Report Number: WUCS-87-7

1987-06-30

Advanced Communications Systems

Jonathan S. Turner

The Advanced Communication Systems Project is concerned with new communication technologies that can support a wide range of different communication applications in the context of large public networks. Communications networks in common use today have been tailored to specific applications and while they perform their assigned functions well, they are difficult to adapt to new uses. There currently are no general purpose networks, rather there are telephone networks, low-speed data networks and cable television networks. As new communications applications proliferate, it becomes clear that in the long term, a more flexible communications infrastructure will be needed. The Integrated Services... **Read complete abstract on page 2.**

Advanced Communications Systems

Jonathan S. Turner

Complete Abstract:

The Advanced Communication Systems Project is concerned with new communication technologies that can support a wide range of different communication applications in the context of large public networks. Communications networks in common use today have been tailored to specific applications and while they perform their assigned functions well, they are difficult to adapt to new uses. There currently are no general purpose networks, rather there are telephone networks, low-speed data networks and cable television networks. As new communications applications proliferate, it becomes clear that in the long term, a more flexible communications infrastructure will be needed. The Integrated Services Digital Network concept provides a first step in that direction. We concerned with the next generation of systems that will ultimately succeed ISDN. The main focus of the effort in the ACS project is a particular switching technology we call broadcast packet switching. The key attributes of this technology are (1) the ability to support connections of any data rate from a few bits per second to over 100 Mb/s, (2) the ability to support flexible multi-point connections suitable for entertainment video, LAN interconnection and voice/video teleconferencing, (3) the ability to efficiently support bursty information sources, (4) the ability to upgrade network performance incrementally as technology improves and (5) the separation of information transport functions from application-dependent functions so as to provide maximum flexibility for future services.

ADVANCED COMMUNICATIONS SYSTEMS

Jonathan S. Turner

WUCS-87-7

July 1, 1986 - June 30, 1987

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

with Mark A. Franklin, Pierre Costa, Riccardo Melen, Shahid Akhtar, Victor Griswold, Mark Hunter, Shabbir Khakoo, George Robbert, James Sterbenz, Bernard Waxman and David Wexelblat.

Acknowledgements

The Advanced Communications Systems Project operates within the Computer and Communications Research Center, an inter-departmental research laboratory in the School of Engineering and Applied Science at Washington University. The ACS project began on January 1, 1986 with support from Bell Communications Research and Italtel SIT. Major funding is now provided by the National Science Foundation through grant DCI 8600947.

The Center's research program seeks an appropriate balance between theoretical and practical issues and has attracted considerable interest world-wide. Program sponsors interact with the Center through exchange of information and personnel. Our current sponsors are

National Science Foundation
Bell Communications Research
Italtel SIT
Nippon Electric Corporation

We thank all our sponsors for their collaboration and support. Special thanks go to Gil Devey and Steve Wolf at NSF, Eric Nussbaum of Bell Communications Research, Maurizio Dècina of Italtel and Akihiro Kitamura of NEC.

Research Objectives

The Advanced Communications Systems Project is concerned with new communications technologies that can support a wide range of different communications applications in the context of of large public networks. Communications networks in common use today have been tailored to specific applications and while they perform their assigned functions well, they are difficult to adapt to new uses. There currently are no general purpose networks, rather there are telephone networks, low-speed data networks and cable television networks. As new communications applications proliferate, it becomes clear that in the long term, a more flexible communications infrastructure will be needed. The Integrated Services Digital Network concept provides a first step in that direction. We are concerned with the next generation of systems that will ultimately succeed ISDN.

The main focus of the effort in the ACS project is a particular switching technology we call broadcast packet switching. The key attributes of this technology are (1) the ability to support connections of any data rate from a few bits per second to over 100 Mb/s, (2) the ability to support flexible multipoint connections suitable for entertainment video, LAN interconnection, and voice/video teleconferencing, (3) the ability to efficiently support bursty information sources, (4) the ability to upgrade network performance incrementally as technology improves and (5) the separation of information transport functions from application-dependent functions so as to provide maximum flexibility for future services. The reader is referred to reference [46] for additional background on the project.

Contents

List of Figures

1. Summary of Progress

The research program of the ACS project can be divided into four major areas: (1) switching system architecture, (2) connection management, (3) network control problems, such as routing and congestion control and (4) design of communications applications in the context of broadcast packet networks. The effort in the last year has concentrated on switching system architecture and related issues, although significant progress has also been made in connection management and network control. Work on application design is being deferred until issues in the other areas are better understood.

We have been active in publishing our results on broadcast packet switching. Papers have been presented at several conferences and revised versions have appeared or are scheduled to appear in leading journals. Patent applications have been filed on broadcast packet switching and invited lectures have been given at many industrial and academic laboratories. (See Figures 1.1,1.2 for details.) Our work has generated a great deal of interest throughout the world, and appears to be having an influence on the research programs at several major industrial laboratories. We find this impact of our work particularly gratifying and expect to see it continue as our research program develops.

The following subsections summarize the progress we have made in several specific areas during the past year and outline our plans for the coming year. More detailed accounts of each of these topics appear in later sections.

Switch Architecture and Hardware Design

The most novel aspect of our research program is its focus on networks supporting flexible multi-point communication. Any switching system supporting multipoint communication must be able to connect any subset of its incoming channels to any subset of its outgoing channels. This is in contrast to point-to-point switching systems which need only connect input-output pairs.

Our work is based on a particular switching system architecture for multipoint communication. During the past year we have also been studying some competing

Published Papers

"Design of a Broadcast Packet Switching Network," by Jonathan S. Turner, *Proceedings of Infocom 86*, pp. 667–675, 4/86. Also, to appear in *IEEE Transactions on Communications.*

"New Directions in Communications," by Jonathan S. Turner, *IEEE Communications Society Magazine*, 10/86.

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

"Performance of a Broadcast Packet Switch," by Richard Bubenik and Jonathan Turner. To appear in *Proceedings of ICC 87* and *IEEE Transactions on Communications.*

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

Invited Lectures

Bell Atlantic, Great Gorge, NJ (2/87)
Midwest Workshop on Communications Systems, St. Louis, MO (11/86)
Computer Communications Workshop, Warner Springs, CA (9/86)
ITT Advanced Technology Center, Shelton, CT (9/86)
Digital Equipment Corporation, Littleton, MA (5/86)
Bolt, Beranek and Newman, Cambridge, MA (5/86)

Tutorial on "Integrated Networks for Diverse Applications," at *Infocom 87.*

Program committee for *ISS 87*, *ICC 87*, *Midwest Workshop on Communications Systems.* Guest editor for special issue of *IEEE Journal on Selected Areas in Communications*

Course on switching systems (CS 577).

Figure 1.1: Publications and Related Activities

---

architectures, in particular the Starlite architecture based on Batcher's bitonic sorting networks which is being developed at Bell Communications Research, and the Prelude system which is based on an extension of the classical time-slot interchanger design and is under development at CNET in France. Each architecture has its respective advantages and disadvantages and none clearly dominates the others. Prelude and Starlite have two properties not shared by our architecture; they preserve packet sequence and they allow a single shared

"Performance of a Broadcast Packet Switch," by Richard Bubenik and Jonathan Turner, WUCS-86-10.

"Thesis Proposal: Routing of Multipoint Connections," by Bernard Waxman, WUCS-87-2.

"An Architecture for Connection Management in a Broadcast Packet Network," by Kurt Haserodt and Jonathan Turner, WUCS-87-3.

"System Testing of a Broadcast Packet Switch," by Shabbir Khakoo and Jonathan Turner, WUCS-87-4.

"Specification of Integrated Circuits for a Broadcast Packet Network," by Jonathan Turner, WUCS-87-5.

"The Challenge of Multipoint Communication," by Jonathan Turner, WUCS-87-6.

"Design of a Broadcast Translation Chip," by George Robbert, WUCS-87-9.

Figure 1.2: Technical Reports

---

buffer. The ability to preserve packet sequencing is potentially important for certain high speed applications. Shared buffering can provide substantially lower packet loss rates in the presence of highly bursty traffic. Both of these issues will be explored in detail in the coming year, first to assess their real importance, and second to identify extensions of the basic switch architecture that can address them.

Work on a laboratory prototype of our switching system was started about one year ago, when we began design work on two integrated circuit chips. The first of the two chips is the packet switch element that makes up the copy, distribution and routing networks. This is a multi-function switch element that can be configured for any of the three networks, with two input and output ports per switch element. The second chip is the broadcast translation circuit which performs the translation for multi-point packets. It contains two random access memories implementing a pair of lookup tables controlling the translation process, plus associated control circuitry. The chips are being designed in a scalable CMOS process with two layers of metal.

These preliminary designs have just recently been completed and are ready for fabrication. We will use these preliminary designs to guide us in the specification and design of the next set of chips that we will incorporate in our laboratory prototype. We also plan to design a two chip implementation of the packet processor and a datagram router. These are being tackled in a broader context. We have found that several of the chips we need contain similar parts which

are profitably viewed as special cases of a more general *synchronized streams processor*. We have begun design of a special-purpose silicon compiler that will take as input a specification of a streams processor and produce a description of a circuit implementing that specification. We expect this approach to greatly reduce the effort required for the design of several of the chips we require. It will also provide a powerful tool for the design of other similar chips.

Performance of a Broadcast Packet Switch

During the past year we have undertaken a series of studies aimed at developing a detailed understanding of the performance of the broadcast packet switch fabric. The most novel element of this work has been the evaluation of the copy network and the effect on performance of the contention caused by extensive packet replication. The performance evaluation includes both analysis and simulation results, characterizing the dependence of copy network performance on fanout and the location of the active sources. We can place a theoretical bound on the performance degradation possible in the copy network in the worst-case and our simulation results show its behavior in a wide variety of situations both typical and atypical. Summarizing the results briefly, we found that throughput depends strongly on both fanout and the number of active sources, with the best performance obtained with fanouts equal to powers of two and many active sources. The worst performance was obtained for fanouts between powers of two and a small number of active sources grouped on adjacent input ports. While the worst-case analysis shows that the factor of two speed advantage provided for the copy network is insufficient to handle the worst-case traffic pattern, our simulation results convince us that it is adequate to handle all but the most contrived of situations. Furthermore, we have determined that the addition of two distribution stages in front of the copy network is sufficient to eliminate the possibility for overload even in the worst-case.

The performance evaluation work also included a study of architectural alternatives in the design of the routing and distribution networks. The issues examined included, the effect of cut-through operation in the switch nodes on delay and throughput, the effect of the distribution network, the effect of node size and the effect of a modified queueing discipline. Contrary to expectations, our initial results on node sizes showed that networks composed of small nodes performed better than networks composed of large nodes. This turned out to be caused by a subtle effect of the FIFO queueing discipline used by the nodes. The alternative queueing discipline we considered improved performance considerably and gave a further advantage to large nodes over small ones.

Performance evaluation of packet switching fabrics such as the ones used in broadcast packet switches suffers from the lack of a unified theory or even

appropriate terminology. Terms like *non-blocking network* which are meaningful in discussions of circuit switching fabrics cannot be applied to packet switching fabrics in any technically precise way. This makes it difficult to compare or classify competing architectures. We feel that our work to date puts us in a good position to begin to correct this situation and we hope to do that in the next year.

Testing of Broadcast Packet Switch Fabrics

The ability to adequately test a switching network is vital to its successful application in a practical communications system. The testing facilities must allow complete coverage of the entire fabric and precise location of failing elements.

While other researchers have developed testing methods for binary routing networks, there has been little work on cascaded networks of the sort used in our architecture. We have extended earlier work on single networks to cascaded networks, showing how a cascade of m binary routing networks can be tested using $(m+1)n$ test packets, where n is the number of input ports to the network. Such a test sequence is sufficient to detect and identify any single link or single node fault. We also have shown how to perform the testing when the processor performing the test has access to only one pair of ports on the network.

Connection Management

Connection management refers to the collection of algorithms used to create and maintain multi-point connections in a broadcast packet network. A multi-point connection is intended to be a flexible mechanism that can support a wide variety of different applications. To achieve this flexibility, it must be possible to configure a multi-point connection for different uses. One of the first challenges in creating a useful and practical connection management system is deciding exactly what set of primitive capabilities the network should provide to enable users to configure connections. The subsequent challenge is designing the mechanisms needed to implement these capabilities.

We have identified a method of configuring connections based on the concepts of sub-channels within a connection and permissions. Sub-channels allow a connection to be broken down into several distinct information flows, which can be configured differently but because of their close relationships are controlled by the network in a unified way. Permissions give the user a mechanism for controlling access to sub-channels and assist the network in managing its resources (primarily trunk bandwidth).

Based on these ideas, we have developed a specification of a simple connection management architecture and a series of scenarios showing how it can be used to support a variety of applications including broadcast video and multi-media conferencing. The connection management architecture has been designed at several levels of abstraction, with explicit interfaces at each level. The primary abstraction level, from the user's perspective, is the one that defines the interface between the network and a *user agent*. At this level, the network is viewed as a single entity which modifies connections in response to control messages. This level is currently the most well-defined. The next level of abstraction below this defines the interfaces between switching systems in the network and it is at this point that explicit reference must be made to the distributed algorithms and data structures that implement the higher level abstractions. While preliminary work has been done at this level, there is still a great deal that needs to be done. We have also begun looking at a higher level of abstraction corresponding to the interface between user agents. At this level application-dependent issues appear. User agents cooperatively determine how connections should be configured to suit the client applications, and direct the network to configure them via control messages.

Our plans for the coming year include the development of a complete specification of the connection management architecture and a preliminary implementation, in the form of a software simulation. We intend to use this simulation to obtain a better understanding of the issues involved in the design and use of a multi-point connection management system.

Routing

The objective of the routing problem is to determine a set of network resources (primarily trunk bandwidth) sufficient to support communication among a specified set of users. In conventional circuit switched networks, all connections require the same amount of bandwidth and (almost all) have exactly two endpoints. Such a network can be described formally as a graph in which each edge has both a capacity and a length. A set of connections for such a network is simply a collection of vertex pairs. A feasible route assignment is an assignment of each connection to a path joining the connection's endpoints that doesn't exceed the capacity of any edge. An optimum routing algorithm is one that can find a feasible assignment whenever one exists.

Of course, this version of the problem is a static one. In a real communications network, the set of connections changes with time and the network must implement a routing policy that manages the changing set of connections in a way that makes it unlikely that a new connection will be blocked. In the interests of efficiency, it is generally assumed that once a connection has been assigned

a route, that assignment will remain fixed as long as the connection is present. These considerations lead to a routing policy based on the heuristic strategy of routing connections by the shortest path available at the time the connection is established.

If connections can have an arbitrary bandwidth associated with them, the routing problem becomes a bit more complicated. One must now consider the network to be a graph in which vertices can be joined by multiple edges. To prevent blocking of connections with large bandwidth requirements, new connections should be assigned to the fullest edges with sufficient capacity along the assigned route. This strategy preserves large blocks of bandwidth for use by high speed connections.

In broadcast networks, a connection can involve an arbitrary number of endpoints. A feasible route assignment for a set of connections is an assignment of each connection to a subtree connecting its endpoints, in a way that does not exceed the capacity of any edge. As in the case of point-to-point networks, connections come and go over time, and so the appropriate routing policy is to assign each connection to the subtree with shortest total length available at the time the connection is established. This can be viewed as a generalization of the Steiner tree problem in graphs. This problem is known to be NP-complete, meaning that there is unlikely to be an efficient algorithm that can always find an optimal solution. On the other hand, there are several efficient algorithms that yield solutions that are close to optimal. The best known one is called the minimum spanning tree heuristic (MST).

Connections in broadcast networks are dynamic in another way. They grow and shrink with time as individual endpoints come and go. The challenge here, is to maintain a good connection topology without doing a great deal of recomputation each time an endpoint is added or dropped. Practical algorithms must be suitable for distributed implementation, with each node making decisions based on local information. The simplest algorithm is a greedy strategy that adds new endpoints by joining them to the connection by the shortest available path and dropping branches of the connection tree when endpoints drop out.

Our research objective is to develop practical and efficient algorithms that can be used in actual multi-point communication networks. To this end, we have been studying the performance of the MST and greedy algorithms from both a worst-case and average case point of view. A prerequisite for our evaluation of the average case performance, has been the development of a simple probability model that can yield data relevant to real networks. We have developed such a model and have begun using it to evaluate the MST and greedy algorithms. We have shown experimentally, that the average case performance of the MST algorithm is excellent, usually within 5% of optimum. While this algorithm is probably impractical for application in a real network, our results show that

it can serve as a useful standard of comparison against which other algorithms may be measured. In particular, we have used it to study the performance of the greedy algorithm in dynamically changing connections. Our results show that the solutions produced are generally within 20% of the value obtained for the MST algorithm. The performance deteriorates during long sequences of deletions, because the algorithm simply prunes rather than re-routing during such sequences. This sort of degradation is not unique to the greedy algorithm, but is intrinsic to any algorithm that makes only incremental changes and does not re-route.

Our research plans include continued experimental evaluation of these algorithms and others. We also hope to attack the average case performance of these algorithms analytically, in order to obtain greater insight into the factors limiting their performance. We also plan to design and implement distributed versions of these algorithms.

Congestion Control

A principal advantage of packet switched networks is their ability to dynamically allocate bandwidth to the users who need it at a particular instant. Since networks are subject to rapid statistical variations in demand, care must be taken to ensure acceptable performance under conditions of peak loading. Congestion control refers to the collection of methods used to ensure each user acceptable performance under a variety of load conditions. The high speed and multi-point connection capability of broadcast packet networks place new demands on congestion control methods.

A prerequisite to the development of an effective congestion control method is an understanding of the impact that bursty sources have on queueing in the network. The popular M/M/1 queueing model, while theoretically tractable and widely applicable, is insufficient to model the behavior of a small number of high speed, but very bursty sources. A key part of our work in congestion control has therefore been to obtain an understanding of such sources. We are focussing on a simple model that treats each source as a two state Markov chain. The source is active in one state and idle in the other. Parameters of the model include average holding times in each state and the rate of packet transmission while active. This model can be used for a wide variety of bursty sources, including coded video. Our results to date indicate that such sources can lead to serious performance degradation if not handled carefully.

The basic congestion control mechanism under consideration involves user specification of several parameters defining peak and average bandwidth requirements, plus a measure of burstiness. The network uses these parameters to calculate an *effective data rate*, which is used for allocating link bandwidth. The

network also ensures that individual users don't exceed their specified rate, using a simple *traffic valve* at the edge of the network. One simple implementation of a traffic valve can be viewed as a *pseudo-buffer* for which the user specifies the peak arrival rate, the serving rate and the buffer size. Whenever the user sends a real packet, the network adds a pseudo-packet to the pseudo-buffer. If this does not cause the pseudo-buffer to overflow, the real packet is immediately accepted by the network. Otherwise it is discarded. (Note that only pseudo-packets go into the pseudo-buffer.) This mechanism is simple enough to be implemented within packet processor chips at the boundary of the network. We hope to demonstrate that it provides sufficient control to handle a wide range of bursty sources.

Administrivia

In the past year, we have grown from a small base to a research team that now includes two faculty members, one full-time staff person, one visiting research associate and eight graduate students. Additional faculty are also being recruited in both the Computer Science and Electrical Engineering departments, so we expect the growth to continue. While our expansion has led to some natural growing pains, the problems are being successfully addressed and the project is now functioning well.

Our efforts to obtain funding have been fairly successful. In addition to the support being received from the National Science Foundation, we have three corporate sponsors participating in our research program. We hope to add at least one additional sponsor in the near future, and are expecting that in the coming year, our support will be divided evenly between NSF and our corporate sponsors.

While the project's funding situation is in fairly good shape at the moment, we anticipate that additional funding will be required if we are to achieve all our major goals. The most likely source of new funding in the short term is through expansion of the Consortium to six members. This could provide adequate funding through September 1988. After that time, substantial new funding may be required. We are beginning to explore possible sources of that funding including an NSF Engineering Research Center grant.

The project currently supports professors Jonathan S. Turner and Mark Franklin (part-time) plus eight graduate research assistants (see Figure 1.3). We plan to add three additional graduate students in the coming year. In addition, we have one professional staff member (Pierre Costa), one visiting research associate (Riccardo Melen) and expect another visitor in the fall.

For administrative purposes, the ACS Project operates within the Computer and Communications Research Center directed by Professor Mark Franklin. The

Name	Degree (exp. graduation date)	Research Area
Shahid Akhtar	MS (8/87)	congestion control
Victor Griswold	DSc (1/90)	connection management
Mark Hunter	MS (5/88)	connection management
Shabbir Khakoo	MS (5/88)	switch architecture
George Robbert	MS (1/88)	switch architecture
James Sterbenz	DSc (1/90)	switch architecture
Bernard Waxman	DSc (1/89)	routing
David Wexelblat	MS (5/88)	connection management

Figure 1.3: Graduate Student Staff

center has a central office suite housing professors Franklin and Turner, plus eight graduate students, on the third floor of Bryan Hall, across from our main laboratory facility. This laboratory houses our VAX 750, and a cluster of terminals for graduate student use and also serves as an informal meeting room. We have a second laboratory on the fifth floor of Bryan which will be devoted to our hardware prototyping efforts. We also have several graduate students located in offices on the fifth floor adjacent to the laboratory.

The center's base of equipment includes the VAX 750 mentioned earlier, a collection of terminals in laboratories and offices, a MicroVax II/GPX for VLSI design work plus a second design station based on an AED color graphics terminal. We also have a Tektronix logic analyzer and IC tester. We plan to purchase additional workstations to support the software development efforts we are undertaking.

The Center's space and facilities are adequate at the moment, but are fragmented and cannot meet our future needs, based on anticipated growth. While the Engineering School is planning a new building which will relieve the expected space shortage, interim steps will have to be taken to ensure adequate space over the next couple of years. We are currently working with the administration to address this problem.

2. Switch Architecture Studies

Principal Investigator	Jonathan Turner
Research Associate	Riccardo Melen
Graduate Students	Shabbir Khakoo
	George Robbert
	James Sterbenz

2.1. Comparison of Alternative Switch Fabrics

The switch fabric for the Broadcast Packet Network (BPN), described in [49] is based on buffered binary routing networks. It is topologically simple and well-suited to VLSI implementation. The overall structure is shown in Figure 2.1. The system consists of a set of *Packet Processors* which interface to the external links and provide all per-packet protocol processing, a *Connection Processor* which sets up and maintains multipoint connections, and a switch fabric consisting of a *Copy Network*, a set of *Broadcast and Group Translators*, a *Distribution Network* and a *Routing Network.*

Packets enter one of the Packet Processors at left, where an address translation is performed. For point-to-point packets this yields an outgoing link number and an outgoing channel number. These are placed in the header of the packet, which then passes through the CN, one of the BGTs and the DN, following some arbitrary path. When the packet reaches the RN, it is routed using the outgoing link number. The RN is a conventional binary routing network with sufficient storage at each node to store a small number of complete packets. When the packet reaches the outgoing PP, the extra header information added at the incoming PP is stripped off and the packet is transmitted on the outgoing link. The role of the DN is to randomly distribute packets it receives across its outputs. This prevents congestion that can otherwise occur in the RN when subjected to traffic patterns with strong "communities-of-interest."

When a packet belonging to a multipoint connection is received at an incoming PP, it undergoes a similar translation process, but the information added to the

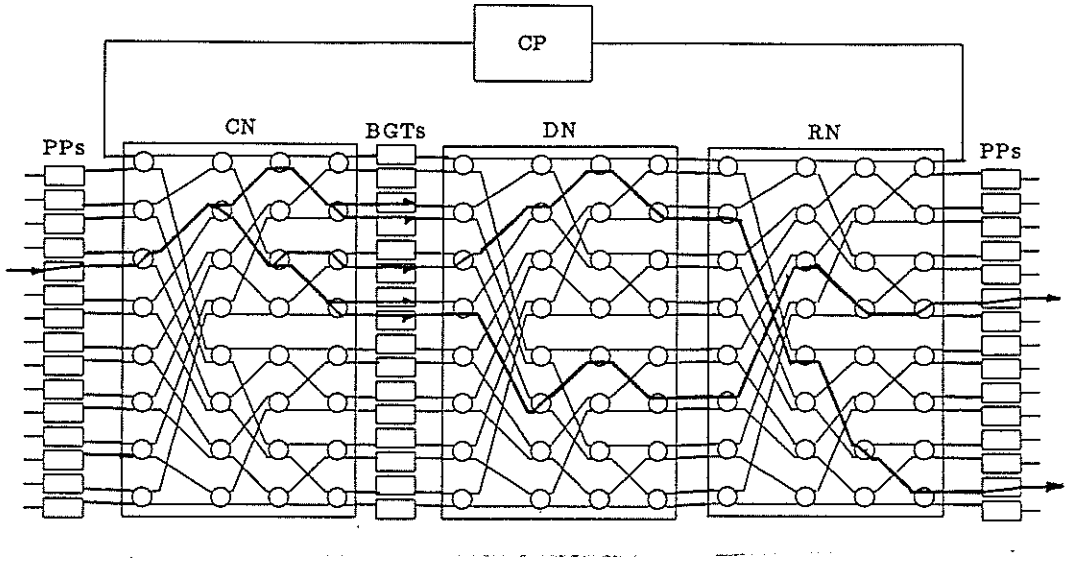

Figure 2.1: Broadcast Packet Switch Fabric

packet header is different. It consists of two fields, a *Fanout* field which specifies the number of outgoing links which are to receive copies of the packet, and a *Broadcast Channel Number*, used by the BGTs. The CN replicates multipoint packets using the fanout field to guide its decisions. At each switch element where replication is performed, the fanout fields of the two copies are modified (essentially by halving the original fanout), so that a short time after the original packet enters the CN, the appropriate number of copies appears at its outputs. The BGTs then perform a translation similar to that done in the PPs, using the broadcast channel number in the copies to index a table, yielding a set of outgoing link and logical channel numbers. These are added to the packet header and used to guide the copies to the proper outgoing links.

This design is well-suited to implementation in a medium speed, high density technology like CMOS. While the per node buffering makes the individual switch elements moderately complex, the topological complexity is very low. The only large memories are in the PPs and BGTs, and these need be accessed only once per packet cycle, permitting the use of high density memories with relatively long cycle times.

In the last five or six years, several experimental switching system designs have been proposed that can support multirate and multipoint communication in a flexible fashion. In addition to the BPN fabric being designed at Washington University, there is the Starlite system originally developed by Alan Huang and Scott Knauer at AT&T Bell Labs and currently being developed further by a group at Bell Communications Research, and the switching matrix for the Prelude experimental wide band switching system, developed by Coudreuse et. al.

at CNET in France. Starlite and Prelude are described below.

Starlite

Starlite is the name given to an experimental switching system developed by Alan Huang and Scott Knauer at AT&T Bell Labs [21,22,23]. The Starlite architecture was motivated by the observation that sorting networks, can be used to construct rearrangably non-blocking switching fabrics with distributed control. This observation was first put forward by Batcher [2] in 1968 in his seminal paper describing his *bitonic sorter* that sorts a set of n numbers using a network of approximately $(n/4)(\log n)^2$ simple comparison elements. For circuit switching applications, this observation leads to switching networks that are non-blocking, operationally very simple and eminently suited to VLSI implementation. To accommodate packet switching, mechanisms are needed to resolve contention between packets that arrive concurrently and are destined for the same output port. Multipoint communication requires additional mechanisms for packet replication. Huang and Knauer's contribution was the development of inexpensive VLSI implementations of Batcher's sorting network and the invention of a variety of supplementary networks which support packet switching and multipoint communication when used in concert with the sorting network.

While Huang and Knauer made no serious attempt to develop complete systems, they did develop a variety of useful tools that can be used for the construction of such systems and suggested ways in which they could be used. Figure 2.2 shows one possible implementation of a packet switch supporting multipoint communication. Packets arriving on external links enter a set of *Packet Processors* at left, which perform some address translation. For point-to-point packets this results in a destination PP number being placed in the packet header. This is used to guide the packet to the appropriate outgoing link.

For the moment, we will ignore the initial sort and copy networks at the top left and concentrate on what happens to packets when they enter the main sorting network at the middle of the figure. This network sorts packets in increasing order of their destination addresses, meaning that when the packets exit the sorting network, all packets with same destination address occupy a contiguous set of output links. The filters at the exit of the sorting network mark all but one packet destined for a particular address, by comparing the destination addresses of packets on adjacent links; if a packet has the same destination address as the packet on the next lower link, its *wait bit* is set. Packets for which the wait bit is zero are forwarded to the routing network at right which routes them to the proper outgoing links. Packets with the wait bit set are sent to one of a set of delay elements, which delays them for approximately one packet time, after which they are recirculated through the sorting network. It's useful to extend this

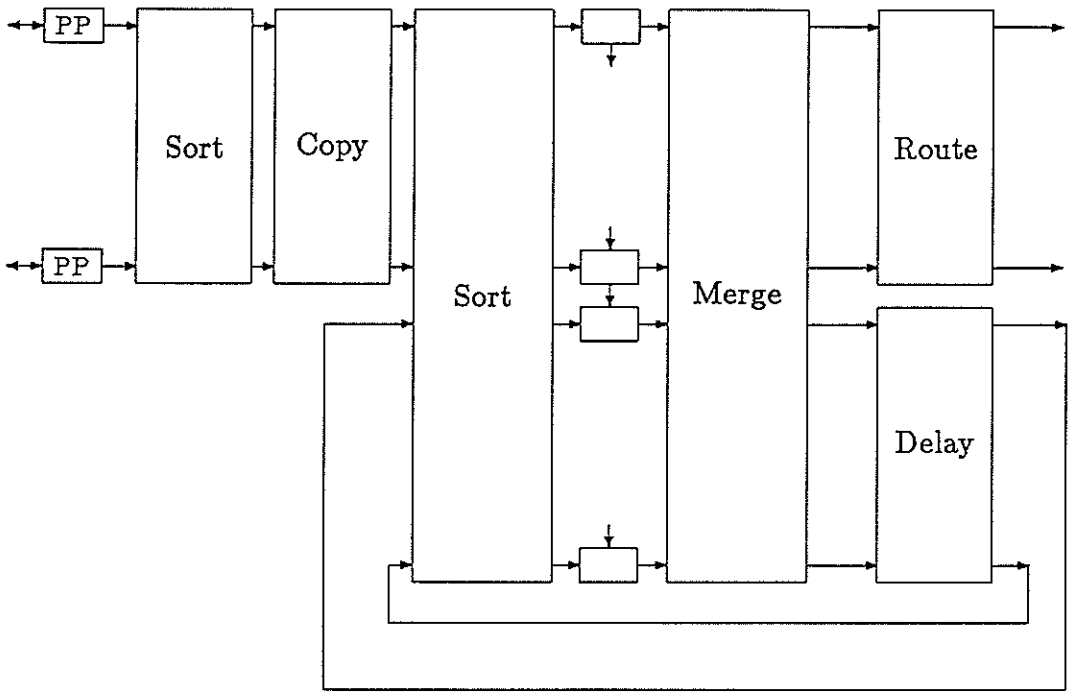

Figure 2.2: Starlite Switch Fabric

basic scheme by adding a second field to each packet which records the number of times a packet has recirculated. By having the main sorting network use this field as a secondary sort key, we can also order packets by their age, giving older packets priority over newer ones. This ensures that packets are transmitted in the same order in which they were received. If the network supports n external links and the main sort and merge networks have m input and output ports, up to $m - n$ packets may be recirculating at any time. Packets may be lost if during a cycle, more than $m - n$ packets have their delay bits set. The value of m is selected based on statistical considerations, to yield an acceptably low probability of packet loss.

We now turn to the issue of packet replication. The network is designed around the notion of a coordinated copy between source and destinations. That is, the source and the destinations must synchronize when a packet is to be copied. When the source PP sends a packet into the network, the destination PPs simultaneously send *blank packets* containing their address plus the address of the source in the headers. The initial sorting network sorts these packets on the source addresses, which places the original packet and associated blank packets on a contiguous set of links, upon exit from the sorting network. The copy network then copies the information from the source packets to each of

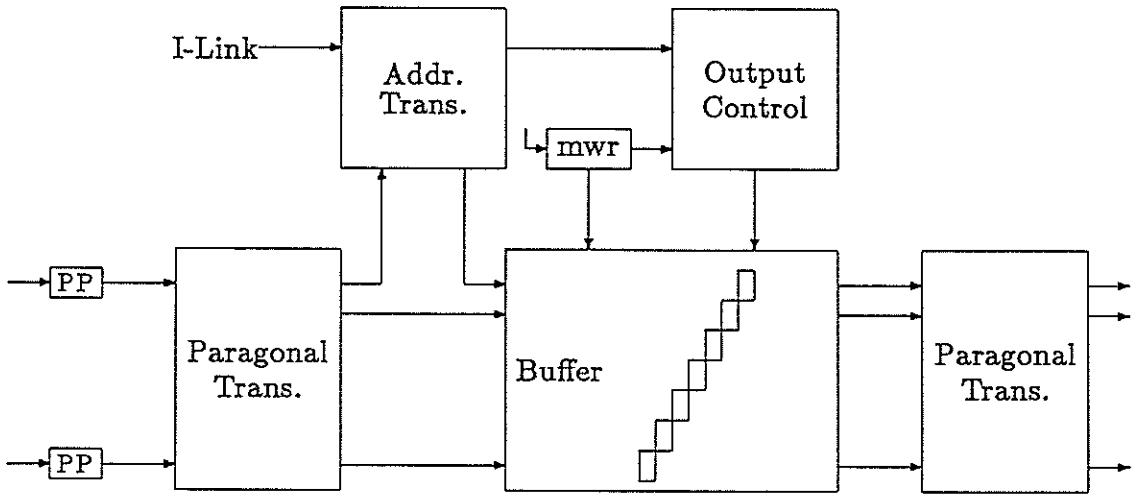

Figure 2.3: Prelude Switch Fabric

the blank packets, a relatively straightforward process, given the sorted arrangement. When these packets enter the main sorting network, they are routed using destination addresses in the same way as point-to-point packets.

The Starlite system has some very attractive properties. The basic switch elements making up the various networks are simple and have a regular interconnection pattern, which makes the design of high speed VLSI implementations quite straightforward. The network is non-blocking and has a latency of only one bit time per stage of switching. It maintains packet sequencing, so that packets are received in the same order in which they are transmitted. The sharing of buffering across the switch fabric, rather than dedicated it to individual links provides more predictable performance in the face of statistical fluctuations in traffic.

The synchronization required for copying is a drawback of this approach, when used in a general packet switching environment. Some form of arbitration is required at the front end to schedule packets that must be copied, and while such a mechanism is probably feasible, no detailed proposal has been put forward. Also, while the switch elements are very simple, their interconnection is topologically complex relative to competing proposals. Finally, the dimensioning of the main sort and merge networks is problematical; it appears likely that to achieve satisfactory performance in a general packet switching environment, these networks must have at least four to eight times as many inputs as there are external links. Nevertheless, the Starlite approach is a very promising one, and is a convincing demonstration of the power of a few simple ideas.

Prelude

The Prelude project began at CNET in France in the early eighties, with the objective of creating a flexible switching system that could provide point-to-point and multipoint communication at speeds up to a few hundred megabits per second [6,10,17]. It is based on a particularly simple form of fast packet switching referred to as *asynchronous time division multiplexing*, and uses a novel high speed switch fabric.

The basic structure of the Prelude switch fabric is shown in Figure 2.3. Packets enter at the transmission interfaces at left, which perform framing and synchronization functions. The packets are then passed through a rotative switch which transforms each packet to a so-called parallel-diagonal format (*paragonal*) in which each packet is distributed across the outputs of the rotative switch, with the first byte of each packet on the first output, the second byte on the second output, and so forth. This transformation places the headers of all packets on the first output of the switch, where they can all be processed by a centralized address translator. The address translator modifies the channel number in the header of the packet and then the modified packet is stored in a central buffer memory, still in the paragonal format. At the same time, the address translator passes to the output control circuit, a bit vector defining which outputs are to receive copies of the packet. The output circuit stores the address at which the packet header was written, in queues associated with the selected outputs. This information is used later to retrieve the packet from the central buffer. There is an output process that examines these queues in a cyclic fashion, initiating a new packet retrieval on each clock cycle. Broadcast is accomplished simply by reading the packet from the buffer once for each output that requires a copy. Note that these reads need not all take place during one packet cycle. Finally, a second rotative switch transforms the packets from the paragonal format back to the normal format so that they can be output on their respective links.

This design has several attractive features. The basic elements are simple; the rotative switches can be implemented as barrel shifters, requiring about $n \log n$ gates, the address translator and buffer are essentially just random access memories with a modest amount of control circuitry, and the output control consists of a fairly simple and regular collection of queues and address registers. As with Starlite it maintains packet sequencing and provides a single shared buffer rather than per line buffers. It is, on the whole simpler than the Starlite fabric and handles multipoint communication in a more satisfactory way.

The main drawback of this approach is its dependence on high speed memories, particularly in the central buffer. It must be possible to access this memory twice per clock time, once for reading and once for writing. There does not appear to be any architectural way to reduce the required memory cycle time for

individual memory chips since the memory read-out process can access the memory chips in random order. Another drawback is that since channel translation takes place before packets are replicated, all the downstream copies of a multipoint packet carry the same channel number. This places operational restrictions on the assignment of channel numbers, that may be problematical, depending on the number of channels and multipoint connections. It is most troubling for general multipoint connections in which there are several transmitters. It appears that either all the links involved in such a connection must use the same channel number, or there must be a different channel number for every incoming port that can be the source of the packet. The latter solution requires that the downstream switches treat all those channel numbers similarly.

Remarks

One major difference between the BPN approach and Starlite and Prelude, is the use of buffering within the switch fabric. While this leads to more complex switch elements, the added complexity has little impact on cost or performance. The reason is simply that cost is determined primarily by component count, which because of pin limitations is determined primarily by topological complexity, not circuit complexity. While simple switch elements are better suited to low density technologies such as ECL, they have little advantage in the context of a high density technology like CMOS. Furthermore, the packet processors required in all three systems, have inherently high circuit complexities and must be implemented in high density technologies to be economical.

Starlite and Prelude do have two potentially significant advantages over the BPN approach. First, they both implement shared buffering, leading to more predictable performance in the presence of highly bursty traffic. While the addition of some shared buffering to the BPN fabric is feasible, no detailed study of such an arrangement has yet been made.

Starlite and Prelude also guarantee that packets are transmitted in the same order as they are received. While it is possible to modify the BPN design to provide a similar guarantee, the required changes may impose operational constraints and degrade performance.

This brief review leads to three conclusions. First all three of the switch fabrics reviewed are viable architectures; they can all be used to support high speed packet networks and multipoint communication in an effective way. Second, none clearly dominates the others; each has a different set of advantages and drawbacks. Third and perhaps most important, each offers some useful lessons to architects of future systems. In the next section, we consider some of the ways in which these lessons might be applied.

2.2. Refinements to the BPN Switch Fabric

We now briefly consider several key architectural issues that will be addressed in the coming year. The first is the problem of resequencing packets; second is the issue of shared buffering; third, a potential application of partial sorting to the BPN fabric; and fourth, some issues that arise in the design of very large systems.

Resequencing

The design of the BPN switch fabric allows packets belonging to a particular logical channel to exit the fabric in a different order from the order in which they are received. We briefly consider several alternative methods for dealing with this problem.

The first method is not to allow packets to get out of order in the first place. This is the approach taken by Starlite and Prelude and can be adopted with the BPN fabric. The key is to distribute traffic in the CN and DN on a per channel basis rather than a per packet basis. This complicates the hardware slightly and the control software substantially. It also degrades the performance of the system under heavy loads. This approach is being taken by a group at CSELT in Italy in their fast packet switching research project.

A second method is to allow resequencing to be done on an end-to-end and application-dependent basis. Preliminary performance data indicate that the probability of packets arriving out of order is extremely unlikely for applications with peak data rates of less than 20% of the FOL data rates. The reason is simply that the time between arrivals of successive packets at a switch is larger than the time it takes a packet to propagate through the switch fabric. Only applications with very high peak data rates are expected to experience a significant rate of packet misordering. This implies that the majority of applications can ignore the resequencing problem and suggests that the provision of a general mechanism for handling it may be unwarranted. Applications such as video or file transfer can handle the resequencing problem in a simple and straightforward way (for video, for example one merely places received packets in the proper position in the frame buffer based on a sequence number).

Resequencing can also be done on an end-to-end, application-independent basis. The most general solution requires buffering and a retransmission protocol. A simpler method is possible if one is willing to tolerate a small but non-zero probability of misordered packets (e.g 10^{-6}). To provide such a guarantee, one adds sequence numbers to packets as they enter the network and uses the sequence numbers to resequence packets on exit. The resequencing device buffers packets as necessary, but never holds a packet longer than a specified time. The

amount of buffering required and the timout value depend on the misordering probability of the network and on the target residual misordering probability. Resequencing can also be done on a per switch basis. The method is similar to the end-to-end, application-independent method but is somewhat simpler to implement.

Detailed evaluation of these options has not yet been attempted. The circuit complexity and performance of end-to-end and per-switch resequencing in particular, requires closer study.

Shared Buffering

While the FOL switch fabric contains buffers, which can in some sense be viewed as shared, most of the buffering actually occurs in the outgoing packet processors. For traffic with Poisson arrival statistics, per-link buffers with 64 buffer slots are sufficient to achieve packet loss rates (due to buffer overflow) of well under 10^{-6} at link occupancies of 80%. Unfortunately, many real applications have arrival statistics that are highly non-Poisson. For these bursty applications, larger buffers or lower link occupancies are required for satisfactory packet loss rates.

One way to achieve the effect of larger buffers is to provide the buffering on a shared basis as done in Prelude and Starlite. For the BPN fabric, a set of shared buffers can be placed between the distribution and routing networks, with a hardware control mechanism used to control the flow of packets from the shared buffer. A simple way of implementing this control is to provide a simple TDM control ring, that carries one bit of flow control information from each PP to each of the shared buffers.

While shared buffering appears to be worthwhile, the situation is really less clear than it might seem. One of the advantages of per-link buffering is that it puts an upper bound on the queueing delay at each switch. In a high speed packet switching system, it appears desirable to have a maximum per-switch delay in the neighborhood of a few milliseconds. This means that the amount of buffering per link is a few hundred kilobits (assuming 100 Mb/s link speeds), not an unreasonable amount to provide on a per-link basis. A detailed evaluation of the cost and advantages of shared buffering remains to be done.

Partial Sorting

The Starlite system suggests a possible improvement to the BPN fabric design. The routing network used in the Starlite design is a form of binary routing network (specifically, a banyan network) that will pass a sorted sequence of packets

without conflict. It will also pass without conflict, a set of input packets that arrive on an input port with the same port number as their destination.

Suppose we replace the distribution network in the BPN fabric with a network that routes packets whenever it can do so without delaying them, and distributes packets otherwise. Intuitively, such a network appears likely to present the routing network with a better traffic pattern than is obtained by simply randomizing the traffic; this could lead to substantially higher throughputs. We plan to examine such a strategy in detail in the next few months and quantify the resulting performance advantage.

Design of Large Systems

The BPN switch module has been designed as a component that can be used to construct the large switching systems required for supporting ubiquitous public networks. We have formulated a possible design for a local switching system, in order to obtain a better idea of the scale of the system and the associated control issues.

The proposed design supports up to 65,000 access lines and 4096 trunks, and is based on a hypothetical traffic mix[1] including 200 entertainment video sources, video telephones with an effective bandwidth of 8 Mb/s and busy 20% of the time, plus other traffic that is equivalent to the video telephone traffic in total resource requirements. For the broadcast video, it is assumed that a 90:10 rule applies; that is, in any group of users, 90% will be accessing 10% of the available channels. This, along with the assumptions on video phone traffic were used to select concentration ratios.

A block diagram of the system appears in Figure 2.4. The lines terminate at the bottom of the figure, trunks connect at the top. It is constructed from two types of switch modules, one with 64 links and the other with 80, and providing 4:1 concentration. The system has 128 *back-end switch modules* that connect to 4096 trunks and provide access to 16 *major groups*, which terminate 4096 lines each. The major groups each contain 16 back-end switch modules and 8 *minor groups*, terminating 512 lines each. The minor groups each contain 8 back-end switch modules and 16 front-end switch modules.

The hardware complexity of this system can be estimated based on the assumption that one unduplicated switch module can be implemented on 8–12 printed circuit boards. If the front-end switch modules in the minor groups are

[1]It should be noted, that this is purely hypothetical and not supported by any detailed traffic projections. The purpose is just to present a rough picture of a possible traffic mix, and how one might configure a system to support it.

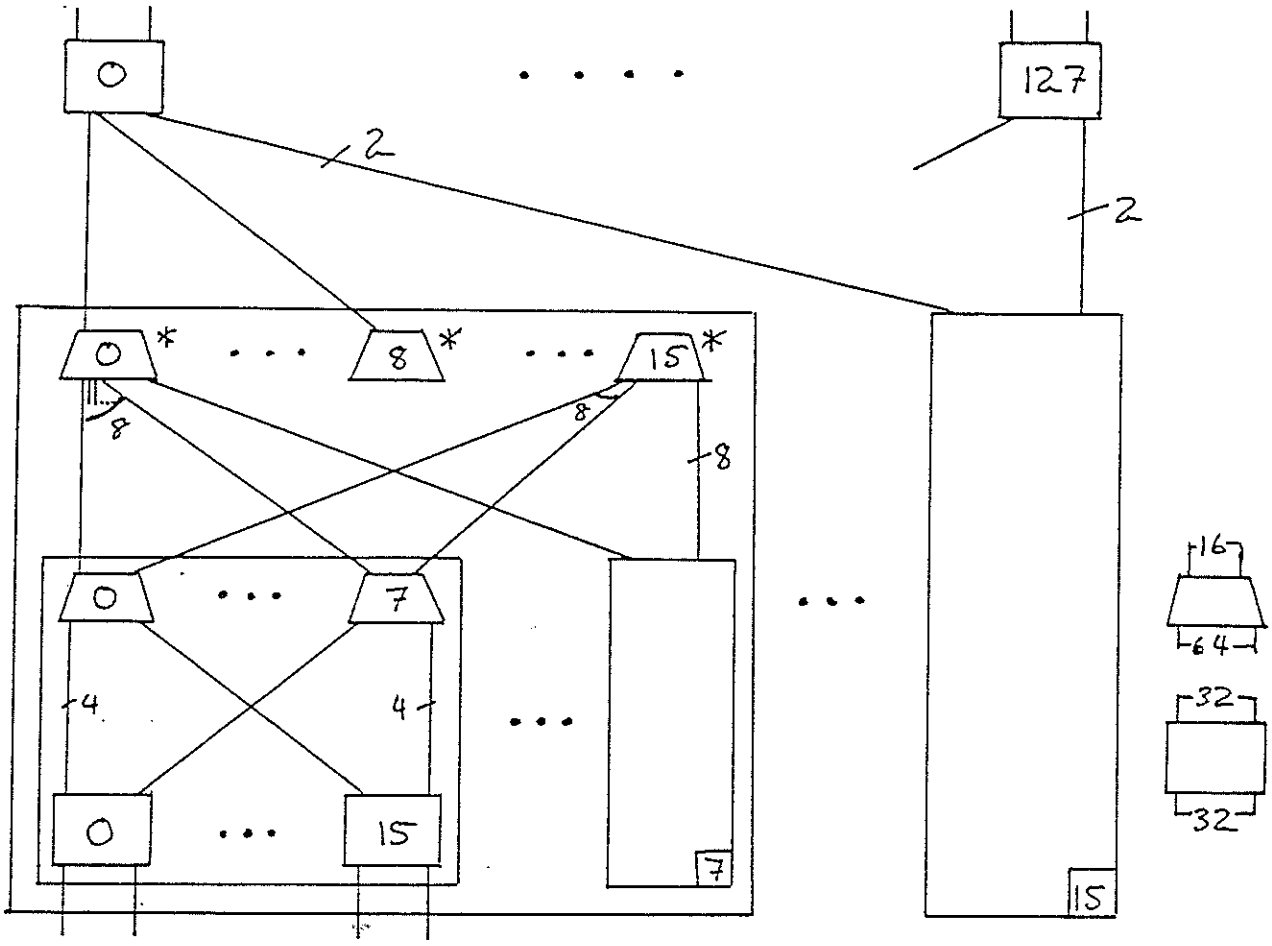

Figure 2.4: Local Switch Design

unduplicated and everything else is duplicated, a minor group requires two equipment frames, a major group requires 18 frames and the complete switch requires 304 frames. This is roughly comparable in complexity to a telephone central office supporting the same number of lines.

This design raises several questions. First, one might ask, why structure the system in this way in the first place? Why not instead, simply scale the switch module to a much larger size? This turns out to be impractical for a variety of reasons, including expandability, timing, reliability and control partitioning. A large system of this sort must be broken down into a large number of fairly small, independent modules. This raises an interesting point with respect to the Starlite and Prelude switch fabrics. Those fabrics, like the BPN fabric, are useful mainly for constructing switch modules of moderate size; to build very large systems they must be configured together into a larger structure. In that context, many of the apparent distinctions among the different approaches become less significant.

Another question that arises is how connection management is handled in a large system. The simplest method conceptually is to allow the connection processors in each switch module to operate independently. While this can be workable, it requires more effort and more delay in the connection establishment process than is strictly necessary. It is possible to speed up the process considerably by using datagram routing within the switch, limiting the logical channel-based routing to the external lines and trunks. The regularity of the interconnection topology makes the implementation of datagram routing straight-forward for point-to-point connections. It remains unclear whether such an approach is practical for multipoint connections as well.

Using datagram routing on the inter-module links eliminates the need for most internal resource allocation, but still leaves the problem of resource allocation on the external links. To manage this, a database is required that tracks the status of the external links. We plan to explore the issue of how such a database should be organized to provide rapid access by the connection processors, that must use the information in the database to make routing decisions.

3. Prototype Hardware Design

Principal Investigator	Jonathan Turner
Research Associate	Pierre Costa
Graduate Students	Shabbir Khakoo
	George Robbert
	James Sterbenz

A prototype of a BPN switch module is being designed. The purpose of this prototyping effort is to provide a convincing demonstration of feasibility, allow detailed examination of implementation issues and provide a testbed for future experimental efforts at higher levels.

The prototype switch module will have an eight port fabric with just a Copy Network and Routing Network, as shown in Figure 3.1. The group translation function will be omitted from the Broadcast and Group Translators; we refer to the resulting elements as Broadcast Translation Circuits (BTC). We're planning to use a a general purpose workstation-class machine (such as a Microvax) as the Connection Processor (CP).

Custom integrated circuits are being designed for the switch elements, BTCs and Packet Processors (PP). The BTC and switch element designs will require one chip apiece, the PP design will require two or three chips, one of which may be a standard memory part. A total of approximately 50 custom chips are required to implement the prototype switch module. In addition, a global timing circuit is required, as well as an interface to the connection processor.

At this time, initial designs have been completed for the BTCs and switch elements. A high level design of the PP has been completed, and detailed circuit designs will be developed during summer 1987. Also, during the summer, samples of the BTC and switch element chips will be fabricated and tested and design revisions will be made. Design of the CP interface and timing circuitry will also be started during the summer.

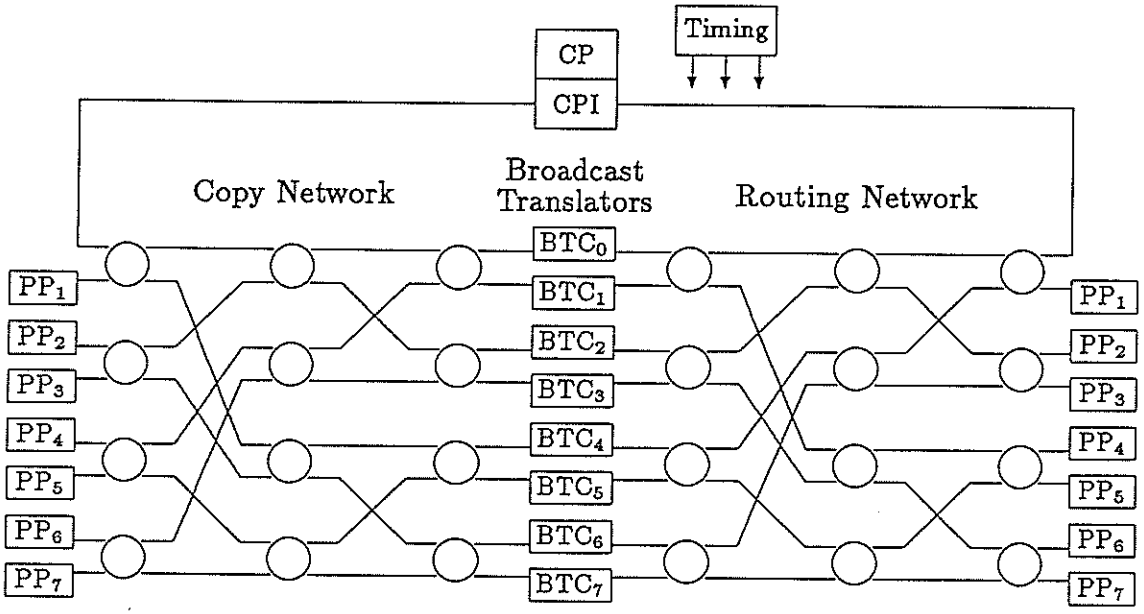

Figure 3.1: Prototype Switch Module

3.1. Packet Formats

This section describes the formats of packets used in the switch. There are two primary packet formats: external and internal. Packets are carried in external format on the fiber optic links connecting switches, and in internal format within each switch. The PP translates between these two formats. Note that higher level processes may define additional packet formats; this section details only those fields that are of direct concern to the prototype hardware.

External Packet Format

Each external packet is organized as a sequence of `wsize` (8) bit wide words. Each packet contains exactly `epsize` (76) words, the first `ephsize` (3) of which constitute the packet *header*. The last `csize` (1) words of the packet are used for a frame checksum. When transmitted on the external transmission links, external packets are separated by a SYNC pattern that allows the receiver to identify packet boundaries. The meanings of the external fields are given below.

- *Packet Type* (PTYP). Identifies one of several types of packets, including ordinary data packet, control packets and test packets.
- *External Logical Channel Number* (ILCN). Logical channel numbers are used to identify which connection a packet belongs to. For th prototype, only 256 distinct logical channels are recognized.

- *Information* (I). Normally contains user information. In the case of control packets, may contain additional control information. Individual words are denoted I[0], I[1], I[2], ... with I[0] being the first word of the I field.
- *Frame Check* (FC). The frame check is used to detect errors in the packet. The choice of code is to be made later.

Internal Packet Format

Each internal packet is organized as a sequence of `wsize` (8) bit wide words. Each packet contains exactly `ipsize` (80) words, the first `iphsize` (5) of which constitute the packet *header*. The structure of the packet is shown in Figure 3.2. Each word has an associated (odd) parity bit. The meanings of the fields are given below.

- *Routing Control* (RC). This field determines how the packet is processed by the switch elements. The possible interpretations are listed below.

  0 *Empty Packet Slot*

  1 *Point-to-Point Data Packet*

  2 *Broadcast Packet*

  4 *Specific-Path Packet*

- *Operation* (OP) This field specifies which of several control operations is to be performed for this packet. The possible values of the field and the corresponding functions are listed below.

  0 *Vanilla Packet.* No control functions.

  1 *Read* LCXT *Block.* Directs PP to read a block of 16 entries from the Logical Channel Translation Table. I[0] specifies which block to read. The data is copied into I[1]–I[64] and the packet is returned to the CP.

  2 *Write* LCXT *Block.* Directs PP to write a block of 16 entries to the Logical Channel Translation Table. I[0] specifies the block to write. The data to be written is in I[1]–I[64].

  3 *Read* PP *Parameter Block.* Read the contents of the PP parameter block into I[1]–I[64] of the packet and return the packet to the CP.

  4 *Write* PP *Parameter Block.* Write the contents of I[1]–I[64] into the PP paramenter block.

  5 *Switch Test Packet.* When received by a PP is returned to the SF with a new routing field. The new routing field is obtained by rotating the entire contents of the packet by five words.

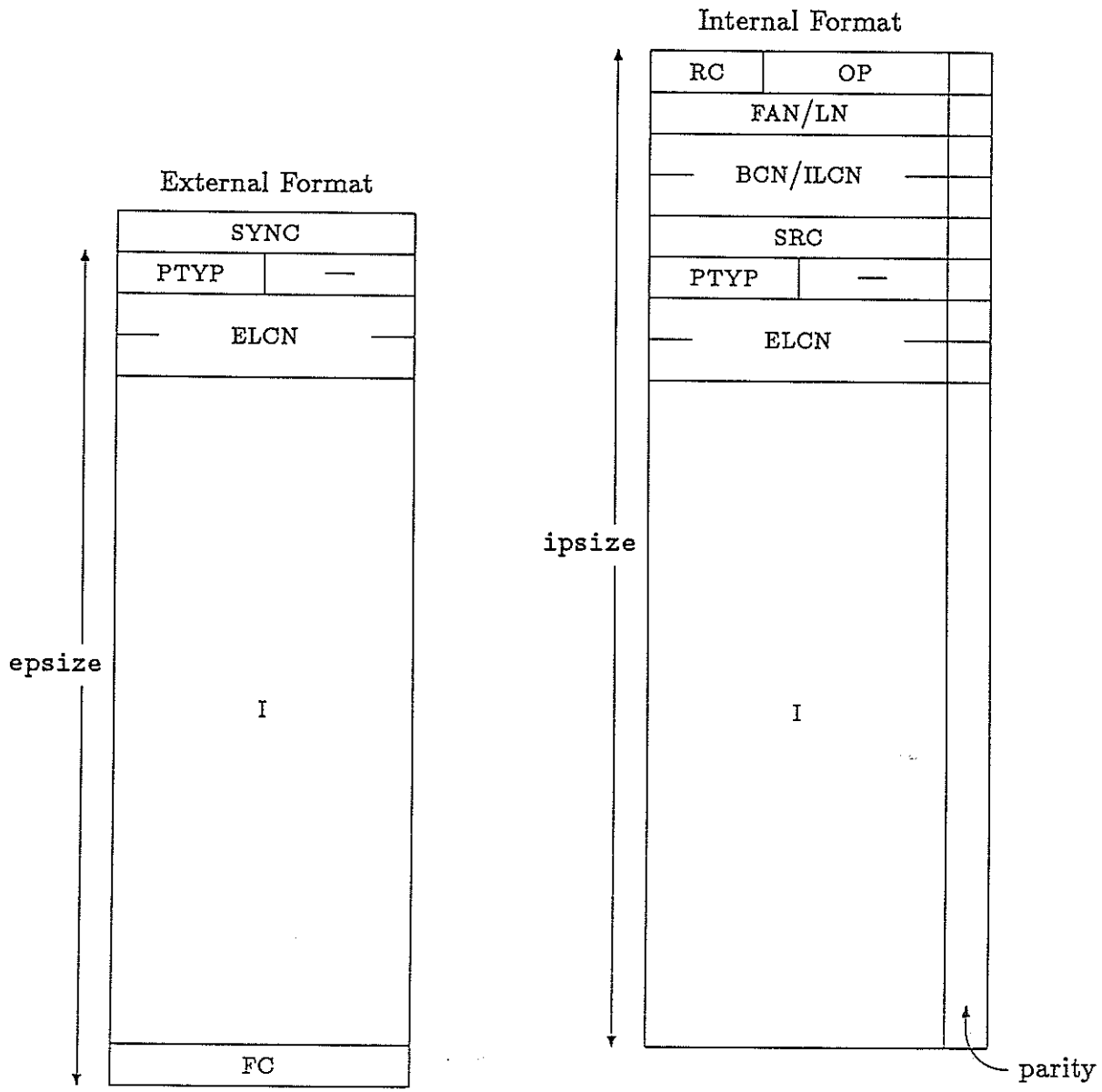

Figure 3.2: Packet Formats

6 *Read* BTT *Block.* Directs BTC to read and return a block of 16 entries from the Broadcast Translation Table (BTT). I[0] field specifies which block to read. The data is copied into I[1]–I[64] and the packet returned to the CP.

7 *Write* BTT *Block.* Directs BTC to write information into a block 16 entries of the BTT. I[0] field specifies which block to write. The data to be written is in I[1]–I[64].

8 BTT *Single Entry Update.* Directs all BTCs in a given group to update an entry in their BTTs. BCN gives the broadcast channel number of the connection, I[0] is the new fanout, I[1] is the block of Broadcast Copy Indices that are to be updated (16 BCIs to a block) and I[2]–I[65] contains the block of 16 entries. Each BTC calculates its broadcast copy index, j, for the connection and if $\lfloor j/16 \rfloor$ equals the block number in I[1], it copies I[4(j mod 16)] – –I[(j mod 16) + 3] to BTT[BCN].

9 *Read* BCIT *Block.* Directs the BTC to read the contents of one block of 64 BCIT entries into I[1]–I[64] and return the packet to the CP. I[0] specifies the block to read.

A *Write* BCIT *Block.* Directs the BTC to write the information in words I[1]–I[64] int one block of the BCIT. I[0] specifies the block to write.

C–FF *Reserved.*

- *Destination* (DST). The interpretation of these three words depend on the value of RC.
  - *Fanout* (FAN). If RC = *Broadcast Packet*, the second word of the packet is taken to be the fanout, that is the number of switch fabric output ports that require copies of the packet.
  - *Broadcast Channel Number* (BCN). If RC = *Broadcast Packet*, the third and fourth words of the packet are taken to be the broadcast channel number. All packets within a particular multi-point channel have the same broadcast channel number. Only 256 distinct BCNs are recognized.
  - *Link Number* (LN). If RC = *Point-to-Point Packet*, the second word of the packet is taken to be the number of the outgoing link to which the packet should be delivered.
  - *Internal Logical Channel Number* (ILCN). If RC = *Point-to-Point Packet*, the third and fourth words of the packet are taken to be the internal logical channel number. This will become the external logical channel number when the packet exits the switch module.
  - *Specific Path Specification.* If RC = *Specific-Path Packet*, the three words of the DST field specify output ports for each of the three networks. The packet will be routed through each of these.

- *Source* (SRC). The number of the most recent PP through which the packet has passed. For vanilla packets, this will be the number of the link on which the packet entered the switch. For test packets it will be changed as the packet passes through different PPs.

3.2. Packet Switch Element

The Packet Switch Element chip (PSE) is the 2×2 VLSI switch element used in the binary routing, copy and distribution switch networks. The PSE directs packets to one or both outputs based on packet type (point-to-point, broadcast, or specific-path), switch operation mode (routing, copy, or distribution), and packet header LN or FAN fields.

Operation

A single PSE circuit is used to implement the routing, copy and distribution networks. PSE routing decisions are based on the operation mode and RC field. In particular, if the operation mode is:

- *Route* — Use appropriate bit of the LN field to select an output port.
- *Copy* — If RC is broadcast, and FAN exceeds one half the number of reachable outputs, send copies of packets to both output ports. If RC is specific-path, use appropriate bit of LN field to select an output port. Otherwise, distribute.
- *Distribute* — If RC is specific-path, use appropriate bit of LN field to select an output port. Otherwise, distribute.

Each PSE operates on the basis of a *packet cycle*, indicating the timing of packet arrival. Packets are either buffered, or if possible cut-through directly to the appropriate output port. Each input is capable of fully buffering a single 80 word packet.

Before each packet cycle, a corresponding *grant cycle* occurs, during which time grants are propagated backward through the fabric, indicating that successive switch element stages will be able to accept packets (either due to an empty buffer slot, or knowledge that a packet will be able to leave an occupied slot during the upcoming packet cycle). When arbitrary routing choices can be made, the following policies are used to make decisions:

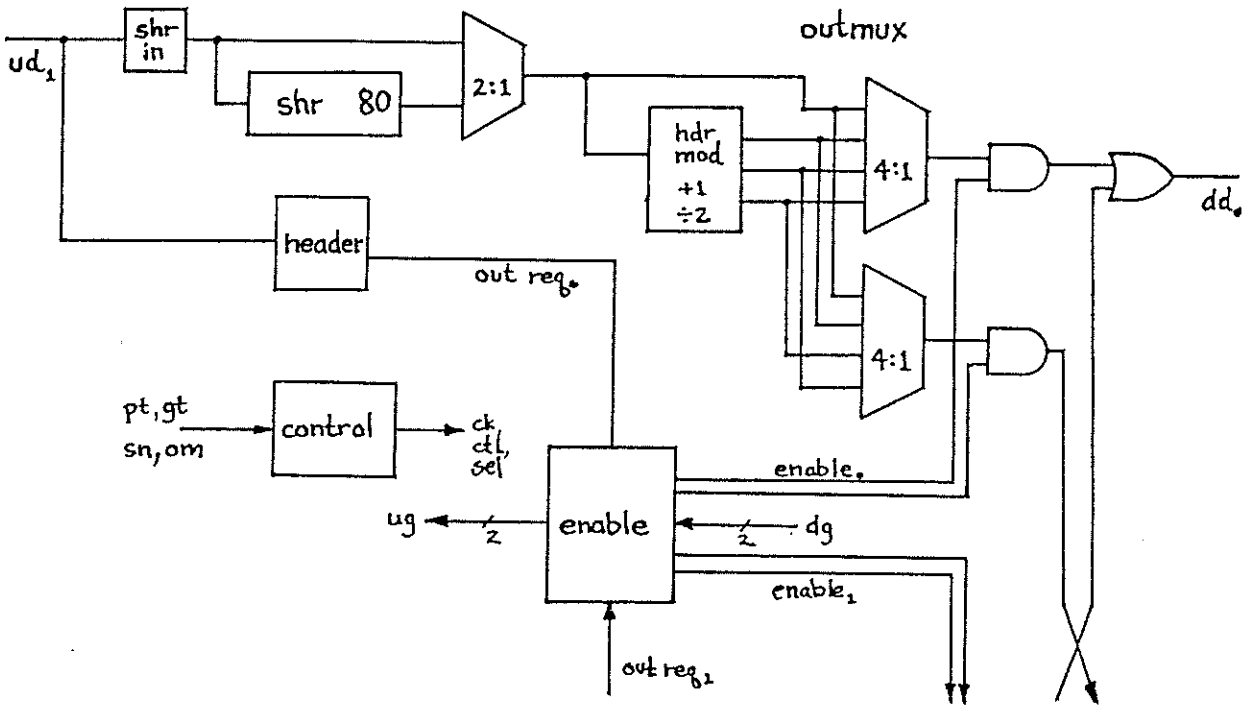

Figure 3.3: Block Diagram of Packet Switch Element Chip

- Ties among input ports for a given output port are arbitrarily broken based on the last input port favored, to avoid individual starvation.
- Packets that can proceed to either output are uniformly and arbitrarily distributed (all packets in distribution network, *point-to-point* packets and *broadcast* packets not replicated in copy network).
- Packets requesting both outputs in the copy network are favored over packets requiring only one.
- Packets requesting a specific output are favored over packets which can use either.

Internal Components

A block diagram of the PSE appears in Figure 3.3.

- *Input Shift Register* — The two input shift registers (one per switch input) are six stage static shift registers with an output tap after the first stage. Packets are shifted into each input shift register from the corresponding upstream data lines. The first stage of shift delay is for synchronization. The

remaining five stages allow sufficient time for control and routing decisions to be made by packet header and output enable decision logic.

- *Packet Header Logic* — Each packet header logic block (one per switch input) receives the packet from the output of the first stage of the corresponding input shift register. The first four words, comprising the packet header, are loaded into four registers. The logic decodes the packet type, and determines the output port(s) requested based on packet type, switch operation mode, and FAN or LN fields. Invalid header information sets error flags, and forces the packet type to *point-to-point* (so that bad packets are not broadcast through the fabric).

- *Packet Buffer and Cut-Through Path* — The packet buffers (one per switch input) are 80 stage static shift registers. Based on the availability of requested outputs, a packet is either shifted into a packet buffer, or cut through directly to an output port. A 2:1 multiplexor selects either buffer or cut-through path to the output routing logic.

- *Output Routing Logic* — Each output routing block (one per output) consists of a PLA to modify the fanout of broadcast packets ($\div 2$ or $\div 2 + 1$), a single stage shift register to delay packet words for specific path packet header manipulation, and two 4:1 multiplexors to select the appropriate path to the outputs (undelayed, delayed, delayed $\div 2$, or delayed $\div 2 + 1$). The output of each multiplexor passes through five 2-input nand gates, which are enabled by the output enable decision logic to route the packet to the appropriate output(s). The outputs from one of the sets of nand gates are merged with the corresponding outputs from the other output routing logic block (only one of which has been enabled) using five 2-input nand gates. These are latched by a register which drives the PSE outputs.

- *Output Enable Decision Logic* — A single PLA-based output decision logic block determines how the output logic routing will enable packets to PSE output ports, based on the requests generated by the packet header logic, and by the downstream grant lines (indicating whether packets may flow through each output). Upstream grant lines are also set, based on whether each input port will be able to accept new packets.

- *Switch Element Control* — Switch element control blocks (one per switch input) are PLA-based finite state machines which operate essentially in lock-step, to control operation of each switch half.

3.3. Broadcast Translation Circuit

The Broadcast Translation Chip (BTC) provides unique addresses for each of the copies of a broadcast packet replicated by the copy network. It also provides a hardware assist for updating the table of new addresses for a single broadcast channel.

Operation

The BTC's operation depends upon the type of packet passing through it.

- *Ordinary Data Packet.* These packets are passed straight through the main shift register unchanged.
- *Broadcast Data Packet.* The routing field is replaced with a new field selected from an internal *Broadcast Translation Table* (BTT). The new field is selected using the BCN of the packet.
- *Read/Write* BTT *Block.* These two packet types are used for updating the BTT in large chunks and for reading it for auditing and testing purposes.
- *Read/Write* BCIT *Block.* These packets read and update the entire broadcast copy index table.
- *Single Entry Update.* This packet supplies a set of new routing fields for this BCN. The BTC chooses which one to write into the BTT depending on the value of the *broadcast copy index*, which is determined from an internal *Broadcast Copy Index Table* (BCIT). I[0] specifies the new fanout for the connection, I[1] specifies which group of routing fields are contained in the packet. I[2]–I[65] contains the routing fields.
- All other packets are passed through like ordinary data packets.

Internal Components

A block diagram of the BTC appears in Figure 3.4.

- *Main Shift Register.* A 16 stage shift register. This gives the BTC sufficient time to perform its internal operations while the packet is still accessible. It also allows for access to several words of the packet in parallel.

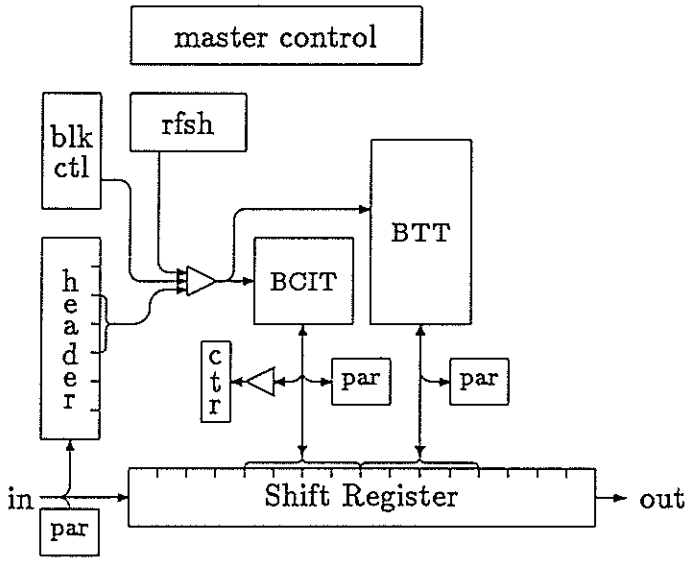

Figure 3.4: Block Diagram of Broadcast Translation Chip

- *Header Shift Register.* A 7 stage shift register. Stores the packet header so that the BTC has access to it during the entire time it is processing the packet.
- *Broadcast Translation Table.* A 256 entry table, with four words per entry. It stores the new packet headers for broadcast packets. It is implemented using a 3-transistor dynamic memory.
- *Broadcast Copy Index Table.* A 128 entry table with one word per entry. This stores the values so that the *bci* may be calculated quickly by table lookup.
- *Refresh Control Block.* This is a finite state machine that controls the refreshing of the two dynamic memories on this chip.
- *Block Transfer Controller.* This control block generates the signals used for block reads and writes of the two on-chip memories.
- *Master Controller.* This control block looks at the packet header, determines what type of packet it is and performs the correct action. In the case of block memory transfers, it "calls" the block transfer controller supply the correct control signals.

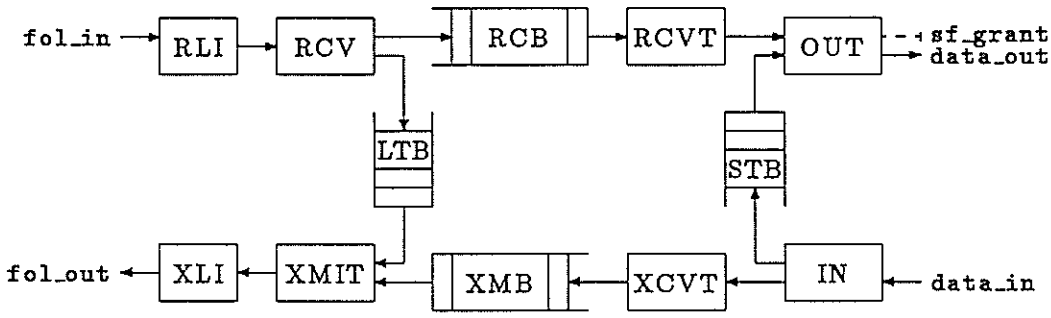

Figure 3.5: Packet Processor Circuit

3.4. Packet Processor

The Packet Processors (PP) form the interface between the external fiber optic links and the switch module's internal data paths. They perform all the link level protocol functions, including the determination of how packets are routed.

Operation

The processing of packets by the PPs is determined by the PTYP field for external packets (received from FOL) and by the OP field for internal packets (received from SF).

- *External Data Packet.* Converted to internal format, with the routing field determined by a lookup in an internal *Logical Channel Translation Table* (LCXT). The packet is then transmitted to the switch fabric.
- *External Link Test Packet.* The PTYP field is changed to external data packet, and the packet is returned on the outgoing FOL.
- *External Control Packet.* Converted to internal format, with the LN field set to 0 and the RC set to ordinary data packet. Transmitted to SF.
- *Internal Data Packet.* Converted to external format, with contents of internal LCN field transferred to external LCN field. Transmitted to FOL.
- *Switch Test Packet.* The RC field is set to 0 and then the first five words of the packet are moved to the end of the packet and everything else shifted up. In other words, the whole packet is rotated by five words. The packet is then returned to the SF.

Internal Structure

A block diagram of the PP appears in Figure 3.5. The various components are described briefly below.

- *Buffers.* The PP contains four packet buffers. The *Receive Buffer* (RCB) is used for packets arriving from the FOL and waiting to pass through the SF. The *Transmit Buffer* (XMB) is used for packets arriving from the SF that are to be sent out on the FOL. The *Link Test Buffer* (LTB) and *Switch Test Buffer* (STB) provide paths for test packets used to verify the operation of the FOL and SF respectively. The RCB has a capacity of 16 packets, the XMB has a capacity of 32. The LTB and STB can each hold two packets. Together, the four buffers require a total of about 38 Kbits of memory.
- *Receive Link Interface* (RLI). Converts the incoming optical signal to an eight bit electrical format, synchronized to the local clock.
- The *Receive Circuit* (RCV). Checks incoming packets for errors, adds parity, strips off CRC, routes test packets to the LTB and other packets to the RCB.
- *Receive Conversion Circuit* (RCVT). Adds five bytes of header information to the front of each packet received from the RCB.
- *Output Circuit* (OUT). Performs logical channel translation, using an internal *Logical Channel Translation Table* (LCXT) and sends packets to the SF. Also reads test packets, LCXT read/write packets and PP parameter block read/write packets from the STBand processes them appropriately.
- *Input Circuit* (IN). Routes internal data packets to the XCVT and others to the LTB. Performs the rotation required for switch test packets.
- *Transmit Conversion Circuit* (XCVT). Removes the first five words of the packet then transfers it to the XMB.
- *Transmit Circuit* (XMIT). Takes packets from the XMB, adds the SYNC field, strips parity and computes the frame check. Also processes test packets from the LTB.
- *Transmit Link Interface* (XLI). Converts from eight bit electrical format to optical format.

4. Performance Studies

Principal Investigator	Jonathan Turner
Graduate Students	Richard Bubenik

The performance of the broadcast packet switching system has been studied both analytically and using simulation. The objectives of these studies are (1) to determine if the proposed architecture performs acceptably under a wide range of loading conditions, (2) to assess the impact of a variety of design options and (3) to provide a set of tools that can be applied to a broader class of switch fabric designs.

While the performance of binary routing networks has been studied by a variety of authors, (see for example [7,8,9,13,16,25,28,36]) there are no published works that provide the kinds of detailed comparisons needed to assess design trade-offs. The simulation program we have developed is flexible enough to permit evaluation of many different system configurations, is readily extensible, and is efficient enough to support detailed studies. Some of the design trade-offs we have examined are described below. We have also characterized the performance of the Copy Network, which performs packet replication. Ours is the first performance study of this type of network.

4.1. Evaluation of Architectural Trade-offs

Delay and throughput curves for the Routing Network are shown in Figure 4.1. This models a switch consisting of input buffers feeding directly into a six stage Routing Network. The CN and DN have been omitted to permit a detailed examination of the RN. This delay plot (and all subsequent ones) gives delay in packet times where a packet time is just the time required to transmit a packet across one of the internal data paths. Assuming 5000 bit packets and a 200 Mb/s data rate on the internal data paths, one packet time is 25 μs. Two sets of delay curves are given. The curves that flatten out at large offered loads give the delay through the RN alone, while the other curves give the total delay through the

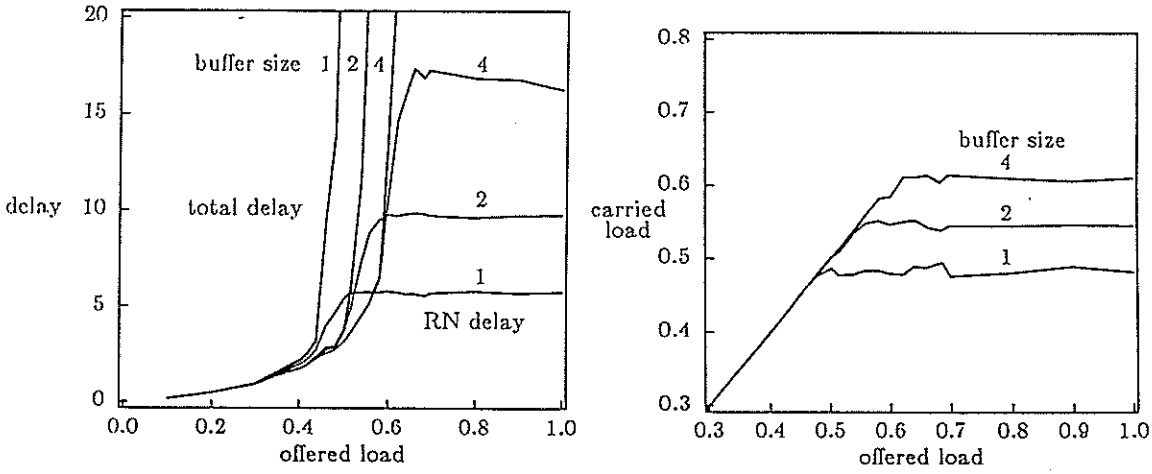

Figure 4.1: Delay and Throughput Curves for Routing Network

input buffers and the RN. These simulations were run under the *uniform traffic assumption*—that is, packets generated at the sources were assigned random destination addresses with each destination address equally likely.

In these plots, offered load is expressed as a fraction of the capacity of the switch fabric under ideal conditions. Under such ideal conditions, the switch fabric is capable of passing a packet out of every output port during every packet cycle. What has been plotted is the offered load measured at the packet sources. Carried load is the fraction of the ideal capacity that the system actually delivers and is measured at the outputs of the switch fabric.

The results in Figure 4.1 give the delay for nodes with 1, 2 and 4 buffer slots per input port. As one would expect, increasing the number of buffers increases the throughput of the network and reduces total delay. The carried load curves show clearly the effect of node buffer size on network throughput. Since the internal data paths operate at twice the rate of the external links, the offered loads within the switch fabric cannot exceed 0.5. If the external links carry a load of 0.8, the internal loading is 0.4. The carried load plot shows that even with a buffer size of one, the RN has sufficient throughput to carry this load. On the other hand, the delay plot shows that the safety margin is unacceptably small with this buffer size. A buffer size of two is probably required to provide an adequate margin against occasional traffic surges. It's worth noting that at low loads, most of the delay is incurred within the RN while at high loads the RN delay becomes constant while the input buffer delay becomes very large. Note that at an offered load of 0.4, the total delay is approximately two packet times, which translates to about 50 μs, assuming 5000 bit packets and a 200 Mb/s data rate.

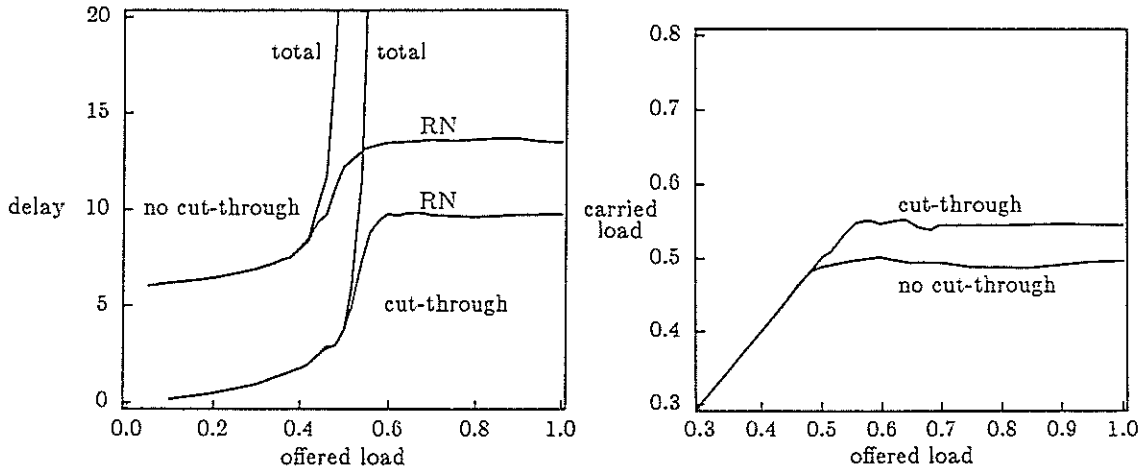

Figure 4.2: Effect of Cut-Through

Perhaps the principal difference between the Routing Network described here and buffered binary routing networks discussed elsewhere is the use of *cut-through operation* in the switch nodes. In most previous work, packets are buffered completely at each node in the switch fabric before being advanced to the next stage. We relax this restriction, allowing a packet to proceed immediately to the outgoing link whenever possible. Cut-through switching has also been employed in the Fast Packet Switch described in [32,45,47]. The concept of cut-through switching was first discussed in print in [26], although in a different context. Their analysis of the performance of cut-through switching shows significant advantages, but does not apply to the specifics of our situation. To quantify the advantage of cut-through switching we performed a simulation of the Routing Network in which cut-through was not used. This simulation is for a 64 port routing network with two slot buffers at the node inputs. The resulting comparison appears in Figure 4.2.

Notice that at low loads, the total delay without cut-through is at least six packet times, corresponding to the enforced buffering at each stage of the simulated network. When cut-through is used, the total delay is less than a single packet time. At an offered load of 0.4, the total delay with cut-through is about two packet times, while without cut-through, it is about eight. This is a significant advantage in the target application as it translates to a delay of 50 μs vs 200 μs. The delay advantage is even more striking than this comparison reveals, since (as we will see later) most of the delay in the switch fabric comprising the CN, DN and RN occurs in the RN when cut-through is used. Without cut-through the delay would be at least 18 packet times, whereas with cut-through, it is less than three packet times. We also note that the use of cut-through increases the throughput of the switch fabric from 0.5 to 0.55.

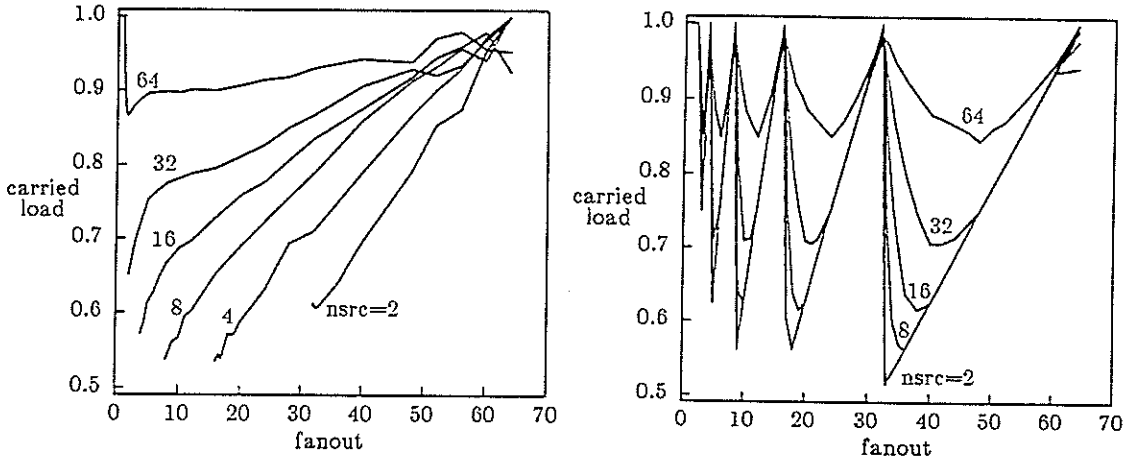

Figure 4.3: Dependence of CN Throughput on Fanout and Number of Sources

Other Trade-offs

The effectiveness of the Distribution Network has been studied both analytically and through simulation. Both approaches lead to the conclusion that the combination of a DN and RN is robust in the face of pathological traffic patterns.

We have also studied the effect of switch element size on RN performance. A surprising discovery was that fabrics made up of 2×2 switch elements can perform better than fabrics made up of larger switch elements. This turns out to be a subtle side-effect of the switch element design. In particular it is caused by the fact that packets are buffered on the input side of the switch element and are served in a strict FIFO order. The strict FIFO ordering limits throughput unnecessarily in cases where the first packet in a buffer is unable to proceed, and there is a packet behind it in the buffer that could proceed if it were able to jump to the head of the queue. It turns out that this effect has a greater impact on systems made up of large switch elements than it does on systems made up of small ones. We have simulated a system that allows *bypass queueing*, and obtained substantial performance improvements over the strict FIFO queueing. We have also found that when bypass queueing is used, networks constructed from large switch elements out-perform networks constructed from small ones.

We plan additional studies in order to obtain a better understanding of the performance implications of different buffering strategies. In particular, we plan to compare input buffering, output buffering and shared buffering.

4.2. Copy Network Performance

The performance of the Copy Network is limited by the contention caused by packet replication. The results presented here are for a 64 port CN wth two port nodes, each having two buffer slots per input. The CN is preceded by a set of input buffers. This configuration allows us to focus on properties determined entirely by the CN. We note that offered load must be defined differently for a Copy Network than for an ordinary binary routing network, since the number of packets exiting the CN is generally larger than the number entering. Consequently, we define offered load to be the product of the input load and the average fanout.

Figure 4.3 shows how throughput is affected by fanout and the number of active sources. For this plot, the offered load was held at one and packet arrivals were limited to the first *nsrc* inputs to the CN. In the plot shown on the left, the fanouts assigned to each packet were selected from a truncated geometric distribution. In the plot on the right, the fanouts were held constant for each simulation run.

Consider first the case of random fanouts. The best performance is obtained when all inputs are active ($nsrc = 64$). Note that the throughput starts at 1, drops to a minimum of about .87 when the average fanout is approximately 2, and then gradually increases. The initial drop in throughput is caused by the contention producing effects of packet replication. The subsequent increase in throughput with fanout is caused by the fact that to maintain an offered load of 1 as fanout increases, the packet arrival rate at the inputs must decrease. This reduces contention and balances the increase in contention caused by packet replication. When the number of active sources is small, there is a lot of contention in the early stages leading to sharply lower throughputs at small fanouts. We note again that as fanouts increase, throughput climbs rapidly due to the lower arrival rates at the input ports. Note that in all cases, the throughput exceeds 0.5.

The plot on the right in Figure 4.3 shows the effect of different fanout values more clearly. For each value of *nsrc*, the throughput takes on values close to one whenever the fanout is a power of two. Between powers of two, the throughput drops sharply achieving its minimum value when the number of sources is small. The excellent performance at fanouts equal to powers of two is an expected effect as is the sharp deterioration just above powers of two. This deterioration is explained by noting that when the fanout passes a power of two, copying begins one stage earlier in the network than previously, while the packet arrival rate is about the same. This leads to additional contention and the observed performance deterioration. As the fanout continues to increase, the arrival rate decreases (in order to maintain the same offered load), reducing contention in the early stages and improving performance.

4.3. Fluid Flow Loading Analysis

Performance studies of packet switch fabrics generally assume some form of uniform traffic distribution. For binary routing fabrics for example, it is common to assume that each packet entering the system is assigned a random destination with all destinations equally likely. While this yields legitimate and useful performance data, it does not tell the full story, since there are pathological traffic patterns that can lead to congestion.

We have developed a simple method for determining the approximate effect of a particular traffic pattern on a packet switching fabric. This *fluid flow loading analysis* allows us to prove theorems about the performance of packet switching fabrics under general loading conditions. For example, we can show that there is no legitimate traffic pattern that can overload any of the internal links in a switch fabric consisting of a distribution network and a routing network.

We illustrate our approach for the Copy Network. We view the traffic imposed on an n port Copy Network as coming from a collection of sources $S_1, \ldots, S_m$. Each source S_i is described by an ordered triple (P_i, B_i, F_i), where P_i is the input port at which packets from S_i arrive $(0 \leq P_i \leq n)$, B_i is the average arrival rate of packets from S_i $(0 \leq B_i \leq 1)$ and F_i is the required fanout $(1 \leq F_i < n)$. A *traffic configuration* (or simply configuration) is a set of sources. In a *legal configuration* the sum of arrival rates of sources associated with any particular input port must be at most 1. That is, for $i \in [0, n-1]$,

$$\sum_{j, P_j = i} B_j \leq 1$$

Also, the sum of the products of arrival rate and fanout must be no more than n.

$$\sum_{j=0}^{m} B_j F_j \leq n$$

Configurations that violate these restrictions exceed the capacities of either the input ports or the output ports of the CN and hence its performance under such conditions is immaterial.

We label the links in the CN with ordered pairs (i, j) where i is the stage number of the link and j is its index within the stage. Stage 0 links are those feeding the nodes in the first stage, stage 1 links connect the nodes in the first and second stages, and so forth. Links are numbered within stages from top to bottom at their left ends, starting from 0. Hence, the first link leaving the top node in the first stage is link $(1, 0)$.

Let $S = (P, B, F)$ be a source. We define the *load induced by* S on a link (i, j) in the Copy Network by

$$\lambda_{ij}(P,B,F) = \begin{cases} 0 & \text{if there is no path from link } (0,P) \text{ to link } (i,j) \\ 2^{-i}B & \text{if there is a path and } 0 \le i \le k - \lceil \log_2 F \rceil \\ 2^{-(k-\lceil \log_2 F \rceil)}B & \text{if there is a path and } i \ge k - \lceil \log_2 F \rceil \end{cases}$$

This definition simply reflects the behavior of the CN with respect to a particular source. When a packet from source S enters the CN it follows a random path until it reaches stage $k - f$ where $k = \log_2 n$ is the number of stages and $f = \lceil \log_2 F \rceil$. Stage $k - f$ is the first at which replication takes place. For a set of sources S, we define

$$\lambda_{ij}(S) = \sum_{(P,B,F) \in S} \lambda_{ij}(P,B,F)$$

The following theorems are proved in [4].

THEOREM 1. Let f be an integer in $[0, \log_2 n]$ and let $S = (S_1, \ldots, S_m)$ be any legal configuration in which $F_i = 2^f$ for $1 \le i \le m$. Then, for any link (i, j), $\lambda_{ij}(S) \le 1$.

Theorem 1 tells us that when all the sources have fanouts equal to the same power of two, there is no legal configuration that can induce an overload on any internal link. If we drop the restriction to powers of two we get the following.

THEOREM 2. Let F be an integer in $[1, n]$ and let $S = (S_1, \ldots, S_m)$ be any legal configuration in which $F_i = F$ for $1 \le i \le m$. Then, for any link (i, j), $\lambda_{ij}(S) \le 2$.

So, if the fanouts are all equal, the CN can handle any legal configuration in which the arrival rates on all input ports are ≤ 0.5 without inducing an overload on any internal link. We get a similar result ($\lambda_{ij} \le 2$) if we insist on fanouts equal to powers of two but drop the restriction that all fanouts be equal. Finally, if we drop all restrictions we obtain the following.

THEOREM 3. Let $S = (S_1, \ldots, S_m)$ be any legal configuration. Then, for any link (i, j), $\lambda_{ij}(S) \le 3$.

This implies that the CN can handle any legal configuration in which the arrival rates on all input ports are $\le 1/3$ without inducing an overload on any internal link. The worst-case performance of the CN can be improved by adding a few distribution stages at the front end. In particular, we have

THEOREM 4. Let $S = (S_1, \ldots, S_m)$ be any legal configuration for a k stage copy network preceded by r distribution stages. Then, for any link (i, j), $\lambda_{ij}(S) < 1 + 2^{1-r}$, if $i < k + r$ and $\lambda_{ij} < 2$ if $i = k + r$.

So for example, if two distribution stages are added, the maximum induced load (excluding the last stage) is 1.5. Such a configuration should be able to

handle any legal traffic pattern assuming a speed advantage of two. In each of the above theorems, the bound on $\lambda_{i,j}$ cannot be improved. That is, in each case there is a worst-case traffic pattern that induces loads on some internal links that equal or approach the stated bounds.

5. Testing Cascaded Binary Routing Fabrics

Principal Investigator	Jonathan Turner
Graduate Students	Shabbir Khakoo

Testing is essential for a practical implementation of a broadcast packet switch fabric. We have developed a method that allows easy identification of single link and node failures in switch fabrics consisting of several cascaded binary routing networks. Our method generalizes one developed by Feng and Wu [11]. Their method was designed for unbuffered networks in which the testing processor has direct access to all input and output ports. We have extended it to handle cascaded networks in which the testing processor has direct access to only one input/output port pair. For a switch fabric with n input and output ports and three cascaded networks, our method can detect and locate any single fault with no more than $6n$ test packets. Feng and Wu's method, when applied to a single network requires $2n$ test packets.

5.1. Testing a Single Network with Full Access

Feng and Wu's test method for a single binary routing network involves two test phases. In the first test phase, n packets are sent and at each node, a packet entering on input 0 also exits on output 0, while a packet entering on input 1 exits on output 1. This is the *straight-through* phase and is illustrated on the left in Figure 5.1. During the second phase, packets entering a node on input 0 exit on output 1, while packets entering on input 1 exit on output 0. This is the *cross phase* and is illustrated on the right in Figure 5.1

We assume that a failing link or node causes all packets passing through it to fail. (This assumption can be satisfied by the inclusion of redundant hardware.) Under this assumption, a single failing link causes the failure of exactly one test in each phase. The failing tests have exactly one link in common, which consequently must be the faulty link. A single failing node will cause two tests in each phase to fail and again, there is exactly one node common to all the failing tests, hence the faulty node is easily identified.

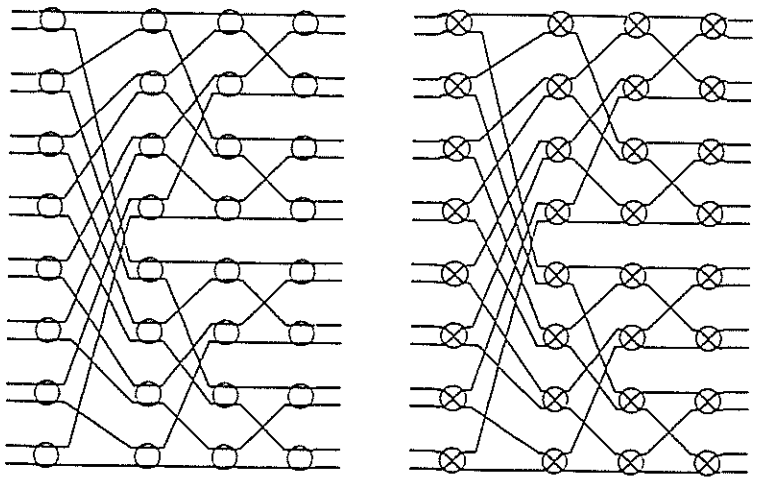

Figure 5.1: Two Test Phases for Single Network

5.2. Testing Cascaded Networks

An *m-cascade* is a sequence of m binary routing networks connected in sequence. To permit effective testing, no two nodes should be directly connected by more than one link. This means that the outputs of one network in the cascade cannot be connected directly to the corresponding inputs of the following network. Hence we permute the connections slightly, as illustrated in Figure 5.2.

To test an m-cascade we perform a test sequence consisting of $m+1$ phases. In phase 0, packets pass through each node using the straight-through paths, in all m networks in the cascade. For phase i $(1 \leq i \leq m)$, packets pass through network i using the cross paths and all other networks using the straight-through paths. This is illustrated in Figure 5.3.

A failure of a single link causes exactly one test in each phase to fail. There is exactly one link that is common to all these paths. So we need only find that common link to identify the fault. A failure of a node causes two tests to fail in each phase. Again, there is only one node common to the paths used by all these tests and we need only find that node to identify the fault.

5.3. Testing Networks With Limited Access

In the broadcast packet switch fabric, testing is performed by the Connection Processor (CP). The only direct access that the CP has to the switch fabric is

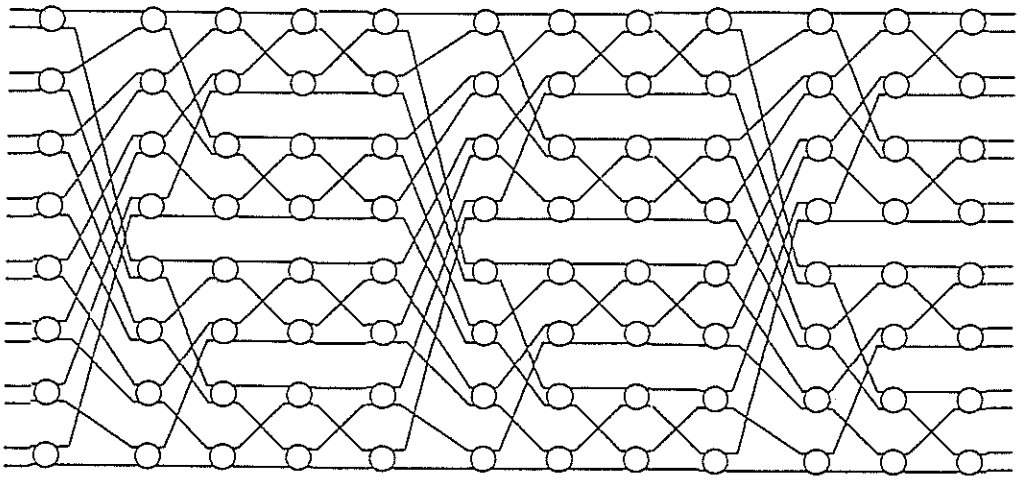

Figure 5.2: Three Cascade

through the one port pair by which it's connected. To perform a complete test, it requires the cooperation of the Packet Processors (PP). A test packet is sent by the CP to a specified PP (through the switch fabric). Upon receipt, the PP takes a field from the body of the packet, puts it in the routing field and sends the packet back to the switch fabric. It is received by a second PP, which puts the CP's address in the routing field and sends it once more to the switch fabric, which returns it to the CP. The middle leg of the journey forms the main part of the test. The first and last legs are just a mechanism that gives the CP effective access to all ports.

One must of course, account for the possibility that test packets are lost during the first and last legs of a test, in order to interpret test results correctly. Our approach is to perform a set of *access tests* before performing the detailed tests described in the previous section. the access tests verify that that there is some usable path from the CP to each PP. If the access tests are successful, the test method discussed above can be applied directly to check for and identify any faulty links. If we cannot find a usable path to some PP, we attempt to localize the fault as far as possible. It turns out that it is not possible to isolate the fault to a single node or link. The best that can be done in general is to localize the fault to two links and two nodes.

The access tests require $2n$ test packets, meaning that a full test of a 3-cascade requires $4n$ tests, and in general $(m+3)n$ tests are required for an m-cascade. A complete description of our testing methods appears in [31].

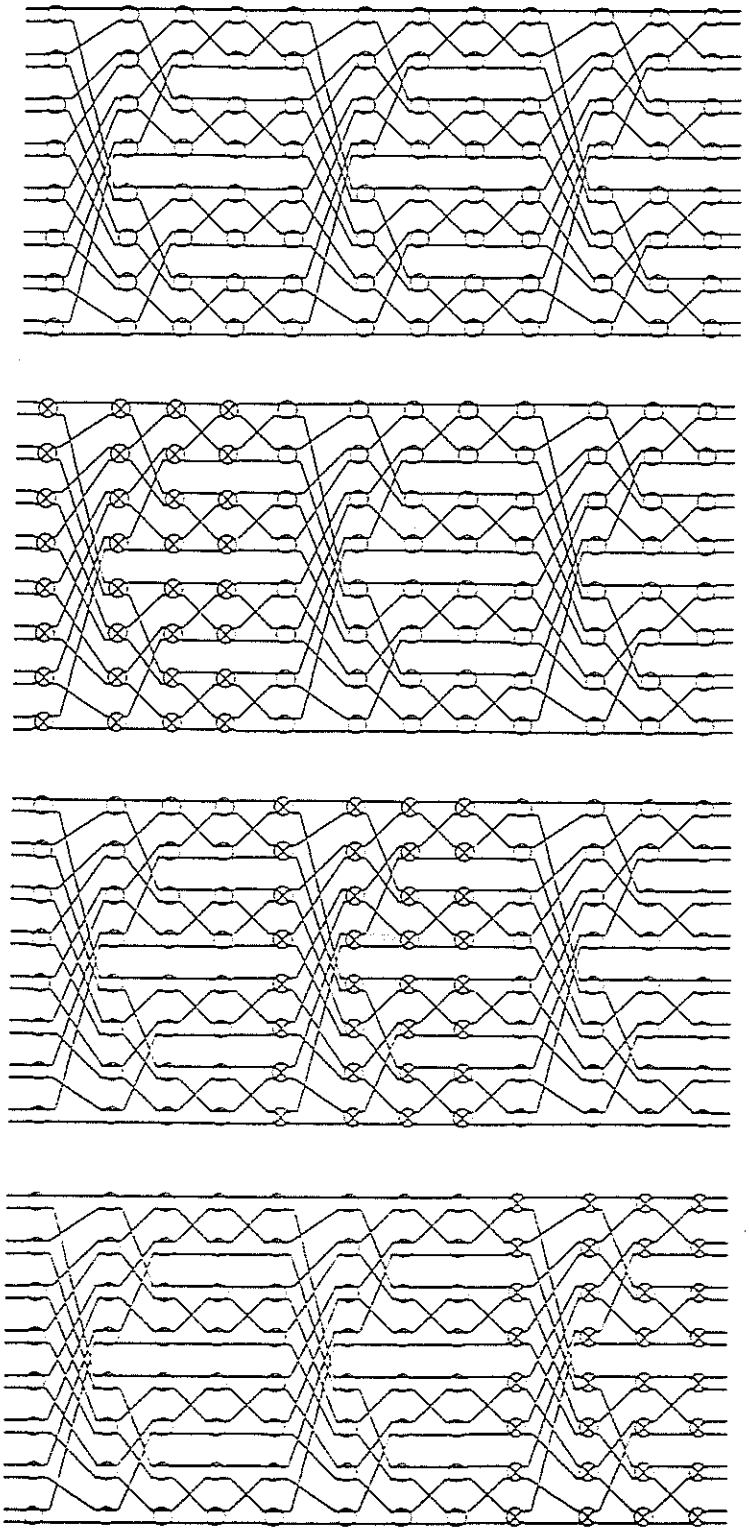

Figure 5.3: Four Test Phases for Cascaded Networks

6. Automatic Generation of Synchronized Streams Processors

Principal Investigator	Jonathan Turner
Graduate Student	George Robbert

The implementation of the prototype packet switch for the ACS project will require several custom VLSI chips. Most of these chips are composed primarily of functional blocks 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 *synchronized streams processor*. This has led us to consider the possibility of a program that takes a high level specification of a streams processor and uses it to generate a circuit implementing that specification.

6.1. Synchronized Streams Processors

A generic synchronized streams processor (SSP), is a module with one or more typed input and output *ports*, a local clock sychronized by external timing signals and a function which can be described in a style similar to a conventional programming language (see Figure 6.1). 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 ϕ. 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. As an example, the following declaration defines the format of an internal packet in the BPN prototype.

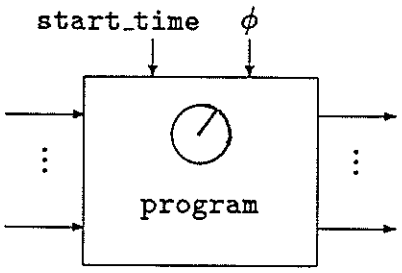

Figure 6.1: Generic Synchronized Stream Processor

```
struct ipfmt {
    bit op[5], rc[3];
    union { bit fan[8]; bit ln[8]; };
    union { bit bcn[16]; bit ilcn[16]; };
    bit src[8];
    bit fill[4], ptyp[4];
    bit elcn[16];
    bit info[72][8];
};
```

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.

We now turn to a simple example to illustrate how an SSP can be described. The circuit we will describe combines the functions of the IN and XCVT circuits of the packet processor. It accepts an input packet from the switch fabric on one port and based on the control field directs the packet to either the XMB or STB. For test packets, the RC field is set to 0 and the packet is rotated by five words. In addition, packets sent to the XMB are placed in a format that is intermediate between the internal and external formats; basically, it is the internal format with the first five words removed. As part of this transformation, the logical channel number in the ILCN field is copied to the ELCN field.

```
struct hpfmt {              // Hybrid packet format
    bit fill[4], ptyp[4];
    bit elcn[16];
    bit info[72][8];
};
typedef bit tpfmt[80][8]; // Test packet format
union mpfmt {               // Mixed packet format
    struct ipfmt;
    struct tpfmt;
};
module inxcvt(port[8] struct mpfmt  <sfp@0, >ltbp@2,
              port[8] struct hpfmt >xmbp@6 )
    if sfp.op == OP_DATA ->
        xmbp.fill = sfp.fill;
        xmbp.ptyp = sfp.ptyp;
        xmbp.elcn = sfp.ilcn;
        xmbp.info = sfp.info;
     | sfp.op == OP_STEST ->
        ltbp[0:74] = sfp[5:79];     // Rotate
        ltbp[75:79] = sfp[0:4];
        ltbp[75][0:2] = 0;          // Clear rotated RC
     | sfp.op != OP_DATA && sfp.op != OP_STEST ->
        ltbp = sfp;
    fi;
end
```

The example module defines three eight bit wide ports; `sfp` is an input port (indicated by <) carrying data in `mpfmt`, starting at time 0; `ltbp` is an output port, also carrying data in `mpfmt`, starting at time 2; `xmbp` is an output port carrying data in `hpfmt`, starting at time 6. The program specifies the appropriate action based on the `OP` field of the incoming packet. The assignments define the contents of the various output fields. Unspecified output fields are filled with zeros. Constant ranges are allowed in subscript expressions. Notice that the only times that must be defined explicitly are the times at which data items start to flow across ports. Although not shown in this example, addition, subtraction and logical operators can appear in expressions. Multiplication and division are permitted in constant expressions.

The simple paradigm of typed, synchronized ports can also be used to define control signals that must be exchanged between different modules. This allows us to define more complicated interfaces such as are required on the output side of various buffers. Modules can also have local variables that can be used to save information across time epochs.

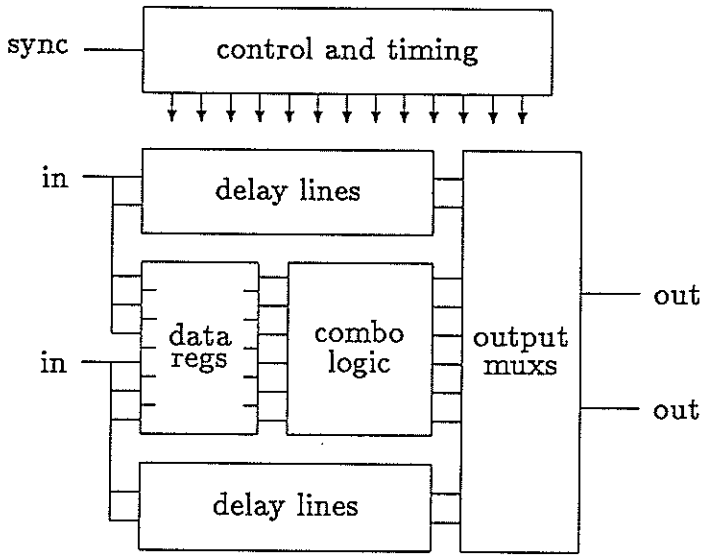

Figure 6.2: Target SSP Architecture

6.2. Implementation of SSPs

SSPs that perform simple functions, as are typical in the Packet Processors, fit nicely into a common architecture illustrated in Figure 6.2. This architectecture supports several input and output ports of varying widths. Selected input fields are latched in a set of *data registers*, which in turn feed into a *combinational logic section*. The data registers provide storage for local variables in addition to latching input values. *Delay lines* are provided to delay the data as needed to satisfy timing constraints, and *output multiplexors* are used to select the data to be transmitted on the various output ports. A global *control and timing section* provides the required timing signals to the input registers and output multiplexors.

Our objective is to develop a circuit generator that takes a high level description of an SSP and creates a circuit implementing it, by tailoring the target architecture. We plan to divide the translation into several parts as illustrated in Figure 6.3. The *compiler* will take the high level module description and generate an intermediate-level description. This will be further processed by an *assembler* which will generate the actual mask-level description of the module, using a library of standard cells and existing tools such as a PLA generator and router.

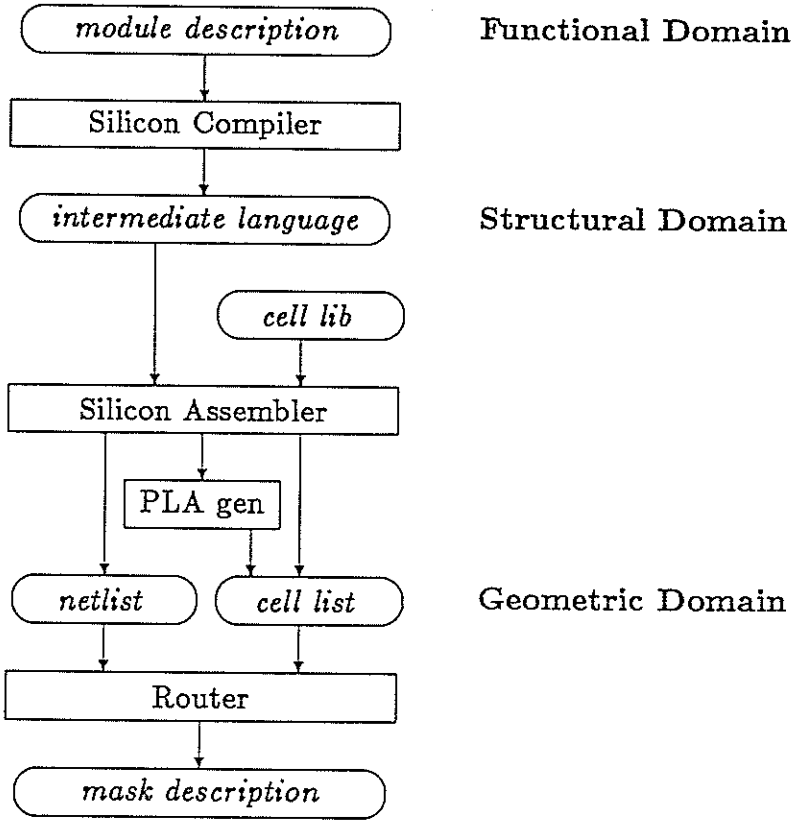

Figure 6.3: Structure of A Silicon Compiler

7. Connection Management

Principal Investigator	Jonathan Turner
Graduate Students	Victor Griswold
	Mark Hunter
	David Wexelblat

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. This section briefly describes one approach to connection management, that provides a flexible way of defining multipoint connections supporting a wide variety of applications and which has the potential for being effectively implemented in large networks.

We start by describing a simple one-way broadcast connection, illustrated in Figure 7.1. The connection has a single transmitter G and receivers C, D, F and J. The subscripts in the diagram represent the channel numbers that the various terminals use to identify this particular connection among the set of connections present on the access links. The internal nodes in the diagram represent switching systems and at various points the stream of packets originating at G is replicated and forwarded to the appropriate points. 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

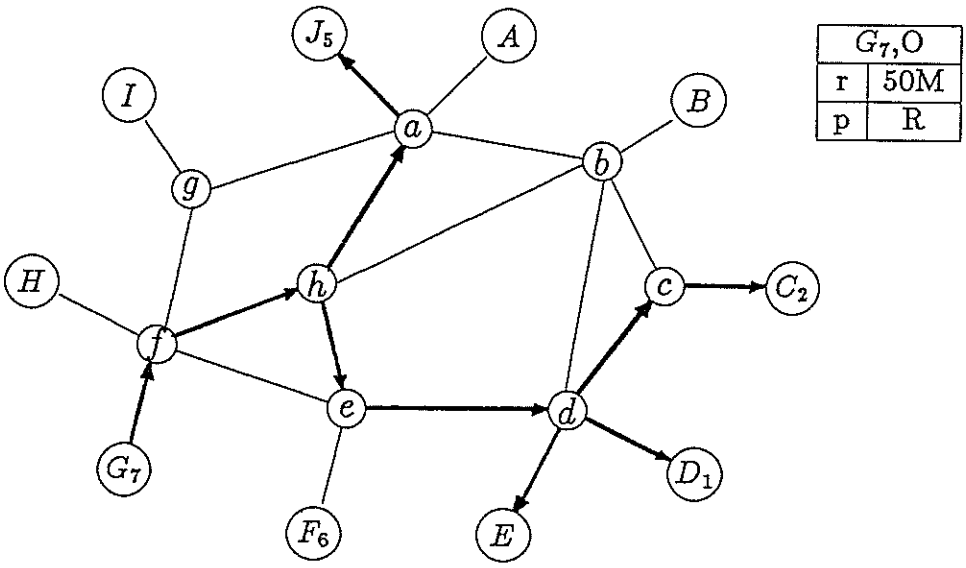

Figure 7.1: One-to-Many Connection

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

This example has illustrated the essential notions of the multipoint connection mechanism. A point-to-point connection for voice communication can be readily described using these mechanisms. In that case, the rate specification might be 50 kilobits per second rather than 50 megabits, the default permission would be TR, for transmit/receive and the outside access specification would be C (closed). Notice that in neither example, is there any need for the switching systems supporting the connection to have detailed knowledge of the application, such as whether the signals carried are voice or video. Such information is required in the terminals and possibly network interface equipment (depending on the desired form of access), but is not needed anywhere in the core of the network. This *application-ignorance* is important for maintaining the flexible nature of the network. By keeping that information from the network core, we ensure that changes in the applications cannot affect the network's operation in any fundamental way.

Figure 7.2 gives an example of a more complicated connection requiring two related but separate channels within a single connection. This connection is intended for use in a video lecture situation. It includes a downstream channel from A to C, E and G, which has a rate of 30 megabits per second and a

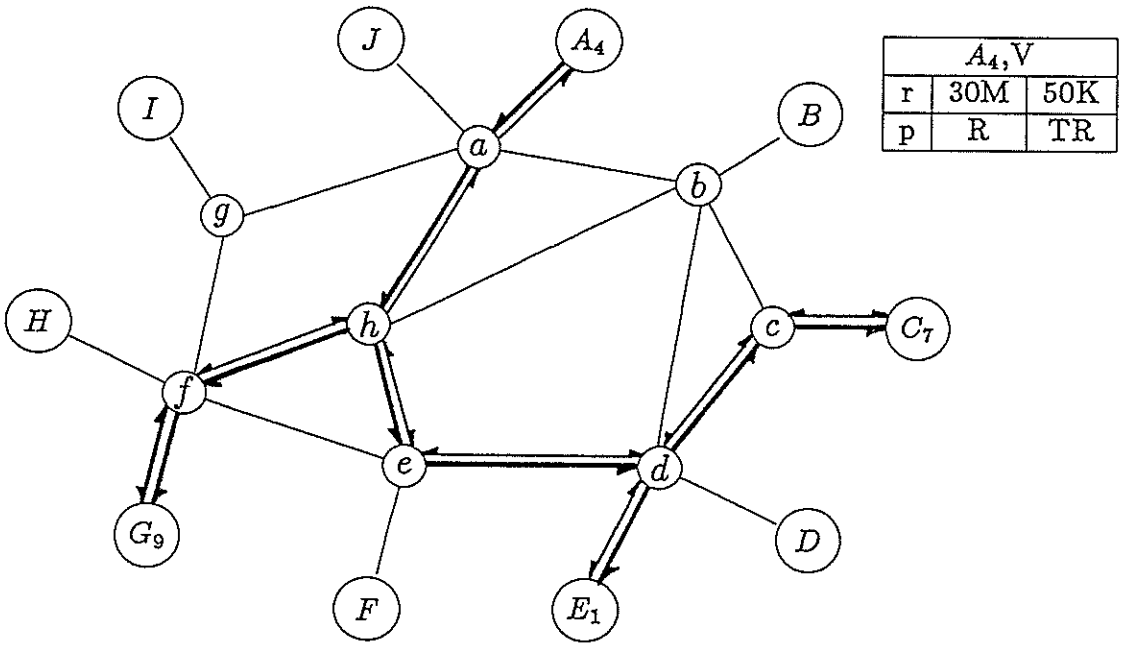

Figure 7.2: Connection for Video Lecture

default permission of receive-only. It also has a second channel, which can be used for upstream audio, allowing the audience members to direct questions to the speaker. It has its own rate and default permissions specified. The network allocates link bandwidth separately for the two channels, but routes them through the same set of switches, for convenience in handling addition of new endpoints. The concept of multiple channels within a connection, is an important extension of the basic multipoint connection mechanism that can be useful in more complex applications.

8. Multipoint Routing

Principal Investigator	Jonathan Turner
Graduate Students	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 modelled as a graph $G = (V, E)$ where the nodes of a graph represent switches and the edges represent links. In addition we have two functions cap : $E \rightarrow \Re^+$ and cost: $E \rightarrow \Re^+$ which give us the bandwidth and cost of each edge (link). In this model we equate cost and edge length. At the time a connection is established a shortest path connecting the pair of endpoints is selected. Of course only paths consisting of edges with sufficient unused bandwidth may be chosen.

Routing multipoint connections may be modelled 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 of a graph connecting a set of points is a classical problem in graph theory known as the Steiner tree problem in graphs [14]. This problem has been shown to be NP-complete by Karp [33] in 1972. Consequently one is forced to consider approximation algorithms which are not guaranteed to produce optimal solutions.

8.1. Approximation Algorithms

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) [27] is probably the best known approximation algorithm. It can be shown that a solution produced by this algorithm will have cost that is never worse than twice that of an optimal solution. Our experimental results indicate that MST typically yields

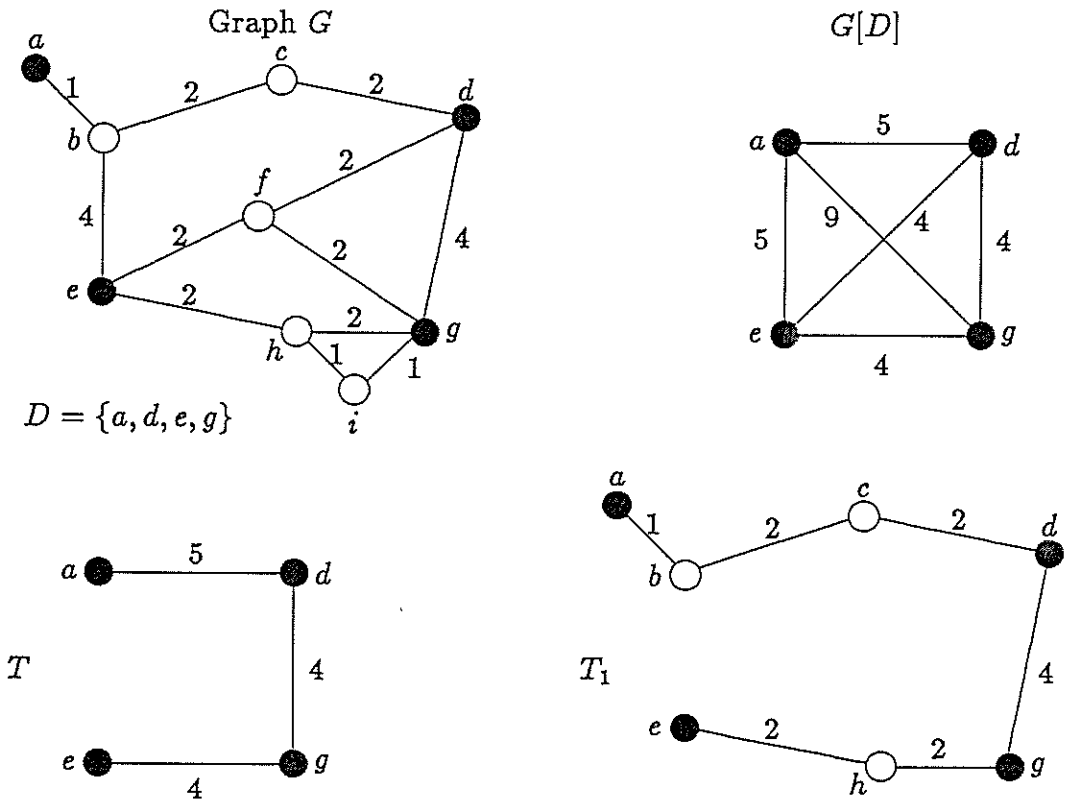

Figure 8.1: An Example of the Application of MST

solutions that are within five percent of optimal. Figure 8.1 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 the 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.

To gain some insight into the probable performance of MST we have implemented two versions of this heuristic using the C++ language. The first version we refer to as the basic MST heuristic and is the algorithm described in the previous paragraph. We refer to the second version as the improved MST heuristic. Improved MST takes the results of basic MST and forms a set D_1 which contains

all of the nodes of D plus any additional intermediate nodes used in the solution produced by basic MST. Basic MST is then applied to the set D_1. From the resulting tree any branch which does not contain nodes in D is pruned. Improved MST will always yield a solution that is at least as good as the solution produced by basic MST, but in the worst case will do no better.

For the purpose of running these experiments we developed a simple random graph model (RG1), which has some of the characteristics of an actual network. In RG1 n nodes are randomly distributed over a rectangular coordinate grid. Each node is placed at a location with integer coordinates. Edges are then introduced between pairs of nodes u, v with a probability that depends on the Euclidean distance between them. The probability function is given by

$$P(\{u,v\}) = \beta \exp \frac{-d(u,v)}{\alpha L}$$

where $d(u,v)$ is the Euclidean distance between nodes u and v, L is the distance along the diagonal of the rectangular region, and α and β are parameters in the range $(0,1)$. The size of β determines the overall edge probability independent of distance, while small values of α decreases the density of longer edges relative to shorter ones. Finally the length or cost of each edge is set equal to the Euclidean distance between its endpoints.

The experimental results of applying MST to the graph model RG1 are illustrated in Figure 8.2. These experiments were run on five different twenty five node random graphs. Each experiment consists of choosing a subset of the graph nodes at random and then executing both basic and improved MST. In addition an exact algorithm was used to determine the cost of an optimal solution. Each data point on the graph represents a set of fifty experiments, ten on each of the five graphs. The graph displays the average ratio of the cost of the solution given by MST to an optimal solution. It also shows the ratio of the cost of the tree in $G[D]$ to the cost of the optimal solution and the ratio for the improved MST.

We have also investigated a second approximation algorithm due to Rayward-Smith [37], which we refer to as RS. This algorithm addresses the one major problem of MST. MST considers only the distance between pairs of nodes to be interconnected and does not give any consideration to the importance of intermediate nodes. Our improved MST heuristic does consider intermediate nodes, but not in a systematic way. Thus improved MST does not have worst case performance any better than that of basic MST. On the other hand, RS makes an attempt to choose intermediate nodes in a systematic way, based on a collection of functions $f_\ell : V \rightarrow \Re^+$. At each stage ℓ of this algorithm RS chooses a path through a node for which the function f_ℓ has a minimum value. We have proved the the worst case performance of RS is within twice optimal and have found a class of problems for which the performance approaches 3/2 optimal. It

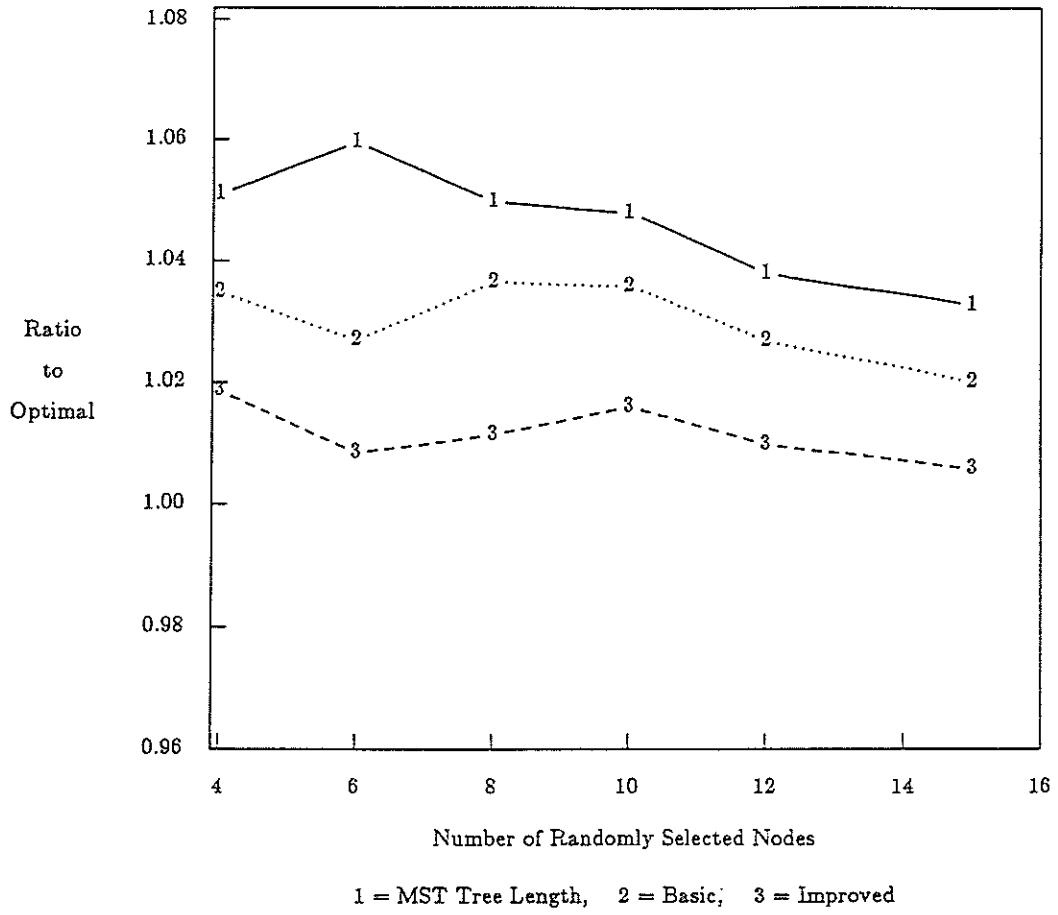

Figure 8.2: Experimental Performance of MST

is apparent from considering a number of simple examples that RS should do better than MST in most cases. At this point we actually believe that the worst case performance is actually within 3/2 of optimal but as yet have not been able to prove this result. If the worst case performance of RS is actually within 3/2 of optimal, this leads to the interesting possibility of modifying RS to yield even better worst case results while still maintaining polynomial time complexity.

8.2. The Dynamic Steiner Tree Problem

The algorithms just presented assume that the problem is bascially static, as can be solved in a centralized fashion. In a large network one must rely on distributed algorithms in which no individual processor has global knowledge. In addition one cannot expect to know in advance all of the nodes that will be in

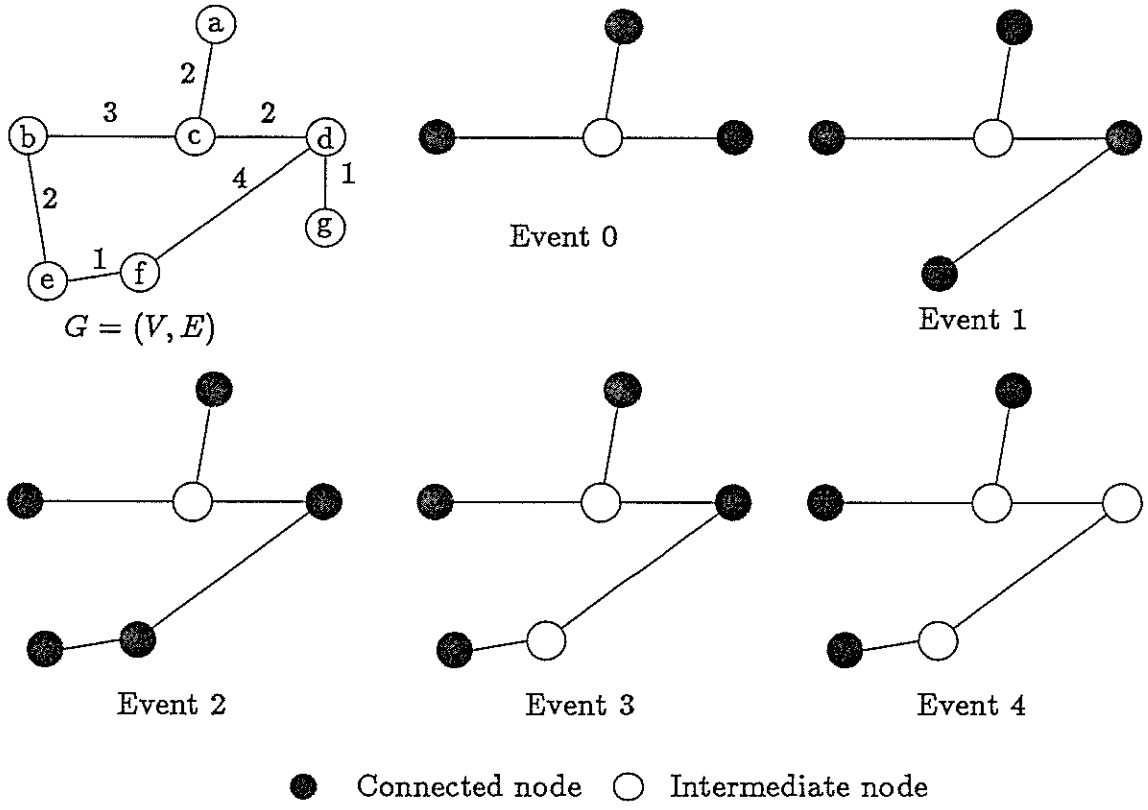

Figure 8.3: Dynamic Greedy Algorithm with Sequence: a,b,d,f,e, $\bar{f}, \bar{d}$

a given connection. Thus, in their present form the MST and RS heuristics will prove to be most useful as tools against which the performance of more realistic algorithms can be measured.

We now consider the problem of the dynamic multipoint problem in which nodes are allowed to join or leave a connection at any point during the lifetime of that connection. Probably the simplest algorithm for handling dynamic multipoint routing is one which we refer to as the greedy algorithm. This greedy algorithm adds new endpoints to a connection by using the shortest path from the endpoint to a node already in the connection. Nodes are removed from the connection by deleting that branch which serves only the node to be removed. Note that if a given node is an endpoint in a connection as well as an intermediate node in a path to other nodes, removing it as an end point will not allow us to delete it from the connection. Figure 8.3 illustrates a sequence of five events handled by the greedy algorithm. Note that event four gives an example of a situation where the connection is not optimal. In fact this connection has a cost of three units more than that of an optimal solution.

We have implemented this greedy algorithm in C++ and have run a set of experiments on 195 node graphs, again using our random graph model RG1. For these experiments we used a simple probability model to determine if an event should be an addition or deletion of a node from a connection. The function

$$P_C(k) = \frac{\alpha(n-k)}{\alpha(n-k)+(1-\alpha)k}$$

was defined for this purpose. Here P_C is the probability that an event is an addition of a node, k is the number of nodes in the current connection, n is the total number of nodes in the network and α is a parameter in $(0,1)$. The value of α determines the fraction of nodes in the connection at equilibrium. In other words αn is roughly the average number of nodes in the connection. For example, when $k/n = \alpha$, $P_C(k) = 1/2$.

Figure 8.4 shows the results from one experiment. The figure contains two curves, the bottom one showing the number of endpoints in the connection as the experiment progressed and the top one showing the ratio of the length of the solution produced by the greedy algorithm to the length of the MST solution at each point in time. (This experiment was run on a graph with 195 nodes and with $\alpha = 6/195$.)

These experimental results indicate that the greedy algorithm does reasonably well in comparison to MST. On average the performance of the greedy algorithm is within 35% of MST. It does poorly on a few isolated events, but notice that this poor performance occurs at times when several nodes leave the connection; that is, it is not caused by bad routing decisions, but by the inability of the greedy algorithm to reconfigure a connection after its been reduced in size. There are several techniques that might be used to reduce the sharp deterioration in these cases, at the cost of worsening the performance slightly in other cases. For example, the choice of a connection path for a new endpoint could be based on function of both the length of the path to a node in the connection and the distance of that node from the center of the current connection.

8.3. Distributed Routing Algorithms

We now briefly consider the problems involved with the development of a distributed algorithm without access to complete global information. Most of the work in this area deals with point-to-point networks. Algorithms have been developed which perform such functions as updating routing tables and finding shortest paths in a distributed fashion. For example the ARPANET makes use of distributed packet routing algorithms. Unfortunately most of the work with distributed routing algorithms cannot be applied directly to a network which

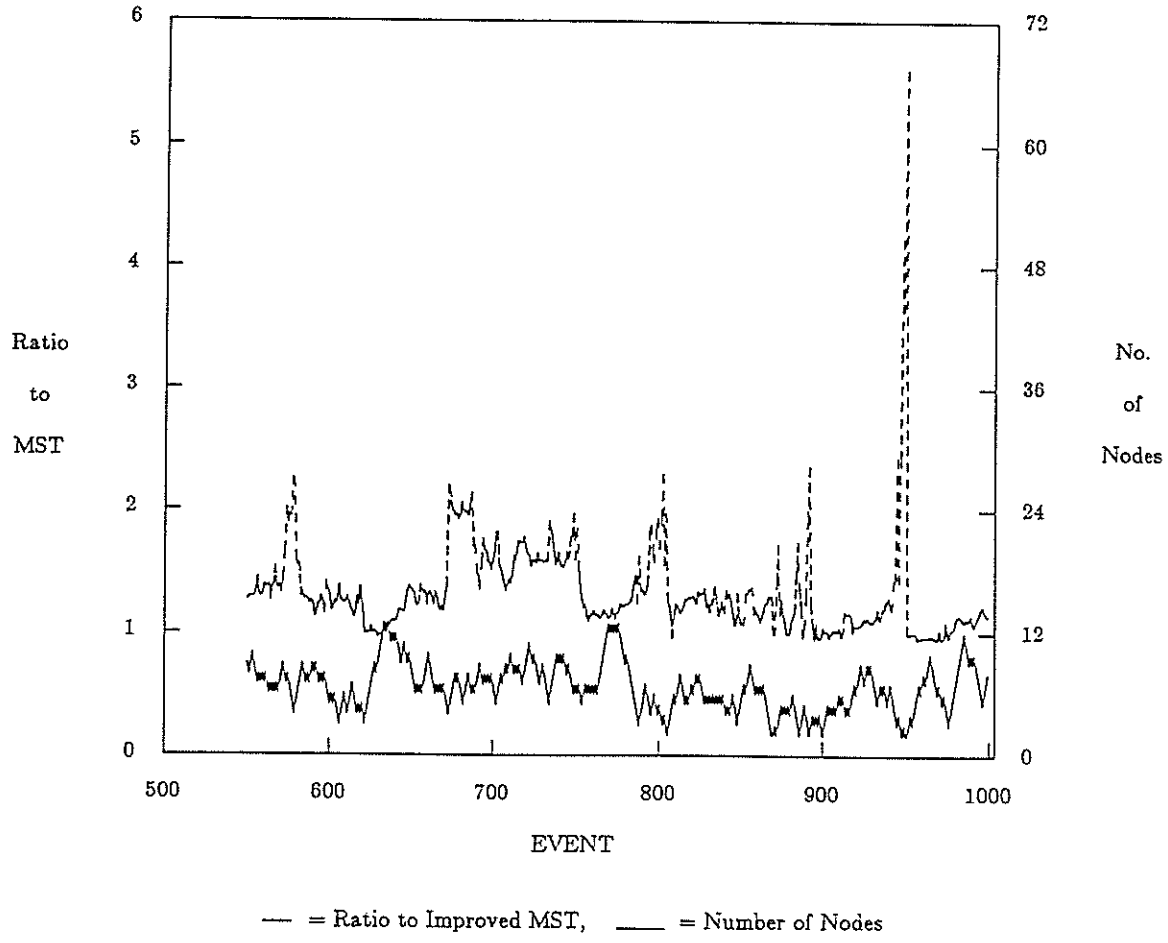

Figure 8.4: Experimental Results for the Greedy Algorithm

handles multipoint connections. Thus most of the existing results will serve primarily as a starting point for the development of new algorithms.

The work of J.M. Jaffe [24] is of particular interest. He has considered the problem of multipoint routing with limited information. His work considers the effects of limited information on the Steiner tree problem. He shows given certain assumptions that it is not possible for any algorithm to produce optimal solutions in all cases. For example, Jaffe presents a class of algorithms which are restricted to *local information* and shows that the worst case performance for these algorithms can be no better than $2k/3$, where k is the number of nodes to be connected. These results indicate, that with limited data storage at each node, we are not likely to find a routing algorithm which yields optimal routing. In fact Jaffe's results lead us to believe that any distributed algorithm based on the quantity of data, which can reasonably be assumed to be available for local processing in a very large network, is not likely to have worst case performance

even within a constant factor of optimal. On the other hand, Jaffe's results do not tell us anything about average performance, which is of great importance for a real network. Finally these results suggest that the performance of the greedy algorithm is reasonably good, and thus the greedy algorithm seems to be a reasonable starting point for the development of distributed multipoint routing.

9. Congestion Control

Principal Investigator	Jonathan Turner
Research Associate	Riccardo Melen
Graduate Students	Shahid Akhtar

A principal advantage of packet switched networks is their ability to dynamically allocate bandwidth to the users who need it at a particular instant. Since networks are subject to rapid statistical variations in demand, care must be taken to ensure acceptable performance under conditions of peak loading. Conventional packet networks employ a variety of congestion control methods that attempt to monitor the traffic in the network and limit new traffic when the network is heavily loaded. In high speed networks, many of the classical techniques don't work well. Several problems commonly arise. First, in high speed networks, traffic changes can occur very rapidly while congestion control mechanisms require a long time to take effect. Second, in high speed networks, congestion control mechanisms must be implemented in hardware, and many of the classical mechanisms are simply too complex for effective hardware realization. Third, in high speed networks, the resource requirements for different user applications can vary over more than six orders of magnitude, and for many applications, (eg voice, video) these are hard requirements that must be met in order to ensure satisfactory service; in this kind of environment, the network must explicitly account for the resource needs of individual connections, something that conventional packet networks do not do.

We feel that an effective congestion control system for a high speed packet network requires several specific methods, each acting on a different time scale. Long term overloads can be prevented by explicit allocation of bandwidth to connections and the refusal of new connections if the requisite bandwidth is not available. This implies that the network must provide a mechanism for users to specify their bandwidth needs and an indication of the burstiness of their transmissions, and must enforce limits to prevent users from exceeding their allocations. Short term demand variations are handled by buffering within the network. To limit delay variability and keep memory buffer memory sizes at a reasonable level, the amount of buffering at each link should probably be limited

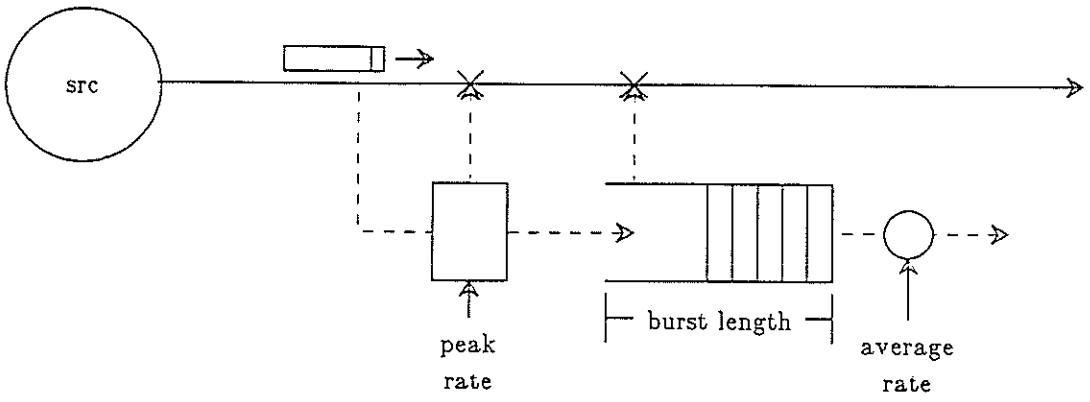

Figure 9.1: Simple Bandwidth Enforcement Mechanism

to a few milliseconds worth. Medium term variations are handled by selective shedding of traffic, with user-assigned priorities used to determine which packets to discard first.

The remainder of this section briefly reviews our work on congestion control, including a brief discussion of the new issues raised by multipoint connections with multiple transmitters.

9.1. Bandwidth Specification and Enforcement

When a user establishes a connection, he must provide a *rate specification* that describes some of the traffic characteristics of the connection. It must include the connection's *peak rate*, its *average rate* and some measure of how bursty it is. The network must be able to use the rate specification to determine what portion of each link's capacity should be allocated to the connection; this is referred to as the *effective rate* of the connection. In addition, the network must have a straightforward way of preventing the user from sending traffic at a higher rate than allowed by his rate specification. This is done by having the network monitor the connection's traffic as it enters the network and discard any packets that exceed the rate spec. We refer to this enforcement mechanism as a *traffic valve*. (Alternatively, the valve could reset the priority of the packet to a lower value and allow it to proceed, or buffer the packet and flow control the user across the access link.)

One simple way to implement a traffic valve is called the *pseudo-buffer* mechanism. The network monitors the flow of traffic entering on the connection and simulates the effect of this traffic on an imaginary buffer. When the user sends

a packet, the length of the pseudo-buffer is increased by one. So long as this doesn't cause the pseudo-buffer to overflow, the packet simply passes over the connection; if the pseudo-buffer does overflow, the packet is discarded. This idea is illustrated in Figure 9.1. The pseudo-buffer accepts packets at a certain maximum rate (the connection's peak rate), is drained at a constant rate (the connection's average rate) and has a maximum length (the connection's burst factor). These three parameters may all be specified by the user, allowing him in effect, to specify the characteristics of a *virtual private link*. The network uses these parameters to determine the connection's effective rate. The effective rate is selected to ensure acceptable performance (measured primarily in terms of packet loss rate) for all connections sharing links with the given connection. In general, the effective rate will lie somewhere between the peak and average rates. If the burst factor and/or the peak rate are small, the effective rate can be close to the average rate. If the burst factor is high (meaning comparable in size to the link buffers) and the peak rate is high (meaning comparable to the link rates), the effective rate may have to be close to the peak rate to ensure acceptable performance. This will be discussed in more detail in the following section.

A pseudo-buffer mechanism that simultaneously monitors all the logical channels on a single access link can be implemented in hardware in a straight-forward fashion. The main component is a memory that contains the values of the parameters associated with each pseudo-buffer, the state of the pseudo-buffer at the time it was last updated and a time field indicating when its state was last updated. Whenever a packet arrives on the link, the packet's logical channel number is used to select the appropriate set of parameters from the memory, the parameters are updated and written back to memory and a decision is made as to whether or not to discard the packet. The amount of information required for each channel is sixteen bytes, implying that 512 channels can be monitored using 64K bits of memory.

9.2. Determination of Effective Rates

In order for the network to allocate bandwidth effectively, it must be able use the rate spec provided by the user to compute an effective rate. This remains an unsolved research problem, although we are making progress in understanding it. In this section we consider data on the queueing behavior of bursty sources in order to provide a first-order indication of how effective rates might be determined.

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 bursty. When a bursty source becomes active it stays active

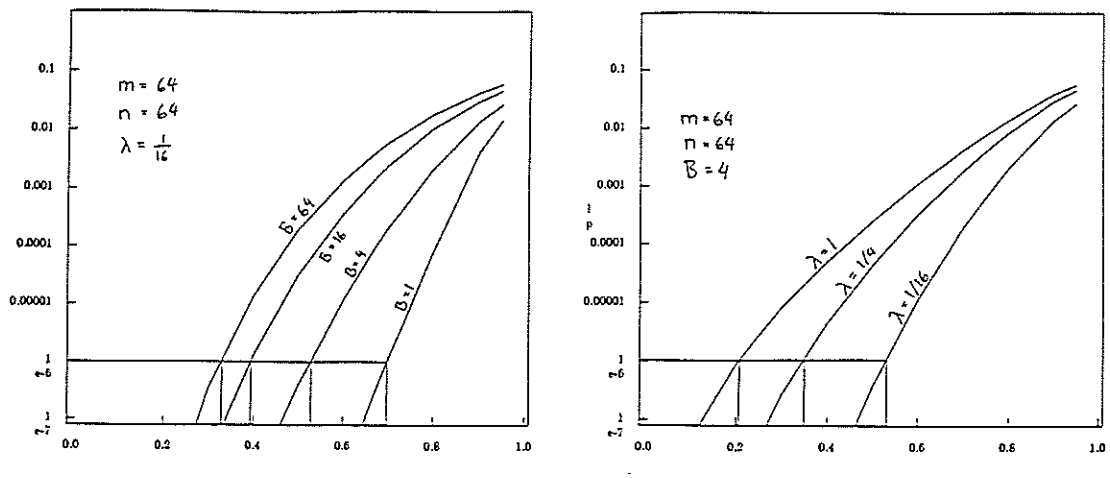

Figure 9.2: Packet Loss Rates

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 by m independent and identical bursty sources, as a finite Markov chain. This Markov chain can be solved numerically to determine the fraction of transmitted packets lost due to queue overflows. We show two plots obtained in this way in Figure 9.2. These plots show packet loss rate for a queue of length $n = 64$, fed by $m = 64$ sources. In the plot on the left, the peak rate λ is held constant at 1/16, while the offered load (ρ) and burst factor B are varied. In the plot on the right, the burst factor is held constant at 4, while the offered load and peak rate are varied. In both plots, we note packet loss rates substantially higher than predicted by an M/M/1 queueing model. Even for a burst factor of 1, our target loss rate of $10^{-}6$ is achieved at an offered load of about 70%, whereas an M/M/1 model predicts this loss rate at an offered load of about 83%. Increasing either the burst factor or the peak rate increases the loss rate substantially, requiring a corresponding decrease in the allowable offered load.

We are in the process of collecting an extensive set of data of the sort shown here. Based on these data we expect to be able to establish computationally efficient methods of determining effective data rates, given a rate specification.

9.3. Selective Traffic Shedding

While bandwidth allocation and enforcement can limit long term congestion, other methods are needed to cope with congestion of intermediate duration. One method for coping with short and medium term overloads is selective shedding of traffic, based on user-specified priorities. If half the packets on a specific link have low priority, that link can tolerate peak periods of long duration without having to discard high priority packets. Priorities can be used to advantage for signals containing large amounts of redundant information. For example, video signals can be transmitted with the high order bits of each pixel carried in high priority packets and the low order bits carried in low priority packets. Occasional loss of low priority packets would likely be imperceptible. Periods of a few seconds during which many low priority packets are lost, are likely to be perceptible, but only mildly annoying to the viewer. Similar methods have been used very effectively for packet transmission of voice signals.

Priorities can also be useful for applications with "soft" bandwidth requirements. Many data applications can tolerate a wide range of variability in throughput. For such applications, the higher loss rates associated with low priority packets may be tolerable. Economic incentives are of course required to induce users to take advantage of the lower priority.

9.4. Multipoint Congestion Control

The previous sections deal mainly with point-to-point connections and multipoint connections with a single transmitter. In these cases, it is straightforward to provide the bandwidth enforcement function entirely at the edge of the network. Connections with two or more transmitters and three or more receivers raise new issues, because 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.

When one considers applications involving multiple transmitters, it becomes evident that often, while there are many *potential transmitters* there is only one or a few *concurrent transmitters.* This suggests that the connection description should be augmented to include a specification of the number of concurrent transmitters. Given this information, the network can allocate sufficient bandwidth to support the specified number. It then of course, must ensure that this number is not exceeded.

There are several possible approaches to this last problem. If one wishes to control the problem entirely at the network boundary, one can apply the pseudo-buffer idea; the required change is that the length of the pseudo-buffer

is incremented for every packet entering and leaving at a given boundary point. Doing this at all boundary points allows one to prevent long term congestion. The problem of course, is that the enforcement mechanism comes into play only after the congestion has occurred, when it can no longer do any good. One can modify the basic mechanism so that it "punishes" users who exceed their allocation, but this does not seem fully satisfactory.

Another approach is to have some explicit arbitration mechanism. The most obvious possibility is to require that each active transmitter hold a special packet called a *token* before being allowed to transmit. A connection may have several tokens, passed among the set of potential transmitters. The key drawback here is that it appears to require a mechanism for reliable token transmission, adding considerable complication to both the network and its use.

A third approach is the brute force one. Perform bandwidth enforcement throughout the network rather than just at the boundary. As we have seen, a bandwidth valve for several hundred channels is simple enough to implement on a single custom integrated circuit, so the cost does not appear excessive. This brute force approach can be refined somewhat. For example, within the network it may suffice to regulate the peak rate on a connection rather than both peak and average. This reduces the amount of data required per connection by a factor of four. We also observe that only connections with fairly high peak rates need be regulated within the network, and at any particular switch module a connection need be regulated only if traffic from that connection can enter the switch module from two or more links. These considerations suggest that a traffic valve capable of regulating the peak rates of 64 connections per link may be sufficient. This, requires a sufficiently modest amount of circuitry that it can likely be incorporated within the packet processors.

Bibliography

[1] Agrawal, D. P. "Testing and Fault Tolerance of Multistage Interconnection Networks," *Computer*, 4/82.

[2] Batcher, K. E. "Sorting Networks and Their Applications," *Proceedings of the Spring Joint Computer Conference*, 1968, 307–314.

[3] Beckner, M. W., T. T. Lee, S. E. Minzer. "A Protocol and Prototype for Broadband Subscriber Access to ISDNs," *International Switching Symposium*, 3/87.

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

[5] Bubenik, Richard and Jonathan S. Turner. "Performance of a Broadcast Packet Switch." Washington University Computer Science Department, WUCS-86-10, 6/3/86.

[6] Coudreuse, J. P. and M. Servel. "Asynchronous Time-Division Techniques: An Experimental Packet Network Integrating Videocommunication," *Proceedings of the International Switching Symposium*, 1984.

[7] Dias, D. M. and J. R. Jump, "Packet Switching Interconnection Networks For Modular Systems," *Computer*, vol. 14, no. 12, 12/81, 43-53

[8] Dias, D. M. and J. R. Jump, "Analysis and Simulation of Buffered Delta Networks," *IEEE Transactions on Computers*, vol. C-30, no. 4, 4/81, 273-282

[9] Dias, D. M. and Manoj Kumar. "Packet Switching in $N \log N$ Multistage Networks," *Proceedings of Globecom 84*, 12/84, 114–120.

[10] Dieudonne, M. and M. Quinquis. "Switching Techniques Review for Asynchronous Time Division Multiplexing," *International Switching Symposium*, 3/87.

[11] Feng, Tse-yun and Chuan-lin Wu, "Fault-Diagnosis for a Class of Multistage Interconnection Networks," *EEE Transactions on Computers*, vol. c-30, no. 10, 10/83.

[12] Feng, Tse-yun. "A Survey of Interconnection Networks," *Computer*, vol. 14, no. 12, 12/83, 12–30.

[13] Franklin, Mark A. "VLSI Performance Comparison of Banyan and Crossbar Communications Networks," *IEEE Transactions on Computers*, vol. C-30, no. 4, 4/81, 283-290.

[14] Gilbert, E. N. and H.O. Pollak. "Steiner Minimal Trees." *SIAM J. Appl. Math.*, 16(1):1–29, 1968.

[15] Giorcelli, S., C. Demichelis, G. Giandonato and R. Melen. "Experimenting with Fast Packet Switching Techniques in First Generation ISDN Environment," *International Switching Symposium*, 3/87.

[16] Goke, L. R. and G. J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems," *Proceedings of the 6th Annual Symposium on Computer Architecture*, 4/79, 182-187.

[17] Gonet, P., P. Adam, J. P. Coudreuse. "Asynchronous Time-Division Switching: the Way to Flexible Broadband Communication Networks," *Proceedings of the International Zurich Seminar on Digital Communication*, 3/86, 141–148.

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

[19] Hayward, G., L. Linnell, D. Mahoney and L. Smoot. "A Broadband Local Access System Using Emerging Technology Components," *International Switching Symposium*, 3/87.

[20] Hoberecht, William L. "A Layered Network Protocol for Packet Voice and Data Integration," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1006–1013.

[21] Huang, Alan and Scott Knauer. "Starlite: a Wideband Digital Switch," *Proceedings of Globecom 84*, 12/84, 121–125.

[22] Huang, Alan. "Distributed Prioritized Concentrator," U.S. Patent 4,472,801, 1984.

[23] Huang, Alan and Scott Knauer. "Wideband Digital Switching Network," U.S. Patent 4,542,497, 1985.

[24] Jaffe, J. M. "Distributed multi-destination routing: the constraints of local information." *SIAM J. Comput*, 14(4):875–88, November 85.

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

[26] Kermani, P. and L. Kleinrock. "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, vol. 3, 1979, 267–286.

[27] Kou, L., G. Markowsky, and L. Berman. "A fast algorithm for Steiner trees." *Acta Informatica*, 15:141–5, 81.

[28] Kruskal, C. P. and M. Snir. "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Transactions on Computers*, vol. C-32, no. 12, 12/83, 1091–1098.

[29] Lea, Chin-tau. "The Load-Sharing Banyan Network," *IEEE Transactions on Computers*, 12/86.

[30] Karol, M. J., M. G. Hluchyj and S. P. Morgan. "Input vs. Output Queueing on a Space-Division Packet Switch," *Proceedings of Globecom*, 12/86.

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

[32] Kulzer, John J. and Warren A. Montgomery. "Statistical Switching Architectures for Future Services," *Proceedings of the International Switching Symposium*, 5/84.

[33] Miller, R. E. and J. W. Thatcher. *Complexity of Computer Computations*, chapter "Reducibility Among Combinatorial Problems," by R.M. Karp, pages 85–103. Plenum Press, 1972.

[34] Montgomery, Warren A. "Techniques for Packet Voice Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1022–1028.

[35] Muise, R. W., T. J. Schonfeld and G. H. Zimmerman III. "Experiments in Wideband Packet Technology," *Proceedings of the International Zurich Seminar on Digital Communication*, 3/86, 135–139.

[36] Patel, J. H. "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Transactions on Computers*, vol. C-30, no. 10, 10/81, 771-780.

[37] Rayward-Smith, V. J. "The Computation of Nearly Minimal Steiner Trees in Graphs," *International Journal of Math. Ed. Sci. Tech.*, 14(1):15–23, 1983.

[38] Rettberg, R., C. Wyman, D. Hunt, M. Hoffman, P. Carvey, B. Hyde, W. Clark and M. Kraley. "Development of a Voice Funnel System: Design Report," Bolt Beranek and Newman, Report No. 4098, 8/79.

[39] Richards, Gaylord and Frank K. Hwang. "A Two Stage Rearrangeable Broadcast Switching Network," *IEEE Transactions on Communications*, 10/85.

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

[41] Sincoskie, W. D. "Transparent Interconnection of Broadcast Networks," *Proceedings of the International Zurich Seminar on Digital Communication*, 3/86, 131–134.

[42] Staehler, R. E., J. J. Mansell, E. Messerli, G. W. R. Luderer, A. K. Vaidya. "Wideband Packet Technology for Switching Systems," *International Switching Symposium*, 3/87.

[43] Stroustrup, Bjarne, "The C++ Programming Language," Addison-Wesley, 1986.

[44] Takeuchi, Takao, Hiroshi Suzuki, Shin-ichiro Hayano, Hiroki Niwa and Takehiko Yamaguchi. "An Experimental Synchronous Composite Packet Switching System," *Proceedings of the International Zurich Seminar on Digital Communication*, 3/86, 149–153.

[45] Turner, Jonathan S. and Leonard F. Wyatt. "A Packet Network Architecture for Integrated Services," *Proceedings of Globecom 83*, 11/83, 45–50.

[46] Turner, Jonathan S. "Fast Packet Switching System," United States Patent #4,494,230, 1/15/85.

[47] Turner, Jonathan S. "Design of an Integrated Services *Packet* Network," *Proceedings of the Ninth ACM Data Communications Symposium*, 9/85, pages 124–133.

[48] Turner, Jonathan S. and L. F. Wyatt, "Alternate Paths in a Self-Routing Packet Switching Network," United States Patent #4,550,397, 10/29/85.

[49] Turner, Jonathan S. "Design of a Broadcast Packet Network," *Proceedings of Infocom*, 4/86.

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

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

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

[53] Turner, Jonathan S. "The Challenge of Multipoint Communication," Washington University Computer Science Department, WUCS-87-6. Also, to appear in *Proceedings of the ITC Seminar on Traffic Engineering for ISDN Design and Planning*, 5/87.

[54] Turner, Jonathan S, "On the Performance of Packet Switched Interconnection Networks in Communications Systems," Washington University Computer Science Department technical report, in preparation.

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

[56] Wirsching, J. E. and T. Kishi, "Conet: A Connection Network Model," *IEEE Transactions on Computers*, vol. C-30, 4/81.

[57] Yeh, Y. S., M. G. Hluchyj and A. S. Acampora. "The Knockout Switch: a Simple Modular Architecture for High Performance Packet Switching," *International Switching Symposium*, 3/87.