MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

ChaosLAN: Design and Implementation of a Gigabit LAN Using Chaotic Routing

Neil R. McKenzie,	MERL
Kevin Bolding,	Seattle Pacific University
Carl Ebeling,	University of Washington
Lawrence Snyder,	University of Washington

TR-97-08 July 1997

Abstract

In recent years, the Chaos Project at the University of Washington has analyzed and simulated a dozen routing algorithms. Three new routing algorithms have been invented; of these, the chaotic routing algorithm (a.k.a. Chaos) has been the most successful. Although the Chaos router was developed for multicomputer routing, the project has recently directed its attention towards the application of Chaos technology to LAN switching. The present task is to implement a gigabit LAN called ChaosLAN, based on a centralized switch (hub) and high speed serial links to workstations. The switch itself is a fully-populated two-dimensional torus network of Chaos routers. The host adapter is Digital's PCI Pamette card. To evaluate the performance of ChaosLAN, we are supporting the Global Memory System (GMS), a type of distributed virtual memory also developed at UW. We also describe an application involving real-time haptic rendering used in a surgical simulator.

This report has also been published in the Proceedings of the 1997 Parallel Computer Routing and Communication Workshop (PCRCW'97), Atlanta GA, June 26-27, 1997.

1. First printing, July 8, 1997

1 Introduction

The case for a high-performance local area network for the construction of a parallel computer using a network of workstations (NOW) is well understood [1]. One goal is to provide a scalable platform for parallel programs using low-cost workstations or PCs, comparable in performance to that of a multiprocessor system but at far lower cost. Another goal is to provide a high-performance distributed computing environment that is typical in robotics, medical simulators and virtual reality systems, where real-time performance is critical to achieving a high quality experience. Standard LAN technology such as Ethernet, initially developed for less demanding applications such as e-mail or file transfer, does not provide the performance necessary for these kinds of distributed real-time applications. Faster versions of Ethernet provide higher network throughput but do not significantly improve the underlying latency, which is on the order of a millisecond. Low latency communication, on the order of a few tens of microseconds or less, is critical for providing both a high-performance cluster computing environment for parallel execution of scientific codes and for real-time response in distributed systems running complex applications.

Since the early 1990's the Chaos Project at the University of Washington [2] has studied network communications and routing algorithms, in the context of multicomputer routing. The group has analyzed and simulated dozens of routing algorithms and also studied approaches to fault tolerance and constructing network interfaces. Three new routing algorithms have been invented: chaotic, zenith, and triplex. Of these, the chaotic routing algorithm [3] (a.k.a. Chaos) has been the most successful. The Chaos router has been extensively simulated on several topologies under a variety of work loads. Generally, it delivers higher throughput than its competitors with similarly low latency. In fact, the Chaos router delivers better than 95% of the theoretical maximum bandwidth for random traffic on a 256-node torus. The results for Chaos are consistent across traffic patterns and on other topologies using a variety of performance metrics (e.g. throughput, latency and network saturation level). Additionally, the Chaos router is logically simple in structure, so that it can be readily implemented in silicon inexpensively.

Chaos is characterized as an *adaptive, non-minimal* algorithm for fixed-length (or bounded-length) packets. Long messages are segmented into packets at the sender and re-assembled at the receiver. Adaptivity increases the number of paths between sender and receiver, so that more of the network can be active at once and help distribute the work load evenly compared with an *oblivious* algorithm in which there is only one possible path from sender to receiver. Adaptivity also complicates the re-assembly of packets at the receiver, because packets can overtake one another due to internal congestion, and cause packets to arrive in a different order than they were injected into the network. Hardware assistance for re-assembly of non-ordered packets has been studied extensively [4] and an inexpensive, streamlined host adapter to interface with an adaptive network has been proposed by McKenzie [5].

Although the Chaos router was developed for multicomputer routing, the project has recently directed its attention towards applying Chaos technology to LAN switching. This move is similar to Myricom's use of the Caltech Mosaic router in a gigabit LAN [6]. The Mosaic router, based on the oblivious dimension-order routing algorithm, was also developed initially for multicomputers. Since the underlying performance potential of Chaos is greater than that of Mosaic for similar cost, there

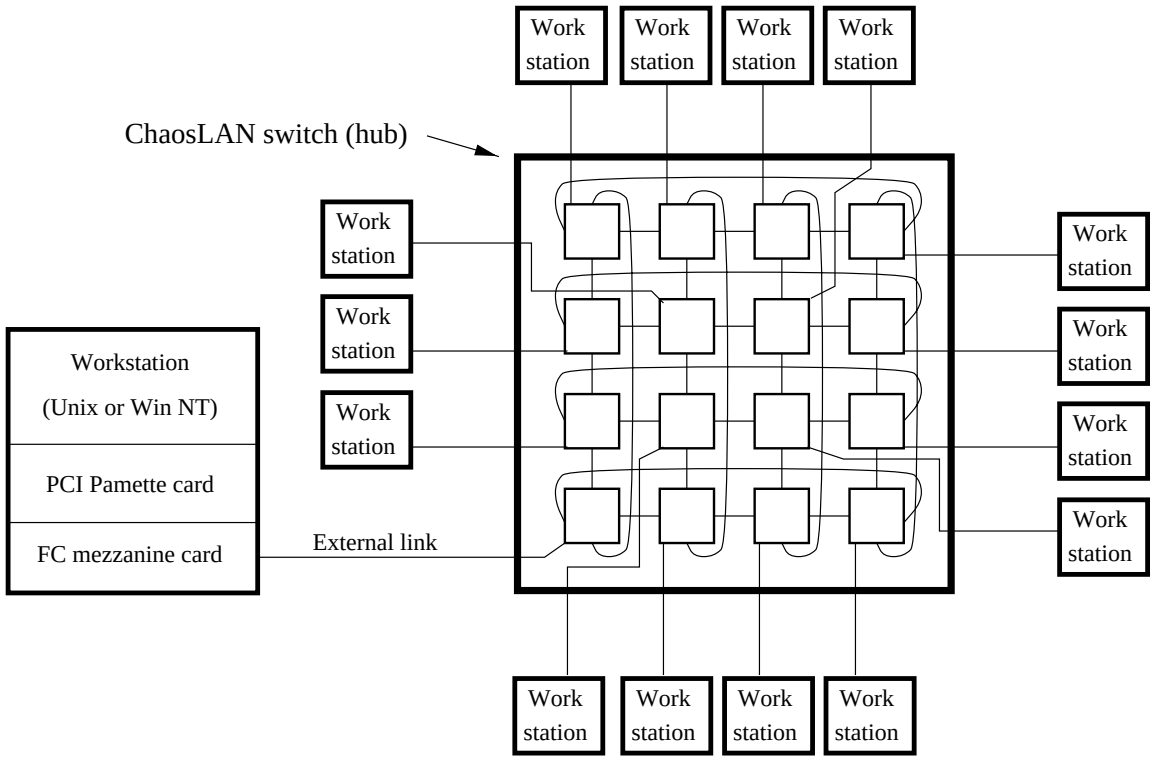

Figure 1: Overview of the implementation of ChaosLAN

is an opportunity for LAN technology based on Chaos to become adopted on a wide scale. The first step in making the research contribution of Chaos widely available is the development of a prototype system called ChaosLAN.

The remainder of this paper is organized as follows. Section 2 presents an overview of the ChaosLAN switch. Section 3 discusses the approach to constructing the host adapter that provides the interface between the network and an attached workstation. Two software environments are described: Section 4 introduces the Global Memory System and Section 5 discusses the use of ChaosLAN in a real-time distributed system for use in surgical simulation.

2 ChaosLAN switch

The idea of using a network of multicomputer routers to form a LAN switch is very natural. The essential idea is to take a multicomputer system (e.g. a massively parallel processor or MPP system) and replace the directly attached processing nodes with remote workstations, as illustrated in Figure 1. In this example, sixteen network router chips organized as a torus are contained within the switch. Since each router chip can be linked to a workstation via its processor port, such a network can interconnect up to sixteen workstations. It is also possible to form larger networks by connecting

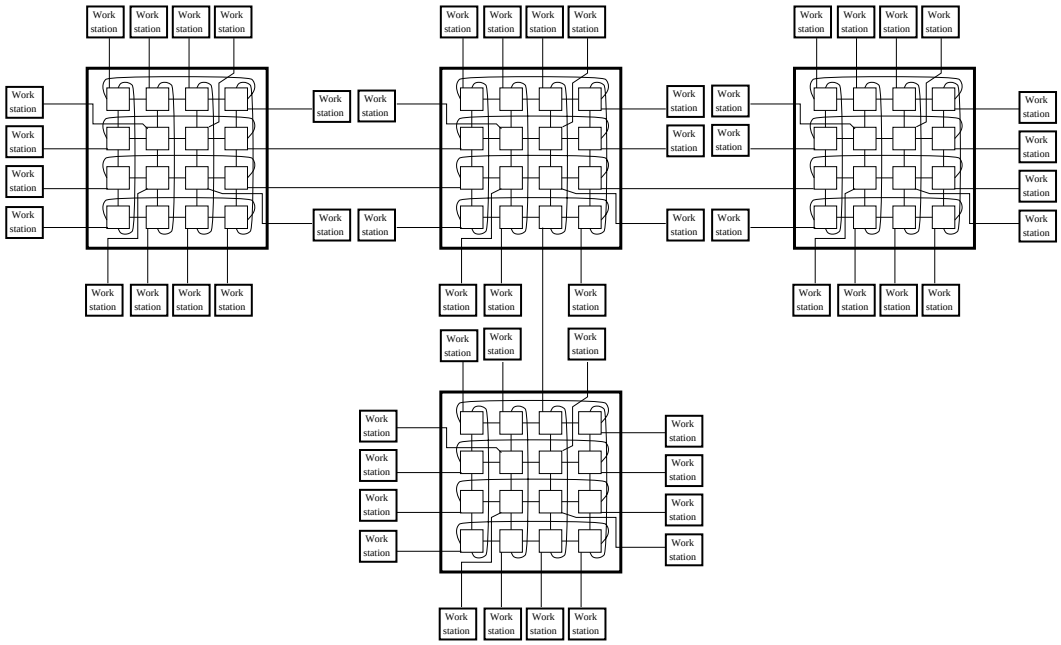

Figure 2: A ChaosLAN configuration with four switches and 56 workstations

multiple switches together through their processor ports. There are a large number of possible network configurations using multiple switches. As an example, Figure 2 demonstrates a configuration using four switches to connect 56 workstations. Note that redundant connections between pairs of switches may be used to increase the inter-switch bandwidth.

Although the conceptual transformation of a multicomputer network into a LAN switch is straightforward, there are some complicating issues. Three issues that arise when using Chaos networks are described below:

- The Chaos router expects the network to be a fully-populated regular mesh or torus. LANs usually have irregularly-shaped networks.
- The Chaos router relies on network nodes being physically close together (i.e. within 10cm of each other) so that transmission-line delays between nodes are negligible. LANs usually allow workstations to be at least 30-50 meters apart.
- The Chaos router relies on *de-routing*, in which packets are sent along potentially non-minimal paths to route around congestion. De-routing provides deadlock freedom and performance benefits. For de-routing to be beneficial, the cost of an extra network hop must be small. This is not the case in a LAN environment, where an extra hop may cause a large amount of additional latency.

All of these issues arise from the fact that the Chaos router is designed to take advantage of the special constraints provided by a multicomputer environment. To allow the Chaos router to work in the more general LAN environment, the ChaosLAN isolates the Chaos router inside of the switch by designating *internal* and *external* links. Internal links connect routers within a switch, providing a closely controlled environment that meets all of the needs of the Chaos router: a regular network, short internal links, and small delays per-hop. External links are used to connect workstations to the switch and to connect switches to other switches.

The interface between internal and external links is the processor port of each router. Specifically, each Chaos router chip has an extra port reserved for communication between the router and a host processor. The protocol for this port is allowed to be different from the protocol used for router-to-router communication. Because the processor port is not constrained to use the same transmission protocol as the regular router ports, it can be designed to tolerate the distances and irregularity found in LANs. Furthermore, once packets have been transmitted out of a switch and into the processor port, they are considered to be outside the Chaos router network and are no longer subject to its deadlock prevention protocol or de-routing. Thus, internal links use the Chaos protocol, while external links use a protocol designed specifically for the longer distances found in a LAN.

2.1 External Links

The linkage between the switch and a workstation (or from switch to switch) is serial and simultaneously bi-directional. The physical link protocol is based on the Fibre Channel physical interface (FC-PH), an ANSI standard for gigabit serial interconnection [8]. While the physical transfer protocols are the same as specified in the Fibre Channel standard, a custom protocol is used for higher levels. The specification of FC-PH describes three different layers, called FC-0, FC-1 and FC-2, representing the physical medium, the parallel-to-serial conversion and packet framing, respectively. Only FC-0 and FC-1 are used in ChaosLAN. Variants of FC-0 use either copper wire or optical fiber. The link length of low cost copper media can be up to 25 meters and the more expensive media allow link lengths to be up to several kilometers. The serial bit rates are specified at 1062.5 Mbit/s, 531.25 Mbit/s and 265.625 Mbit/s. FC-1, the protocol for parallel to serial encode and decode, is based on an 8b/10b coding scheme patented by IBM [9]. Eight bits of message data are converted into 10 bits of serial data using multiple codings. To improve the reliability of the link, the encoder chooses an encoding on a byte-by-byte basis over long byte strings that best equalizes the total number of high and low bits sent. The 8b/10b encoding provides redundancy; if the decoder detects an illegal serial bit sequence, it asserts that the corresponding byte has been corrupted and causes the packet containing this byte to be discarded. The encoding scheme also allows the transmission of control rather than data values, such as an acknowledgment character or the start or end of a data block. FC-1 is simple to implement: the encoder and decoder require only about 100-200 gates each, so that high-speed versions can be readily constructed.

2.2 The ChaosLAN Router Chip

Once we have made the separation of links into internal and external switch links, the job of modifying the Chaos router chip for LAN operation is much simpler. Most of the modification is concentrated on the external (processor) port, although some modifications must also be made to the internal ports as well. To facilitate the transfer of data from the external ports to internal ports, we use a common packet size and format throughout the entire network. Specifically, we use a packet with a 64-byte payload and a 12-byte header, requiring a total of 76 bytes in the packet, as opposed to the original Chaos router chip's 40-byte packets. Also, since the chosen external channel interfaces are effectively 8-bit uni-directional channels, the internal channels have been changed to reflect 8-bit words instead of the original 16-bit words.

The external link port has been modified extensively since it now must communicate over a relatively long wire length using a bit-serial protocol. The original router was able to function with single-packet input/output buffers because the short links between routers allowed for very fast transmission of flow control information. To support the relatively long length of the links on ChaosLAN's ports, more buffering has been added. A credit-based scheme provides the flow control. This scheme is implemented by providing buffering for four packets on the input side of each link. The sender keeps a count of the number of free buffers at the receiver and decrements the counter each time it sends a packet. When the receiver frees up a buffer (by forwarding the packet) an acknowledgment is sent to the sender, causing the sender to increment its count of free buffers at the receiver. Full-speed transmission can proceed as long as the round-trip time across a link is shorter than the time to send four packets. With a packet size of 76 bytes, this amount of buffering allows links of over 300 meters in length to run at full rate.

Finally, to allow packets to follow routes through networks with more than one switch, the ChaosLAN router must support some sort of source-based routing. Our implementation allows each packet to specify up to four packet headers. Each header is used to route a packet from a source link to a destination link in a single switch, allowing packets to cross a maximum of four switches in their route from source to destination. Each header consists of one byte, thereby allowing a maximum switch size of 256 routing nodes. In all there are four bytes needed to specify the route through four switches. Each router byte is subdivided further into two four-bit fields to represent the X and Y displacement within a single switch. An internal router-to-router hop causes either the X or the Y displacement to be incremented or decremented by one. When the header becomes zero then the whole packet is ejected from the internal network and out of the switch along an external link. Each time a packet leaves a switch, the set of four headers is rotated to place the next routing byte at the front of the packet.

3 Host Adapter

The host adapter provides an interface between the peripheral bus in the workstation host and the serial link to the ChaosLAN switch. We chose PCI as the peripheral bus standard due to its wide availability across IBM PC compatibles, Power Macintoshes and Unix-based workstations. To ex-

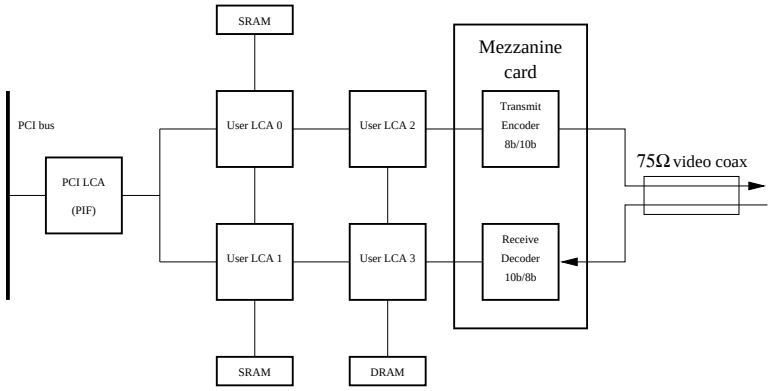

Figure 3: Organization of the Pamette card and the Fibre Channel mezzanine card

pedite the construction of the host adapter card, we are basing it on Digital Equipment Corporation's PCI Pamette card [7]. Figure 3 shows an overview of the Pamette. It contains five Xilinx logic cell arrays or LCAs, more generically known as field programmable gate arrays (FPGAs). One FPGA, called the PCILCA or PIF, provides an interface with the PCI bus. The other four FPGAs are the user programmable LCAs that contain the network interface circuit. The development environment for the Pamette includes a compiler and simulator for the Verilog hardware description language and the Xilinx logic synthesis tools. Interaction with the serial link requires a mezzanine card to perform the 8b/10b encode on the transmit side and the 10b/8b decode on the receive side, along with the voltage level conversion required for the physical medium.

To be successful, the network interface circuit in the host adapter must provide user programs with low latency access. The implementation of the Pamette network interface circuit is based on the Cranium network interface architecture [4, 5], a low latency interface framework that also provides compatibility with adaptive routers such as Chaos. Providing a low-latency, low-overhead solution requires two important features. The first requirement is to bypass the operating system and provide direct application program access to the network wherever possible. The second requirement is for the host adapter to perform bus-master direct memory access (DMA) to the host's system DRAM over the PCI bus. The idea is to transfer data to and from the network at the full rate of the PCI bus. There should be no intervention from the host processor in the common case. DMA channels at the sender automatically segment the message into packets and inject them one-by-one into the network without host intervention. Likewise, the DMA channels at the receiver perform automatic re-assembly of incoming packets. The host adapter notifies the host processor when all packets for a given DMA channel have arrived.

The following is a brief overview of how direct application access is implemented in the host adapter. (Also see the documents on Cranium [4, 5] for a more detailed description.) The host adapter must act on behalf of the operating system to check arguments in message commands coming from the application program, to prevent buggy or malicious user programs from crashing the operating system, for instance. In a send command, the user program selects the ID of the destination workstation, a reference to a local message buffer containing the message to be sent, and a DMA channel to execute the command. All message buffers are the size of an MMU page; a single send command re-

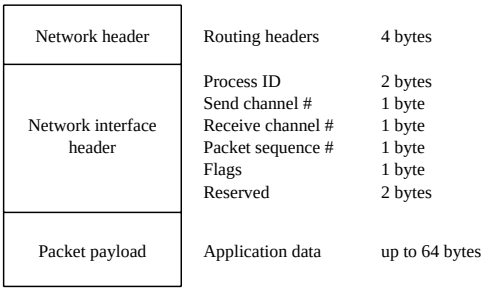

Figure 4: Packet format

sults in the sending of a message up to the size of a page. In order to use a particular message buffer, the user program must register it with the operating system in advance so that the OS can pin it into memory. The OS passes back a *handle* (a small integer) to the user program to represent this message buffer. Under direct application access, the user program executes a send command by passing the buffer handle directly to the host adapter instead of sending the address of the buffer. Mapping tables in the network interface are used to translate the handle into the physical address of the message buffer. Unmapped handles are rejected; if the entry in the table indexed by the buffer handle contains a null value, then the message command is cancelled. A receive command works the same way: it specifies the ID of the workstation representing the source of the message, the handle of a local buffer to place the incoming data and a DMA channel. Note that a receive DMA channel must be activated before any packets regarding that channel are sent.

The user can specify two different ways for each receive DMA channel to be managed by the network interface. For small messages, the receive DMA channel is managed as a ring-queue; the memory location assigned to an incoming packet is determined by the receiver's queue pointers that index into the message buffer. For large messages, the locations of incoming packets are determined by using the sequence number in the packet header as an offset from the physical address of the message buffer. The effect is that packets are re-ordered in the message buffer automatically regardless of what their arrival order is. A hardware packet counter determines when all the packets in the message have arrived. The host processor can poll a status register to test for the arrival of a complete message. An alternative approach is for the network interface to notify the host using an interrupt.

Figure 4 describes the packet format used in ChaosLAN. The routing information is contained in of the first four bytes of the packet and is manipulated by the switches directly. The next eight bytes contain the network interface header. The send channel and receive channel fields are both single byte fields, meaning that each field can address up to 256 DMA channels. The packet payload contains up to 64 bytes worth of application data; the payload length is specified in the flags field in the header. Assuming that the page size is 8K bytes, a single DMA transfer contains up to 128 packets. The sequence number field (also a single byte) represents the cache-line offset into the memory page. For most processor architectures, a single full length packet represents two or four cache lines worth of data. The last two bytes in the network interface header are reserved for future expansion.

4 Application: Global Memory System

To evaluate the effectiveness of ChaosLAN and the Pamette-based network interface, we plan to run a software environment called the Global Memory System (GMS) [10, 11]. GMS is based on the idea of paging over the network to idle workstations instead of to local disk. As network bandwidth approaches the bandwidth of a locally attached disk, paging over the network becomes the faster solution once the communication latency becomes sufficiently low.

Here is a brief introduction to GMS. Under standard virtual memory, recently accessed pages remain in main memory while inactive pages must be read from disk. GMS introduces a new class of page called a *global page*. A local page is a recently accessed page, currently available in main memory. A global page is one that was accessed some time ago but not recently enough to keep it in main memory. A page is demoted from local to global when the user program runs out of page frames. The user program makes room by selecting the least recently used local page and moves it onto another workstation (hence making it a global page). The global page is promoted to local when the user program faults it back into memory. In many cases, a single page fault results in an exchange of pages between two workstations: first a page is pushed out, then another page is pulled in.

The requirements of GMS are a natural fit with the Cranium network interface architecture because both are page-oriented. One simplification is that all network activity is the side effect of taking a page fault, rather than being activated explicitly by the underlying user program built on top of GMS. Therefore the capability of direct application access is not strictly necessary. The primary benefit of Cranium is the use of bus-master DMA to move pages at the full rate of the hardware without host processor intervention on a packet-by-packet basis.

An extension to the simple page-oriented technique is to use sub-page transfers. In this way the application program that executes the page fault can start operating immediately after the particular cache line containing the data element that was requested in the page is brought in. The remainder of the page is sent shortly afterward. Since Cranium is cache-line oriented, there is a natural fit with the sub-page extension to the basic GMS framework.

5 Application: Surgical Simulation

The idea behind a surgical simulator is to model a surgical instrument (such as a scalpel or forceps) and allow the user to perform simulated surgery on a computer model of the human anatomy by manipulating a simulation of the instrument. It is very important for the simulation to be as realistic as possible, by displaying visual changes to the anatomical model in real time with no perceived lag. The most effective surgical simulators also provide force feedback, also known as *haptic* (touch) rendering. Thus, the computer system must provide haptic rendering as well as graphics rendering.

Computational approaches for surgical simulation include finite element modeling, volume rendering and other compute-intensive techniques. In most cases, single processor systems are not adequate to provide enough realism to make the system usable. Currently it is necessary to use large

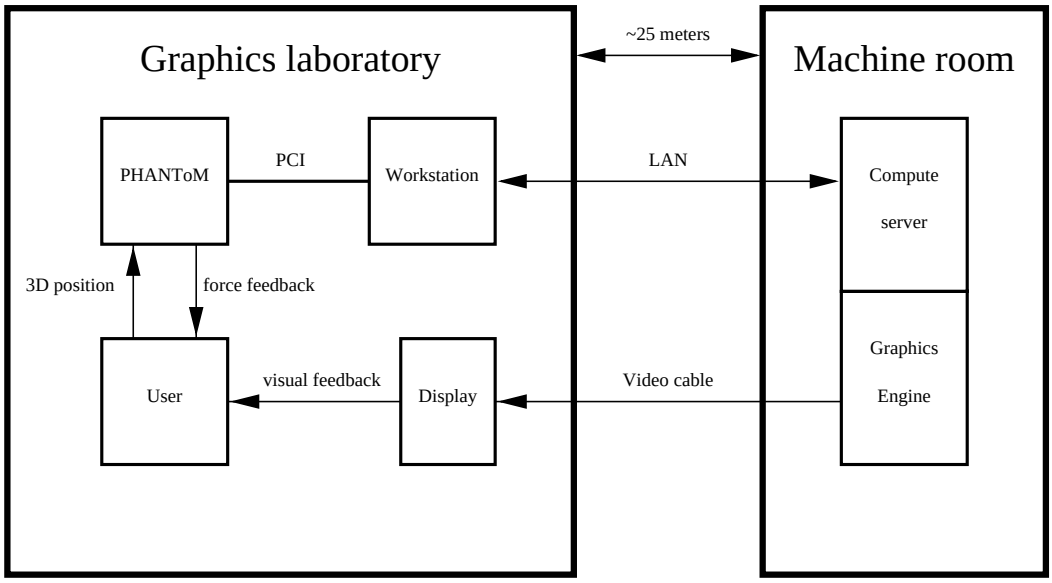

Figure 5: Schematic diagram of the surgical simulation demonstration system at MERL

multiprocessor compute servers. The electrical and environmental requirements of these compute servers are more substantial than those of ordinary workstations and personal computers; it is common for medium to large compute servers to be placed in a dedicated machine room containing additional cooling capability and high-wattage electrical power distribution. Thus, the compute servers are usually located remotely from the laboratory where the user interacts with the simulator. The effect is that the haptic input device cannot plug directly into the compute server due to cable length limitations. The solution is to drive the haptic device off a local workstation and communicate with the compute server over a LAN.

Figure 5 describes the environment for a surgical simulation demonstration system that has been constructed at MERL – A Mitsubishi Electric Research Laboratory [12]. The haptic input device is the PHANToM from SensAble Devices. The position of the PHANToM stylus in 3-space is captured by a local workstation. The local workstation performs haptic rendering and a remote compute server performs graphics rendering. Position information from the PHANToM is transferred over the LAN to the compute server, an SGI Onyx SMP system containing eight R10000 processors and an InfiniteReality graphics accelerator. The Onyx performs the graphical rendering in real time. The graphics output is delivered from the Onyx back to the room with the PHANToM over video cable and then displayed on a monitor.

Currently, the haptic model is contained entirely within the workstation attached to the PHANToM. Eventually we would like to perform haptic rendering on the compute server. The advantages are that the larger computational capability of the Onyx will allow us to model deformable tissues and simulate cutting, tearing and squishing. Currently, this capability is defeated by the large latencies of the Ethernet-based LAN. Haptic forces must be re-calculated and refreshed a rate of at least 1000 Hz and ideally 5000 Hz or greater. With standard LAN technology a round trip takes 2 to 5

milliseconds. Therefore it is impossible to sustain a 1000 Hz refresh rate. By contrast, the human visual system tolerates a much slower refresh rate, on the order of 20-25 Hz (40-50 milliseconds per video frame), so the latency of standard LANs is not detrimental to the visual quality.

A high-performance LAN such as ChaosLAN with a low-latency network interface provides a solution to the problem of locating the haptic model remotely on the compute server. If the round-trip latency due to the network and host adapter can be reduced to 50 to 100 microseconds instead of 2 to 5 milliseconds, then the haptic refresh rate can be improved to an acceptable level. Unlike the case of GMS, this application requires direct access from the user level on the sender to the user level on the receiver to bypass the operating system entirely. The amount of information in each message is small (on the order of three to eight floating point values) and can be contained in a single packet. Therefore, the application can manage the receive DMA channels as ring-queues. A credit-based protocol can be used to prevent overflow: the workstation sends one packet (containing position information) and waits for a response from the compute server (containing force information) before it sends its next packet.

6 Project Status

The implementation is divided into several tasks: the router chip, the circuit board containing the router chips to implement the switch, the mezzanine circuit board to extend the PCI Pamette to provide a Fibre Channel physical interface, and programming the Pamette's FPGAs to implement the host interface circuitry. The ChaosLAN router chip is expected to be completed and fabricated in the summer of 1997. Using an 0.8 micron process and 8-bit ports, it should be easy to achieve the 125 MHz operating speed necessary to provide the 1 Gbit/s transmission rate per link. Likewise, the printed circuit boards are expected to be fabricated during the summer of 1997. If all goes well, we expect to have an operational system by the end of the year.

At time of publication we do not have any performance measurements available for ChaosLAN. Results of simulation studies are available on the Chaos routing algorithm [3] and the Cranium network interface [5]. Some variables not captured by prior simulation studies on chaotic routing are the effects of the PCI bus implementation and the operating system on overall system performance. Both factors are expected to have a significant impact on the performance of ChaosLAN. Moll and Shand [13] measured the PCI bandwidths and interrupt response times for a number of different workstations and PCs, under both the Digital Unix and the Microsoft Windows NT operating systems. Their measurements demonstrate that PCI performance results vary considerably across different implementations. Interrupt response times for a *single* implementation vary widely, from a few microseconds to a few hundreds of microseconds, and under rare circumstances are over a millisecond. Since the interrupt response time profile is difficult to model analytically, the best simulation is provided by a physical implementation.

7 Summary

We describe ChaosLAN, a high-performance local area network based on the Chaos router, originally used in multicomputer routing. The network consists of a hub (switch) containing 16 Chaos routers; the switch is able to connect up to 16 workstations over serial links. Some changes to the Chaos chip are needed to address the problems of the long wire length from the switch to the workstation. The serial link is based on the Fibre Channel physical layer. We are using the PCI Pamette card to implement the host adapter. A subset of the Cranium network interface architecture is executed by the Pamette's FPGAs to provide a low-latency interface with the network. Two software environments with latency-critical properties are under consideration to test and exercise ChaosLAN: the Global Memory System (GMS) and an environment tailored for haptic rendering and surgical simulation.

Acknowledgments

We appreciate the contribution of UW undergraduate students Ville Aikas, Jason Murray and Kevin Taylor to the development of ChaosLAN. We also acknowledge the assistance of Mark Shand at Digital's System Research Center.

References

[1] Thomas E. Anderson, David E. Culler and David A. Patterson. A case for networks of workstations (NOW). *IEEE Micro*, Feb. 1995.

[2] Lawrence Snyder et al. The World Wide Web home page for the Chaos research group at the University of Washington. URL: `http://www.cs.washington.edu/research/projects/lis/chaos/www/chaos.html`.

[3] Kevin Bolding. *Chaotic routing: design and implementation of an adaptive multicomputer network router.* PhD dissertation, University of Washington, Dept. of CSE, Seattle WA, July 1993.

[4] Neil R. McKenzie, Kevin Bolding, Carl Ebeling and Lawrence Snyder. Cranium: an interface for message passing on adaptive packet routing networks. *Proceedings of the 1994 Parallel Computer Routing and Communication Workshop*, Seattle WA, May 1994, pp. 266-280.

[5] Neil R. McKenzie. *The Cranium network interface architecture: support for message passing on adaptive packet routing networks.* PhD dissertation, University of Washington, January 1997.

[6] Nanette J. Boden et al. Myrinet: a gigabit-per-second local area network. *IEEE Micro*, February 1995. Also available on the World Wide Web through `http://www.myri.com/research/index.html` .

[7] Mark Shand et al. The PCI Pamette V1. World Wide Web site, `http://www.research.digital.com:80/SRC/pamette/`.

[8] Fibre Channel – Physical and Signaling Interface (FC-PH). ANSI document X3.230-1994.

[9] A. X. Widmer and P. A. Franaszek. A DC-balanced, partitioned-block, 8b/10b transmission code. *IBM Journal of Research and Development*, 1983.

[10] M. J. Feeley, W. E. Morgan, F. H. Pighin, A. R. Karlin, H. M. Levy, and C. A. Thekkath. Implementing global memory management in a workstation cluster. *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.

[11] H. A. Jamrozik, M. J. Feeley, G. M. Voelker, J. Evans II, A. R. Karlin, H. M. Levy, and M. K. Vernon. Reducing network latency using subpages in a global memory environment. *Proceedings of the Seventh ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, October 1996.

[12] Sarah Gibson, Joseph Samosky, Andrew Mor, Christina Fyock, Eric Grimson, Takeo Kanade, Ron Kikinis, Hugh Lauer, Neil R. McKenzie, Shin Nakajima, Hide Ohkami, Randy Osborne and Akira Sawada. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. *Proc. of the Joint Conference on Computer Vision and Virtual Reality in Medicine and Medical Robotics and Computer Assisted Surgery*, Grenoble, France, March 1997.

[13] Laurent Moll and Mark Shand. Systems performance measurement on PCI Pamette. *Proc. of FCCM'97 Symposium on Field-Programmable Custom Computing Machines*, Napa CA, April 1997.