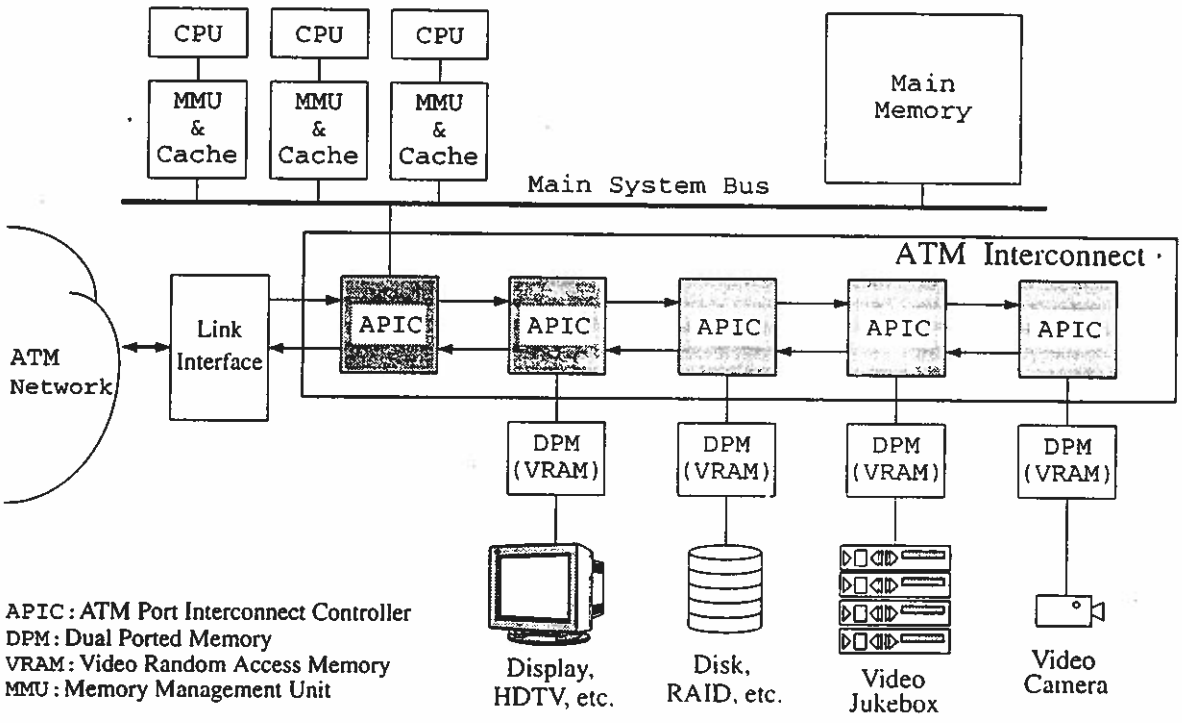
Research in Information Networking

August 1994



Washington University's Gbps ATM Desk Area Network

Computer and Communications Research Center
Departments of Computer Science and Electrical Engineering
Washington University, St. Louis

Research in Information Networking

Information networking refers to the combination of computers, networks and distributed systems software to support applications that enable and enhance collaborative work, direct interpersonal communication, remote access to information and real-time presentation of information from a variety of sources. Over the next decade, we expect such systems to become central to the infrastructure of our increasingly information-driven society. This report summarizes research activities in the Computer and Communications Research Center, and the Departments of Computer Science and Electrical Engineering at Washington University that seek to promote the creation of effective information networking systems and develop the fundamental understanding needed to determine how such systems can be best designed and used.

In the last decade, computing and communication systems have become increasingly intertwined and interdependent. New high speed network technologies and the emerging, but still primitive, multimedia capabilities of personal computers and workstations, will vastly accelerate this trend. *Information networking* can be a powerful tool for collaboration, for gathering and analyzing information and then using it to synthesize new knowledge. Because these new systems represent a major departure from traditional computing, developing useful, cost-effective solutions requires substantial rethinking of how we design and implement computer and communication systems at all levels.

Our work spans four major areas and encompasses a variety of specific topics. The major areas are *scalable gigabit networks*, *advanced computing platforms*, *multimedia systems software* and *distributed multimedia applications*. Our research program seeks to maintain a strong systems focus combining both experimental and theoretical research. We are committed to the position that successful engineering research requires an intimate knowledge of the practical issues involved in building complex systems, as well as strong analytical capabilities. We find that theoretical research, if disconnected from practical concerns, can become so inwardly directed that it loses focus and drifts into arcane and ultimately sterile pursuits. On the other hand, experimental work that is uninformed by a deeper understanding of fundamental issues, can have only limited and short term value. The activities of our four research areas, come together through various experimental systems projects. These projects provide a testbed for our research ideas and a departure point for theoretical studies aimed at broadening our understanding and preparing the ground for successive generations of information networking systems. They also provide a concrete demonstration of our research ideas, facilitating their effective transfer into commercial use.

Our research program enjoys support from the following sponsors.

Advanced Research Projects Agency
National Science Foundation
Ascom Timeplex
Bell Communications Research
Bell Northern Research
Goldstar Information and Communications
Italtel SIT
Nippon Electric Corporation
Nippon Telephone and Telegraph
Southwestern Bell
SynOptics Communications
Tektronix

We thank all our sponsors for their collaboration and support.

Personnel

Affiliated Faculty
- G. James Blaine
- Andreas Bovopoulos
- Jerome R. Cox, Jr.
- Paul Min
- Guru Parulkar
- W. David Richard
- Jonathan Turner
- George Varghese

Collaborators
- Ron Cytron
- Martin Dubetz
- Mark Franklin
- Mark Frisse
- Kenneth J. Goldman
- Dan Kimura
- James McNally
- Michael Miller
- Catalin Roman
- Fred Rosenberger
- John Schnase
- Doug Schmidt
- David States

Research Staff
- Tom Chaney
- Pierre Costa
- Ken Cox
- John DeHart
- Maynard Engebretson
- Andy Fingerhut
- Margaret Flucke
- Brian Gottlieb
- Craig Horn
- Mike Richards
- Randy Richards
- Hossein Saidi

Doctoral Students
- Ehab Al-Shaer
- Akira Arutaki
- Millind Buddhikot
- Charles Cranor
- Zubin Dittia
- Andy Fingerhut
- Raman Gopalakrishnan
- Adiseshu Hari
- Saied Hosseini-Khayat
- Seyyed Mahdavian
- Nader Mirfakhraei
- Christos Papadopoulos
- Ammar Rayes
- Hossein Saidi
- Einir Valdimarsson
- Dakang Wu
- Peter Yan
- Ellen Witte Zegura

Masters Students
- Bob Akl
- Jim Barta
- Xiaolin Bi
- Andres Blazquez
- Alex Chandra
- Joy Chatterjee
- Adam Costello
- Apostolos Dailianas
- Ian Flanigan
- Pete Flugstad
- Jason Fritts
- Rex Hill
- Rob Jackson
- Ken Krieger
- Anurag Maunder
- Jyoti Parwatikar
- Chan Park
- Yang Qian
- M. Shreedhar
- Chris White

Visitors
- Peter Cheung (GoldStar)
- Sam Y. Choi (GoldStar)
- Chul Hee Kang (ETRI)
- Taeck Geun Kwon (GoldStar)
- Yong Geun Jeong (GoldStar)

Executive Summary

This report covers the period from January 1, 1993 through June 30, 1994. From this year forward, our progress reports will cover periods corresponding to academic years (July through June), rather than calendar years.

The last 18 months have seen a number of important accomplishments. In March 1993, SynOptics Communications announced a new line of ATM switch products and demonstrated them at the InterOp trade show. These products use intellectual property and technology licensed from Washington University and created with assistance from our faculty and research staff. In August, SynOptics began shipping products to customers and we have moved forward to install these systems within our own campus network, for applications ranging from electronic radiology, to multimedia conferencing and digital libraries. We also completed a platform-independent multimedia communications device, called the Multimedia Explorer (MMX), which provides two-way audio and video over ATM and has been licensed to a local start-up company (STS Technologies), which is now selling the MMX commercially.

Some major new efforts got under way during this reporting period. First, Paul Min, with support from ARPA, Goldstar and ETRI has started a project on multichannel switching that will lead to the creation of a prototype ATM switch based on the multichannel switching concept, which has been developed by Dr. Min and doctoral student Hossein Saidi. Multichannel switching seeks to overcome the bandwidth mismatch between fiber optic transmission and electronic switching, by integrating inverse multiplexing into the basic structure of an ATM switching system. Min expects to complete a first prototype within the coming year.

Professors Cox, Franklin, Parulkar and Turner obtained a contract from ARPA for research on gigabit network technology that will include the creation of a gigabit ATM testbed with link speeds of up to 2.4 Gb/s. The testbed will incorporate a novel, multipoint switch architecture that provides optimal scaling properties, allowing large systems to be built at a much lower cost than was previously possible. It will also use a new ATM Port Interconnect Chip (APIC) to create host-network interfaces that provide a closer coupling between network and application, allowing data to be delivered to and from applications at up to 600 Mb/s. Since this initiative began in September of 1993, we have expanded staff, developed a detailed plan for the gigabit technology components, including a very detailed system architecture specification for the gigabit switch, have acquired and installed new VLSI design tools, established a close working relationship with a commercial foundry and initiated design work on the three new integrated circuits needed for the switching system. Updating our VLSI design tools was a time-consuming but important step, since we had been relying previously on a fairly old set of tools developed at other universities in the early eighties. When we went to transfer technology from earlier research projects to industrial organizations, we found that our use of tools that were incompatible with current industry standards was a significant impediment to technology transfer. Consequently, we have moved to a standard commercial tool set, based on VHDL and cell libraries provided and supported by a commercial foundry.

Several major funding initiatives were undertaken during this reporting period. Washington University is a member of a consortium led by Bellcore that obtained a Technology Reinvestment Program contract from the federal government. This contract will lead to the installation of a 10 Gb/s SONET ring in the St. Louis metropolitan area. Washington University will demonstrate advanced ATM applications over the SONET

ring, based on our gigabit networking technology. Other participants in the program include AT&T, Rockwell, Southwestern Bell and Tektronix. It is expected to start in September, 1994.

We have received a major grant from the National Science Foundation on applications of gigabit networking to scientific grand challenges. This is a five year grant, starting in fall 1994, which will support applications in biology and neurosciences.

The Department of Computer Science has obtained a Research Infrastructure grant from NSF to substantially expand and upgrade our campus ATM network, in support of research in networking, distributed multimedia, remote visualization and scientific imaging. This expansion will take place over a three year period, using commercially available ATM switches and workstation adaptors.

Several smaller grants and contracts were also obtained this year. Guru Parulkar led a successful effort to obtain a contract from the Air Force's Rome Development Laboratory for network and protocol development in support of applications in distributed simulation. George Varghese obtained grants from NSF for work on self-stabilizing protocols and for high performance protocol processing. Jon Turner obtained a grant from NSF for approximately optimal network design.

Ken Goldman, who is becoming an important collaborator, also obtained a new grant from NSF to support work on programming environments for multimedia applications based on the I/O abstraction concept. Together with Jerry Cox, Guru Parulkar and Jon Turner, Dr. Goldman has submitted a proposal to ARPA to support a more comprehensive realization of the I/O abstraction concept, including a high performance implementation based on some novel enhancements to our gigabit network technology.

In addition to our federal support, we continue to have strong relationships with a number of industrial organizations. This provides a major component of our research funding and just as importantly, provides us the opportunity to interact with organizations who are engaged in delivering products and services to end-users and can help us understand both the needs that must be met and the possibilities that technological developments are making available. These relationships also provide us a pathway by which our research results can influence industry practice, either through direct technology transfer or through the exchange of ideas. We are now in the process of reshaping our Industrial Partnership Program to reflect the broadening scope of our activities and to move toward a more consistent set of relationships.

As always, there have been changes in the students, faculty and staff associated with our projects. In September of 1993 George Varghese joined the computer science faculty after receiving his doctorate at MIT in distributed computing and networking. George also spent nine years with Digital Equipment Corporation, where he participated in a variety of advanced research and development projects, including the development of the DECNET protocol suite, the DEC LAN bridge products and the FDDI Gigaswitch. This summer, Doug Schmidt joined the faculty of computer science, after receiving his doctorate from the University of California, Irvine. Dr. Schmidt's research has centered on parallel protocol processing for high speed network applications, and he is expected to become a key collaborator.

At the end of 1993, Andreas Bovopoulos left Washington University to take a position at Chipcom. Three staff members (Mike Gaddis, Rick Bubenik and Pete Flugstad) left Washington University in spring 1993 to help start the new St. Louis office of Ascom Timeplex. Pierre Costa also left the university to take a new position where he will be introducing advanced network and information technology in a major hospital system centered in Peoria, Illinois. Ellen Zegura, after completing her doctorate, took a faculty position in the School of Computer Science at

Georgia Tech. Ammar Rayes returned to Bellcore, following completion of his doctorate and both Andy Fingerhut and Hossein Saidi have taken positions on the research staff at Washington University after obtaining their doctorates. In addition to Andy and Hossein, Maynard Engebretson, Craig Horn and Brian Gottlieb all joined the research staff. Students receiving masters degrees in the last eighteen months include Xiaolin Bi, Apostolos Dailianas, Pete Flugstad, Jason Fritts, Brian Gottlieb, Craig Horn, Rex Hill and Ken Krieger. Rex and Ken have gone onto SynOptics. New students include Ehab Al-Shaer, Bob Akl, Jim Barta, Joy Chatterjee, Adam Costello, Rob Jackson, Jyoti Parwatikar, Chan Park, Yang Qian and Chris White.

We continue to publicize our results extensively through technical journals, conferences and other means. This reporting period has been a very productive one, with seventeen journal articles, twenty three conference papers, thirty four technical reports, three MS theses, four doctoral theses and three patents granted. Faculty, students and staff associated with Washington University presented eight papers at Infocom in 1993 and 1994. The series of short articles that follow provide concise summaries of our activities. More information can be found in the articles and reports listed in the references.

Design and Analysis of Switching Systems

Optimal Multicast Virtual Circuit Switching

Jonathan Turner

Multicast virtual circuit networks support communication paths from a sender to an arbitrary number of receivers. Multicast virtual circuits induce a tree in a network connecting a sender to one or more receivers. Switching systems participating in the virtual circuit replicate received cells using *virtual circuit identifiers* in the cell headers to access control information stored in the switching system's internal control tables, then use this information to identify the outputs the cells are to be sent to and relabel the copies before forwarding them on to other switching systems.

If a switch has n inputs and outputs and each output supports up to m virtual circuits, one can describe any collection of multicast virtual circuits with mn words of memory. One simply provides for each (output,VCI) pair, the identity of the (input,VCI) pair from which it is to receive cells. Unfortunately, this method of defining a set of multicast connections is not particularly helpful in switching, as it does not give one a way to go from an (input,VCI) pair to the desired list of (output,VCI) pairs. Existing virtual circuit switch architectures describe multicast virtual circuits in different ways, which while suitable for switching, use far more than mn words of memory. The broadcast packet switch [49], for example, requires $mn^2/2$ words of memory under worst-case conditions (although various refinements can reduce this to $O(mn^{3/2})$ or less, giving acceptable complexity for systems with up to a few hundred ports). Moreover, the time required to update a multicast connection grows with the size of the connection. Other architectures require even greater amounts of memory. For example, Lee's multicast switching system [30] requires $mn^3/2$ words of memory under worst-case conditions.

We have devised a novel multicast switch architecture with $O(n \log n)$ hardware complexity that is nonblocking, in the sense that it is always possible to accommodate a new multicast connection or augment an existing one, so long as the required bandwidth is available at the external links, and which requires $< 2mn$ words of memory for routing cells in multicast virtual circuits. Moreover, the overhead for establishing or modifying a multicast connection is independent of the size of the connection or the switching network. This architecture provides the first example of a multicast switch architecture that is efficient enough to allow cost-effective scaling to thousands of ports, while maintaining nonblocking operation.

The architecture is based on the concept of *cell recycling* and is illustrated in Figure 1. To implement a multicast connection, a binary tree is constructed with the source switch port at its root and the destination switch ports at its leaves. Internal nodes represent switch ports acting as relay points, which accept cells from the switch but then recycle them back into the switch after relabeling the cells with a destination pair identifying the next two switch ports they are to be sent to. There are many possibilities for constructing the switching network. A Beneš network in which the switches in the first half of the network distribute cells dynamically in order to balance the load, and in which local buffers are used to resolve contention, provides the lowest cost solution. With this choice, the load placed on any of the switching network's internal links is at most equal to the load on the most heavily loaded external port. In other words, there is no collection of virtual circuits that can be handled by the external links that cannot also be handled by the network. That is, this network is nonblocking. Other switching networks, suitably extended to provide the copy-by-two function, can also be used in the recycling architecture.

The lower part of Figure 1 details the hardware associated with each port of the switching system. Given a virtual circuit identifier, obtained from a cell's header, the *Virtual*

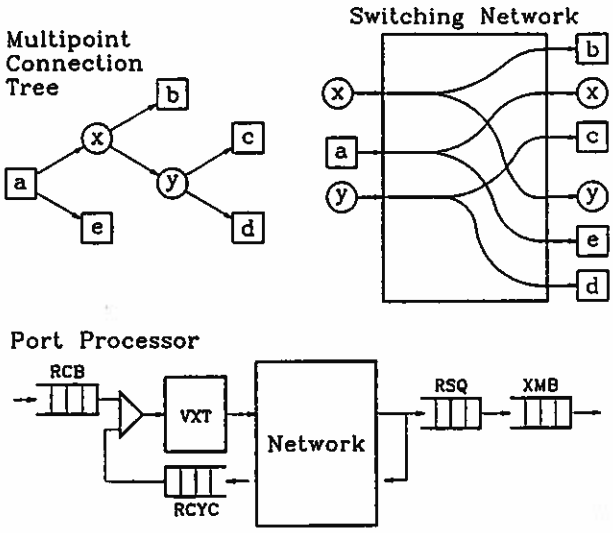


Figure 1: Multicasting by Recycling Cells

Circuit Translation Table (VXT) provides two (output,VCI) pairs that are added to the cell header plus two additional bits that indicate, for each pair, whether it is to be recirculated another time, or not. The *Receive Buffer* (RCB) holds cells received from the input link that are waiting to enter the switching network, while the *Transmit Buffer* (XMB) holds cells waiting to be transmitted on the outgoing link.

Because networks that perform dynamic load distribution may deliver cells in a different order than that by which they enter, the ports are typically augmented with a resequencing buffer to restore the proper ordering on output [50]. The simplest resequencer implementations measure the time that cells spend in transit through the network and delay cells that pass through quickly in a resequencing buffer for a long enough period to ensure that cells that are delayed an unusually long period of time have a chance to catch up. In the recycling architecture, the resequencer also ensures proper ordering of cells during additions and deletions to multicast connections. The resequencing buffer is labeled RSQ in Figure 1.

Recycling cells does require that some portion of the switching network's capacity be devoted to carrying the recycled traffic. It's easy to show that the amount of capacity that is consumed by recycling traffic is less than the traffic leaving the system, meaning that the amount of extra capacity that must be added to the system, relative to a point-to-point system is not too large. Moreover, the amount of added capacity can be engineered, based on the amount of multipoint traffic that is expected. It can also be adjusted in an operational system to accommodate changes in user behavior. See [57, 59] for further details.

Design of a Gigabit Switching System

Tom Chaney, Peter Chung, Zubin Dittia, Andy Fingerhut, Margaret Flucke, Brian Gottlieb, Craig Horn, Saied Hosseini, Randy Richards, Jonathan Turner

A key element of our networking research program is the creation of a gigabit ATM testbed comprising a set of six gigabit switching systems with link speeds of 600 Mb/s, 1.2 Gb/s and 2.4 Gb/s. This testbed will include three switches at the Washington University hilltop campus, two at the Medical School Campus and one at Barnes West, a remote hospital. The Barnes West site will be linked to the rest of the network through a 10 Gb/s SONET ring that is to be installed as part of an ARPA-funded Technology Reinvestment Program (TRP) contract, involving AT&T Bell Laboratories, Bellcore, Rockwell, Tektronix and Washington University. This is illustrated in Figure 2.

The switching systems that are being designed for this testbed will be based on the recycling architecture described earlier. Three custom integrated circuits are being created to implement the architecture, a *Switch Element* (SE) chip, an *Input Port Processor* (IPP) and an *Output Port Processor* (OPP). Four of the SE chips are used to construct an eight port switch element with a central, shared cell buffer and 36 bit wide data paths. At a clock rate of 121 MHz, this provides a cell processing rate which is (4/3) times the cell rate on a 2.4 Gb/s link, which is sufficient to keep up with fully loaded links. The SE chips are being designed to allow them to be used in multistage networks with up to 32,768 ports, giving a maximum system throughput as high as 78 terabits per second. The number of chips needed for systems with 8^k ports is $(k - 1/2)$ per port, so the largest system ($k = 5$) requires 4.5 switch chips per port. This is far superior to competing architectures.

The IPP and OPP chips, together implement a complete port processor for the gigabit switching system. The IPP provides synchronization, cell buffering, reformatting and virtual path/circuit translation functions. The OPP provides cell resequencing (to accommodate differing delays in the switching network), cell buffering and congestion control mechanisms. The two chips are joined by a recycling data path, used both for recycling cells in multipoint virtual circuits, and for control operations, such as testing the switching network, reading and writing entries in the virtual path/circuit translation table and for accessing hardware control and status registers.

The chip set supports a simple set of flow control mechanisms. These include separate buffering of ABR traffic from CBR and VBR, with CBR and VBR getting higher priority access to the link. In addition, an enhanced AAL 5 block discard mechanism is being implemented on selected virtual circuits. Using this mechanism, when an output queue becomes congested a timer is set and for any virtual circuit that loses a cell while the timer is running, cell discarding is continued until the end-of-frame following the timer expiration. Simulation studies have shown that this simple scheme delivers throughputs above 70% even for large overloads (4:1) and switch buffers that are only slightly larger than the AAL 5 frame size.

The chip set also supports a range-copy mechanism which can improve the efficiency of handling multipoint virtual circuits in certain circumstances; and it provides an upstream discard feature to support general multipoint virtual circuits in which there are both multiple transmitters as well as multiple receivers.

One of the most challenging aspects of building a high speed switching system operating at clock rates above 100 MHz is controlling the timing of signals with sufficient precision that they can be passed reliably from chip to chip. The chip set being designed for the project uses a deskewing circuit at the input port of each chip that inserts delay into an incoming signal path in order to compensate for differing wiring delays between chips. The circuit can

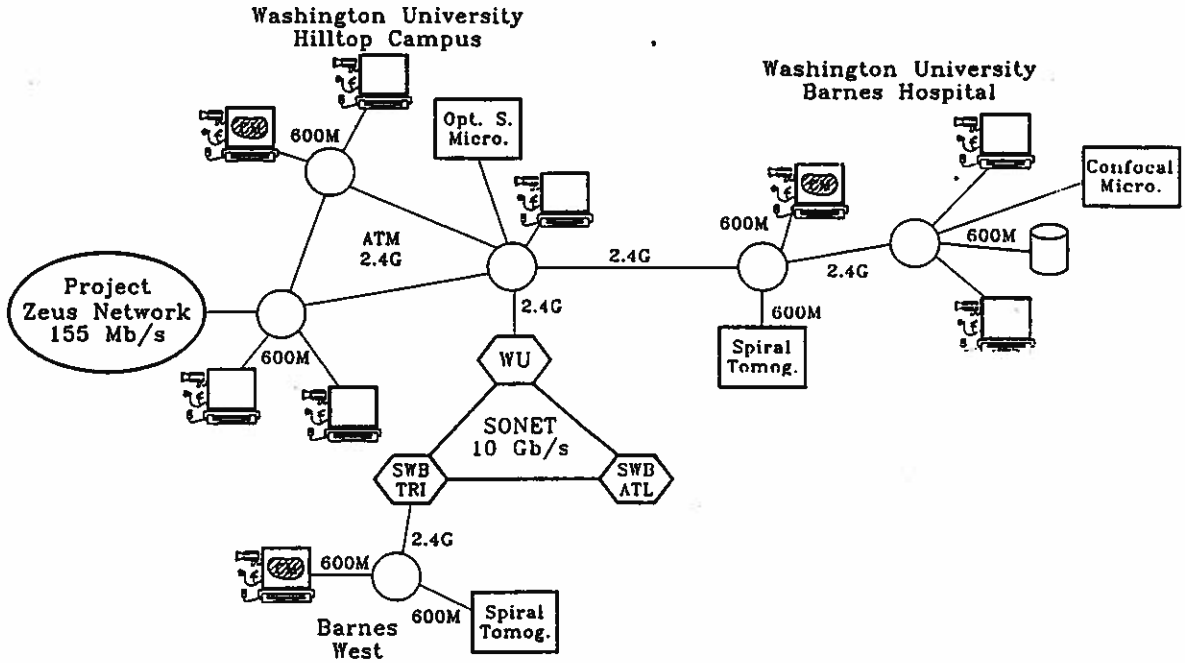


Figure 2: Gigabit Testbed Configuration

accommodate up to 3 clock ticks of skew (about 24 nanoseconds), providing sufficient margin for large systems requiring many circuit boards. The area used by the deskewing circuit is roughly equal to the area of the associated I/O pads. It is being designed using only elements from a vendor-supplied standard cell library and is robust across process and temperature variations (which can vary by a factor of four).

The switches used in the testbed network will be small prototypes with a single stage switching network and eight ports. Six of the eight ports will use Hewlett-Packard G-link transmission chips and will be individually configurable to operate at either 600 Mb/s or 1.2 Mb/s (matching the cell rate of the corresponding SONET formats). They will be configured with 820 nm optical transceivers with a range of 500 meters, but pin-compatible 1320 nm transceivers are also available at a higher cost, allowing selected links to be operated over longer distances. The two remaining ports on the switch will be implemented on replaceable line cards, allowing flexibility in configuring switches to meet different needs. Initially, we are planning a line card with a single 150 Mb/s SONET interface (for control in early configurations) and a line card which implements a 2.4 Gb/s link over two parallel optical fibers, using a pair of the G-link chip sets. We are also planning a line card that will include a 600 Mb/s SONET interface, possibly combined with one or more 150 Mb/s interfaces. As part of the TRP project, a 2.4 Gb/s SONET interface will be developed for the switch. The entire switch is to be packaged in a desktop enclosure, about the same size as a workstation system unit.

Current estimates indicate that the component costs for the prototype units will be between $25,000 and $30,000, with about 40% of the cost in the transmission components, 40% in the ATM switching components and the remainder in circuit boards, power supply, enclosure, fans and other miscellaneous parts. When configured with two ports at 2.4 Gb/s and three ports each at 600 Mb/s and 1.2 Gb/s, this corresponds to a cost per 150 Mb/s of capacity of between $370 and $440. Larger quantity production could cut these costs by a factor of two or more.

As indicated above, the chip set being

developed for the gigabit switch is being designed to support the configuration of larger systems as well. We have done several design exercises to explore the utility of the chip set in configuring different kinds of systems. One configuration supports 72 ports at 150 Mb/s each plus six ports at 600 Mb/s. This makes an ideal workgroup switch and has a cost which is primarily determined by the SONET interface chips and the optical devices. If configured with twisted pair interfaces on the 150 Mb/s ports, the parts needed for such a system would cost less than $15,000 or about $150 per 150 Mb/s of port capacity (assuming volume production). A larger configuration with 576 ports at 150 Mb/s plus 48 at 600 Mb/s has a similar cost per 150 Mb/s of capacity. Another interesting configuration is a switch with 96 ports at 1.2 Gb/s, using the G-link interfaces. We estimate that such a system would cost about $90,000 to construct in moderate volumes, or under $120 per 150 Mb/s of capacity.

Complete details of the system architecture can be found in [14]. Details of the three chips can be found in [7, 8, 21].

Topics in ATM Switching

Jonathan Turner

We continue to explore a variety of topics relating to the design of of ATM switching systems in order to improve performance and add functionality. This section reports on several aspects that have received attention during this reporting period.

Congestion Control for Unconstrained Bursty Traffic. In the last few years, it has become clear that ATM networks must be prepared to handle data traffic which is bursty and highly unpredictable. For most data applications (particularly interactive sessions), there is simply no way to predict average usage and while the flow of data to the network can be constrained by a Usage Parameter Control (UPC) mechanism, this unnecessarily limits users when the network is lightly loaded. Adaptive techniques that allow the users to change their rate of transmission in response to network loading can give subjectively more satisfactory performance, but make the flow of data to the network less predictable.

To provide good performance in the presence of unpredictable bursty traffic, the network must queue the bursty traffic separately and give it lower priority access to transmission resources. In addition, it should provide mechanisms to ensure that losses at the cell level do not cause excess loss rates at the packet level. That is, the switching systems should attempt to discard on a packet or frame basis, rather than a cell basis. In this work, we have examined three simple mechanisms for preserving the integrity of user-level frames, delineated by AAL 5 framing protocols and carried out some initial simulation studies to assess their performance. The simplest is AAL 5 frame level discarding. In this mechanism, when a cell is lost from an AAL 5 frame due to buffer overflow, the remaining cells in the frame are discarded to the end of the frame. We've carried out simulations in which virtual circuits with transmission rates of 50 Mb/s and 150 Mb/s share a link with a capacity of 600 Mb/s. As the number of virtual circuits of each type is increased beyond the capacity of the link, the effective throughput (the fraction of the link's capacity used to carry complete frames) drops from 100% to about 50% with a 2:1 overload and to about 25% with a 4:1 overload. In these simulations the frames were 8 Kbytes long and the cell buffer had a capacity of 256 cells or about 12 Kbytes.

The second mechanism we studied is a simple refinement of frame level discarding in which a timer is set whenever the link enters a congested state. Any virtual circuit that loses a cell due to buffer overflow while the timer is running is marked for discarding until the end of frame following the timer expiration. This approach dramatically improves the effectiveness of the frame discarding mechanism. With a timer value of 1.5 ms, it achieves a throughput of over 70% even with a 4:1 overload.

A third mechanism that we've studied is full-frame discarding in which discarding is triggered when the link buffer occupancy rises above a threshold. In this scheme, whenever the first cell of a packet is received, the packet is rejected if the buffer is above threshold; otherwise it is accepted. With a buffer threshold of 128 cells, this scheme performs fairly well, but not quite as well as the frame discard with timer. However, if larger buffers are used, we can show that this scheme can achieve 100% efficiency during overload, under worst-case assumptions. In particular, we have shown that if the buffer size is equal to the sum of the frame sizes for the set of virtual circuits, then you can achieve 100% throughput, no matter how large the overload. We also believe that the scheme can be refined to provide similar performance with much smaller buffers (buffer size equal to two frames).

Adaptive Resequencing to Reduce Switch Latency. Switching systems that route cells dynamically to balance the traffic can get cells

out of order, as a result of differing delays in the switching network. Resequencing buffers in the output port processor of a switch can restore the proper ordering by buffering cells and sending them out in they order that they *entered* the switching network, rather than the order in which they exited. This can be accomplished using a timestamp assigned to cells when they enter the network. To ensure that cells are not misordered, the resequencer must hold cells for long enough to ensure that other cells delayed in the network have a chance to catch up. The delay imposed by the resequencer (which is determined by its depth) is chosen to be equal to the largest delay expected in the network under heavy load conditions. When the system is lightly loaded, this added delay may be unnecessary, and for certain applications it can be a limiting factor in the performance.

Adaptive resequencing adjusts the delay imposed by the resequencer in order to minimize the delay during periods when the system is lightly loaded. While a variety of schemes are possible, we have devised one particularly simple scheme that appears to be easy to implement, quite robust and easily configured. The key parameter is a *short-term skew bound*, which represents the maximum amount by which two cells sent consecutively from the same input port to the same output port can be misordered at the output. Our adaptive resequencer adds only this much delay to the delay imposed by the switching network itself, significantly reducing latency when the switch is lightly loaded (we estimate it can reduce the resequencer latency to 30-50% of the delay in the non-adaptive case). Adaptive resequencing can also reduce the required size of the resequencing buffer in the output port processor, especially in larger system configurations, and we believe it will lead to more robust performance under heavy load conditions.

Using Range-Copy Mechanism in Recycling Switch. The recycling switch architecture described above, uses a copy-by-two mechanism in the switching network to allow configuration of arbitrary multipoint virtual circuits. This can result in multipoint trees with up to $\log_2 F$ recycling passes, where F is the maximum fanout that the system can support. In some circumstances, it may be desirable to reduce the number of passes. This can be accomplished using a *range-copy* mechanism, in which the two output port numbers in a cell header are interpreted not as the two destinations to which the cell is to be sent, but as upper and lower bounds of a range of outputs, all of which are to receive copies of the cell. Since the cell only has room for two virtual path/circuit identifiers, the different outputs do not receive unique VPI/VCI values, limiting the application of the range copy mechanism somewhat. However, in certain circumstances, it can be used to good effect.

One such situation is for a switch that is configured for video distribution. In such a system, popular video sources can be assigned a fixed VPI/VCI combination, so the constraint imposed by the range-copy mechanism is not a problem in this case. When range-copy is used in conjunction with a high speed switching network in which each switch port supports a large number of lower speed access ports, we get a very efficient video distribution arrangement. For example, consider a recycling switch using the chip set described above, with a five stage switching network providing 512 high speed ports. Assume 64 of the high speed ports are used for trunks to other systems and 64 are reserved for recycling of general multipoint connections. That leaves 384 high speed ports free and if we multiplex 16 ports at 150 Mb/s on each on these high speed ports, we can support a total of over 6,000 users in this system.

Notice that 100 video feeds at 20 Mb/s each consume less bandwidth than one of the 64 trunks. We can use the range-copy mechanism to deliver the most popular feeds to the 384 ports supporting line interfaces and then copy these feeds to individual line interfaces at the demultiplexors. If 50 feeds are distributed this

way, that still leaves half the capacity of the 384 line interface ports available for general purpose use. If these 50 feeds account for 90% of video distribution in the switch, then 6 of the 64 recycling ports provide sufficient capacity to handle all remaining video distribution, assuming that there is an average of one video feed being used by each user. This leaves the remaining recycling ports available for other uses. Without recycling, video distribution would require ten times this many recycling ports, doubling the amount of system capacity that must be dedicated to recycling.

The range-copy mechanism can also be used to support general connections. We have devised a general algorithm for adding and removing endpoints to multipoint connections that use the range-copy mechanism. While the algorithms are somewhat more complex than the ones for the copy-by-two mechanism, they can be practical to implement and can cut the depth of the trees needed to support a multipoint connection (and hence the maximum delay experienced) by a factor of two in large configurations.

Switch-Level Support for Reliable Multicast Messages. Many distributed computing applications require the ability to send copies of a single message reliably to a large number of recipients. Ideally, the amount of work that a sender needs to do to accomplish this does not depend on the number of recipients. While the multicast mechanism in ATM networks allows efficient delivery of messages to many recipients, it does not guarantee delivery, making it necessary for transport protocols to process acknowledgements and retransmissions much as they would if multiple point-to-point transmissions were used in the first place.

We have devised an extension to the ATM multicast mechanism that reduces the amount of work that must be done by the transport protocol software in order to handle multicast message transmission. This mechanism has a particularly simple implementation on the recycling switch architecture. The basic idea of the mechanism is for switching systems to keep track of acknowledgements sent by recipients of a multicast message and deliver an acknowledgement up the tree only when all recipients in the subtree have sent their acknowledgements. This means that the sender of a multicast message need only process a single acknowledgement, no matter how many recipients there are to the message. If some recipients fail to receive the message when it is first transmitted, the same mechanism can be used to deliver retransmissions to only those recipients that failed to receive the initial transmission. In the recycling architecture, we can support 48 concurrent messages on a single virtual circuit by using two table entries for virtual circuits that implement the reliable multicast message feature. This allows applications to pipeline a window of multicast messages over a single virtual circuit, making it possible to achieve high throughput even over connections with fairly long latencies.

The same basic mechanism can be used to implement a reliable many-to-one virtual circuit. When combined with a one-to-many virtual circuit, this can be used to support applications requiring reliable many-to-many communication in which all participants receive messages in the same order.

Performance Evaluation of Switching Networks

Einir Valdimarsson

ATM switching systems are expected to be the central component of the emerging information superhighway. Understanding the performance of these systems under various traffic conditions is important for system architects, network designers and network operators. This research concerns the development of better methods to evaluate these systems using both analysis and simulation.

Queueing Analysis of Multistage Switching Networks with Non-Uniform Traffic. Most analyses of multistage switching networks assume a uniform random traffic model, in which the addresses of each cell are assumed to be selected from a uniform distribution. This fails to model situations in which the distribution of cell destinations is uneven. While some authors have considered non-uniform distributions, this has generally been in the context of binary switch elements with limited buffering and these analyses do not apply to networks with large switch elements or shared buffering. We have devised a novel queueing model for multistage networks with shared buffering that can be applied to networks constructed from large switch elements. To capture the uneven traffic distributions, we use d coupled Markov chains to model each switch element. Each chain focuses on a single output and has a two-dimensional state, with one state representing the number of cells in the switch element as a whole, and the other representing the number of cells destined for the specific output. To write the state transition probabilities for these Markov chains, we consider the effect of arriving and departing cells on the cell occupancy at both each particular output and the switch element as a whole. The analysis yields accurate results for single stage networks and for networks with large switch elements. For multistage networks with small switch elements and inter-stage flow control, the model overestimates network throughput, because it neglects inter-stage dependencies. This is a common characteristic of multistage switching network models.

Queueing Analysis of Copy Networks. Multistage queueing analyses are generally limited to routing networks, while networks that perform cell copying to support multicast communication have thus far been studied, only through simulation. We have devised queueing models for multistage copy networks with buffering and inter-stage flow control. To describe the copying processes in switch elements, the state representation of the Markov chains which model the switch elements must include the number of multicast cells destined to each subset of the outputs. This expands the state space considerably, forcing us to restrict attention in this work to copy networks with binary switch elements. We have devised versions of the model that apply to four types of switch elements; an input buffered switch element, an output buffered switch element and two variations of a shared buffer switch element. In the first of these, cells which are not to be copied and can be sent to either output are randomly assigned to one of the two outputs when they arrive. In the second, these 'distribution' cells are assigned to outputs on departure. The switch element models can be used to model networks that carry a mixture of different cell types, including copy cells, routing cells and distribution cells, with an arbitrary distribution among the types. The models also support different distributions among the inputs and outputs, making them applicable to networks with uneven load distributions.

Transient Performance of Switching Networks with Bursty Traffic. We have created a general purpose switching system simulation tool that supports interactive network construction and visualization, allowing switching system architects to easily experiment with a wide variety of design alternatives and traffic characteristics. We've used this tool to study

the transient queueing behavior of a point-to-point ATM switching system when subjected to different traffic conditions. The particular system we studied was a three stage Beneš network using shared buffer switch elements and dynamic routing (individual cells are assigned independent paths through the switching network). In one set of experiments, we studied the time it took the system to recover after an overload period. Specifically, we placed a load of 100% on all the switching system's output links for an extended period of time, then reduced the load to some background load level and observed the time required for the buffers to drop to their long-term level. We studied both systems with flow control between the last stage of the switch and the output buffer and systems without flow control. There was a striking difference between these two types of systems as the internal data rate of the switching network was increased relative to the external link rate. In the systems without output buffer flow control, congestion persisted in the switching network for a significant time period following the removal of the overload condition, while in the systems without output buffer flow control, the overload condition had a much more limited effect on the switching network. These systems closely approximated the performance of an ideal output buffered switching network, even when subjected to highly bursty traffic.

Another experiment we carried out was to study the effect of coincident bursts at a single output on the performance the switching system as a whole. Here also, there was a significant difference between systems with and without output buffer flow control. The systems with output buffer flow control generally remained congested for a longer period of time and the congestion was observed not only at the overloaded output, but in the switching network itself. For large bursts (500 cells), increasing the switching network speed in these cases had remarkably little impact on the time to recover from the transient overload. For systems without output buffer flow control, we observed that a speed advantage brought a major improvement in performance, reducing both the impact of the overload on the switching network and its time duration. In these cases, the coincident bursts are flowing directly through to the output buffers and while large parts of these bursts may be lost, there is no impact on other outputs, allowing the overall system performance to be greatly improved.

Comparison of Switching Network Performance. Another attractive feature of our switching system simulation tool is that it makes it very straightforward to construct side-by-side comparisons of alternative switching network architectures. We have used this to do a careful comparison of a variety of ATM switch architectures under both Bernoulli and bursty traffic conditions. The systems studied include the Knockout switch [62], the tandem banyan switch [48], Bellcore's Sunshine switch [22], Lee's hybrid of the Sunshine and Knockout [31] and a Beneš network with shared buffer switch elements. Since the Knockout switch provides ideal queueing performance (although less than ideal scalability), we used it as a standard of comparison from a performance standpoint. In particular, for each of the traffic types studied, we determined how much output port buffering was needed in the Knockout switch to achieve acceptable cell loss rates at output link occupancies of 80%. We then studied each of the competing architectures and attempted to identify the least-cost configuration of each of the competing architectures that could yield comparable performance for the same amount of output buffering. In general, the architectural characteristic that had the greatest effect on queueing performance (particularly for bursty traffic) was the bandwidth into the output buffer. We found that for the bursty traffic situations considered, we needed this bandwidth to be two to three times the external link rate in order to give comparable performance.

More details on this work can be found in [60].

Analysis of Nonblocking Copy Networks with Shared Buffering

A. Chandra and P.S. Min

This research investigates the performance of copy networks employing the shared buffering scheme.

A copy network can be viewed abstractly as a functional block with N input channels and N output channels. At each input channel, a fixed length packet may appear with probability p per time slot, independently of events in any other time slot, and independently of events in any other input channel.

Packets enter the copy network with copy fanout request values that are distributed independently from packet to packet. An arriving packet has associated with it the fanout request value, F, which corresponds to the number of copies to be made by the copy network. The fanout request values for different packets are assumed to be independent and identically distributed (i.i.d.) random variables with distribution $\mathcal{F}$ according to which k copies are requested with probability f_k ($\sum_{i=1}^{N} f_k = 1$).

The class of copy networks analyzed is internally nonblocking in the sense that any packet selected to be copied in a given time slot can reach the output channels of the copy network without further buffering or loss. Packets are replicated in the copy network and each output channel allows at most one of the replicated copies to depart from the copy network in a time slot. Thus no more than N copies of the packets which are present can leave in the same time slot.

Shared buffering is achieved by providing a common buffer module of size b that is accessible from all input channels. In each time slot, packets in the buffer module are considered first for copying (in a FIFO manner) and then the new packets arriving at the input channels are considered such that no input channel is given any priority over others.

Let the random variable R be the fanout request value to be satisfied in the beginning of a time slot; during a certain time slot, if at least one of the available packets is not copied, the R value to be satisfied in the beginning of the following time slot is set equal to the fanout request value of the first packet that needs be copied. If all available packets are copied, we set $R = 0$. We note that R is associated with the packet that is at the head position of the buffer module prior to new arrivals. The fanout request value of a newly arriving packet, which happens to arrive to an empty buffer module and thus assumes the head position, is represented by F.

Figure 3 shows $\mathcal{R}$ recorded from the simulated data under different $\mathcal{F}$, wherein for the purpose of comparing to $\mathcal{F}$, $\mathcal{R}$ is normalized by $(1 - P(R = 0))$. We observed during our numerical study that the distinguishing shapes in $\mathcal{R}$ in Figure 3 have a significant impact on the packet loss probability, the delay, and the throughput. Without explicit consideration of the differences between $\mathcal{R}$ and $\mathcal{F}$ (i.e., assuming that R and F are identically distributed), the numerical results deviate significantly from the simulated values.

We use the technique of tagged Markov chains to derive the stationary distributions for the occupancy of the buffers. Based on these stationary distributions, delay, throughput, and packet loss probability are calculated as critical performance measures of the copy network. Figure 4 compares the results of the analyses with the simulated data with $\mathcal{F}$ set to be truncated geometric. We see from the figures that our analyses are accurate for all values of p ($= p \cdot E(F)$).

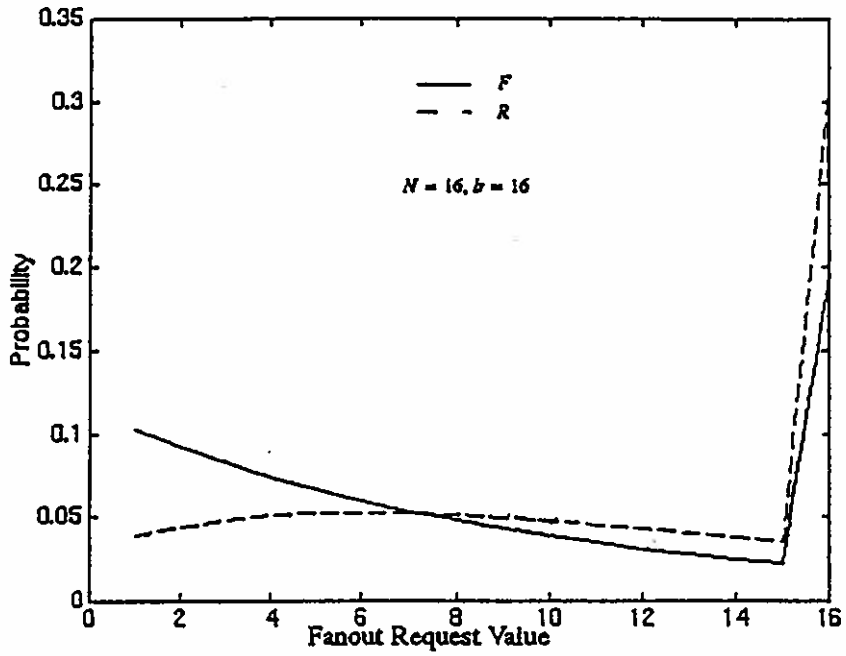


Figure 3: Fanout Request Value ($\mathcal{F}$: Truncated Geometric, $E(F) = 8$)

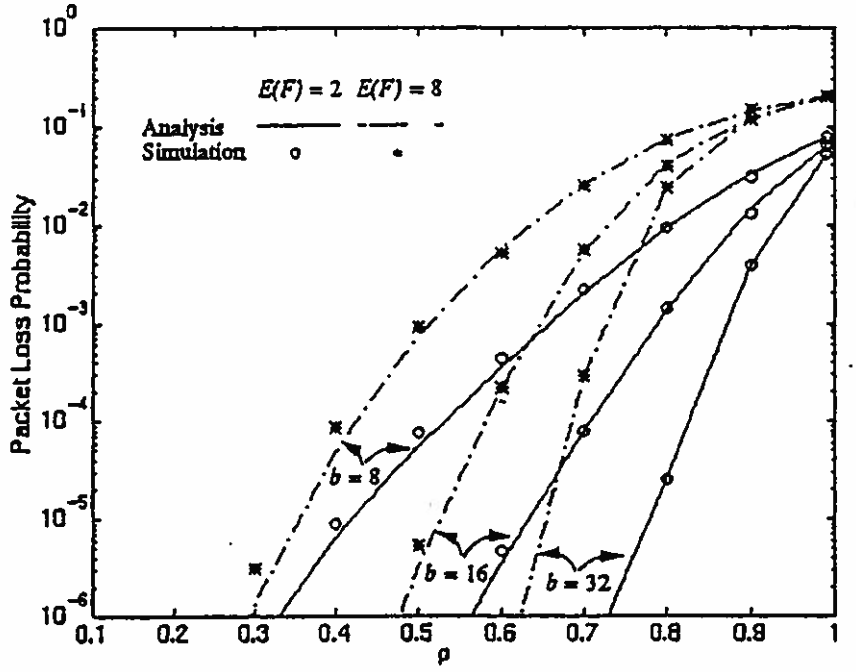


Figure 4: Packet Loss Probability of Shared Buffering with $N = 16$

Washington University Multi-Channel Switch

P. Yan, H. Saidi, and P.S. Min

Washington University Multi-Channel Switch (WUMCS) is a promising alternative to the traditional ATM switches, in which cells are allowed to traverse over multiple channels defined as a channel group while maintaining the session identities. WUMCS is based on a particular nonblocking condition derived for Flip networks [47], and is able to support partitioning of input/output channels into an arbitrary number of channel groups of arbitrary sizes. Using one Flip Network recursively a number of times based on the number of channel groups, WUMCS is an efficient architecture in the sense that the cross point complexity can approach $O(N\ (log_2 N)^2)$ for N inputs.

Prototyping activities are under way for a version of the WUMCS architecture with 16 input channels, each running at 155Mbs [5] [6]. It is expected that the first working prototype will be completed in early 1995. Below, we summarize the distinguishing features of WUMCS. Specifics of these features are discussed in detail in [47].

- *Multi-Channel Switching*
  WUMCS has the ability to group channels arbitrarily and can support super-rate switching of any bandwidth. Partitioning into channel groups is flexible and does not burden the switch with additional hardware and control complexities.

- *In-Order Cell Sequencing*
  In multi-channel switching, cells belonging to a session can traverse multiple channels, and thus they can experience a widely differing amount of delay and congestion. Maintaining the time ordering among them becomes a challenging issue. WUMCS guarantees in-order cell sequencing without any resequencing mechanism.

- *Simplicity*
  Multi-channel switching is facilitated without increasing the complexity of the switch. For example, the core of the 16-input prototype of WUMCS is a CMOS IC constructed from a standard technology (i.e, 0.8 μ, standard die and package sizes), and this IC performs the entire multicasting or routing function in a bit sliced manner.

- *Nonblocking and Bufferless Fabric*
  The internal fabric of WUMCS is nonblocking and bufferless. A switch fabric is nonblocking if every permutation of inputs which does not cause output conflicts can be realized. A bufferless fabric implies that at each switching element, all input cells are transferred to the output without delay.

- *Multicasting*
  WUMCS supports multipoint connections.

- *Multi-Rate Switching*
  It is desired that a single switch supports bit pipes of different bandwidths. WUMCS allows channel group sizes to be defined arbitrarily, and thus any combination of bit rates, each of which is an integer-multiple of 155Mbs, can be switched simultaneously in a single fabric.

- *Multiple Performance Requirements*
  It is important to perform switching according to the differing performance requirements of services, especially when the network is congested by heavy traffic. WUMCS can accommodate the characteristics of services as part of managing capacity of the network, with a moderate amount of increase in the switching complexity.

- *Fairness*
  Sessions established for different input-output pairs may experience different delay or throughput. WUMCS can ensure fairness at the cost of moderately increasing switching complexity.

The overall architecture of WUMCS is presented in Figure 5 as a two phase structure involving the Multicasting Phase followed by the Routing Phase. A shared buffering method is employed to resolve contentions at both phases.

In order to maintain the time sequence integrity of cells, we introduce the concept of the relative ordering, which is defined among the channels belonging to the same channel group. (The relative ordering among channels is monotonic with respect to the channel indices.) We then equate at each time slot, the time ordering among the cells belonging to the same session with the relative ordering of the channels they utilize and maintain this relative ordering at each switching entity throughout the network. As a result, within a channel group, a lower indexed link has the higher time ordering than do higher indexed links at each time slot.

Operations in Figure 5 are as follows: A cell arriving at the input channel is processed by the Port Processor (PP) and the required number of copies are determined. The cell then enters the $(N + R_M)$-input Flip network (e.g., $N = 16$ and $R_M = 48$ for the prototype) wherein active cells are concentrated at the upper portion of the output with the relative ordering maintained among them.

After passing through the Flip network, the active cells enter the Address Encoder over a compact interval and it is determined whether or not the fan-out request of the cells can be accommodated by the Copy Network, starting from the top. It can be seen that this maintains the time ordering among the cells; if the fan-out request of a cell is rejected, all fan-out requests of subsequent cells belonging to that session would be rejected during the same time slot due to their relative ordering.

Cells are then routed to the second Flip network where they are either passed to the Banyan Copy Network, or routed to the recirculation paths being fed back to the R_M inputs of the Flip network reserved for recirculation. The recirculated cells are tried again at the next time slot at which time their priorities have been increased due to their positions, i.e., they appear at the top, at the input of the Flip network; they are given higher priorities than the cells entering the switch for the first time since the Address Encoder examines the fan-out of the cells beginning from the top. Maintaining the relative ordering among the recirculated cells is accomplished by twisting recirculation paths before connecting back to the input.

Next consider the Routing Phase in Figure 5 where R_R of the $(N + R_R)$ inputs available in the first Flip network are reserved for recirculation. (e.g., $N = 16$ and $R_R = 48$ for the prototype.) Cells targeted for various channel groups enter at the input and iterate recursively through the first Flip network. After completing the required number of recursions, they appear at the output of the Flip network according to a Gray coded sequence of channel group addresses.

Cells enter the next Flip network where the contentions for output are identified and excess cells are separated from the rest while maintaining the relative ordering. The cells passed to the Banyan Router can be routed to the proper destinations without further possibility of contention since the Banyan Router is nonblocking with respect to compact inputs that are destined for monotone ascending output addresses. On the other hand, the excess cells are recirculated for attempts at the next time slot at which time, they are given priority due to the higher relative ordering of the recirculation paths. Such a mechanism ensures that the in-order cell sequencing is maintained.

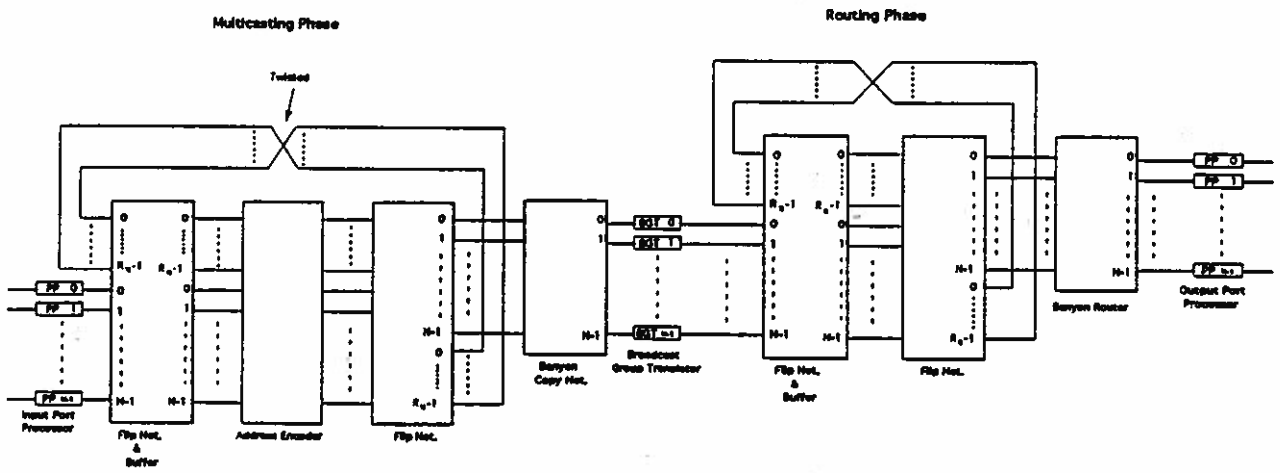


Figure 5: WUMCS Architecture

Performance Benefits of Multi-Channel Switching

H. Saidi and P.S. Min

The goal of this research is to verify the performance benefits of channel grouping for nonblocking ATM switches. Assume that the switching networks have N input channels and N output channels which are labeled between 0 and $N-1$. Arrival processes at different input channels are independent, and a cell appearing at an input channel is equally likely destined for all channel groups at the output. The output channels are divided into t equal sized channel groups, each with $m = \frac{N}{t}$ channels where t is a positive integer resulting in an integer value for m. We label the t channel groups between 0 and $t-1$ for convenience. Let p be the probability that there is a cell in a time slot at each input channel, and hence $\frac{p}{t}$ is the probability that there is an arriving cell destined for a specific channel group.

For the purpose of illustration, consider a nonblocking ATM switch wherein a buffer module is located at every input channel. Cells are served in a first-in-first-out (FIFO) manner from the buffer modules. In time slot n, let q_n^i be the number of buffer modules with the HOL cell destined for channel group i which is blocked in time slot $n-1$ due to output contention. $\sum_{i=0}^{t-1} q_n^i$ is the number of buffer modules with HOL blocking in time slot $n-1$, and F_n, which denotes the number of cells that move to the head of a buffer module in the beginning of time slot n, becomes,

$$F_n = N - \sum_{i=0}^{t-1} q_n^i.$$

F_n is also the number of cells that are routed through in time slot $n-1$.

Let A_n^i be the number of cells, amongst the F_n cells that freshly assumed the head position in a buffer module, that are destined for channel group i. Then,

$$Pr(A_n^i = k) = \binom{F_n}{k} \left(\frac{m}{N}\right)^k \left(1 - \frac{m}{N}\right)^{F_n - k} \quad (1)$$

The throughput, ρ, equals the probability that a cell departs from a certain buffer module and routes through the network in a time slot, and from the symmetry among all channel groups,

$$\begin{aligned} \rho &= \frac{F}{N} \\ &= 1 - \frac{1}{N}\sum_{i=0}^{t-1} q^i \\ &= 1 - \frac{q^i}{m} \end{aligned}$$

where F and q^i are the average values for F_n and q_n^i respectively.

Let λ be the average number of cells that are at the head of a buffer module, that are destined for a certain channel group, and that are actually transmitted to the output. Again from the symmetry among the channel groups each consisting of m channels, $\lambda = m \cdot \rho$. Recalling that there is a cell in each of the N buffer modules and that these cells are equally likely destined among all channel groups, the average number of cells at the head that are destined for any channel group is m. Thus,

$$q^i = m - \lambda$$

Now considering for time slot n, the dynamics of each buffer module is described by,

$$q_{n+1}^i = max(q_n^i + A_n^i - m\ ,\ 0). \quad (2)$$

To determine the stationary distribution for q_n^i, we need to construct the balance equations governed by (2). Noting $Pr(A_n^i = j)$ is equal for all n and i due to the i.i.d. nature of the arrival process and the symmetry among the channel groups, we drop the indices n and i in what follows: A corresponds to the number of cells that freshly assume the head position of a buffer module in the beginning of a time slot and that are destined for a certain channel group.

With p_j defined as $Pr(q_n^i = j)$,

$$p_0 = \sum_{k=0}^{m} p_k \cdot Pr(A \leq m-k)$$

$$p_j = \sum_{k=0}^{m+j} p_k \cdot Pr(A = m+j-k) \quad \text{for} \quad 1 \leq j \leq N-m. \quad (3)$$

Let the z-transform for $p_j = Pr(q^i = j)$, denoted as $P(z)$. The numerator of $P(z)$ is a polynomial of degree m, and thus has m complex roots. Since p_j is a probability, $P(z)$ is bounded and analytical on and inside the unit circle. Consequently, every root of the denominator in $P(z)$ on or inside the unit circle must be canceled by the numerator so they should also be the roots of the numerator. Thus $P(z)$ can be written as,

$$P(z) = \frac{K \cdot \Pi_{s=1}^{m}(z - z_s)}{z^m - A(z)} \quad (4)$$

where

$$A(z) = \sum_{j=0}^{N-m} Pr(A = j) \cdot z^j.$$

z_s is the s^{th} root of the denominator on or inside the unit circle and K is a certain constant to be determined shortly. Since $A(1) = \sum_{j=0}^{N-m} Pr(A = j) = 1$, the denominator vanishes at $z = 1$ and there is a root for the numerator at $z = 1$, which is labeled as the m^{th} root for convenience, i.e., $z_m = 1$.

To determine K, we use the fact that the limit of $P(z)$ as $z \to 1$ is one. Since $z = 1$ is the root of both the numerator and the denominator, we use the L'Hôpital's law, and,

$$lim_{z \to 1} P(z) = \frac{K(1-z_1)(1-z_2)\cdots(1-z_{m-1})}{m - A'(z)_{|z=1}} = 1.$$

Noting from (1) that A_n^i (and A) approaches Poisson for large values of N (and thus large values of F and F_n),

$$A(z) = e^{-\lambda(1-z)}, \quad (5)$$

and therefore,

$$K = \frac{m - \lambda}{(1-z_1)(1-z_2)\cdots(1-z_{m-1})}. \quad (6)$$

Thus, once z_s is known for $s = 1, 2, \ldots, m-1$, (6) determines K.

Using the properties of the z-transform we know that

$$q^i = \sum_{j=0}^{N-m} j \cdot p_j = \frac{d}{dz} P(z)_{|z=1}.$$

Since both the numerator and the denominator of the derivative of $P(z)$ vanish at $z = 1$, the L'Hôpital's law needs be applied twice to $P(z)$ and as a result, q^i is determined as,

$$q^i = \sum_{s=1}^{m-1} \frac{1}{1-z_s} - \frac{m(m-1) - \lambda^2}{2(m-\lambda)}. \quad (7)$$

From (2), we substitute $m - \lambda$ for q^i in (7), and solve for λ. The resulting value is valid if $\lambda < m$ for the stability of the buffer occupancy which is assumed implicitly. The throughput, ρ, then can be determined from $\lambda = m \cdot \rho$.

The procedure of determining ρ can be summarized in the following manner:

- **Step 1.** Assume an arbitrary value for λ such that $0 < \lambda < m$.
- **Step 2.** Determine z_s, $s = 1, 2, \ldots, m-1$ by determining the roots of $z^m - A(z)$ from (5).
- **Step 3.** Determine the new value for λ from

$$\sum_{s=1}^{m-1} \frac{1}{1-z_s} - \frac{m(m-1) - \lambda^2}{2(m-\lambda)} = m - \lambda. \quad (8)$$

If the new value is close to the previous one, stop. Otherwise, go to **Step 2.**

For $m = 1$, (8) simplifies to

$$1 - \lambda = \frac{\lambda^2}{2(1-\lambda)}$$

resulting in $\lambda = 0.58$ which conforms the well known result of input buffering for single-channel switching.

When $m = 2$,

$$2 - \lambda = \frac{1}{1 - z_1} + \frac{\lambda^2 - 2}{2(1-\lambda)}$$

where the z_1 is the only root of $z - e^{-\lambda(1-z)}$ inside the unit circle. Then, $\lambda = 1.372$ and $\rho = 0.686$. For the values of $m > 2$, the roots of $z^m - A(z)$ where $A(z)$ is given as (5) can be determined numerically. We observed from our exercise that the increasing values of ρ with respect to the increasing values of m, which asserts the benefits of channel grouping in multi-channel switching.

ATM Congestion Control

Efficient Hop-by-Hop Flow Control for ATM Networks

C. Ozveren, R. Simcoe, G. Varghese

In connection or flow oriented networks like ATM, where it is possible to reserve buffers, it has long been known (e.g., X.25) that deadlock and fairness issues can be resolved. Each flow or connection is allocated at least one buffer on each link, which prevents deadlock.[1] In addition, network devices must also give each flow *fair* access to each shared resource that the flows contend for.

Per flow hop by hop flow control combines two mechanisms. The first is a *strict partioning of the available buffer space among active flows* and *selective backpressure* on a per flow basis. The second is *fair access to all resources* shared by flows within a single network device. The result is twofold: *packets are never dropped* due to congestion within the network, and *the available bandwidth is fairly divided up among all competing active flows.* With no contention in time, each flow has access to the entire bandwidth on each link; when there is contention, each flow has a fair share of the bandwidth. Recently, this approach has been advocated for ATM networks under the *Available Bit Rate* (ABR) service category.

Despite these attractive properties, there are two difficulties in implementing hop-by-hop flow control. The first is the reliability of the backpressure protocol. Second, the strict partioning of buffers among VCs is reasonable for a LAN environment but is wasteful and expensive for the long links of a wide area network. We address both these issues. More details can be found in [43][2].

In credit based hop by hop flow control, each sender keeps a credit register for each virtual circuit, which is initialized to the number of buffers allocated to the VC. The sender sends cells for VC y only when y has credits to send; after a cell for VC y is sent, its credit register is decremented. At the receiving switch, whenever a cell is removed from the buffer for VC y, the switch sends a credit to the receiver. Finally, when a credit message for VC y arrives at the sender, the credit register is incremented.

Strict partioning of buffers among VCs is reasonable for a LAN environment but is wasteful and expensive for a wide area network. For a coast-to-coast wide area network with a worst-case propagation delay of 20 msec, the buffering required to allow each VC the maximum bandwidth (say 600 Mbps) is about 24 Mbits, which is much too large to allow more than a small number of VCs to be supported. In order to address this inefficiency, we use a *buffer sharing* mechanism.

Buffer sharing is an old idea in networks. However, the problem considered here is different for two main reasons. First, we must do buffer sharing and yet *prevent deadlock*; buffer sharing is now more than a few heuristics, it needs a proof of correctness. Second, the information for buffer sharing needs to be distributed to the upstream node (so that it has sufficient information to send packets only when buffers are guaranteed); thus we need a distributed algorithm for buffer sharing.

The simplest approach to buffer sharing is to physically divide the buffer space into a common pool together with a private pool for each VC. To avoid deadlock, it appears necessary to mark data cells and credits as belonging to either the common or private pools. However, with current ATM cell formats, while cells are marked with a VC, it is hard to also mark them as belonging to a private or public pool. Also, the naive scheme requires additional complexity to guarantee that a VC does not exceed a maximum number of buffers.[3]

[1] Recently, the same approach has been advocated by the designers of AN-2. AN-2 is an experimental local area network based on ATM that is being built by DEC's System Research Center.

[2] Work done with coauthors at DEC. Cuneyt Ozveren and Robert Simcoe are at DEC, Littleton

[3] No VC requires more buffers than the "pipe size" of the link. Some VCs may require even less because there

We propose a simple, elegant scheme which conceptually divides the buffer space into common and private pools without marking cells, credits or buffers. Assume that there are N VCs and *MaxBuffers* cell buffers that can be used by all the VCs. Conceptually, this buffer space is partitioned so that each VC has a private pool of *Min* buffers and there is a common pool of size $(MaxBuffers - N * Min)$ buffers. Let *Max* denote the minimum number of cells required to fill one round trip delay time between the upstream and the downstream nodes.

The protocol runs in two modes: *congested* and *uncongested.* When congested, each VC is restricted to *Min* outstanding cells; when uncongested, each VC is allowed *Max* (which will typically be larger than *Min*) outstanding cells. All cell buffers at the downstream node are anonymous; any buffer can be assigned to the incoming cells of any VC. However, by carefully restricting transitions between the two modes, we allow buffer sharing, while preventing deadlock and cell loss.

Define an *upper threshold* $UT = MaxBuffers - N * Min$, which is the size of the shared pool. Then, the link is congested if the number of outstanding cells (measured across all VCs) is $\geq UT$. The choice of the upper threshold ensures no cell loss; ensuring that all VCs always have at least $Min > 0$ credits ensures no deadlock. More details can be found in [43].

Intuitively, this buffer sharing protocol performs as follows: During light load, when there are only a few VCs active, each active VC gets *Max* buffers and goes as fast as it possibly can. More precisely, if we assume that $UT = kMax$ then up to k VCs can obtain enough buffers to potentially use the full link bandwidth. Choosing a reasonable value for k will depend on particular design parameters, however, we believe that 10 will probably be a reasonable number for most cases.

Under the standard flow control scheme, recall that a 600 Mb/s link with a 20 msec propagation delay (40 msec roundtrip) required 24 Mbits, or 3 MB of memory. In this case, 64 MB total memory would only support 21 VCs. With buffer sharing and with about the same amount of memory, we would be able to support thousands of VCs while, say up to 20, of them could be active at a given time without any loss in performance.

Notice that if credits are lost, the system will steadily lose performance. To make the protocol reliable, we use a technique of local checking (that we have invented elsewhere to make protocols *self-stabilizing*) to periodically "audit" the credit based scheme and "reset" the protocol if it is a bad state. In the simplest scheme, we send a special marker cell on VC y when we wish to resynchronize this VC. We also stop sending cells on VC y until the marker returns. At the downstream node, we assume that the *marker flows through the buffer for the VC before it is sent back to the upstream node.* Thus after the marker returns, it has "flushed" the pipe of all cells and credits. Thus at the point the marker returns we can set the credit register to the maximum value. This scheme is very simple. But it has the disadvantage that it requires the VC to be idled periodically, and it is hard to bound how long it will take for the marker to return.

Thus we augment the basic scheme in two ways. First, we allow the VC to send cells after the marker has been sent, but we keep track of the cells sent since the marker was launched at the sender. When the marker returns, we adjust the correction to take into account the cells sent since the marker was launched. Second, we let the marker bypass the queue for the VC and "reflect back" immediately. To do so, the marker must return with the number of free cell buffers for this VC at the receiver.

are other, slower links on the path of that VC.

Congestion Control Mechanisms in ATM Networks

Apostolos Dailianas, Andreas Bovopoulos

In this research, we propose and analyze congestion control mechanisms for ATM networks that support multiple classes of service with distinct quality of service guarantees for each class. Our work concerns both the cell level processing and the connection admission control algorithms.

We consider four classes of service. In class A, we guarantee zero cell loss due to buffer overflow and bound cell delay variation across the network. In class B, we allow some cell loss to occur, but provide guarantees on the loss of cell bursts, using a burst reservation scheme. In the third class, we provide guaranteed cell loss rates but make no attempt to control burst loss rates. In classes B and C, we bound the cell delay variation, as in class A. Finally, in class D, we provide no guarantees on either cell loss or delay variation. We require that connections' peak and average rates be known in advance for classes A, B and C and perform connection admission control based on the assumption that connections transmit at the peak rate whenever they are active.

Cell delay variation is controlled via a framing scheme. A frame is just a time period in which some number F of cells can be received from the external link. If a cell arrives in frame i on one link, it will be transmitted on some other link in frame $i + k$, where k is chosen to ensure that the cell is guaranteed to reach the desired switch output port in the intervening frame periods. In a system supporting 150 Mb/s links and a frame time of 64 cell periods, a value of $k = 3$ will typically be sufficient to provide the required guarantee. The framing can be either implicit or explicit. In an implicit framing scheme, there is no need for synchronization of frames between switching systems, but we can only bound the delay variation on a per switch basis. That is, the cell delay variation can be plus or minus one frame period per switch. If an explicit framing mechanism is used (that is, the switches at each end of a transmission link group cells into frames in the same way), we can bound the delay variation on an end-to-end basis to plus or minus one frame period. On the other hand, this does require some explicit synchronization between the switching systems. This could be done without significant modification to the ATM transmission standard, by simply including frame position information in idle cells sent between switches.

For connections in classes A and B, we support burst-level allocation of bandwidth, so that when a connection in one of these classes begins to transmit, we allocate bandwidth and maintain it for the duration of the burst (bursts are delineated by start and end cells). When a class B burst begins, it is accepted if the amount of bandwidth currently being used by class A and other class B bursts, plus the amount required by the class B burst, does not exceed the link's capacity. Class A bursts may preempt class B bursts, but not other class A bursts. Cells in class C are accepted so long as there are unallocated slots in the current transmission frame. Class D cells are buffered separately from classes A, B and C and cells are sent from this queue whenever there is 'space available.'

The introduction of framing mechanisms to control cell delay variations introduces some complications in the connection admission control algorithms, since the framing effectively discretizes the link bandwidth. For example, if a class A connection transmits at a rate of 20 Mb/s (when it is active) on a 150 Mb/s link that uses a 64 cell frame, the closest 'natural rates' for the link are 18.75 Mb/s ($8 \times 150/64$) and 21.1 Mb/s ($9 \times 150/64$). To accommodate this connection with zero loss, we would need to allocate 9 cells per frame to this connection, even though it won't always use them all. In the same way, if this were a class B connection, we would allocate 9 cells per frame when a burst is accepted, to ensure zero loss during the burst. For class C connections however, we can

achieve more efficient link utilization by treating this as a connection with two rates, where the probability of sending at each of the two rates is weighted to give an average rate of exactly 20 Mb/s. We have designed connection admission control algorithms that do this explicitly, in order to allocate bandwidth more precisely. This is particularly important when there are connections with rates smaller than one cell per frame (for example 64 Kb/s voice has a rate of .027 cells per frame, assuming a 64 cell frame), but which still require bounded cell loss and delay variation. We have derived both exact and approximate algorithms for class C connection admission control. The exact algorithm can be computationally excessive when we have large numbers of class C connections. However, the approximate algorithm can achieve running times of a few milliseconds, even for hundreds of sources, while sacrificing very little accuracy.

One limitation of standard bandwidth allocation mechanisms in ATM networks is that when different virtual circuits are handled together (for example, they share a single link queue) it is difficult to assign different qualities of service to the different connections. We have devised ways to use assign cell loss priorities to cells in connections that are handled together by the basic cell processing hardware, but which allow the loss rates of different connections to be tailored to their specific requirements. In situations where connections have widely varying cell loss requirements, this leads to a significant improvement in the link loading levels that can be achieved while providing performance guarantees. Our approach is to assign high priority to a certain percentage of cells in each traffic stream, with the remainder receiving low priority. This allows us to tailor the loss rates of the different streams without requiring that they be separately processed. We consider both static and dynamic priority assignment mechanisms, where the dynamic priority mechanisms modify the priority assignment in response to changing traffic conditions.

More details on this work can be found in [12, 13].

Analysis of Congestion in ATM Networks

Seyyed Mahdavian

One of the key challenges of designing ATM networks is accommodating the wide range of different traffic types that these networks are designed to handle. It's been found useful to divide traffic into three main classes. The *Constant Bit Rate* (CBR) class includes applications like PCM voice, that operate at a fixed rate over an entire session. The *Variable Bit Rate* (VBR) class includes applications like coded video that can vary their transmission rate during a session, but have well-understood statistical properties, making it possible for the peak and average rate of the traffic stream (plus, possibly a measure of the time scale of variations) to be specified when a session is initiated. The *Available Bit Rate* (ABR) class includes classical data applications, such as file transfer and interactive terminal sessions, which are inherently unpredictable. For applications in this class, we can typically specify only the peak rate in advance. On the other hand, these applications are generally able to adjust their transmission rates, in response to either direct or indirect feedback from the network.

This research is concerned with analyzing the performance of ABR traffic in an ATM network. Our analyses make use of the *small buffer assumption* to make the analysis more computationally tractable. The small buffer assumption is based on the observation that since traffic bursts are generally larger (often much larger) than the link buffers in ATM switches, data loss is likely to occur whenever the sum of the rates of the actively transmitting sources exceeds the link rate. By assuming that data loss does occur whenever there is excess traffic, we can make the analysis more tractable, allowing more complex situations to be analyzed.

We have recently developed new analytical models for two situations. The first is directed at understanding packet loss (where by packet, we mean the intermediate-level data unit used by a transport protocol, not a cell or a burst) in simple statistical multiplexing situations and the second is directed at understanding data loss in fast reservation protocols. In the first analysis, we model a bufferless multiplexor receiving traffic from a *foreground traffic source* and a collection of identical *background sources*. Both the foreground and background sources are two state bursty sources, which transmit at a fixed rate when in their active state. The background sources are assumed to have exponential holdng times in the idle and active states. Our analysis allows us to compute the probability that at any instant during a time interval of length τ, the background sources generate traffic at a high enough rate so that they, together with the foreground source, exceed the capacity of the link. If τ is the time required to transmit a foreground packet, this gives us an estimate of the probability of foreground packet loss. The analysis is carried out using Laplace transforms and is computationally very fast, even for large numbers of sources.

One of the crucial parameters that affects information loss in this situation is the ratio of the foreground packet duration to the average cycle time of the background sources (the time between successive transitions to the active state). We have found that there are two distinct regions of operation. If the packet duration is much smaller than the cycle time (say 1% or less), the loss probability is approximately equal to the cell loss probability. Above this point, the loss probability increases linearly until it is close to 1 and then levels off, approaching 1 asymptotically. This implies that in many bursty applications, cell loss rate is a good predictor of packet loss rate. We have used this technique to estimate the probability of loss over different time intervals for JPEG-encoded video sources, where we modeled the JPEG sources as two state sources with a low rate of 10 Mb/s, a high rate of 25 Mb/s, an average rate of 15 Mb/s and an

average cycle time of 10 seconds. If we select the time interval to be the time it takes to transmit one horizontal slice of transform coefficients, corresponding to eight rows of pixels (0.5 ms), we can achieve offered loads of 70% at loss rates of 10^{-6} on 600 Mb/s links. Surprisingly, we can lengthen the time interval to be the time it takes to transmit a complete video frame (33 ms) and obtain very nearly the same loss rates. Hence, we see very little difference in loss performance if error recovery is done on a slice basis or a frame basis.

We have devised exact analytical models for the performance of a burst reservation system that is far more accurate than previous models under heavy load conditions. The improvement can be attributed to our explicit modeling of sources that while actively transmitting data, have failed to reserve the bandwidth they require and hence do not consume link bandwidth (previous models just assumed that all active sources consume link bandwidth). In the analysis, we model a collection of two state sources by a continuous time Markov chain with indices i and j where i is the number of active sources for which bandwidth has been allocated and j is the number of active sources for which no bandwidth has been allocated. The Markov chain decomposes in a way that simplifies the balance equations allowing for much more efficient computational procedures than would otherwise be possible. The analysis can be used to model packet-level loss under small buffer assumptions. Initial results indicate that fast reservation protocols are robust, in the sense that they are not subject to congestion collapse when the offered load exceeds the link bandwidth, but provide throughput approaching 100% as the offered load gets large. We are currently studying approximate models that can rival the accuracy of our exact models while yielding improvements in computational performance.

ATM Signaling

Design of the Connection Management Software System for the GBN Switch

John DeHart

We anticipate that the heterogeneity of ATM switching network equipment will add greatly to the complexity of controlling these networks. As shown in Figure 6, network switching systems will be produced by various vendors and (although all will presumably conform to the appropriate ATM standards) may differ considerably in their control requirements and protocols. The same will undoubtedly be true of the client terminals connected to the network. A third source of network heterogeneity lies in the links connecting the switches and the terminals, which will vary in bandwidth.

Managing such networks requires introduction of a number of control abstractions which serve to encapsulate and conceal the differences in equipment. Figure 7 illustrates a number of these abstractions. A node abstracts a switch or collection of switches, providing a view of the group as a single large switch with a known interface and capabilities. A control processor (CP) is an abstraction of the control software for a single node; in some cases the software may actually run on a single machine, while in others it may be distributed. In our control model, this software is the Core CMSS. The control software for the nodes must communicate to set up inter-nodal connections. This communication is in accord with specified, uniform internal protocols (e.g., CMNP [28] in our system). Finally, the terminal-network control interface is encapsulated by access protocols which specify the manner in which clients request connections through the network. Examples of such access protocols would be CMAP [10] and Q.93b [1].

The Core CMSS present at each node is structured in three layers. The Connection Management Layer is actually distributed across all the nodes of the network, and uses the internal protocols of Figure 7 to set up inter-nodal connections. At each node, the Connection Management Layer communicates with the Node Management Layer for that node. The Node Management Layer abstracts the collection of switches in the node so that they "look like" a single large switch to the Connection Management Layer, and is responsible for managing intra-nodal connections within the node. It issues commands to the Switch Management Layer, which handles connections within the individual switches and conceals hardware dependencies from the other layers.

In the current design, the Core CMSS is realized as a tree of processes running on the node's CP, as shown in Figure 8. The top process in the tree is the Connection Manager (CM) for the node. This process communicates with the CMs of other nodes and with one subsidiary process. For the node shown in the figure, where three switches are grouped and managed as a single node, the subsidiary process is a Node Controller (NC). This NC in turn communicates with its subsidiaries; in the example these are one Switch Controller (SC) for each switch. If, as in the example, the switch hardware is supplied by several different vendors, these SC processes will be from different executables, each tailored to control the specific hardware. However, the API provided by each SC is identical, and is identical to the NC API. The GBNSC [11] is one type of SC.

CMSS processes communicate only along the links of the process tree. For example, in Figure 8 the CM and NC may communicate and the NC may communicate with each SC. However, the CM does not communicate directly with the SCs, nor do the SCs communicate with one another - indeed, the encapsulation provided by the CMSS is such that these processes have no knowledge that the others exist.

This design decision - that the SC and NC have

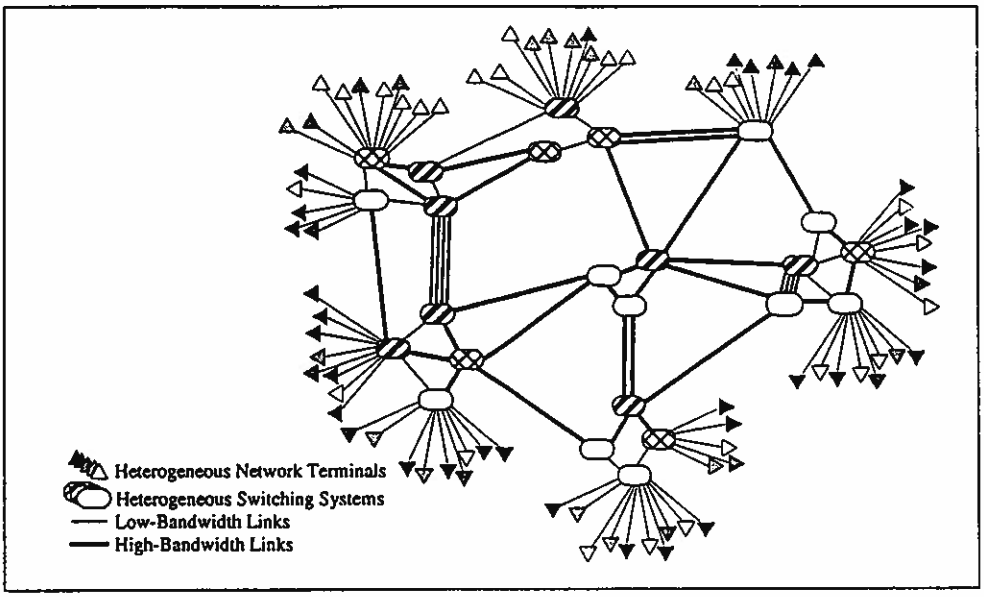


Figure 6: Heterogeneous ATM Networks

the same API - was originally motivated by the desire to have a node "look like" a switch to the layers above the node. However, it also has some nice consequences for the setup and management of the process tree, as shown in the examples of Figure 9. The upper example shows a single switch which is to be managed as a node. In this case we can omit the Node Management Layer entirely, and have the CM communicate directly with the SC for the switch. The lower example shows a node hierarchy, where one of the subsidiary elements of a node is another node. The process tree reflects the nested structure, and because the NC and SC APIs are identical the upper NC is unaware that one of its subsidiaries is actually a multiple-switch node.

The API used by the NCs and SCs is represented by a software object called the Node Controller Managed Object (NCMO). The NCMO encapsulates the interactions between two processes (a parent and a subsidiary) in the CMSS process tree. The NCMO provides the means whereby the process tree is established. When a parent process determines it has a subsidiary, it creates an NCMO object for that subsidiary. The creation of this object causes the creation of the subsidiary process and the communications links to that process. Subsequent interactions with the subsidiary are handled through function calls on the NCMO. These calls typically cause communication with the subsidiary; the NCMO uses the Node Controller Communications Protocol (NCCP) for this communication.

Processes outside the CMSS tree may also need to communicate with the NCs or SCs. For example, for network management functions it might be useful to have a tool that allows managers to issue commands directly to an SC, bypassing the CM and NCs. Jammer, also developed by ARL, is an example of this type of tool. The user of Jammer may interact directly with the GBNSC to examine or modify any of the GBS tables, to set up connections, and to execute test suites that verify switch functionality. For reasons of simplicity, Jammer also uses NCCP to communicate with the switch.

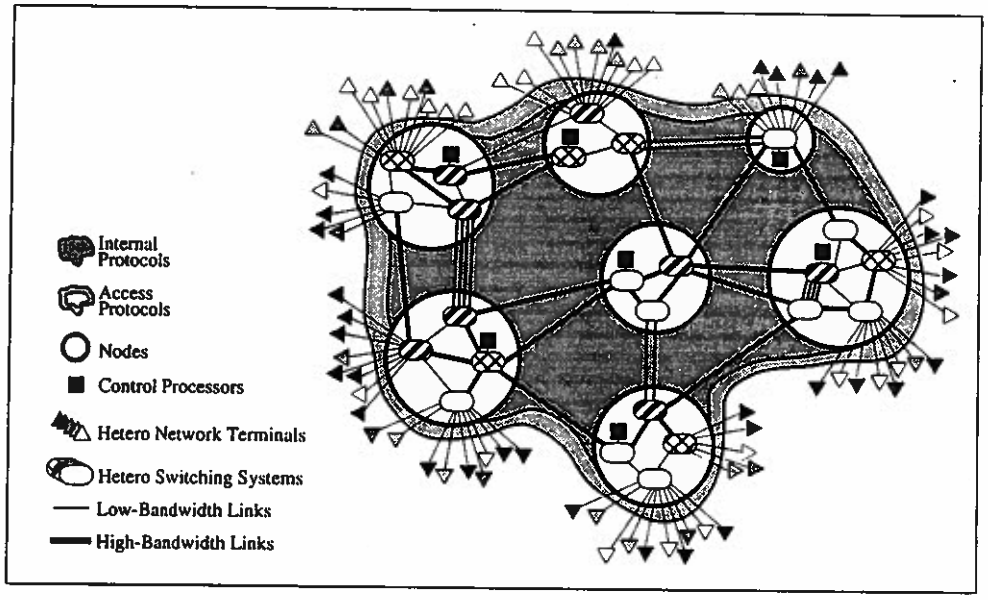


Figure 7: Control Abstractions for ATM Networks

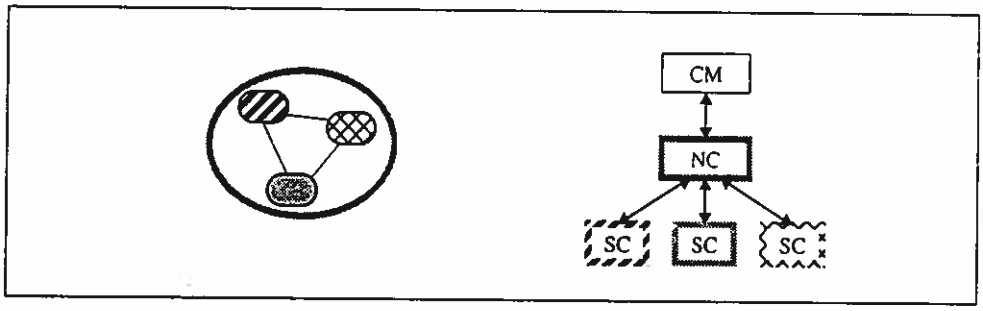


Figure 8: Example CMSS Process Structure

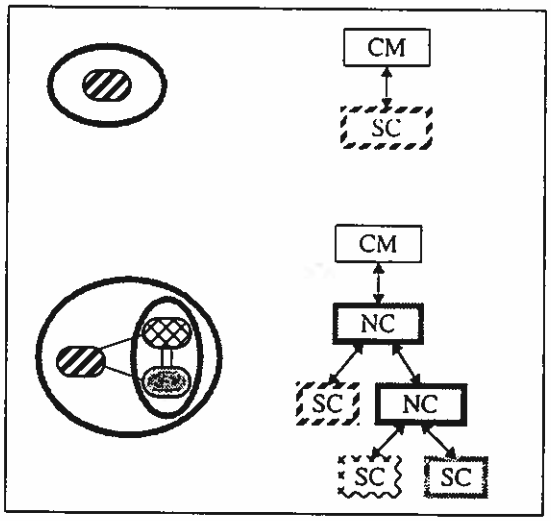


Figure 9: Additional CMSS Process Structures

Multimedia Systems and Protocols

The Washington University MultiMedia eXplorer

William D. Richard, Jerome R. Cox, Jr., A. Maynard Engebretson, Jason Fritts, Brian L. Gottlieb, and Craig Horn

The Washington University MultiMedia eXplorer (MMX) is an important component of Project Zeus; it provides a complete multimedia system capable of transmitting and receiving video, audio, and radiographic images over the Washington University broadband ATM network. The MMX is a second-generation multimedia system based on the earlier, host-based Washington University Multimedia System (MMS) [45]. A block diagram of a typical MMX configuration is shown in Figure 10.

An MMX is typically used with an NTSC video camera, microphones, and amplified stereo speakers. The analog NTSC video signal generated by the video channel is usually fed into a "video-in-a-window" card installed in the host computer as shown in Figure 10. In this configuration, the received video is displayed in a window on the normal workstation display eliminating the need for a separate video monitor. A high-resolution radiological image display is supported for medical applications. Other configurations are possible, including the use of a separate video monitor or the use of a laser disk player or other audio/video source [46].

To reduce medical costs, hospitals are seeking cost saving opportunities through the sharing of information via networks. For example, using an MMX and medical video, a specialist can efficiently support ultrasound and fluoroscopic examinations at a number of satellite locations. The advent of broadband networks promises to remove the barrier to electronic communication for the deaf and hard of hearing. We have demonstrated the effectiveness of the MMX to provide the capability for the deaf to communicate over arbitrary distances as if they were face to face. These two applications foreshadow large-scale usage of broadband networks when metropolitan and wide-area tariffs become sufficiently low.

The MMX is designed to operate with any host. A standard RS-232C interface is used to control the MMX from an application program running on the host. If the host is equipped with an ATM interface card, it can receive normal network traffic via an extension port on the MMX. The extension also multiplexes cells from the host ATM card with cells originating from the MMX to form a single outbound cell stream. Audio, video, and radiological image delivery functions are performed by the multimedia subsystem.

The ATMizer subsystem consists of a 16 MHz Motorola MC68030 CPU with 4 Megabytes of DRAM, 128 Kilobytes of EEPROM, a Multi-Function Peripheral Chip (MFP), and interfaces to the ATM network and to the multimedia subsystem. The EEPROM holds the embedded control software for the MMX. The DRAM is used for serial I/O buffering and CPU channel ATM cell management. The MFP provides timing and interrupt services, as well as the serial port used to communicate with the host.

Embedded code for the local MC68030 CPU is written in C and currently compiled using the native C compiler on a NeXT host. A library of I/O function calls, e.g., printf and scanf, was written to replace the stdio C library to facilitate code development and debugging. A low-level monitor program, which is part of the embedded code, performs memory dumps, allows program execution, etc. For debugging or diagnostic purposes, a terminal can be used with the on-board serial port to interact with this monitor.

The interface to the ATM network is implemented using an Advanced Micro Devices TAXIchip transmitter/receiver pair. The ATMizer is configured to transmit and receive data via either twisted pair or multi-mode fiber using the framing protocol specified by the ATM

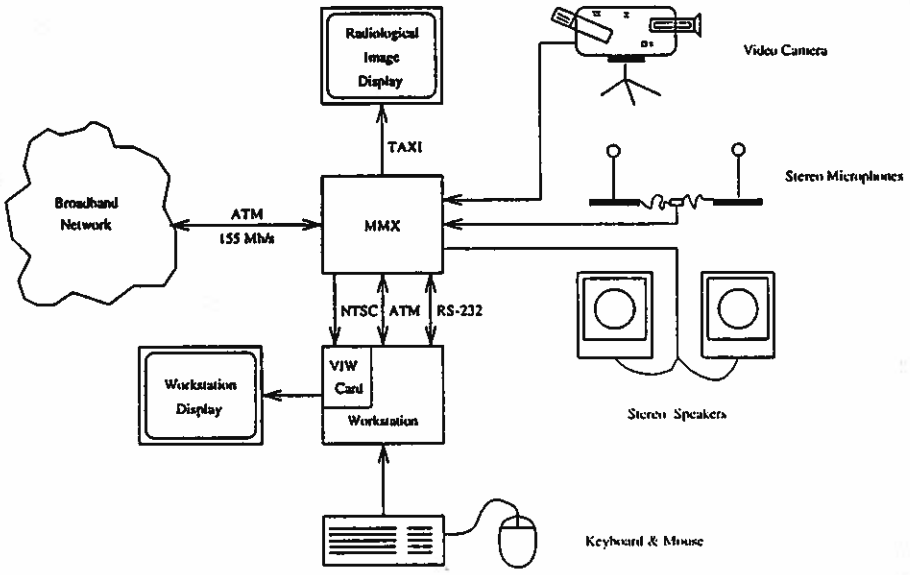


Figure 10: An MMX is typically configured with a video camera, stereo microphones, and amplified stereo speakers.

Forum for ATM systems operating at 155 Mb/s (8B/10B). The extension port, using analog circuits, copies incoming signals and transmits them to both the local TAXI receiver and to the ATM link to the host. The ATM interface on the MMX can be programmed to ignore cells not specifically directed to either the local CPU or one of the multimedia channels. Similarly, the host must ignore cells bound for the MMX.

There are currently three multimedia channels used with the ATMizer. The video channel digitizes an input color NTSC video signal, compresses the digital video data stream using a hardware Joint Photographic Experts Group (JPEG) compression engine, stuffs the appropriate framing tags into the data stream, and routes the resulting data stream to the ATMizer Bus for transmission. The video channel also accepts a compressed video data stream from the ATMizer, strips the framing tags, decompresses the data stream, and produces an output NTSC video signal. A frame buffer is used in conjunction with a novel Phase-Locked-Loop (PLL) scheme to eliminate the synchronization problems caused by separate transmitter and receiver clocks. Two hardware JPEG engines are used by the video channel so that a full 640-by-480 pixel video image can be compressed for transmission and decompressed for display simultaneously at 30 frames per second.

The second multimedia channel, digitizes an input analog stereo audio signal, i.e., both left and right channels, using a stereo audio codec and produces a CD-quality digital audio data stream for transmission by the ATMizer. The codec oversamples at 64 times the output word rate of 44.1 kHz and generates 16-bit values for both the left and the right inputs. The 48-byte payload of each transmitted cell contains twelve digital audio data words, i.e., twelve 16-bit left and 16-bit right channel sample pairs. With this sampling rate and payload format, one audio cell is transmitted every 272 microseconds. This channel also accepts a digital audio stream from the ATMizer and generates an output analog stereo audio signal. Monaural input and monaural output are also supported by the system.

A TMS320C50 Digital Signal Processor (DSP) provides volume control, loopback, left/right mixing, and amplification functions, as well as single-source payloads and multiple-source, monaural payloads. Multiple-source payloads are accepted by the receiving channel on a

priority basis and mixed together as a blend of the two audio streams. Source priorities are established when a receiving MMX system is initialized and are stored in global memory. The DSP implements a voice activation switch so that audio sources generate cells only when sounds above a threshold are present. Feedback cancellation filters, also implemented by the DSP, are used to eliminate the direct signal path from the loudspeaker to the microphone. The echo cancellation feature of the audio board can be enabled to eliminate feedback due to having a microphone in close proximity to the speakers. This feature is most useful in a conferencing situation, where multiple conference participants are using microphones and speakers. If not properly controlled, such a configuration can have a severe feedback loop. With the echo cancellation enabled, this loop is broken with less need for precise arrangement of the speakers and microphones.

The third multimedia channel accepts high-resolution radiographic images from the ATMizer and reformats them for transmission to a high-resolution monochrome display. The radiographic images used with this channel are served to the ATM network via a dedicated transmitter. This transmitter accepts a data stream in the format required by the display and reformats it for transmission over the ATM network.

The MMX software suite consists primarily of a low level monitor running on the embedded CPU and a library of C routines for accessing the monitor functions via the serial port. This library has been used to develop several application programs for the Project Zeus testbed on multiple workstation platforms.

For ATM communications, the embedded software provides the ability to filter and route incoming cells to any of the devices in the MMX. The routing is based on the VPI and VCI of the cell. For each VPI/VCI pair, the MMX may be programmed to pass to the destination device any combination of the cell header, HEC byte, and payload. On the transmit side, the outgoing headers for the audio and video channels may be programmed to send the audio and video streams using any VPI/VCI pair.

For the video channel, the library provides routines for adjusting the JPEG Quantization (Q) Factor and for selecting which of three video inputs should be encoded and transmitted. There is also a general command for manipulating the various video parameters.

For the audio channel, there are functions for controlling the volume of the audio output, the mixing of local audio with the incoming ATM audio, the sampling rate, echo cancellation, and the number of incoming ATM audio streams. There are six controls for audio volume. Two are for listening to the local audio. These send the audio from the stereo inputs to their respective outputs, bypassing the ATM link. The remaining four volume controls are for mixing the audio channels at the outputs. By using the appropriate controls, the MMX can send both audio channels to the same speaker, converting the stereo signal into a monaural signal.

A library of C routines was written to provide a simple programming interface to the MMX. The library effectively places a wrapper around all of the functions described above. This wrapper allows a programmer to make a simple function call without having to worry about the specifics of the serial control protocol associated with the MMX. This abstraction allows for changes in future versions of the embedded software without the need to rewrite the applications running on the workstations.

Collaborative Projects

Jerome R. Cox, Jr., G.J. Blaine, Nilesh Gohel, James D. Miller

The purpose of Project Zeus is to develop networking components, to install a networking testbed, and to develop challenging applications that stress the testbed. Two such applications will be described below.

Increasingly, to reduce medical costs, hospitals are merging into systems containing dozens of hospitals and healthcare facilities. These systems provide cost-saving opportunities through the centralization of expertise, the use of automation, and the sharing of information via networks. Such an opportunity exists in connection with fluoroscopic and ultrasonic examinations carried out at satellite clinics and hospitals.

For example, a technologist conducts the examination at the remote site (see Figure 11). A specialist (gastrointestinal radiologist or obstetrician) located at the medical center provides guidance to the technologist via video conferencing and by monitoring medical video captured from the fluoroscopic or ultrasound equipment. Both the patient and the technologist can interact with the physician despite the distance. It is important to provide a good audio channel for interaction between the Center-of-Expertise and the Remote Site both during video conferencing and during audio conferencing while the technologist and physician are examining medical video sequences.

Thus, with this arrangement, the specialist can efficiently support examinations at a number of satellite locations. There is no time lost for travel by the specialist, and a number of satellite locations can be placed throughout the community and in rural areas. For those cases where the equipment is expensive or a technologist cannot be occupied fully at the remote location, a mobile examining facility can be used to increase utilization. Of course, it will be necessary to have broadband access at each of the mobile exam locations, but that access should be readily available within a few years.

Early psychophysical results indicate that compression of the medical video for these specialties can be accomplished using motion JPEG at average data rates of 11 Mb/s or less without producing visually detectable artifacts as a result of image reconstruction. Using a staircase psychovisual protocol, Gohel, et al. [24] used recordings of video examinations both with a variable amount of compression and without compression. These recordings were taken from examinations in the three fields of OB/GYN ultrasound, abdominal ultrasound, and GI/GU fluoroscopy. The threshold of artifact detection for specialists in these three fields of medicine yielded average data rates that ranged from 2.5 to 11 Mb/s.

Several demonstrations have been conducted with MMXs set up as shown in Figure 11. The physicians participating in these trials have agreed that the quality of the video sequences provides the support needed for supervision of examinations at remote locations.

The deaf and hard of hearing population has been disenfranchised with respect to electronic communication. The hard of hearing have had some ability to communicate with each other and the hearing world by amplified sound of telephone bandwidth, while the deaf have been limited to communications through teletypewriter networks. The advent of broadband networks promises to remove these communication barriers.

The deaf and their hearing interpreters can communicate using video via lip reading, finger spelling, signing, or some combination thereof. For example, two deaf individuals can communicate directly, or a deaf person can communicate with a hearing person indirectly through a hearing interpreter acting as an intermediary (see Figure 12). The three individuals can be in three different locations: the deaf person and the hearing interpreter at

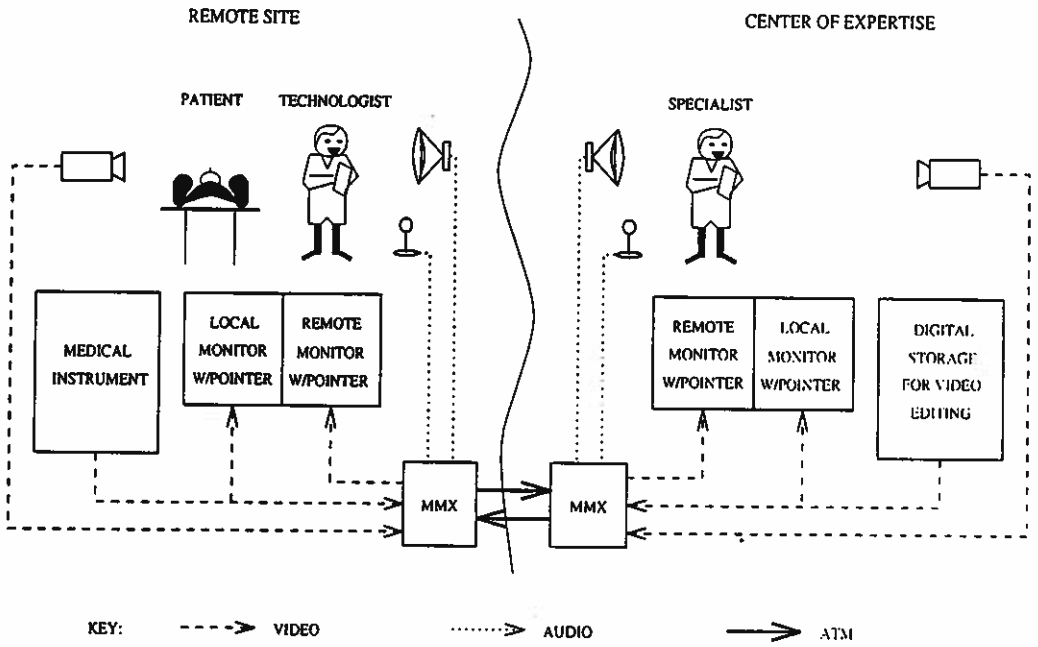


Figure 11: Use of MMX to support remote examinations requiring medical video.

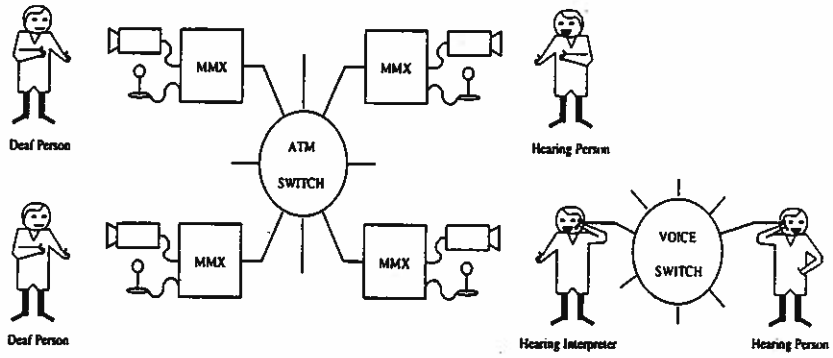


Figure 12: The MMX can be used effectively by the deaf and hard of hearing to communicate between themselves or with the hearing world directly, or indirectly through a hearing interpreter.

locations with MMXs and the hearing party communicating with the interpreter over a conventional telephone connection.

We have demonstrated on several occasions the effectiveness of the MMX to provide communication for the deaf over arbitrary distances as if the parties to the conversation were face to face. Initial studies of the amount of compression tolerable suggests that JPEG compression at 8 Mb/s with modest reconstruction artifacts do not harm the viewer's ability to communicate [34]. However, lip reading was substantially more sensitive to compression artifacts than were finger spelling or sign language.

A Gbps ATM Desk Area Network

Zubin D. Dittia, Jerome R. Cox, Jr., Guru M. Parulkar

The emergence of very high speed networks 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.

The current state-of-the-art in workstation architectures 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[4], 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.

Thus, solving problems inherent in the software architecture or protocols would not help in overcoming limitations imposed by the peak system memory and bus bandwidths. 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.

We describe the design of a gigabit ATM desk area network which will serve as an I/O subsystem and 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

[4]The effective bandwidth of the fastest buses in use is no longer much higher than the link bandwidth of Gpbs neworks. With multiple bus traversals, the available bandwdith is typically much lower than the network bandwidth.

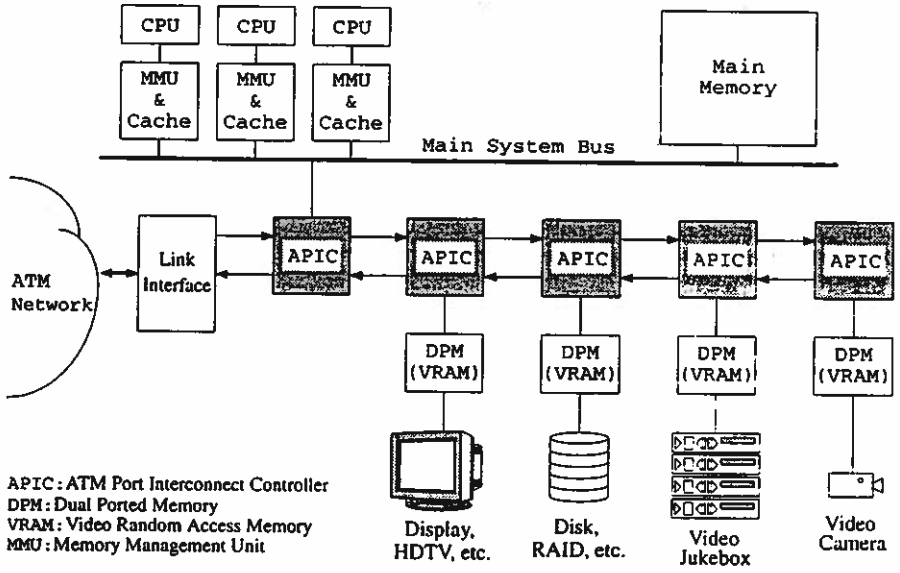


Figure 13: A Gbps ATM Desk Area Network

example applications.

Proposed Architecture. A high level schematic of our proposed architecture is shown in Figure 13. 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, 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 (*e.g.*, Mbus); 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.

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 researchers at the University of Cambridge in England. 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 - 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.

A more detailed description of the internal design of the APIC chip 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 are presented in [15, 16]. In these papers, 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.

Design of a Large Scale Multimedia Storage Server

Milind M. Buddhikot, Guru M. Parulkar, Jerome R. Cox. Jr.

Large scale multimedia storage servers will be essential for future applications such as multimedia mail, orchestrated presentations, high quality on-demand audio and video, collaborative multimedia document editing, browsing remote multimedia archives and virtual reality environments. The primary requirements of such large scale servers are: 1) support potentially thousands of concurrent customers all accessing the same or different data 2) support large capacity (in excess of terabytes) storage of various types 3) deliver storage and network throughput in excess of a few Gbps, 4) provide deterministic or statistical QOS guarantees in the form of bandwidth and latency bounds, and 5) support a full spectrum of interactive stream playout control operations such as *fast forward (ff), rewind (rw), slow play, slow rewind, frame advance, pause, stop-and-return and stop.* In addition to these requirements, a multimedia server may have to provide a lot of compute power to support near real-time media processing.

The existing network based storage servers suffer from serious network and storage I/O bottlenecks and will not be able to meet the aforementioned requirements. Therefore, designing large scale high performance servers is a challenging task that requires significant architectural innovation. To this end, we have undertaken a project called the *Massively-parallel And Real-time Storage* (MARS) architecture. It consists of a set of independent storage nodes that are connected together by a fast packet based interconnect. The terms "massively-parallel" and "real-time" signify that the system allows real-time retrieval of data streams from a large storage system in a parallel fashion. The interconnect can be a packet switched bus, a ring or even a general purpose multicast switch. Each storage node provides one or more of the resource management functions such as file system support, scheduling support, compute support and admission control.

Our work also uses some of the well known techniques in parallel I/O such as data striping and Redundant Arrays of Inexpensive Disks (RAID) and innovative data striping and real-time scheduling to allow a large number of guaranteed concurrent accesses and use separation of metadata from real data to achieve a direct flow of the media streams between the storage devices and the network.

Note that we are interested in supporting an environment that allows media retrieval as well as edits. However, the solutions currently being studied are aimed at a retrieval environment. We believe that the issue of media edits is tied to the data layout schemes and is orthogonal to the hardware architecture. Therefore, we plan to extend the data layout and the associated scheduling schemes to support a read-write environment.

Figure 14 shows a prototype architecture of a MARS server. It consists of two basic building blocks: the *ATM Port Interconnect Controller* (APIC) based interconnect and the storage node. Our ATM interconnect is made out of an ASIC called APIC (ATM Port Interconnect Controller), that is currently being developed as a part of an ARPA sponsored gigabit local area ATM testbed. The APIC chip is the basic building block for a high bandwidth *networked-I/O subsystem*, that provides a direct interface to the network for the host (workstations as well as servers) and a variety of I/O devices. The details on the interconnect and chip internals can be found in [15]. The central manager shown in Figure 14 acts as a resource manager responsible for managing the storage nodes and the APICs in the ATM interconnect. For every media document, it decides how to distribute the data over the storage nodes and manages the associated metadata information. It receives the connection requests from the remote clients and

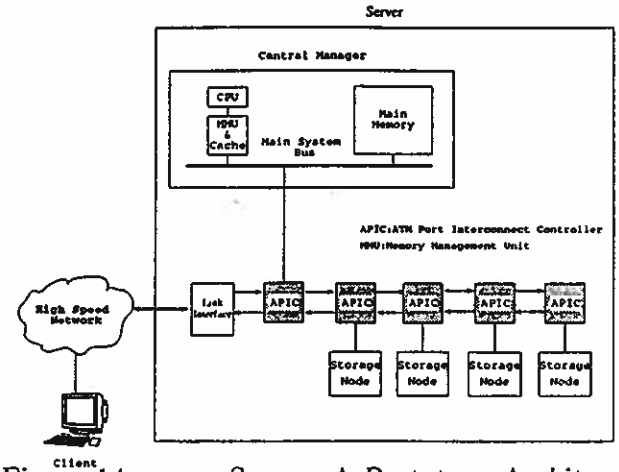


Figure 14: MARS Server: A Prototype Architecture

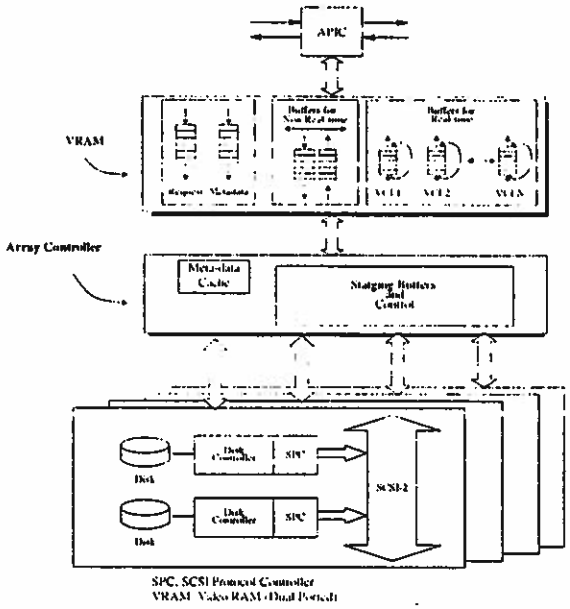


Figure 15: Storage node architecture

based on the availability of resources and QOS required, admits or rejects the requests. For every active connection, it also schedules the data read/write from the storage nodes by exchanging appropriate control information with the storage nodes. Note that the central manager only sets up the data flow to and from the storage devices and/or the network and does not participate in actual data movement. This ensures high bandwidth path between the storage and the network.

Storage Node. The architecture of the storage node described here assumes the storage in the form of a RAID, however it can be easily extended to accommodate other forms of storage such as a set of independent high capacity optical and magnetic disks. A RAID storage node is illustrated in Figure 15. The disk array at the node is constructed out of a set of *Small Computer System Interconnect* (SCSI) strings, with a fixed number of disks per string. The multiple SCSI strings are controlled by an array controller, which interfaces to the APIC through a dual ported memory, such as a video RAM (VRAM).

As shown in Figure 15, in a read-only environment, the array controller asynchronously writes the data in the VRAM and the APIC consumes it to transfer it to the network. The VRAM is shared by the buffers for periodic streams, buffers for non-real time tasks, request buffers, and positional metadata buffers.

The array controller is responsible for managing the local storage and providing one or more of the resource management functions outlined earlier. It maintains small staging buffers similar to the ones maintained in the VRAM. In order to minimize accesses to the metadata queues in the VRAM, the array controller maintains a metadata cache. The control block, the heart of the array controller, is a finite state machine or an embedded processor that executes the SCSI protocol for disk read/writes and controls the staging buffer and metadata caches.

Work in Progress. Our ongoing work includes design and implementation of the disk array controller to work with APIC interconnect and development of multilevel real-time scheduling, data layout and admission control algorithms to guarantee QOS requirements to large number of clients.

Distributed Layout, Scheduling and Playout Control in a Multimedia Storage Server

Milind M. Buddhikot, Guru M. Parulkar, Jerome R. Cox, Jr.

In the previous section we briefly described our prototype architecture of a multimedia storage server that transparently connects storage devices to the external network. The data layout, scheduling schemes and implementation of playout control operations, have implications on the extent to which this architecture can meet various requirements outlined earlier. Here, we will present some of the possible data layouts and scheduling schemes that efficiently support full spectrum of playout control operations and highlight the interaction between them.

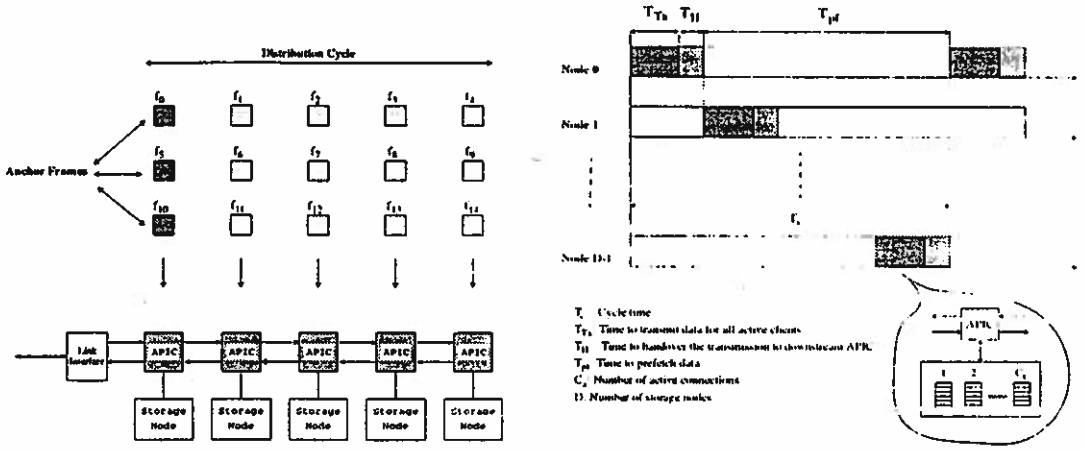
Data Layout and Scheduling. The periodic nature of multimedia data is well suited to spatial distribution or striping. For example, a logical unit for video can be a single frame or a collection of frames. Each such logical unit or the parts of it can be physically distributed on different storage devices and accessed in parallel. In our architecture, we use this property of multimedia data to stripe it in a hierarchical fashion. We outline only one of the many possible data layout schemes that satisfy real-time playout requirements and allow a large number of concurrent accesses to the same or different data in a retrieval environment. It must be noted that the architecture, data layout, data compression, and scheduling interact very strongly, a detailed discussion of which can be found in [4].

Consider an example system, shown in Figure 16 (a), with five storage nodes numbered 0 to 4 from left to right. The frames f_0, f_1, f_2, f_3, f_4 are assigned to nodes D_0, D_1, D_2, D_3, D_4 respectively. The frame f_5 is again assigned to node D_0, thus following a frame layout topology that looks like a ring. Given that there are D storage nodes, this Distributed Cyclic Layout (DCL) has the property that at any given node, the time separation between the successive frames of the stream is $D \times$ *Inter-frame time*. Thus, the effective period of each stream seen by a storage node is D times longer, which facilitates prefetching of data to mask the high rotational and seek latencies of the magnetic disk storage. Note that the granularity of data striping over the storage nodes can be in units of multiple frames called a *chunk* [4]. Also, depending on the nature of the storage at the node, the frames assigned to it can be stored in multiple ways. For example, in the case of a RAID, the blocks of a frame can be further striped on the disks of the RAID.

Next, we will present a simple scheme for scheduling data retrieval from storage nodes when clients are assumed to be bufferless. In this scheme, shown in Figure 16(b), each storage node maintains C_a buffers, one for each active connection. In a retrieval environment, the data read from the disks at the storage node is placed in these buffers and read by the APIC. At the time of connection admission, every stream experiences a playout delay required to fill the corresponding buffer, after which the data are guaranteed to be periodically read and transmitted as per a global schedule.

The global schedule consists of periodic cycles of time length T_c. Each cycle consists of three phases: *data transmit*, *handover* and *data pre-fetch*. During the *data transmit phase* (T_{Tx}), the APIC corresponding to a storage node reads the C_a buffers and transmits them over the interconnect to the network. Once this phase is over, the APIC sends control information (a control cell) to the downstream APIC so that it can start it's data transmit phase. The last phase in the cycle, namely the *data pre-fetch phase* (T_{pf}) is used by the storage node to pre-fetch data for each connection that will be transmitted in the next cycle.

As explained in [3], with the number of storage nodes D=15 and a simple disk array capable of



(a) Example layout

(b) Simple scheduling scheme

Figure 16: An example layout and scheduling scheme

delivering sustained 5 MBps throughput at each node, as many as 110 standard MPEG streams or 27 HDTV compressed streams can be supported. It must be emphasized here that our scheme allows all the clients to concurrently and independently access the same stream or different streams.

Playout Control. Future on-demand multimedia applications will require interactivity in the form of stream playout control that allows user to do *fast forward, rewind, slow play, slow rewind, frame advance, pause*, and *stop-and-return* on a media stream. Unlike the linear access supported by present-day video cassettes, a user may access the media streams in a random fashion (as permitted by existing CD-ROMS and laser disks). However, such playout control operations have strong implications on the data layout and scheduling in the server. Due to space constraints, here we will focus on *ff* and *rw* operations. In our work, we implement these operations using a Sequence Variation (SV) scheme. A SV scheme keeps the display rate same as for normal playout but modifies the sequence of frames displayed to achieve a perception of fast forward or rewind.

The simple data layout and scheduling scheme described above works for normal stream playout but not for *ff* and *rw*. Consider an example system with four connections and six ($D = 6$) storage nodes. Assume that the fast forward is implemented by skipping alternate frames. Figure 17 illustrates two consecutive (i^{th} and $(i+1)^{th}$) scheduling cycles in this system. Assume that in the $(i+1)^{th}$ cycle connection C_0 starts doing fast forward whereas rest of the connection are still performing normal playout. Thus, for connection C_0, upon *ff*,the frame sequence for normal playout is $\{0, 1, 2, 3, 4, 5, \ldots\}$ is altered to to $\{0, 2, 4, 6, 8, 10, \ldots\}$. This implies that in this example, the odd-numbered nodes are never visited for frame retrieval during *ff*. Also, as the display rate is constant, node 2 for example, must retrieve and transmit data in a time position which is otherwise allocated to node 1 in normal playout. Thus, there are two main problems: first, the stream control alters the sequence of node-visits from the normal linear (modulo D) sequence. In other words, the transmission order is no longer the same for all connections when some of them are doing fast forward or rewind. Therefore, the transmission of all connections can no longer be grouped into a single transmission phase. Secondly, it forces some nodes to retrieve and transmit more often, creating "hot-spots" and, in turn, requiring bandwidth to be reserved at each node to deal with the overloads. Such additional bandwidth reservation will lead to

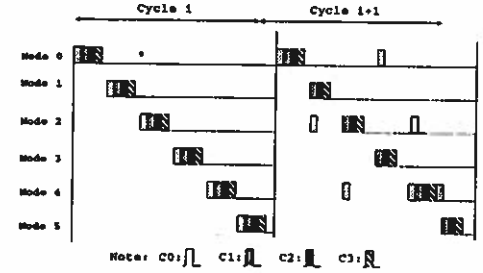


Figure 17: Revised schedule when C_0 fast forwards

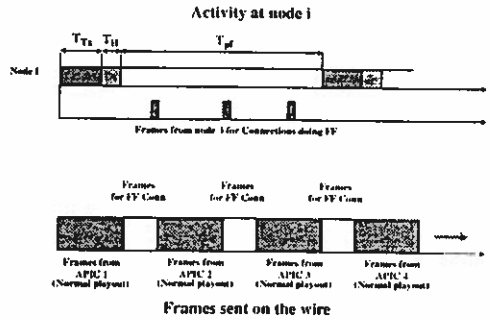


Figure 18: General case of M out of C_a connections doing fast forward

conservative admission control and poor utilization. If no such bandwidth reservation is made, QOS may be violated during overloads.

The first step in fixing these problems is to decide the order of transmissions for different nodes on a per-connection basis. Figure 18 illustrates this when M out of C_a active connections are performing fast forward. At a typical node i the transmission occurs in multiple phases, one of which is for connections performing normal playout and rest are for connections performing fast forward. These phases cannot be combined into a single phase as the transmission order of all the M connections performing fast forward is not identical. The sequence of frames appearing on the wire consists of two sequences: a sequence of frames transmitted from an APIC followed by frames transmitted from possibly all APICs for connections performing fast forward. It must however be noted that at any time, only one APIC transmits frames for a connection. The side effect of this revised schedule is that now the pre-fetch and the transmission phases for a storage node overlap. In presence of a large number of connections doing fast forward and rewind, this overlap makes it difficult to guarantee that data to be transmitted has been pre-fetched into the buffers. Hence, to achieve a smooth transition from normal playout to fast forward, we provide two ping-pong buffers per connection: one buffer used to store the data being pre-fetched and the other used to transmit the previously pre-fetched data. This effectively decouples transmission and pre-fetching and allows transmission order to be modified on a per cycle basis.

As can be seen in the above example, the load distribution is still unbalanced. Load balance is ensured if we can guarantee that each storage node fetches and transmits a fixed number of frames per connection in each pre-fetch and transmission cycle irrespective of whether a connection is performing normal playout or other playout control operations. This is achieved by either constraining the data layout or modifying the data layout. We have proved that if number of storage nodes D and the fast forward distance d_f are relatively prime, then the node set for fast forwarded frames is balanced. We have also shown how our data layout schemes can compensate for the load-imbalance inherent in distributing frames for an MPEG stream. Also, our dynamic cycle-by-cycle scheduling allows the VBR characteristics of such streams to be exploited. The details regarding this can be found in [4].

Work in Progress. In our ongoing work, we are developing and analyzing the distributed, multilevel real-time scheduling, data layout and admission control algorithms to guarantee QOS to a large number of clients during normal playout as well as a full spectrum of playout control operations. Several issues such as design and evaluation of node-specific storage policies, implementation schemes for distributed scheduling, metadata design and distribution will be examined.

QoS Support for Multimedia Applications within the Endsystems

R. Gopalakrishnan, Guru Parulkar

Emerging broadband networks can cater to the needs of a variety of media, both continuous (such as video) as well as discrete (data). Likewise, modern high performance workstations can store and process continuous media (CM), and can send and receive CM data over the network in the same way as discrete data. These two technologies have provided the impetus for the development of a variety of distributed multimedia applications.

An important aspect of multimedia applications is that they require quality-of-service (QoS) guarantees from the endsystems (for processing) as well as from the network (for communication). Much research has been conducted on how to provide QoS guarantees in networks. To keep up with increasing transmission rates, it is necessary that the host must possess sufficient processing capacity and internal datapath bandwidth. In addition to this, the network subsystem and the host operating system (OS) must ensure that this aggregate capacity is shared among the application processes according to their individual QoS requirements. There are four issues that must be considered to provide QoS guarantees for protocol processing. These are:

QoS Specification. This involves developing mechanisms to specify QoS requirements for a large variety of application types. Since it is not possible to foresee all possible requirement types while designing the networking software, it is necessary to identify a few application classes that can capture the needs of most applications. Within each class, it is necessary to identify parameters to specify QoS requirements.

QoS Mapping. The QoS specifications are at the application level. Since several resources (such as CPU, memory, and network connections) are involved in communication, the specifications must be mapped to resource require ments. For example, QoS parameters (such as bandwidth) have to be derived for the network connection. Like wise, the amount of processing required must be determined so that the CPU capacity can be allocated to ensure that protocol data units (PDUs) are processed at the desired rate.

QoS Enforcement. Enforcement has to do with scheduling shared resources during data transfer to satisfy the requirements of each connection. These requirements are derived by the mapping operation. We are mainly concerned with efficient CPU scheduling mechanisms to reduce context switching, and to integrate these scheduling mechanisms into protocol code.

Protocol Implementation. The solutions for QoS specification, mapping and enforcement have to be integrated into protocol implementations. We have adopted an application level protocol implementation model. In this model, protocols are part of each application process instead of being implemented within the os. Since the process is the basic unit of resource management within the endsystem, we can associate a certain amount of resource capacity with each process as dictated by its QoS requirements. Then, existing resource management mechanisms can be used to enforce these allocations. Having chosen the protocol implementation model, we must develop efficient mechanisms to reduce the overhead associated with data movement and context-switching since they domi- nate protocol processing costs.

Overview of Solutions. Figure 19 shows the QoS specification and mapping operations. These two operations are performed during connection setup, and determine the performance during the data transfer phase. For each connection, the application specifies values for QoS parameters. The parameters and their interpretation depend on the class of the

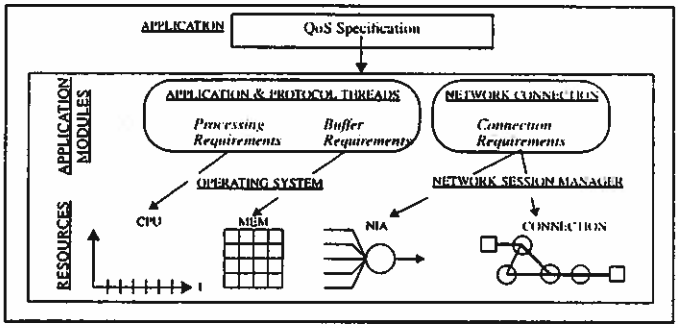


Figure 19: QoS Specification and Mapping Framework

data stream and are described in [25]. Given the specification, the mapping operation determines requirements for resources. In Figure 19, the mapping operation derives the processing requirements for a connection as follows. Each connection is associated with a thread. The thread represents the processor resource since it does all the processing of PDUs over its connection. For bandwidth critical applications, we reserve processor capacity by associating a time period with each thread and scheduling it to run for some amount of time in each period. This guarantees that in each period the thread processes a certain number of PDUs. The choice of thread period and the number of PDUs to be processed in each period are derived based on QoS parameter values, and measured parameters such as the time to process a PDU of a given size. The os is then requested to create the thread and bind it to the protocol code. Similarly, the mapping operation derives the bandwidth required for the network connection and specifies it to the network in terms of the connection attributes.

The QoS enforcement and protocol processing operations come into play during the data transfer phase. This is shown in Figure 20 which depicts the organization of the host communication subsystem. It shows two schedulable resources in the endsystem that are involved in supporting a connection: the network interface adaptor that sends and receives frames, and the thread that processes the protocol data units(PDUs) contained in these frames.

The PDUs are exchanged on connections supported by the underlying atm network. For the sake of understanding, we have shown connections as half-duplex. The adaptor takes outgoing PDUs enqueued for each connection, segments it into cells and sends these cells at the rate determined by the connection bandwidth. This action is referred to as pacing. Likewise, the adaptor reassembles incoming frames on each connection into PDUs and places them in the host memory to be processed by the application process. The adaptor is controlled by its device driver. The driver maps memory pages containing network data between the process address space and per connection queues. The driver also sets up the adaptor to dma data for each connection from (or to) its queues in host memory. Physical copying is avoided whenever possible.

The CPU resource is shown with a single application process that has periodic threads that do protocol and application level processing. The application thread does operations such as presentation layer functions and other functions that could depend on the nature of the data stream. QoS enforcement is mainly concerned with scheduling threads for all the connections so that they obtain their required share of the CPU in each period. This is the same as ensuring that all threads meet their deadlines. We have developed a scheduling mechanism that is especially suitable for scheduling protocol processing threads efficiently.

The CPU and the adaptor execute concurrently.

The PDUs processed by a thread in each period are paced out by the adaptor within one period so that they leave the host before the next batch of PDUs are enqueued. The rate at which cells are paced out is determined by the mapping operation and the connection bandwidth is determined accordingly.

In current state-of-the-art implementations, the networking subsystem is implemented in the kernel and is accessed by user processes through a high level ipc interface (such as sockets). This model is referred to as the kernel resident protocol model. In this model, the kernel maintains state for all protocol sessions in protocol control blocks (PCBs). It schedules protocol operations, handles network events and moves data across protection domains for all sessions.

With the emergence of connection oriented networks such as ATM, it is possible to move network data directly to/ from the process space based on the connection identifier and so there is no need for the kernel to demultiplex data based on transport header fields. This allows protocol processing to be performed in the user process itself and so the protocol code can be part of the user program. This model is referred to as application level protocol (ALP) model. In this model, protocol code maintains the PCBs only for the sessions involving the process. It also handles protocol events such as packet arrivals and time-outs for these sessions. Therefore the user-kernel boundary is now at the network interface driver level rather than at the ipc level.

The advantages of the ALP model is that there is no central agent (such as the kernel) that handles all protocol activity. Besides it has the flexibility that comes with user level implementations. However, the main implication of the ALP model is that it leverages the existing resource management paradigm that treats the process as a unit for accounting and resource allocation, to solve the problem of providing QoS guarantees. Accordingly, the ALP model locates protocol processing within each process (or thread). By controlling the resource allocation to each thread, we can control the QoS offered to each protocol session. This approach cleanly separates the two concerns of implementing the protocol specific code, and managing resources on a per connection basis in accordance with the QoS requirements of each connection. The latter is automatically taken care of by the os. This allows the protocol implementor to be unaffected by the platform specific resource management details.

While the ALP model seems attractive for the above reasons, attention must be paid to implementation details so that it is as efficient as state-of-the-art kernel implementations. For example, a naive ALP implementation leads to increase in data movement and context switching, which would compromise the performance and thus applications' ability to use high bandwidth networks. We have designed appropriate implementation techniques for ALP which would ensure that the ALP can achieve at least as good performance as existing implementations, while also providing QoS guarantees. For details see [25, 26]

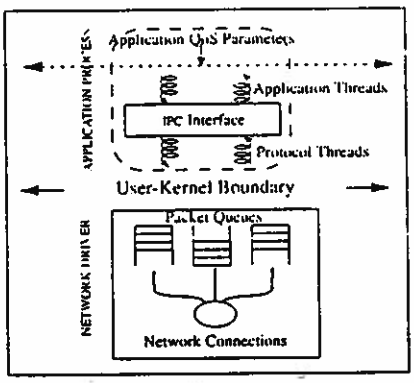


Figure 20: Application Level Protocol Implementation Model

Designing Robust Protocols using Counter Flushing

G. Varghese

A protocol is *self-stabilizing* if when started from an arbitrary global state it exhibits "correct" behavior after finite time. Typical protocols are designed to cope with a specified set of failure modes like packet loss and link failures. A self-stabilizing protocol copes with a set of failures that subsumes most previous categories, and is robust against transient errors. Transient errors include memory corruption, as well as malfunctioning devices that send out incorrect packets. Transient errors do occur in real networks and cause systems to fail unpredictably. Thus stabilizing protocols are attractive because they offer *increased robustness* as well as *potential simplicity*. Self-stabilizing algorithms can be simpler because they use uniform mechanisms to deal with many different kinds of failures.

In 1993, we have worked on inventing and refining a powerful new technique, called *counter flushing* that can be used to make protocols self-stabilizing. We have a leader[5] that wishes to broadcast a sequence of messages to every node in the network. Correct executions of the protocol can be partitioned into an infinite number of cycles: in each cycle the leader sends a message exactly once to all network nodes. Cycle n begins after cycle $n-1$ ends. We present a brief overview below. More details can be found in [61].

Token Passing on Rings

Our first example may be a reasonable alternative to the FDDI and IBM Token Ring protocols Usually the relevant state of each node is a boolean flag that indicates whether the node has the token. When a node finishes sending data, it clears this flag and sends a *token packet* to its downstream neighbor. However, in a stabilizing setting, the token protocol can deadlock (if a token is ever lost) and can livelock (if two more more tokens are present). To make a token ring protocol stabilizing, we augment the state of each node i with a counter c_i; we also add an extra counter field to each token packet. A counter is simply an (perhaps 32-bit) integer.

In Figure 21 the lower left hand corner (configuration D) describes a good global state of our protocol. There are four nodes which we call North, East, South and West and all packets travel clockwise around the ring. North is the leader. Each node other than North has a counter value of 8 and there is a token carrying the value 8 in transit from West to North.

Each node periodically retransmits its counter value in a token packet downstream. Thus mere receipt of a token packet is not enough for a node to assume it has the token. Instead when any node i receives a token from its upstream neighbor, node i does the following. If i is not the leader and the counter value in the token (say c) *is different* from the counter stored at node i (i.e., c_i), then node i assumes it has received a *valid token* and sets c_i equal to c; if $c = c_i$ and i is not the leader, i ignores the received token. If i is the leader, however, a different rule is used: if the counter c in the token *is equal* to the leader's local counter, then the leader assumes it has received a *valid token*, and increments its counter value mod 2^{32}; if $c \neq c_i$, then the leader ignores the received token.

Consider legal configuration D of Figure 21. In this state all token packets on links have a counter value 8, and the counter values at all nodes except North is also 8. The counter value at North is $9 \neq 8$. In that case, we say that North has the token. Eventually North will transmit a token packet containing 9. When this packet reaches East, East sets its counter value to 9. This process continues with the token moving clockwise until West receives the token and transmits it to North. North chooses a new value (10) and the cycle continues.

[5] Self-stabilizing protocols that calculate a leader in time proportional to the network diameter are known

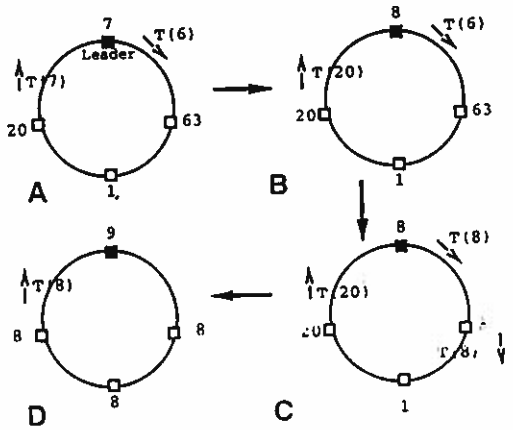


Figure 21: Progress to Stabilization in a Token Ring with Counters. Starting with an arbitrary state (A) we reach a state in which the leader has a fresh value (B). The fresh value (i.e., 8) moves round the ring (C) until we reach a legal state (D)

In legal states the ring can be partitioned into two bands. The first band starts with the leader and continues up to (but not including) the first counter value (either in a token packet or at a node) whose counter value is different from that of North. The remainder of the ring (including links and nodes) is a second band containing a counter value different from North. The valid token is at the boundary between the two bands.

The protocol stabilizes to legal states regardless of *initial values of node and packet counters.* Assume that c_{max}, the number of distinct counter values stored in nodes and links in the initial state, is less than the counter size Max (e.g., 2^{32}). For example, with a 1000 node ring transmitting at 100 Mbp/s and assuming 10 mile links, $c_{max} = 31,000$. The stabilization argument is illustrated in Figure 21. Informally:

- **In any execution, North will eventually increment its counter.** Suppose not. Then North's value will move around the ring until North gets a token with a counter value equal to its own.
- **In any execution, North will reach a "fresh" counter value not equal to the counter values of any other process.** (see Figure 21, configuration B). In the initial state there are at most c_{max} distinct counter values. Thus there is some counter value say m not present in the initial state. Since North keeps incrementing its counter, North will eventually reach m; in the interim no other process can set their counter value to m since only North "produces" new counter values.
- **Any state in which North has a fresh counter value m is eventually followed by a state in which all processes have counter value m.** (see Figure 21, configurations C,D). The value m moves clockwise around the ring "flushing" any other counter values, while North remains at m.

Define a *rotation time* equal to $2N$ time units (i.e., the time it takes for a packet to travel around the ring with a unit delay at each node and link.) The worst-case stabilization time of this protocol is equal to 2 rotation times. Here is the intuition.

Consider an execution with initial state s_i and some state s_f that occurs 1 rotation time later. Let the counter value of the leader in s_i and s_f be $c(i)$ and $c(f)$ respectively. In one rotation time, there is enough time for information from the leader to "flow" through the entire ring.

Thus all node counters in state s_f must have been "produced" by the leader since the execution began. More formally, in s_f, all counter values are in the range $[c(i) \ldots c(f)]$. If $c(f)$ is fresh, we are done (see stabilization argument) 1 rotation time later. Otherwise, the next leader increment will cause a fresh value because $c(f) + 1 \notin [c(i) \ldots c(f)]$. (This is guaranteed since $c_{max} < Max$). But the next increment will happen after at most one ring rotation time.[6]

Existing token passing protocols recover from lost tokens using global timers that are refreshed whenever a token is seen. In the IBM token ring the monitor (i.e., leader) uses a timer that is set to the longest possible delay it can take for a token to traverse the ring. When this timer expires, the monitor reinitializes the ring. Thus the recovery time of the IBM token ring protocol *is proportional to the worst-case delay around the ring.* The recovery time of our protocol is *is proportional to the actual delay around the ring, which can be an order of magnitude faster.* Self-stabilization also takes care of initialization. In practice, we would choose $c_{max} = 2^{32}$ as an architectural constant that should suffice for the largest size ring.

Actual token protocols are complicated by a requirement that each node in the ring only stores 1 bit of an incoming message or token before forwarding it on. If *every* node were to store a complete token or message before forwarding it, this would enormously increase the latency around the ring. To apply counter flushing to a real ring, we propose adding an extra delay to *only* the monitor node. This delay is sufficient to receive an entire counter; the delay allows the monitor to check if a received counter is equal to the monitor's counter and possibly increment the counter. All the other operations for counter flushing at other nodes can be implemented with 1-bit delays; details will be described in a future paper. Note that adding (say) a 32-bit delay to *only* the monitor node will not significantly affect the performance of the protocol.

Our protocol is a message passing version of a shared memory protocol due to Dijkstra. While our work is a significant extension of Dijkstra's work even for token passing, our major contribution is abstracting the mechanism and applying it to many other important examples, as we show below.

More Applications

Consider a pair of neighbors connected by a pair of unidirectional links. This topology can be considered to be a two node ring. Both Request-Response and Data Link protocols can be simulated using token passing on a two node ring.

For a general graph with a leader, we can easily construct a spanning tree and use this to construct a *virtual* ring. A virtual ring that spans all links is a cyclic path that starts and ends at the leader and in which each link is visited at least once. Virtual rings can be used for token passing in a general graph. More interestingly, a virtual ring can be used for *deadlock detection.*

Next, in Propagation of Information with Feedback (PIF), the leader wishes to broadcast a sequence of data items to all nodes in a tree; only after the i-th item is broadcasted to all nodes is the $i+1$-st value broadcasted. The tree is rooted at node r. Without stabilization, it is easy to solve this problem. The root sends a token packet with a new data item to all its children; other nodes accept new values only from their parents, upon which they send the value to their children. When a leaf node gets a new data item, it acks its parent. Nodes other than the root send an ack up to their parents, when they have received acks from all children. When the root receives an ack from all children, the root starts a new broadcast cycle.

To make the protocol stabilizing, we add a counter to the state of each node and to each packet. When the root starts a new broadcast cycle, it chooses a new counter and sends the

[6] This seems to imply that it takes 3 rotation times for this case; however, a more careful accounting shows that 2 rotation times suffice.

new counter along with the new data item to its children. A non-root node i accepts a data item from its parent only when the data item is tagged with a *different* counter value from i's stored counter. A node i accepts an ack only when the counter in the ack is *the same* as i's stored value. Each node periodically retransmits its most recent data item and the associated counter value to its children, and also retransmits acks. This can be used of a very fault-tolerant topology update protocol.

The technically hardest part of this work is the extension of of counter flushing to general graphs. The main difficulty is deciding how to reply to token packets received on cross links. We use this technique to design what is probably the simplest self-stabilizing protocol to reset a network protocol to an arbitrary state. Details can be found in [61].

Network Design and Routing

Near Optimal Network Design

J. Andrew Fingerhut, Jonathan Turner

In our last progress report, we reported on a new initiative aimed at developing better methods to configure cost-effective ATM networks that deliver good performance to end users. Our approach is based on a novel way of specifying traffic requirements that better reflects the limited information that network managers typically have at their disposal. Using estimates of the maximum traffic entering and leaving either individual switching systems or clusters of switching systems, we can configure networks that never block for traffic that operates within the bounds imposed by the estimated traffic. Since our last progress we have obtained a number of new results relating to these efforts and have obtained a deeper understanding of the strengths and limitations of this approach. In addition, we have obtained a three year grant from NSF to explore these questions in more detail.

There is a variety of ways one can formulate the network design problem. Different formulations focus attention on different aspects of the problem, bringing certain features to the fore, while suppressing others. For ATM network design, we prefer formulations in which we design networks that can handle an arbitrary set of user requests meeting specified constraints. That is, with respect to network traffic we take a worst-case point of view, rather than a probabilistic one. This is motivated by our observation that in ATM networks, the diversity of the applications and our limited understanding of how they will be used makes it difficult to obtain usable statistical predictors.

The simplest formulation of the problem is as follows. We are given a complete graph, $G = (V, E)$, where each vertex represents a switching system. For each vertex pair (u, v), we have a value $c(u, v)$ that represents the *cost per unit capacity* for installing a link from u to v. For each vertex u, we are also given a *source capacity*, $\alpha(u)$ and a *sink capacity*, $\omega(u)$ These give upper bounds on the total traffic that can originate or terminate at each vertex. Our objective is to assign a capacity $\gamma(u, v)$ to every vertex pair (u, v) that minimizes the cost of the network (defined as $\sum_{(u,v)} c(u, v)\gamma(u, v)$), while supporting any traffic configuration permitted by the source and sink capacities. A set of capacities that yields a network supporting all possible traffic patterns is termed a *nonblocking network*.

Given this formulation, one can obtain lower bounds on the cost of any nonblocking network for the specified capacities. We have shown that star networks (that is networks in which all vertices are connected through a single central vertex) are no more costly than any tree networks when the total source and sink capacities are balanced. We conjecture that a similar result holds for networks which are not constrained to a tree topology. In fact, simulation studies indicate that star networks achieve costs that are within a few percent of the lower bound on network cost and for problem instances based on random distributions of points in a Euclidean space, we can show analytically that the ratio of the star network cost to the lower bound approaches 1 as the number of points gets large.

Our basic formulation can be extended to situations in which there is a hierarchy of clusters and each cluster has its own source and sink capacity. This is useful for modeling situations in which groups of switches define "communities of interest" in which there is a lot of local traffic, but limited traffic outside the cluster. We have developed methods to compute lower bounds on nonblocking network costs when traffic is specified in this way using network flow and linear programming techniques. We have also carried out simulation studies that show that tree networks which follow the cluster hierarchy can produce results that are close to optimal. We expect that this can be shown analytically for random problem instances as well.

The fact that tree-structured networks can provide optimal or near-optimal cost-performance when traffic is expressed in terms of hierarchical source and sink capacities is surprising and very powerful. The simplicity of tree-structured networks makes them very attractive from an operational standpoint and allows immediate extension from point-to-point to multipoint traffic. However, tree-structured networks also have drawbacks, in that they may require switching systems with unrealistically large capacities and make large systems more vulnerable to failures. Consequently, we are also studying general networks with more general routing algorithms. One initial result is a negative one. We have shown that in general networks that use *shortest available path routing* (that is, new connection requests are routed using the shortest path available in the network at the time the request is made), it is NP-hard simply to determine if the network can block connection requests. It may still be possible to design optimal or near-optimal networks that are nonblocking and use shortest available path routing, but determining if a given network blocks is hard.

We have obtained results on approaches that are intermediate between tree-structured and shortest-path routing in general networks. In particular, for general networks using fixed path routing, we have developed algorithms to dimension links optimally for a given set of routes. In addition, we have defined a simple alternate path routing algorithm in which there is a pair of paths specified for each pair of endpoints, and have devised an algorithm for link dimensioning in networks that use this routing method.

In most of our analyses to date, we have assumed that the cost of a given amount of bandwidth between two switches is the product of a fixed cost per unit bandwidth and the amount of bandwidth required. This does not account for the fact that real links come in fixed bandwidth increments and the cost per unit bandwidth can vary depending on the bandwidth capacity of a given link. It also ignores the fact that a new connection with bandwidth B requires that one of the links joining a pair of switches has B units of bandwidth available. We have devised a straightforward procedure for converting a network designed under our naive assumptions to a realistic network design. In particular, given any network design in which the amount of bandwidth required between adjacent pairs of switches to achieve nonblocking operations is known, we can convert to a design using discrete link rates and which is nonblocking for new connections of rate $\leq B$ for some fixed B. We are now in the process of analyzing the cost of the realistic network relative to the original network. We are also studying our lower bound methods to determine if they can be extended to directly model link costs more accurately. We note that our results showing that star networks are optimal among tree networks is at least partly a consequence of these assumptions regarding link costs. We plan to explore how these results change when realistic link cost models are factored in.

Another direction we are currently exploring is distance-dependent source and sink capacities. Here, we specify for each switch in the network both its general source and sink capacities and its source and sink capacity with respect to other switches within a given distance bound (this can be extended to multiple distance bounds as well). Our lower bound methods can be extended to traffic requirements expressed in this way and we have begun to consider network designs that can produce cost-effective solutions for random point distributions in the plane. We believe that in some situations, models like this may better express the way traffic is distributed in networks than the hierarchical clustering models.

More details on this work can be found in [17, 18, 19, 20]

Routing in Broadband Networks

A.S. Maunder and P.S. Min

The purpose of this research is to develop and analyze routing schemes for multi-rate, multi-point traffic. Such a form of traffic is expected to be prevalent in the broadband networks that support a variety of services including high speed data, video, and other multi-media traffic as well as low speed voice and file transfer applications that are common in the telephone networks. Three distinct types of routing schemes, namely Random Alternate Routing (RAR), Least Loaded Routing (LLR) and Minimum Cost Routing (MCR), have been proposed by extending similar routing schemes that are well known in the circuit-switched network literature. An important difference to be noted in the broadband environment, however, is that the traffic in the broadband network is bursty and not periodic, and consequently, the routing policy must take into consideration of the effects of statistical multiplexing.

In the multi-rate environment of broadband communication, calls with different transmission rates may experience widely differing levels of the performance, and it is important to provide the ability to control the grade of service for individual call types, as well as the overall network performance aggregated across all call types.

Let k be the index corresponding to different call types supported by the network. For example, a call type may correspond to voice traffic that enters the network at a constant and known bit rate. Another example is compressed video with a rate of the transmission that is time variant. Such a call type may be described in terms of the maximum and the average transmission rates. In what follows, we use the term *OD pair* $[i, j]$ to denote the two nodes i and j in the network such that a call from OD pair $[i, j]$ is connected between i and j.

A quintessential measure for the performance of a network is the rate at which the reward is earned by the network, which is described by

$$\sum_k \sum_{[i,j]} w^k \lambda_{ij}^k (1 - L_{ij}^k)$$

where,

L_{ij}^k=Blocking probability for OD pair $[i, j]$ for call type k,

λ_{ij}^k=Rate at which call type k from OD pair $[i, j]$ arrives,

w^k=Reward of carrying one call of type k in the network.

The parameters w^k and λ_{ij}^k are given as the constraints in the network, and L_{ij}^k is determined by the nature of call assignments dictated by the routing policy denoted as R. We write such a dependency specifically, and L_{ij}^k becomes,

$$L_{ij}^k = g(R, X, \{\lambda_{ij}^k\}_{[i,j],k}).$$

X is the set of feasible states in the network. We describe the meaning of X shortly.

We adopt the popular notion of the states in the network such that state x illustrates the level of occupancy in the network. We define the set of feasible states X as the Cartesian product of the feasible states on all links in the network. Thus,

$$X = X_1 \times X_2 \times \cdots \times X_L,$$

where X_l, the set of feasible states on link l, is defined to be the occupancy of various call types that results in the performance measures (e.g., packet loss probability, packet delay) for link l within the prespecified values. Within X_l, calls are guaranteed of certain quality of service in using link l.

The tasks of determining X_l are addressed in the literature in the context of the admission control. There are numerous known admission control methods and the routing policies are developed independently of the specific types of the admission control.

With the definitions specified above, the routing policies considered are stationary Markov policies that map one feasible state in the network to another, upon the arrival of a call, with the measure of performance given as $\sum_k \sum_{[i,j]} w^k \lambda_{ij}^k (1 - L_{ij}^k)$ such that $L_{ij}^k = g(R, X, \{\lambda_{ij}^k\}_{[i,j],k})$ as before.

The optimal routing policy based on the above formulation is mathematically intractable due to the enormous cardinality in the set of feasible states in realistically sized networks. Rather than pursuing the optimality, the proposed RAR, LLR, and MCR routing policies are based on certain routing principles which are proven to be sound alternatives for the circuit switched, single-rate networks.

Next, we describe these routing policies very briefly. For the brevity of this abstract, we omit the description for the routing of multi-point calls; routing a multi-point call must consider the effects of sharing intermediate capacity in connecting to different end-points associated with the call.

Random Alternate Routing (RAR)

For OD pair $[i, j]$, assume that a set of paths, $\mathcal{R}[i, j]$, is specified. Amongst the paths in $\mathcal{R}[i, j]$, we choose one-link path (if it exists), as the most preferred path in establishing a call connection. If for some reason, an arrived call cannot be connected through this path, we choose one of the multi-link paths in $\mathcal{R}[i, j]$ randomly. All multi-link paths have the equal probability of being chosen. Once a multi-link path is chosen, the arrived call is carried on the path if by assigning the call, the resulting network state is still feasible. Otherwise, the call is rejected.

Least Loaded Routing (LLR)

LLR is similar to RAR in that an arriving call is first attempted on the one link path. The call is carried if the acceptance of the call results in a feasible network state. If the call cannot be carried on the one-link path, the call is attempted on the least loaded two-link path: the least loaded two-link path for OD pair $[i, j]$ is the one which minimizes $\max(I_{ib}, I_{bj})$ over b where b is an intermediate node connecting between OD pair $[i, j]$. I_{ib} and I_{bj} are the loaded-ness measures for link (i, b) and (b, j) respectively. LLR resolves any tie in determining the least loaded path randomly. Two simple criteria for determining the loaded-ness of a link considered are:

- Total Peak Rates: The loaded-ness of a link equals the sum of the peak rates of the calls that are present.
- Total Average Rates: The loaded-ness of a link equals the sum of the average rates of the calls that are present.

We also consider the capacity reservation schemes which give preference to one-link calls over two-link calls when a link is highly congested, by reserving a fraction of the link capacity for one-link calls. Different approaches to determine the level of capacity reservation are considered.

Minimum Cost Routing (MCR)

An arrived call is routed to the path with the minimum cost. No explicit preference is given to one-link paths. The cost of a path is the expected loss in the future reward by accepting the call just arrived. Thus, only the minimum cost path with the cost less than w^k is used if the arrived call is of type k. If there exists no path with the cost less than w^k, the call is rejected and cleared from the network. Unlike in the telephone networks where all calls are of the same transmission rate, finding a close form solution for the cost of a path is not possible in the multi-rate networks. In our derivation, we approximate the cost of a path, through the policy-value iteration evaluated exactly once from the initial routing policy that considers one-link path only.

Shadow Prices for State Dependent Routing

Paul S. Min

We demonstrate how to calculate shadow prices with respect to exogenous traffic for adaptive routing schemes. Consider asymmetric networks where $\mathcal{N}$ is the set of nodes, $\mathcal{L}$, the set of links, N, the number of nodes and $\mathcal{O}$, the set of Origin-Destination (OD) pairs. Each OD pair $[i,j]$ has a set of routes, $\mathcal{R}_{ij}$, from which one is chosen to transmit an arriving call: these routes consist of the single-link direct route (the link (i,j) with capacity C_{ij}) and two-link alternate routes. Links are assumed to be bidirectional and can transport traffic in either direction. Calls arrive for OD pair $[i,j]$ as a Poisson process with rate λ_{ij} and these processes are independent for different OD pairs. When a call arrives for one OD pair, it is either connected or blocked. If the call can be accomodated in a route chosen from the routing set according to a routing policy then it is connected; otherwise it is blocked. Any call connected on one of the links requires the use of one unit of the capacity available, i.e, one trunk.

In link state n, the arrival rate is $\alpha_{ij}(n)$, the offered traffic to link (i,j) when this is in state n, and the departure rate is equal to the number of calls connected on that link. Let $p_{ij}(n)$ denote the probability that link (i,j) is in state n and B_{ij} denote the blocking probability for OD pair $[i,j]$. As shown in [35], [42], the probabilities $p_{ij}(n)$ and B_{ij} can be obtained from a fixed point of the mappings from the $\alpha_{ij}(n)$'s to the $p_{ij}(n)$'s and from the $p_{ij}(n)$'s to the $\alpha_{ij}(n)$'s. Finally, we let $\mathcal{T}_{ij}$ denote the set of *intermediate* nodes on the alternate routes for OD $[i,j]$ and let Ω_{ij} denote the set of states for link (i,j).

Assume that for each accepted call on OD pair $[j,k]$, the expected revenue is w_{jk}. We first write the expression for the rate of return, where we consider $\underline{\lambda}$ as the vector of exogenous traffic to OD pairs, w_{ij} as the revenue generated by accepting a call on OD pair $[i,j]$ (when blocking probability is the performance measure, all the w_{ij}'s are 1), $\underline{p}$ as a vector which is obtained by the concatenation of the stationary probability vectors of each link, $\underline{\alpha}_{ij}$ as a vector containing the offered traffic for all the states of link (i,j) and $\underline{B}$ is the vector of OD pair blocking probabilities. Then we have

$$W(\underline{\lambda},\underline{B}) = \sum_{[j,k]\in\mathcal{O}} \lambda_{jk} w_{jk} \left(1 - B_{jk}(\underline{p})\right), \qquad (9)$$

$$\frac{dW(\underline{\lambda},\underline{B})}{d\lambda_{jk}} = \frac{\partial W(\underline{\lambda},\underline{B})}{\partial \lambda_{jk}} + \sum_{[r,s]\in\mathcal{O}} \left[\frac{\partial W(\underline{\lambda},\underline{B})}{\partial B_{rs}(\underline{p})} \frac{dB_{rs}}{d\lambda_{jk}}\right], \qquad (10)$$

$$\frac{\partial W(\underline{\lambda},\underline{B})}{\partial \lambda_{jk}} = w_{jk}\left(1 - B_{jk}(\underline{p})\right), \qquad (11)$$

$$\frac{dB_{rs}(\underline{p})}{d\lambda_{jk}} = \sum_{(a,b)\in S_{rs}} \sum_{n=0}^{C_{ab}} \frac{\partial B_{rs}(\underline{p})}{\partial p_{ab}(\underline{\alpha}_{ab},n)} \cdot \frac{dp_{ab}(\underline{\alpha}_{ab},n)}{d\lambda_{jk}}, \qquad (12)$$

$$\frac{\partial W(\underline{\lambda},\underline{B})}{\partial B_{rs}(\underline{p})} = -\lambda_{rs} w_{rs}, \qquad (13)$$

where $W(\underline{\lambda},\underline{B})$ is the rate of return from the network and is written to show its dependence on $\underline{\lambda}$ and $\underline{B}$, $p_{ab}(\underline{\alpha}_{ab},n)$ is written to show explicitly its dependence on $\underline{\alpha}_{ab}$ and $B_{rs}(\underline{p})$ is written to show explicitly its dependence on $\underline{p}$.

The calculated shadow prices for a network with 4 nodes are shown in Figure 22. Figure 23 plots the total rate of return, $W(\underline{\lambda},\underline{B})$ in the network versus external arrival rate per OD pair for the 4-node network.

We use shadow prices to get the sum capacity in asymmetric networks, i.e., the maximum sum of exogenous arrival rates such that the blocking probability of each OD pair is less than or equal to some maximum blocking probability. To obtain, this define a constrained nonlinear optimization problem with objective

function being the rate of return and constraints being the blocking probabilities. The independent variables are the exogenous arrival rates, i.e., $\underline{\lambda}$. Let $\underline{\eta}$ be a vector whose value is the maximum blocking probability for each OD pair and let $\underline{0}$ be the zero vector. Then the optimization problem is:

$$max_{\underline{\lambda}} \quad W(\underline{\lambda}, \underline{B}) = \sum_{[j,k] \in \mathcal{O}} \lambda_{jk} w_{jk} \left(1 - B_{jk}(\underline{p})\right), \tag{14}$$

$$\underline{B} \leq \underline{\eta},$$

$$\underline{\lambda} \geq \underline{0}.$$

The solution for the above optimization problem gives the maximum traffic that the network can carry for a given blocking probability vector. The optimization is achieved by using the shadow prices in a gradient descent algorithm that gives the direction in which the vector of exogenous arrival rates has to be varied to get the desired maximization.

The solution to the optimization algorithm allows to compare different adaptive routing schemes in terms of their sum capacity. The evaluation of the sum capacities for the 4-node network is given Figure 24.

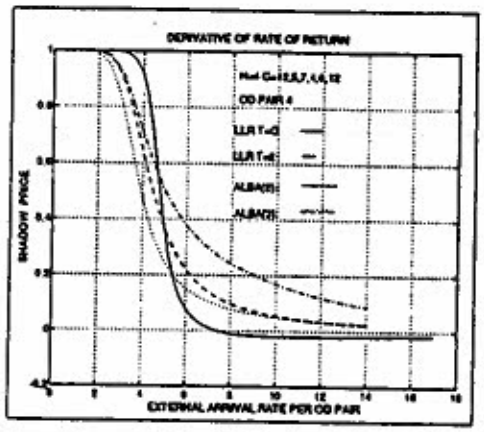


Figure 22: Shadow Price for 4-node Network with respect to External Arrival for OD pair [1,2]

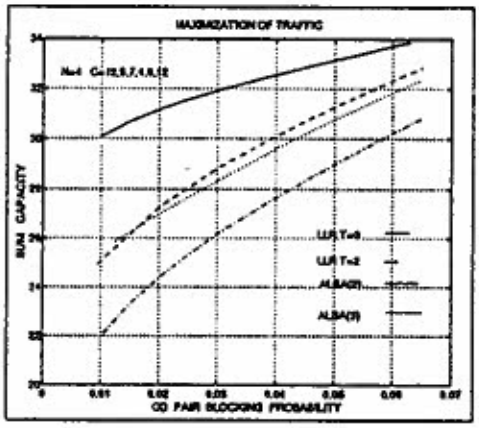


Figure 23: Optimization of Sum Capacity in the 4-node Network using LLR and ALBA

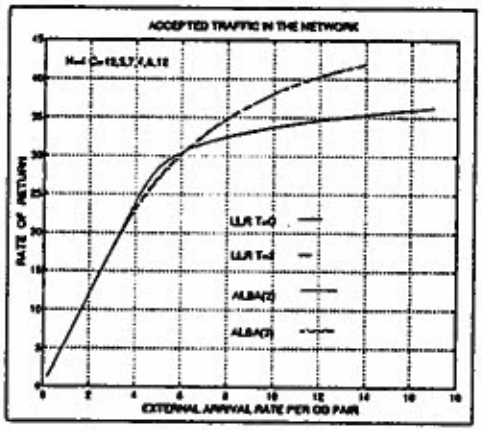


Figure 24: Total Accepted Traffic in the 4-node Network

Capacity Expansion in State Dependent Routing

A. Rayes and P.S. Min

This research is devoted to developing a method for capacity expansion for the State Dependent Routing (SDR) in circuit-switched networks. Suppose that a network is given by $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ where $\mathcal{G}$ employs a known routing policy which may or may not be an SDR. Let $\mathcal{S} = \{s_{ij}, (i,j) \in \mathcal{L}\}$ be the set of link capacities and $\Lambda = \{\lambda_{ij}, [i,j] \in \mathcal{P}\}$ be the set of origin/destination (OD) pair traffic intensities. Based on the fixed point methods reported in the literature [42] [44], the overall performance characteristics of $\mathcal{G}$ are known including the network blocking, and direct and overflow traffic patterns.

The goal of capacity expansion is to determine new link capacities $\mathcal{S}' = \{s'_{ij}, (i,j) \in \mathcal{L}\}$ such that the blocking performance of the network satisfies the grade of service defined on $\mathcal{G}$, when the exogenous traffic changes to $\Lambda' = \{\lambda'_{ij}, [i,j] \in \mathcal{P}\}$. More precisely, the *optimal* capacity expansion can be described as,

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j)\in\mathcal{L}} C_{ij}(s'_{ij}, s_{ij}) \\ \text{subject to} \quad & \frac{\sum_{[i,j]\in\mathcal{P}} \lambda'_{ij} b_{ij}}{\sum_{[i,j]\in\mathcal{P}} \lambda'_{ij}} \le \bar{b}, \\ & b_{ij} = f_{ij}(\Lambda', \mathcal{S}'), \quad [i,j] \in \mathcal{P}, \\ & s'_{ij} \ge s_{ij}, \quad (i,j) \in \mathcal{L}, \end{aligned}$$

where $C_{ij}(s'_{ij}, s_{ij})$ is the cost of adding $(s'_{ij} - s_{ij})$ circuits to link (i,j), b_{ij} is the blocking probability for OD pair $[i,j]$, and $\bar{b}$ is the prescribed blocking probability that the network needs to satisfy.

$f_{ij}(\cdot)$ is a function of Λ' and $\mathcal{S}'$ which the routing policy determines. The expression for $f_{ij}(\cdot)$ is not known in a closed form for SDR and can be found only through the fixed point iterations, making the above minimization problem extremely difficult to solve numerically, if not impossible, even for small networks.

With Λ and $\mathcal{S}$ given, the quintessential measure of the network performance can be described by

$$W = \sum_{[i,j]\in\mathcal{P}} w_{ij}\lambda_{ij}(1 - b_{ij})$$

where w_{ij} is the reward of accepting one call from OD pair $[i,j]$.

$\frac{\Delta W}{\Delta s_{ij}}$ is a quantity that illustrates the efficiency of the capacity of link (i,j). It equals the change in W as one additional circuit is added on link (i,j). The usefulness of $\frac{\Delta W}{\Delta s_{ij}}$ in capacity expansion is easy to see: when the blocking probability of a network exceeds the prescribed value $\bar{b}$, it is necessary to provide more circuits. The link on which one additional circuit can make the greatest reduction in the blocking probability is link (i,j) with the maximum value of $\frac{\Delta W}{\Delta s_{ij}}$. Therefore, in performing capacity expansion, additional capacity can be added to the network, one circuit at a time, according to the order determined by the values of $\frac{\Delta W}{\Delta s_{ij}}$.

Unfortunately, evaluating $\frac{\Delta W}{\Delta s_{ij}}$ is a daunting task for SDR. We suggest, through some heuristics, a method of reducing the computational requirement in determining W and thus in evaluating $\frac{\Delta W}{\Delta s_{ij}}$. Consider link (i,j) with the occupancy distribution known as $p_{ij}(x_k)$ for $x_k = 0, 1, \ldots, s_{ij}$. Suppose that there is a function $g_{ij}(x_k)$ such that $|g_{ij}(x_k) - p_{ij}(x_k)|$ is small for the range of integer valued x_k where $p_{ij}(x_k)$ is significant. Furthermore, we insist that $g_{ij}(x_k)$ be in the form of

$$g_{ij}(x_k) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{-\frac{(x_k - \mu_{ij})^2}{2\sigma_{ij}^2}}.$$

Thus, $g_{ij}(x_k)$ is a Gaussian density function with the mean μ_{ij} and the variance σ_{ij}. Before we illustrate the usefulness of $\{g_{ij}(x), (i,j) \in \mathcal{L}\}$ in reducing the computational requirement in the fixed point

calculation, we state an important assumption we utilize regarding the link occupancy distribution. This assumption is consistent with the observations we made during the numerical studies and can aid us in simplifying the fixed point calculation in a significant manner. Specifically, we assume that for the values of the exogenous traffic near the nominal values $\{\lambda_{ij}, [i,j] \in \mathcal{P}\}$, a parameter of interest, ρ_{ij}, remains roughly constant for all $(i,j) \in \mathcal{L}$. ρ_{ij} is defined to be the ratio of σ_{ij}^2 to μ_{ij} which is analogous to the peakedness in the carried traffic in the context of the Fixed Alternate Routing (FAR) [23].

We conjecture that $\{\rho_{ij}, (i,j) \in \mathcal{L}\}$, which determines the shape of the Gaussian function $g_{ij}(x)$, remains constant near the nominal value of Λ, while Λ and $\mathcal{S}$ vary in the manner so as to keep the network blocking probability roughly constant. Our conjecture has been confirmed for a wide range of the network load and sizes we studied.

In LBAR, calls arriving from OD pair $[i,j]$ are attempted first for the one-link path (i,j), the occupancy distribution $F_{ij}(x) = P(X_{ij} > x)$ for which we found an excellent approximation with a miminal level of computation. With $F_{ij}(s_{ij}-1)$, the probability that the one-link path is not available for OD pair $[i,j]$, a two-link path consisting of links (i,k) and (k,j) is chosen such that

$$k = \text{Argmin}_{b \in \mathcal{N}_{ij}} \text{ Max}[X_{ib}, X_{bj}] \qquad (15)$$

where $\mathcal{N}_{ij}$ is the set of via nodes for OD pair $[i,j]$ and X_{ib} (or X_{bj}) is the number of busy circuits on link (i,b) (or (b,j)).

For two-link path $\{(i,k),(k,j)\}$, define

$$H_{ij}^k(x) = P(\text{ Max}[X_{ik}, X_{kj}] > x).$$

Then,

$$\begin{aligned} H_{ij}^k(x) &= P(X_{ik} > x \text{ or } X_{kj} > x) \qquad (16) \\ &= F_{ik}(x) + F_{kj}(x) - F_{ik}(x)F_{kj}(x). \end{aligned}$$

Define γ_{ij}^k to be the probability that an arrived call is assigned to $\{(i,k),(k,j)\}$. Then,

$$\gamma_{ij}^k = F_{ij}(s_{ij}-1) \qquad (17)$$
$$\sum_{x=0}^{Max[s_{ik},s_{kj}]} \left[(H_{ij}^k(x-1) - H_{ij}^k(x)) \prod_{b \in \mathcal{N}_{ij}-\{k\}} H_{ij}^b(x) \right].$$

where the ties are broken randomly. $H_{ij}^k(-1) = 1.0$ for convenience.

The amount of the *carried load* by path $\{(i,k),(k,j)\}$ due to OD pair $[i,j]$ is,

$$\lambda_{ij}\gamma_{ij}^k(1 - F_{ik}(s_{ik}-1))(1 - F_{kj}(s_{kj}-1)).$$

Let the *total carried load* by link (i,j) be denoted by $\tilde{\alpha}_{ij}$. Then,

$$\begin{aligned} \tilde{\alpha}_{ij} &= \lambda_{ij}F_{ij}(s_{ij}-1) \qquad (18) \\ &+ \sum_{[i,k] \in \mathcal{A}_{ij}} \lambda_{ik}\gamma_{ik}^j(1 - F_{ij}(s_{ij}-1))(1 - F_{jk}(s_{jk}-1)) \\ &+ \sum_{[j,k] \in \mathcal{A}_{ij}} \lambda_{jk}\gamma_{jk}^i(1 - F_{ij}(s_{ij}-1))(1 - F_{ik}(s_{ik}-1)) \end{aligned}$$

where $\mathcal{A}_{ij}$ is the set of OD pairs that share exactly one end node with OD pair $[i,j]$.

The Appoximate Fixed Point Algorithm

Assume $\{\rho_{ij}, (i,j) \in \mathcal{L}\}$ are known.

Step 1: Assign arbitrary values for $\{\mu_{ij}, (i,j) \in \mathcal{L}\}$.

Step 2: Determine $\{\sigma_{ij}, (i,j) \in \mathcal{L}\}$ by the relation, $\sigma_{ij} = \sqrt{\mu_{ij}\rho_{ij}}$.

Step 3: Determine $F_{ij}(x)$, $H_{ij}^k(x)$, γ_{ij}^k, and $\tilde{\alpha}_{ij}$.

Step 4: Determine new values for $\{\mu_{ij}, (i,j) \in \mathcal{L}\}$ such that

$$\mu_{ij} = \text{Argmin}_{\mu} \left| \tilde{\alpha}_{ij} - \sum_{x=0}^{s_{ij}} \frac{G\left(\frac{x-\mu}{\sigma_{ij}}\right) - G\left(\frac{-\mu}{\sigma_{ij}}\right)}{G\left(\frac{\infty-\mu}{\sigma_{ij}}\right) - G\left(\frac{-\mu}{\sigma_{ij}}\right)} \right|. \qquad (19)$$

Step 5: If the new values for $\{\mu_{ij}, (i,j) \in \mathcal{L}\}$ are close to the previous ones, stop. Otherwise, go to Step 2.

At the end of the iteration, the approximate blocking probability, $\tilde{b}_{ij}$, for OD pair $[i,j]$ becomes,

$$\tilde{b}_{ij} = F_{ij}(s_{ij}-1) \cdot \sum_{k \in \mathcal{N}_{ij}} \gamma_{ij}^{k} [1-(1-F_{ik}(s_{ik}-1))(1-F_{kj}(s_{kj}-1))]$$

and the approximate network blocking probability, $\tilde{b}$, is given by

$$\tilde{b} = \frac{\sum_{[i,j] \in \mathcal{P}} \lambda_{ij} \tilde{b}_{ij}}{\sum_{[i,j] \in \mathcal{P}} \lambda_{ij}}.$$

The Capacity Expansion Algorithm

All evaluations of the network blocking probability is according to the Approximate Fixed Point Algorithm. With the prescribed network blocking probability given as $0 < \bar{b} < 1.0$,

step 1 If $\tilde{b} \leq \bar{b}$, stop. Otherwise, calculate $\{\frac{\Delta W}{\Delta s_{ij}}, (i,j) \in \mathcal{L}\}$ with $w_{ij} = 1$ for all $[i,j] \in \mathcal{P}$.

step 2 Add one additional circuit on link (i,j) where

$$(i,j) = \text{Argmax}_{(s,t) \in \mathcal{L}} \frac{\Delta W}{\Delta s_{st}}. \qquad (20)$$

Go to Step 1.

References

[1] ATM Forum, "ATM User-Network Interface Specification," Version 3.0, September10, 1993.

[2] Bianchi, Giuseppe and Jonathan Turner. "Improved Queueing Analysis of Shared Buffer Switching Networks," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, August 1993, pp. 482-490.

[3] Buddhikot, M., Parulkar, G., and Cox, Jerome, R. Jr., "Design of a Large Scale Multimedia Storage Server," Proceedings of the INET'94/JENC5, Conference of the Internet Society and the Joint European Networking Conference, Prague, June, 1994.

[4] Buddhikot, M., Parulkar, G. and Cox, J., "Scheduling, Data Layout and Playout Control in a Large Scale Multimedia Storage Server," *Technical Report in preparation*, Department of Computer Science, Washington University in St. Louis, Aug. 1994.

[5] Chandra, A.E., Saidi, H., Yan, P.Y., and Zar, D.M., "Washington University Multi-Channel Switching (WUMCS) Project: Functional Requirements for Prototype Switch Version 1.0," *Technical Report*, Department of Electrical Engineering, Washington University, December, 1993.

[6] Chandra, A.E., Saidi, H., Yan, P.Y., and Zar, D.M., "Washington University Multi-Channel Switching (WUMCS) Project: System Architecture Document," *Technical Report*, Department of Electrical Engineering, Washington University, December, 1993.

[7] Chaney, Tom, Craig Horn and Brian Gottlieb. "Gigabit Switch Chip Description: The Switch Element," Applied Research Lab Working Note 94-06, 6/94.

[8] Chung, Kyonghoon, Saied Hosseini, Randy Richards and Brian Gottlieb. "Gigabit Switch Chip Description: The Input Port Processor," Applied Research Lab Working Note 94-09, 6/94.

[9] Cox, Jerome R. Jr., Mike Gaddis and Jonathan Turner. "Project Zeus: Design of a Broadband Network and its Application on a University Campus," *IEEE Network*, March 1993, pp. 20-30.

[10] K.C. Cox, J.D. DeHart, M.E. Gaddis and R.G. Bubenik. Connection Management Access Protocol (CMAP) Specification. Washington University, Department of Computer Science, Technical Report WUCS-94-21, Version 3.0. July 1994.

[11] K.C. Cox, "GBNSC: The GigaBit Network Switch Controller," Applied Research Laboratory, ARL Working Note ARL-94-12, Version 1.0, August 26, 1994.

[12] Dailianas, Apostolos and Andreas Bovopoulos. "Real-time Admission Control Algorithms with Delay and Loss Guarantees in ATM Networks." *Proceedings of Infocom*, 6/94.

[13] Dailianas, Apostolos. "A Congestion Control Framework for ATM Networks," Masters thesis, Washington University Computer Science Department, 5/94.

[14] Dittia, Zubin, Andy Fingerhut and Jonathan Turner. "A Gigabit Local ATM Testbed for Multimedia Applications: System Architecture Document for Gigabit Switching Technology," Applied Research Lab, Working Note 94-11.

[15] Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "Catching Up With the Networks: Host I/O at Gigabit Rates," Technical Report WUCS-94-11, Department of Computer Science, Washington University in St. Louis, 1994.

[16] Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "Using an ATM

Interconnect as a High Performance I/O Backplane," Proceedings of the Hot Interconnects Symposium, August 1994.

[17] Fingerhut, Andy. "Designing Communication Networks with Fixed or Nonblocking Traffic Requirements," Washington University Computer Science Department, WUCS-91-55.

[18] Fingerhut, Andy. "Algorithms for Designing Nonblocking Communication Networks with General Topologies" Washington University Computer Science Department, WUCS-TM-92-05.

[19] Fingerhut, Andy. "Approximation Algorithms for Configuring Hierarchical Nonblocking Communication Networks," *Proceedings 2nd International Conference on Telecommunication Systems*, 1994. Also WUCS-93-19.

[20] Fingerhut, Andy. "Approximation Algorithms for Configuring Nonblocking Communication Networks," Washington University Computer Science Department, Doctoral dissertation, May 1994.

[21] Flucke, Margaret, Saied Hosseini and Randy Richards. "Gigabit Switch Chip Description: The Output Port Processor," Applied Research Lab Working Note 94-07, 6/94.

[22] Giacopelli, Jim, Jason Hickey, William Marcus, W. David Sincoskie and Morgan Littlewood./h "Sunshine: A High Performance Self Routing Broadband Packet Switch Architecture." *IEEE Journal on Selected Areas in Communications*, 10/91.

[23] Girard, A., *Routing and Dimensioning in Circuit-Switched Networks*, Addison-Wesley Publishing Company, 1990.

[24] N.R. Gohel, G.J. Blaine, J.R. Cox, Jr., D.R. Fuhrmann, D.M. Balfe, D.L. Gray, and W.D. Middleton, "Considerations in the transmission and storage of JPEG-encoded medical video," *Proc. IS&T/SPIE Int. Symp. on Electronic Imaging: Science and Technology, High-Speed Networking and Multimedia Computing Conference*, San Jose, CA, February 6-10, 1994, vol. 2188, pp. 243-253.

[25] R. Gopalakrishnan and Guru. M. Parulkar, "Efficient Quality of Service Support in Multimedia Computer Operating Systems", Technical Report WUCS-94-26, Dept. of Computer Science, Washington University in St.Louis.

[26] R. Gopalakrishnan and Guru. M. Parulkar, "Application Level Protocol Implementations to Provide QoS Guarantees at Endsystems," *Ninth Annual IEEE Workshop on Computer Communications*, October, 1994.

[27] Guttag, K., Gove, R., and Aken, V., "A Single-Chip Multiprocessor for Multimedia: The MVP," IEEE Computer Graphics and Applications, pp. 53-64, Nov. 1992.

[28] J.D. DeHart, Dakang Wu, "Connection Management Network Protocol (CMNP) Specification," Applied Research Laboratory, DRAFT, Version 1.0, November 4, 1993.

[29] Hill, Rex. "Design Analysis of ATM Access Switch Architectures," Masters thesis, Washington University Electrical Engineering Department, 5/93.

[30] Lee, Tony T. "Non-Blocking Copy Networks for Multicast Packet Switching," *IEEE Journal on Selected Areas in Communications*, 1455-1467, 12/88.

[31] Lee, Tony T. "A Modular Architecture for Very Large Packet Switches," *IEEE Transactions on Communications*, 7/90.

[32] Maxemchuk, N. "The Manhattan Street Network," *IEEE Transactions on Communications*, 5/87.

[33] Melen, Riccardo and Jonathan Turner. "Nonblocking Multirate Distribution Networks," *IEEE Transactions on Communications*, 2/93.

[34] J.D. Miller, R.M. Uchanski, E.J. Frederick, A.F. Heidbreder, "Effects of video compression on visual language communication," *Proceedings of 1994 IEEE Workshop on Vidual Signal Processing and Communications,* Sept. 19-20, 1994.

[35] Hegde, M.V., Min, P.S., and Rayes, A., "State Dependent Routing: Traffic Dynamics and Performance Benefits" *Journal of Network and Systems Management*, June, 1994.

[36] Mirfakhraei, Nader. "VLSI Design of a Gigabit Switch" *Proceedings of IEEE Workshop on VLSI in Communications*, 9/93.

[37] Mirfakhraei, Nader. "Wafer-Scale Integration as a Technology Choice for High Speed ATM Switching Systems" *Proceedings of IEEE Conference on Wafer Scale Integration*, 1/94.

[38] Mirfakhraei, Nader. "Design of a CMOS Buffered Switch for a Gigabit ATM Switching Network" To appear in *IEEE Journal of Solid State Circuits*, 1994.

[39] Mirfakhraei, Nader. "Performance Analysis of a Large-Scale Switching System for High-Speed ATM Networks" To appear in *Proceedings of IEEE Globecom*, 11/94.

[40] Mirfakhraei, Nader. "A Wafer-Scale ATM Switching System based on the Manhattan Street Networks" *Proceedings of IEEE Conference on Wafer-Scale Integration*, 1/95.

[41] Mitra D., Gibbens R.J., Huang B.D. "State-Dependent Routing on Symmetric Loss Networks with Trunk Reservations, I.," *IEEE Transactions on Communications*, vol. 41, no. 2, pp. 400-411, February 1993.

[42] Mitra, D., Gibbens, R.J. and Huang, B.D., "Analysis and Optimal Design of Aggregated Least-Busy-Alternative Routing on Symmetric Loss Networks with Trunk Reservations." *International Teletraffic Congress, 13*, Copenhagen 1991.

[43] C. Ozveren, R. Simcoe, and G. Varghese, "Reliable and Efficient hop-by-hop flow control", *ACM SIGCOMM '94*, London, England, September 1994.

[44] Rayes, A., Min, P.S., and Hegde, M.V., "Analysis of State-Dependent Routing (SDR) and Modified SDR," *Washington University Technical Report*, WUEE 92-126.

[45] W.D. Richard, J.R. Cox, Jr., B. Gottlieb, and K. Krieger, "The Washington University Multimedia System." *ACM Multimedia Systems*, vol. 1, pp. 120-131, 1993.

[46] W.D. Richard, J.R., Cox, Jr., A.M. Engebretson, J. Fritts, B.L. Gottlieb, and C. Horn, "Production Quality Video Over Broadband Networks: A Description of the System and Two Interactive Applications," *Proceedings of the Gigabit Networking Workshop, GBN'94*, June 1994, Toronto, Ontario, Canada.

[47] Saidi, H., Min, P.S. and Hegde, M.V., "A Nonblocking Architecture for Broadband Multi-Channel Switching," *Technical Report*, Department of Electrical Engineering, Washington University, 1993.

[48] Tobagi, Timoth Kwok and Fabio Chiussi. "Architecture, Performance and Implementation of the Tandem Banyan Fast Packet Switch." *IEEE Journal on Selected Areas in Communications*, 6/91.

[49] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE*

Transactions on Communications, June 1988.

[50] Turner, Jonathan. "Resequencing Cells in an ATM Switch," Washington University Computer Science Department, WUCS-91-21.

[51] Turner, Jonathan, "Managing Bandwidth in ATM Networks with Bursty Traffic," *IEEE Network Magazine*, vol. 6, no. 5, 9/92, 50–58.

[52] Turner, Jonathan, "Bandwidth Management and Congestion Control Scheme for Multicast ATM Networks." U.S. Patent #5,176,556, January 1993.

[53] Turner, Jonathan, "Nonblocking Multicast Switching System." U.S. Patent #5,179,551, January 1993.

[54] Turner, Jonathan, "A Practical Version of Lee's Multicast Switch Architecture," *IEEE Transactions on Communications*, Vol. 41, No. 40, 4/93.

[55] Turner, Jonathan, "Queueing Analysis of Buffered Switching Networks," *IEEE Transactions on Communications*, vol. 41, no. 2, 2/93, pp. 412-420.

[56] Turner, Jonathan, "Packet Switch With Broadcasting Capability for ATM Switching." U.S. Patent #5,229,991, July 1993.

[57] Turner, Jonathan. "Progress Toward Optimal Nonblocking Multipoint Virtual Circuit Switching Networks," *Proceedings of the Thirty-First Annual Allerton Conference on Communication, Control, and Computing*, September 1993, pp. 760-769.

[58] Turner, Jonathan. "Data Packet Resequencer for a High Speed Data Switch," U. S. Patent #5,260,935, 11/9/93.

[59] Turner, Jonathan. "An Optimal Nonblocking Multicast Virtual Circuit Switch," *Proceedings of Infocom*, June 1994, pp. 298–305, Also, WUCS-93-30.

[60] Valdimarsson, Einir. *Performance Evaluation of ATM Switching Systems*, Doctoral dissertation, Washington University Department of Electrical Engineering, 9/94.

[61] G. Varghese, "Self-stabilization by Counter Flushing", *13th ACM Principles of Distributed Computing (PODC) '94*, Los Angeles. August 94.

[62] Yeh, Y. S., M. G. Hluchyj and A. S. Acampora. "The Knockout Switch: a Simple Modular Architecture for High Performance Packet Switching," *IEEE Journal on Selected Areas in Communications*, 9/87.

[63] Zegura, Ellen Witte. "Architectures for ATM Switch Systems." *IEEE Communications Magazine*. 2/93.

[64] Zegura, Ellen Witte. "Evaluating Blocking Probability in Distributors," *Proceedings of Infocom*, 3/93.

[65] Zegura, Ellen Witte. "Analysis of Switching Networks for Multipoint and Multirate Communication," Doctoral Dissertation, Washington University Computer Science Department, 8/93.

[66] "Analysis of Blocking Probability in Multipoint Switching Networks." *Proceedings of Infocom*, 6/94.