SM is essentially a lightweight remote procedure call (RPC) with bulk data transportation facilities. Each message in SM contains the index of a remote function and the function’s arguments. The receiver uses the index to find the address of the message-handling function. Executing the SM on the remote machine becomes no more difficult than a local procedure call.

Unlike GAM and FM, SM is not targeted to a single network protocol or interface and supports multiple types of networks simultaneously. To allow concurrent adapter servicing, a thread is created to monitor each high-performance network in the system, checking for incoming SM’s to handle. As soon as a new message arrives, a thread is created to execute the handler and the monitoring thread returns to its function. Since a thread is dedicated to each network adapter and a thread is created for each incoming message, communications are truly concurrent.

High-performance messaging layers like SM bring application-to-application, round-trip communication latencies down into the 20 µs region. However, for modern high-speed microprocessors, many thousands of instructions can be executed in this amount of time. To achieve latency hiding and higher efficiency, it is desirable to switch to another thread of execution while awaiting completion of a network transaction. Unfortunately, the delay for a thread switch on a typical high-speed workstation is approximately 20 µs. Since the thread switch penalty is on the order of the network transaction latency, multithreading using traditional thread packages yields no benefit.

Thread-creation time presents another issue for high-performance network interfaces. If a new thread is created to service each incoming message to exploit concurrent processing of transactions, the thread
creation time becomes a critical part of the overall latency to service each message. Unfortunately, the thread creation delay is typically over 200 µs, which is an order of magnitude larger than the end-to-end latencies on a high-speed network.

Since multithreading is essential to exploit parallelism and hide network delays, the SCALE Suite includes a fast user-level thread library called SCALE Threads (ST). Thread-creation times for ST have been reduced to under 2 µs while thread switching occurs in only 2.5 µs, thereby making the concurrency characteristics of multithreading advantageous.

At the lowest layer of the SCALE Suite lies SCALE Channels (SC). A SCALE Channel provides a standard interface to a high-performance network adapter or low-level software driver. Common operations such as connect, disconnect, send, receive, and poll are provided. SC performs any error recovery or flow control necessary to guarantee data integrity. An application can interface directly to SC when maximum raw throughput or minimum latency is desired. Alternately, applications can take advantage of SM which uses SC. SCALE Messages can be multiplexed onto an arbitrary number of channels, allowing a single program to utilize all available network resources.

SC is designed to support computers connected to more than one network at a time and provides for concurrent communication across multiple networks using several channels. Such a heterogeneous environment may support a high-speed SAN connection such as IEEE 1596-1992 Scalable Coherent Interface (SCI) or Myrinet for cluster-computing operations, a fast Fibre Channel interface for disk I/O, and an ATM or Gigabit Ethernet interface for Intranet or Internet connectivity. Multiple connections of the same type may be used to form more elaborate network topologies (e.g. a mesh, tree, or torus) to provide a higher aggregate bandwidth and lower round-trip latency, or allow for fault-tolerance in the presence of failures or network congestion.

Currently, SC implementations exist for a number of high-performance networks and interfaces, including SCI/SBus-1 (1.0 Gb/s) and SCI/SBus-2B (1.6 Gb/s) adapters from Dolphin Interconnect Solutions, Myrinet/SBus (1.28 Gb/s) adapters from Myricom, UNIX Sockets, and local shared-memory segments for symmetric multiprocessors (SMP). SC implementations for SCI/PCI (1.6 and 4.0 Gb/s), Myrinet/PCI (1.28 Gb/s), Gigabit Ethernet/PCI (1.0 Gb/s) adapters from Packet Engines, OC-3c ATM/PCI
and ATM/SBus (155 Mb/s) adapters from FORE Systems, HiPPI/PCI (1.6-Gbps) adapters from GigaLabs, and Fibre Channel (1.0 Gbps) are in development.

Figure 1 shows the combination of SCALE Messages, SCALE Threads, and SCALE Channels with which a computer system is able to concurrently communicate across multiple networks with a single unified, lightweight interface. Network abstraction is provided to the higher layers, while the underlying subsystem is designed exclusively to fully leverage each network protocol and adapter interface.

![Figure 1. The SCALE Suite](image)

This figure depicts the relationship between SCALE Messages, SCALE Threads, and SCALE Channels. SCALE Threads allow a channel to spawn a thread for each incoming message, or multiple threads to operate concurrently on one or more channels. Multithreading allows the messaging interface to use multiple channels concurrently with different threading or communication techniques used for each channel, depending on the underlying network.

### CLASSES OF NETWORK ADAPTER ARCHITECTURES

To capitalize on the architecture of network adapters, the operation and structure of adapters must be examined. In support of the SCALE project, a classification system for network adapters has been developed based on the path data traverses to move from a user-level application to the network fabric. Network adapters can support one or more of four modes of data transfer to communicate between the workstation and the network. Figure 2 illustrates these transfer modes:

- **Small-buffer Mapped (SBM)** – The main processor maps a piece of memory on the remote system. The network adapter contains only a small queued-output interface necessary to package the data for transmission on the network medium.
• **Large-buffer Mapped (LBM)** – The main processor maps a segment of memory physically located on the network adapter and pushes the data into the network output buffer. The network adapter’s microcontroller transfers the message into the queued network output interface.

• **Small-buffer DMA (SBD)** – The main processor pushes the data into kernel memory where the network adapter’s microcontroller pulls the data via a DMA transaction into the queued network output interface for transmission.

• **Large-buffer DMA (LBD)** – This mode follows the same path as 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 packaged by the message processor and transmitted onto the network.

![Diagram](image.png)

**Figure 2. Four modes of transfer**

This diagram shows the four different modes of transfer for network adapter interfacing: small-buffer mapped transfers, large-buffer mapped transfers, small-buffer DMA transfers, and large-buffer DMA transfers.
SBM adapters can achieve the lowest latency because there are no intermediate copies required in the data transfer operation. The sender simply maps a piece of remote memory and writes to it using a `memcpy()` or store operation. At the receiving end, the network adapter writes the data directly into user memory. The drawback to this approach is that the receiver is responsible for detecting an incoming message, which requires some form of notification system such as opportunistic polling. Opportunistic polling is a synchronous polling technique that uses multithreading to sustain high processor utilization while waiting for a network event.\(^4\)

Another drawback to the SBM mode occurs when the network is congested or the adapter’s output buffer becomes full. The processor may stall until the copy can be completed. LBM alleviates this problem by allowing part or all of a message to be buffered in the adapter until it can be transmitted. The CPU can burst data until the adapter buffer is full, then complete other work while the data is output onto the network.

Both mapped transfer modes require the CPU to write data through the I/O bus. The CPU is blocked during this relatively slow operation. The DMA transfer modes can reduce the load on the CPU by allowing the network adapter’s DMA engine to perform the data transfer over the I/O bus. This method requires that the data be located in a kernel buffer. The context switch and extra copy operations required to move the data into kernel space adversely affect the latency of DMA transfers. Once the CPU has performed the memory-to-memory copy and set up the transfer, the latency can be hidden by allowing the CPU to work on other tasks while the adapter performs the rest of the data transfer.

To achieve optimum performance from any given network interface, the SCALE Channels layer takes advantage of each of these transfer modes depending on the architecture employed on the network adapter.

**PUTTING IT ALL TOGETHER**

Hardware components can no longer be viewed as opaque “black boxes” if maximum efficiency is to be achieved. Instead, detailed knowledge of all components in the system must be taken into account so that each component can be fully utilized. The SCALE Suite attempts to capitalize on the unique abilities of each component and combine these optimizations to produce a system-wide optimal solution while providing a unified view of heterogeneous network interfaces. The current implementation of the SCALE
Suite for SPARC-based workstations includes optimizations in using the processor, memory bus, I/O bus, and network adapters.

**Processor architecture optimizations**

Although the SCALE Suite will support other processors, optimizations for the UltraSPARC processor illustrate that attention paid to processor architecture can be worthwhile. The UltraSPARC processor has three unique features that facilitate high-performance communications and computations. First, the processor is able to issue atomic memory transactions that can be used to synchronize threads running on different processors. Mutual exclusion locks, spin locks, and semaphores can be implemented at the user level without operating-system intervention. Second, the UltraSPARC uses multiple sets of registers to speed nested function calls by eliminating excess register copying to the stack. This optimization makes context switching slower since all of the register sets must be flushed to perform a context or thread switch. However, by coding the register flushing procedure as a user-level procedure, the speed of the entire switching process can be improved. The SCALE Threads library takes advantage of both of these optimizations to accelerate thread management and synchronization operations by one or two orders of magnitude over traditional software.

Third, the UltraSPARC processor offers a unique memory optimization that allows it to load or store a 64-byte block of memory in one operation without caching the transaction. For bulk data movement to and from communication devices, this non-cached operation can leave the cache intact after the data movement. If the data is already in the cache, the 64-byte access will use the cached copy. Using block moves improves the effective throughput of memory operations as well as I/O operations since most I/O buses support 64-byte burst transactions.

The performance for moving a large block of memory using the 64-byte memory transaction is much higher than traditional load and store operations which transfer only 8 bytes at a time. The 64-byte block move yields over 376 MB/s from main memory. In contrast, 8-byte accesses can sustain only 208 MB/s from main memory. Using the block move over the UltraSPARC’s SBus, 84MB/s can be streamed, as compared with 8-byte transactions peaking at 39 MB/s and typical DMA engines at 55 MB/s.
Network adapter architecture optimizations

The SCI/SBus-1 and SCI/SBus-2B adapters from Dolphin Interconnect Solutions provide two modes of operation: SBD and SBM. The latter mode is most interesting since the SCI adapters are among the first interfaces to offer such a mode. The SCI adapters provide a distributed shared memory by allowing user processes to map portions of the global SCI address space into their virtual address spaces. If two processes on different workstations map the same global SCI address, they can communicate using simple processor loads and stores instead of message-passing semantics.

The SCI/SBus-2B adapter has been designed to accept the UltraSPARC’s block move and is able to transmit a 64-byte shared memory transaction in a single SCI packet. Thus, no actual message processing takes place; rather a simple command and address translation is required to complete a transaction. Once the shared memory region is mapped, the operating system is no longer involved in any communication. By using block moves, the SCI adapter is able to achieve a 6.3 \( \mu \)s one-way latency for 64-byte blocks from one user application to another. For an 8-byte transaction which transfers only one-eighth as much data, the one-way latency is 3.9 \( \mu \)s. One-way sustained bandwidth through the adapter using block moves is 31 MB/s, while two-way sustained bandwidth saturates at 44 MB/s.

Myricom’s Myrinet/SBus adapter is unique in encouraging the user to reprogram the on-board microcontroller for individual applications. The microcontroller is a LANai 4.1 processor with 256 kB of SRAM and is programmable using standard C source code. The host workstation can communicate with the Myrinet using either the LBM or LBD modes. Nearly all sites that have Myrinet use the LBD mode of adapter interfacing. By contrast, the SCALE solution combines the UltraSPARC’s block move command with the LBM mode of interfacing to produce a more efficient communication layer than the DMA-based versions. By directly mapping the adapter’s SRAM into the user process, a copy operation to the kernel buffer is eliminated, lowering latency. In addition, the UltraSPARC is able to sustain more write bandwidth to the Myrinet adapter than the adapter’s DMA engine can read from main memory, which makes the LBM approach perform better for both throughput and latency.

While SCALE Channels is offered for many high-performance networks, GAM and FM are targeted for Myrinet and both use the LBD mode of the Myrinet adapter. The TCP/IP implementation for Myrinet also uses the LBD mode. The format of a message in GAM and FM is highly dependent on the network
adapter used, and this limitation makes multi-protocol implementations difficult. UCB has recently developed an updated version of GAM that, like SCALE, also includes support for local shared-memory segments.\textsuperscript{10}

**EXPERIMENTAL SYSTEM RESULTS**

Three basic metrics are often used when measuring the efficiency of a messaging or network subsystem: streaming throughput, half-bandwidth message size, and round-trip latency. These measurements are taken at the user application level by a benchmark program and represent the level of performance an application can expect from the communication system. Streaming throughput is measured as the amount of data per second that a sending workstation can reliably transfer to a receiving workstation. The half-bandwidth size is the message size where the streaming bandwidth is half of the peak bandwidth, and are rounded to powers of two for these experiments. Round-trip (RT) latency is measured as the time taken for one application to send a small message, another workstation to receive and retransmit the message, and for the initial sender to receive the message back.

Table 1 provides adapter characteristics and performance measurements for several key network protocols and interfaces. To reduce the number of independent variables in the experiments, a common I/O-bus platform, SBus, was used for all tests, and the latest versions of GAM, FM, and SCALE for SBus-based interfaces were used.\textsuperscript{*}

\textsuperscript{*} FM and SCALE are also targeted for PCI-based interfaces to leverage faster I/O bus clock speeds and wider datapaths. However, since no common denominator exists for FM, GAM, and SCALE in terms of processors, operating systems, and drivers, direct comparisons with PCI interfaces are impractical at this time.
<table>
<thead>
<tr>
<th>Adapter</th>
<th>Bus Interface</th>
<th>Peak Data Rate (MB/s)</th>
<th>Software Protocol</th>
<th>Transfer Mode</th>
<th>Streaming Throughput (MB/s)</th>
<th>½ BW Msg. Size (Bytes)</th>
<th>Min. RT Latency (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Bus</td>
<td>SBus</td>
<td>100</td>
<td>Raw</td>
<td>LBM</td>
<td>84.0</td>
<td>32</td>
<td>1.1</td>
</tr>
<tr>
<td>Dolphin SCI (SCI/SBus-2B)</td>
<td>SBus</td>
<td>200</td>
<td>Raw</td>
<td>SBM</td>
<td>31.4</td>
<td>32</td>
<td>7.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SC</td>
<td>SBM</td>
<td>31.3</td>
<td>64</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SM</td>
<td>SBM</td>
<td>30.6</td>
<td>64</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Raw</td>
<td>SBD</td>
<td>25.7</td>
<td>2048</td>
<td>140</td>
</tr>
<tr>
<td>Myricom Myrinet (M2F-Sbus32)</td>
<td>SBus</td>
<td>160</td>
<td>Raw</td>
<td>LBM</td>
<td>47.1</td>
<td>64</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SC</td>
<td>LBM</td>
<td>46.9</td>
<td>128</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SM</td>
<td>LBM</td>
<td>46.8</td>
<td>256</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Raw</td>
<td>LBD</td>
<td>43.0</td>
<td>128</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GAM</td>
<td>LBD</td>
<td>36.2</td>
<td>256</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FM</td>
<td>LBD</td>
<td>16.1</td>
<td>64</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TCP/IP</td>
<td>LBD</td>
<td>35.1</td>
<td>1024</td>
<td>430</td>
</tr>
<tr>
<td>Ancor Fibre Channel (FCS-1062)</td>
<td>SBus</td>
<td>125</td>
<td>TCP/IP</td>
<td>SBD</td>
<td>11.3</td>
<td>1024</td>
<td>270</td>
</tr>
<tr>
<td>FORE ATM (SBA-200E)</td>
<td>SBus</td>
<td>19.4</td>
<td>TCP/IP</td>
<td>SBD</td>
<td>15.0</td>
<td>512</td>
<td>580</td>
</tr>
<tr>
<td>Sun Microsystems Fast Ethernet (on-board)</td>
<td>SBus</td>
<td>12.5</td>
<td>TCP/IP</td>
<td>SBD</td>
<td>10.6</td>
<td>256</td>
<td>270</td>
</tr>
</tbody>
</table>

As shown in Table 1, the Myrinet protocols that use LBM mode (i.e. SC and SM) achieve fundamentally higher throughput than those that use LBD mode (i.e. GAM, FM, and TCP/IP). Because both GAM and TCP/IP use the same mode of transfer, they attain comparable levels of streaming throughput. While GAM has slightly better latency characteristics than SM, SM has much more functionality for only a small performance penalty. Furthermore, the SCALE Suite permits an application to use both SC and SM simultaneously, providing the application with the option of using SM for more functionality, SC for higher performance, or any combination in support of multiple networks. For instance, applications may elect to invoke SM calls to communicate with other machines in a cluster for parallel computing, and at the same time invoke SC calls to communicate to a high-speed file server.

While the LBM transfer modes with Myrinet provide the highest throughput in these experiments, the SCI adapters are able to achieve remarkably low latency performance using SBM. The SBM-based SCALE Messages for SCI achieved the lowest round-trip latency of any messaging layer on a network of workstations. All of the DMA-based performance numbers suffer from lower throughput, although the
FORE ATM adapters and the Sun Microsystems Fast Ethernet adapters do attain much of the available bandwidth associated with their 19.4 and 12.5 MB/s data rates, respectively.

In order to harness the potential of high-performance networks, a number of key design challenges must be overcome. Communication efficiency can be made optimal by examining all of the layers involved in the movement of user data from one workstation to another. Performance is improved by developing and exploiting new techniques or enhancing existing methods to minimize the bottleneck imposed by the software and hardware layers. The SCALE Suite is a set of software layers that attempts to hide the network interface internals by providing standardized software interfaces optimized for multiple high-performance adapters. SCALE also recognizes the need for hybrid network support and allows multiple heterogeneous SCALE Channels and Messages to be active concurrently in a multithreaded fashion.

The flexibility in the design of the SCALE Suite is intended to more readily exploit new hardware features as they become available. For instance, most high-performance network adapters will soon be designed to support the PCI I/O bus with 64-bit data paths at 66 MHz. Contrary to popular opinion, experience suggests that the I/O bus alone is by no means the primary contributor to current network adapter bottlenecks. However, new optimizations for messaging, multithreading, and channels interfacing in a high-speed network environment should make the peak bandwidth of PCI more readily attainable. Preliminary testbed experiments with SCI and Gigabit Ethernet adapters based on the PCI bus indicate that substantially higher throughputs and lower latencies may soon be realized when coupled with SCALE.

When latencies approaching the nanosecond range are required to support parallel processing based on fine-grain algorithms, the traditional approach of interfacing high-speed networks to the I/O bus of the computer may no longer be satisfactory. The next step is likely to be an interface directly between the processor and the network by means of a CPU/Memory bus adapter. Of course, some of the difficulties here lie in the proprietary nature of many existing CPU/Memory buses. However, we anticipate the future development of some form of Advanced Network Port standard – somewhat comparable to the AGP standard becoming popular for video interfacing in personal computers.
As the networks and network interfaces accelerate in data rate and efficiency, the overall system performance will become even more dependent upon software layers. SCALE addresses these issues by providing a lightweight, robust, portable, and easily adaptable interface to high-performance networks of today and beyond.

Acknowledgements

This work was partially supported by contracts from the Office of Naval Research, the National Security Agency, and the Naval Air Warfare Center. We are also grateful for testbed equipment provided by Dolphin Interconnect Solutions, Packet Engines, and GigaLabs.

References


