VISTAnet Network Interface Unit: Future Communications Research

(TR91-018)

Raj K. Singh
Stephen G. Tell
David Becker
Shaun J. Bharrat

Microelectronic Systems Laboratory
The University of North Carolina
Department of Computer Science
CB# 3175 Sitterson Hall
Chapel Hill, NC 27599-3175

The work undertaken for the VISTAnet project is supported, in National Science Foundation and the Defense Advanced Resea Agency under cooperative agreement NCR 8919038 with the Co National Research Initiatives. Support is also provided by BellS and GTE Corp.

CONTENTS

VISTAnet Network Interface Unit: Future Communications Research

Abstract

Heterogeneous parallel and distributed computing is gaining broad acceptance as a viable means of speeding up a large class of target applications such as medical imaging, weather prediction, and geological simulations. High-speed communication links are necessary and common among such infrastructures to enable exchange of code, data, and control information among the host computers in the network. However, the diversity of hardware and software architectures of machines pose major hurdles in the path of integrating these systems in a high-bandwidth network. Programmability to accommodate unforeseen needs and performance to meet the throughput demands of the applications are competing requirements on the part of a network interface unit that needs to be resolved in order to make networked-computing more practical.

VISTAnet is a research endeavor that attempts to harness the capabilities of three geographically dispersed supercomputers in North Carolina. These computers are interconnected via a public network operating at gigabit speed. We have designed and developed a high-performance, programmable, custom Network Interface Unit (**NIU**) for interfacing the Pixel-Planes 5, a custom graphics supercomputer developed at UNC, to the VISTAnet. Although the NIU is primarily designed as an error tolerant, low-latency communication interface, it will also serve as a platform to support future communications research. We present an outline of the proposed research in real-time operating system and process management, protocol design and development, novel algorithm design, and to explore the tradeoffs between hardware and software subsystems required to support emerging wide area gigabit networks.

1. INTRODUCTION

The VISTAnet project is one of five national testbeds organized by the Corporation for National Research Initiatives and funded in part by the National Science Foundation, the Defense Advanced Research Projects Agency, and corporations such as BellSouth and GTE. The project advances research into both the design and operation of such gigabit networks as well as applications capable of harnessing gigabit communications.

The infrastructure consists of three supercomputers, namely, a **Cray Y-MP** at the North Carolina Supercomputer Center, Pixel Planes 5 (**PXPL5**), and a MasPar **MP-1** in the Computer Science Department at UNC, and a Silicon Graphics **340 VGX** which is the medical workstation in the Department of Radiation Oncology at UNC. PXPL5 is a heterogeneous multicomputer optimized for graphics applications. The host computers are connected via a B-ISDN network installed and managed by BellSouth and GTE.

The target application is Dynamic Radiation Therapy Planning (**DRTP**). Current radiation treatment planning involves examining two-dimensional Computer Aided Tomography and Magnetic Resonance Imaging views of various slices of the patient's anatomy. Generally, only a few slices are used to prevent the explosion of information generated by large numbers of slices. The limiting of the number of views may result in sub-optimal treatment plans in some cases. The DRTP application displays, in three-dimensions, the proposed radiation beams superimposed over

the patient's anatomy and enables the physician to interactively adjust beam parameters (direction, intensity) or change the point of view. A modification of the radiation beam results in recomputing of dosage tables on the Cray which are then sent to the PXPL5 for rendering. A change of view point is transferred directly to the PXPL5. In either case, an updated image is produced on the medical workstation in real-time.

We have outlined a plan to explore the suitability of object-oriented software, traditional protocol architectures, and single-processor network interfaces in this environment. This involves profiling delay in a object-oriented operating system, performance measurement of various protocol stacks and evaluation of alternative methods of error handling and flow control. Sections 2 & 3 provide an overview of the B-ISDN network, network interfaces to the hosts and performance goals for the NIU. The specifics of the future research are presented in section 4.

2. OVERVIEW

2.1 B-ISDN Network

The VISTAnet network (Fig.1) consists of a prototype ATM switch developed by Fujitsu and a broadband circuit switch developed by GTE. The two switches are connected by a 2.488 Gbps fiber-optic link carrying multiple OC-12c channels. The network is extended to the test sites with single 622 Mbps links and is terminated at each site by a Network Terminal Adapter (**NTA**). The NTA is an ATM/HIPPI gateway and is directly connected to the HIPPI interface of each host computer. A detailed account of the network infrastructure is presented by Basch et al [1]. The switching fabric of the Fujitsu switch is described by Kato et al [6].

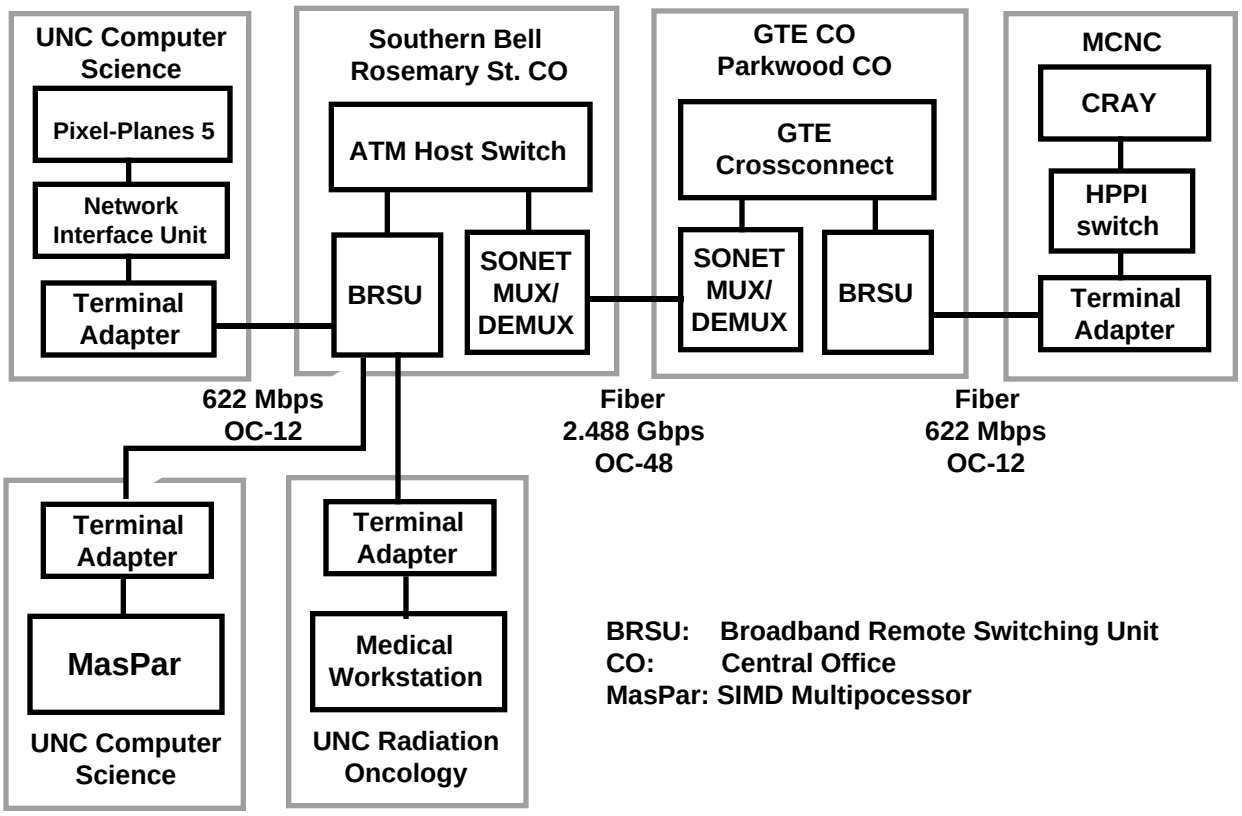

Figure 1: The VISTAnet B-ISDN Network

2.2 Network Interfaces

The Cray host interface is a proprietary standard HIPPI interface called model-D IOS channel. The protocol software consists of an implementation of TCP/IP stack with intermediate layers such as HIPPI-LE, IEEE 802.2LLC and HIPPI-FP. The SGI 340 VGX is currently connected via a HILDA HIPPI interface [9] (designed for network traffic analysis) which will be replaced by a standard HIPPI interface from SGI when available. The planned research does not include modifications to these interfaces although some changes in the protocol software may be necessary. The PXPL5 communicates over the network through NIU, a Ring/HIPPI gateway. A detailed account of the PXPL5 system is presented in [5]. The NIU is the subject of future research and its salient attributes are summarized here.

The NIU hardware subsystems are shown in Fig. 2. The HIPPI Destination subsystem accepts HIPPI packets from the network, validates, and then buffers them in 16 KB FIFO buffers. The RingBound subsystem transfers this data from the buffers onto the PXPL5 ring. Since packets are written and read in round-robin fashion to the buffers, the HIPPI Destination and RingBound system operate concurrently. The NetBound subsystem accepts packets from the PXPL5 ring and buffers them in a 16 16KB SRAM buffer and the HIPPI Source subsystem, in turn, transfers the HIPPI packets onto the network. The Netbound and HIPPI Source subsystems also operate concurrently. These four subsystems are controlled by the processor subsystem consisting a 25 MHz SPARC processor (CY7C611) with 1 MB of local memory.

The software architecture is shown in Fig. 3. The operating system is multithreaded and written primarily in C++ with some SPARC assembly routines. There are four main concurrent tasks: a Destination task, a RingBound task, a NetBound task, and a Source task. Each task supports the associated hardware subsystem. The current protocol stack implementation, also in C++, consists of HIPPI-PH, HIPPI-FP. The IP, TCP, HIPPI-LE and IEEE 802.2 layers are planned for the future. The hardware and software specifications of the NIU are covered in detail in [8].

3. NIU PERFORMANCE AND SERVICE GOALS

The performance goals for the NIU hardware and software can be gleaned from the throughput, latency, and error requirements of the DRTP application. The Fig. 4 shows the data flows within the system. The physician enters the changes in the radiation treatment at the medical workstation. These changes, less than 1K of data, are transported to the Cray where the dosage distribution tables are computed. This dosage distribution is then sent to the PXPL5 for rendering and the output images forwarded to the SGI workstation.

The dosage data frame transferred from the Cray to the PXPL5 is typically 1 MB in size but may be as large as 8 MB. Depending on whether the radiation beam is being continuously changed or not, the rate of dose data frame recalculations on the Cray can vary from once every 200 ms to once every 100 ms resulting in a bandwidth requirement, from the Cray to PXPL5, of 160 Mbps to 640 Mbps. The rendered images sent from the PXPL5 to the SGI workstation is currently transmitted as analog data.

The DRTP application is interactive and requires low latency. The maximum tolerable latency is subjective and depends on the medical usefulness of the rendered image. A suggested goal for the latency is based on the ability to display four data frames per second or 250 ms. Although the DRTP application produces images, it is not a traditional imaging application. Radiation dosage data is sent from the Cray to PXPL5 in multiple packets. For this purpose, a loss of entire dose data (an improbable situation) may be acceptable, whereas, a loss of partial data (a more likely scenario) is not. The DRTP application therefore requires error-free network service from the Cray to PXPL5. Data transfers between the SGI workstation and the Cray, and between the SGI workstation and the PXPL5 should also be error-free.

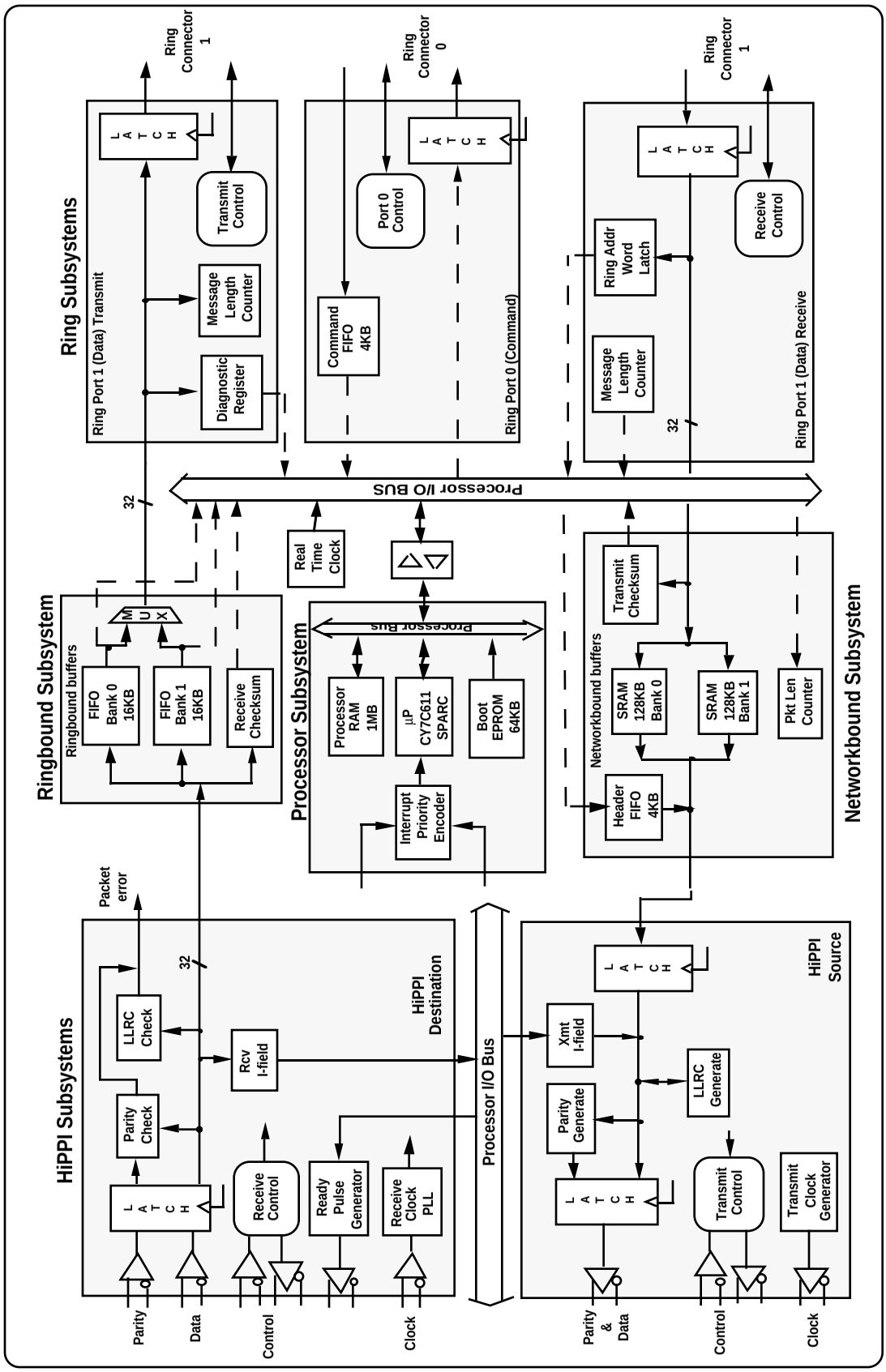

Figure 2: NIU System Block Diagram

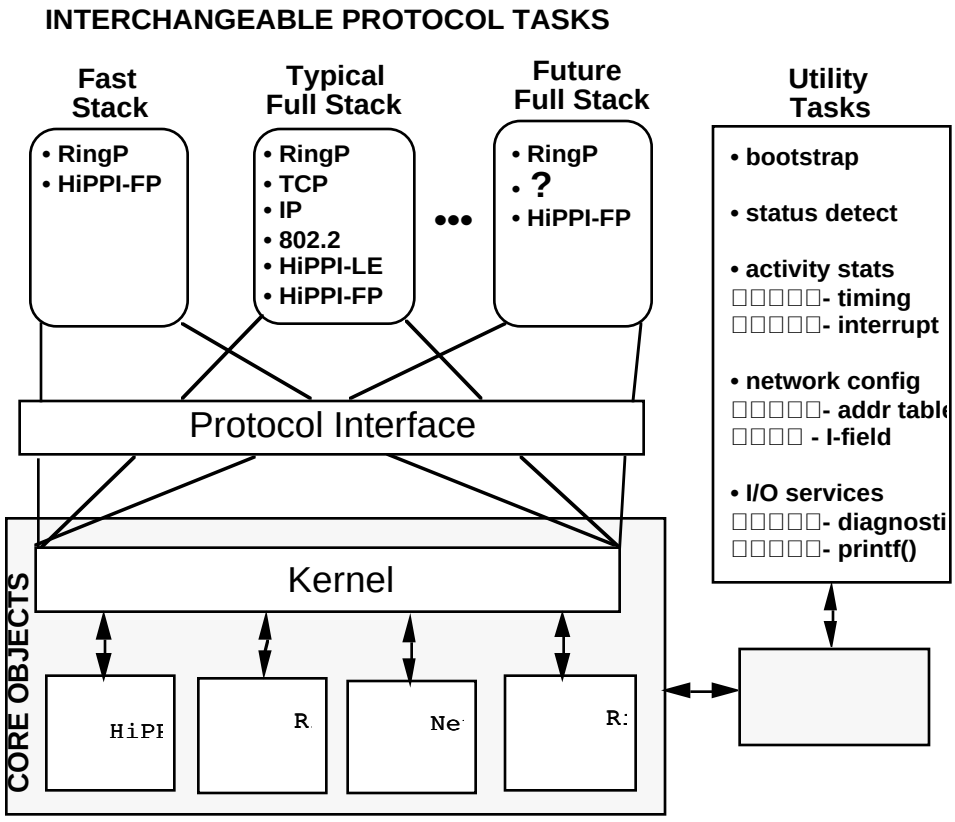

Figure 3: NIU Software Architecture

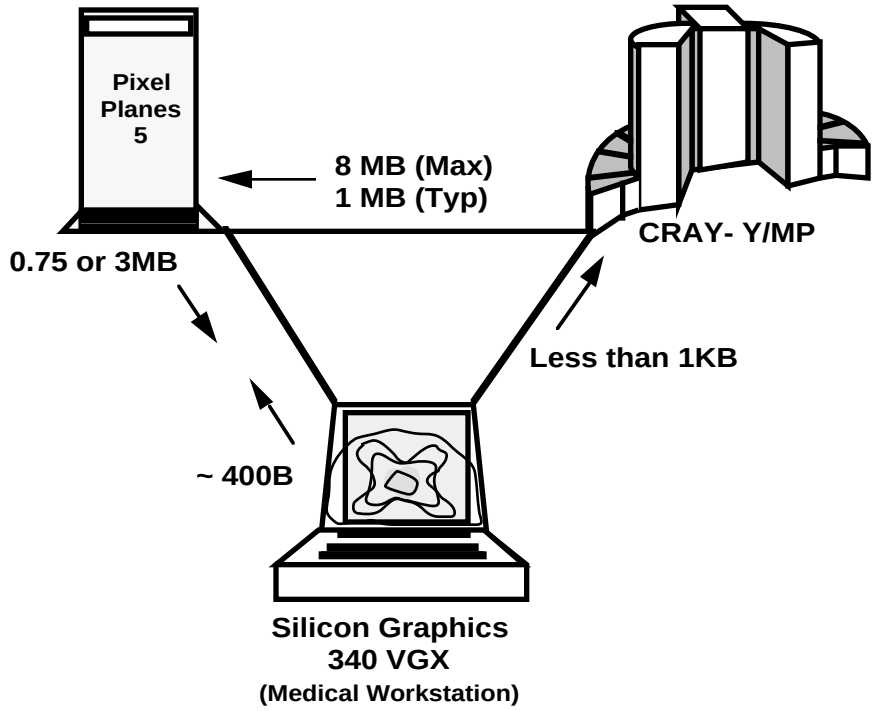

Figure 4: Application data flow per dose calculation frame

4. FUTURE RESEARCH

4.1 Software Architecture

4.1.1 Object-oriented design

The basic performance requirement for the NIU is throughput at gigabit rate. This requirement often directly conflicts with goals of software modularity, portability, and maintainability. The NIU software architecture conforms with current software engineering paradigm of object-oriented design and implementation which emphasizes modularity and extensibility. The performance costs of this approach need to be determined, however, the method for evaluating these costs is not clear. A naive approach of implementing a traditional real-time operating system for the NIU and comparing performances of individual components is not practical. A reasonable approach in this case may be to evaluate the performance of only a few critical components such as the kernel and the protocol processing objects.

4.1.2 Division of function

The functionality of the NIU software is distributed among four main concurrent threads. This approach exploits the expected asynchronous nature of the four functional blocks (receiving traffic from the PXPL5 ring, sending traffic onto the HIPPI network, receiving traffic from the HIPPI network, and sending traffic onto the PXPL5 ring) and follows established software engineering principles of strict modularity. However, such a division introduces the overhead associated with context switching. This overhead needs to be quantitatively documented and the asynchronicity of the functional areas must be confirmed. Such an examination could dictate either a merging of functional areas to ameliorate context switching overhead or a further subdivision of areas to exploit additional asynchronicity.

4.1.3 Priority/Preemption mechanism

The traffic patterns for this network are well defined. The bulk of the data flows from the Cray to the PXPL5. The other data flows are almost negligible by comparison. The NIU exploits this imbalance by prioritizing the inbound path over the outbound path. At present, an inbound packet will always be serviced before an outbound packet. The applicability of this approach for a more general traffic pattern needs to be investigated. Detailed measurement of the work done by the four main threads can be used to devise a static real-time schedule that guarantees best performance.

4.1.4 Delay profile of operating system

Effective optimization of the NIU software requires first the identification of the potential bottlenecks. While the overall performance of the software (with various protocol software layers) has been determined, the delay profiles of the individual components will be documented.

4.2 Protocol Stack

4.2.1 Delay profile of protocol stack

A full protocol stack for the NIU will be implemented. The stack consists of the HIPPI-FP, the HIPPI-LE, IEEE 802.2, IP, and TCP sublayers. The stack is being implemented as a set of C++ classes built on a base **protocol class**. Strict layering principles are being used: each sublayer is a class derived from the sublayer below it; flow of control starts at the top (for transmission) or bottom (for reception) and passes serially through each layer. Each class provides a semantically equivalent **send-packet** and **receive-packet** function with the upper layers' functions associating more services with these functions. For example, the send-packet at the TCP layer includes sequencing and error control. Additionally, the TCP layer also provides a socket interface for application programs.

This approach helps our research by simplifying implementation and facilitating proof of correctness and performance testing. We intend to determine empirically the bounds on throughput and processing delay for various partial stacks. The strict layering is helpful here because sublayers can be excluded by simply terminating the protocol object derivation at any particular class. Each class has a generic send and receive function so changes to the testing program are minimal. Since each protocol layer corresponds to a software layer, the bounds for any particular layer can be incrementally determined.

4.2.2 Alternate implementations of TCP/IP

Layering in communication protocols is an established paradigm. It offers a means for effective abstraction, ease of implementation, and verification as each layer is presumably independent of the others. Additionally, protocol software layering also makes explicit the processing requirements for each protocol layer. This information can be useful in evaluating the feasibility of implementing a particular layer in hardware.

The alternative approach is a monolithic or semi-monolithic protocol stack. Such combinations would present the interface of the top-most and bottom-most layers but include the functionality of all included layers. Because of the minimal processing required for the HIPPI-FP, HIPPI-LE, and 802.2LLC layers, their amalgamation into one protocol entity may improve overall throughput. Modularity and independence are sacrificed with monolithic and semi-monolithic approaches so creating other protocol stacks will be more difficult.

4.2.3 Error detection and recovery

The DRTP application is not error-tolerant. The loss of a packet results in an incomplete dosage table and the application may hang. It is clear that packet loss must be detected. It is less clear as to what error recovery steps should be taken. The TCP packet retransmission strategy in this interactive real-time environment may cause visible effects at the application level. If the dosage tables are being constantly updated (such is the case when the physician is continuously altering the radiation beam parameters), it may be better to skip the incomplete dosage table and go on to the subsequent one. The protocol software cannot do this transparently to the application. The application must make the decision to discard incomplete dose tables. The protocol software can only inform the application of packet loss. We intend to investigate this possibility of light-weight error handling.

4.2.4 Flow control

The HIPPI-FP protocol provides end-to-end flow control. In the future, however, the possibility exists for the HIPPI lines to be multiplexed across several ATM connections. In such a scenario, the throttling effect of the underlying network is removed and need for transport-level end-to-end flow control becomes acute. Questions about the efficiency of the TCP/IP sliding window scheme in certain high-bandwidth environments have been raised and we need to evaluate it in our specific environment to ensure that it does not inadvertently throttle data transfer but just prevents the NIU from being overrun.

4.2.5 Evaluation and implementation of alternate protocols

The throughput of the NIU has been measured with the HIPPI-FP protocol stack in local loopback mode. The maximum throughput is in excess of 784 Mbps with single threaded performance firmware. The full service multi-threaded firmware transfers data over 400 Mbps. The implementation of HIPPI-LE, IEEE 802.2LLC, and IP layers should not add much overhead since the processing required in these layers is minimal. However, the addition of TCP layer with acknowledgments is likely to add significant latency. The throughput is also expected to drop. We will study these aspects quantitatively in detail.

Much has been written in the literature about alternative protocols for high-speed low-error networks [2,3,4,7] as well as adaptations of common protocols [4]. The feasibility of alternative transport protocols for the NIU will be evaluated in conjunction with efforts to install these protocols on the Cray and medical workstation.

4.3 Hardware Architecture

4.3.1 Buffer size

Because of expected buffer limitations on the NTAs and the unavailability of sufficiently fast high density FIFOs at the time of the NIU design, the decision was made to provide two 16 KB buffers each for the source and destination subsystems. Consequent testing has revealed that the throughput of the Cray is severely limited due to the 16 KB packet size. Additionally, the buffer limitations on the NTA have been eliminated and higher density FIFOs are now available. We intend to expand the FIFO buffers on the NIU in order to accommodate larger packet sizes and thereby increase the throughput between the Cray and the PXPL5. We do not anticipate that this expansion will necessitate any board changes other than the replacement of the FIFOs which were socketed by design.

5. SUMMARY

The Network Interface Unit (NIU), the communications interface for the Pixel Planes 5 graphics supercomputer, provides a fertile testbed for research in the operating systems, protocols and hardware necessary for supporting gigabit communications. Various issues need to be resolved. The advantages of object-oriented design in applications development has been ably demonstrated but its suitability in a real-time, high-bandwidth environment has to be verified. The performance profiles of various components of our software, including the multi-threaded kernel and the protocol stack, have to be compiled and compared to more traditional implementations. The environment also raises concerns about the limitations of TCP and the benefits of scaled-down versions and alternative protocols need to be carefully examined. Error handling and flow control are two issues that we feel must be handled differently in this type of environment.

The NIU hardware design has proved to be flexible and testing has indicated that it is performing near the theoretical bounds. The design verifies that a single-processor system can provide near gigabit performance for lower layer processing. We need to determine whether such rates can be maintained when transport protocol processing is added. A tentative change envisioned for the NIU is the expansion of the FIFO buffers to accommodate packet sizes larger than 16 KB. This upgrade will not require any changes in the circuit board since future devices will be pin-compatible .

ACKNOWLEDGMENT

The work undertaken for the VISTAnet project is supported, in part, by NSF and DARPA under cooperative agreement NCR 8919038 with CNRI. Support is also provided by BellSouth services and GTE Corp.

REFERENCES

[1] B. E. Basch, et al, "VISTAnet: A B-ISDN Field Trial," IEEE-LTS, pp. 22-30, August 1991.

[2] D. Cheriton, "VMTP: A protocol for the next generation of communication systems," ACM SIGCOMM '86 Symp., Stowe, VT, Aug 5-7, 1986, pp. 406-415.

[3] G. Chesson, "The protocol engine project," UNIX Rev., vol. 5, no. 9, pp. 70-77, Sept., 1987.

[4] W. A. Doeringer et al, "A Survey of Light-Weight Transport Protocols for High-Speed Networks," IEEE Tran. Comm., vol. 38, no. 11, Nov. 1990.

[5] H. Fuchs et al, "Pixel Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor Enhanced Memories," Computer Graphics, 23, 79-88, 1989.

[6] Y. Kato, T. Shimoe, and K. Murakami, "A Development of a High Speed ATM Switching LSIC," Proc. ICC, paper no. 310.3.1, 1990.

[7] "XTP Protocol Definition 3.4", Protocol Engines, Inc., 1900 State St., Suite D, Santa Barbara, CA 93101, 1989.

[8] Raj K. Singh, S. G. Tell, and D. Becker, "VISTAnet Network Interface Unit: Prototype System Specifications," TR91-017, Department of Computer Science, University of North Carolina at Chapel Hill, 1991.

[9] Dan Winkelstein and Dan Stevenson, "HIPPI Link Data Analysis System: Test Equipment for High Speed Network Analysis," presented at Tri-Comm 91, Chapel Hill, NC, April 18-19, 1991.