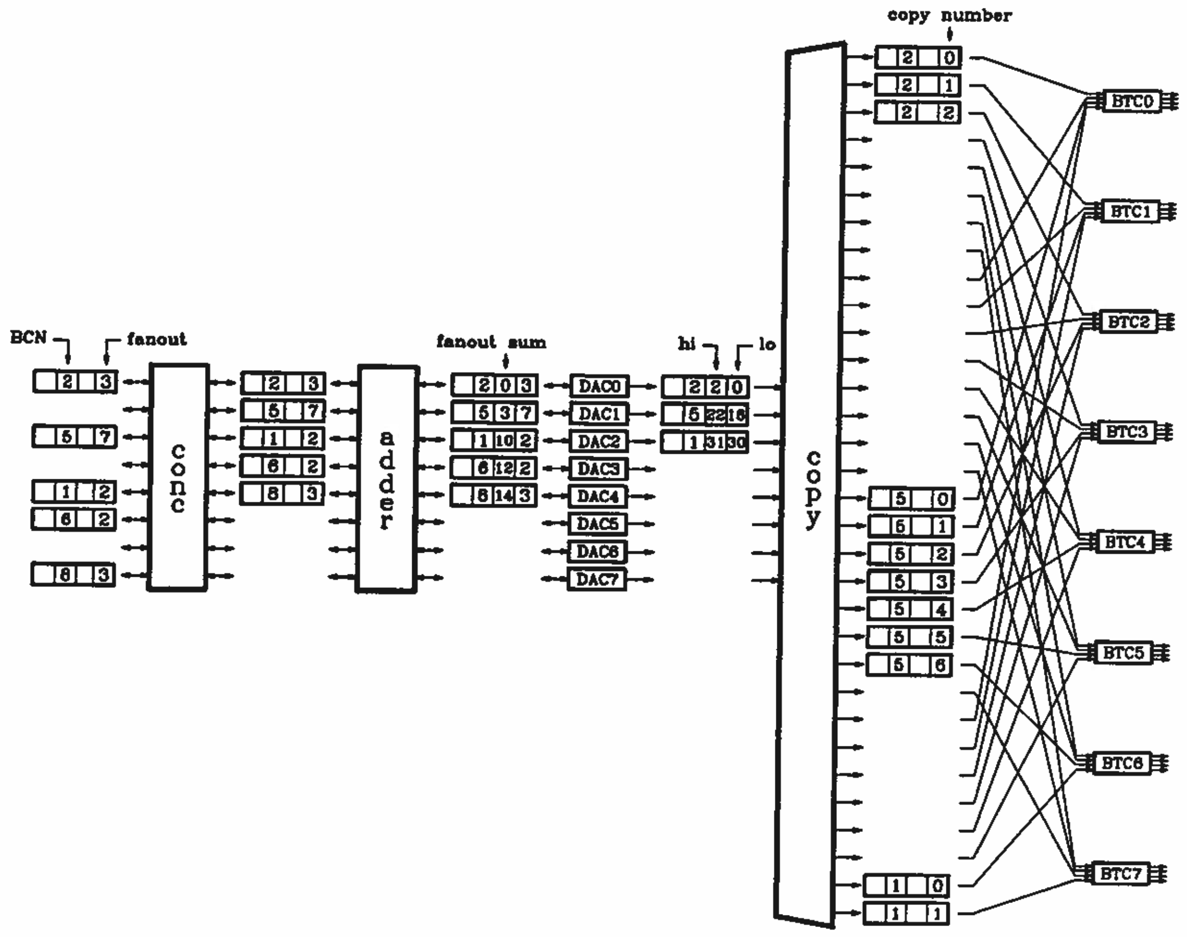
Advanced Networks Group

Research Summary, 1990



Computer and Communications Research Center
Washington University, St. Louis

The Advanced Networks Group

The Advanced Networks Group (ANG) is concerned with new communication technologies that can support a wide range of different communication applications in the context of both public and private networks. Fast packet or ATM networks promise a far more flexible communications infrastructure than is currently available. ANG is particularly concerned with systems that are capable of supporting ubiquitous multicast communication, suitable for applications such as video distribution, voice/video teleconferencing and LAN interconnection. We are developing an experimental switching system supporting links operating at 100 Mb/s and have devised economical switch architectures that can support link speeds in excess of a gigabit per second and having total throughput exceeding a terabit per second.

Our work spans a variety of topics including switching system design and analysis, performance evaluation of switching systems and networks, multicast connection management, algorithms for multicast routing, buffer and bandwidth management in the presence of bursty traffic, internetworking of high speed networks, image and video compression and design of specialized computer-aided design tools. Our research program includes a strong experimental component, which currently centers on the development of a prototype fast packet switching system. We have developed five circuits to be used in this prototype system and plan to assemble a network of four switches to demonstrate applications of fast packet switching. The experimental work is a crucial element of the overall research program, exposing detailed issues not apparent in higher level studies and providing a strong focus for the other activities.

Faculty
- Jonathan Turner
- Andreas Bovopoulos
- Guru Parulkar

Doctoral Students
- Akira Arutaki
- Millind Buddhikot
- Andy Fingerhut
- Larry Gong
- Victor Griswold
- Diamantis Kotoulas
- James Sterbenz
- Einir Valdimarsson
- Ellen Witte

Masters Students
- Haifeng Bi
- Charles Cranor
- Zubin Dittia
- Gaurav Garg
- Rex Hill
- Sanjay Kapoor
- Nader Mirfakhraei
- Tony Mazraani
- Christos Papadopoulos

Visitors
- Lorenzo Favalli

Related Activities

ANG is part of the Computer and Communications Research Center (CCRC), an interdepartmental research center that spans the departments of computer science and electrical engineering. In addition to the work in networking and communication work CCRC carries out research activities in parallel computer architecture, parallel algorithms, design automation and performance analysis and modeling. Recent activities have centered on the application of parallel computers to logic simulation and the associated problem of task allocation. This has included development of parallel simulated annealing algorithms and their application to task allocation for logic simulation.

The Center is headed by Dr. Mark Franklin and includes Dr. Roger Chamberlain, Dr. Ken Wong and several graduate students in addition to those in the Advanced Networks Group. The Center's facilities include three Sun file servers and over twenty workstations, a variety of software for VLSI circuit design, software development and performance modeling. In addition, a 64 processor NCUBE multiprocessor is available for work on parallel algorithms and a well-equipped electronics laboratory provides facilities for assembly and testing of systems. Other facilities available to the Center include a Pixar image computer and a video production facility.

The Computer Science Department's Applied Research Laboratory is a closely allied organization that carries out projects with the objective of transferring faculty research results into industrial practice. ARL currently has a full-time engineering staff of ten, including three long term visitors, and is engaged in a project whose objective is to demonstrate the potential of the broadcast packet switching technology, developed by the Advanced Networks Group. The project involves construction of a network of four switches at different locations in the St. Louis area, and demonstration of their use in support of video and medical imaging applications. The project also has a research component centering on algorithms and systems for video and image compression, and a second research component centering on image workstations for medical applications. Faculty and graduate students in Computer Science, Electrical Engineering and the Mallinkrodt Institute of Radiology are involved in various aspect of these activities. The project is supported by Southwestern Bell Corporation and NEC America.

ARL Staff

Jonathan Turner – Acting Director

Hardware Design Group
- Pierre Costa – Group Leader
- Neil Barrett
- Tom Chaney
- Noritaka Matsuura (NEC)
- Randy Richards
- Kenichi Sato (NEC)

System Software Group
- Mike Gaddis – Group Leader
- Rick Bubenik
- John DeHart
- Ken Katsumata (NEC)

Companies currently supporting the work of the Advanced Networks Group include:

- Bell Communications Research
- Bell Northern Research
- Digital Equipment Corporation
- Italtel SIT
- NEC America
- Nippon Telegraph and Telephone
- SynOptics Communications

Industrial Partnership Program

The Industrial Partnership Program (IPP) provides a mechanism for research interactions between the staff of the Advanced Networks Group and interested industrial organizations. Members receive technical reports and publications and have opportunities to meet with the research staff to discuss their activities. The program provides three levels of membership to meet the differing needs of program members.

Associates pay an annual fee that is currently $40,000 and receive the following benefits.

- Timely access to all technical developments, including one copy each of technical reports and annual progress reports produced by the Advanced Networks Group.
- The right to send two representatives to annual progress review meetings.
- A royalty-free license for internal use of all patents, software and hardware designs developed by the Advanced Networks Group and the right to commercially license patents and other technical developments of the Advanced Networks Group.
- A 20% discount for attendance at short courses offered by the Center.
- The right to send visitors to work at the Center for an additional annual fee of $20,000 per visitor.
- The opportunity to meet with and observe outstanding graduate students who may be candidates for industrial employment.

Program *Supporters* pay an annual fee of $60,000 and receive, in addition, the following benefits.

- The right to send four representatives to semi-annual progress review meetings.
- The right to republish ANG technical reports and distribute them within their organizations.
- The same licensing rights as above with the additional extension of such rights to Supporter's affiliates and subsidiaries. Furthermore, Supporters receive a credit against future royalty payments for all research funding.
- A 40% discount for attendance at short courses offered by the Center.
- The right to send visitors to work at the Center for an annual fee of $10,000 per visitor.

Program *Sponsors* pay an annual fee of $80,000 and receive the above benefits plus:

- The right to specify a mailing list of up to ten individuals who will receive all technical reports and annual progress reports produced by the Advanced Networks Group.
- The right to send any number of representatives to semi-annual progress review meetings.
- A 2:1 credit against future royalty payments for all research funding.
- A 60% discount for attendance at short courses offered by the Center.
- The right to send visitors to work at the Center with no annual fee.
- One 1–2 day consulting visit per year at Sponsor's location by Center staff.

Program members can reduce their annual fees by $10,000 per year by making a two year commitment to remain in the program.

Program Overview

The research program of the Advanced Networks Group includes a blend of experimental, analytical and theoretical work with a strong systems focus. We are committed to the proposition 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, in the absence of a strong connection to practical concerns, too easily drifts into activities of interest only to an ingrown research community, and is ultimately sterile and unprofitable. 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 program we have constructed, is based on this commitment to both engineering science and practice, and is unusual among university-based research programs in this regard.

Over the past several years, we have engaged in research activities focussing on packet switching systems operating at link speeds of about 100 Mb/s. This work has included the design and implementation of integrated circuit components and their use in experimental switching systems; performance evaluation of switching systems from a variety of different points of view; design of connection management protocols for multicast networks; design of multicast routing protocols; high speed internetworking and host-network interfaces; buffer and bandwidth management; distributed debugging systems; special purpose computer-aided design tools; and video coding algorithms. These activities are described briefly in the short articles that follow and more fully in the references. A prime distinguishing feature of our research activities has been our focus on the problem of multicast communication. The experimental switching system we are constructing will be one of the first high speed packet switching systems in the world that is capable of supporting large amounts of multicast communication and our research directed to understanding how to efficiently operate such systems is unique.

Our research agenda is now directed increasingly toward the design and analysis of switch architectures capable of supporting gigabit transmission rates. We are designing architectures capable of supporting such speeds using wide internal data paths and moderate speed circuit technology. We are also interested in switching systems suitable for high speed datagram switching, and in particular, the integration of datagram switching in a system supporting point-to-point and multicast connections. Another key element of our evolving research agenda is the application of high speed networks to computer-based applications, particularly those involving high resolution images and multimedia workstations. This work involves the design of high speed connection-oriented internet protocols, and the design of host interfaces and gateways capable of implementing them. We also have a continuing interest in the network control and optimization problems associated with emerging high speed networks. In particular, we continue to work toward better methods for bandwidth allocation and management in fast packet networks, as well as more efficient distributed algorithms for multicast connection routing.

Within the university we have recently initiated an effort called Project Zeus, which will apply the emerging ATM technology within a campus environment. This work will build on our earlier efforts, but will move toward the design and construction of operational networks, rather than research prototypes. The project will be carried out by the Applied Research Lab and the Office of the Network Coordinator (which operates the existing campus network) in cooperation with a number of industrial partners and collaborators throughout the university. We anticipate that Zeus will provide a rich source or research problems for ANG faculty and graduate students and a stimulating context within which to direct our activities.

The program provides ample opportunities for discussion and collaboration with IPP members. In addition to providing members with timely access to technical reports and publications, our regular progress review meetings are designed to stimulate interaction between members and ANG's faculty and students. Through these discussions, members have the opportunity to point out new research directions and help guide the research activities in order to ensure its practical relevance. Interactions with students and staff can also be promoted through extended visits by ANG staff to a member's location or by extended visits by member personnel to Washington University. Such interactions have been a crucial part of our research program with several of our visitors being key collaborators who played a strong role in our program, while providing their companies with deeper insight and understanding of ANG's research activities.

Design and Analysis of Switching Systems

Broadcast Packet Switching

Neil Barrett, Hai Feng Bi, Pierre Costa, Gaurav Garg, Tony Mazraani, Randy Richards, George Robbert, James Sterbenz

Broadcast packet switching is a novel approach to the problem of general multicast communication that was proposed and developed within the Advanced Networks Group. The switch architecture comprises a copy network which replicates packets associated with multicast virtual circuits, followed by a set of broadcast translation circuits that assigns a distinct destination to each copy and a routing network that guides the copies to their destination. Buffered switching networks with flow control are used to resolve contention and balance traffic dynamically.

The Applied Research Lab has constructed a prototype broadcast packet switching system using components that were developed within ANG. The prototype system supports link speeds of 100 Mb/s and fully implements the architecture's multicast capabilities. The organization of the prototype is shown in Figure 1. The system consists of several major components, a set of *Packet Processors* (PP) which provide the interface between the core of the system and the transmission links, a *Control Processor* (CP) which configures connections in response to signaling messages received from users and other switching systems, a *Copy Network* (CN) which performs the packet replication required for multicast applications, a set of *Broadcast Translation Circuits* BTC which perform a second stage address translation and a *Routing Network* (RN) which guides packets to the appropriate outgoing links.

The operation of the system is best illustrated by considering the flow of a packet through the system. When a packet is received at an incoming Packet Processor, the packet's *virtual circuit identifier* (which is stored within the packet's header) is extracted and used to select an entry from a routing table within the Packet Processor. The entry includes a *fanout* which specifies the number of outputs the packet is to be sent to and a secondary identifier called the *Broadcast Channel Number*. The fanout is used by the Copy Network to control the packet replication process. As the packet passes through the Copy Network it is replicated at various points and the fanout values in the replicated packets are halved. The replication process delays replication of a packet as long as possible to reduce loading in the early stages of the network. At those points where replication is not required, the packet is routed arbitrarily to one of the two outputs.

When the multiple copies of a packet reach the Broadcast Translation Circuits, the Broadcast Channel Number is used to extract an entry from a secondary routing table. This table provides an outgoing link number for the packet which is then used by the Routing Network to guide the packet to its ultimate destination. The contents of the routing tables in the Packet Processors and the Broadcast Translation Circuits are determined by the Control Processor, which can modify specific entries by means of control packets sent through the switching network. Control Processors in different switching systems can exchange signaling messages with one another to request establishment or modification of a user connection.

Integrated circuits have been designed to implement the various elements of the architecture described above. Each Packet Switch Element used within the copy and routing networks is a single integrated circuit designed in a 2 μm CMOS process; the number of transistors is approximately 45,000. Each one implements a two input, two output switch with eight bit data paths and hardware flow control. A clock rate of 25 MHz provides a data path speed of 200 Mb/s, which will support link speeds of 100 Mb/s without internal congestion. Each broadcast translation circuit is implemented as a single chip with

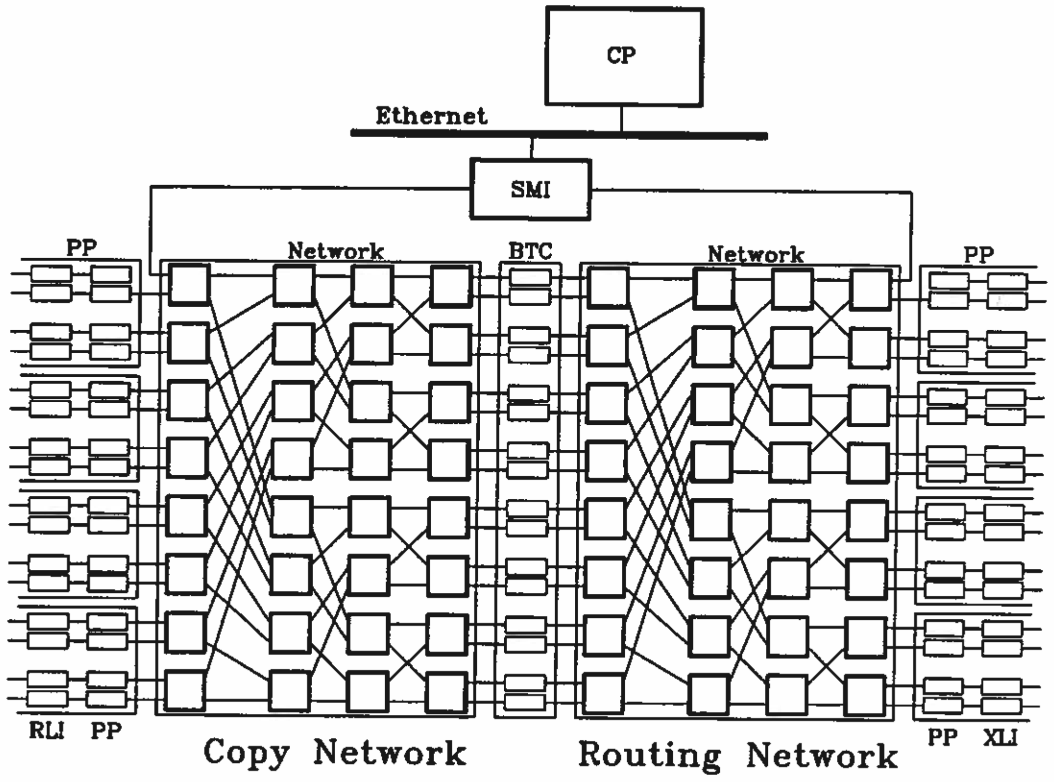


Figure 1: Prototype Broadcast Packet Switch

approximately 25,000 transistors. The packet processors include four chips each, one interfacing to the transmission link and providing both bit and packet level synchronization, another interfacing to the network and providing the routing translation function, and two others which buffer packets waiting to be sent to the copy network or the transmission link. These circuits have complexities ranging from about 40,000 transistors to over 200,000.

Our experimental 16 port switch consists of a total of eight circuit cards. The network cards each contain a complete 16 port switch with 32 packet switch elements. The physical organization of the system uses a conventional backplane configuration with high density connectors providing about 800 pins on each of the network boards. Each network board has 32 input and output links comprising 10 signals each, including parity and flow control. This gives 320 signal pins and another 320 signal grounds at the backplane connector.

The prototype switching system will be replicated during the summer of 1991 and used in a demonstration network consisting of four switches located throughout the St. Louis metropolitan area. This network will be used to demonstrate a variety of visual communication applications including video distribution, video conferencing and medical imaging. It will also provide a testbed for further application development and play an important role in the early phases of Project Zeus.

See references [6, 47, 57, 59, 89, 74, 75] for further details.

Design of a Second Generation Broadcast Packet Switch

Jonathan Turner, Rex Hill, Nader Mirfakhraei

Our experience with the broadcast packet switch architecture to date has convinced us that while the basic approach is sound, there are several areas in which improvements can be made. The use of larger switch elements can significantly reduce the chip count for the system as it is scaled to large sizes. Second, shared buffering within the switch elements can give higher system throughputs allowing the system to be operated with a smaller speed advantage. Third, by partitioning the broadcast channel number space, we can significantly expand the number of multicast connections that can be supported, without increasing the BTC memory requirements. Fourth, the combination of higher density circuit technology and different circuit design techniques can allow us to speed up the system, allowing it to be used with higher speed links. With these improvements, we estimate that a system supporting 256 links at 600 Mb/s each, can be built using just two chips per port in the copy and routing networks.

Figure 2 shows the organization of the second generation broadcast packet switch. As in the original architecture, it comprises a copy network, a set of broadcast translation circuits and a routing network. Now however, the copy and routing networks each have a Beneš topology and are constructed from large switch elements (16 ports in the figure) using shared buffering. The switch elements are implemented using a bit-sliced organization that maximizes the amount of circuitry per chip while minimizing the pin count. The shared buffering allows the system to operate with a smaller speed advantage than is possible in systems with input buffered switch elements. Performance studies indicate that a system with an internal data rate of 800 Mb/s can support external link rates of 600 Mb/s. The second generation switch divides the multicast connections into two classes, large fanout connections and small fanout connections. This division allows a given broadcast channel number to be used in conjunction with several small fanout connections, greatly increasing the number of multicast connections that can be supported.

We have initiated detailed design work for a chip set that can be used to implement the second generation switch. The target for this effort is a 16 port switch element with a shared buffer for up to 48 ATM cells and grant-based flow control. The switch element will support eight bit data paths and is being designed to operate at a clock rate of 100 MHz. We have decided to depart from the conventional two phase non-overlapping clock scheme, which has been used in our previous CMOS IC designs. Instead, we have adopted a true single phase clocking scheme which is less subject to skew problems and requires only two clock transitions per clock cycle instead of four, allowing higher speed operation. On the other hand, the single phase clocking scheme requires the use of complementary latches and careful control of clock skew between adjacent circuit elements.

The switch element consists of a control chip and a set of *data slices*, each of which accepts one bit slice from each of the 16 ports. The data slice comprises a set of shift-register buffers that store waiting cells and a pair of crossbars that connect the buffers to the inputs and outputs. The control chip has a buffer control circuit for each buffer in the data slice plus an arbitration circuit, which during each operation cycle, determines which buffered cells are to be sent to the outputs. The arbitration circuit uses a bit-serial distributed contention resolution circuit for each of the outputs. Each buffer requiring access to a particular output shifts its waiting time and address onto a serial contention bus. The bits are wire-or'ed by the bus and whenever a contender observes a "1"

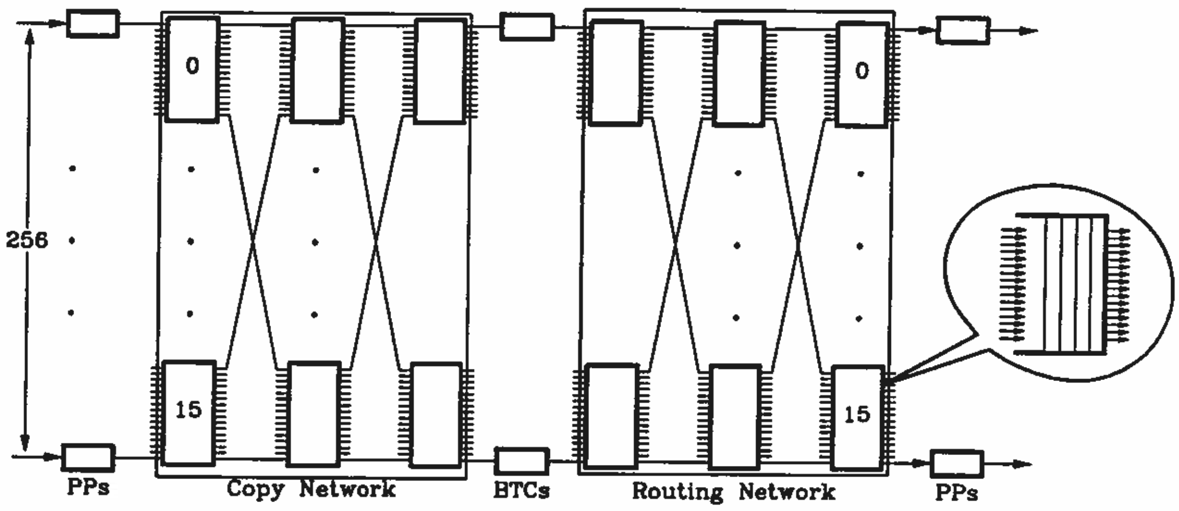


Figure 2: Second Generation Broadcast Packet Swtich

on the bus when it had transmitted a "0", it stops contending. As a result, at the end of the contention cycle, only one of the original contenders is still contending for the bus and the winning contender is one with the longest waiting time.

See reference [76] for further details.

Resequencing Packets in an ATM Switch

Jonathan Turner

ATM switching systems such as the broadcast packet switch have the property that packets passing through the system may follow different paths, allowing packets to exit the system in a different order from that which they entered. This can cause problems for applications which often require that packets be received in the same order as they were sent. We have devised an economical mechanism for reordering packets after they exit an ATM switch that effectively eliminates this problem. A patent application for this invention is being prepared.

To implement our mechanism, we require knowledge of the amount of time that each packet exiting the switch fabric has spent within the fabric. This can be accomplished with a simple time stamp field, added as the packet enters the fabric and extracted as the packet exits. When packets exit the fabric they enter an output buffer which is managed by a *Buffer Controller* (BC). As the packet leaves the switch, the BC computes its age from its time stamp and the current time and stores the age and the packet's location in the output buffer in an internal table.

When a packet is to be read from the buffer, the buffer controller selects the oldest packet in the buffer and provides its address to the output buffer which then sends it out. If the oldest packet is "not old enough," no packet is output. The purpose of this is to allow packets stored within the switching network to catch up with packets that have already reached the output buffer. This mechanism ensures that all but a vanishingly small fraction of misordered packets are put back in order.

The key element in this mechanism is the buffer controller. In our judgement, a reasonable buffer controller should have a complexity of no more than 25% of the buffer. The design we have devised can, with careful implementation, meet that target. The buffer controller has two main functions. First, during a buffer output cycle, it must select the packet with the oldest time stamp, check that the packet is old enough and if so, supply the packet's address to the buffer. Second, during a buffer input cycle, it must select an empty slot that the arriving packet can be placed in and supply its address to the buffer.

The buffer itself, is organized as a set of *slots* with each slot being large enough to hold a single ATM cell. The address of the cell in the buffer is simply the index of its slot. Figure 3 shows the organization of the buffer controller. It consists of several parts. At the right is a precharged wire-or contention bus used to select the slot to be read from or written to. For each slot in the buffer there is a *Slot Control Circuit* (SCC) and a *Bidirectional Shift Register* (BSR). The BSR contains the age of the packet (if any) stored in the corresponding buffer slot as well as the slot number for that buffer slot. The age of the packet is computed by the *Input Circuit* (IC) when the packet enters the buffer control circuit by subtracting its time stamp from the current time and it is maintained by incrementing it for each cycle that the packet remains in the buffer. The *Timing Generator* (TG) generates the signals needed to coordinate the activities of the other components.

During a buffer read cycle, each slot control circuit corresponding to an occupied slot in the buffer, shifts the time stamp stored in its shift register onto the contention bus, one bit at a time with the most significant bit first. At the same time it monitors the state of the contention bus to see if the bus state differs from the bit it shifted onto the bus. If the bus state does differ, the slot stops contending for the bus. Because of the wire-or nature of the bus, the slot containing the oldest packet will win the contention. Since there may be more than one oldest packet, each slot also places its slot number on the bus after the time stamp. This guarantees that there is a unique winner in the contention and provides the winner's slot

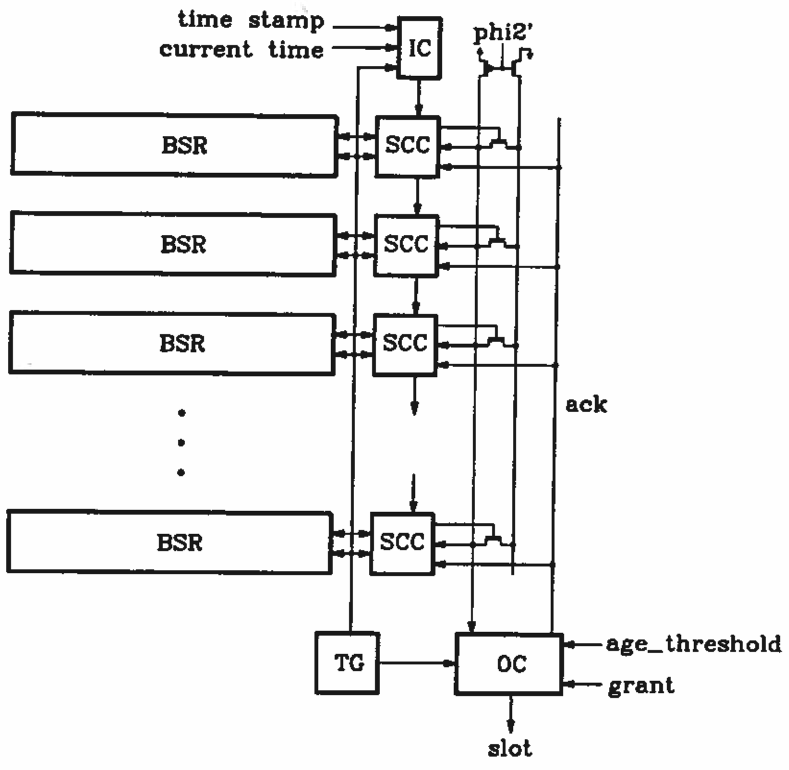


Figure 3: Buffer Controller

number to the *Output Circuit* (OC), which forwards it to the buffer, assuming that the packet is at least as old as the specified *age threshold* and that the downstream circuitry can accept it (indicated by the **grant** signal). The OC also sends an acknowledgement signal (**ack**) to the winning slot control circuit in this case.

During a buffer write cycle, each slot control circuit corresponding to an empty slot contends for the bus using its slot number alone as the contention key. The time stamps of waiting packets are also incremented during the buffer write cycle.

See references [84, 87] for more details.

Performance of Buffered Switching Networks

Haifeng Bi, Rick Bubenik, Jonathan Turner

We have studied the performance of buffered switching networks such as the broadcast packet switch from several different points of view. Our earliest work was an extensive simulation study that examined a number of key architectural trade-offs and was used to guide our initial implementation efforts. This work has been augmented with analytical studies that characterize the worst-case loading conditions and queueing behavior of a broad class of switching networks.

Our simulations showed that the performance of the broadcast packet switch is determined primarily by the routing network. We have shown that a system with six stages of binary switch elements, each containing two buffer slots per input, can achieve a throughput of about 55%; this is more than adequate, given a 2:1 speed advantage over the external transmission links. With a single buffer slot the maximum throughput is about 50% and with four slots it is about 60%. We have investigated the improvement obtainable using larger switch elements. Surprisingly, a reduction in throughput is observed when FIFO queueing is used in the switch elements, as a consequence of head-of-line blocking effects. When bypass queueing is substituted, we see a substantial improvement in throughput for both binary and larger switch elements, and the expected advantage for larger switch elements. Specifically, throughputs of about 63% can be achieved with binary switch elements and two buffer slots per switch element input; throughputs of 75% can be achieved using four port switch elements with comparable buffers.

The switch elements used in the broadcast packet switch employ *cut-through switching*, in which a packet is passed directly from the input to an output if the output is idle when the packet arrives. Our simulations show that that this yields about a 10% improvement in throughput, relative to the no cut-through case. More important, cut-through reduces the delay through the system substantially. For a 64 port broadcast packet switch with binary switch elements, the delay through the copy and routing networks is about 3 packet times under normal operating conditions, with cut-through and about 14 packet times without it.

In the copy network, congestion is caused by packet replication and is hence primarily dependent on fanout and the number of inputs generating traffic. We find that when all inputs are busy and fanouts are assigned randomly, the maximum throughput generally improves as the average fanout increases. This is due to the reduction in packet arrival rate required by increasing fanout in order to accommodate the limited capacity of the output links. Throughput reaches its lowest point when only a few adjacent inputs generate all the traffic. For deterministic fanouts a striking dependence emerges. Throughputs of 100% are achieved whenever the fanout is a power of two, then decreases sharply as the fanout increases past that point. This is attributable to the binary structure of the copy network which makes it most efficient when the fanout is a power of two.

We have developed an analytical method for evaluating the worst-case loading for switching systems like the broadcast packet switch in which packets are routed individually by the switching network (that is, packets in a given virtual circuit are not constrained to follow the same path). For example, we have shown that an extended delta network with h distribution stages and d port switch elements requires a speed advantage of $d^{\lfloor (k+h)/2 \rfloor}$ where $k = \log_d n$ in order to handle all possible traffic configurations. As a corollary, we have that an ordinary delta or banyan network requires a speed advantage of $\sqrt{n}$ if $\log_d n$ is even and $\sqrt{n/2}$ if $\log_d n$ is odd. Also, we have that a Beneš network requires no speed advantage. The result quantifies the improvement obtainable by adding distribution stages, allowing a designer to engineer the system to meet his needs without adding more distribution stages than necessary.

Our method applies to networks that perform packet replication, as well as point-to-point networks. One surprising result is that the worst-case performance of a copy network can deteriorate as the size of the switching elements is increased. In particular, the required speed advantage for a copy network with delta or banyan topology using d port switch elements is $d + 1$, if copies are sent to all outputs of a given switch element when copying. This disadvantage of large switch elements can be eliminated however if one takes care to make only the minimum number of copies necessary at a given stage in the network.

In a widely cited paper [42], Jenq described a method for analyzing the queueing behavior of binary banyan networks with a single buffer at each switch input. The method, while not yielding closed form solutions, does permit the efficient computation of the delay and throughput characteristics of a switch. A key element of the analysis is the inference of the state of a single switch from the state of its two buffers, based on the assumption that the states of the two buffers are independent. This independence assumption is not valid but does not yield gross inaccuracies in the systems that Jenq studied. More recently, Szymanski and Shaikh [67] have extended Jenq's method to switching systems constructed from switches with an arbitrary number of inputs and an arbitrary number of buffer slots. They have also applied it to systems with different buffering techniques.

We have extended the previous work to cover switching systems in which the buffer slots in a switch are shared among all the inputs and outputs, rather than being dedicated to either particular inputs or particular outputs. Such systems require an analysis which explicitly models the state of the entire switch rather than the states of individual input or output queues, from which one infers the state of the switch. We can also apply our method to systems using parallel bypass input buffering, a class of systems that cannot be analyzed directly using the previous methods. In addition, we have applied this approach to systems with fifo input buffering yielding significantly more accurate results in some cases.

We have also performed an extensive simulation study comparing systems constructed using input queueing and systems using output queueing. These studies have shown that the difference between fifo input queueing and conventional output queueing has less to do with the position of the queues than with the extra switch bandwidth implied by output queueing. Bypass input queueing equalizes this difference to at least some extent. In our study, we have compared a generalized form of input queueing, where each queue can send up to r packets during each cycle, with a generalized form of output queueing, where each queue can receive up to r packets during each cycle (the total number sent and received by a switch during one cycle is still at most d). Interestingly, there is very little difference between these two cases, but input queueing has a slight advantage due to "boundary conditions" involving the first and last stages of the network.

See references [5, 26, 27, 75, 80] for further details.

Blocking in Multirate Networks

Riccardo Melen, Jonathan Turner, Einir Valdimarsson

Multirate networks are a generalization of conventional circuit switching that can be used to model multirate time-division switches and ATM switching networks that use fixed path routing (that is, packets in a given virtual circuit are constrained to follow the same path). Multirate networks model several experimental switch architectures now under development around the world and hence our results are directly applicable to these systems.

An important parameter affecting the blocking characteristics of a given switching network is its *speed advantage*, which is defined as the ratio of the network's internal data path rate to the rate of its external transmission links. We have shown that any multirate network that has only a 1:1 speed advantage must have $\Theta(n^2)$ complexity in order to be strictly nonblocking. Given an appropriate speed advantage, many classical results can be generalized to the multirate environment. For example, we have shown that a multirate version of the three stage Clos network is nonblocking for exactly the number of middle stage switches as in the single rate case if the speed advantage is at least two. Similarly, the number of planes required to make the Cantor network strictly nonblocking is exactly the same as for the circuit switching case when the speed advantage is two. However, in multirate networks, we have the possibility of trading off the speed advantage against the network's topological complexity. Hence, a Beneš network, which can be viewed as a single plane Cantor network, is strictly nonblocking if we employ a speed advantage approximately equal to the logarithm of the number of inputs and outputs. In particular, an r stage Beneš network is strictly nonblocking when operated with a speed advantage of r. So for example, a three stage network comprising 32×32 switch elements which has 1024 inputs and 1024 outputs is strictly nonblocking with a 3:1 speed advantage. We have also determined the conditions under which various networks are rearrangeably nonblocking. As an example, the three stage Beneš network mentioned above is rearrangeably nonblocking when the speed advantage is at least two. In general the speed advantage required for rearrangeable operation is proportional to $\log\log n$, where n is the number of inputs and outputs.

This work also extends to multipoint networks, that is switching systems that distribute a signal from an input to multiple outputs. We have derived the conditions that lead to nonblocking operation for single rate networks due to Ofman and Thompson and Pippenger. The three stage Beneš network, when used for multipoint traffic, requires a speed advantage of approximately $\sqrt{n}$ in order to be nonblocking; the situation is even worse for five stage networks. This has serious implications for several experimental systems now under development which use a Beneš topology. Such systems are likely to experience high blocking probabilities in the presence of substantial amounts of multipoint traffic if operated with little or no speed advantage. We have shown however that the blocking characteristics of a Beneš network with respect to multipoint connections can be improved by adding extra distribution stages. In particular, a pair of Beneš networks cascaded together are wide sense nonblocking with respect to new multicast connections.

While nonblocking networks allow us to avoid blocking completely, they do so only at the expense of a potentially large speed advantage. This can substantially increase system cost and may not always be justified. For this reason, we are also interested in determining the likelihood of blocking in blocking multirate networks.

We have developed an analytical method to evaluate the blocking probability in multirate systems. The method is based on the well-known static models of Lee and Pippenger for space division networks. In these models, we let p be the probability that one of the switch's internal data paths is busy; Lee assumes that the busy/idle probabilities for different links are completely independent, while Pippenger assumes independence for network inputs and outputs and assumes random routing, but correctly accounts for the dependencies among links incident to a given switch within the network.

In a multirate network, a switch's internal data paths are not simply busy or idle but may have some fraction of their bandwidth in use. We extend Lee's and Pippenger's models by assuming a link occupancy distribution and letting p_δ be the probability that a link's occupancy exceeds $1 - \delta$. Using p_δ in place of p, we can apply Lee or Pippenger's method to multirate networks to determine the probability that a connection of weight δ will block. We obtain an approximate link occupancy distribution from a given connection bandwidth distribution by solving a single link blocking problem. This technique is computationally straightforward and useful for comparison purposes, but does tend to overestimate the probabilities of high link occupancies.

We have also performed a simulation study assessing blocking in multirate Beneš networks. Our dynamic simulator model uses exponential interarrival times and exponential holding times for connections. It also allows for several different connection classes and routing algorithms. As one would expect, the blocking increases with increasing offered load in the network, and the blocking causes the carried load to deviate from the offered load. For high load, links are usually not fully utilized, and fragmentation occurs.

Our results on how blocking is affected by network parameters such as switch element size, number of inputs and number of stages, are very interesting. Blocking is reduced drastically by increasing the size of the switch elements but it is surprisingly unaffected by increased number of inputs or stages. By operating with a speed advantage compared to the external communication links, we can reduce the load and therefore the blocking in the network. Melen and Turner [52] have shown that Beneš networks are nonblocking with a speed advantage which is a logarithmic function of the number of inputs. Our simulations have shown that if we are willing to accept some small but nonzero blocking probability we can get by with a substantially smaller speed advantage and one that is virtually independent of the network size.

In a mixture of connections of different bandwidth we found that the usual blocking probability can be misleading as it neglects the bandwidth differences among blocked connections. Consequently, we have adopted a new measure called the *weighted blocking probability*. If $p(w)$ is the density function for the probability of connections of weight w and $b(w)$ is the density function for the probability of blocking connections of weight w then the weighted blocking probability is defined by $\cap_0^1 wp(w)b(w)dw$.

We have tried using several routing algorithms to reduce the overall blocking and also to equalize the blocking for different bandwidth classes. The idea of equalization is to reduce the blocking of a large bandwidth class at the expense of a small bandwidth class. We conclude that a simple packing algorithm offers the best overall performance. Segregation of different bandwidth classes can work well in some cases but does not usually perform as well as packing. Reserving bandwidth for large bandwidth classes has little effect unless a lot of bandwidth is reserved, leading to unacceptably high blocking for the other classes.

See references [52, 53, 54, 90, 91] for further details.

Improvements to Two Multicast Switch Architectures

Jonathan Turner

We have developed two significant improvements to existing multicast switch architectures. The first involves systems that use a Beneš topology with fixed path routing and the second involves Lee's multicast extension to the Sunshine switch architecture. Both are the subject of pending patent applications.

Several switching system manufacturers have designed switching systems for ATM networks that employ a three stage Beneš network with fixed path routing. In order for such a system to be non-blocking, the internal links must be operated at a higher rate than the external links. The ratio of a switch's internal data rate to its external data rate is called the system's speed advantage. Minimizing the required speed advantage is important because the speed advantage contributes directly to the system's cost. When the three stage Beneš network is used to carry point-to-point traffic, a speed advantage of 3 is sufficient to ensure that blocking never occurs. However, a speed advantage of $\sqrt{n}$ is required to make an n port network nonblocking for multipoint traffic. We have found a way to construct and operate a five stage network so that it is effectively nonblocking and requires a speed advantage of only 3. This reduces the cost of a nonblocking multipoint switch by a factor of $.2\sqrt{n}$; hence the cost reduction is a factor of 3.2 for a 256 port switch and 6.4 for a 1024 port switch.

The system guarantees that blocking will not occur when setting up a new multipoint connection. However, the possibility of blocking does occur when adding a branch to an existing connection. This blocking can be eliminated by rerouting all or part of the connection. While this means that the network is only rearrangeably nonblocking, note that the only connection to be rearranged is the one being augmented. Moreover, the cost of performing the rearrangement is quite modest, requiring changes to the multicast translation tables in only ten of the multicast switches for a 256 port network constructed from 16 port switches. Further details can be found in [82].

The Sunshine switch, under development at Bellcore, is one of a class of packet switching systems uses sorting networks as a key component. The Sunshine architecture can be extended to support multicast as described in [46]. Lee's multicast extension adds several components to the front of a point-to-point switching system; The first is a concentrator which places the packets on consecutive outputs so as to ensure non-blocking operation of the subsequent networks. Next, the packets pass through a running adder network, which for the packet on output port i computes the sum of the fanouts of all packets entering on ports 0 through $i-1$ and places this sum in a field of the packet. Following the adder, the packets enter a set of *dummy address encoders* (DAC), which perform two functions. First, the DAC at output i sets lo to be the sum computed by the adder network for output i and lets $hi = lo + f_i - 1$ where f_i is the fanout of the packet at output i. If hi exceeds the number of outputs of the entire network, the packet is discarded. The DACs return an acknowledgement to the sending input for each packet that is not discarded. The discarded packets are retransmitted later. For the packets that are not discarded, lo and hi are inserted into the packet header as the packet is sent to the copy network. The copy network sends copies of each packet to all the outputs in the range from lo to hi. The copy network also labels the copies and when each copy reaches a *Broadcast Translation Circuit* (BTC) its multicast address and the copy number are used to produce an output address that the point-to-point switch uses to guide the packet to its ultimate destination.

This architecture has two shortcomings. First, its worst-case performance can be very poor, second, the total memory requirements are

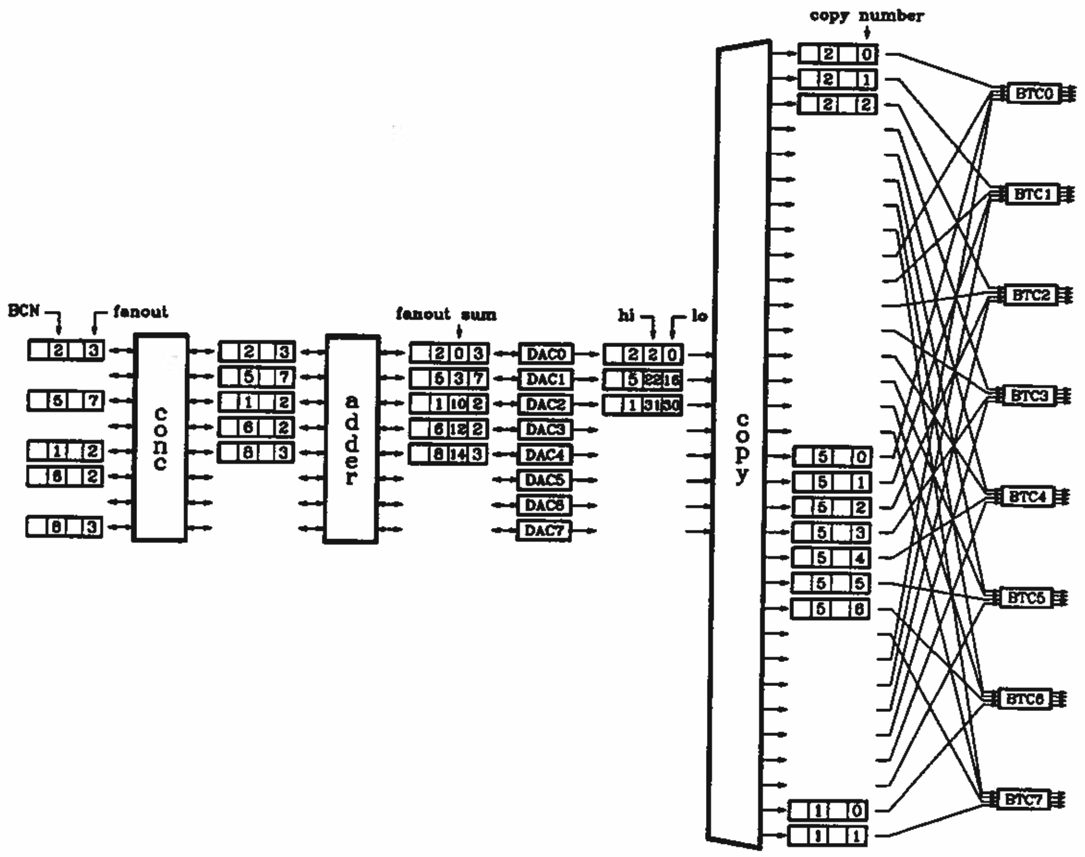


Figure 4: Example of Modified Multicast Architecture in Operation

extremely large. The worst-case improvement can be improved simply by expanding the copy network to have $N > n$ outputs where n is the number of outputs of the point-to-point network. This ensures that the copy network can produce at least $N - n$ packets in each operational cycle. To reduce the memory requirements, three changes are needed. The first involves sharing each BTC among multiple copy network outputs. The second involves partioning the space of multicast addresses so that connections with small fanout can be allocated a smaller amount of memory. The third change alters the dummy address encoders so that copies sent to the BTCs are constrained in such a way to limit the set of distinct copies that can arrive at each BTC. Implementing this last change leads to some inefficiency in the copy network and leads to a further expansion of the number of copy network outputs. To ensure nonblocking operation, this number must be at least $6n$. Figure 4 illustrates the modified scheme; in the figure shown, the number of copy network outputs is $4n$ instead of $6n$ to simplify the example.

The net result of the modifications is to cut the memory requirements by as much as three orders of magnitude. So for example, in the original design, a 256 port system with 65,536 distinct multicast addresses would have required 250 Mbits per port, assuming BTC entries are 16 bits each. The revised design requires about 250 Kbits per port. The overall cost of adding multicast to such a switch (assuming 155 Mb/s links) is less than one chip per port. A more detailed description of the multicast extension can be found in [83].

Control and Operation of Multicast Communication Networks

Connection Management in Multicast Networks

Rick Bubenik, John DeHart, Mike Gaddis, Victor Griswold, Kurt Haserodt, Mark Hunter, Ken Katsumata, Jonathan Turner

Connection management refers to the collection of algorithms, data structures and protocols used to create and maintain connections among users. In conventional networks, connections join two endpoints. In multipoint networks, connections may join an arbitrary number of endpoints. Several types of connections appear to be useful, including point-to-point connections and simple broadcast connections having one transmitter and many receivers. Connections in which all participants can both transmit and receive are also useful for conferencing and LAN interconnect applications among others.

As one considers applications of multipoint communication, one soon realizes that what is needed is a general multipoint connection capability that realizes point-to-point, broadcast and conference connections as special cases. We illustrate this method by describing a simple one-way broadcast connection, shown in Figure 5. The connection has a single transmitter I and receivers A, C, F and L. The internal nodes in the diagram represent switching systems. At various internal nodes, the stream of packets originating at I is replicated and forwarded to the appropriate destinations. The connection induces a tree in the network, in much the same way that a point-to-point connection induces a path in a conventional network. The table in the upper right-hand corner of the diagram summarizes some global information describing this connection. The $I7$ at the top is the *connection identifier*, which is a globally unique name identifying this connection and distinguishing it from all other connections in the network; the motivation for having a connection identifier will be explained below. One simple way of providing such an identifier is to use the address of the *owner* of the connection together with an integer distinguishing this connection from others that the owner may also be participating in. The owner of the connection is just that termination that is responsible for the connection and controls access to it. This scheme has been used in the example, implying that I is the owner, as well as the only transmitter in the connection. The 50M denotes a *rate specification* of 50 megabits per second. In a real network, a more complex rate specification is required, allowing specification of peak rate, average rate and some measure of "burstiness," but we will ignore this issue here. Each endpoint participating in the connection can have *transmit-only* permission, *receive-only* permission or *transmit/receive* permission. The permission concept provides the basic mechanism needed to allow the specification of general multipoint connections, that can be tailored to different applications. The network uses the permission information to allocate resources, (primarily link bandwidth) appropriately. The R at the bottom of the table defines the *default permission* to be receive-only, meaning that whenever an endpoint is added to the connection it is initially assigned receive-only permission; this can of course be changed by the owner if some other permission is required.

The example illustrates a connection that might be appropriate for distributing an entertainment video signal. To establish such a connection, G would send a control message to the network, describing the type of connection required. At that point it could begin transmitting on the connection, but initially there would be no one to receive the signal. Endpoints can be added to the connection in one of two ways. First G, as the owner, can send a message to the network asking that a particular endpoint be added. In response, the network would send a *connection invitation* to the specified endpoint and if the endpoint agrees (by exchange of control messages) to join the connection, the network would allocate the

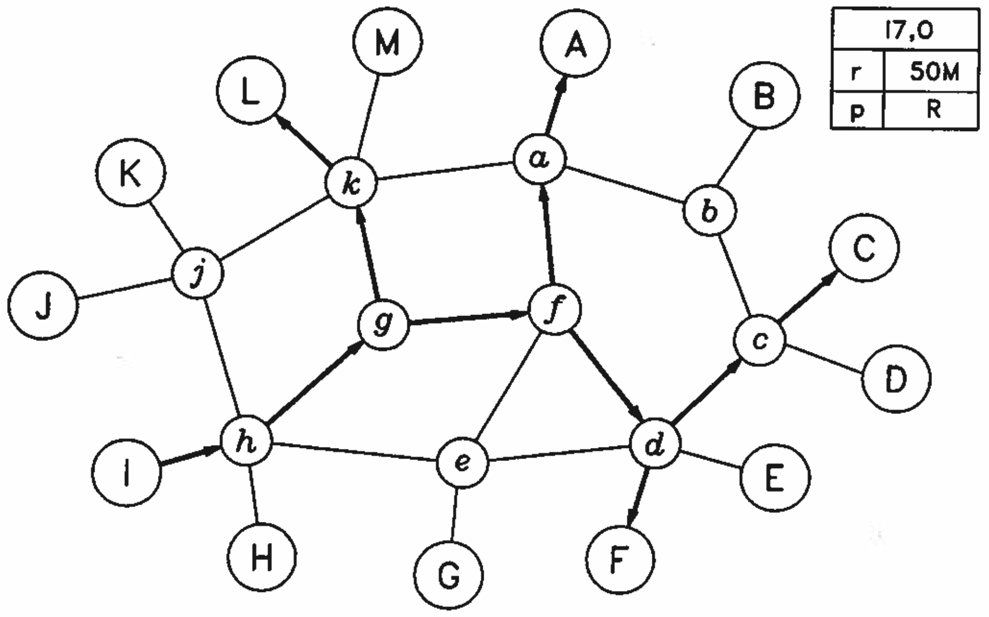


Figure 5: One-to-Many Connection

necessary resources to include the new endpoint in the connection. For entertainment video signals, a more appropriate way of adding an endpoint is at the endpoint's request. That is, an endpoint could send a control message to the network, requesting that it be added to a specified connection. To make such a request, the endpoint must specify the appropriate connection identifier. For entertainment video signals, this information would typically be widely available and could be built into terminal equipment, or programmed in, as appropriate. In response to such a request, the network would first search for the nearest place that the specified connection is available and then attempt to add the new endpoint by creating a branch at that point.

Addition of new endpoints from "outside" the connection raises the need for some form of authorization. In the example, the *O* at the top of the table specifies that this connection is *open*, meaning that anyone who wishes to join the connection may do so without explicit authorization from the owner. This would probably be the appropriate specification for a commercial video broadcast. Other options include *closed*, meaning that no one can join from outside and *verify*, meaning that outsiders may join, but only after getting explicit permission from the owner.

The highly dynamic nature of large multipoint connections makes it necessary to use control protocols that allow concurrent changes in separate portions of a large connection. However, changes that affect data at common points in the network must be sequenced carefully to ensure that the distributed database describing the connection configuration does not become inconsistent. We have developed a simple transaction-based protocol that supports highly dynamic multipoint connections using a three phase request-acknowledge-commit protocol.

See references [39, 70, 28] for further details.

Routing of Multipoint Connections

Makoto Imase, Bernard Waxman

In a packet switched network which uses virtual circuits, the primary goal in routing connections is to make efficient use of the network resources. For example we favor an algorithm which can handle the largest number of connections for a given set of network resources. In a point-to-point network, routing is often treated as a shortest path problem in a graph. Here the network is modeled as a graph $G = (V, E)$ where the nodes of a graph represent switches and the edges represent links. In addition we have two functions cap:$E \rightarrow \Re^+$ and cost:$E \rightarrow \Re^+$ which give us the bandwidth and cost of each edge (link). In this model we equate cost and edge length. At the time a connection is established, a shortest path with sufficient available bandwidth connecting the pair of endpoints is selected.

Routing of multipoint connections may be modeled in a similar way. In the multipoint problem we wish to connect a set $D \subset V$. Instead of the shortest path, one is interested in the shortest subtree which contains the set D. Finding the shortest subtree connecting a set of points is a classical problem in graph theory known as the Steiner tree problem in graphs. This problem has been shown to be NP-complete by Karp. Consequently one is forced to consider approximation algorithms which are not guaranteed to produce optimal solutions.

There are several polynomial time approximations algorithms for solving the Steiner tree problem, which we have used as a starting point for work on multipoint routing. The *minimum spanning tree heuristic* (MST) produces solutions whose costs are never worse than twice that of an optimal solution. Our experimental evaluations of MST indicate that it typically yields solutions that are within five percent of optimal. Figure 6 illustrates an example of the application of MST. Here we are asked to connect the set of four nodes $D = \{a, d, e, g\}$. The first step of the algorithm involves constructing a derived graph $G[D]$. This graph is a complete graph on the four nodes in D, where the length of each edge corresponds to the length of the shortest path in the original graph G. The second step involves finding a minimum spanning tree for $G[D]$. This can be done using one of several polynomial time algorithms. Finally the edges of the minimum spanning tree for $G[D]$ are mapped back to paths in the original graph, taking advantage of path overlap. Note that the solution here has cost two units more than optimal.

We have studied a more sophisticated algorithm for the Steiner tree problem known as Rayward-Smith's algorithm and have shown that it also produces solution that are no worse than twice optimal, and surprisingly, that it can produce solutions that are that bad. We have devised a generalization of Rayward-Smith's algorithm which we conjecture produces solutions that can be arbitrarily close to optimal, at the cost of increased, but still polynomial running time. We have also developed iterative versions of MST and Rayward-Smith's algorithm which while they have the same worst-case performance as the ordinary versions give slightly better performance in practice.

We have studied a dynamic version of the Steiner tree problem that more closely models the behavior of communication systems in which endpoints are added and removed from a connection over time. In particular, we have shown that if no rearrangements are allowed, the best possible solutions can be about $(1/2)\log_2 n$ times the cost of solutions obtainable with unlimited rearrangement. Furthermore, a simple greedy algorithm produces solutions that are never more than about $\log_2 n$ times that obtainable with unlimited rearrangement. When limited rearrangements are allowed, solutions with cost no more than eight times that of an optimal

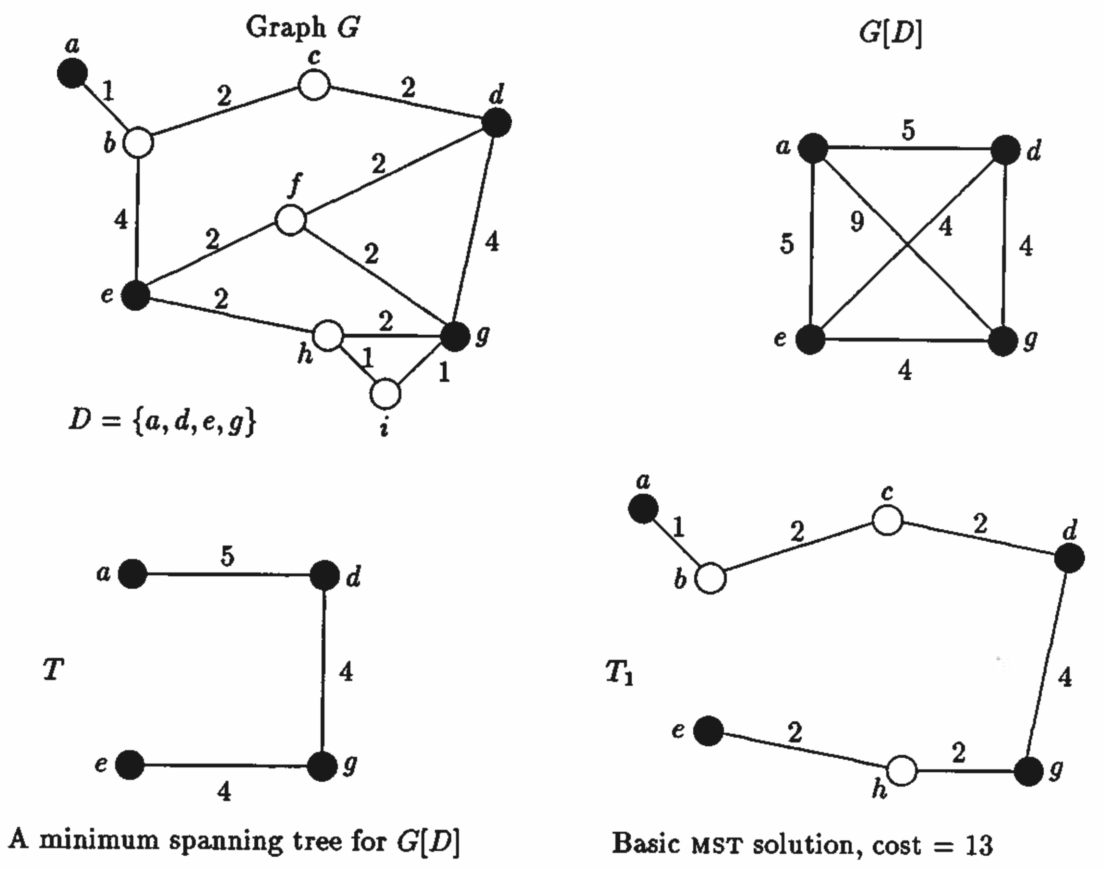


Figure 6: An Example of the Application of MST

solution are possible, but the number of rearrangements is large, (approximately the square root of the number of endpoints).

Probabilistic analysis has also been applied to compare the performance of the various algorithms for the Steiner tree problem. These results show for example that both MST and Rayward-Smith's algorithms produce optimal solutions with high probability in large networks with large multicast connections. If the size of the connection is small, Rayward-Smith's algorithm still yields optimal solutions with high probability but MST generally yields suboptimal solutions.

See references [40, 41, 92, 93, 94, 95, 96] for further details.

End-to-End Communication in High Speed Networks

High Speed Internetworking

Guru Parulkar, Chuck Cranor, Sanjay Kapoor, Tony Mazraani

We have proposed a very high speed internet abstraction (called VHSI) which can efficiently support guaranteed levels of performance for a variety of applications, and can cope with the diversity of underlying networks [48, 55, 56]. Important components of this abstraction are shown in Figure 7. We continue to make good progress on the research and development of various components of the VHSI abstraction. Each component of the VHSI abstraction is summarized in the following paragraphs:

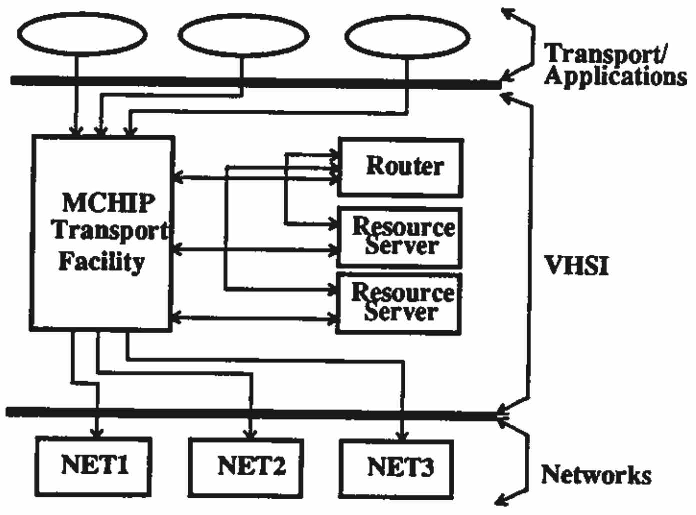


Figure 7: VHSI Abstraction

MCHIP. MCHIP is a novel multipoint congram-oriented high performance internet protocol, equal in status to IP in terms of the protocol hierarchy. The congram service primitive aims at combining the strengths of both classical connection and datagram approaches.

MCHIP includes two types of congrams: user congram (UCon) and persistent internet congram (PICon). UCons provide support for the connection-oriented applications, and PICons for datagram applications. Specifications of UCon and PICon management have been completed [1, 48]. We are developing an experimental MCHIP implementation in the kernel of SunOS Unix 4.0. We have created a new socket family for MCHIP, which allows application to use the standard socket interface for communication.

Resource Server. The VHSI abstraction provides performance guarantees to applications by preallocating resources to congrams, based on the application needs. However, a number of networks do not do resource management on a per congram basis, and therefore the VHSI abstraction includes resource servers to provide this functionality. We have completed a simulation study of a resource server for Ethernet with a variety of traffic sources, including Poisson and bursty. The study is summarized in the next two sections, and details are available as part of a MS thesis [49].

We have also made considerable progress on the simulation study of PICon multiplexing. Note that the PICons are provided to multiplex datagram traffic from a number of sources, and this simulation study is aimed at understanding how many and what kind of datagram sources can be successfully multiplexed on a given PICon.

Gateway Architecture. The VHSI abstraction requires that the gateway architectures be able to support data rates of at least a few hundred Mbps, to interface with diverse networks, and to implement MCHIP without becoming a performance bottleneck. The per-packet processing in a VHSI gateway is simplified and is performed all in hardware to achieve high throughput in packets/second. A paper design of a simple two port ATM-FDDI gateway has been completed and is summarized in a subsequent section.

Performance Analysis of the Ethernet under Bursty Traffic Conditions

Tony Mazraani, Guru Parulkar

In this section we summarize a study on performance analysis of Ethernet under conditions of bursty traffic. This study is motivated by the following two step reasoning.

First, we believe that at least in the first phase of deployment of broadband networks, they will be used as backbone networks at the campus, regional, and national level, and the access networks will continue to be local area networks (LANs) such as Ethernet, token ring, and FDDI.

Second, broadband network deployment will also bring with it a whole set of new applications, such as scientific imaging and visualization, multi-media conferencing, video distribution, and others. We claim that most of the broadband applications need to be modeled as bursty sources, and therefore, previous analyses of Ethernet with Poisson sources cannot be used to accurately characterize the performance obtained by applications running on the Ethernet in a broadband communication environment. In this section we report results of our simulation study aimed at characterizing Ethernet performance for broadband applications.

The simulation model simulates the operation of the standard Ethernet at the level of the data link layer. Some features of the physical layer, such as cable propagation and repeater delay are also included. The network consists of a single channel and a set of stations. Each station has an independent packet generator and a buffer of infinite capacity as shown in Figure 8. Hence, packet loss in this model occurs only when the Ethernet protocol discards a packets after 16 unsuccessful retransmission attempts[1]. In each station, the first packet in the queue is completely transmitted before a second packet, if it exists, is dequeued. A *balanced star* topology is assumed, *i.e.* the propagation delay between any two stations on the channel is constant. The simulator uses discrete-event simulation techniques, and is written in C and runs under the Unix operating system.

[1]However, we do monitor the packet queue of each station, and have found that the average and maximum queue lengths remain reasonable.

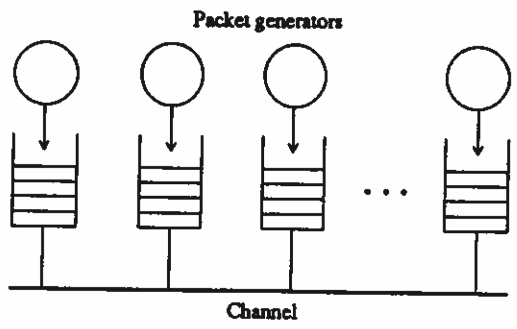


Figure 8: Network configuration

In this study, a bursty source is modeled as a *Markov chain* consisting of two states, *active* and *idle*, as shown in Figure 9(a). When the source is in the active state, it generates packets at a rate λ_p. In the idle state the source is shut off. The holding times in both the active and the idle states are exponentially distributed with means $1/\beta$ and $1/\alpha$, respectively. These holding times are computed using the peak bw, average bandwidth, and burst factor.

Note that for Poisson traffic, the mean holding times in the active and idle states of the packet generator are not specified. The packet generator is configured to remain in the active state for the duration of the experiment (*i.e.* $1/\alpha = 0$ and $1/\beta = \infty$). The only remaining packet generation parameter is λ_p, the mean packet arrival rate per station.

Figures 10 and 11 present mean packet delay and packet loss rate vs. throughput for different values of burst factor. The results clearly show that Ethernet performance under conditions of bursty traffic is different from the performance obtained when using traditional analyses with

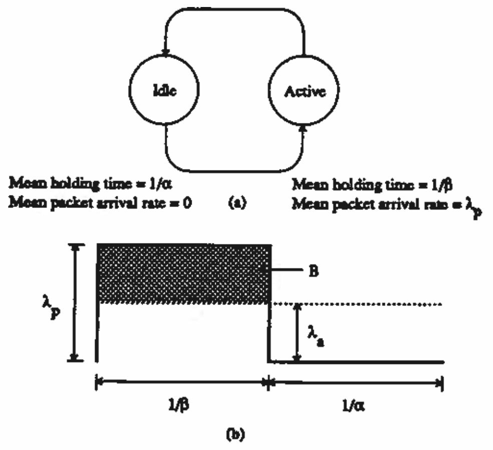

Idle
Active
Mean holding time = $1/\alpha$
Mean packet arrival rate = 0
(a)
Mean holding time = $1/\beta$
Mean packet arrival rate = λ_p
λ_p
B
λ_a
$1/\beta$
$1/\alpha$
(b)

Figure 9: Model of a bursty source

Poisson traffic. The simulation results show that the performance of Ethernet deteriorates as the burst factor increases. For instance, assuming a peak-to-average ratio of 4 and 1500 byte packets, Figure 10 shows that in order to guarantee less than 10 msec mean packet delay, the Ethernet utilization has to be much lower than the channel capacity, which is roughly 90%. Under conditions of Poisson traffic, the Ethernet utilization should be lower than 72%. However, under conditions of bursty traffic, this value deteriorates quickly to 57% and 43% for burst factors of 10 and 100, respectively. Bursty traffic has a more dramatic effect on packet loss. Figure 11 shows that in order to guarantee less than 10^{-4} packet loss under conditions of Poisson traffic, the Ethernet utilization should always be less than 62%. Under conditions of bursty traffic, this value decreases quickly to 39% and 27% for burst factors of 10 and 100, respectively.

The study also includes results that show how Ethernet performance changes with peak-to-average ratio and source cycle time as parameters of bursty sources [49].

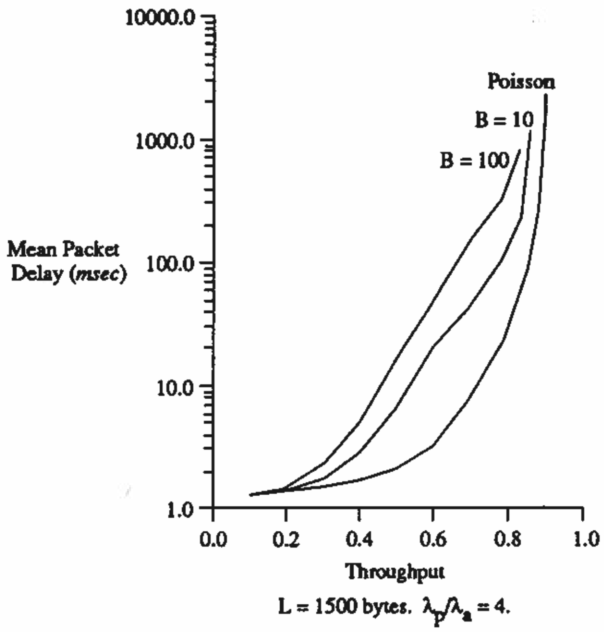


Figure 10: Mean packet delay vs. throughput and burst factor

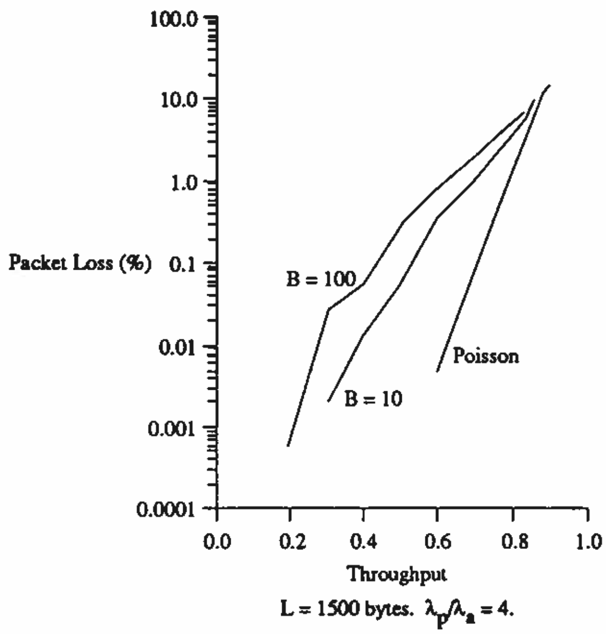


Figure 11: Packet loss vs. throughput and burst factor

Resource Management in Local Area Networks

Tony Mazraani, Guru Parulkar

The internet layer must support emerging broadband applications which include video distribution, distributed scientific computation and visualization, computer imaging, and multimedia conferencing. Supporting such applications means that the internet layer has to provide variable grade service with performance guarantees. If the underlying network is capable of managing its own resources and guaranteeing performance to applications, the internet layer can simply take advantage of this network facility. However, in the case of networks with no resource management capabilities, it is the responsibility of the internet layer to do resource management on behalf of the network. In order to provide performance guarantees at the internet level, the VHSI uses *resource managers* which manage the resources of those networks with no resource management capabilities.

In this section, we summarize a resource management model for the Ethernet in a connection-oriented communication environment, under conditions of bursty traffic. This model uses a central resource manager in a directly connected gateway. The role of the resource manager is to keep track of all active connections and their resource usage, and accept or block new connections depending on resource availability. The resource manager is always consulted before a connection can be established.

The goal of the resource manager is to guarantee performance to established connections, and to manage the network resources efficiently. Efficient resource management in this environment means maximum utilization of the network and minimum connection blocking, packet loss, and packet delay. Note, however, that minimizing packet delay may result in lower channel utilization. This study however shows that a simple resource management model can be devised to provide bounded packet loss and delay to various applications with reasonable channel utilization and blocking.

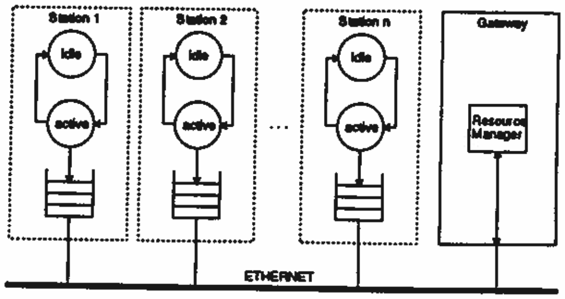


Figure 12: Network Configuration

The network for this study is shown in Figure 12. It consists of a set of stations and one gateway, all connected via an Ethernet. Communication in this network is connection-oriented, and connection management uses the new internet protocol [48]. This protocol requires that a station ready to start a connection send a connection request to the resource manager which resides in the directly connected gateway[2]. The resource manager gets the connection attributes from the request, and then accepts or blocks the connection depending on resource availability on the network. When the connection is terminated, the resource manager deallocates reserved resources for the connection. Traffic from each station is assumed to be bursty. Stations use five attributes to characterize their resource requirements; peak bandwidth, average bandwidth, burst factor, delay, and packet loss. Stations abide by their bandwidth and burstiness attributes, and use the standard Ethernet protocol to access the channel.

[2]Note that the resource manager does not have to be in the gateway. Since Ethernet is a broadcast network, any station on the channel may host the resource manager.

The resource manager accepts a new connection as long as its attributes (as well as those of the the most constraining active connection) are satisfied. It makes its decision based on a resource database which contains simulation data obtained from the performance analysis of the Ethernet under conditions of bursty traffic. The data is arranged in a matrix format with six columns representing the following six parameters: peak-to-average ratio, packet length, burst factor, average bandwidth, delay, and packet loss. Since the resource database consists of a set of discrete points, the resource manager uses linear interpolation to estimate those points that are not covered by the database. The details of the resource database and decision making algorithm of the resource manager can be found in [49].

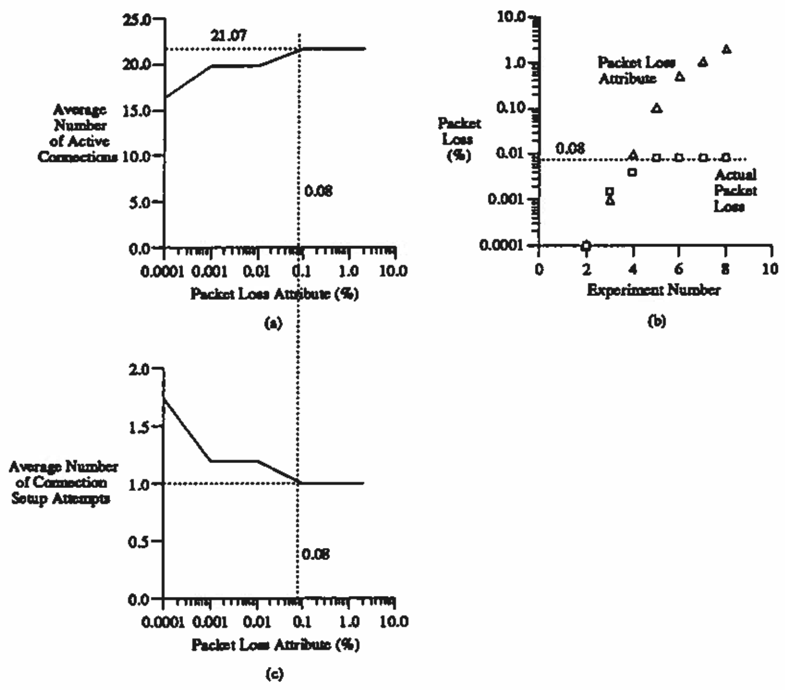
We decided to use a database of simulation data because there are no accurate analytical models that describe Ethernet performance in terms of the three parameters we use to characterize bursty sources (peak bandwidth, average bandwidth, and burst factor). The database approach has the advantage of being simple to implement and can also be easily expanded to incorporate new data about the performance of the network. This expansion is achieved by simply adding the new data to the resource database of the resource manager. The disadvantage of the database approach is its dependency on six variable parameters, especially packet length. This dependency means that a large amount of data is required to predict the performance of the network under all possible real-time conditions.

Figures 13 and 14 show simulation results obtained using connections with a peak-to-average ratio of 2 and average bandwidth of 0.2 Mbps. The average number of active connections as a function of the packet loss attribute is shown in Figure 13(a). The solid lines correspond to use of the resource manager, and the dotted lines correspond to results with no resource manager. These results show that when the resource manager is off, the average number of active connections on the channel is 21 and the mean packet loss rate is 0.08%. However, when the resource manager is on, packet loss rate less than 0.08% can be achieved at the expense of decreasing the average number of active connections. Note that as the packet loss attribute becomes larger than 0.08%, the average number of active connections starts to converge to 21. At this point, all connections are established on the first attempt. The actual packet delay on the channel ranges from 1.97 msec for experiment 1 to 3.17 for experiment 8.

Figure 13(b) shows the packet loss attribute requested by connections and the corresponding actual packet loss on the channel for each experiment (simulation run). These results show that by using the resource manager, the actual packet loss is always less than the requested loss attribute. When the packet loss attribute increases beyond the worst-case packet loss of 0.08%, all connections requests can be satisfied, and thus the actual packet loss converges to 0.08%.

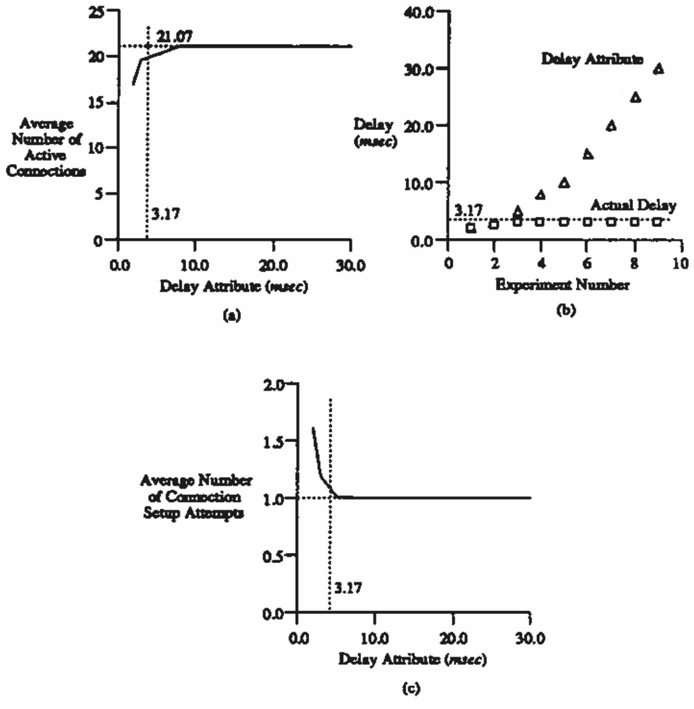
The average number of attempts made to set up a connection, which can also be considered as a measure of blocking, is shown in Figure 13(c) as a function of the packet loss attribute. When connections request very low loss rate, the resource manager blocks connections more often to guarantee the required performance. As a result, the average number of attempts to set up a connection increases, as shown in Figure 13(c) for loss attributes less than 0.08%. However, when the delay attribute becomes higher than the worst-case packet loss on the channel, the average number of connection setup attempts converges to 1, which means that connections are established on the first attempt.

More details of our performance study can be found in [49].



L = 1500 bytes. B = 10 packets.
Peak-to-average Ratio = 2. Average Connection Bandwidth = 0.2 Mbps.

Figure 13: Resource manager performance – similar connection attributes, variable packet loss attribute



L = 1500 bytes. B = 10 packets.
Peak-to-average Ratio = 2. Average Connection Bandwidth = 0.2 Mbps.

Figure 14: Resource manager performance – similar connection attributes, variable delay attribute

Design of an ATM-FDDI Gateway

Sanjay Kapoor, Guru Parulkar

In this section we summarize the design of an ATM-FDDI gateway that implements the VHSI functionality and is able to sustain high throughput. We believe that ATM and FDDI are excellent target networks to explore the VHSI gateway architecture for two reasons: these networks represent two important component networks of the next generation internet, and thus the ATM-FDDI gateway has a lot of practical value; also these networks are sufficiently different to pose the necessary challenges to the gateway designer. For example, an ATM network is essentially connection-oriented using small fixed size cells with explicit resource management. An FDDI network on the other hand is a datagram network using relatively large variable size frames with no explicit resource management.

An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing, and should be implemented in hardware for speed and performance. The non-critical path consists of connection management, resource and route management, and is best implemented in software due to complexity, need to do fine tuning, and flexibility. It is not an efficient approach to mix these paths as is generally done in present day gateways.

Figure 15 shows a high level schematic of a two port gateway. The gateway architecture consists of the following main components: ATM Interface Chip (AIC), SAR Protocol Processor (SPP), MCHIP Protocol Processor (MPP), Node Processing Element (NPE), reassembly buffer, FDDI interface chipset (SUPERNET[3]). We now explain each of these building blocks within the gateway.

[3]SUPERNET is a registered trademark of Advanced Micro Devices Inc.

ATM Interface Chip (AIC)

This chip interfaces with the fiber optic link from the ATM network/switch. It synchronizes the incoming bit stream from the fiber optic link to an internal clock. It also performs an error check on the 5 byte ATM header. Similarly, it generates a CRC check for the ATM headers of outbound cells. Any cells with an error in the header are simply discarded by the AIC.

SAR Protocol Processor (SPP)

This chip interfaces to the AIC and MPP. The SPP receives ATM cells from the AIC, and reassembles them into frames, using the SAR protocol headers. Similarly, it fragments FDDI frames received from the MPP into ATM cells and appends the necessary SAR headers to them. Some of the SPP's functionalities include: reassembly buffer management, reassembly timer management, writing cells into the reassembly buffer and reading frames out of the reassembly buffer concurrently for up to 16 active congrams.

MCHIP Protocol Processor (MPP)

This chip interfaces with the SPP, NPE, and transmit buffer memory. The SPP passes the reassembled frames to the MPP, which then decodes the frame type and routes them appropriately. Control frames are routed to the NPE without any table lookup. There are two types of control frames: network level control frames (that is, frames for ATM signaling), and internet or (MCHIP) control frames. For data frames, MPP does a table lookup to determine the outgoing destination address on FDDI. Similarly it reads an FDDI frame from the receive buffer memory, and appends the ATM header to it. It then loads the frame into the SPP FIFO. The MPP can receive control frames and initialization data from the NPE, meant for both the SPP and the MPP. The MPP forwards SPP initialization and control frames to the SPP without modification.

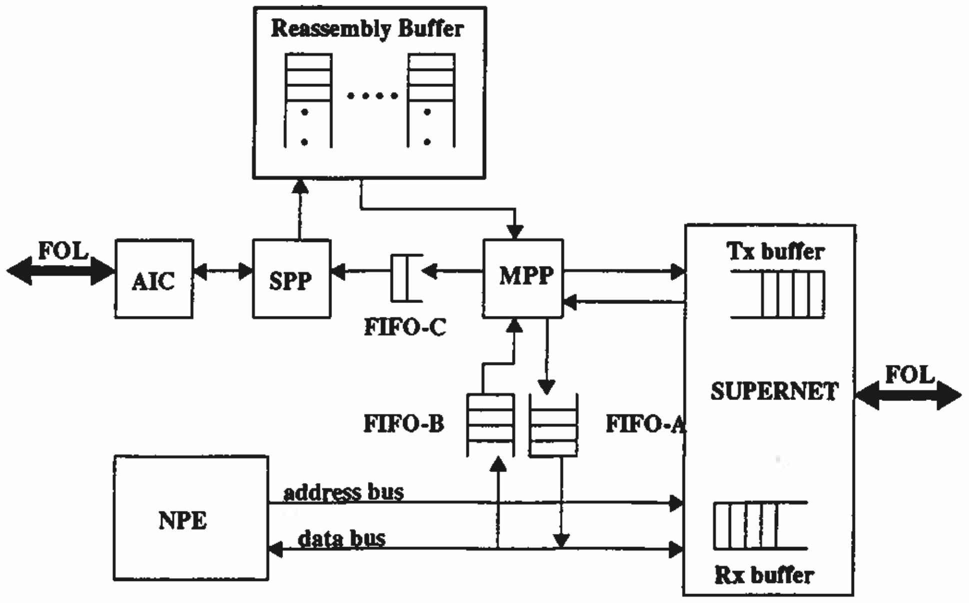


Figure 15: ATM-FDDI Gateway Architecture

SUPERNET

After loading frames in the transmit buffer, the MPP sends a transmission request to the NPE. This request is forwarded to SUPERNET by the NPE. Once the SUPERNET captures a valid token, it transmits the frames on the ring. When frames are received by the gateway, the NPE informs the MPP that there is a data frame in the receive buffer. The SUPERNET chip set implements the PHY layer and MAC layer protocols in VLSI. Station management and connection management are not part of the SUPERNET implementation, though it provides some registers to keep track of ring statistics, that can be used by the node processor to do station management and monitoring.

Node Processing Element (NPE)

The main functions of the node processing element are: running the ATM signaling, MCHIP connection management, and FDDI station management protocols and maintaining the resource server and the routing tables. Besides this, it also performs the housekeeping and initialization functions for the various chips of the gateway.

Buffer Memories

It should be noted that there are three distinct buffer memories in the gateway, and they are: the reassembly buffer, transmit buffer, and receive buffer memory. We decided on separate reassembly and transmit buffer memories in order to keep the SUPERNET interface clean and simple. The exact size of the buffers is a matter of further study, and can only be determined after a thorough simulation of the expected traffic intensity, and the type of traffic possible.

There are 3 FIFOs used in the gateway, labeled A, B, and C in Figure 15. FIFO-A buffers control frames received from the ATM network to be processed by the NPE. FIFO-B buffers ATM control frames and the initialization data for the MPP and the SPP originating from the NPE. The MPP decodes the type of packet and then routes SPP control and initialization frames to the SPP. The FIFO-C provides buffering for the FDDI fame being fragmented by the SPP and being sent to the ATM network.

Detailed designs of the AIC, SPP, AND MPP have been completed and can be found in [43]. The designs are in sufficient detail to allow a prototype implementation.

Interprocess communication for Remote Visualization

Larry Gong, Guru Parulkar

During the past year, we have initiated a research project to develop an efficient interprocess communication (IPC) facility at the transport level that is targeted for distributed pipelines. A distributed pipeline is a pipeline whose stage processors are physically distributed across (inter)networks. We believe that such a pipeline can be effective in supporting a class of remote visualization applications. Therefore we use pipelined televisualization (PTV) as the main application model in our IPC research. A typical configuration of a distributed pipeline for televisualization is shown in Figure 16. The first stage of the pipeline is usually a computer accessing a mass storage device or a supercomputer generating a large amount of data from simulation and/or computation. There are intermediate stages that perform various data transformations. The last stage performs graphics rendering and display, and accepts user input to steer the simulation. This project has been motivated by the following observations:

- Very little is understood about the real communication requirements for distributed computations on wide area networks, despite the fact that such understanding is necessary for design of efficient protocols to support these computations.

- Although pipelining is a simple special case of general distributed computation, it is readily applicable to many televisualization applications; and televisualization is a very important and challenging class of network applications because it is intensive both in computation and communication.

- Significant progress is being made in the development of high-speed packet switching networks, internetworks, and host-network interfaces. This development sets the stage for the development of IPC facilities that are necessary for converting the high bandwidth of networks into high performance for distributed applications.

To provide efficient interstage communication for a distributed pipeline, we have proposed an IPC solution which consists of three important parts: a segment streaming IPC paradigm, a two-level flow control method, and an application-oriented error control method. We summarize each of the components in the following paragraphs. A more detailed description is available in [36].

The new IPC paradigm, called *segment streaming*, is specifically proposed for supporting exchange of a stream of segments among processes with high bandwidth and low latency. Segment streaming is provided through two transport level operations: **send-stream** and **get-stream**. They are invoked by applications by making the corresponding system calls. For a distributed pipeline, the set of data items to be processed is defined as a segment group with each data item being a segment. A network connection is established for each segment stream upon the single request (get-stream or send-stream) by the application process. Each segment will be transmitted when ready across the connection without the latency of request or setup. Therefore, it supports segment prefetching and allows overlapping between the visualization computation and the communication processing.

With our distributed asynchronous pipeline, flow control becomes an important issue. Since each pipeline stage is usually a multiprogrammed system, change of multiprogrammed tasks will cause a corresponding change in the effective speed of the pipeline process. Moreover, underlying network connections can only provide statistical guarantees for bandwidths but not absolute rates. Without an effective flow control, buffer

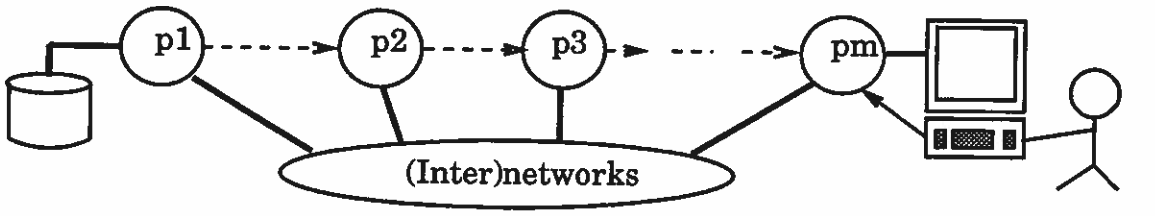


Figure 16: A Typical Televisualization Pipeline Configuration

overflow may occur at bottleneck stages. Any overflow can in turn cause significant performance degradation due to the linear data dependency among the pipeline stages. Our objective for flow control is to maintain the flow rate of the pipeline at that of the bottleneck stage and avoid any overflow. To achieve this objective, we use a two-level flow control method. The two-level method consists of a simple window flow control on top of the rate control mechanism. The simple window control only serves the interstage (end-to-end) flow control purpose and it is not overloaded with error control and congestion control as in TCP and other transport protocols. The rate control mechanism enforces data transmission and reception rates which are derived from the flow requirement of the simple window control and agreed upon by the underlying network at the time of connection setup.

In network environments with high bandwidth-delay product, time delay for error control (e.g. detection and error retransmission) becomes a more significant overhead to many applications. Besides, different application may have different tolerances for errors such as packet corruption and packet loss. Therefore we included an application-oriented error control mechanism in the proposed IPC solution. We have an initial design for the error control mechanism which is based on a moving average scheme. To be more specific, let E_i be an estimation of packet error rate for transmissions up to packet i and X_i be an error indicator for packet i (i.e. $X_i = 0$ if no error and $X_i = 1$ otherwise). Let μ ($0 \leq \mu < 1$) be a decay factor, then E_i is updated according to the following equation:

$$E_i = \mu E_{i-1} + (1-\mu)X_i \tag{1}$$

By specifying values for μ and an error rate threshold, application can express its tolerance of errors with different burst lengths. The control mechanism will request retransmissions for only those buffers that exceed the threshold. We are also developing a recurrence model to characterize the performance of a distributed pipeline. We plan to use this model to further understand the impact of variations in processor speed, error retransmissions, and different flow control strategies on IPC efficiency and performance of distributed pipelines. The last phase of this project will include an experimental implementation of the IPC facility and implementations of two typical visualization applications using the PTV model.

See [36] for more details.

The Axon Host-Network Interface Architecture

James Sterbenz, Guru Parulkar

The Axon architecture [62] is aimed at providing high bandwidth with low latency to the applications within the constraints of the host architecture and operating system. The important components of the Axon architecture include Network Virtual Storage (NVS), an application-oriented lightweight transport protocol for object transfer (ALTP-OT), and a pipelined communications processor (CMP). We have completed a paper design of these components, and they are available as reports [61, 63, 64].

The Axon architecture interfaces the CMP (communications processor) to the back end of a special dual-ported CMM (communications memory module). The CMM has a conventional random access port which appears like any other memory bank to the processor–memory interconnect. The second port is a high speed serial access interface to the CMP. The goals for the design of the CMP include the ability to perform critical path functions in real time, with no packet buffering, and the ability to incorporate the necessary functions in VLSI.

The ALTP-OT critical path, consisting of the data path and per-packet processing, is implemented in the CMP. The CMP consists of datapath (CMP_d) and control (CMP_c) portions. The CMP datapath interfaces to the VHSI optical links and the sequential ports of the CMM, and performs such functions as encryption/decryption and format conversion (encode/decode). The CMP control functions are those directly related to the datapath such as header build/decode, checksum generate/compare, CMM address generation, rate specification timing, as well as the per-packet congram multiplexing and control.

The Axon interface in fact also includes a high performance microprocessor, the *CMP assist processor* (CAP), which performs functions that are not part of the critical path, but require higher performance than could provided by the host CPU. The CAP is responsible for building control packets and passing them to the CMP for transmission and checksumming. Similarly, control packets received by the CMP are passed to the CAP for full decoding and subsequent action, which may involve interaction with the host CPU. The CAP is involved in the timer management for error packet retransmission, and in the packet arrival to page presence mapping ($\pi \rightarrow p$), in particular setting page and segment presence indicators and subsequent host notification for fault recovery.

The host CPU is responsible for link/segment/page fault handling (NVS), and per-congram functions (ALTP-host) directly corresponding to application requests.

In the following paragraphs we summarize the design of the CMP (Figure 17).

Congram control

The multiplexing of congrams (which may be considered to be soft connections) is handled by the *congram control logic*:

MPX – MULTIPLEXING control logic performs the hardware context switching of the CMP in response to transmission requests and received congram ids.

CSR – CONGRAM STATE REGISTERS hold all of the state information for each active congram, to allow rapid control and pipeline configuration changes for multiplexed congrams using the CMP. There is a set of CSRs for both the transmitting and receiving side of the CMP.

Packet data paths

The *transmit data pipe* and *receive data pipe* are the main data paths of the CMP. The transmit data pipe modules are:

ECD – ENCODE performs any transformations required by internetwork data format standards

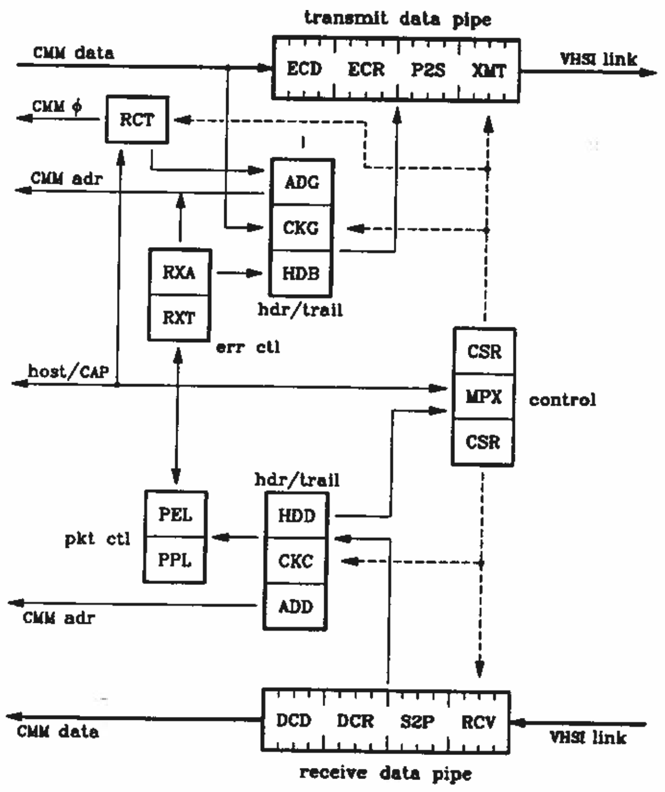


Figure 17: Communications Processor Block Diagram

ECD – ENCODE performs any transformations required by internetwork data format standards and accommodates heterogeneous hosts (such as byte ordering).

ECR – ENCRYPT performs data encryption of packet data and internal control fields.

P2S – PARALLEL-TO-SERIAL datapath conversion.

XMT – TRANSMIT performs line coding and transmission functions.

The receive data pipe modules are:

RCV – RECEIVE takes the bit stream derived from the optical receiver, handles line coding and clock recovery. for the receive data pipeline.

S2P – SERIAL-TO-PARALLEL datapath conversion.

DCR – DECRYPT performs data decryption of the packet data and internal control fields.

DCD – DECODE performs any required internetwork-to-host data format transformations.

Per packet control
Associated with the transmit data pipe are control modules:

RCT – RATE CONTROL uses the rate specification r_i for each congram i to determine the timing of data reads from the CMM, and thus the rate at which packets are clocked out of the transmit pipeline for each congram. The r_i values for each congram are obtained from the corresponding CSR.

ADG – ADDRESS GENERATE uses the initial CMM address of each page to form the addresses for each packet read from the CMM.

CKG – CHECKSUM GENERATE sums the packet data fields as they pass through the data pipeline. The computed checksum is then inserted in the packet trailer.

and network) are done at one time. The congram and request ids (c, q) and packet id (segment id k, page number j, packet number i) are inserted into a header template.

Associated with the receive data pipe are control modules:

HDD – HEADER DECODE determines the congram id for CMP configuration, and determines whether the packet type is control or data. Control packets are passed to the CAP. For data packets, HDD determines the packet address in the CMM from the packet index (ijk) and the base address of the page from the corresponding CSR. The congram and request ids (cq) are used by MPX to select the appropriate CSR.

CKC – CHECKSUM COMPARE sums the packet data fields as they pass through the data pipeline. The computed checksum is then compared with the actual checksum in the packet trailer. If a mismatch is found, the PPL and PEL (packet presence and error logic – see below) are notified to indicate that the packet has been discarded after initial receipt.

ADD – ADDRESS DECODE uses the initial CMM address of each page (from the CSR), and the packet index (ijk) to form the CMM address for the writing of each packet into the CMM.

The *packet control* logic is responsible for recording packet arrivals and missing or corrupted packets.

PPL – PACKET PRESENCE LOGIC keeps track of packet arrival to allow the CAP and host to determine the presence of complete pages and segments, so that the appropriate page descriptor table and segment descriptor table presence bits can be set, and host CPU application resumed.

PEL – PACKET ERROR LOGIC keeps track of corrupted (from CKC) and missing (from RXT) packets so that retransmission requests can be made, and also invalidates corrupted packets to the PPL.

The *error control* logic is responsible for generating the appropriate selective retransmission requests and packet addresses at the receiving end, and retransmitting packets on the sending end.

RXT – RETRANSMIT TIMERS determine (in conjunction with the CAP) when packet retransmission requests should be made, based on the retransmission policy. Timer values are accumulated to the proper granularity (packet, page, segment, or group). The fetch and preemption options are then used to determine when the **retransmit-packets** control packet should be sent, and the PEL is used to construct the retransmit packet bit map.

RXA – RETRANSMIT ADDRESS generates the (local) CMM addresses for packets to be retransmitted, using the base address of the corresponding page and the retransmit packet bit map in the **retransmit-packets** control packet received.

We are also completing a simulation of the Axon architecture to better understand fundamental issues and tradeoffs in the end-to-end data path as data rates scale above 1 Gbps, and determine the optimal partitioning of the communication function between hardware and software and between host processor and off-board processor.

Resource Management in Communication Networks

Performance Evaluation of a Traffic Management Mechanism in a UNI

Andreas D. Bovopoulos, Einir Valdimarsson

Integration has been the motivating idea behind the development of the broadband integrated services digital network (B-ISDN). Integration addresses issues related with the economical deployment of new services. Such services can be classified based on their requirements regarding bandwidth, buffering, signaling, and Operations, Administration, Maintenance and Service Provisioning (OAM&P). Our research has focused on the development of a switching and transport infrastructure that could support service integration in an efficient way. The ATM transport mechanism could be utilized in such an infrastructure if basic problems related to the bandwidth allocation were resolved.

It is our belief that the traffic control problem appearing in the ATM layer will defy solution as long as the problem is fragmented into a number of subproblems, each of which is studied in isolation. Our work tries to address collectively the problems of traffic characterization, services characterization, traffic control, and the appropriate hardware infrastructure. Our aim is to create a network traffic control architecture that serves as a framework for bandwidth allocation in ATM networks. The major characteristics of this architecture are its modularity, its ability to support classes of traffic, and its integration of the service specification, control and hardware design, and performance requirements. The cornerstone of this architecture is a traffic control module (TCM) that is able to shape, monitor, and control the incoming traffic.

The Traffic Control Module

The traffic control module (TCM) is the fundamental building block of an under-development bandwidth allocation and traffic control architecture. The incorporation of TCMs in various traffic handlers will facilitate the development of a reliable control infrastructure capable of supporting the ATM layer requirements.

The bandwidth allocation and traffic control that must be provided by a traffic handler for a given cell sequence (CS) is implemented via a TCM (Figure 18). The TCM has two buffers: the token buffer of size M and the cell buffer of size K. An incoming cell passes through the TCM without delay as long as the token buffer contains a token. One token is removed from the token buffer each time a cell passes through the TCM. If an incoming cell finds the token buffer empty, the cell is stored in the cell buffer, as long as the cell buffer is not full. Tokens may have a maximum lifetime; if this option is exercised, a token not used during a particular time interval is eliminated. Such a policy could be important for network management. Notice that a token represents the bandwidth required for the service of a single cell.

In an ATM network, predictable control policies are favored because of the desirability of minimizing the need for feedback control. For this reason, in the TCM, tokens are generated *with a deterministic and periodic pattern.* Specifically tokens are generated in accordance with a T-state, cyclic, deterministic Markov chain. The number of tokens generated in state l of the Markov chain is equal to s_l for $0 \leq l \leq T-1$. The time between two transitions of the Markov chain is D units of time, and therefore the period of the deterministic and periodic pattern is $D \times T$ units of time. The token patterns that can be generated in this fashion are predictable and yet at the same time rich enough to implement a wide variety of control policies. With the time scales D and $T \times D$, the two buffers K and M, and the number of tokens generated at each transition of the Markov chain, s_l, for

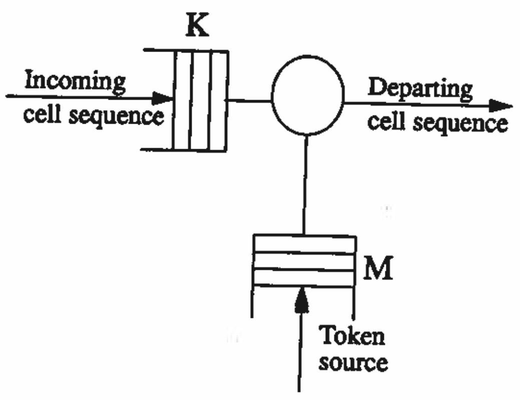


Figure 18: Traffic Control Module

$0 \le l \le T-1$, a wide variety of grade-of-service requirements can be provided by a traffic handler to an incoming CS.

The state of the token generating Markov source and the state of the token buffer can be easily implemented with two counters, one which records the state of the token buffer and the other which keeps track of the state of the token Markov chain. If keeping track of the lifetime of the tokens is necessary, the token buffer implementation is more complex.

The policing of cells could include two functions: counting the number of cells passing through the TCM and possibly blocking cells. Both of these functions could be implemented using the token counters and would not require any change of form of a cell or any additional memory requirements.

Traffic shaping could be effected by storing timing information for each incoming cell. The timing information would record the number of renewal epochs after which an incoming cell would be eligible for service. (A renewal epoch is the time at which new tokens are generated.) The timing information associated with a particular cell would be saved when that cell was stored in the cell buffer. The storing of timing information would require the provision of additional memory per cell in the cell buffer.

Note that traffic shaping at a UNI would affect the time at which cells of a particular CS would be allowed to enter the network. Furthermore, in a multiplexing environment, traffic shaping of one or more of the multiplexed CSs could result in the CSs being served in a manner other than FIFO. These capabilities are crucial for effective traffic management. Further details can be found in [23].

Performance Evaluation of the TCM

We have studied the performance of a UNI utilizing a TCM for traffic management in [22]; we summarize that work below. It is assumed that the traffic source generates cells according to a Poisson distribution. The inter-arrival time is exponentially distributed with rate c. Every D seconds the token source generates s tokens, where s is a fixed positive number.

Let q be the number of cells in the input buffer and w be the number of tokens in the token pool. Notice that the probability that both q and w are positive is zero. This is because outstanding cells always utilize any available tokens. The system state is described by the number of tokens that are required for the service of all outstanding cells in the input buffer plus the number of tokens that are needed to fill up the empty positions in the token buffer. Therefore when $(q, w) = (0, w)$, the state of the UNI is $M - w$, for $0 \le w \le M$. When the state of the two buffers is $(q, 0)$ for $0 \le q \le K$, the state of the UNI is $q + M$.

We observe the state of the UNI just prior to the generation of the s tokens. We call these instances in time the renewal epochs. At those instances the behavior of the UNI is described by an embedded Markov chain on the state space $\{0, 1, \cdots, K + M\}$. Let the state before the last renewal epoch be j. In general

$$q = (j - M)^+$$

and

$$w = (M - j)^+$$

Now let

$$w(j) \overset{\text{def}}{=} min\{(M - j + s)^+, M\}$$

and

$$q(j) \stackrel{\text{def}}{=} (q-s)^+ = (j-M-s)^+$$

Notice that if j is the state of the system before the last renewal, $w(j)$ and $q(j)$ are the total number of available tokens and cells respectively after the renewal. Furthermore notice that it is impossible for both $w(j)$ and $q(j)$ to be positive.

Let $A(u)$ be the total number of arrivals in the u seconds immediately following the last renewal, and let $\alpha_k(u)$ be the probability that $A(u) = k$, for $k \geq 0$. Then $\alpha_k(u) \stackrel{\text{def}}{=} P[A(u) = k] = e^{-cu}\frac{(cu)^k}{k!}$ for $k \geq 0$. For simplicity we write α_k instead of $\alpha_k(D)$. Let

$$b_{j_1,j_2} \stackrel{\text{def}}{=} \begin{cases} \alpha_{j_2-(j_1-s)^+}, & \text{if} j_2-(j_1-s)^+ \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

for $0 \leq j_2 < K+M$. Furthermore, let

$$\bar{b}_{j_1,j_2} \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if} j_2 < K+M \\ 1-\sum_{n=0}^{K+M-1} b_{j_1,n}, & \text{if} j_2 = K+M \end{cases}$$

for every value of j_1, $0 \leq j_1 \leq K+M$. The transition matrix of the embedded Markov chain is given by

$$\mathbb{P} \stackrel{\text{def}}{=} \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & \bar{b}_{0,M+K} \\ b_{1,0} & b_{1,1} & \cdots & \bar{b}_{1,M+K} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M+K,0} & b_{M+K,1} & \cdots & \bar{b}_{M+K,M+K} \end{pmatrix}$$

Let π_j be the equilibrium probability that the system is in state j immediately before the generation of tokens, and $\pi \stackrel{\text{def}}{=} [\pi_0, \pi_1, \cdots, \pi_{M+K}]$ be a vector such that $\pi\mathbb{P} = \pi$ and $\pi\mathbf{e} = 1$. π is the equilibrium probabilities vector of the state of the UNI at the instances immediately before the renewal epochs of the token source. $\mathbf{e}$ is a $M+K+1$ vector with all its elements equal to one. Let $E\gamma$ be the throughput, *i.e.* the expected number of cells that pass through the UNI system in the time period between two renewal epochs of the token source. Then

$$E\gamma = \sum_{j=0}^{K+M} \pi_j \Big\{ (K + w(j) - q(j)) -$$

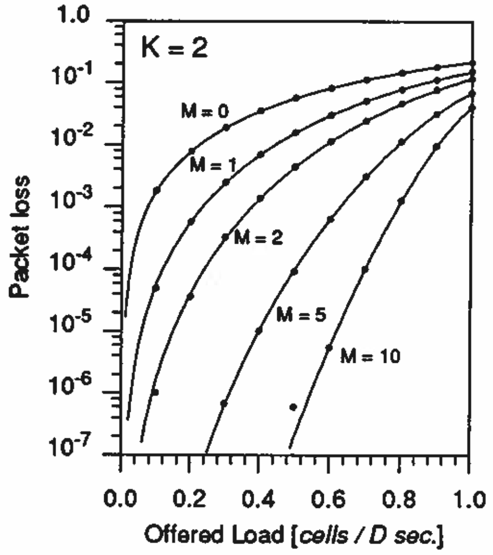


Figure 19: TCM Performance

$$\sum_{n=0}^{K+w(j)-q(j)-1} \alpha_n \times \Big(K + w(j) - q(j) - n\Big)\Big\}$$

The cell loss probability P_L is given by

$$P_L = 1 - \frac{E\gamma}{cD}$$

In Figure 19 we present the cell loss probability of the UNI for different values of the offered load, for different values of the buffer sizes, $s = 1$, and $D = 1.0$. The dotted points are derived via simulation. We observe that the theoretical results match the simulation results. In addition we observe that the throughput and the cell loss are a function of $K + M$.

In the future we intend to design traffic handlers that use the TCM. In addition we will initiate the development of a UNIX-based prototyping environment written in C++ on which evolving traffic control architectures can be tested, experimented with, and demonstrated.

Congestion Control Using Fast Buffer Reservation

Jonathan Turner

A central objective in ATM networks is to provide virtual circuits that offer consistent performance in the presence of stochastically varying loads on the network. This objective can be achieved in principle, by requiring that users specify traffic characteristics when a virtual circuit is established, so that the network can select a route that is compatible with the specified traffic and allocate resources as needed. While this does introduce the possibility that a particular virtual circuit will be blocked or delayed, it allows established virtual circuits to receive consistent performance as long as they remain active.

Ideally, a bandwidth management and congestion control mechanism should satisfy several competing objectives. First, it should provide consistent performance to those applications that require it, regardless of the other virtual circuits with which a given virtual circuit may be multiplexed. Second, it should allow high network throughputs even in the presence of bursty traffic streams. Third, the specification of traffic characteristics should be simple enough that users can develop an intuitive understanding of the specifications and flexible enough that inaccurate specifications don't have seriously negative effects on the user. Fourth, it should not artificially constrain the characteristics of user traffic streams; the need for flexibility in ATM networks makes it highly desirable that traffic streams be characterized parametrically, rather than by attempting to fit them into a pre-defined set of traffic classes. Fifth, it must admit a simple realization for reasons of economy and reliability. Less crucial, but in our view, also important, is the requirement that the bandwidth management mechanism accommodate multicast virtual circuits with multiple transmitters. All proposals we have seen for connection management in ATM networks have serious deficiencies with respect to at least one of these objectives. We describe here an approach using fast buffer reservation which appears to satisfy them all.

To preserve the integrity of user information bursts, the network must detect and track activity on different virtual circuits. This is accomplished by associating a state machine with two states with each virtual circuit passing through a given link buffer. The two states are *idle* and *active*. When a given virtual circuit is active, it is allocated a prespecified number of buffer slots in the link buffer and it is guaranteed access to those buffer slots until it becomes inactive, which is signaled by a transition to the idle state. Transitions between the active and idle states occur upon reception of user cells marked as either *start-of-burst* or *end-of-burst*. Other cell types include *middle-of-burst* and *loner*, the latter is used to designate a low priority cell that is to be passed if there are unused buffer slots available, but which can be discarded if necessary. A forced transition from active to idle is also made if no cell is received on the virtual circuit within a fixed timeout period.

Figure 20 illustrates the buffer reservation mechanism. For virtual circuit i, the mechanism stores the number of buffer slots needed when the virtual circuit is active (B_i), the number of buffer slots used by unmarked cells (b_i) and a state variable (s_i: idle, active). The mechanism also keeps track of the number of unallocated slots in the buffer (B). The detailed operation of the state machine for virtual circuit i is outlined below.

When a start cell is received:

- If the virtual circuit is in the idle state and $B - B_i < 0$, the cell is discarded.
- If the virtual circuit is in the idle state and $B - B_i \geq 0$, s_i is changed to active, a timer for that virtual circuit is set and B_i is subtracted from B. If $b_i < B_i$, b_i is incremented and the cell

is placed (unmarked) in the buffer. If $b_i = B_i$, the cell is marked and placed in the buffer.

If a start or middle cell is received while the virtual circuit is in the active state, it is queued and the timer is reset. The cell is marked, if upon reception, $b_i = B_i$, otherwise it is left unmarked and b_i is incremented.

If a middle or end cell is received while the virtual circuit is in the idle state, it is discarded.

If an end cell is received while the virtual circuit is active or if the timer expires, s_i is changed from active to idle and B_i is subtracted from B.

If a loner is received, it is marked and placed in the buffer.

Whenever a cell is sent from the buffer, the appropriate b_i is decremented (assuming the transmitted cell was unmarked).

When a virtual circuit is routed, the software that makes the routing decisions attempts to ensure that there is only a small probability that the instantaneous demand for buffer slots exceeds the buffer's capacity. This probability is called the *excess buffer demand probability* and might typically be limited to say 1%.

To make this precise, let λ_i denote the peak data rate of a given virtual circuit and let μ_i denote the average rate. If the link rate is R and the buffer has L buffer slots, the number of slots needed by an *active source* with peak rate λ_i is defined to be $B_i = \lceil L\lambda_i/R \rceil$.

Since B_i buffers are allocated to a virtual circuit when it is active, the virtual circuit's instantaneous buffer requirement is either 0 or B_i. If we let x_i be a random variable representing the number of buffer slots needed by virtual circuit i at a random instant then $\Pr(x_i = B_i) = \mu_i/\lambda_i$ and $\Pr(x_i = 0) = 1 - \mu_i/\lambda_i$.

Consider then, a link carrying n virtual circuits with instantaneous buffer demands $x_1, \ldots, x_n$.

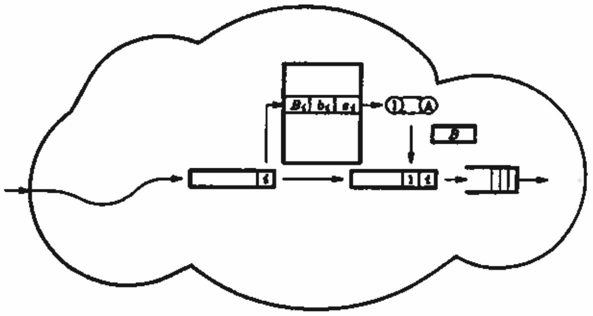


Figure 20: Fast Buffer Reservation

Define $X = \sum_{i=1}^{n} x_i$. Note that X represents the total buffer demand by all the virtual circuits. Suppose we have a new virtual circuit with buffer demand x_{n+1} and we want to decide if it can be safely added to the link. We first must compute, the probability distribution of the random variable $X' = X + x_{n+1}$. This can be obtained by numerical convolution of the distribution of X with the the distribution of x_{n+1}, assuming that the idle, active behavior of the new virtual circuit is independent of the the existing virtual circuits. To decide if the virtual circuit can be accepted we then simply verify that $\Pr\{X' > L\}$ is small. For a link with a 256 slot buffer, we estimate that about 2000 multiplications and 1250 additions are required to comptute the distribution of X' and $\Pr\{X' > L\}$. Using fixed point arithmetic, this can be done in less than half a millisecond on a 10 MIPS processor.

To complete the bandwidth management scheme, a traffic monitoring mechanism is required at the user-network interface to ensure that the virtual circuit's long term average data rate does not exceed the value specified at virtual circuit establishment. We have designed an appropriate mechanism of this sort, and generalized it to support multicast virtual circuits with multiple sources. We have studied the implementation complexity of this approach and estimate that the incremental cost of adding bandwidth management hardware to a port controller of an ATM switch to be no more than 10%.

See [85] for further details.

Resource Allocation for Markovian Networks

Andreas Bovopoulos

For a number of years we have investigated the interplay between routing and flow control in packet switched networks. We have studied packet switched networks that are monitored and controlled by a controller with complete, partial, or no information about the activities in the network. Our objective has been the derivation of the optimal routing and flow control in a single class network based on the following objective: the maximization of throughput under the constraint that the expected time delay of packets in the network not exceed a given upper bound.

A packet switched network is modeled as a Markovian queueing network. The previous problem is formulated as an optimization problem with respect to the routing and flow control parameters. A summary of the main contributions of our work follows.

First, we assume that the network controller has complete information about the activities inside the network. At any given moment it knows the total number of packets in each of the network's processors. Since the routing is considered to be state dependent, the network analyzed does not have a product form solution. The state of the system is characterized by the total number of packets in each of the network processors. Packets arrive at the controller with rate c, and the controller makes state-dependent routing and flow control decisions. The optimal state dependent routing and flow control parameters that maximize the average network throughput under an average time delay constraint are given by means of an iterative linear programming procedure. The optimal routing is shown to be essentially deterministic, and the optimal flow control mechanism of a generalized window type. This work is presented in [7, 8, 9, 10, ?].

Second, we alternatively assume that the network controller at any given moment knows only the total number of packets for which it has not yet received an acknowledgment. Consequently, the network controller has only partial information about the activities inside the network. The optimization problem analyzed in this section is a centralized optimization problem. Although the routing is considered to be state dependent, the network is approximated by one that has a product form solution, an approximation that makes its analysis tractable. The optimization of the routing inside the network is achieved by maximizing the value of its state dependent Norton equivalent. A resource allocation algorithm is derived to solve the resource allocation problem for this class of Markovian queueing networks. This work is presented in [13].

Third, the resource allocation problem for Jackson networks is investigated. The state of the network is represented by the total number of packets for which the source has not yet received an acknowledgment. We assume that the acknowledgment packets are subject to delay as they travel from destination to source. The routing is assumed to be state independent. The flow control is assumed to be state dependent, while the acknowledgment delays are assumed to be greater than zero. The objective is to maximize the average throughput of the network such that the end-to-end expected time delay of the packets in the forward network does not exceed an upper bound. The optimal flow control is shown to be a window flow control, and the routing policy derived balances the traffic inside the network. Based on the previous results, we study the effect on network performance of the amount of information available to the controller. Specifically, we study the effect of acknowledgment delay on network performance. We also compare network performance using state dependent (window) flow control or state independent (rate) flow control. We derive conditions under which each of these flow control policies is superior with respect to end-to-end network performance. This work is presented in [11, 12, 14].

Finally we studied the effect of the availability of either complete, partial, or no information about the network's state at the controller. In [20] it is shown that network performance improves as more information is made available to the controller. It is also shown that as long as the flow control and routing parameters are computed from the same information, they can be treated identically. More specifically, the flow control parameter can be treated as one of the routing parameters.

Adaptive and Fair Resource Allocation

Andreas Bovopoulos

Understanding the dynamics and control of multi-class networks has been one of our research focal points. We have investigated the problem of finding the decentralized flow control of a BCMP network. Each network user is assumed to operate with either a state-dependent arrival rate (i.e., an arrival rate which depends upon the number of the user's packets that have not yet been acknowledged) or a state-independent arrival rate (which the user chooses). We have developed two alternative formulations of the decentralized flow control problem. With the first approach we are mainly interested in achieving a better utilization of the network resources. As a result the network controller computes and enforces the network flow control policies. Two different optimization criteria are considered. Under the first optimization criterion, the decentralized flow control corresponding to each of the network users maximizes the throughput of the network, under the constraint that the expected time delay of the packets in the network not exceed a preassigned upper bound. Under the second optimization criterion, the decentralized flow control corresponding to each of the network users maximizes the throughput of the network, under the constraint that the expected time delay of each particular class of packets not exceed a preassigned (user dependent) upper bound. The results of the research have been published in [17].

With the second approach users are able to change their policies at will. Each user operates using a rate based flow control. The power based optimization criterion is employed for the derivation of the optimal flow control for each of the network's users. We have shown that the optimal arrival rates correspond to the unique Nash equilibrium point of a non-cooperative game problem. In addition we have derived asynchronous algorithms for the computation of the Nash equilibrium point of the network. Among them, the nonlinear Gauss-Seidel algorithm is distinguished for its robustness and speed of convergence. The results of the research are described in [15, 16, 18]. We are currently in the process of extending this work.

We plan to investigate a number of resource allocation problems appearing in packet networks that must provide *performance guarantees* to their users. We wish to study three inter-related problems: *flow control, congestion control, and routing.* We aim to derive decentralized algorithms that can be implemented in packet networks. Furthermore we intend to develop a set of analytical tools that can be used for the design of *performance-oriented networks.* With such networks, a user requests a certain quality of service (QOS) at call set up time. This QOS has implications for both the transport and network layers. Specifically the QOS affects flow control decisions at the transport layer and routing and congestion control decisions at the network layer.

In creating a performance-oriented network design, both the desires of the network user and service provider must be taken into account. A user wants a requested service; the service provider wishes to make the most efficient utilization of network resources and therefore to maximize revenues. Our goal is to create a performance-oriented network design that considers both desires. We plan to provide a game theoretical formulation of the resource allocation problems because game theory encapsulates the *conflicting requirements* that are imposed on network management. Our initial results are reported in [19] and are quite encouraging.

We would also like to bridge the dichotomy between user and network requirements. Specifically, the resource allocation decisions should be based on the users' requirements while at the same time, resulting in an effective and fair utilization of network resources. We expect to demonstrate the way in which congestion control, routing and flow control algorithms must be modified in order to provide fair service and performance guarantees to network users. Such an environment should be distributed, adaptive and able to operate with a minimum exchange of information. As an initial step in this direction, we are currently investigating the problem of adaptive flow control.

Queueing Behavior of Bursty Traffic Streams

Shahid Akhtar, Akira Arutaki, Jonathan Turner

One of the principal advantages of packet switching is its ability to support communication channels of any rate across a potentially wide range. Not only can different channels operate at different rates, but the rates of individual channels may vary over time. This latter property leads to the possibility of overload since there may be periods when the total offered traffic exceeds the network's capacity.

In conventional, low speed packet networks, such overload periods are controlled using a variety of feed-back oriented techniques, which attempt to detect overload and then apply control mechanisms that reduce the offered load. A common approach is to allow transmission of a packet from one switch to another only when the receiving switch is known to have a buffer available. During overload periods, such networks become congested with traffic backing up toward the sources, which are ultimately forced to reduce their rate of transmission until the congestion clears. This technique works well in networks with low speed or physically short transmission links. It works less well in networks with high speed links connecting switches that are separated by large geographic distances. The fundamental reason is that many packets (hundreds or thousands) can be in transit across a long, high speed link at any instant in time. With conventional data link protocols, buffers to store each of these packets are required in the receiving switch even though under normal conditions, only a few of these packets will be present in the switch at the same time. This leads to unreasonably high buffer requirements. Consequently, high speed packet networks use protocols that do not preallocate buffers. Instead, they simply permit packets to arrive in a relatively unconstrained fashion, and discard packets if insufficient buffer space is available. To keep the frequency of packet loss at an acceptable level, connections are allocated a portion of link bandwidth based on their traffic characteristics.

A key problem in the design of fast packet networks is how to perform this bandwidth allocation. This in turn depends on the behavior of information sources that may be very bursty. We can model a bursty source as a two state Markov chain. When in the *idle* state, a source transmits no data and when in the *active* state, it transmits λ packets per second. Sources that make infrequent transitions between the active and idle states are called *bursty*. When a bursty source becomes active it stays active for a relatively long period of time.

We can model the behavior of a queue of length n receiving traffic from m independent and identical bursty sources, as a finite Markov chain with states $s_{i,j}$ $1 \leq i \leq m$, $1 \leq j \leq n$. We interpret state $s_{i,j}$ to mean that i sources are in their active state and j packets are in the buffer. This Markov chain can be solved numerically to determine the state probabilities and from these the, fraction of transmitted packets lost due to queue overflows.

The bandwidth allocation problem in fast packet networks is to determine if a given set of connections with known traffic characteristics can share a link with acceptable packet loss rate. If the sources can be adequately described by two state Markov chains, this problem can be solved in principle by the methods mentioned above. Unfortunately, the time required for solving a Markov chain model with many different source types is prohibitive, particularly in the context of a practical communications network, where the time available to make such a decision is on the order of 10 ms or less. We address this problem by using a Markov chain model to compute an *effective bandwidth* for a particular connection type, then use the effective bandwidth as the basis for bandwidth allocation decisions. Rather than compute the effective bandwidth when a connection is established, the required value can be extracted from a pre-computed table using interpolation to approximate values that don't appear in the table.

One drawback of the above approach to modeling the queueing behavior of bursty sources is that it neglects the effect of the bandwidth enforcement mechanism (traffic valve) typically placed at the access to the network. An alternative approach is to explicitly focus on the traffic valve and considers only traffic behaviors permitted by it. We hypothesize that assuming sources act independently, the worst-case behavior for any source is to be as bursty as possible, that is to alternate between transmission of a maximum allowed burst at the maximum allowed transmission rate and silence. Such a hypothesis leads to periodic behavior for the sources and a queueing problem that is dependent only on the random phase relationships among the different sources. We have developed an approximate analytical technique to evaluate queueing delay and packet loss rates for periodic bursty sources and are working to improve its accuracy and computational requirements.

See references [2, 3] for further details.

Supporting Research

Distributed Debugging and Monitoring

Victor Griswold

The control and understanding of a system under development is of considerable importance during its testing and debugging. Distributed systems present special problems during debugging beyond those encountered with strictly sequential systems; the developer has less control over the system, and may experience difficulties modeling the system's behavior. Problems include the nondeterministic nature of distributed execution; the potentially long period of time required to generate a problem situation; and the frequently vast amount of monitoring information gathered during a test execution, only a small portion of which turns out to be critical to the test.

The structure of such a distributed debugging and monitoring system is shown in Figure 21. It consists of a number of *subjects* to be monitored (these are typically processes or shared data objects) and a monitoring system comprising a distributed collection of local monitors plus a global monitor. *Event declarations*, describing occurrences that are of interest in a particular application of the monitoring system, are provided by the user. When the user's programs are prepared for execution, these event declarations cause generation of additional program instructions that detect the occurrence of declared events and pass descriptions of event instances to the local monitors.

The user also provides the system with a set of *precedence rules*, which allow the system to determine ordering relationships between events. For example, a common precedence rule would be that a given *message_send* event precedes a given *message_receive* event if the message identifier associated with the two events is the same. In addition, the user may specify a collection of *constraints*, which typically describe correctness conditions. These constraints usually take the form of a pair of events describing a time interval, together with some condition that is required to hold over that time interval. The user may also specify actions that are to be taken if the constraint fails to hold.

When the system is in operation, event instances detected at subjects are passed to the monitor which must infer ordering relationships among events, incorporate these relationships into its internal data structures, then use this information to detect the occurrence of constraint intervals and detect constraint violations.

Towards this end, we are developing algorithms for the efficient evaluation of temporal interval logic constraints by such an event-based distributed system monitor. All events are to be ordered using logical time so that the true nondeterministic nature of the subject is not concealed in a false sense of real time.

Currently, we are working with a small subset of interval logic and intend to expand this subset to the point where we can no longer find suitably efficient evaluation algorithms. We consider the following three problems central to the efficient evaluation of interval logic constraints:

1) Determination of the logical-time order of occurrence of two events **A** and **B**, as well as all events "between" them.

2) Determination of when an event can no longer take place within the subject, so that final evaluations can be made on long-duration constraints.

3) Dynamic (runtime) detection and matching of important event characteristics in order to support the above problems.

Process intervention, the alteration by the monitor of the state of the subject, is a secondary goal of our current research. We will

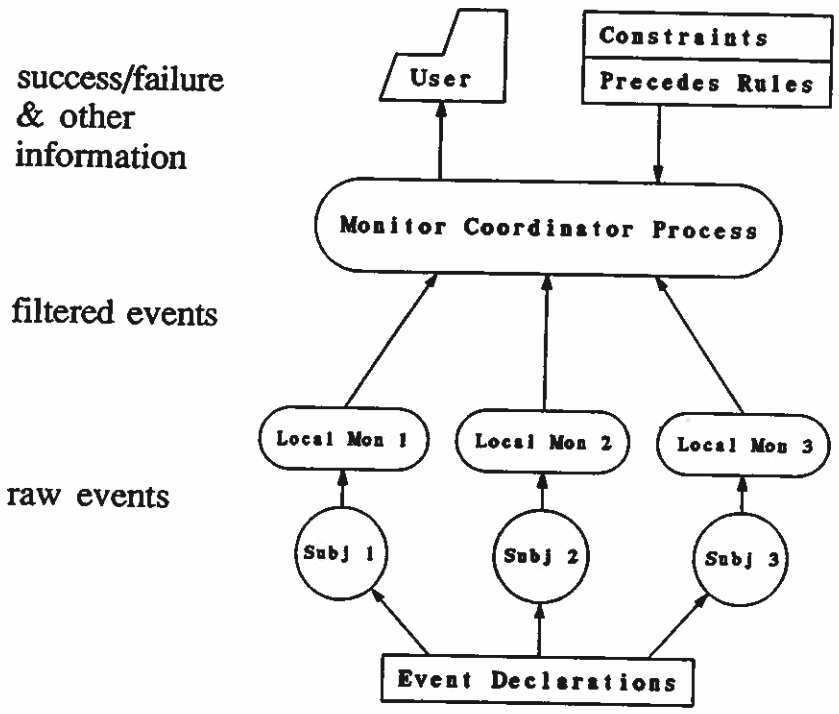


Figure 21: Organization of a Distributed Debugging and Monitoring System

investigate our evaluation algorithms with respect to their implications concerning intervention, but do not expect to develop an actual intervention formalism and algorithms to implement it.

We have had success with the problems of determining all events between two bounding events (the interval between those bounds) and of rapidly classifying events by their characteristics. Two approaches pose the most promise for the interval-detection problem. One of these, which we call the search tree method, offers good memory efficiency and the ability to rapidly answer any "does A come before B" query. The other, called the wavefront method, offers improved memory efficiency when knowledge of the scope of future operations is known, but can only answer a limited category of "comes before" queries.

Our event classification strategy is optimized towards matching events which, according to the user-specified subject behavior description, are ordered in time with respect to one another. Event characteristics are put into a normal form which allows the application of set containment operations to determine time-order matches not just between individual events, but between entire groups of events at one time.

A simulator of the interval-detection algorithms is undergoing continual development. This simulator supports both an interactive mode in which event histories may be displayed graphically and an autonomous parameter-driven mode for statistical data collection. Plans for the simulator include augmenting it for testing the event-classification algorithm.

See [37, 38] for more details.

Automated Circuit Generation

George Robbert

Many of the circuits required in a fast packet switching system contain a large number of functional modules that accept packets on one or more input ports, modify the packet headers and transfer the packets to one or more output ports. The various modules operate in tight synchronism because of the use of fixed length packets. We have come to view each of the specific modules as special cases of a generic *synchronous streams processor* or SSP.

An SSP, is a module with one or more typed input and output *ports*, a local clock synchronized by external timing signals and a function which can be described in a style similar to a conventional programming language. The local clock is set to 0 when the external synchronization signal `start_time` is received, and is then incremented on every tick of the global system clock. The period between successive `start_time` signals is referred to as an *epoch* and all events happen at specific times during an epoch.

Each port has a type associated with it. The base type is *bit* and complex types can be constructed using arrays and structures. In addition to its type, a port has a *start time* and a *width*. The start time defines at what point in each epoch the data item defined for that port begins to appear on the port. The width of the port defines the number of bits available to carry the data. These pieces of information are sufficient to define when in an epoch and where on a port, specific items of data appear. This allows a designer to describe the function of an SSP in terms of actions on port fields, ignoring the details of timing and bit location.

SSPs that perform simple functions, as are typical in the packet processors, fit nicely into a common architecture illustrated in Figure 22. This architecture supports several input and output ports of varying widths. Input ports connect to a common input bus and outputs to a common output bus. Between these are a set of processing elements (PE). Each processing element has data registers which latch selected input fields. The *guard evaluation logic* in addition, contains the combinational logic to evaluate the conditions in conditional statements. The *expression evaluation logic* evaluates expressions on the right side of assignments. The *delay lines* are used to delay the passage of certain fields to the output bus in order to satisfy timing constraints. The control and timing element provides timing signals for latching input data and controlling access to the output bus.

We have developed a circuit generator that takes a high level description of an SSP and creates a circuit implementing it, by tailoring the target architecture. We have divided the translation into several parts. The *compiler* takes the high level module description and translates it to a simple *register transfer language*. This is further processed by an SSP *assembler* which translates it further to a PE description language. This is further processed by a PE *assembler* which generates the actual mask-level description of the module, using a library of standard cells and a set of PE generators, which include existing tools such as PLA generator.

See reference [58] for more details.

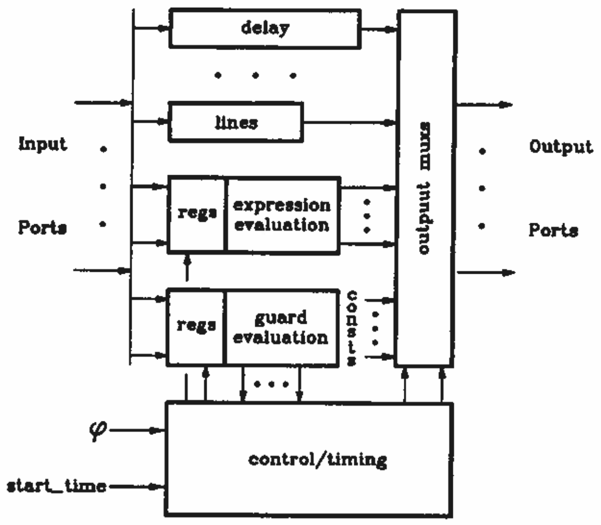


Figure 22: Target SSP Architecture

Signature Based Search Algorithms for Video Codecs

Shabbir Khakoo, Jonathan Turner

Currently, high compression coding techniques have been applied primarily for video conferencing, in order to reduce the bandwidth to the range of a few hundred Kb/s. While most of these techniques make use of motion compensation algorithms, they perform poorly in the presence of even moderate amounts of motion. We have investigated these algorithms, have determined the reasons for this poor performance and have devised a class of algorithms capable of far better performance in the presence of substantial amounts of motion. We believe that these algorithms will yield substantially better performance in a diverse application environment.

Most high compression video codecs in common use today are based on *transform coding*. First, an image is broken into sub-blocks of (typically) 16 by 16 pixels and then for each sub-block, a two-dimensional transform such as the 2D discrete cosine transform (DCT) is computed giving a 16 by 16 matrix of transform coefficients. The transform coefficients form an alternative representation of the initial sub-block, which has the advantage that the perceptually important information is concentrated in a relatively small number of transform coefficients. Consequently, one can transmit a fraction of the transform coefficients (or transmit them with varying precision) and recreate the image from those few coefficients without significantly degrading the image.

If one is transmitting a sequence of similar images, as in a video sequence, additional compression can be obtained by maintaining a copy of the previous image and transmitting the difference between the current image and the previous one. Typically, each sub-block to be transmitted is compared to the corresponding sub-block in the reference image, a difference sub-block is calculated and the DCT of the difference sub-block is computed and sent with varying precision. This can be improved further through a form of motion compensation called *block matching*. This involves comparing the sub-block not just to the corresponding sub-block in the reference image but to all sub-blocks of the reference image within a given distance of the sub-block to be transmitted. The coder then identifies the sub-block of the reference image that differs least from the given sub-block and transmits the difference information relative to that sub-block, along with the identity of the selected reference sub-block.

A key element in the performance of a block matching codec is the algorithm used to identify the sub-block of the reference frame that is most similar to the current sub-block. To compare two sub-blocks, one typically computes the sum of the squares of the differences between corresponding pixels. For 16 by 16 sub-blocks this requires 256 multiplies and 512 additions. To compare a given sub-block to every reference sub-block within say eight pixels in any direction, requires 289 comparisons or a total of about 220,000 arithmetic operations. Since in a 512 by 512 pixel image, this must be done 1024 times per frame, the computational requirements are pretty clearly prohibitive for real-time coding.

Consequently video codecs that employ block matching don't attempt to consider every sub-block in the region. Rather they try to find the best match using some form of local search. Such techniques can work well if there are no local minima in which the search algorithm can get stuck. We have found however that for many typical images, local minima are common, and that the local minima are typically not nearly as good as the global minimum. Consequently, conventional search techniques often fail to achieve the highest possible compression rates.

We have devised a class of search algorithms that attempts to solve this problem. It is based on the idea of computing a concise *signature* for each sub-block in the search region and then comparing the input sub-block to the reference sub-blocks on the basis of their signatures. The signatures must be quickly computable and small enough that it's reasonable to perform an exhaustive comparison of the signatures. Then several sub-blocks with closely matching signatures are compared to the input sub-block using the squared-difference measure and the best match selected. We have evaluated one version of this technique in which the signatures are selected coefficients of a one-dimensional DCT, computed over the sum of the pixel values in each row of a sub-block. Our results show that the signature-based algorithm gives compression rates of three to five times that achieved by other algorithms in the presence of substantial amounts of motion.

Details can be found in [45].

References

[1] Anderson, Jim, Zubin Dittia and Guru Parulkar. "Persistent Connections in High Speed Internets," Washington University Computer Science Department, WUCS-91-13.

[2] Akhtar, Shahid. "Congestion Control in Fast Packet Networks," Washington University Electrical Engineering Department, MS thesis, November 1987.

[3] Arutaki, Akira. "Trajectory Method for Evaluating Periodic Bursty Traffic," Washington University Computer Science Department, WUCS-89-53.

[4] Bi, Haifeng. "Design of PP3: a Packet Processor Chip," Washington University Computer Science Department, WUCS-89-42.

[5] Bi, Haifeng. "Queueing Analysis of Buffered Switching Networks," Washington University Computer Science Department, MS thesis, 8/90.

[6] Barrett, Neil. "Design of a VLSI Packet Buffer," Washington University Computer Science Department, WUCS-88-32.

[7] Bovopoulos, A.D. and Lazar, A.A., "Optimal Routing and Flow Control of Time Division Multiplexed Channels," *Proceedings of the 12th International Conference on Mathematical Programming*, 3/85.

[8] Bovopoulos, A.D. and Lazar, A.A., "Optimal Routing and Flow Control of a Network of Parallel Processors with Individual Buffers," *Proceedings of the Twenty-Third Annual Allerton Conference on Communication, Control and Computing*, 10/85.

[9] Bovopoulos, A.D. and Lazar, A.A., "Optimal Routing and Flow Control of a Network of Parallel Processors," *Proceedings of ORSA/TIMS 1985*, 11/85.

[10] Bovopoulos, A.D. and Lazar, A.A., "Optimal Load Balancing for Markovian Queueing Networks," *Proceedings of the 30th Midwest Symposium on Circuits and Systems*, 8/87.

[11] Bovopoulos, A.D. and Lazar, A.A., "Optimal Load Balancing of a Network with Nonzero Acknowledgment Delays," *Proceedings of the Computer Networking Symposium*, 4/88.

[12] Bovopoulos, A.D. and Lazar, A.A., "Load Balancing Algorithms for Jacksonian Networks with Acknowledgment Delays," *Proceedings of the IEEE INFOCOM'89 Conference*, 4/89.

[13] Bovopoulos, A. D., "Resource Allocation for Markovian Queueing Networks: The Partial Information Case," Washington University Computer Science Department Technical Report WUCS-89-22, 1989.

[14] Bovopoulos, A. D., " On the Effect of Delayed Feedback Information on Network Performance," Washington University Computer Science Department Technical Report WUCS-89-23, 1989.

[15] Bovopoulos, A.D. and Lazar, A.A., "Decentralized Algorithms for Optimal Flow Control," *Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control and Computing*, University of Illinois, Urbana-Champaign, Illinois, 9/87.

[16] Bovopoulos, A.D. and Lazar, A.A., "Asynchronous Iterative Algorithms for Optimal Load Balancing," *Proceedings of the Twenty-Second Annual Conference on Information Sciences and Systems*, 3/88.

[17] Bovopoulos, A.D. and Lazar, A.A., "Decentralized Network Flow Control," *Proceedings of the 9th International Conference on Computer Communication*, 11/88.

Conference on Computer Communication, 11/88.

[18] Bovopoulos, A.D. and Lazar, A.A., "Asynchronous Algorithms for Optimal Flow Control of BCMP Networks," Washington University Computer Science Department Technical Report WUCS-89-10, 1989.

[19] Bovopoulos, A. D., "Resource Allocation as a Nash Game in a Multiclass Packet Switched Environment," Washington University Computer Science Department Technical Report WUCS-89-18.

[20] Bovopoulos, A. D., "Resource Allocation Algorithms for Packet Switched Networks," *First ORSA Telecommunications SIG Conference, "Operations Research in Telecommunications,"* 3/90.

[21] Bovopoulos, Andreas and Aurel Lazar. "The Effect of Delayed Feedback Information on Network Performance," Washington University Computer Science Department, WUCS-90-06, and *First ORSA Telecommunications SIG Conference*, 3/90.

[22] Bovopoulos, Andreas and Aurel Lazar. "The Effect of Delayed Feedback Information on Network Performance," Washington University Computer Science Department, WUCS-90-06.

[23] Bovopoulos, Andreas and Einir Valdimarsson. "Performance Evaluation of a User Network Interface for ATM Networks," Washington University Computer Science Department, WUCS-90-32.

[24] Bovopoulos, Andreas. "Performance Evaluation of a Traffic Control Module for ATM Networks," Washington University Computer Science Department, WUCS-91-22.

[25] Bovopoulos, Andreas and Aurel Lazar. "Optimal Resource Allocation for Markovian Queueing Networks: the Complete Information Case," *Stochastic Models*, 1991.

[26] Bubenik, Richard. "Performance Evaluation of a Broadcast Packet Switch," Washington University Computer Science Department, MS thesis, 8/85.

[27] Bubenik, Richard and Jonathan S. Turner. "Performance of a Broadcast Packet Switch." *IEEE Transactions on Computers*, 1/89.

[28] Bubenik, Richard, John DeHart and Mike Gaddis, "Multipoint Connection Management in High Speed Networks," *Proceedings of Infocom*, 4/91.

[29] DeHart, John. "CMAP/CMIP Scenarios: A Tutorial," Washington University Computer Science Department, ARL-90-03.

[30] Dehart, John. "BPN Connection Management Access Protocol Specification," Washington University Computer Science Department, ARL-89-06.

[31] Gaddis, Michael E. "Prototype Connection Management: a Progress Report," Washington University Applied Research Laboratory, ARL-89-01.

[32] Gaddis, Mike. "FPP Packet Formats and Link Level Protocol Descriptions," Washington University Computer Science Department, ARL-89-07.

[33] Garg, Gaurav "Design of a VLSI Broadcast Translation Circuit," Washington University Computer Science Department, WUCS-89-52.

[34] Garg, Gaurav. "BTC Modifications," Washington University Computer Science Department, WUCS-90-19.

[35] Garg, Gaurav. "BTC Test Methodology," Washington University Computer Science Department, WUCS-90-20.

[36] Gong, Larry "Segment Streaming for Efficient Pipelined Televisualization," Washington University Computer Science Department, WUCS-90-39.

[37] Griswold, Victor. "Determining Interior Vertices of Graph Intervals," Washington University Computer Science Department, WUCS-90-40.

[38] Griswold, Victor. "Core Algorithms for Long Term Monitoring of Distributed Systems," Washington University Computer Science Department, DSc thesis, 2/91.

[39] Haserodt, Kurt and Jonathan Turner. "An Architecture for Connection Management in a Broadcast Packet Network," Washington University Computer Science Department, WUCS-87-3.

[40] Imase, Makoto and Bernard Waxman. "Worst-case Performance of Rayward-Smith's Steiner Tree Heuristic," *Information Processing Letters*, 12/8/88.

[41] Imase, Makoto and Bernard Waxman. "The Dynamic Steiner Tree Problem," Washington University Computer Science Department, WUCS-89-11.

[42] Jenq, Yih-Chyun. "Performance Analysis of a Packet Switch Based on a Single-Buffered Banyan Network," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1014–1021.

[43] Kapoor, Sanjay and Guru Parulkar. "Design of an ATM-FDDI Gateway," Washington University Computer Science Department, WUCS-91-11.

[44] Khakoo, Shabbir and Jonathan Turner. "System Testing of a Broadcast Packet Switch," Washington University Computer Science Department, WUCS-87-4.

[45] Khakoo, Shabbir. "Improved Search Algorithms for Video Codecs," Washington University Electrical Engineering Department, MS thesis, June 1988.

[46] Lee, Tony T. "Non-Blocking Copy Networks for Multicast Packet Switching," Bell Communications Research, 1987.

[47] Mazraani, Tony. "Design of a Clock Generator Chip" Washington University Computer Science Department, WUCS-88-36.

[48] Mazraani, Tony and Guru Parulkar. "Specification of a Multipoint Congram-Oriented High Performance Internet Protocol," *Proceedings of Infocom 90*, 6/90.

[49] Mazraani, Tony. "High Speed Internet Protocols and Resource Management in the Ethernet," Washington University Computer Science Department, MS thesis, 8/90.

[50] Mazraani, Tony. "VLSI Area Comparison of Beneš and Crossbar Communications Networks," Washington University Computer Science Department, WUCS-90-09.

[51] Melen, Riccardo and Jonathan S. Turner. "Distributed Protocols for Access Arbitration in Tree Structured Communication Channels," *Proceedings of ICC 88*, June 1988.

[52] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Multirate Networks," *SIAM Journal on Computing*, 4/89.

[53] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Networks for Fast Packet Switching," *Proceedings of Infocom 89*, April 1989.

[54] Melen, Riccardo and Jonathan Turner. "Nonblocking Multirate Distribution Networks," *Proceedings of Infocom 90*, 6/90.

[55] Parulkar, Guru and Jonathan Turner. "Towards a Framework for High Speed Communication in a Heterogeneous Networking Environment," *IEEE Network* 3/90.

[56] Parulkar, Guru. "The Next Generation of Internetworking," *Computer Communication Review*, 1/90.

[57] Robbert, George. "Design of a Broadcast Translation Chip," Washington University Computer Science Department, WUCS-87-9.

[58] Robbert, George. "A Circuit Generator for Synchronous Streams Processors," Washington University Computer Science Department, MS thesis, May 1988.

[59] Sterbenz, James. "Design of a VLSI Packet Switch Element," Washington University Computer Science Department, WUCS-88-5.

[60] Sterbenz, James P.G. and Gurudatta M. Parulkar, "Axon: Network Virtual Storage Design", to appear in *Computer Communication Review*, Vol.20 #2, ACM, New York, April 1990.

[61] Sterbenz, James. "Host-Network Interface Design," Washington University Computer Science Department, WUCS-90-07.

[62] Sterbenz, James and Guru Parulkar. "Axon: Network Virtual Storage Design," *Computer Communication Review*, 4/90.

[63] Sterbenz, James and Guru Parulkar. "AXON: a High Speed Communication Architecture for Distributed Applications," *Proceedings of Infocom 90*, 6/90.

[64] Sterbenz, James P.G. and G.M. Parulkar, "Axon Network Virtual Storage for High Performance Distributed Applications", *Proceedings of ICDCS*, 6/90.

[65] Sterbenz, James P.G. and G.M. Parulkar, "Axon: Application-Oriented Lightweight Transport Protocol Design", *Proceedings of ICCC*, 11/90.

[66] Sterbenz, James P.G. and G.M. Parulkar, "Axon: Host-Network Interface Architecture for Gbps Communication," *Proceedings of IFIP Workshop on High Speed Networking*, 11/90.

[67] Szymanski, Ted and Salman Shaikh. "Markov Chain Analysis of Packet-Switched Banyans with Arbitrary Switch Sizes, Queue Sizes, Link Multiplicities and Speedups," *Proceedings of Infocom 89*, 4/89.

[68] Turner, Jonathan S. "New Directions in Communications," *IEEE Communications Magazine*, 10/86.

[69] Turner, Jonathan S. "Design of an Integrated Services *Packet* Network," *IEEE Journal on Selected Areas in Communications*, 11/86.

[70] Turner, Jonathan S. "Advanced Communication Systems Progress Report," Washington University Computer Science Department, WUCS-87-22.

[71] Turner, Jonathan S. "Specification of Integrated Circuits for a Broadcast Packet Network," Washington University Computer Science Department, WUCS-87-5.

[72] Turner, Jonathan S. "The Challenge of Multipoint Communication," *Proceedings of the ITC Seminar on Traffic Engineering for ISDN Design and Planning*, 5/87.

[73] Turner, Jonathan S, "Fluid Flow Loading Analysis of Packet Switching Networks," *Proceedings of the International Teletraffic Congress*, June 1988.

[74] Turner, Jonathan, "Broadcast Packet Switching Network," Unites States Patent #4,734,907, March 1988.

[75] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE Transactions on Communications*, June 1988.

[76] Turner, Jonathan S. "Advanced Communication Systems Progress Report," Washington University Computer Science Department, WUCS-88-28.

[77] Turner, Jonathan S. "Practical Wide-Sense Nonblocking Generalized Connectors," Washington University Computer Science Department, WUCS-88-29.

[78] Turner, Jonathan, "High Speed Data Link," Unites States Patent #4,829,227, May 1989.

[79] Turner, Jonathan, "Buffer Management System," U. S. Patent #4,849,968, July 1989.

[80] Turner, Jonathan. "Queueing Analysis of Buffered Switching Networks," Washington University Computer Science Department, WUCS-90-04.

[81] Turner, Jonathan. "Cross-Connect for Switch Modules," U.S. Patent #4,901,309, 2/90.

[82] Turner, Jonathan. "A Practical Multicast Switching System for ATM Networks," unpublished note, jst-90-1, 5/90.

[83] Turner, Jonathan. "Practical Multicasting in Sunshine Switch Architecture," unpublished note, jst-90-2, 5/90.

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

[85] Turner, Jonathan. "A Proposed Bandwidth Management and Congestion Control Scheme for Multicast ATM Networks," Washington University Computer and Communications Research Center, WUCCRC-91-1.

[86] Turner, Jonathan. "Packet Switch with Broadcasting Capability for ATM Networks," U.S patent application, 1/91.

[87] Turner, Jonathan. "Data Packet Resequencer for a High Speed Data Switch," U.S patent application, 3/91.

[88] Turner, Jonathan. "Nonblocking Multicast Switching System," U.S patent application, 4/91.

[89] Valdimarsson, Einir. "Design of an Eight Bit VLSI Packet Switch Element," Washington University Computer Science Department, WUCS-88-23.

[90] Valdimarsson, Einir "Evaluation of Blocking Probability in Multirate Networks," Washington University Computer Science Department, MS thesis, 4/90.

[91] Valdimarsson, Einir "Blocking in Multirate Networks," *Proceedings of Infocom*, 4/91

[92] Waxman, Bernard. "Thesis Proposal: Routing of Multipoint Connections," Washington University Computer Science Department, WUCS-87-2.

[93] Waxman, Bernard. "Probable Performance of Steiner Tree Algorithms," Washington University Computer Science Department, WUCS-88-4.

[94] Waxman, Bernard. "Routing of Multipoint Connections," *IEEE Journal on Selected Areas of Communications*, 12/88.

[95] Waxman, Bernard. "New Approximation Algorithms for the Steiner Tree Problem," Washington University Computer Science Department, WUCS-89-15.

[96] Waxman, Bernard. "Evaluation of Algorithms for Multipoint Routing," Washington University Computer Science Department, doctoral thesis, 8/89.