Appeared in Proceedings of Hot Interconnects '94

Using an ATM Interconnect as a High Performance I/O Backplane†

Zubin D. Dittia zubin@dworkin.wustl.edu
Jerome R. Cox, Jr. jrc@hobbs.wustl.edu
Guru M. Parulkar guru@flora.wustl.edu

Department of Computer Science
Washington University in St. Louis
St. Louis, MO 63130.

Abstract

We present the design of an ATM-based interconnect that serves as a high speed packet-switched equivalent of an I/O bus in a workstation/server. By effectively "extending" an external ATM network into the desk-area, the interconnect also functions as a high performance host-network interface. The prototype interconnect, which we are building as part of an ARPA sponsored gigabit local ATM testbed, boasts an aggregate sustained bidirectional data rate in excess of a gigabit per second. The interconnect design, which is targeted specifically for multimedia workstations/servers, employs a single custom VLSI chip called the *ATM Port Interconnect Controller* (APIC); by daisy-chaining a number of these chips, and connecting each to an I/O device, we can support the uninterrupted flow of continuous media data from device to device, as well as from a device to the network and vice-versa.

Introduction

The emergence of very high speed networks [2, 5] has opened up a plethora of possibilities for new and exciting distributed applications. But even as the first gigabit networks make their appearance, it is becoming increasingly evident that merely raising raw network bandwidth does not translate to improved performance for end-applications. The bottleneck is easily identified to lie within the end-host (i.e., the workstations or servers that run the end-applications). Within the end-host, a number of problem areas can be identified that contribute to the bottleneck, not the least of which is a lack of integration between the hardware architecture (particularly that of the host-network interface), the operating system, and network communication protocols.

Solving problems inherent in the software architecture or protocols would not help in overcoming limitations imposed by the peak system memory and bus bandwidths, which are often lower than the peak network bandwidth in a gigabit network. By rethinking the host I/O subsystem, and moving processing power closer to network I/O, it is possible to bypass the main system memory and bus altogether. This approach is especially well-suited for continuous media data (e.g., video, audio, etc.), where the processing is usually of a very specialized nature (e.g., signal processing, compression, encryption, etc.) and needs to be carried out in real-time. For most other types of data, and in particular for discrete media data, it is still important to provide for a high speed pathway to the main system memory.

† This work was supported in part by the Advanced Research Projects Agency (ARPA), the National Science Foundation (NSF), and an industrial consortium of Ascom Timeplex, Bell Communications Research, Bell Northern Research, Goldstar Information and Communication, NTT, Italtel SIT, NEC America, Southwestern Bell, SynOptics Communications, and Tektronix.

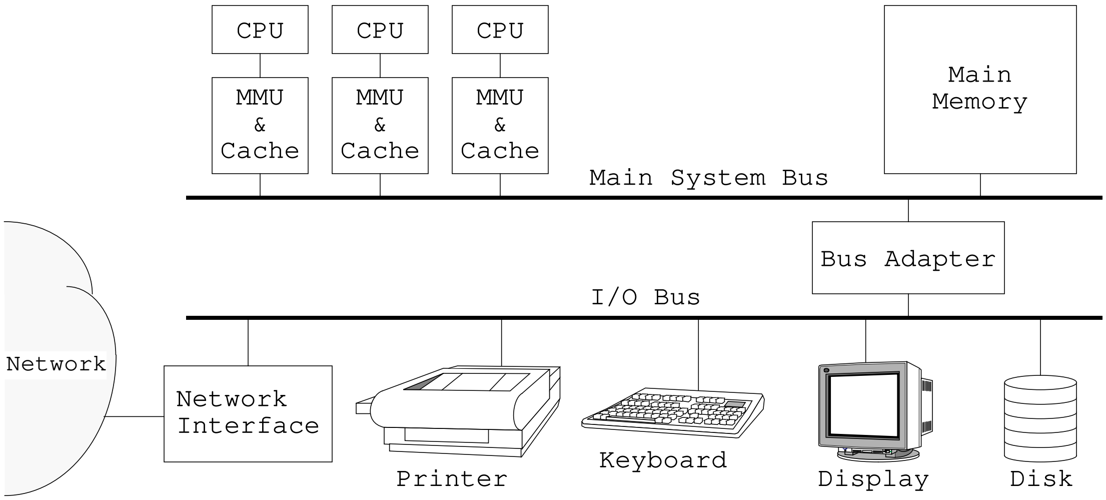

Figure 1: A Typical Workstation Architecture.

In this paper, we describe the design of an I/O subsystem based on an ATM interconnect; our design is targeted toward high performance multimedia workstations and servers. A prototype of this interconnect, which will allow for a sustained data transfer rate of 622 Mbps simultaneously in both directions (into and out from any I/O device or the network), will be built as part of a bigger project (funded by ARPA) involving the setup of a gigabit local ATM testbed at Washington University, and the demonstration of some example applications. We already have an installed and operational 155 Mbps packet switched network developed locally [3], and we are currently in the process of building an ATM switching system [9] that will support data rates of 622 Mbps (SONET) and 2.4 Gbps (G-link) per port. Our primary objective in designing the interconnect has been to support applications that can effectively utilize these data rates.

Background and Motivation

The current state-of-the-art in workstation architectures (see Figure 1) typically includes: support for multiple processors; a main system bus that interfaces to the processors and main memory, and provides support for snooping based cache coherency protocols; a main memory that is organized as a single DRAM bank; and an I/O bus to which most of the I/O devices, including the network interface, are connected. This type of architecture is not well suited to the class of multimedia applications—the primary limitations arise from the bottlenecks imposed by the two buses[1], and from the fact that the effective memory bandwidth is usually no more than 500 Mbps. To compound the problem, protocol stack implementations in existing operating systems require data to be copied multiple times (once from the network interface to kernel space, and then from kernel space to the application's address space). Each such copy step increases the number of times the data has to traverse the system bus, and also increases the number of times memory is accessed. Furthermore, an application may need to access the data, possibly more than once, in order to perform some computation on the data, or just to copy it on a disk. Add to this the fact that multiple processors working on multiple media streams have to share the same bus, and it becomes apparent why many of the interfaces designed to support high speed networks fail to deliver even a small fraction of the network bandwidth to end applications. Clearly, there is a need to revise the host communication architecture and I/O subsystem, if we are to be able to meet the demands imposed by our choice of target applications.

[1] The effective bandwidth of the fastest buses in use today is no longer much higher than the network bandwidth (of emerging networks). With multiple bus traversals, the available bandwidth is typically much lower than the network bandwidth. Typical high performance workstation effective bus bandwidths range from 500 Mbps to 2 Gbps.

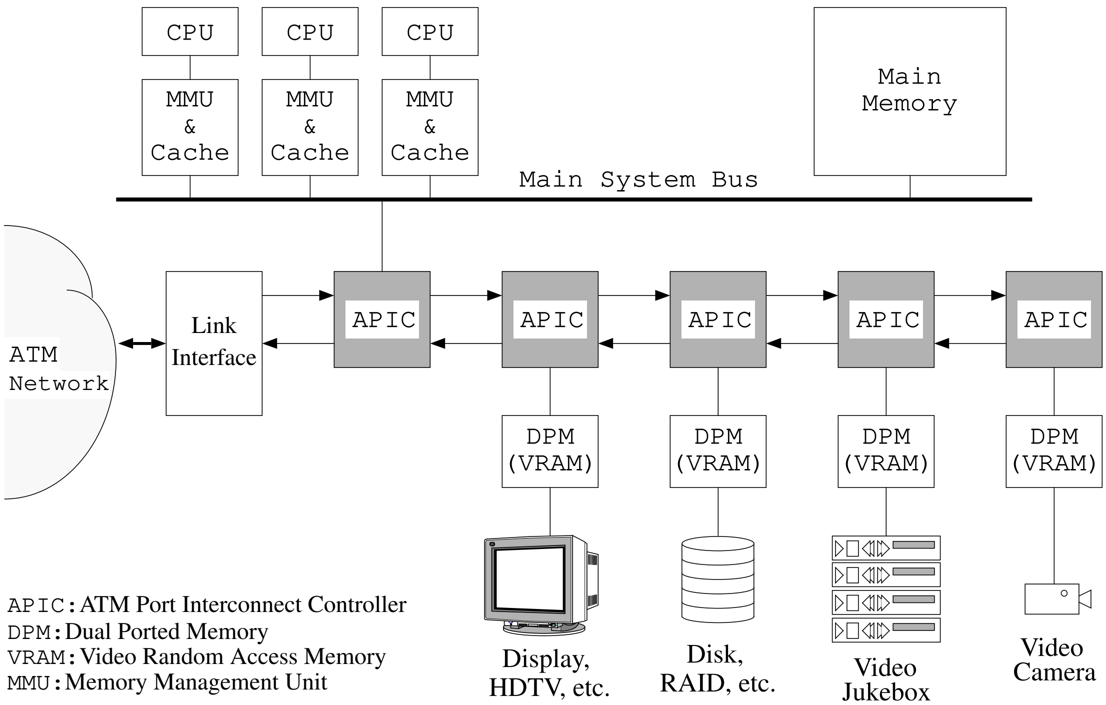

Figure 2: Host-Network Interface Architecture.

Proposed Architecture

A high level schematic of our proposed architecture is shown in Figure 2. At the heart of this architecture is a daisy-chained interconnect comprising of a number of *ATM Port Interconnect Controller* (APIC) chips. Individual chips interface either directly to the main system bus, or through a dual ported memory (in this paper, we shall concentrate only on dual ported memories that are implemented as VRAMs) to an I/O device One of the APIC chips on the interconnect is connected to the external ATM network through a link interface. ATM cells arriving from the network pass from one APIC to another until they reach an APIC chip that is directly connected to the destination device (or to the system bus). Similarly, data originating from a source device is passed as an ATM cell stream from the APIC directly connected to that device, down the APIC interconnect towards the link interface, and finally out to the network. It is also possible for two local devices to communicate over the APIC interconnect, without cells ever having to leave the desk area. Thus, for example, a video cell stream originating at a camera could traverse the interconnect in order to reach the local display device. The APIC which directly interfaces to the system bus can be used for traffic originating from, or destined for, the system's main memory. All other traffic movement happens only over the APIC interconnect, thus sparing the main system bus of this burden. In our prototype implementation, each APIC will interface to another APIC over two 622 Mbps links (one is used for incoming traffic, and the other for outgoing traffic). The link to the ATM network switch port also operates at 622 Mbps. Thus, our prototype interface will be capable of supporting an aggregate sustained bidirectional data rate in excess of a gigabit per second.

It is easy to see that the APIC interconnect bears a strong resemblance to the traditional notion of an I/O bus. There are several advantages to using I/O buses in addition to CPU-memory buses: I/O buses can be long, can have many types of devices connected to them, and normally follow a bus standard. In contrast, CPU-memory buses are short, cater to very few different types of devices, can change frequently to track changes in technology, and are designed to optimize the performance of the CPU, cache, and memory. It is instructive to contrast our approach with the I/O bus approach with a view to gathering an understanding about why the APIC interconnect can outperform I/O buses in most cases:

- While most of the early I/O buses were asynchronous, many of the new high performance I/O buses are of the synchronous variety (examples include Intel's PCI bus [6], Sun's SBUS, and variants such as the Apple Ring). The APIC interconnect is asynchronous (at the cell level). This allows for high performance and at the same time, the interconnect can span a greater area (i.e., it can be longer). Furthermore, it is easier to support devices of widely varying characteristics if the I/O interconnect is asynchronous.
- Almost all existing I/O buses are circuit-switched. In contrast, the APIC interconnect is packet switched. Packet switching has been gaining popularity in interconnects used within hosts[7]; however, its use has predominantly been confined to the processor-memory interconnect, where it allows for a greater effective bandwidth through the use of split transactions. The primary advantage of using packet switching in our design stems from the fact that we utilize the same packets on the APIC interconnect as those used in the external network. In effect, we are "extending" the external network into the end-host. This allows for devices to be able to stream ATM cells out into the network (or vice-versa) with minimal latency and very little extra hardware.
- An I/O bus is a bus, and therefore it suffers from all the usual drawbacks associated with bus-based architectures; in particular, buses usually scale poorly (to large numbers of devices), they cannot be very long without also becoming unacceptably slow, and only one device can be using the bus at any point of time. Our daisy-chained interconnect topology does not suffer from any of these drawbacks.

By using a standard physical interconnection between APIC chips, an additional advantage of our interconnect becomes apparent—we can make the interconnect longer than would otherwise be possible. For example, our prototype design uses the UTOPIA[2] standard [1] for connecting two APIC chips within a single "box". In going from one box to another within the desk-area, we would switch to an optical medium (622 Mbps SONET). An off-the-shelf chipset can be used to translate from electronic (UTOPIA) to optical (SONET) medium, and vice-versa. The link interface to the ATM network can also be built using this chipset; thus, our entire interconnect architecture requires that we design only one ASIC (the APIC). Finally, note that our architecture does not preclude the use of an I/O bus; in fact, it would be simpler and more cost effective to use an I/O bus for slow devices (such as keyboards, printers, etc.).

The APIC chip, which is central to our design, behaves like a 3×3 switch, two of whose ports are connected to two other APIC chips (data arrives and leaves these ports as ATM cells), and the third port interfaces to external memory (possibly over a shared bus). One of the jobs of the APIC is to take specified data from the memory, create an ATM cell stream using that data, and forward the cell stream to a downstream APIC. In the reverse direction, the APIC is responsible for doing a VPI/VCI virtual channel lookup on every arriving cell to determine if it is destined for the local device. If so, the cell payload is placed in memory, else it is forwarded to a downstream APIC. The APIC is also responsible for implementing full AAL-5 segmentation and reassembly; thus, frames up to 64 Kbytes in length can be reassembled. Entire frames are not reassembled within the APIC itself; instead, the APIC gradually builds a frame in the external memory by writing the payload from a few cells at a time. Finally, the APIC is also responsible for *pacing* outgoing cells. This serves two purposes: first, it ensures that the APIC's local device does not get an unfair share of the ATM interconnect bandwidth, and second, it prevents bursty sources from causing network congestion or from swamping the receiver with a burst that cannot be handled (by the receiver).

---

[2] UTOPIA, which is an acronym for *Universal Test & Operations PHY Interface for ATM*, is a standard proposed by the ATM forum for the data path interface between the ATM layer (performing segmentation and reassembly), and the physical (PHY) layer components of an ATM subsystem. It can support a wide range of speeds: from sub-100 Mbps to 155 Mbps (with an 8-bit wide data path), and up to 622 Mbps (with a 16-bit wide data path).

Configuration of APIC and Control of APIC and Local Devices

Individual APIC chips in the interconnect can be remotely configured at the time of initialization by using special control cells. The operating system running on the host CPU(s) issues these control cells by providing appropriate commands to the APIC that directly interfaces to the main system bus. Once an APIC has been initialized, its control can be taken over by a local control processor that is responsible for controlling both the APIC as well as the locally connected device (the control processors are not shown in Figure 2; in a later section, we shall describe how the APIC interfaces with the DPM, control processor, and I/O device). There are several advantages to following this approach of dedicating a control processor for each device:

- Data can "flow" from network-to-device (or vice-versa), as well as from device-to-device, without any intervention from the main host CPU.
- It is easier to provide real-time support for devices if they are controlled by a dedicated processor.
- The operating system running on the host CPU(s) can be relieved of having to service interrupts and perform context switches every time new data arrives, or needs to be sent from a local device.

Note that we are not advocating that one processor should be dedicated to every I/O device in the machine. We would use control processors only for devices that reside on the APIC interconnect, and since these devices would typically be very high bandwidth (and possibly expensive) devices, it is justifiable to have a low-cost processor available for their control. For all other devices, the method of interfacing to the host would be no different from what it is today—they would either reside on the main system bus, or on a separate I/O bus. Also, note that a control processor need not be very powerful, since it does not have to touch the data

Due to space restrictions for this extended abstract, we have chosen not to elaborate on the software issues that arise in the control of the various devices. In brief, the operating system running on the main system CPUs communicates with the operating system on the embedded control processors through the use of an IPC mechanism that is built on top of the APIC interconnect architecture. We shall include a more detailed description of these software related issues in the final version of this paper.

Interfacing Devices to the APIC

In Figure 3, we show how a continuous media I/O device can be interfaced to an APIC through the use of a VRAM. The bus transceiver is merely an analog buffer that serves as a directional switch; depending on its state, it allows data on the stream bus to go onto the local bus, or vice-versa, or it could be open (no connection). When the APIC is ready to place some data in the VRAM, it issues a read transfer cycle on a row address to the VRAM controller over the Local Bus (the transceiver is configured to allow the address to pass from the Stream Bus to the Local Bus). This results in data from the specified row of the VRAM being read into the serial access shift register within the VRAM. Following this, the APIC can use the Stream Bus to clock data into the shift register over the serial port of the VRAM. The transceiver would behave like an open switch during this phase of the transfer. Once the serial transfer is completed, the APIC would issue a write transfer cycle with the row address so that the contents of the shift register get written back into the appropriate row of the (random access memory in the) VRAM. The transceiver would again be in the "Stream bus to Local bus" state during this brief transaction. The device can also read data from the VRAM over the Stream Bus, using a similar sequence of operations. Note that the Local Bus interfaces to the random access port of the VRAM, thereby enabling the control processor to read or write the contents of the VRAM; this is required in order for the processor to perform control related actions. Also, the control processor can issue commands to the APIC over the Stream Bus by placing the transceiver in a state where it allows data movement from the Local bus to the Stream bus. Note that in most cases, the continuous media data need not move over the Local Bus, so this bus does not need to achieve high performance.

The theoretical peak transfer rate into the serial port of a typical 40 MHz VRAM over a 64-bit Stream Bus is 2.56 Gbps. If we take into account the fact that each data unit would have to traverse the Stream Bus twice, and allowing for overhead due to bus arbitration, start-up, and transfer control, there is enough band-

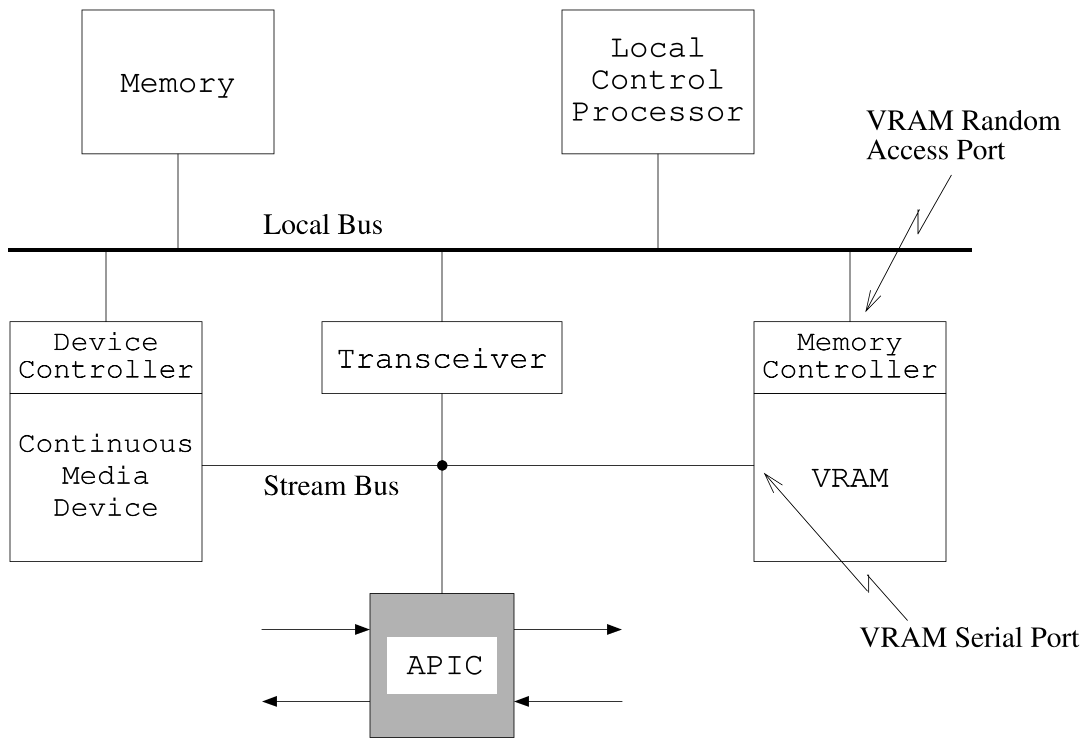

Figure 3: Interfacing APIC to an I/O Device.

width left over to easily accommodate the entire 622 Mbps link rate. If the device is full duplex and can simultaneously source and sink up to 622 Mbps, then the design can be slightly modified to use a triple-ported VRAM (one having two serial ports and one random access port). This would ensure that data movement between the APIC and the VRAM does not interfere with data movement between the VRAM and the device.

Media Processing

We now address the problem of how to do media processing on data originating from or destined for a device on the interconnect. We have three approaches for tackling this problem. One simple way is to relegate all media-specific processing to the local control processor of a device. Clearly, this approach cannot be used when the data rate is very high; the limiting factor would be the bandwidth of the random access port of the VRAM (see Figure 3). A second approach is to have a special media processor residing as a separate "device" on the interconnect. This method would work especially well when the processing device is an encryption/decryption engine, for example. Data destined for the network could be made to "pass through" the encryption device on its way to the link interface. Similarly, data arriving from the network could be made to pass through this device so the data can be decrypted before being forwarded to the destination I/O device. A third way to support media processing in our design is to embed a signal processor on the data path between a continuous media device and the VRAM (see Figure 3). This method suffers from the drawback that the processor cannot be easily shared by multiple devices. However, in some cases this would be the approach of choice, such as, for example, when the media processor is a hardware compression/decompression engine. This ensures that uncompressed data would not have to traverse the interconnect, which is important because the data rate of the uncompressed stream might be a significant fraction of, or more than, the maximum bandwidth of the interconnect.

Richer ATM Interconnects

An architecture in which a generalized packet switched interconnect is used to connect processors, memories, and devices has widely come to be known as a “desk area network” (DAN). The concept of a DAN was first introduced by David Tennenhouse [8]. We believe that desk area networks hold promise for the future of computer architecture. Clearly, the APIC interconnect architecture itself falls into the class of DAN-based architectures. If we generalize the APIC so that it becomes a full-fledged ATM switch that can interface simultaneously over different switch ports to a number of devices (including CPUs and memory modules), then we arrive at a more futuristic DAN-based architecture. While a system based on a such an architecture would likely deliver higher performance than an APIC based architecture, it does not come without some drawbacks, at least in the short term. Among these are hardware costs—the APIC solution comes out ahead both in terms of the number of chips that would need to be designed, as well as in the number of chips that would actually be used in an implementation of comparable size. Furthermore, the APIC scheme provides an incremental cost solution: only as many APICs need to be used as there are devices in the host. The APIC interconnect can also span long distances by using optical fiber for connecting widely separated APIC chips.

Conclusions

We presented the design of a high performance interconnect that serves both as a high speed replacement for the I/O bus in a multimedia workstation or server, as well as an efficient host-network interface. A detailed description of the internal design of the APIC chip (which forms the heart of the interconnect), as well as a description of the APIC’s interface to the processor-memory bus and its interaction with the operating system and network protocols is described in [4]. In the same paper, we have shown that it is possible to achieve a zero copy interface to the system’s main memory through the use of the APIC and by utilizing page-remapping techniques within the operating system. With zero copy semantics, parts of a transport level packet can be in transit on the network even before the entire packet has been read from memory; this allows us to achieve very high application throughput.

We are currently in the process of refining our design. We hope to complete the design before the summer of 1994, and we shall subsequently undertake the prototyping of the APIC chip using a VHDL tool. We shall also be prototyping APIC interconnect based interfaces for a few target workstations and servers that will include I/O devices such as video cameras, high resolution displays, video storage, and possibly a video wall. This work is part of a bigger effort sponsored by ARPA that is aimed at the design and deployment of a gigabit local ATM testbed at Washington University. In addition to prototyping, we are also planning to use simulation and analytical techniques to evaluate several other trade-offs that were mentioned in this paper.

References

[1] "An ATM PHY Data Path Interface: Draft Version 0.5", an ATM Forum Document Draft, May 1993.

[2] "Broadband Switching: architectures, protocols, design, and analysis", edited by Chris Dhas, Vijaya K. Konangi, and M. Sreetharan, *IEEE Computer Society Press*, 1991.

[3] Cox Jr., Jerome R.; Gaddis, M.; and Turner, Jonathan S.; "Project Zeus: Design of a Broadband Network and its Application on a University Campus", *IEEE Network*, Vol. 7 No. 2, March 1993, pp. 20-30.

[4] Dittia, Zubin D.; Cox Jr., Jerome R.; and Parulkar, Guru M; "Catching Up With the Networks: Host I/O at Gigabit Rates", Submitted for review to *ACM SIGCOMM '94*.

[5] Partridge, Craig; "Gigabit Networking", *Addison Wesley*, 1994.

[6] "Peripheral Component Interconnect (PCI), Revision 1.0 Specification", Intel Corporation, June 22, 1992.

[7] "SPARCcenter 2000 Architecture and Implementation", Technical White Paper—Draft Edition of 11/18/92, Sun Microsystems, Inc., November 1992.

[8] Tennenhouse, D.; "Telemedia, Networks, and Systems: Research Report", 1993.

[9] Turner, Jonathan S.; "An Optimal Non-Blocking Multicast Virtual Circuit Switch", *Technical Report WUCS-93-30*, Department of Computer Science, Washington University in St. Louis, MO, 1993.