Design of a Broadcast Packet Switching Network

JONATHAN S. TURNER, MEMBER, IEEE

Abstract—**Communications applications are becoming more diverse all the time. Current switching technologies, however, have been designed around single applications or small classes of applications. There is a growing need for communications systems that can handle a heterogeneous and dynamically changing mix of applications. This paper provides an overview of a system designed to meet this need. It is based on fiber optic transmission systems and high-performance packet switching and can handle applications ranging from low-speed data to voice to full-rate video. The most novel feature is a flexible multipoint connection capability suitable for broadcast and conferencing applications. The paper describes the architecture of a switching system that can be used to support this network.**

I. INTRODUCTION

THIS paper proposes a design for a high-performance packet switching network that can be used to provide voice, data, and video communication on a large scale. A novel feature of the system is its flexible multipoint connection capability, which makes it suitable for a wide range of applications, including commercial television distribution and voice/video teleconferencing. The basic switching capability is provided by a high-performance packet switch called a *switch module* (SM) which terminates a set of fiber optic communications links (FOL), each operating at a speed of 100 Mbits/s. The SM is designed as a modular component that can be used to construct systems ranging from a medium-sized business system to a large packet switching exchange, capable of providing service to over 200 000 users. The paper provides a high level description of the network as a whole plus a detailed description of the design of the switch module hardware.

The proposed system was motivated by the observation that while communications applications are becoming increasingly diverse and dynamic, current systems are ill-equipped to handle a heterogeneous and changing application mix. This is because they have been designed around specific applications and tailored to those applications so closely that adding new applications is usually very difficult. The system described here was designed to handle a wide class of applications ranging from low-speed data to full-rate video with equal ease. Its most novel feature is its ability to handle multipoint connections. This feature facilitates a variety of new applications that cannot be provided effectively on conventional point-to-point networks. This research is an outgrowth of earlier work that the author and colleagues did on the design of large-scale packet switching systems for voice and data at Bell Laboratories. Details of that earlier work can be found in references [8], [10], [12], and [14]–[22]. Similar work has been done at Bolt, Beranek, and Newman [13].

This work has been influenced by two very different approaches to communications and switching. The first is circuit switching as practiced in current telephone systems and the second is packet switching as practiced in computer networks. The inherent flexibility of packet switching makes is an obvious candidate for a network designed for diverse applications. On the other hand, current packet switching systems compare poorly to circuit switching systems on the basis of speed, throughput, and cost. We (and others) have observed that this relatively poor showing is not inherent to packet switching but is an artifact of current implementations which are largely software systems running on general purpose computers. Circuit switching systems, on the other hand, have always been designed using special purpose switching networks. To match the speed, throughput, and cost of circuit switching systems in a packet switched network, one must resort to the same techniques—special purpose switching networks implemented using high-speed hardware. A third major influence on this work has been the design of interconnection networks for parallel computers (see [7]). The essential component in our switching network is a buffered banyan network, tailored to the particular needs of a large-scale communications system. Some earlier work has been done on broadcast and multipoint routing of packets in computer networks (see, for example, [3], [4], [6]), but this work has little direct bearing on our research. The Prelude project described in [5] has similar goals to our work. Their system uses a switching technique called asynchronous time-division multiplexing, which is similar in spirit to ours, but differs greatly in details. Huang and Knauer [9] offer a different approach to switching high-speed broadcast channels.

The major components of the proposed network are identified in Fig. 1. Access to the network is provided through *network interfaces* (NI), which provide concentration, local switching, network protection, and accounting functions. Switching is provided by *packet switches* (PS), which are interconnected using *fiber optic links* (FOL). The PS's and NI's can be configured in a variety of sizes and configurations. A later section gives example configurations of both. FOL speeds in the neighborhood of 100 Mbits/s are achievable with current technology, and this transmission speed is used as a target for the designs presented below.

The network provides three basic communications services.

- *Point-to-Point Channels:* These are two-way channels joining pairs of users. The user requests a channel capable of providing a certain average bandwidth and the network allocates the requested bandwidth to the connection. If the requested bandwidth is not available, the connection is blocked. These connections do not provide flow control or error correction.
- *Datagrams:* These are individual packets, not associated with a preestablished connection. The network makes an effort to deliver them, but does not guarantee delivery.
- *Broadcast Channels:* Any user can set up a broadcast source that other users can then connect to. The average bandwidth of the source must be specified when it is established, and can range from a few bits per second to over a gigabit per second. There is no limit on the number of users that can receive a given broadcast signal. Thus, it is suitable for a variety of applications including commercial television and voice or video conferencing.

Paper approved by the Editor for Communication Switching of the IEEE Communications Society. Manuscript received October 8, 1985; revised July 2, 1987. This work was supported by the National Science Foundation under Grant DCI 8600947, and under Grants from Bell Communications Research, Italtel SIT, and NEC. This paper was presented at INFOCOM '86, Miami, FL, April 1986.

The author is with the Department of Computer Science, Washington University, St. Louis, MO 63130.

IEEE Log Number 8820344.

0090-6778/88/0600-0734$01.00

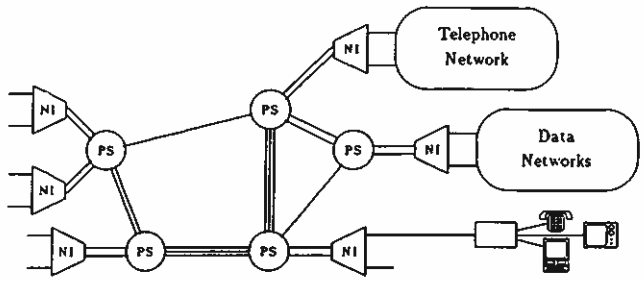

Fig. 1. Network architecture.

Fig. 2 illustrates the way broadcast channels work. Two broadcast sources are shown at the top of the figure. At various points in the network, the signals are split into multiple copies which are ultimately delivered to the appropriate users. Thus, each broadcast source and its associated connections induce a tree in the network. When a user disconnects from a broadcast channel, the corresponding branch of the tree is removed. When a user requests connection to a channel, the network attempts to find a nearby branch of the broadcast tree and connects the user at the nearest point. Thus, the broadcast trees grow and shrink as usage patterns change, but for widely distributed channels, one can expect the bulk of the activity to be concentrated near the leaves.

A special-purpose switching fabric is used to support the broadcast service. This switch fabric splits a broadcast channel into multiple parts as needed. The switching fabric also allows a set of communications links to be treated as a group with traffic distributed over all links in the group. This makes it possible for the network to provide communications channels that require more bandwidth than is available on a single FOL.

The network is predominantly connection-oriented, allowing bandwidth allocation on a per connection basis. Because the network is based on packet switching, the user consumes bandwidth only when actively sending information. This means that the instantaneous bandwidth usage varies over time. In order to permit the network to allocate bandwidth appropriately, users must specify their bandwidth requirements when a connection is established. The bandwidth specification must include average bandwidth requirements plus a measure of the burstiness of the user's application. Given such a specification, the network can allocate appropriate resources and can guarantee the availability of those resources for the duration of a connection, ensuring users adequate performance for their needs. (A detailed treatment of the bandwidth specification and allocation mechanisms is beyond the scope of this paper.)

Three mechanisms for congestion control are proposed. The primary one is connection blocking—new connections are refused if adequate bandwidth is not available. An enforcement mechanism at the NI's ensures that users do not exceed their allocated bandwidth. The second mechanism is a system of user-specified packet priorities. During periods of heavy load, low-priority packets are preferentially discarded to reduce congestion. This mechanism can be very effective if low-priority packets constitute a significant portion (say > 10 percent) of the total network traffic. The third mechanism is a congestion control field within each packet which can be modified by the network to inform the users of internal congestion. Users are expected to respond to this by temporarily slowing down their rate of transmission. Enforcement is provided at the NI's. Note that the first congestion control mechanism operates on a fairly long time scale, the second on a very short time scale, and the third on an intermediate time scale.

The system's communications protocols have been designed to facilitate the implementation of high-performance hardware protocol processors and to exploit the high-speed and low-

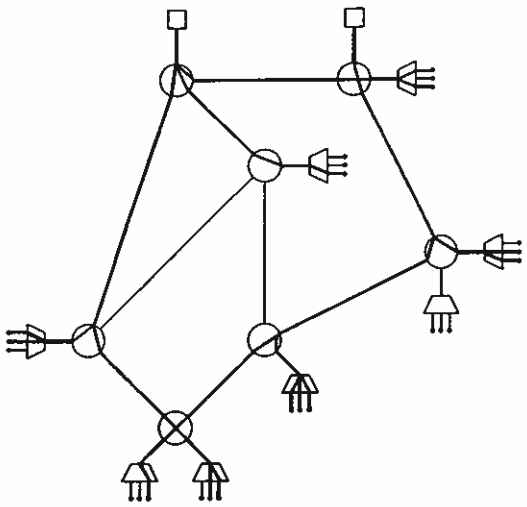

Fig. 2. Broadcast trees.

Fig. 3. Protocol structure.

error rates of the fiber optic links. The protocols are divided into three levels—the *link level*, the *network level*, and the *user level*. The interrelationships among the different levels are indicated in Fig. 3. There is a single link level protocol, two network level protocols (one for connection-based communication and one for datagrams), and a variety of user level protocols. The user level protocols are not discussed in detail here, but the intention is that these protocols would be application-dependent and built on top of one of the two network level protocols. One of these would be a telephony protocol. Another might be an internet protocol such as the DARPA IP to facilitate communication among different data networks. Still another might be a connection-oriented internet protocol.

There are essentially three packet formats of interest to the network—*connection control packets, data transfer packets,* and *datagram packets.* These packet formats are illustrated in Fig. 4. All packets are the same length; this substantially simplifies the design of high-performance protocol and switching hardware.

Aside: Fixed length packets immediately raise the question of what packet length is most appropriate. The answer is necessarily a compromise, since no one length is optimal for all applications. Applications like voice are best served by fairly short packets; a good choice is a packet length long enough for 10–20 ms worth of digitally encoded speech. Assuming 64 kbit/s coding, this translates to 640–1280 bits of speech per packet. High-speed applications like video are best served by much larger packets. Datagrams, which require a fairly high overhead per packet are also best served by long packets. Long packets also make it easier to design high-speed switching systems, since they give per packet control operations a relatively long time to take place; because data and control operations take place on very different time scales, it becomes easier to speed up the data path. Given these conflicting requirements, how does one strike a compromise? The choice proposed here is to use packets with a length of about 600 bytes. This is long enough to carry 512 bytes of data along with a generous allocation for the overhead required by common end-to-end protocols. Applications like voice would not be able to use such long packets efficiently, but would have to transmit packets that were largely empty. While this would

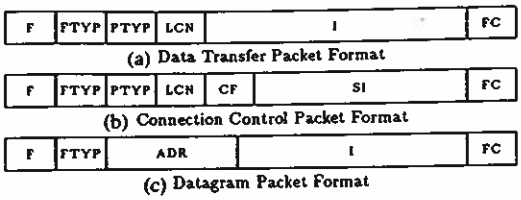

Fig. 4. Packet formats.

lead to inefficient bandwidth use for voice applications, the efficiency of the network as a whole could still be very good, since we expect that most of the packets carried by the network would belong to high-speed connections. In fact, high-speed networks such as we are proposing only make sense if this expectation is true. The decision on short versus long packets boils down to a choice between what is best in the short term versus what is best in the long term. Short packets are best for the mix of traffic that is typical today. Long packets are best for the higher speed mix of traffic expected in the future and will better enable us to speed up switching systems in order to match the speed increases in transmission. The choice proposed here is a compromise; while not optimal in the short term, it is acceptable and will become increasingly attractive as high-speed applications grow and technology pushes us to higher speed switching.

Link Level Protocol: The link level protocol provides frame delimiting, link transparency, error detection, and congestion control, but not error correction. There are two fields used by the link level protocol, the frame type field (FTYP) and the frame check field (FC). The FTYP field is one byte long and identifies the frame as a test frame, a datagram, or a frame belonging to a connection. It also has a subfield containing a user-specified priority. The network preferentially discards low-priority frames when internal network buffers are in danger of overflowing. The frame check field (FC) uses a two-byte cyclic redundancy code to detect errors in the frame. Frames are separated by flags (F). Bit stuffing is not used—the flag may appear within the body of the packet. Since the packet length is fixed, frame synchronization is straightforward. Frames with errors can be discarded.

Aside: Packets with errors are discarded, two approaches are possible: one is to discard packets with errors in the header and the other is to discard packets with errors in any field. The first approach can lead to a somewhat smaller packet loss rate, but is complicated by the fact that there are several different packet formats to contend with. The low-error rates typical of fiber optic transmission facilities lead to low packet loss rates even if packets are discarded on nonheader errors. This observation, together with its relative simplicity leads us to favor discarding packets if an error is detected in any field.

Network Level Protocols: There are two protocols at the network level, a datagram protocol and a connection protocol. Packets using the connection protocol contain a one byte packet type field (PTYP) and a two-byte logical channel number field (LCN). The packet type field identifies each packet as either a data packet or a control packet and contains a congestion control subfield used to inform the NI's and users of internal network congestion. The logical channel number field identifies which connection a packet belongs to. Data packets contain user information. Control packets are used to establish and control connections. Datagram packets contain an eight-byte address field which defines a hierarchical address space organized on a geographical basis, to permit routing of connections and datagrams based on local knowledge of network topology.

The remainder of the paper describes the components that make up the network. Section II describes the design of the switch modules. Section III shows how the SM's can be used to construct network interfaces and packet switches. Section IV offers some conclusions and outlines extensions to the SM that provide additional capabilities.

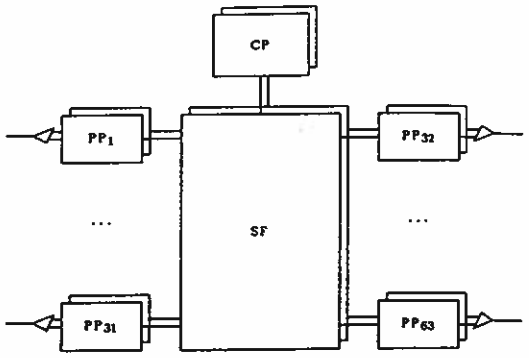

Fig. 5. Switch module.

II. Design of the Switch Module

The overall structure of the switch module (SM) is shown in Fig. 5. The SM is designed as a building block to be used in constructing larger systems.

The *switching fabric* (SF) is the heart of the SM. It has a highly parallel organization and provides multiple paths for packets to ensure good performance. The SF also replicates broadcast packets as necessary and can support distribution of packets across a set of outgoing FOL's. We have chosen to dimension the SF with 64 input and output ports; this size is large enough to provide adequate flexibility in configuring large systems, while being small enough to make high-speed synchronous operation easy to achieve and to support high reliability with a simple duplication scheme. Other choices are of course possible.

The *packet processors* (PP) perform the link level protocol functions, including the determination of how each packet is routed. This includes routing connection control packets to the connection processor and datagram packets to one of several datagram routers (not shown).

The *connection processor* (CP), is responsible for establishing connections, including both point-to-point and broadcast connections. To do this, it exchanges control packets with CP's in neighboring SM's and controls the actions of the PP's and SF by writing information in their internal control tables. It also performs a variety of administrative and maintenance functions.

The *datagram routers* (DR), which are not shown in the figure, are special-purpose devices used to route datagrams. The number of DR's can be engineered to suit the traffic. Each one occupies a port on the SF, replacing one PP. Since the vast majority of the traffic is expected to be connection-oriented, a few DR's (≤ 5) should suffice for most situations.

The duplication scheme shown in the figure is designed to ensure the highly reliable operation required in a large scale public communications network. The SM has two independent planes, each containing a CP, SF, and full complement of PP's. The normal mode of operation is for one plane to be active with the other acting as a "hot standby." A memory update channel is provided between the two CP's so that the memory of the standby CP tracks that of the active one. The standby CP executes an audit program designed to ensure that the tables in the PP's on the standby side also track those in the active side. Optical switches control which plane is connected to the FOL's. Note that the entire plane is switched as a unit. A system reconfiguration would normally cause a loss of all packets in the active switch plane before reconfiguration, but would cause no lost connections.

When a packet enters the SM, it is reformatted by the addition of three new fields, the routing field, the control field, and the source field. The *control* field (C) is one byte long

and identifies different packet types within an SM, including various kinds of control packets. The *source* field (SRC) is one byte long and identifies the origin of the packet within the SM. The routing field (RF) is four bytes long and contains information needed to route the packet through the switch fabric. It has three subfields. The routing control subfield (RC) determines the type of routing the packet is to receive—the most important distinction is between broadcast packets and others. The interpretation of the remaining two subfields depends on the RC value. For packets belonging to point-to-point connections, the RF contains a *group number* subfield (GN) and a *logical channel number* subfield (LCN). To simplify the presentation, a complete discussion of group numbers will be deferred until later—for the present, it will suffice to identify group numbers with link numbers. The LCN subfield contains the LCN that the packet will carry when it leaves the SM. For broadcast packets, the RF contains a *fanout* subfield (FAN) and a *broadcast channel number* (BCN). The FAN subfield is equal to the number of outgoing FOL's that require a copy of this broadcast channel. The BCN is used to distinguish this channel from all other broadcast channels within the SM.

Logical channel translation is the process used to determine how to route a packet belonging to a connection through the SM. Each PP contains a *logical channel translation table* (LCXT) used for this purpose. When a packet is received by a PP, its LCN is used to index the LCXT. The selected entry is copied into the routing field of the packet. The LCXT contains 4096 entries of four bytes each. The entries in the LCXT's are maintained by the CP, which typically writes a new entry each time a new connection is established.

A. Packet Processor

The structure of the packet processor is shown in Fig. 6; it 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. (These buffer capacities were selected based an $M/D/1$ queueing analysis and may change as a result of more detailed analyses.) The four buffers require a total of about 240 kbits of memory.

The *logical channel translation table* (LCXT) is a dual port memory that implements a table with 4096 entries of four bytes each. It can be read by the output circuit and written by the input circuit.

The *receive circuit* (RCV) converts the incoming optical signal to an electrical signal in eight-bit parallel format, synchronizes it to the local clock, discards frames containing errors, routes test packets to the LTB, and other packets to the RCB.

The *output circuit* (OUT) takes packets from the RCB, adds the routing, control, and source fields to the front of the packet as described above and sends the packets on to the switch fabric. It also processes test packets from the STB.

The *input circuit* (IN) routes data packets to the XMB, removing the routing, control, and source fields in the process. It also routes switch test packets to the STB, and updates the LCXT memory in response to LCXT write packets.

The *transmit circuit* (XMIT) takes packets from the XMB, adds the flag field and converts from the eight-bit parallel format to an optical signal. It also processes test packets from the LTB.

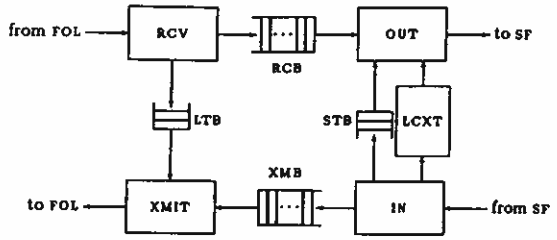

Fig. 6. Packet processor.

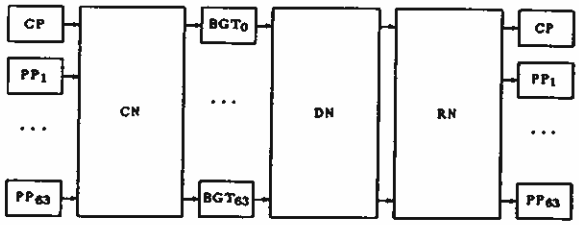

Fig. 7. Switch fabric.

B. Switch Fabric

A block diagram of the switch fabric (SF) is given in Fig. 7. It contains four major components, a *copy network*, a set of *broadcast and group translators*, a *distribution network*, and a *routing network*. The SF runs at a clock rate of 25 Mbits/s and has eight-bit wide internal data paths. This gives an effective data rate of 200 Mbits/s on the internal data paths, or two times the speed of the FOL's. An occupancy of 80 percent on the FOL's translates to a 40 percent occupancy on the internal data paths, which keeps contention and delay low. Simulation studies described in [1] show a delay of less than 100 μs for the entire switch fabric under conditions of heavy load.

When a broadcast packet having k destinations passes through the copy network (CN), it is replicated so that k copies of that packet emerge from the CN. Point-to-point packets pass through the CN without change.

The broadcast and group translators (BGT) perform two translation functions. First, for each arriving broadcast packet they determine the proper outgoing GN (group number) and LCN. Then, they translate the GN to an LN (link number). Discussion of this latter translation is postponed until later in this section. For the moment, we will assume that the GN's and LN's are identical.

The distribution and routing networks (DN, RN) move the packets to the proper outgoing PP. The RN is a straightforward binary routing network and the DN is provided to prevent internal congestion within the RN.

1) Routing Network and Distribution Network: The RN is a 64 × 64 binary routing network. Its structure is illustrated in Fig. 8 which shows a 16 × 16 version. The key property of such networks is that they are *bit addressable*—that is, the route a packet takes through the network is determined by successive bits of its destination address. The figure shows paths from two different input ports to output port 1011. Note that at the first stage the packet is routed out the lower port of the node (corresponding to the first "1" bit of the destination address), at the second stage it is routed out the upper port (corresponding to the "0" bit), and in the third and fourth stages it is routed out the lower ports. The self-routing property is shared by a variety of different interconnection patterns, including the so-called delta, shuffle-exchange, and banyan networks [7].

Buffers are provided at the inputs of each node. Each buffer is capable of holding two complete packets, implying about 20 000 bits of memory per node. (A detailed simulation study

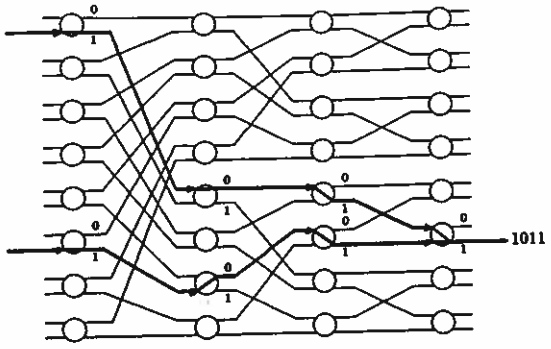

Fig. 8. 16 × 16 binary routing network.

has shown two buffers per input port to be sufficient to obtain the desired throughput. See [1], [2] for details.) A packet may pass through a node without being buffered at all if the desired output port is available when the packet first arrives. Indeed, in a lightly loaded network, a packet can pass through the RN with no buffering at all. In this case, it encounters a delay of just a few clock times in each node.

The data paths between nodes are eight bits wide. In addition, there is a single upstream control lead used to implement a simple flow control mechanism. This prevents loss of packets due to buffer overflows within the fabric. The entire network is operated synchronously, both on a bit basis and a packet basis—that is, all packets entering the first stage do so during the same clock cycle. The clock rate is 25 Mbits/s, giving an effective bandwidth of 200 Mbits/s for each of the paths through the network.

Fig. 9 is a more detailed picture of a single switch node. The switch node consists of two *input controllers* (IC) and two *output controllers* (OC). The IC's determine the routing required for each packet, request the appropriate OC, buffer packets if necessary, and apply upstream flow control to prevent buffer overflow. The OC's arbitrate requests for the outgoing links.

One problem with binary routing networks is that they can become congested in the presence of certain traffic patterns. This is illustrated in Fig. 10, which shows a traffic pattern corresponding to several *communities of interest*. In this pattern, all traffic entering the first four inputs is destined for the first four outputs, all traffic entering the second group of four inputs is destined for the second group of four outputs, and so forth. Note that with this pattern, only one fourth of the links joining the second and third stages are carrying traffic. Thus, if the inputs are heavily loaded, the internal links will be overloaded and traffic will back up. In a 64 × 64 network, there are eight stages and the links between the third and fourth stages can in the worst case be carrying all the traffic on just eight of the 64 links.

The DN solves this problem by evenly distributing packets it receives across all its outputs. It has the same internal structure as the RN, but its nodes ignore the destination addresses on packets and route them alternately to each of their output ports. This strategy is modified if one or both ports is unavailable. In this case, the first port to become available is used. This approach breaks up any communities of interest and makes the combination of the DN and RN robust in the face of pathological traffic patterns.

2) Copy Network and Broadcast Translation: The structure of the copy network (CN) is the same as that of the RN and DN. The CN's function is to make copies of broadcast packets as they pass through. This is illustrated in Fig. 11. The packet entering at left belongs to broadcast channel 36 and is to be sent to seven different links. At the first stage, the packet is routed out the upper port. This is an arbitrary decision—the lower link could have been used at this point. At the second

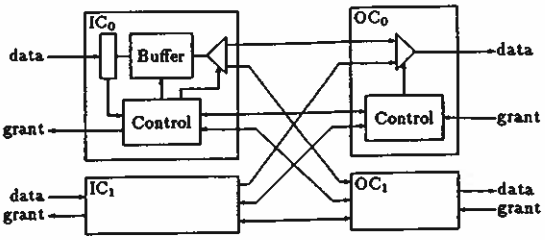

Fig. 9. Switch node.

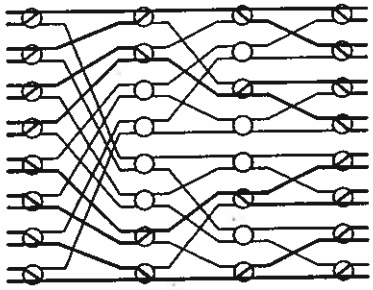

Fig. 10. Congestion in binary routing networks.

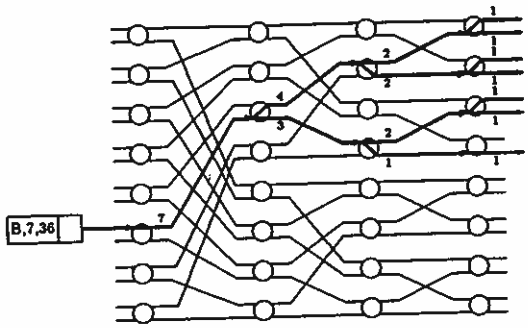

Fig. 11. 16 × 16 copy network.

stage, the packet is sent out on both outgoing links and the fanout fields of the outgoing packets are modified. The upper packet will generate four copies and the lower one, three.

We now describe the CN routing algorithm for broadcast packets. Let BCN and FAN be the broadcast channel field and the fanout field from the packet and let *sn* be the stage number of the node in the switch where stages are numbered from right to left, starting with 0.

- If FAN $> 2^{sn}$ request both output links and when the links are available, simultaneously send the packet out on both.
  - — If BCN is even, the FAN field of the upper packet is set to $\lfloor (\text{FAN} + 1)/2 \rfloor$ and the FAN field of the lower packet is set to $\lfloor \text{FAN}/2 \rfloor$.
  - — If BCN is odd, the FAN field of the upper packet is set to $\lfloor \text{FAN}/2 \rfloor$ and the FAN field of the lower packet is set to $\lfloor (\text{FAN} + 1)/2 \rfloor$.
- If FAN $\leq 2^{sn}$, use the DN algorithm.

Note that this algorithm delays copying packets as long as possible. Another option is to copy the packets early. However, this approach can lead to unbounded congestion in the CN. The late copying algorithm limits the potential for congestion. See [1] for further details.

An important property of the copy algorithm is that given a particular combination of BCN and FAN, one can predict which of the FAN copies can appear at each output of the copy network. Suppose the copies of all broadcast packets were numbered from 0 to FAN − 1 with the indexes increasing as the output port numbers increase. Consider what happens

when a broadcast packet is processed by the copy network. Some CN output ports may receive a copy while others do not, depending on how the arbitrary routing decisions are made. However, if an output port receives a copy, the number of that copy is completely determined. The number of the copy that can appear at a particular output port is called the *broadcast copy index* or BCI for that port. The BCI is a function of the output port number, the FAN field and the low-order bit of the BCN and is denoted bci_k (FAN, BCN). An algorithm to calculate BCI_k (FAN, BCN) appears in Fig. 12.

When the copies of a broadcast packet emerge from the CN, their final destinations are yet to be determined. The principal function of the broadcast and group translators (BGT) is to assign a new routing field to each copy of a broadcast packet in such a way that a copy is received by every PP that is supposed to receive one. This is accomplished by a simple table lookup based on the broadcast channel number (BCN). Each BGT contains a *broadcast translation table* (BTT), used for this purpose. The BTT is indexed by the BCN of the incoming packet and contains four byte entries. When BGT_k receives a copy of a broadcast packet, it replaces the packet's routing field with the contents of BTT_k[BCN].

Note that two BGTs need not have identical entries in their BTTs for a given BCN, since their BCI values may be different. This point is illustrated in Fig. 13 which shows the translation process for two packets copied from a single broadcast packet.

The addition of a new destination to a broadcast channel can radically change the mapping required in the BTT's. In general, when a new destination is added to a particular broadcast channel, the CP must compute a new broadcast copy index tree as described above and use that information to update all the BTT's. Since this can happen frequently, it appears likely to present a substantial computational burden for the CP. Fortunately, there is a simple solution to the problem. It requires that for $0 \leq k \leq 63$, BGT_k contains a table that it can use to compute BCI_k (FAN, BCN). Since, FAN ≤ 64 and only the least significant bit of the BCN is needed to compute BCI, the table has 128 entries, each one byte long. Note that this table is static and is different for each BGT. Now, when the CP wishes to update the BTT's for a particular broadcast channel, it sends a control packet of the form shown in Fig. 14.

The CN replicates the packet so that each BGT receives a copy. When BGT_k receives its copy, it extracts FAN and BCN from the packet, then uses its lookup table to compute $j = BCI_k$ (FAN, BCN) and finally copies RF_j from the packet to BTT_k [BCN].

Using this scheme, the CP keeps a copy of the BTT_*update* packet in its memory. To add a new destination, it increments the FAN field, adds a new RF to the end of the packet and sends it out to the SF. To remove a destination, it decrements the FAN field, removes the proper RF from the packet and sends it out.

3) Link Groups: A small addition to the mechanisms described above allows links to be collected together into *link groups*. The links in a group must have the same destination. Traffic for a link group is evenly distributed over all of its links. This allows dynamic load distribution across a set of links permitting higher utilization. It also allows the network to support channels that require more bandwidth than any one FOL can provide.

When a BGT receives a point-to-point packet, the packet contains a group number (GN) in its routing field. Each BGT contains a table, called the group translation table (GTT), which it uses to translate a GN to a link number (LN). The GTT identifies the links in each group and also contains a pointer to a particular link in the group. Every time a packet for a particular group is handled, this pointer is advanced to the next link in the group, so that each BGT distributes traffic evenly across the group. Broadcast packets are handled similarly, but they must first go through the broadcast translation process described earlier. The organization of the GTT is shown in Fig. 15, which illustrates the entire translation process for a broadcast packet. Note that the GTT is a fairly static table—it changes only when groups are reconfigured, not on a per-connection basis.

```
integer function bci_k(FAN, BCN)
    integer s, f;
    Let the binary representation of k be b_{n-1} ... b_1 b_0,
      where n is number of stages in copy network.
    s := 0; f := FAN;
    p := the least significant bit of BCN.
    for i := n - 1 downto 0 →
        if f > 2^i →
            if b_i = 0 →
                f := ⌊(f + 1 - p)/2⌋;
            | b_i = 1 →
                s := s + ⌊(f + 1 - p)/2⌋;
                f := ⌊(f + p)/2⌋;
            fi;
        fi;
    rof;
    return s;
end;
```

Fig. 12. Computation of broadcast copy index.

One consequence of link groups is that the LCXT's for all links in the same group must be identical. Consequently, on every connection the CP must update the LCXT's for all links in the group. This can be handled easily by sending a broadcast LCXT update packet.

C. Implementation Details

This section is an an attempt to "size up" the switch modules. While the estimates given here are rough, they give a fairly accurate picture of the complexity of the SM's.

The switch nodes can best be implemented using custom integrated circuits. Each of the nodes that makes up the switch fabric consists of about 20 000 bits of memory plus some control circuitry. The memory is the dominant component, so we can estimate the size of a chip needed to implement a switch node by looking at the area required for the memory. The memory in the switch nodes is not extremely large, but it must be fast. Perhaps the simplest way to implement it is using a large dynamic shift register. A single bit of shift register memory can be implemented in 200 μm^2, assuming a 1 μm CMOS process, implying that 20 000 bits of memory will consume about 4 mm^2. This suggests that several nodes can be implemented on a single chip. Not surprisingly, pin limitations restrict the size of the chip more severely than chip area. A single node on a chip requires 36 signal pins plus additional pins for power, ground, and clock. Four nodes can be placed on a chip with about 80 pins if they are configured as a 4×4 switch. Actually, a better way to implement a 4×4 switch is to do it as a single larger node, rather than four small nodes—this structure requires half as much memory and is about twice as fast. It is convenient to have two chip types—one chip contains two independent 2×2 nodes and the other contains a single 4×4 node. Each of these can be implemented in an 80-pin package, preferably a high-density package such as a pin grid array.

The packet processors are more complex than the switch nodes. Each requires about 400 000 bits of memory if one includes all the packet buffers and the logical channel translation table. The largest item is the transmit buffer, which can be implemented as eight 128×150 memory arrays. The

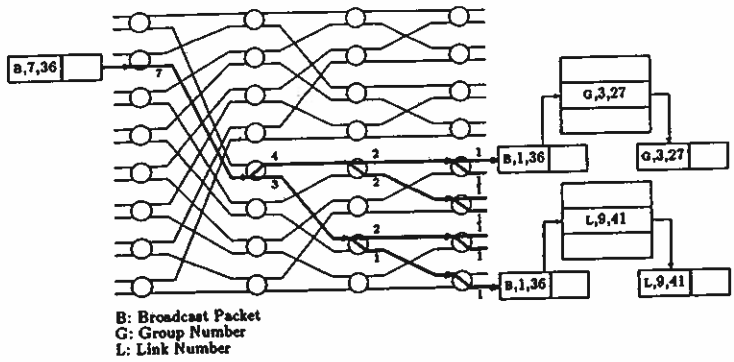

Fig. 13. Example of broadcast processing.

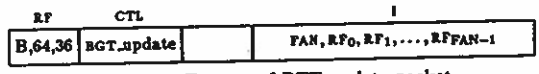

Fig. 14. Format of BTT update packet.

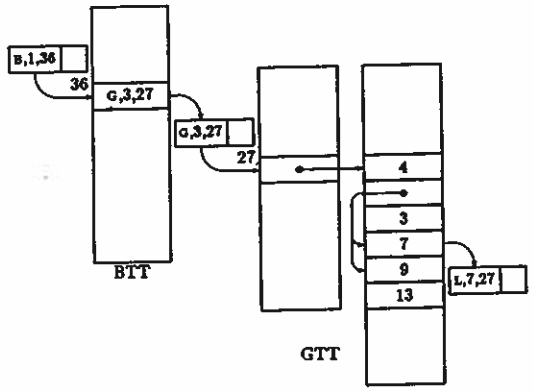

Fig. 15. Broadcast and group translation.

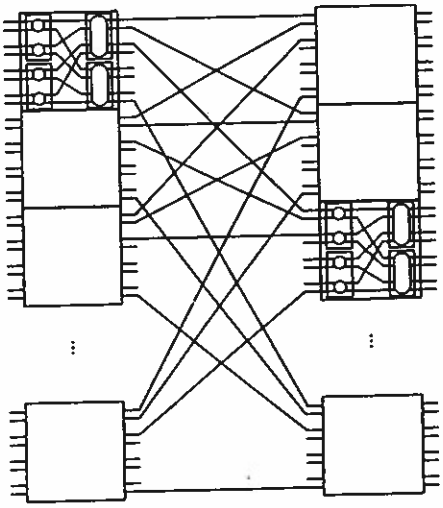

Fig. 16. 64 × 64 network.

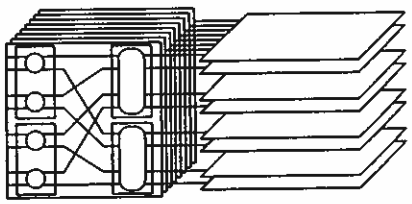

Fig. 17. Arrangement of circuit cards for a network.

cycle time of these arrays is not a critical performance factor—each access reads or writes 150 bits, leading to one access every 3 μs. Thus, fairly dense memory structures can be used. A three transistor memory cell can be implemented in about 100 μm^2 assuming a one micron technology, implying that the memory cells required by the transmit buffer will consume about 16 mm^2. The addition of the supporting circuitry required by the memory arrays could bring the total up to about 30 mm^2. The other packet buffers can be implemented in a similar fashion. Together, they contain less than two thirds the memory of the transmit buffer, but we will estimate that they consume another 30 mm^2. The LCXT contains about 150 000 bits of memory organized in nine 128 × 128 arrays. This adds about another 30 mm^2 to the chip area. If we assume (conservatively) that the memory consumes 75 percent of the chip area required to implement the PP, we arrive at an estimate of 1.2 cm^2 for the entire PP. A three chip implementation appears feasible, with the first chip containing the RCV, XMIT, and STB blocks, the second containing the RCB and XMB, and the third containing the IN, OUT, STB, and LCXT blocks.

The broadcast and group translators each contain several tables. The BCI table, used to compute a broadcast channel index contains 128 bytes, the BTT contains 4096 bytes (assuming 1024 unique BCN's in an SM), and the GTT 320 bytes. The total memory requirement is thus under 40 000 bits. Thus, two can fit comfortably on a single chip.

With these estimates in hand, we can consider how an entire SM might be put together. A convenient way to implement a 64 × 64 binary routing network is to subdivide it into 8 × 8 subnetworks as shown in Fig. 16. Each of the 8 × 8 subnets can be constructed from four 2 × 2 nodes and two 4 × 4 nodes for a total of four chips, and can be implemented on a single board with 144 signal leads. These boards can then be arranged into two orthogonally positioned ranks to give a compact implementation of the 64 × 64 network. This structure is shown in Fig. 17. What makes this arrangement particularly attractive is that the board-to-board signal paths can be kept very short.

An SM, however, contains three such networks. We can employ the same idea in the SM using the arrangement shown in Fig. 18. Here, there are four groups of eight-circuit boards each. The boards at the bottom center contain the first half of the CN. The boards at the right contain the second half of the CN, the BGT's, and the first half of the DN. The boards at the top center contain the second half of the DN and the first half of the RN. The boards at the left contain the second half of the RN and the PP's. The most densely populated boards are the ones at the left, each of which contains four switch node chips and 24 PP chips. Assuming high-density packaging in which chips are mounted in 1 in square pin grid arrays on multilayer ceramic circuit boards, this entire assembly could fit in a volume of roughly 6 in × 12 in × 20 in. A complete SM would contain two of these assemblies plus the CP's. Each CP

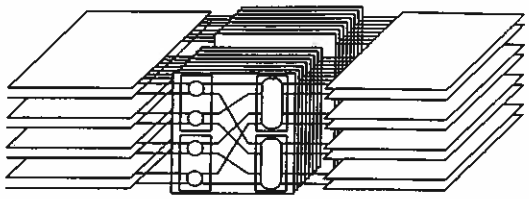

Fig. 18. Switch module packaging.

contains a 32-bit microprocessor with perhaps 2 Mbytes of memory, an interface to the switch fabric, and a single terminal interface. Hence, it could be implemented on three- or four-circuit cards and four complete SM's could be packaged in a single 30 in wide cabinet.

III. Design of Network Components

The switch module is designed as a component that can be used to build larger systems. The decision to build large systems from small components can be criticized on the grounds that the total hardware required can be substantially larger and the performance lower than if a monolithic structure is used. There are several strong arguments in favor of the modular approach, 1) a system constructed out of small modules is easier to design and implement, 2) modular systems can grow over a large range of sizes by adding modules, and 3) manufacturing economics favor systems that contain a large number of identical components.

Modular systems are easier to design because the modules lead to a natural separation of local concerns from global ones. Modules can be defined by their external interfaces and treated as "black boxes" by their environment. Once this separation has been made explicit by a set of module specifications, the design of the individual modules is relatively straightforward. The size of the modules is a central issue. In the case of the SM's, larger modules would have been preferable for constructing larger systems. The decision to use a fairly small module was motivated by three considerations. First, it allows a single processor of moderate proportions to handle all the connection processing, avoiding a number of difficult problems that must be faced in larger systems. Second, it allows a very simple approach to reliability and maintenance—to recover from a hard failure in the active side, you simply switch to the other side. With small enough modules, failure rates are low enough that there is no need to resort to more elaborate recovery algorithms. The small size also simplifies repair in the field—once it has been determined that a particular SM side is faulty, it can be replaced as a unit. Third, hardware timing is easier to handle in a system of modest physical dimensions. The SM's are small enough that it is reasonable to operate them as synchronous systems. Asynchronous operation need only be considered at the interface between the PP's and the FOL's.

Our first example of a larger system component constructed from SM's is the network interface (NI), shown in Fig. 19. A single NI contains 15 SM's, five of which are termed back-end SM's (BSM) and 10 of which are termed front-end SM's (FSM's). Each FSM has six FOL's connecting it to each of the BSMs and each BSM has six FOLs connecting it to each of the FSM's. The NI has 330 FOL's that can be used to connect to other system components. Typically, 30 would be used to connect to the NI's host PS with the remainder going to users. The actual ratio of downstream links to upstream links is completely flexible—a ratio of about 10:1 appears reasonable, since most of the downstream bandwidth is likely to be consumed by a few popular broadcast channels.

SM's can also be used to build a packet switch (PS) as shown in Fig. 20. The PS shown can have as few as one BSM and two FSM's or as many as 30 BSM's and 60 FSM's. In the largest configuration, it terminates 1800 FOL's on its FSM's.

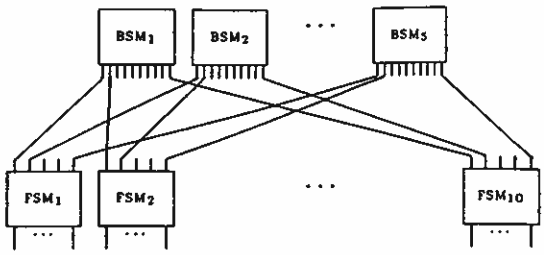

Fig. 19. Network interface structure.

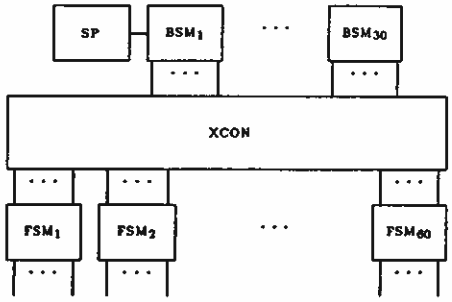

Fig. 20. Small packet switch.

If 1500 FOL's are used to connect to NI's, such a PS can support 15 000 users. This leaves 300 FOL's for connecting to other PS's. Again, the ratio of downstream to upstream links is not limited by the architecture and can be adjusted to suit actual needs—the ratio chosen here appears reasonable. Each PS contains a supervisory processor (SP), which provides office administration and craft interface functions. The SP is a fairly large processor with its own disk subsystem and possibly other peripherals. It does no processing of individual connections, but does supervise the operation of the SM's through its control of their routing tables.

A PS must be capable of growing across a range of sizes. In the smallest configuration, it would have 30 FOL's joining each FSM to the one BSM. In the largest configuration, it would have one FOL joining each FSM to each BSM. If the SM's were joined directly to one another, growth of a PS would require manual recabling. This is avoided by the cross connect (XCON), which can be thought of as an 1800 × 1800 cross-bar switch. Fortunately, the XCON does not require all the functionality of a full cross bar and can implemented in much less expensive way. The structure of the XCON is illustrated in Fig. 21, which shows a small version, appropriate for SMs having just sixteen FOL's. Such an XCON can be used to construct a PS with up to eight BSM's and 16 FSM's. The XCON has a planar structure with a BSM side and an FSM side. The BSM side is organized in rows—all FOL's from a particular BSM terminate on the same row. The FSM side is organized in columns—all FOL's from a particular FSM terminate on the same column. Each of the circles at the intersection of a row and column represents a switching element. These switching elements each have four leads referred to as x^-, x^+, z^-, and z^+. The z^- and z^+ leads are arranged perpendicular to the plane of the XCON so that they can be connected to the links from the SM's. The x^- and x^+ leads are arranged in the plane of the SM in a diagonal direction. The switching elements can be switched in four different ways: (z^-, z^+), (z^-, x^+), (x^-, x^+), and (x^-, z^+). In a fully grown configuration, all the rows on the BSM side are occupied, as are all the columns on the FSM side and all the switching elements are set to (z^-, z^+) position. Hence, all the connections pass straight through the XCON—so each BSM/FSM pair is joined by one FOL. In the smallest configuration, the top row on the BSM side is occupied, along with the first and ninth columns on the FSM side. The switching elements

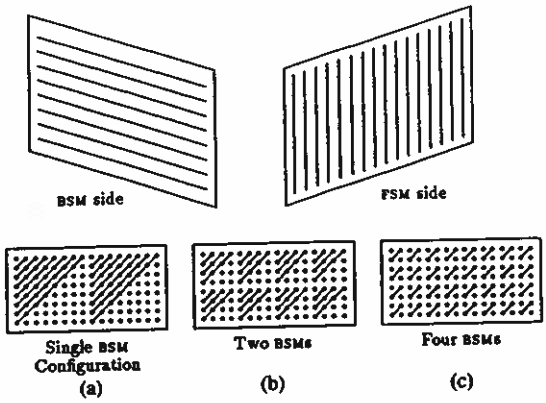

Fig. 21. Cross connect.

are set to provide eight paths connecting the BSM to each of the FSM's, as illustrated in Fig. 21(a). To illustrate, the signal arriving on the FOL connected to the switch element in the first column, fourth row is switched up and to the right by that element, which is set to (z^-, x^+), passes through two elements set to (x^-, x^+) and is then switched out the other side of the XCON by the element in the fourth column, first row, which is set to (x^-, z^+). Fig. 21(b) shows a configuration in which two BSM's are connected to four FSM's, with four FOL's joining each pair, and Fig. 21(c) shows a configuration with four BSM's and eight FSM's. The complexity of an XCON grows linearly with the number of FOL's that connect to it, making it reasonable to construct systems with fairly large XCON's.

Larger packet switches can be constructed by combining modules in a similar way. A PS capable of terminating 27 000 FOL's can be constructed from 900 FSM's and 15 of the smaller packet switches described above, which function as BSM's in that case. If 20 000 FOL's are used to connect to NI's, such a system would serve 200 000 users.

The examples in this section are illustrative of the kinds of systems one can construct using SM's as the basic building blocks and XCON's as the glue used to combine them easily. Many other variants are possible. In particular, a variety of different NI's is desirable, since not all users require the same set of services. An NI that supports say 10 Mbits/s of bandwidth to individual users can be constructed from a single SM supplemented with small front-end concentrators. Such an NI would require 16 FOL's to a PS and would support 750 users, making it substantially less expensive than the one described above.

IV. Summary

It is worth noting that the network described here can support other connection types in addition to point-to-point and broadcast. First, it it possible to include an upstream capability in broadcast channels. No new mechanisms are required in the SM's. The CP must simply set the LCXT entries so that packets received from broadcast recipients get multiplexed onto the link and logical channel going to the broadcast source. Such an upstream capability can be used in a variety of ways—for example, if the channel is being used to broadcast a lecture or other presentation, the upstream capability gives listeners a way to ask questions.

The presence of an upstream capability raises new questions for congestion control. Since the data rates of all the participants add to the total upstream flow, it's not clear that one can control congestion by policies enforced at the NI's. One approach is to make upstream packets low priority, so that they get through the network only if there is sufficient bandwidth available. This may work adequatcly in some situations, but is not a general solution.

The system can also support a multiway broadcast, in which every packet sent by any participant is received by all others. This is a natural sort of connection to use in providing a conference service.

In telephone networks that support conferencing, a new person is added to a conference because of an action initiated by a current participant. In a communications network that supports broadcast television, there is a need for persons not part of a connection to request that they be connected (subject to authorization procedures). Having observed this, one notices that the ability to add on to a connection from "outside" is a generally useful ability. It allows, for example, conferences that users can come and go from as they wish, rather than requiring that they be available when the conference starts. This suggests that every connection should have the potential of adding new participants either from inside or outside, with authorization options used to restrict connection from outside if that is desired.

This paper has described the design of a packet switching network that can support voice, data, and video communication on a large scale. The most novel aspect of the system is its flexible broadcast capability, which is suitable for a range of services including broadcast, television, and conferencing. The main purpose of the paper is to show that such systems are technically feasible.

Acknowledgment

This research is an outgrowth of earlier work that I did while at Bell Laboratories. Many individuals contributed ideas to that earlier work, which continues to influence my thinking—particularly, H. Andrews, D. Creed, M. Dècina, B. Hoberecht, W. Montgomery, and L. Wyatt. Thanks are also due to E. Nussbaum, P. White, and N. Haller of Bell Communications Research for supporting my continued work in this area and for stimulating my interest in broadcast networks.

References

[1] R. Bubenik, "Performance evaluation of a broadcast packet switch," M.S. thesis, Comput. Sci. Dep., Washington Univ., Aug. 1985.

[2] R. Bubenik and J. S. Turner, "Performance of a broadcast packet switch," Comput. Sci. Dep., Washington Univ., Tech. Rep., WUCS-86-10, June 1986.

[3] J.-M. Chang and N. F. Maxemchuk, "Reliable broadcast protocols," *ACM Trans. Comput. Syst.*, vol. 2, no. 3, Aug. 1984.

[4] D. R. Cheriton and S. E. Deering, "Host groups: A multicast extension for datagram internetworks," in *Proc. Ninth Data Commun. Symp.*, Sept. 1985, pp. 172–179.

[5] J. P. Condreuse and M. Servel, "Asynchronous time-division techniques: An experimental packet network integrating videocommunication," presented at Proc. Int. Switching Symp., 1984.

[6] Y. K. Dalal and R. M. Metcalf, "Reverse path forwarding of broadcast packets," *Commun. ACM*, vol. 21, no. 2, pp. 1040–1047, Dec. 1978.

[7] T.-Y. Feng, "A survey of interconnection networks," *Comput.*, vol. 14, no. 12, pp. 12–30, Dec. 1983.

[8] W. L. Hoberecht, "A layered network protocol for packet voice and data integration," *IEEE J. Select. Areas Commun.*, vol. SAC-1, pp. 1006–1013, Dec. 1983.
[9] A. Huang and S. Knauer, "Starlite: A wideband digital switch," in *Proc. GLOBECOM 84*, Dec. 1984, pp. 121–125.
[10] Y.-C. Jenq, "Performance analysis of a packet switch based on a single-buffered banyan network," *IEEE J. Select. Areas Commun.*, vol. SAC-1, Dec. 1983, pp. 1014–1021.
[11] J. J. Kulzer and W. A. Montgomery, "Statistical switching architectures for future services," presented at Proc. Int. Switching Symp., May 1984.
[12] W. A. Montgomery, "Techniques for packet voice synchronization," *IEEE J. Select. Areas Commun.*, vol. SAC-1, Dec. 1983, pp. 1022–1028.
[13] R. Rettberg, 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, Rep. 4098, Aug. 1979.
[14] J. S. Turner and L. F. Wyatt, "A packet network architecture for integrated services," in *Proc. GLOBECOM 83*, Nov. 1983, pp. 45–50.
[15] J. S. Turner, "Packet load monitoring by trunk controllers," U.S. Patent 4 484 326, Nov. 20, 1984.
[16] ——,, "Packet switching loop-around network and facilities testing," U.S. Patent 4 486 877, Dec. 4, 1984.
[17] ——,, "End-to-end information memory arrangement in a line controller," U.S. Patent 4 488 288, Dec. 11, 1984.
[18] ——,, "Interface facility for a packet switching system," U.S. Patent 4 488 289, Dec. 11, 1984.
[19] ——,, "Packet error rate measurements by distributed controllers," U.S. Patent 4 490 817, Dec. 25, 1984.
[20] ——,, "Fast packet switch," U.S. Patent 4 491 945, Jan 1, 1985.
[21] ——,, "Fast packet switching system," U.S. Patent 4 494 230, Jan. 15, 1985.
[22] ——,, "Design of an integrated services packet network," *IEEE J. Select. Areas Commun.*, pp. 1373–1380, Nov. 1986.

★

Jonathan S. Turner (M'77) received the B.S.C.S. and B.S.E.E. degrees from Washington University, St. Louis, MO, in 1977. He received his M.S. and Ph.D. degrees in computer science from Northwestern University, Evanston, IL, in 1979 and 1981, respectively.

From 1977 to 1983 he worked for Bell Laboratories, Naperville, IL, first as a member of Technical Staff and later as a Consultant. His work there included the development of maintenance software and design of system architectures for telephone switching systems. From 1981 to 1983 he was the principal system architect for the fast packet switching project, an applied research project which established the feasibility of integrated voice and data communication using packet switching technology. He is now an Associate Professor of Computer Science at Washington University, where he continues his research on high performance communications systems. His research interests also include the study of algorithms and computational complexity, with particular interest in the probable performance of heuristic algorithms for NP-complete problems.

Dr. Turner is a member of the Association for Computing Machinery and the Society for Industrial and Applied Mathematics. He has also been awarded 11 patents for his work.