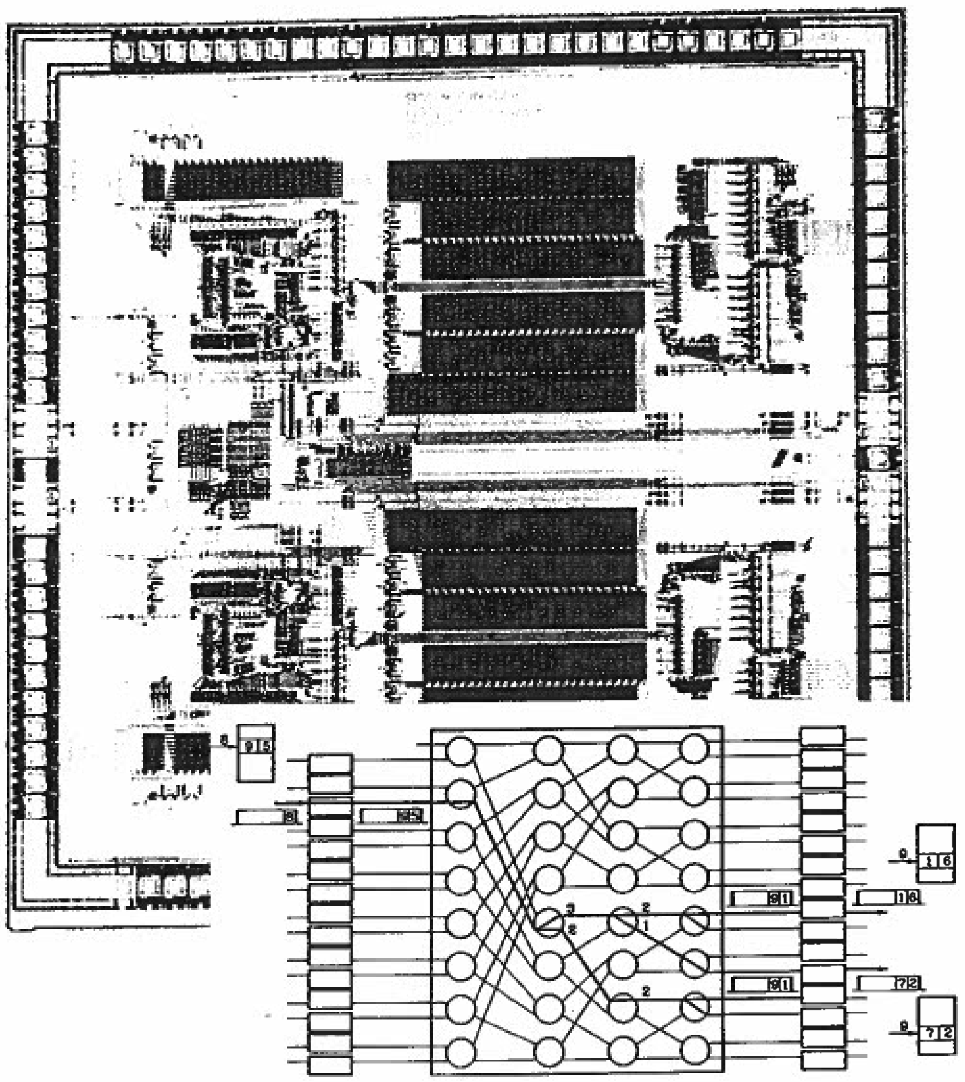
Advanced Networks Group

Research Summary, 1989



Computer and Communications Research Center
Washington University, St. Louis

The Advanced Networks Group

The Advanced Networks Group (ANG) of the Computer and Communications Research Center (CCRC) is concerned with new communication technologies that can support a wide range of different communication applications in the context of large public networks. Fast packet or ATM networks promise a far more flexible communications infrastructure than is currently available. The Advanced Networks Group is in particular 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 our prototype fast packet switching system. We have developed several integrated 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
- Victor Griswold
- James Sterbenz
- Bernard Waxman

Masters Students
- Hai Feng Bi
- Charles Cranor
- Gaurav Garg
- Tony Mazraani
- Einir Valdimarsson

Related Activities

The Computer and Communications Research Center also 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 a dozen 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.

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 six, 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

Neil Barrett
Rick Bubenik
Pierre Costa
John DeHart
Mike Gaddis
Randy Richards

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 in person 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 two representatives to semi-annual progress review meetings.
- 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.

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

Bell Communications Research
Bell Northern Research
Italtel SIT
Nippon Electric Corporation

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 strongly 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 very 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.

Some of our maturing activities have been recently spun off to a new project being carried out by the Computer Science Department's Applied Research Laboratory. This project will demonstrate the application of fast packet technology to support visual communication applications, in particular video applications and medical imaging. The project is headed by Dr. Jerome Cox and includes the construction of a demonstration network of four fast packet switches supporting multicast connections. This effort is being carried out by the Applied Research Lab's full-time professional staff, in collaboration with the Advanced Networks Group. The demonstration network will provide a testbed that will be used to support research on applications and network control and operations, and will play a central role in our continuing research.

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

Experimental Switching Systems

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

The Advanced Networks Group is developing a prototype fast packet switching system supporting link speeds of 100 Mb/s and a general multicast capability, which we call the *Broadcast Packet Switch*. The organization of the switching system 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, a Distribution Network (DN) which provides a load balancing function and finally 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 *logical channel number* (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, Distribution 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. The same chip can be used in any of the three networks by means of two external configuration pins. 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 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

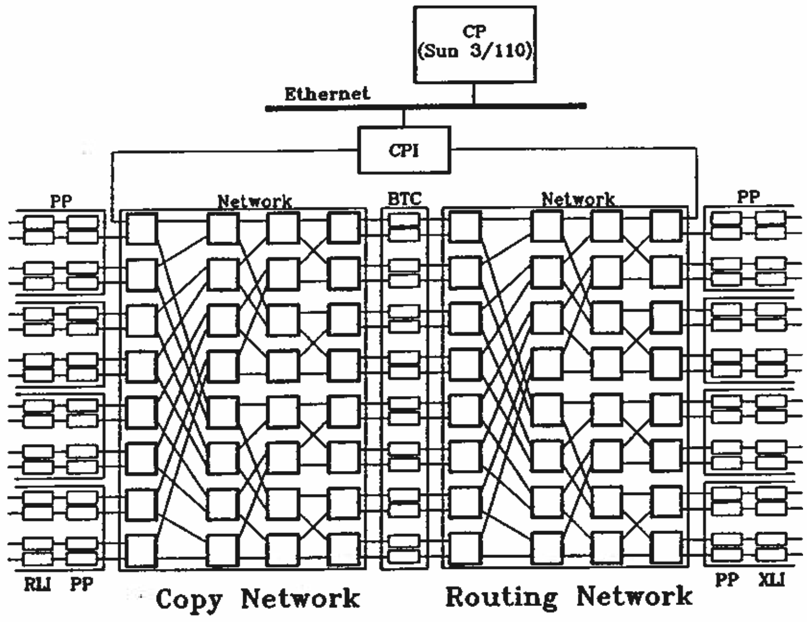


Figure 1: Broadcast Packet Switch Architecture

transistors to over 200,000.

Our experimental 16 port switch will consist of a total of eight circuit cards. The network cards will 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.

See references [2, 26, 34, 36, 54, 48, 49] for further details.

Performance of a Broadcast Packet Switch

Rick Bubenik, Jonathan Turner

We have made an extensive performance study of the broadcast packet switch architecture. We have developed analytical methods which allow us to assess the worst-case loading conditions for a wide variety of specific architectures of which the broadcast packet switch is representative. We have also used simulation to evaluate a variety of design trade-offs in detail. Some of these results are summarized below.

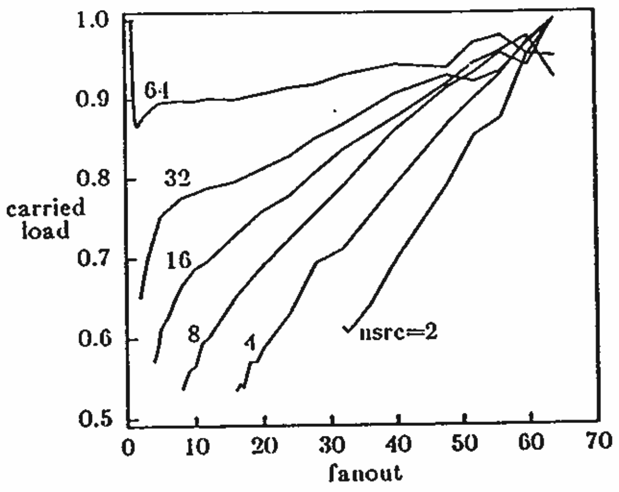
Our simulations indicate that the overall performance of the system is determined primarily be the routing network. We have shown that a system with six stages of binary switch elements, each containing two buffer slots per input, can achieve a throughput of about 55%; this is more than adequate, given 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 the broadcast packet switch, the delay through the copy, distribution and routing networks is about 3 packet times under normal operating conditions, with cut-through and about 20 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. These results are illustrated in Figure 2.

We have developed an analytical method for evaluating the worst-case loading for architectures similar to the broadcast packet switch. 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



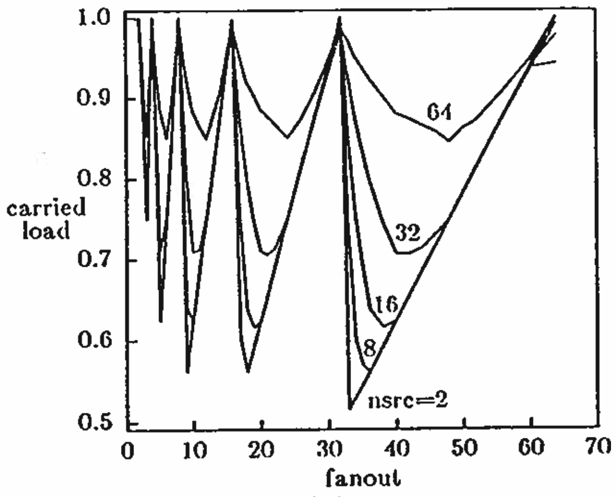


Figure 2: Copy Network Throughput for Random and Deterministic Fanouts

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 $d+1$. However, this can be reduced to less than two by making appropriate choices for the broadcast channel numbers and adding a single distribution stage.

See references [18, 19, 49] for further details.

Design of Gigabit Switching Systems

Jonathan Turner

The broadcast packet switch architecture as described above is most suitable for systems with up to about 100 ports. As the architecture is scaled to larger configurations the copy, distribution and routing networks become the dominant cost components. The number of chips required for these elements on a per port basis is $1.5 \log_2 n$, where n is the number of ports; so for example in a system with 1024 ports, the number of chips required is 15 per port, plus four in the packet processor and one in the Broadcast Translation Circuit. The complexity of this architecture can be reduced by using larger switch elements organized in a bit-sliced fashion. Using such an organization, the chip count for the copy, routing and distribution networks becomes $3(m/d) \log_d n$ where d is the size of the basic switch element and m is the data path width. If we let $m = 8$ and $d = 32$, which is a feasible choice, the chip count for the 1024 port system becomes 1.5 chips per port. Systems with up to 32 thousand ports require only 2.25 chips per port as opposed to 22.5 for a switch constructed from binary switch elements.

Figure 3 illustrates a design of a bit-sliced switch element with d input and output ports and supporting m bit wide data paths. Typical values for d might be 16 or 32. Values for m might range from 8 to 64. Packets enter on one of the d *upstream data* lines (ud_i) at left, and the m bits of each packet are distributed across m separate *data slices* (DS). The packets exit from the switch element on the *downstream data* lines (dd_i) at the right. The switch element contains sufficient internal buffering to store several packets for each port and implements a simple hardware flow control mechanism to prevent packets from overflowing these buffers.

The *control slice* shown at the bottom of the figure contains the circuitry used to control the operation of the switch element. It receives a set of *downstream grant* signals (dg_i) from the downstream neighbors and generates a corresponding set of *upstream grant* signals (dg_i) which are sent to the upstream neighbors. In general, a switch element asserts an upstream grant signal ug_i if it is prepared to receive a packet on the upstream data lines ud_i. The packets flowing through the switch element are organized so that all the control information (in particular, the addressing information) passes through the first data slice DS_0. This allows the control circuit to easily monitor the control information for all packets entering the data slice. Using this information, together with the downstream grants and the internal status of the switch elements, it makes control decisions and broadcasts those decisions to the data slices. In addition, the first bit of the packet in *every data slice* is a control bit indicating the presence or absence of a packet.

There are several options for the organization of the data slices. The buffering can be located on either the input or output side or it can placed centrally and shared by all the inputs and outputs. This last configuration provides by far the best performance for a given amount of buffering. It requires input and output crossbars and a set of buffers, each large enough to hold one slice of a packet. The buffers can be implemented as dynamic shift registers with a feedback path used to recirculate a packet if it is unable to proceed during a given cycle. The output crossbar allows a packet to be sent to multiple outputs during a given cycle, permitting multicast connections. Also, a multicast packet that must be sent to several outputs need not be sent to all outputs simultaneously. If not all outputs are immediately available, it can be sent to the available outputs and recirculated in the buffer until the remaining outputs become available.

The control slice contains no data storage but makes decisions regarding how the packets passing through the data slices are to be routed. The key element of the control slice is

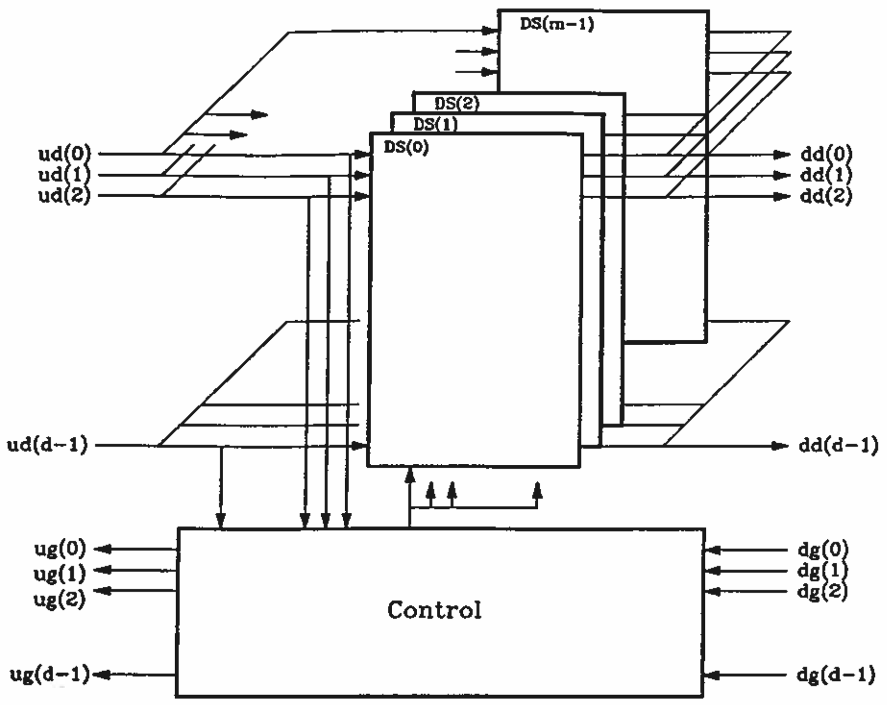


Figure 3: Organization of a Bit-Sliced Switch Element

an arbitration circuit which receives requests from each of the *d* inputs, and grant signals from the downstream outputs and makes the necessary control decisions. The key design challenge is to orchestrate the decision-making in parallel so that all the required decisions can be carried out in a single operation cycle.

Preliminary designs of the bit-sliced switch element indicate that a 32 port switch element with 32 bit wide data paths, shared buffering, 4096 bit packets and three buffer slots per port requires a data slice chip with about 220,000 transistors and a control slice of about 410,000 transistors. While the data slice can be handled using two micron CMOS, the control slice probably requires 1.6 micron technology.

Another important parameter of the design is the amount of control information that must be sent from the control slice to the data slices during each packet cycle. This has implications for the minimum packet length. In particular, a 32 port switch element with three buffer slots per port requires 320 bits of control information. If this information is carried on eight pins, the packet cycle time must be at least 40 clock ticks. Allowing time for control processing, this increases to perhaps 64 ticks. With a 32 bit wide data path, the resulting minimum packet length is 2048 bits.

See reference [50] for further details.

Nonblocking Multirate Networks

Riccardo Melen, Jonathan Turner

This research extends the classical theory of nonblocking networks to switching systems in which the internal data paths carry multiplexed traffic in which individual user channels consume an arbitrary fraction of the bandwidth, subject only to the constraint that the total traffic not exceed the capacity of the data path. This model is applicable to multirate circuit switches and to packet switched networks in which all packets in a given connection follow the same path through the switching system. The new theory has immediate application to several experimental switch architectures now under development around the world.

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 $\log\log n$, wher 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.

See references [29, 30] for further details.

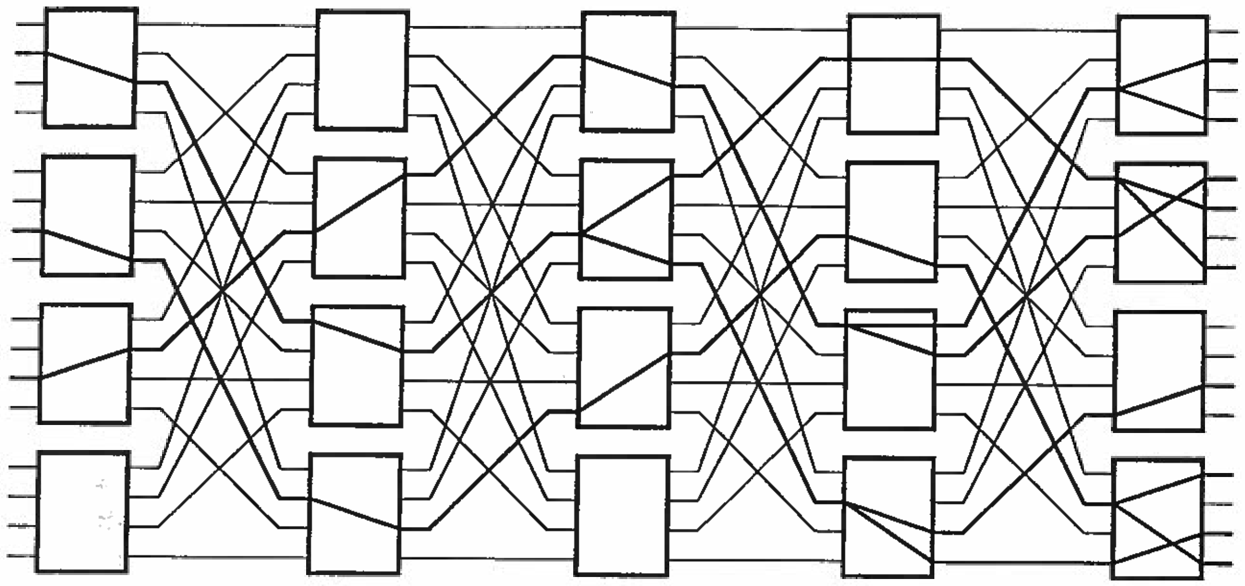


Figure 4: Nonblocking Multipoint Network

Blocking in Multirate Networks

Einir Valdimarsson

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 interested in determining the likelihood of blocking in multirate networks.

Blocking probabilities in circuit switching networks have been well studied, but very little work has been done on examining blocking in the multirate environment where connections share the internal data paths of the switch. 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 currently 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.

We have found our results on how blocking is affected by network parameters such as switch element size, number of inputs and number of stages, 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 [29] 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.

The bandwidth distribution has an impact on the blocking. Apparently large connections are more likely to block than small ones since the large connections need more resources. This behavior is seen in Figure 5 which shows interactions between two different connection classes. The two connection classes both have deterministic bandwidth requirements that are varied between 0 and 1. The probability of blocking for class A is shown. Notice how as the bandwidth of class A increases, the blocking probability also increases. Increasing the bandwidth for class B connections also

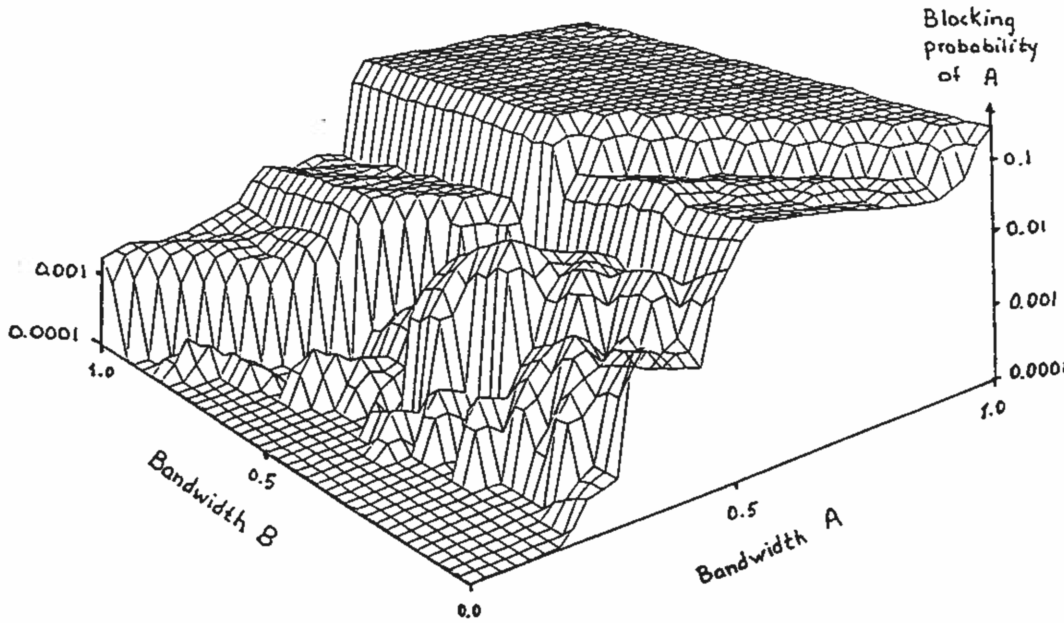


Figure 5: Blocking Probability for Mixed Traffic

increases the blocking probability for A but not as dramatically. The blocking probability map shows several plateaus with steep transitions in between. The shapes of the plateaus are determined by the fragmentation caused by the bandwidth distributions. For example, the top plateau is bounded by the bandwidth combinations that allow at most one connection on a switch's internal data paths.

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

$$\int_0^1 wp(w)b(w)dw.$$

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. Our preliminary conclusion for this portion of the research is 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.

Control of Multicast Networks

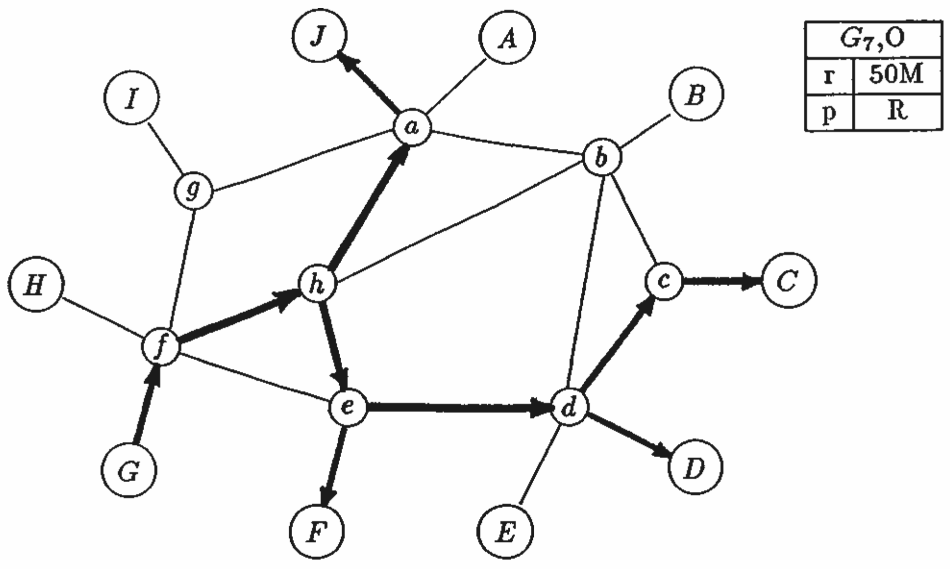
Connection Management in Multicast Networks

Mike Gaddis, Victor Griswold, Kurt Haserodt, Mark Hunter, 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 6. The connection has a single transmitter G and receivers C, D, F and J. The internal nodes in the diagram represent switching systems. At various internal nodes, the stream of packets originating at G 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 G_7 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 G 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 necessary resources to include the new endpoint in the connection. For entertainment video



G_7,O	
r	50M
p	R

Figure 6: One-to-Many Connection

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 [21, 44] 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 $\mathrm{cap}: E \rightarrow \Re^+$ and $\mathrm{cost}: E \rightarrow \Re^+$ 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 7 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

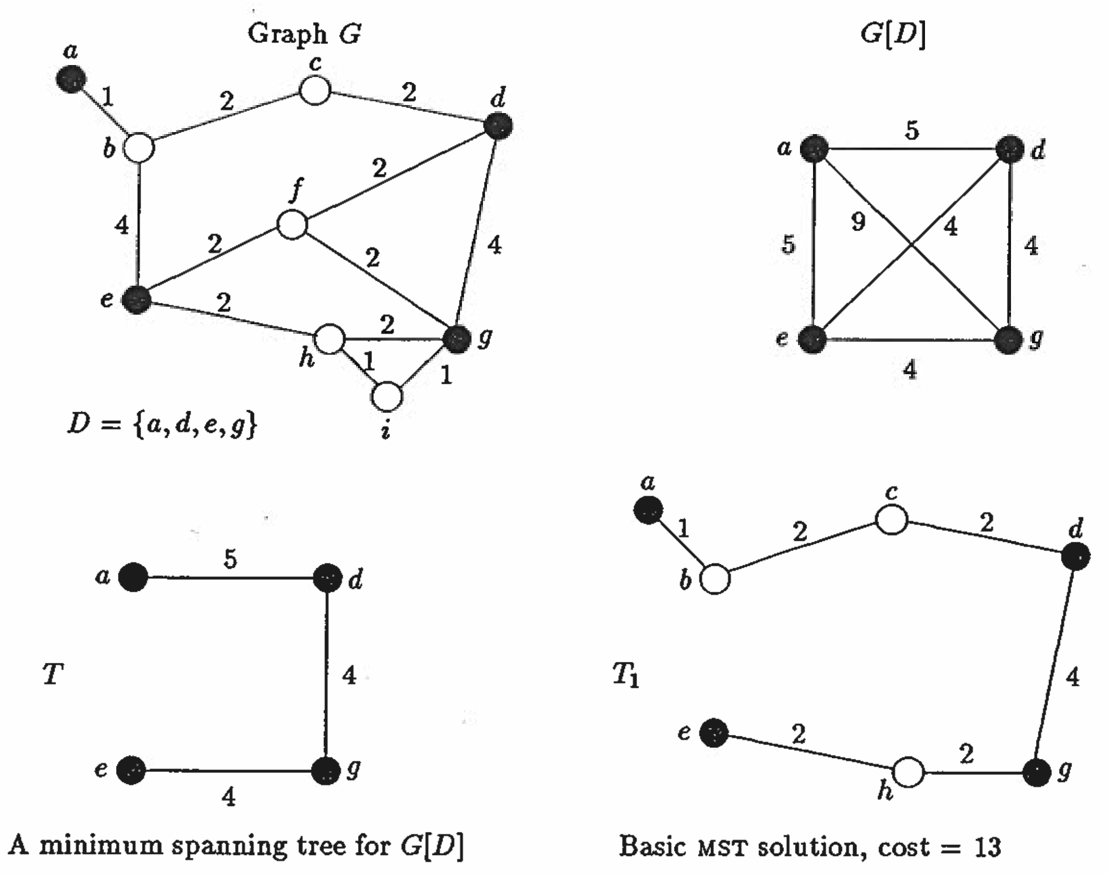


Figure 7: An Example of the Application of MST

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 [22, 23, 55, 56, 57, 58, 59] for further details.

High Speed Host Interfaces and Internetwork Protocols

High Speed Internetworking

Guru Parulkar, Tony Mazranni, Charles Cranor

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. Important components of this abstraction are shown in Figure 8. It resembles the existing internet abstraction in rudimentary ways. For example, the internet level protocol in the VHSI also interfaces with transport protocols and underlying networks, uses transport facilities of the subnetworks to forward packets, and uses gateways to switch packets between subnetworks. However, there are significant differences between the VHSI and existing internet abstractions as outlined in the following paragraphs.

MCHIP. MCHIP is a novel multipoint congram-oriented high performance internet protocol, equal in status to IP in terms of the protocol hierarchy. For applications, such as multimedia conferencing and interactive remote visualization that require strict performance guarantees, MCHIP establishes a congram and makes explicit resource reservations. For applications, such as file transfer and electronic mail that can tolerate loose performance guarantees, MCHIP uses a congram that can multiplex traffic from multiple application conversations and that does not make resource reservations for each individual conversation. Thus, the congram service primitive aims at combining the strengths of both classical connection and datagram approaches.

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.

Router. The VHSI abstraction includes appropriate routing functionality for the intra- and inter-autonomous region routing, with due considerations to the policy and access constraints of autonomous regions.

Gateway Architecture. In order to keep up with very high data transmission rates of communication links, it is essential that packet processing and forwarding be simplified. The solution used in the VHSI involves separating critical and noncritical paths, and simplifying the critical path as much as possible. It is important to note that the critical path is simplified by caching the state information about application conversations; this is naturally achieved in MCHIP with congrams. Depending on the gateway platform and performance requirements, the critical path can be implemented using a general purpose processor, a special onboard processor, or custom VLSI. We believe that the division of functionality into critical and noncritical paths is the key to achieving high-speed and high-performance gateways.

The VHSI research is in its early stages, and considerable work is necessary to demonstrate viability of the proposed approach and understand associated tradeoffs. We have have developed the outlines of the VHSI approach, and have identified and begun work on several research and prototype subtasks that are detailed below.

- Design and specification of MCHIP. We have completed specification of a simplified version of MCHIP. The specification includes packet types, sequence of packet exchange, and representative scenarios, and is reported in [27].
- Implementation of MCHIP under Unix on a workstation. We want to create a separate socket address family under BSD Unix for MCHIP – similar to IP and XNS address families. Applications would use the

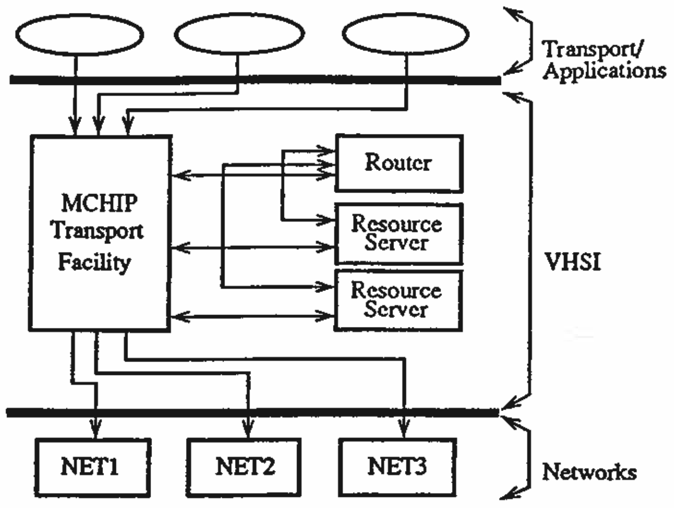


Figure 8: VHSI Abstraction

standard socket interface with socket type MCHIP to use MCHIP services. MCHIP will be interfaced to the standard network device drivers. This implementation will also be partially useful for the gateway as described below.

- Effectiveness of a resource server on a broadcast local area network. We are working on a simulation of a CSMA/CD network with a number of hosts and a resource server. The resource server keeps track of all active congrams, their resource usage, and the network utilization. Hosts can generate both congram and datagram traffic. In the case of a congram, the host first checks with the resource server to determine if adequate resources are available for the congram. Thus, a congram is established only if its resource needs can be met, otherwise, it is blocked. The resource server can throttle datagram sources if they consume more than their fair share of resources. The purpose of the simulation is to understand what kind of performance guarantees can be made to congram traffic at a given network utilization using a central resource server and in the presence of datagram traffic.

- Implementation of a BPN-FDDI gateway. We have started working on a two port BPN-FDDI gateway which will use MCHIP (with other hosts) as the internet protocol. The gateway implementation consists of two parts; G_s is responsible for all MCHIP congram management and G_h for per-packet processing and packet forwarding between networks. G_s would be a slightly modified version of the host MCHIP implementation. G_h involves designing a microprocessor controlled board with an FDDI interface (using an FDDI chip set or existing controller), BPN access chips, and appropriate glue logic.

Further details may be found in [32, 33, 27].

The Axon Host-Network Interface Architecture

Guru Parulkar, James Sterbenz

The research and development of high speed switching systems and networks will result in the ability to construct networks and internetworks that support data rates up to a few Gbps. We refer to this as the VHSI (very high speed internetwork). Similarly, processor and workstation power and functionality is rapidly increasing which make network applications, such as distributed scientific computation and visualization, video distribution on demand, multimedia conferencing, and remote imaging possible. For such applications to fully utilize VHSI performance and functionality, however, the host-network interface architecture must be capable of delivering the high bandwidth to the applications with minimum latency.

The Axon architecture satisfies this need by providing (1) an integrated design of host and network interface architecture, operating systems, and communication protocols stressing both performance and the required functionality for demanding applications such as visualization and imaging, (2) a proper division of functionality in hardware and software for optimal performance, (3) reorganization of end-to-end protocols to take advantage of the increased functionality of emerging high speed networks. The overall Axon architecture is described in [40]. Significant features of the Axon architecture are summarized below, and presented in Figure 9.

System level IPC support and NVS. The system level support for the various application level interprocess communication paradigms is provided by two components: NVS and NMP. NVS (network virtual storage) is the system level shared memory interface for shared variables, GRPC (generalized remote procedure call), and segment streaming. NMP (network message passing) is the system level message passing interface. NMP performs a relatively straightforward transformation of program (*send*, *receive*) primitives to corresponding transport protocol message-object transfer calls. NVS extends the typical virtual storage mechanisms to include systems throughout the VHSI. A segmented programming model is used, with underlying paging to facilitate storage management, as in the Multics operating system.

Transport protocol. At the transport level, applications using the VHSI are best supported by a set of simple ALTPs (application-oriented lightweight transport protocols) for various classes of applications. Key issues in the design of an ALTP are the implementation of critical functions in hardware, rate based flow control, application-oriented error control, and structured collections of packets.

ALTPs have their critical path functions implemented in VLSI hardware. The critical path consists of the data path, and routine control functions allowing data to flow at peak network rates, once a transport operation has been initiated. By optimizing the critical path functions, and by processing multiple packets in a single transport level operation, the per packet processing can be performed in real time at the full sustained data rate. For the protocol to be efficiently implemented in hardware, the protocol, hardware design, and host operating system should be well integrated.

Host and network interface architecture. 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.

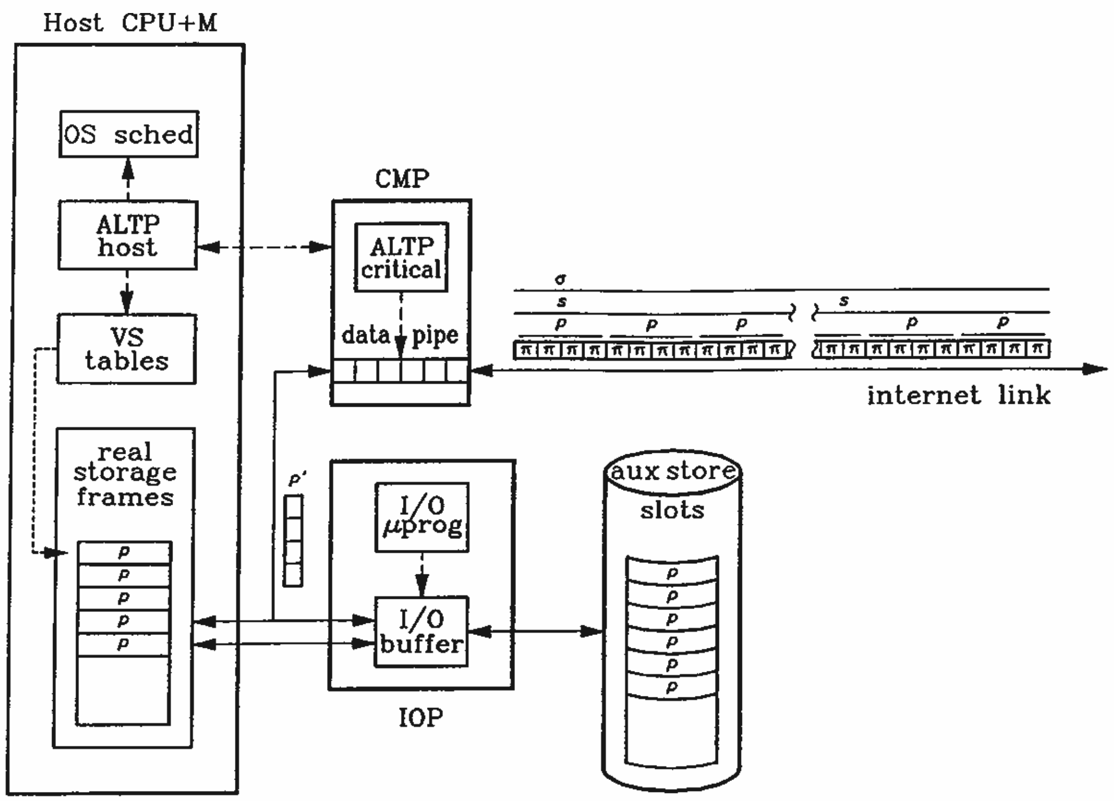


Figure 9: Axon Architecture

The CMP implementation consists of a pipelined set of datapath modules and control modules.

Work is in progress on analytical and simulation models to evaluate the associated tradeoffs more rigorously, on a detailed design of the communication processor, and a prototype implementation of the architecture. Successful completion of this effort will result in the following:

- Demonstration of viable Axon design (functional, performance) from the design specifications, implementations, and overall simulations. This provides the proof of concept.
- Determination of function that is part of the critical path, as data rates scale above 1 Gbps, based on the time–space complexity analysis.
- Specific solutions for the network-host object mapping, determination of components to be included in the critical path, OS interaction, and determination of whether implicit or explicit mapping is better.
- Understanding of relationship between latency and memory requirements in terms of locality and working sets, based on latency, processor performance, and network bandwidth, with respect to the incorporation of function in the critical path.
- Prototype implementation of an Axon interface and demonstration of a number of target applications, especially medical imaging and distributed solid modeling.

Axon: Network Virtual Storage Design

Guru Parulkar, James Sterbenz

This note presents an overview of the design of *network virtual storage* (NVS) in the Axon host communications architecture for distributed applications. Nvs extends segmented paged virtual storage management and address translation mechanisms to include segments located across an internetwork. This provides the ability to efficiently use the shared memory paradigm in non-local environments, as well as the support for a very high speed end-to-end data path between demanding applications such as scientific visualization and imaging. Additionally, segments that are transported across the VHSI are mapped into the address spaces of processes by NVS. This eliminates the need to copy segments from intermediate system buffers into the process address space, resulting in lower latency and system overhead.

Data structures. Addressing of a segment resident on a non-local host is accomplished by including a *host id* field in either the virtual address, or in the segment descriptor table (SDT) entry. When a segment fault occurs for a nonlocal segment (indicated in the segment descriptor), the dynamic address translation facility invokes the transport protocol (ALTP-OT) to get a copy of the segment from the appropriate system. As the segment is returned, the appropriate page and segment descriptor presence bits are set, so that program execution can resume with the normal fault recovery mechanisms. The address translation data structures are presented in Figure 10. Address pointers are represented by arrows on solid lines, the movement of data/requests is represented by arrows with dashed lines.

The local storage management data structures are extended to allow the addressing of segments on other hosts. This is accomplished by adding a *host id* field to the *known segment table* (KST), which holds the symbolic segment bindings. This is an index into the per process *known host table* (KHT), which holds the symbolic host name to address/path bindings. This binding is resolved by searching the *host address table* (HAT) for each host, which gets its binding by invoking an internet name server, using the *host name database* (HND). There are also tables (not shown in the figure) to assist in n-way IPC using multipoint connections.

Segment types. Axon segments are of two types: *memory* and *video*. Memory segments are either *code* or *data* subtype. Memory segments are divided into pages, and may be organized into segment groups, for performance reasons. Video segments are either *text* or *graphics* subtype. Graphics segments are bit-mapped video image frames; text segments correspond to a text window on a workstation. Video graphics segments are divided into scanlines, and may be organized into multi-frame images (eg. a color image of R,G,B frames).

Segments have attributes of *read, write, execute*, indicating the type of access allowed. Code segments are assumed to be pure (refreshable), and therefore always have access attributes of execute-only. Data segments may be readable and/or writable.

Storage management policies. Nvs in Axon involves extensions and additions to storage management policies. The fetch policy is not affected by NVS, except that demand-segment implies a degree of anticipatory-page movement across the network and is, in fact, desired to counter latency effects. The (real) placement policy is not affected by NVS at all, since placement is trivial for paged storage management, and unaffected by NVS.

An entirely new policy, the *remote placement policy*, is used to determine where remote segments are placed while being used by the local system. These include real store, auxiliary store, a combination, or frame buffer placement, with a number of sub-policy options, such as swappable and nailed. Due to

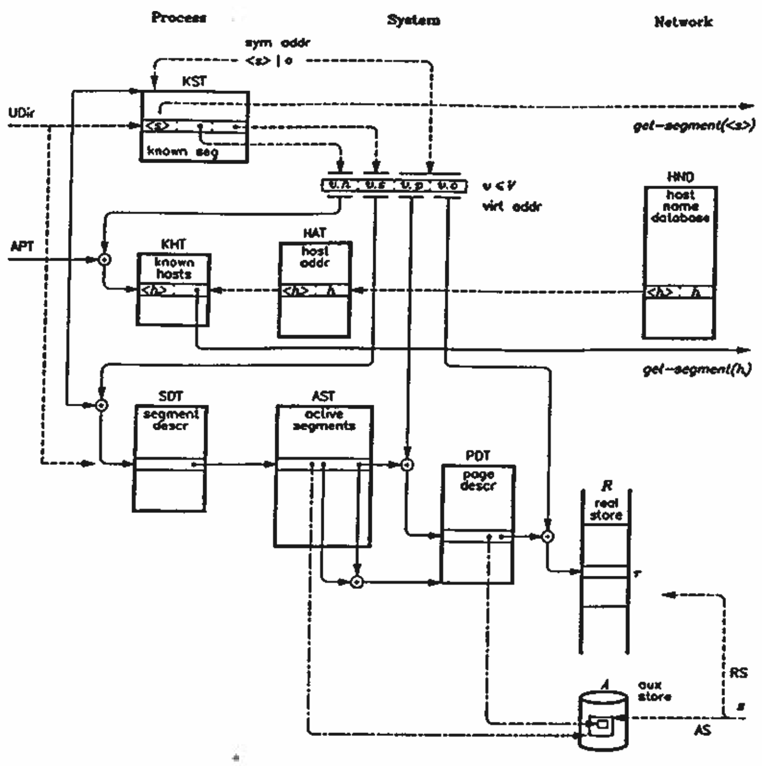


Figure 10: Network Virtual Storage Address Translation

the presence of segments from remote hosts, the conventional replacement policy is affected. In particular, placement of entire remote segments in real store will result in presence of some of the pages that are not really in the process locality set in the real store. This indicates that the estimation of working sets must consider local and remote segments differently.

Nvs and its storage management policies are described in more detail in [37, 41].

ALTP-OT and Communication Processor Design

Guru Parulkar, James Sterbenz

Most current transport protocols (including TCP, X.25, and SNA) are *general purpose*, providing complete flow and error control to all applications. This results in complexity of implementation and operational overhead that is not necessary for particular applications. It may be possible to *functionally partition* a transport protocol, to provide only the functionality needed for various classes of applications, while still allowing the use of a single protocol by adjusting the appropriate parameters. A similar strategy that does not require that a single transport protocol serve all applications, is to have a small set of *application-oriented* transport protocols. A possible set might consist of application oriented transport protocols for object transfer, for voice and for video distribution.

Somewhat orthogonal to the *scope* of the transport protocol is the simplicity of design and efficiency of operation. A protocol that is simple, streamlined, and efficient is referred to as a *lightweight* protocol. Note that while it is possible to design even a general purpose protocol to be lightweight to some degree, it is much easier to do so with an application oriented protocol that can efficiently serve the corresponding application class with the appropriate (and simplified) error and flow control mechanisms. This is the approach taken in Axon with ALTPs (application-oriented lightweight transport protocols).

In the Axon transport level, IPC across the VHSI is supported by ALTP-OT (ALTP for object transfer), which has its critical path function implemented in VLSI hardware, is optimized to provide the kind of performance guarantees and functionality required for object transfer. The ALTP-OT requests include connection establishment/termination, segment/page/message transfer, and packet retransmission. ALTP-OT is described in detail in [38]. ALTP-OT design is summarized in the following paragraphs.

Flow control. When ALTP-OT opens a connection, it specifies attributes of the connection in terms of parameters such as average and peak bandwidth, and a factor reflecting the burstiness of the transmission. These parameters can be translated into buffer requirements, based on a rate between the average and peak specifications. Initial exploration of this allocation has been researched in [1]. Since the connection set up is end-to-end, all the intermediate systems, including various packet switches and gateways, as well as the endpoint hosts that this connection goes through, can make appropriate buffer and resource reservations. The rate specification will have to be negotiated between ALTP-OT and the internetwork/network layers, to ensure that the requested rate does not exceed the capacity of internal network nodes and gateways. As a result, as long as both ends transmit subject to the rate specification, the probability of packet loss due to buffer overruns is very low.

It is assumed that the internet level below has the functionality to support connections with specified bandwidth requirements, and furthermore, that the probability of packet loss, errors, and resequencing is low enough to design the critical path with the assumption that handling such problems is the exceptional case.

The only explicit flow control exercised by ALTP-OT is the control of the CMP (communications processor) data transmission rate to correspond to the rate specification. The benefit of the ALTP approach is manifest in that only attributes significant for object transfer need to be considered and that the rate control functionality necessary on a per-packet basis can be implemented in hardware.

Error control. In the VHSI environment error control is performed, as much as possible, on an end-to-end basis, and is decoupled from flow (rate) control, as described above. The ALTP error control is as simple as possible, based on the target application characteristics. For ALTP-OT, the packet handling is as follows:

- duplicate packets are discarded
- corrupted packets are discarded, and retransmission requested based on application need
- missing packets are detected by the expiration of a timer, and retransmission is requested
- packet sequence is irrelevant because each packet is directly placed into the proper location of application memory space.

The Axon architecture allows application specific selective retransmission of corrupted or missing packets, which gives considerable flexibility in retransmission strategy. Note that due to the orientation of ALTP-OT to this application, the handling of duplicate and out-of-sequence packets is considerably simpler and more efficient than would be the case for a general purpose transport protocol. Since data packets have sufficient header information to indicate the connection and request, and are placed directly into the proper location of target store, the overhead of sequence buffering is not necessary. The simplified error control of ALTP-OT can be efficiently implemented in VLSI hardware.

Communication Processor (CMP) Design. The Axon architecture interfaces the CMP directly to the processor or memory, specifically as a host-network interface processor. On the network interface side, the CMP must be capable of receiving and transmitting packets at the full network data rate. On the host side, the CMP must either interface to the processor-memory interconnect or the special dual ported communication memory, called CMM, depending on the host architecture.

The primary design consideration of the CMP is to serve as the network interface to the VHSI, as part of an end-to-end pipelined data path

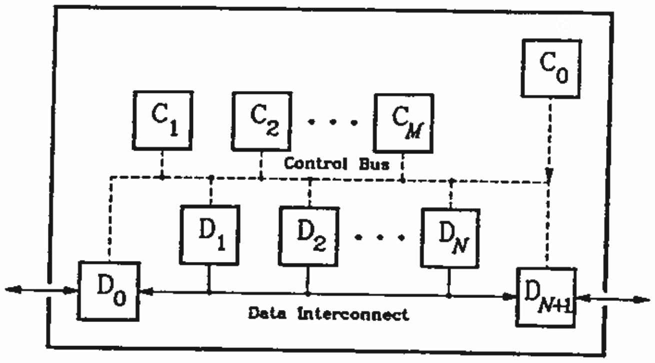


Figure 11: High Level CMP Design

between application memory spaces. Thus the CMP must have the ability to perform critical path functions in real time with no packet buffering, and incorporate the necessary functions in VLSI. This may be realized by organizing the CMP as a pipeline, dynamically reconfigurable based on the ALTP type and options for a particular connection. The pipeline organization allows packets to be processed while moving at the VHSI interface data rates. Examples of data modules include the network receive and transmit interfaces, CMM interface, parallel/serial conversion, data format conversion, encryption/decryption, and video widow coordinate translation. Control modules include rate control, checksum generate/compare, CMM address generate, header generate/decode, packet presence timers, and retransmission logic.

Greater detail on the CMP design (for ALTP-OT in particular) is presented in [39].

Resource Management in Communication Networks

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 [3, 4, 5, 6, 9].

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 [10].

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 [7, 8, 11].

Finally we studied the effect of the availability of either complete, partial, or no information about the network's state at the controller. In [12] 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 [15].

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 [13, 14, 16]. 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 [17] 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.

Buffer and Bandwidth Management

Shahid Akhtar, 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 define the *burst factor* B of a source to be the average time spent in the active state, times the difference between the source's peak and average rates.

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 $s_{i,j}$ $1 \le i \le m$, $1 \le j \le n$. We interpret state $s_{i,j}$ to mean that i sources are in their active state and j packets are in the buffer. This model is illustrated in Figure 12. 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. We address this problem by

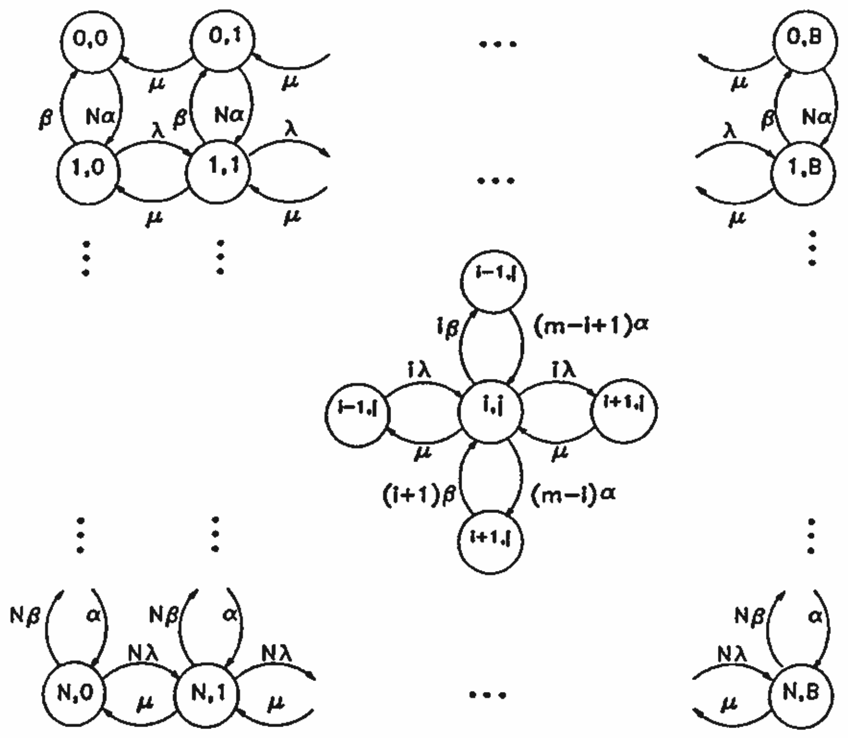


Figure 12: Markov Chain Model

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.

The problem of bandwidth management becomes more complex in the presence of multipoint connections with multiple transmitters, since in this case it becomes possible for packet streams from different transmitters to converge with one another inside the network, creating loads larger than are permitted at the boundary. We have developed a general solution to this problem that allows the network to monitor buffer usage at the various switching systems within a network, and during overloads to preferentially discard packets belonging to connections that are currently using more than their "share" of a link's bandwidth. This approach admits a fairly simple hardware implementation, and is the subject of a recently awarded patent.

See references [1, 44, 53] for further details.

Bandwidth Allocation

Akira Arutaki, Jonathan Turner

The bandwidth allocation problem in fast packet/ATM networks is to determine under what conditions a connection with specified but statistically varying bandwidth requirements can be safely multiplexed with an existing set of connections (also with specified bandwidth requirements). The problem is a difficult one because many of the applications for which fast packet networks are intended have very bursty traffic requirements. While one can always allocate peak bandwidth to such connections, this can be extremely wasteful and should be avoided if possible.

Current approaches to statistical bandwidth allocation typically rely on a stochastic model of the sources that neglects effects due to the mixture of traffic from different types of sources and ignores the effect of the bandwidth enforcement mechanism (traffic valve) placed at the access to the network. We are pursuing a new approach that explicitly focuses on the traffic valve and considers only traffic behaviors permitted by the traffic valve. 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 currently working to improve its accuracy and computational requirements. We have also proposed a criterion for accepting a connection in a network, that is based not on the usual sorts of performance measures (such as packet loss rate or delay), but a measure that we refer to as the *excess packet probability*. We believe this approach will allow incremental maintenance of information about the traffic on a given link, using a fast convolution computation, so that a decision to accept or reject a new connection can be made in no more than a few milliseconds.

This approach to bandwidth allocation has some other promising features. It can be easily extended to handle switching systems with shared buffering, something we feel will be essential to handle highly bursty traffic streams. It can also handle more complex traffic valves with only a minor additional computational cost. Such traffic valves are useful for controlling traffic streams that may have burstiness on several different time scales. For example, when accessing images from an image database a user might look quickly through several images before pausing to examine one image in detail. In such a situation, each image would constitute a burst and the closely spaced collection of images would constitute a "burst group." With the first-order traffic valves currently receiving the most attention, such an application would be handled by making the maximum burst size equal to the size of the collection of images. This can lead to an over-allocation compared to what can be obtained using a slightly more sophisticated traffic valve.

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 13. 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

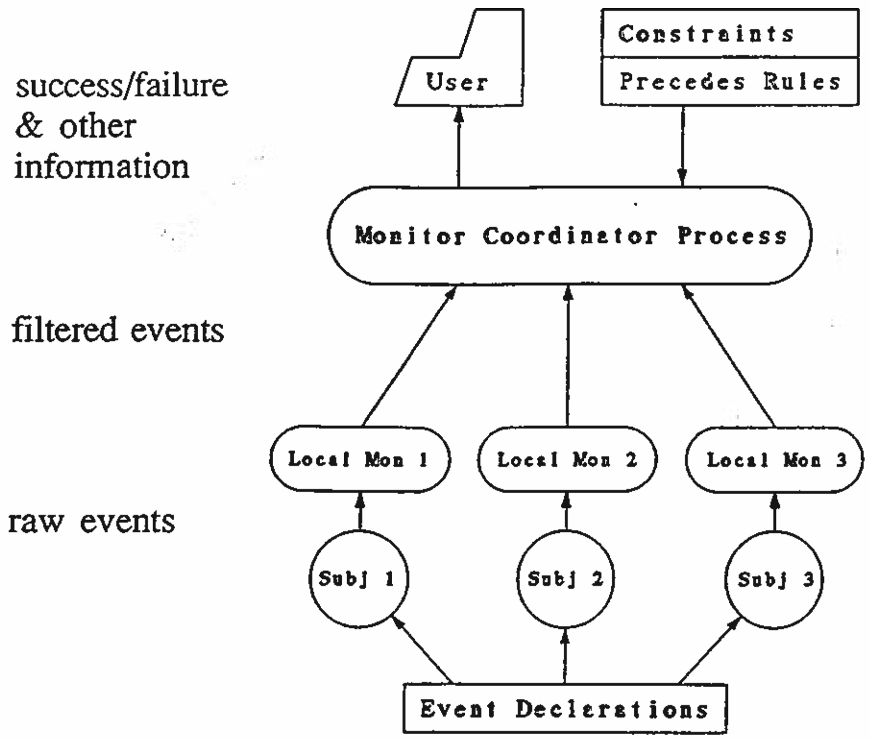


Figure 13: Organization of a Distributed Debugging and Monitoring System

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.

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 14. 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 [35] for more details.

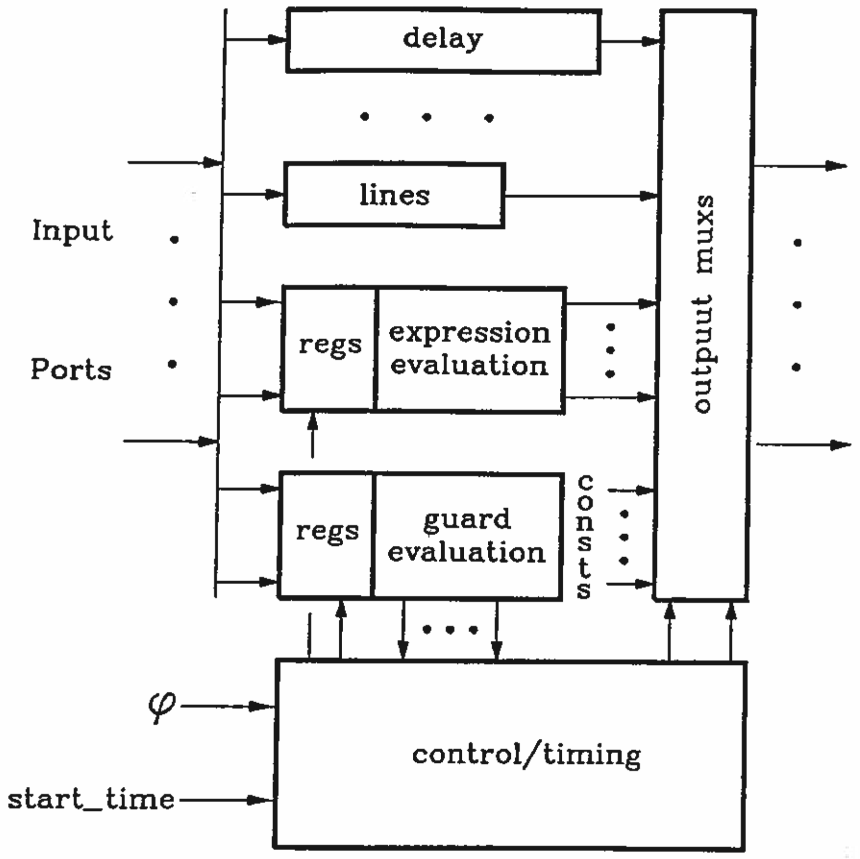


Figure 14: Target SSP Architecture

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

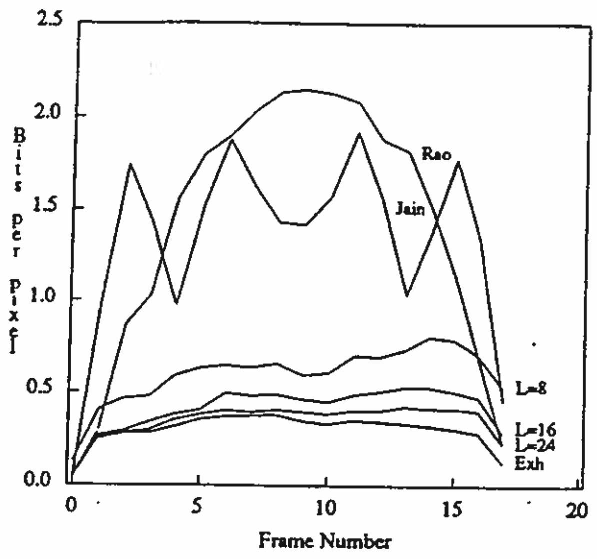


Figure 15: Compression Rate Comparison

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. Figure 15 shows the results of one of our experiments. This plot shows the average number of bits per pixel for a sequence of frames starting from no motion, increasing to a panning motion of eight pixels between successive frames and then decreasing to no motion. The curves labeled Rao and Jain give the performance for two popular algorithms that are commonly used in video conferencing codecs. The curve labeled exhaustive gives the performance obtained by exhaustively searching for the best match and the three other curves give the performance of three variations of our signature-based algorithm. The parameter N refers to the number of sub-blocks with closely matching signatures that were selected for the final comparison. Notice that the signature-based algorithm gives compression rates of three to five times that achieved by the other algorithms in the presence of substantial amounts of motion.

More detailed results may be found in [25].

References

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

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

[3] 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*, Cambridge, Massachusetts, August 1985, p. 13A.

[4] 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*, University of Illinois, Urbana-Champaign, Illinois, October 1985, pp. 564-573.

[5] Bovopoulos, A.D. and Lazar, A.A., "Optimal Routing and Flow Control of a Network of Parallel Processors," *Proceedings of ORSA/TIMS 1985*, Atlanta, Georgia, November 1985, pp. 139-140.

[6] Bovopoulos, A.D. and Lazar, A.A., "Optimal Load Balancing for Markovian Queueing Networks," *Proceedings of the 30th Midwest Symposium on Circuits and Systems*, Syracuse University, Syracuse, New York, August 1987.

[7] Bovopoulos, A.D. and Lazar, A.A., "Optimal Load Balancing of a Network with Nonzero Acknowledgment Delays," *Proceedings of the Computer Networking Symposium*, Washington, D.C., April 1988, pp. 144-151.

[8] Bovopoulos, A.D. and Lazar, A.A., "Load Balancing Algorithms for Jacksonian Networks with Acknowledgment Delays," *Proceedings of the IEEE INFOCOM'89 Conference*, Ottawa, Canada, April 1989, pp. 749-757.

[9] Bovopoulos, A. D. and Lazar, A. A., "Optimal Resource Allocation for Markovian Queueing Networks: The Complete Information Case," Washington University Computer Science Department Technical Report WUCS-89-21, 1989.

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

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

[12] Bovopoulos, A. D., "Resource Allocation Algorithms for Packet Switched Networks," accepted for presentation at the *First ORSA Telecommunications SIG Conference, "Operations Research in Telecommunications,"* Boca Raton, Florida, March 1990.

[13] 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, September 30-October 2, 1987, pp. 979-988.

[14] 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*, Princeton University, Princeton, New Jersey, March 16-18, 1988.

[15] Bovopoulos, A.D. and Lazar, A.A., "Decentralized Network Flow Control," *Proceedings of the 9th International Conference on Computer Communication*, Tel Aviv, Israel, October 30-November 3, 1988,

[16] 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.

[17] 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,

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

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

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

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

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

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

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

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

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

[27] Mazraani, T.Y., Parulkar, G.M., "Specification of a Multipoint Congram-oriented High Performance Internet Protocol," Washington University Computer Science Department WUCS-89-20.

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

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

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

[31] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Multirate Distribution Networks," Washington University Computer Science Department, WUCS-89-34.

[32] Parulkar, Guru and Jonathan Turner. "Towards a Framework for High Speed Communication in a Heterogeneous Networking Environment," *Proceedings of Infocom 89*, April 1989.

[33] Parulkar, Guru. "The Next Generation of Internetworking," Washington University

Computer Science Department, WUCS-89-19.

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

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

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

[37] Sterbenz, James P.G. and Gurudatta M. Parulkar, "Axon: Network Virtual Storage Design," Washington University Computer Science Department WUCS-89-13.

[38] Sterbenz, James P.G., "Host–Network Interface Architecture for Gigabit Communications," Washington University Computer Science Department WUCS-89-35.

[39] Sterbenz, James P.G. and Gurudatta M. Parulkar, "Axon: Application-Oriented Lightweight Transport Protocol Design," Washington University Computer Science Department WUCS-89-14.

[40] Sterbenz, James P.G. and Gurudatta M. Parulkar, "Axon: A High Speed Communication Architecture for Distributed Applications," Washington University Computer Science Department WUCS-89-36.

[41] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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