Blazenet: A Packet-Switched Wide-Area Network with Photonic Data Path

ZYGMUNT HAAS, MEMBER, IEEE, AND DAVID R. CHERITON, MEMBER, IEEE

Abstract—**High-performance wide-area networks are required to interconnect clusters of computers connected by local area and metropolitan area networks. Optical fiber technology provides long distance channels in the multigigabit per second range. The challenge is to provide switching nodes that handle these data rates with minimum delay, and at a reasonable cost.**

In this paper, we describe a packet-switching network with photonic data path, christened Blazenet,[1] that provides low delay and has minimal memory requirements. It can be extended to support multicast and priority delivery. Such a network can revolutionize the opportunities for distributed command and control, information and resources sharing, real-time conferencing, and wide-area parallel computation.

I. Introduction and Motivation

THE potential of computer communication is, at present, severely handicapped by the poor performance of wide-area networks. The geographically dispersed clusters of machines operated by military, commercial, government, and research organizations are information and resource "islands" that limit the efficiency, capability, and responsiveness of these organizations. Distributed environments and more performance-demanding applications will characterize future wide-area communication, requiring wide-area networks that are matched in delay and bandwidth to the performance and requirements of local-area and metropolitan-area networks.

Optical fiber provides a long distance channel technology that makes this goal feasible. Transmission rates of gigabits per second with bit error rate on the order of 10^{-9} over tens of kilometers are already achieved today [1], [2], [3]. Fibers are being installed extensively [4], replacing twisted pairs and coaxial cables, and bringing with them the benefit of very high bandwidth, two or three orders of magnitude higher than that of existing networks. The challenge is to provide switching nodes that handle these high data rates with minimal delay and at a reasonable cost.

Optical switching and processing of optical transmission open new dimensions in future networking. Photonic implementation of data path, as opposed to a conventional electronic implementation, offers increased data rates. However, large and fast memories appear to be a difficult component to realize photonically.

In this paper, we describe Blazenet, a high-performance packet-switched network based on optical fiber and photonically implementable data path. Packets that are blocked in the switch are looped back to the previous node in the route, eliminating the need for memory at the switch. This technique exploits the storage arising from the high bandwidth-distance product of the optical fiber links, and is analogous to the memory delay lines in the earliest computers.

In the photonic implementation of the packet transmission path, transmitted signal is not converted to electrical signal at each switching node, but remains as light. The information is extracted from the light signal by detecting the appropriate fields and converting this information to electrical signal. This signal is electronically processed by the Control that electronically drives the photonic switches. Thus the processing is done electronically, while the packet transmission path is photonic. (Electronic controlling of very fast optical data rates is possible, as discussed in the Appendix B.) Signal regeneration is performed by optical amplifiers, described in [5].

Blazenet uses two key ideas to simplify the switching node design. First, packets are source routed [6], [7] to eliminate the need for all but the simplest routing logic in the switch. Second, because of the elimination of buffers within the switch, there is no need for flow-control mechanism.

Blazenet is presented here as a wide-area network. We see it as a backbone network, whose nodes are gateways to other networks. However, the design can be adapted to smaller networks, including local area networks.

Section II describes the Blazenet design, addressing the issues of packet-switching and traffic congestion. Section III presents a detailed switching node design. Section IV shows the expected performance of Blazenet as determined by simulation. Section V discusses some extended features that can be incorporated into Blazenet's design; priority traffic, limiting packet life-time, and broadcast and multicast. Section VI presents some issues of the higher layers that have direct implication on Blazenet's operation. The final Section VII summarizes our conclusions about the design and the implications. Some implementation issues are addressed in the Appendix.

II. Blazenet Design

A Blazenet is composed of a set of switching nodes interconnected by point-to-point logical links formed by the fiber loops. The hosts and gateways on the periphery of the network act as sources and sinks for the network traffic. Packets generated by hosts are passed to the switching nodes to which they are connected. The packets are then forwarded from node to node until they arrive at the switching node connected to the destination host, where they are removed by the switching node and passed to the destination host. An example of a four node Blazenet is shown in Fig. 1.

A loop, shown in Fig. 2, is built of two point-to-point physical links. In such a configuration each conventional bidirectional link connecting two adjacent nodes is replaced by a single loop. A number of loops can be multiplexed on a single fiber (if the fiber provides enough capacity) by wavelength division multiplexing, for example (see Appendix-C).

The Blazenet packet format, shown in Fig. 3, is composed of two delimiting synchronization fields (*syncs*), a header, and a data portion. Within the header, the *token* identifies whether the packet is a blocked packet (set *token*) or not (reset *token*), and the *hop-selects* dictate the hop-by-hop route for the packet to reach its destination. The *priority* and the *loop-counter* fields are optional extended fea-

Paper approved by the Editor for Lightwave Networks of the IEEE Communications Society. Manuscript received July 14, 1988; revised July 11, 1989. This work was supported in part by the Defense Advanced Research Projects Agency under Contract N00039-86-K-0431, by the Digital Equipment Corporation, AT&T Information Systems, and by IBM. This paper was presented in part at the IEEE Pacific RIM Conference on Communication, Computers, and Signal Processing, Vicoria, B.C., June 4-5, 1987 and at SIGCOMM '87 Workshop, Stowe, VT, August 11-13, 1987.

Z. Haas is with AT&T Bell Laboratories, Holmdel, NJ 07733.

D. R. Cheriton is with the Department of Computer Science, Stanford University, Stanford, CA 94305.

IEEE Log Number 9035797.

[1] The name Blazenet refers to the use of lasers with fiber optics as well as the boomerang aspect of the returning packets that cannot proceed onwards, i.e., Boomerang Laser network. It also refers to the notion of a packet "blazing" a route through the network, and the speed at which is does so.

0090-6778/90/0600-0818$01.00

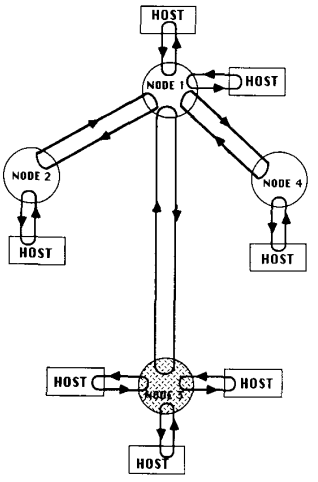


Fig. 1. A four-node Blazenet example.

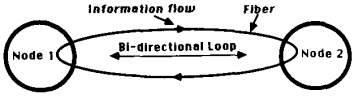


Fig. 2. A Blazenet loop.

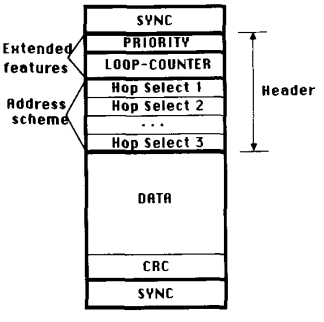


Fig. 3. The Blazenet packet format.

tures and are discussed in Sections V. The data portion contains higher level protocol data and can, optionally, be protected by checksum or CRC.

A. Source Route Packet Switching

Blazenet uses source routing. Each packet contains a sequence of hop-selects, specified by the source host. The hop-selects represent the switching operations to be taken in the sequence of nodes along the packet path through the network from its source to its destination. Each hop-select field indicates the output link on which the packet is to be forwarded for that hop. When a packet arrives at a switching node with its token reset, the first hop-select field in the packet is examined to determine the next output link for the packet. If that output link is available for transmission of a new packet, the first hop-select field is zeroed and the packet is immediately routed to the available output link. (The zeroing of the first hop-select field during the forwarding process means that the first nonzero hop-select field in the packet always represents the current hop selection.) If the output link is not available, the packet is blocked. Handling of blocked packets is explained in Section II-B. A packet with a set token arriving at a switching node indicates that the packet was blocked in the previous switching node. Consequently, such packet is simply left on the loop to be returned to the blocking node, after its token field is reset.

This design has several advantages. First, because the path determination is done at the source, the switch design is relatively simple. Second, it is feasible to perform the switching function at gigabit per second data rates, because of the simple logic required to make the hop selection. In particular, no table lookup is required for the switching decision. Finally, the delay for switching in a node is limited to the time required to interpret the packet header, check the availability of the output link, and perform the actual switching operation (if the output link is available). Because the extra delay introduced by a switching node is a few tens of bits, this delay is only a small fraction of the propagation delay of a link in a wide-area network.

B. Handling Packet Blockage

A packet is said to be blocked if it arrives at a switching node when the next output link is unavailable. A blocked packet is routed back to the previous switching node on the reverse link of the loop on which the packet arrived. Upon its arrival at the previous switching node, the returned packet is looped back, arriving at the blocking switching node one roundtrip time after its first arrival at this node. Thus, the loop effectively provides short-term storage for the packet, causing the packet to reappear at the blocking switching node a short time later.

This approach to handling blockage has several advantages. First, it dramatically reduces the average packet delay through a loaded network and increases the network capacity, compared to a design in which the packet is simply dropped when blocked at the outgoing link, referred to here as a *lossy* network. When a packet is dropped in a lossy network, it has to be retransmitted by the source after some timeout, at least one roundtrip time long. Since the probability of a packet being blocked increases with path length, as does the network investment in the blocked packet, dropping the packet seriously degrades the network performance under load for wide-area networks with realistic diameter.

Second, the design does not require memory in the switching node of the size and speed required to store all blocked packets, such as would be needed for a conventional *store-and-forward* design. Several megabytes of memory operating at 1 Gbps would increase the cost of the switching nodes and make the photonic realization of the data path less attractive. The combination of the high data rates, the wide-area span of the links, and the low-cost of the fiber makes this form of storage attractive. For example, a 100 km link (=200 km loop) operating at 1 Gbps can store nearly 1 Mbit or 125 packets of 1 kbyte each.

Finally, the loopback technique exerts back pressure on the link over which the packet was received, because the loop is then less available for new packets to be forwarded on it. In the extreme, this back pressure extends back from the point of contention to one or more packet sources. Besides alerting the packet source of congestion, the back pressure provides fast feedback to the source routing mechanism, allowing it to react quickly to network load and topological changes.

A potential disadvantage arises when the link between switching nodes is very long, since the roundtrip delay on the loop may be excessive. We avoid this problem by including loopback support in the optical repeaters that are required anyway every few tens of kilometers on a fiber optic link. Thus, a packet that is blocked at a switching node is looped back either to the previous switching node or to the previous repeater, whichever is closer. If, for example, the distance between adjacent switching nodes is 100 km, the roundtrip delay is approximately 1 ms. Because a Blazenet switching node includes the regeneration function between the input and output ports, it can be used as a repeater, thereby automatically supporting the loopback function. (In the case of a repeater, only two input and two output loops of the switching node design are used.) The network is then built from just one type of interconnection component, rather than two. By using such a design, packets can loop at intermediate loops on a long link, reducing its delay through the network. Consequently,

the packet is delayed in time units corresponding to the roundtrip time on the intermediate loop rather than that for the entire link. Another improvement consists of designing the last loop shorter than other loops on the link. Consequently, blockage at low-load operation has smaller effect on the packet delay.

III. Switching Node Design

A. Basic Design

A Blazenet switching node can be implemented as simple interconnection of a number of photonic components and conventional electronics. We assume the availability of fast switching devices capable of switching within a small fraction of duration of a header bit, once the switching command has been initiated. Such devices exist today for at least as fast as 3 GHz operation speed [8]. Slower devices can be employed for lower cost by maintaining an adequate interpacket gap.

Fig. 4 shows the block design of a Blazenet switching node. Each *loop* has a *header detector*, a *delay line*, and an *end-of-packet detector*, built out of a piece of fiber with appropriate taps. The header detector is long enough to contain a packet header, and the delay line is long enough to contain a maximum sized packet and the number of bits corresponding to the time the control logic requires to do the actual switching. Fig. 5 shows the control signals extracted from the transmission and the corresponding timing.

The switching process is initiated by the *new-packet* signal generated by a pattern detection circuit, which searches for the *sync* pattern. Upon *sync* detection, the circuit raises the *new-packet* line, indicating to the Control that a new packet has arrived. The detection can be implemented with devices that are slower than the data rate (see Appendix-B). At this time, the control reads the values of the token and the hop-selects signals. The switching decision (checking the availability of the output loop) is performed during the period of time named "switching delay," during which the packet propagates through the header detector, and at the end of which a switching command is issued to the switching element (represented in Fig. 4 as switches). The packet will be either blocked and returned on the reverse portion of the loop it arrived on, or will be forwarded to the output loop according to the value of the current hop-select. The indication that a packet leaves the node is provided to the control by the end-of-packet line of the loop the packet is forwarded/returned on. In the case that the packet is blocked and need to be returned, and the reverse portion of the loop is busy forwarding another packet, the packet is temporarily clocked into the delay line. Since the length of the delay line corresponds to the maximal packet size, it is guaranteed that when the delayed packet arrives at the end of the delay line the reverse portion of the loop is already free.

The delay line is considered to be free if it does not contain a packet or any part of a packet. By using the two signals new-packet and end-of-packet the control can uniquely determine the status of the delay line. The control keeps a single bit per loop to record the status of a loop. If no transmission is forwarded to a loop and its delay line is free, the loop is considered idle, and its bit in the control is set accordingly.

The switching decision is made in the following way. After the number of the loop, a packet is to be forwarded on, is determined by the hop-selects signal, the availability of this loop is checked by checking the status of its bit in the control. If the loop is idle, the packet from the input loop is clocked onto the output loop. If, on the other hand, the output loop is busy, the packet is blocked and returned by being clocked out on the loop it came on, after its token is set. In case more than one packet tries to enter a specific loop, only one packet wins, and the other(s) are clocked out on their loops. Upon its arrival to the other end of the loop, the blocked packet is clocked into the corresponding delay line, blocking access to this loop for any new arrival. When the packet reaches the end of the delay line it is clocked out onto the loop it came on, after the token is reset. (Another possible implementation is to check the loop availability upon reception of a returned packet. Only in the case the loop is busy (forwarding another packet), the returned packet is entered into the delay line. A packet is, however, directly clocked out if the loop is found free upon the returned packet arrival. This improvement has the advantage of including an additional delay-line delay only in the case the loop is busy. On the other hand, this implementation has a disadvantage of complicating the switching process and therefore, the control itself.)

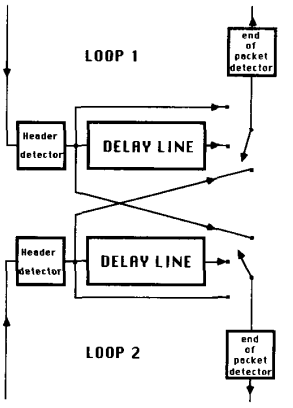


Fig. 4. The switching node design.

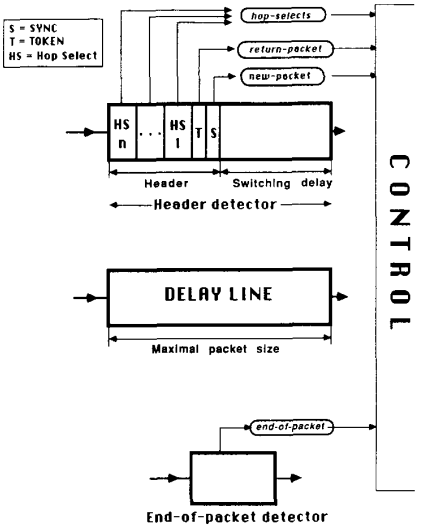


Fig. 5. Timing of delay line signals.

The switching element is capable of connecting each one of its inputs to any one of its outputs. The switching element can be designed in several ways. One such a possibility is to use a switching matrix, as shown on Fig. 6. In this case maximal connectivity can be achieved.

The control performs the actual routing decisions based on the signals that indicate the status of the loops. The signals entering the control are shown in Fig. 7. The routing algorithm takes into account the following parameters:

1) input packet destination,
2) availability of the output loop and its delay line, and
3) priority of the packet (as explained in Section V-A).

In general, a packet is either completely forwarded or returned. However, in some cases where extraordinary priority is needed, it may be necessary to abort transmission of a packet currently being forwarded. A simple mechanism can be incorporated into the design such that upon reception of a high priority packet, the packet is immediately forwarded on the appropriate loop. (Priority traffic is discussed in Section V-A.)

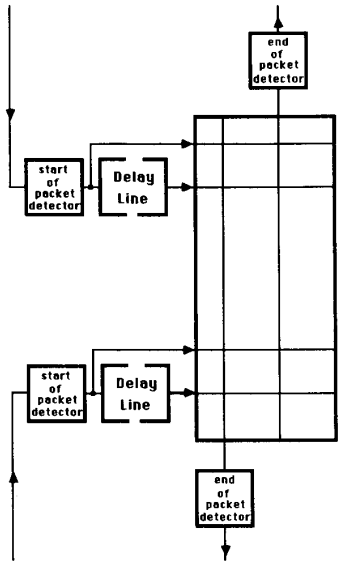


Fig. 6. The switching node design using a switching matrix.

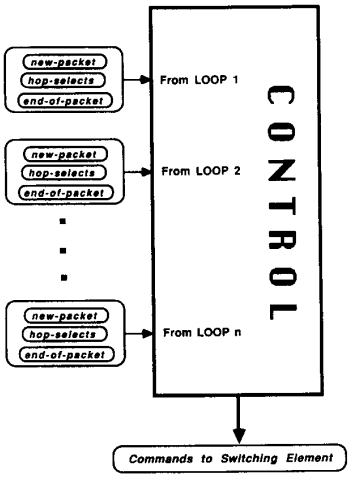


Fig. 7. The switching node control signals.

The input traffic from a host connected to the switching node is switched in a similar way the traffic from any loop is. The main difference is in the indication of an available packet. An indication line from a host to the control continues to show the presence of a packet until it is forwarded. No returning of an incoming packet is ever performed.

The output traffic destined to a host connected to the switching node is received on one of the outputs of the switching element and passed to the appropriate host. A host is assumed to be always ready to accept its traffic. If the host is unavailable, the packets are discarded. Therefore, the major difference between the through traffic and the exiting traffic is that the latter is never returned. The reason for not returning exiting blocked traffic is to avoid situations in which the network can possibly be blocked because of a host malfunction.

B. Double-Loop Design

In the double-loop version of the network two loops replace a bi-directional link of a conventional network. Such a configuration is presented in Fig. 8. Lower portion of loop 1 serves transmission

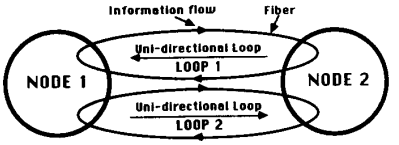


Fig. 8. A double-loop configuration.

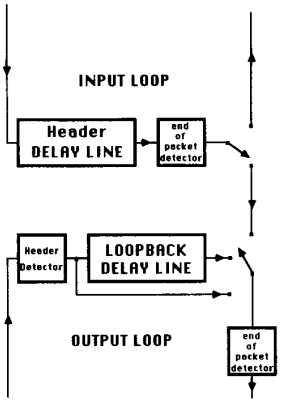


Fig. 9. The double-loop switching node design.

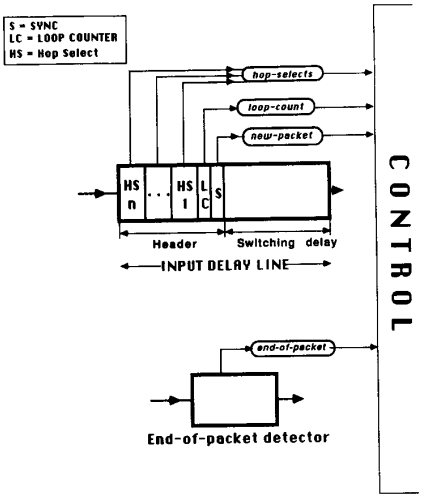


Fig. 10. The header delay line signals and their timing.

from node 2 to node 1, while transmission from node 1 to node 2 uses lower portion of loop 2. Blocked transmission is returned on the upper portion of the loop it came on. No indication of a packet being a returned packet is necessary in the double-loop case, since use of the upper portion of a loop indicates that the packet is a blocked one. For double-loop Blazenet, the token filed is discarded from the single-loop packet format.

Fig. 9 shows the modified block design of Blazenet's switching node to accommodate the double-loop configuration. Each *input loop* has a header-delay-line. The header-delay-line is long enough to contain the leading sync, the hop-selects, and the number of bits corresponding to the time for the control logic to do the actual switching. Fig. 10 shows the signals extracted from the transmission entering the header-delay-line and the corresponding timing. Upon sync detection, a pattern detection circuit raises the new-packet line, indicating to the control a new packet arrival. At this time the control

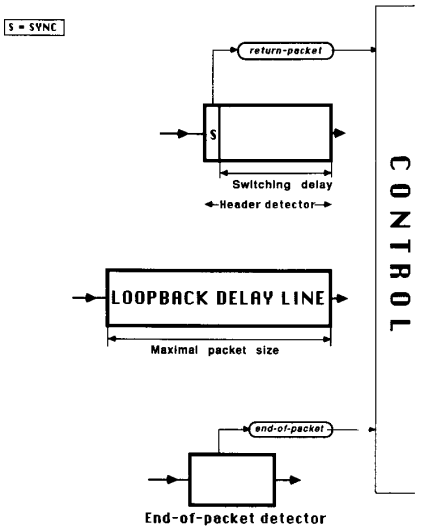


Fig. 11. The loopback delay line signals and their timing.

looks for the value of the current hop-select. The indication that a packet leaves the header delay line is provided to the control by the end-of-input line.

Each *output loop* also has its own header detector and a *loopback delay line*. The header detector contains a pattern detection circuit to identify the sync field and delay line of the length corresponding to the switching delay. The loopback delay line must be of the length of the maximum packet size. The control signals and their timing are presented in Fig. 11. Upon detection of the leading sync pattern of a returned packet, the *return-packet* signal is raised. This indicates the occupation of the loop. Similar circuit, positioned at the end of the loopback delay line, scans for the trailing sync. Detection of the trailing sync by this circuit initiates the end-of-output signal, indicating when a packet leaves the loopback delay line. Using the two signals, the Control can uniquely decide on the loopback delay line state.

The process of forwarding a packet in the double-loop configuration is similar to that of the single-loop case with proper differentiation between the header delay line and loopback delay line functions.

The main advantage of the single-loop over the double-loop version is in reduction of hardware: fibers, transmitters, receivers, etc. The double-loop version is simpler to implement, possesses some reliability advantages, has lower delay and stable throughput under heavy-load. In the following section, we consider the performance of both the single and the double-loop configurations.

C. Slotted Loops

Another alternative of Blazenet implementation is to use slotted loops. In the slotted version, the loops are divided into slots of the packet size (in case of variable packet size, the slots are of the maximum packet size). Packets can be inserted only into empty slots, indicated by some bit within the packet format. By appropriately delaying each input traffic, slots' arrivals to a switching node are synchronized, so that all the packets arrive at the same time. Packet can be, then, interchanged between the slots of the various loops. Blazenet variations can be combined. Thus, single-loop and double-loop Blazenet can operate on slotted or unslotted loops.

The slotted version has some advantage in performance over the nonslotted approach. Nevertheless, the required slot synchronization is a serious disadvantage of the slotted version for wide-area networks. Also, in a network with a variable packet size the usage of the maximum packet length as the slot size may be of some disadvantage. Consequently, the slotted version is not pursued further here and we concentrate on the nonslotted single and double-loop versions.

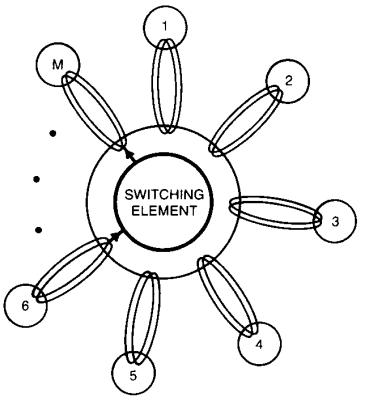


Fig. 12. Star topology.

IV. Blazenet Performance

Blazenet performance is evaluated in terms of the average packet delay through the network as a function of the network throughput. Packet delay is the period of time from when the packet is passed to the network until it is delivered to its destination averaged over all packets entering the network, and includes the queueing time at the network entrances.

A general event-driven simulation program was developed to evaluate Blazenet performance. The simulation enabled us to evaluate Blazenet performance, as well as to compare Blazenet performance to the idealistic case of the nonblocking network, that is with the propagation delay only. We also compared Blazenet performance to the case of the lossy approach.[2] The following graphs show Blazenet performance for several different network topologies and different packet sizes. In all of the examples we assume that the traffic matrix is symmetric, the link capacity is 1 Gbps, and the links are all approximately 100 km long. (The length of the loops has little effect on the relative increase of the delay as a function of link utilization, provided that the loops are long enough to eliminate correlation between subsequent blockages of a packet. This can be ensured, for example, by having the lengths of the loops that terminate on the same switch, differ by at least one packet size.)

Blazenet performance was evaluated for packet sizes of 5 and 10 kbit. These values represent reasonable tradeoff between the long delay line for large packet size and excessive header (Blazenet and high-level protocol) overhead of small packets. For example, the combination of a Blazenet, internetwork datagram and transport layer headers could total 100 bytes, requiring a 10 kbit packet to keep the overhead under 10%. On the other hand, 5 and 10 kbit correspond to 1 and 2 km delay line on 1 Gbps link (or transmission times of 5 and 10 μs), respectively, thus representing a feasible design.

The first example is a *Star* topology with 5 inputs. General Star topology with M inputs is shown in Fig. 12. The delays as a function of network throughput, evaluated for single- and double-loop configurations, are presented in Fig. 15. The propagation delay through the network is also shown for comparison.

The second example is the *Star-of-Stars* topology, shown in Fig. 13. The simulation results are presented in Fig. 16.

The final case is of the *Triangle-of-Stars* topology, shown in Fig. 14, with corresponding results in Fig. 17.

The comparison of Blazenet performance to the lossy network in Star-of-Stars topology is shown in Fig. 18. In the lossy network case it is assumed that a blocked packet is retransmitted immediately after

[2] To the best of our knowledge, other photonic networks utilizing the "hot potato" routing scheme are defined as LAN's or MAN's. In this sense, Blazenet, being a "hot potato photonic WAN" is unique. (The reason that Blazenet is relatively span-independent is the unconstrained topology that Blazenet can be implemented in.) Thus we chose to compare the Blazenet performance only with the lossy network.

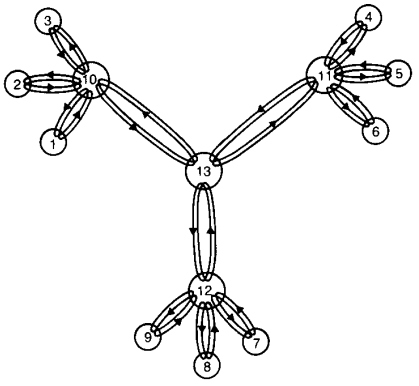


Fig. 13. Star-of-Stars topology.

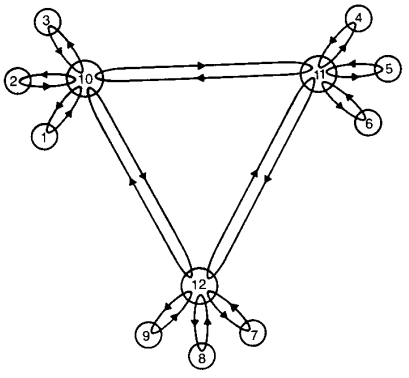


Fig. 14. Triangle-of-Star topology.

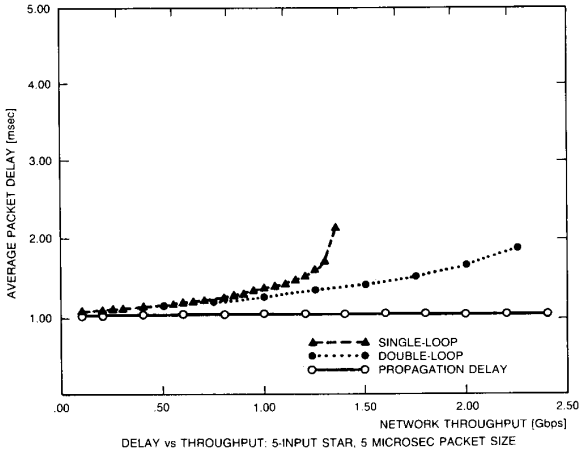


Fig. 15. Delay of star Blazenet.

a single roundtrip delay between the source and the destination without any processing overhead, thus favoring the lossy case. Also, the small network span somewhat favors the lossy approach in this comparison, since the Blazenet advantages are emphasized in networks with large average path length.

These simulation results indicate that the performance of Blazenet is significantly better than a Lossy network at realistic load levels. In particular, Blazenet avoids the dramatic increase in packet droppage that occurs in lossy networks under load and increasing diameter,

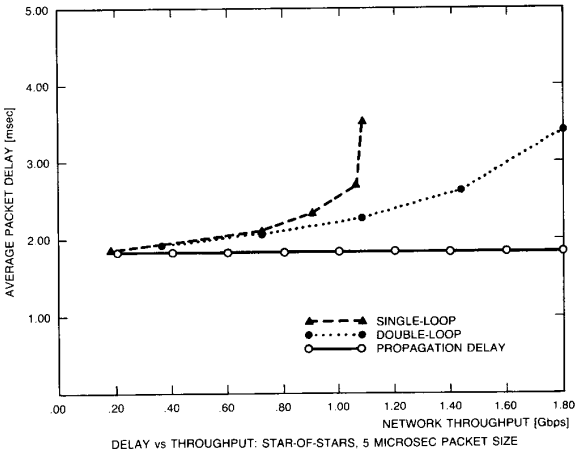


Fig. 16. Delay of Star-of-Stars Blazenet.

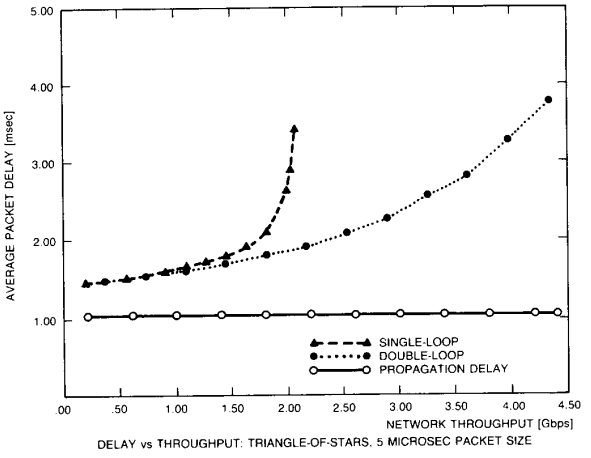


Fig. 17. Delay of Triangle-of-Star Blazenet.

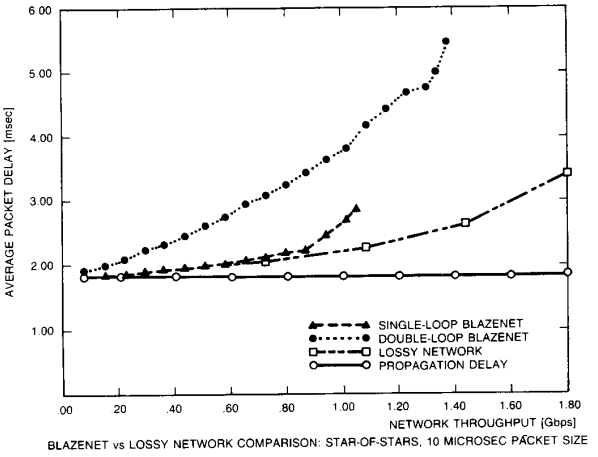


Fig. 18. Lossy and Blazenet performance Blazenet.

because packet blockage is largely decoupled from packet drop. In a lossy network, the probability of a packet being blocked, and therefore dropped, increases as the length of the packet route increases. Thus, lossy networks have the unstable property of tending to drop the packets in which they have invested the most resources. This behavior also unfairly favors short routes over long routes.

Blazenet performance closely approximates the ideal network where the packet delay is the propagation delay for low-load range. The delay for the double-loop configuration smoothly increases with increasing load, whereas the single-loop experiences much sharper knee in its delay curve, i.e., "ALOHA-like" behavior. This difference is explained by the loopback behavior. Under increasing load, an increasing number of packets are looped back because of blockage. In the single loop configuration, these blocked packets can interfere with new packets bound in the opposite direction; however, in the double-loop configuration they do not. In fact, in the double loop configuration, the blocked packets provide a form of back pressure on packets attempting to enter a congested portion of the network. This back pressure allows double-loop Blazenet to handle congestion effectively.

The choice between single- and double-loop configurations is dependent on the expected load and the required behavior under load. One might argue that if Blazenet could offer such high bandwidth than the network would operate at the low load. For example, consider a Blazenet connecting a collection of 10 Mbps Ethernets operating at 10% utilization. Assume further that 25% of an Ethernet traffic is to be transferred on the backbone Blazenet, whose links consists of ten fibers operating at 1 Gbps each. Thus, as many as 4000 Ethernets can coexist on Blazenet utilizing the network only in 10%, utilization that represents low-load condition. On the other hand, historically, users have managed to consume the capacities of networks in the past and with the potential of transferring databases, high resolution images, real-time video and other high volume traffic, it seems feasible to expect even a gigabit network to be loaded at times. Thus, a double-loop configuration might be chosen to ensure greater network performance stability.

The choice between double loop and single-loop configurations is also dependent on cost, complexity and performance tradeoffs that are only known from further understanding of the precise fiber and switching technology to be used. For example, a double-loop may require twice as much fiber or limit the data rate to half that of single-loop configurations. Conversely, the simpler logic of the double-loop switch may be a greater advantage, especially if the switch is the limiting factor for the link data rate. Further development of the photonic technology and experimental implementations of Blazenet are required to more fully understand these tradeoffs.

The performance results presented here summarize and extend those discussed in [9], [10], [11], Overall, Blazenet provides performance characteristics that are an attractively alernative to a lossy network without the implementation complexity of conventional packet-switched store-and-forward networks.

V. Extended Features

The basic Blazenet node design can be extended to include some additional features: priority traffic, limiting of the life-time of a packet, and broadcast and multicast traffic. Besides providing important services to the network users, these features increase the strength of the network to cope with abnormal situations, thereby increasing the network reliability.

A. Priority Traffic

The implementation of priority traffic in Blazenet can be accomplished in two ways: by including a priority field in the packet format or by giving preference to some traffic during the forwarding process. The former approach is considered first.

The priority traffic implementation is achieved by delaying the forwarding of a packet by a period equal to the transmission time of a maximum packet length. At the end of this period, the packet with the highest priority is forwarded, while other packets (if any) are looped back. The hardware of the basic Blazenet node design has

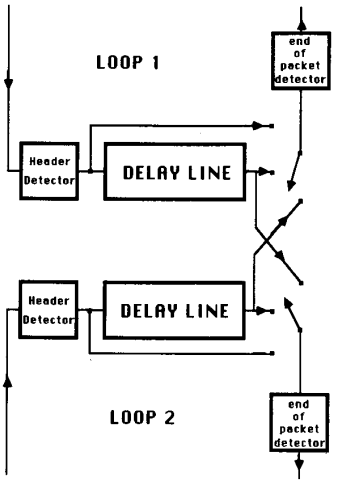


Fig. 19. Modified node design.

to be modified in order to accommodate this additional functionality. The main adjustment is to include a packet detector circuit within the header detector that initiate the packet-ready signal. The modified node design is shown in Fig. 19 and the modified control signals in Fig. 20.

A packet that is clocked into a header detector and has not reached the packet-ready point is called an active packet. The set of active packets at any point in time is the set of packets competing on the loops.

The new packet arriving on a loop is clocked into the header detector. After its main-header (composed of the fields: sync, token, priority, and the hop-selects) are received, the control is notified of the packet arrival and the packet's information is passed to the control. The control gathers all such information from all the header detectors. When a packet is shifted to the packet-ready point in the header detector, the decision is ready whether the packet will be forwarded or looped back. The decision is made according to the following algorithm:

IF ((priority $\geq$ priority of all active packets
with the same hop-select)
AND (no transmission in progress)
AND (destination delay line is free))
THEN forward the packet
ELSE loop the packet back.

The forwarding or blocking (namely, the switching) operations are, otherwise, done as before.

The essence of the above procedure is that, by delaying all packets by one packet length (i.e., one packet look-ahead), the priorities of all the relevant packets can be gathered and the correct decision made about which packet to forward. Therefore, the difference in this modified version of Blazenet is the point in time when the control's decision is made.

Using the above scheme, the delay of a forwarded packet is increased by the transmission time of a packet of the maximum size. However, this time is negligible compared to the propagation delay encountered by a packet on a link in wide-area network. (For example, for 10 kbit packets on 1 Gbps Blazenet the additional delay is only 10 μs, delay that is small compared to 500 μs that is the propagation time of a 100 km link.) The total delay is, therefore, essentially unaffected by this hardware modification.

Another way to implement priority traffic in Blazenet is to give preferences to some traffic during the forwarding process. One such possibility is to give preference to traffic coming from the hosts connected to the node over all the other traffic. Such a mechanism is useful for coping with temporary traffic surges from the node's hosts.

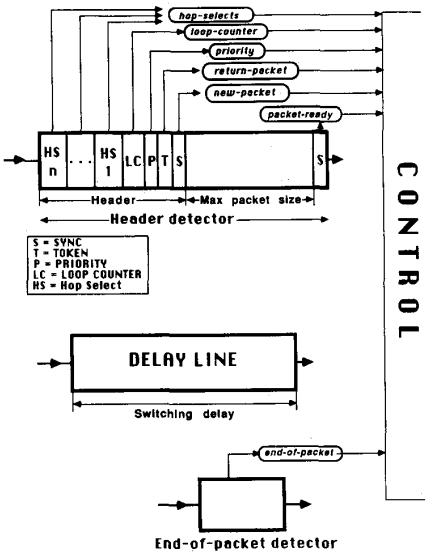


Fig. 20. Modified delay line structure.

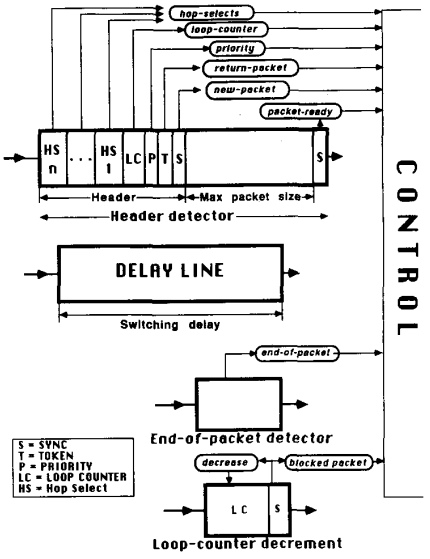


Fig. 21. Control signals for loop-counter implementation.

However, although this approach lowers the delay of the preferred traffic, the mean packet delay in the whole network is increased. Therefore, in order to ensure fairness, usage of such a mechanism should be restricted.

The preferences given to some traffic can be based on other criteria. Traffic arriving on some loops (for example, traffic coming from congested areas) may be given higher priority in the forwarding process. The preferences criteria can be based on various network parameters and can be adjustable in time, as the network load and topology change.

B. Limiting Packet Life-Time

The network needs to limit packet life-time because of three reasons: to eliminate erroneous traffic to exist in the network and interfere with valid traffic, to discard real-time traffic that could not be delivered on time and became obsolete, and to avoid wrap-around of packet sequence numbers in high-level protocols.

In Blazenet, the *loop-counter* provides the mechanism for limiting the life-time of packets within the network. The loop-counter is decreased each time a packet is blocked and returned. When the loop-counter reaches zero, the packet is discarded. The loop-counter represents, therefore, the maximum number of times a packet can loopback. The value of the loop-counter is set by the source host, according to packet type and time limitations on the packet delivery.

Unless the loops are of equal length, the loop-counter mechanism does not provide an accurate mean for limiting a packet life-time within the network. In the case the loops are of unequal length, using the minimum loop length of the packet path for the calculation of the loop-counter can be an adequate approach. Assign n to represent the refractive index of the fiber, $l_{\min}$ the minimum loop length of the packet path (=twice the distance between the adjacent switching nodes), l_i the length of the ith loop, h number of hops on the packet path (=number of switching nodes on the path $-$ 1), $t_{\min}$ minimum life-time of a packet in the network and c the speed of light in vacuum. Therefore, the value of the loop-counter can be calculated using:

$$\text{loop-counter} \geq \frac{c \cdot t_{\min}}{n \cdot l_{\min}} - \frac{1}{2 \cdot l_{\min}} \cdot \sum_{i=1}^{h} l_i$$

(c/n gives the velocity of the light in fiber, $t_{\min} \cdot c/(n \cdot l_{\min})$ gives the maximum number of loops to travel during this time. The second term in the above inequality is a correction required due to the fact that an unblocked packet travel only half a loop.)

Another approach would be to use some weighted average of the loops lengths on the packet path, l_{avg}. In this case the loop-counter is calculated by the same formula as above, substituting l_{avg} for $l_{\min}$.

If the general repeater/switching node design is used, all network loops are of equal length, l (possibly with the exception of the last loop hitting the node), and the calculation of the required value of the loop-counter becomes:

$$\text{loop-counter} \geq \frac{c \cdot t_{\min}}{n \cdot l} - \frac{h}{2}.$$

By imposing some minimum value on the packet life-time $t_{\min}$ the packet will not be discarded because of its lifetime expiration for at least this period of time. This is advantageous in situations where we are more concerned with the possibility of discarding still valid packet, than with the possibility of an obsolete packet living in the network or even being passed to the destination. When we are in the opposite situation, namely, when we are more concerned with the access load created by an obsolete traffic than with the possibility of discarding valid traffic, we should use some maximum permissible value for the packet life-time $t_{\max}$ instead. (The above formulas continue to be valid in this case, with the substitution of maximum loop length $l_{\max}$ for $l_{\min}$ and reversing the unequality sign.) Note, that by manipulating the current value of the packet life-time, the network can regulate its load. Of course, such a manipulation is justified in some special circumstances and for traffic that does not require reliable transport through the network.

The implementation of the loop-counter mechanism includes a decrement circuit. This circuit, as well as the circuit that tests the value of the loop-counter, operate on returned packets only. No action is necessary when the packet is forwarded. The modification to the node design to support the loop-counter operation includes two *loop-counter decrement* circuits each one placed after the end-of-packet detectors. The structure of a delay line is shown in Fig. 21. While the packet enters a delay line, the loop-counter is checked by the control. In case the value of the loop-counter is zero, the packet is discarded. The other possibility is to pass the discarded packet to a host performing the function of switching node monitoring.

As a blocked packet is clocked out of the delay line, the *blocked-packet* circuit detects the sync and the token of the packet, which initiate a *decrease-loop-counter* signal if the token is set. The delay between the blocked-packet and the decrease-loop-counter circuits is exactly of such a distance that when the blocked-packet signal is raised, the loop-counter is received by the decrease-loop-counter

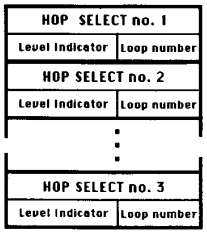


Fig. 22. Modified hop-select structure for multicast delivery.

circuit. The operation is, therefore, fully autonomous, not requiring any intervention of the control.

When the loop-counter is represented by a binary number, the loop-counter decrement hardware may be difficult to implement. A somewhat easier solution may be to use a bit pattern as a loop-counter. In this scheme the loop-counter is composed of a string of 1's, equal in number to the required value of the loop-counter. Each decrement of the loop-counter consists now of resetting one such bit. Zero value is detected by having all zero pattern. This scheme has the disadvantage of providing unnecessary long loop-counter field. Fortunately, the maximum value of the loop-counter is expected to be small. Consequently, the ease of implementation justifies the bit wastage.

Yet another approach to the loop-counter usage is to provide a special loop-counter per each hop. In this case, instead of the hop-selects fields, the packet header contains fields composed of hop-selects and loop-counters. The advantage of this scheme is the possibility of an exact calculation of the packet's life-time, as well as the possibility of selectively limiting the delay of each of the loops on the packet's path.

C. Broadcast and Multicast

Routing of multicast packets on Blazenet is achieved by a tree-like forwarding path, where the source is the root and the destinations are the leaves. A multicast packet is forwarded as a single packet, up to the point where it is split to two or more packets forwarded on different links. The split packets can also be multicast packets, in which case each one is split again at some subsequent node.

A multicast packet address is, in fact, a mapping of this tree graph to a linear notation. The linear notation consists of a list of hop-selects while searching the tree in the following way. Visit the leftmost unvisited son of the current node, if any, whose subtree contains at least one destination. Each hop-select consists now of two subfields: the *level-indicator* and the *output-number*. The level-indicator indicates the level of the current node in the whole tree, while the *output-number* is the number of the loop, the packet has to be forwarded on (in the current node). The level-indicator is actually the hop distance of the current node from the source. Fig. 22 shows the hop-select structure incorporating the above changes.

Upon packet arrival at a switching node, the requested loops are checked for availability and the packet is split to all these requested output loops that are available, if any. The packet is also returned carrying the addressing information of all the blocked outputs, if at least one output loop is unavailable.

While a multicast packet is split within a switching node, the new generated packets carry the addressing representation of the relevant subtree only. The address field is, therefore, divided among the new generated packets, whereas the syncs, the token, the loop-counter, the priority, and the data portion of the packet are replicated within each one of the new packets. The replication is performed by connecting the input loop to more then one output loops. The division of the address field is performed by replicating the whole address field in each one of the new packets and erasing the irrelevant portion of the address field in any one of the new packets.

The following example clarifies the multicast addressing structure. Assume a single packet is to be multicasted to four destinations. The corresponding tree graph is shown in Fig. 23. The initial address

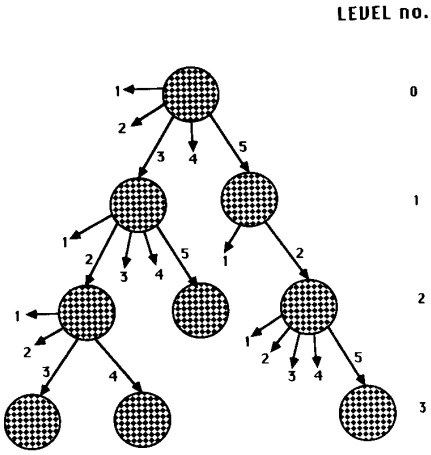


Fig. 23. Tree graph of the multicast example.

Level Indicator:	0
Loop Number:	3
Level Indicator:	1
Loop Number:	2
Level Indicator:	2
Loop Number:	3
Level Indicator:	3
Loop Number:	0
Level Indicator:	2
Loop Number:	4
Level Indicator:	3
Loop Number:	0
Level Indicator:	1
Loop Number:	5
Level Indicator:	2
Loop Number:	0
Level Indicator:	0
Loop Number:	5
Level Indicator:	1
Loop Number:	2
Level Indicator:	2
Loop Number:	5
Level Indicator:	3
Loop Number:	0

Fig. 24. Initial address field for the multicast example.

field is presented in Fig. 24. The first number of each hop-select represents the level indicator and the second one represents the output loop number. The first path is composed of the following sequence of hop-selects: 3, 2, 3, 0. The second: 3, 2, 4, 0. The third: 3, 5, 0. The fourth: 5, 2, 5, 0. A hop-select of the last forwarding node on the packet path (the destination node), is by definition 0. Therefore, all the paths end with hop-select equal to 0.

When the packet in the example arrives at the first node, it is split into two packets: one to be multicast to destinations: 1, 2, 3, and the second to be unicast to destination 4. The second packet is forwarded to its destination along the route: 2, 5, 0, whereas the first packet, when arrived to the second node on its path, is split once more. One of the new packets goes on output line number 2, the other is forwarded directly to its destination on output line number 5.

The address adjustment for the multicast packet is more complicated than for the unicast case because of the necessity of splitting the address field. The control looks for the level-indicator in the first hop-select. The whole address is then split into as many pieces as there are hop-selects with the same value of the first level-indicator. The division of the packet address field into pieces is performed by breaking the address field on the boundary of hop-selects with values of level-indicator equal to the value of the level-indicator of the

Level Indicator:	0
Loop Number:	00101
Level Indicator:	1
Loop Number:	01001
Level Indicator:	2
Loop Number:	00110
Level Indicator:	3
Loop Number:	00000
Level Indicator:	3
Loop Number:	00000
Level Indicator:	2
Loop Number:	00000
Level Indicator:	1
Loop Number:	01000
Level Indicator:	2
Loop Number:	00001
Level Indicator:	3
Loop Number:	00000

Fig. 25. Address field for the multicast example using bit representation.

first hop-select. Each new packet carries one such piece and is then forwarded according to the first hop-select. During the forwarding process the first hop-select is erased. The address field of the new packet is, therefore, composed of only the relevant subtree.

Another possible addressing scheme for multicast on Blazenet is the usage of a single hop-select field to indicate multiple output connection. In this scheme, M bits are used for each route, each bit for one of the M possible output loops. A bit is set if the packet has to be forwarded on the corresponding output loop. In this scheme, as in the previous one, the nested structure of the various paths realize the multicast delivery. This scheme is more efficient in the case of multicast to many destinations, however, the control has to be able to create the hop-select of the returned packet containing the indication of the blocked loops. Thus, this scheme requires more complex control design. Consequently, the preferred solution depends on the implementation requirements. Fig. 25 shows such a representation for the above multicast example.

It is to be noted that in both addressing schemes the packets created by splitting the original packet have unused gaps in the address field. Moreover, even in the unicast case, erasing the used hop-selects creates gaps. Although it is possible to eliminate these gaps, the cost of the gaps is insignificant, since typically the header is only a small portion of the whole packet.

Broadcast can be implemented on Blazenet in two different ways: by using the multicast mechanism with address of all the network destinations, or by a flooding approach.

Flooding can be implemented by dedicating a specific hop-select value to instruct the forwarding nodes to forward the packet on all its loops (possibly with the exception of the loop directed to the node the packet comes from). The first hop-select does not need to be erased in the forwarding process. By slightly modifying the treatment of the loop-counter in the switching nodes, the damping of the flooding process is guaranteed. The loop-counter modification consists of decreasing the loop-counter value each time a packet is received by a switching node, whenever the packet is blocked or successfully forwarded. This modification requires placing the loop-counter decrement circuits before the header detector. In order to make the packet reception by all the network nodes possible, the value of the loop-counter should be specified to the maximum path length from the packet source to any network destination with some reasonable addition for packet loopbacking. Using this flooding mechanism, a broadcasted packet can be received more than once. Consequently, higher layers protocols must discard the duplicated packets. Flooding can be used to cope with abnormal network behavior and to increase the network reliability.

VI. Higher Layers Issues

Blazenet provides high-performance packet delivery for the higher level protocols, most directly the internetwork layer (if used) and the transport layer. However, the design impose some additional requirements and consideration on the higher layers, which we consider below.

A host or gateway connected to Blazenet must be prepared to receive packets that are incorrectly delivered to it because the source route was corrupted in transit or specified incorrectly by the source. Blazenet does not provide error detection on the packet (including header) because the packet will have passed through the switch by the time it would be feasible to detect it was in error. Internetwork and transport protocols include their own checksum or CRC fields, allowing detection of corrupted packets at the host. Thus, sufficient mechanisms exist at the higher layers to deal with this problem, and the expected low error rate of optical-fiber transmission avoids placing a significant overhead on the hosts and gateways.

Blazenet also has the property that a packet is likely to arrive out of order relative to other packets send as part of a logical stream. Current standard transport protocol implementations tend to drop out of order packets, effectively negating some of the performance benefit of Blazenet. However, newer transport protocols such as VMTP [12] are able to accept out-of-order packets because of their provision for selective retransmission, which requires the handling or out-of-order packets by definition. Moreover, the implementation of standard protocols can be extended to handle out-of-order packets by simply buffering misordered packets for some period of time. We expect that this technique would handle most misorderings expected on Blazenet, given that the reorder should be local to a small number of packets.

Finally, Blazenet contains basically no intelligence in the switching nodes, so all routing and network state information must be maintained outside the network by hosts and gateways. We imagine a *routing server* module that executes in hosts and gateways directly attached to Blazenet which maintains this information. It periodically sends monitoring messages along different routes through the network to discover whether the node at the end of the route is operational, its identity, and the delay along that path. Normally, these messages are sent to the routing server on the destination node, which can return a response by the same route or another designated route, as specified in the packet. To check the availability of particular links, the server can also route packets in a loop back to itself. By analyzing information from many network paths and nodes, the server can detect incremental changes in the network load and network topology (i.e., availability of a specific link).

A test packet includes *test-nr*, *path-number*, and *input-time* fields as its data. In order to avoid confusion, the *test-nr* field differentiates between various tests (that can be performed concurrently) and the *path-nr* field uniquely identifies the specific path under the test. The *input-time* field records the time the packet was entered into the network and serves for calculation of the packet delay through the specific path.

In the following discussion we assume that the network changes are incremental that is, the probability of a failure of more than one link or node between any two tests is negligible. Therefore, we can assume that at any time the server notion of the network's link status is incorrect in at most the state of one variable.

The tests are performed in the following manner. Each server sends packets over the network to cover all the network links. If a packet does not return, more tests are initiated in order to determine which link on the missing packet path is down. The intersection of all the missing packets' paths gives the unoperative link (to remind, there is only one unoperative link, if any). However, in the case a link does not have an alternative, the link failure cannot be uniquely identified.

If a server decides on a link being unoperative, it may pass this information to the other hosts and cause changes in their routing tables. Later, from time to time, the server might reissue some tests to check if the status of an unoperative link has changed.

The same approach can be used in order to locate the areas of congestion in the network. However, more sophisticated algorithms must be used in order to analyze the packets' delays and to evaluate the state of the congestion of a specific link or group of links. A useful assumption is that a link's load does not change rapidly. This assumption is reasonable because the network provides high-

throughput so the influence of a small number of communication events on the overall network load is minimal. The mesh network topology also contributes to the smoothing effect.

The frequency of these test messages is such that they do not contribute significantly to the network load. As with conventional routing techniques, there is a tradeoff between the frequency of routing-induced communication overhead and the accuracy of the routing information. By careful design of the routing server and judicious exchange of routing information, the cost and complexity of this "network external" routing is not significantly than conventional routing techniques. Consequently, the performance level achieved by Blazenet as a result of using source routing are not reduced by routing overhead.

In general, we see considerable benefit in moving functionality out of the network to the periphery if there is any significant gain in packet delivery performance to be realized. As illustrated in the cases considered in this section, the higher layers provide the required functionality, or can be easily extended to do so. There is little gain in providing more functionality at the lower layers of a network, beyond the basic packet delivery of Blazenet. The potential of Blazenet and other networks in that performance range can only be realized with "leaner" approaches to protocol architectures than the richly layered standards that are currently in use.

VII. Conclusions

Blazenet is an attractive approach to implementing a multigigabit wide-area packet-switched network using fiber optics implementation. It also appears adaptable to regional and local network implementation. Simulation results indicate that the Blazenet performance is comparable for low-load operation to the ideal case of a nonblocking network, and is much better than that of the lossy networks. Specifically, Blazenet provides multigigabit per second data rates at low delay with good behavior under load using the connectionless datagram interface favored by computer communication. The use of source routing allows each switching node to make switching decisions on the fly, minimizing the switching delay. The use of a loopback channel, which effectively stores packets that are blocked at the switch, minimizes the packet loss under load without requiring additional memory within the switch. The absence of switching buffer memory makes photonic data path implementation feasible. The ability of the design to handle priority and multicast makes it attractive for a variety of traffic loads.

Blazenet demonstrates the feasibility of packet-switching in high speed networks. It is not necessary to resort to circuit-switching to handle the data rates made possible by optical fiber. In fact, when computer traffic has to be carried, packet-switching has some crucial advantages over circuit-switching, advantages that are emphasized in high-speed networks. In particular, a packet-switched network avoids the circuit setup delay and the wasted channel capacity that arises with circuit switching, especially with bursty computer traffic. In this vein, we note that many sources of traffic that appear as a stream at lower data rates appear as bursts at gigabit speeds.

We see Blazenet as a representative of a future class of networks that behave as passive "light pipes" for data, offering high throughput, low delay, and high reliability. With the introduction of this class of wide-area networks, we expect that the computer interfaces, rather than the networks, will become the performance and functionality bottlenecks of the communication process. However, further research and development are required. Today's state of the art in photonic switching permits a photonic realization only to a limited degree ([13]). Nevertheless, this limited realization can serve as a first step towards a future all-photonic communication network.

Appendix

Some Implementation Issues in Blazenet Design

A. Boundary Between Photonics and Electronics

Optical switching and processing of optical transmission opens new dimensions in future networking. Photonic implementation, as opposed to a conventional electronic implementation, offers increased switching speeds [14], [3]. In addition a network built out of optical components is less susceptible to electromagnetic interference and electromagnetic pulse and provides more secure transmission. Unfortunately, the state-of-the-art of photonic processing is still in its infancy. Large and fast memory in particular appears to be a difficult component to realize photonically. However, with the progress in photonic technology, processing in light of more and more functions becomes available. Consequently, a simple network node design is of great importance, if and when the state of the art of photonic processing advances to such a degree that such full photonic implementation will be possible. Blazenet switching node design has limited functionality (i.e., no routing and no flow-control) and thereby lends itself more toward photonic implementation when it becomes feasible.

Below, we give some speculations on a possible full photonic implementation in the future. In the future, full photonic implementation of Blazenet, the detection of fields in the packet format (such as the sync or the hop-selects filed) can be done by the optical delay-line signal processing ([15], [16]), setting/resetting/zeroing of fields in the packet (such as the token or the hop-selects) can be performed by modulating a fast switch to either transfer the input information or to override some of its fields, and information processing and computing can be done by photonic logic ([17]-[20]). Note that because there is little memory needed in the Blazenet switching node design, the control is composed mainly of logic. Signal regeneration (after amplification) can be done by an all optical regenerator ([21]).

B. Speed of Detection

There is a potential problem of the header bit recognition when the speed of the lines increases. To overcome this problem, we suggest the header bits to be of different, smaller rate than the data bits, thus allowing more time for decoding. Consequently, by keeping the header bit "duration" constant, the problem of header bits recognition is essentially independent of the actual line speed.

C. Multiplexing Several Loops on a Pair of Fibers

The wavelength dependency of the Lithium Niobate switches ([22]-[25]) is small enough to permit wavelength division multiplexing, as claimed in Section II. For example, the 2×2 switches described in [8] can operate at $\pm 20\%$ of their central frequency. Thus, a switch that operates at 1.55 μm has a bandwidth of about 0.31 μm. At 1.55 μm, 1 nm is roughly 125 GHz. Thus, the bandwidth of the switch is about 40 000 GHz (!). If the wavelengths are spaced by 1250 GHz (roughly 0.01 μm), one can accommodate as many as 32 colors. In this way, in Blazenet there can be, theoretically, 32 logical loops multiplexed on a couple of fibers.

References

[1] L. C. Blank *et al.*, "120-Gbit·km lightwave system experiments using 1.478-μm and 1.52-μm distributed feedback lasers," in *Proc. Conf. Opt. Fiber Commun.*, 1985, pp. 86-87.

[2] S. R. Nagel, "Optical fiber—The expanding medium," *IEEE Commun. Mag.*, vol. 25, pp. 33-43, Apr. 1987.

[3] F. Guterl and G. Zorpette, "Fiber optics: Poised to displace satellites," *IEEE Spectrum*, Aug. 1985.

[4] A. Bellman, "Switching architectures towards the nineties," in Proc. Int. Sem. Digital Commun., New Directions in Switching and Networks, Zürich, Switzerland, Mar. 11-13, 1986.

[5] M. J. O'Mahoney, "Semiconductor laser optical amplifier for use in future fiber systems," *IEEE J. Lightwave Technol.*, vol. 6, Apr. 1988.

[6] A. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[7] Vineet Singh, "The design of a routing service for campus-wide Internet transport," M.Sc. thesis, Lab. Comput. Sci., M.I.T., M.I.T./LCS/TR-270, Aug. 1981.

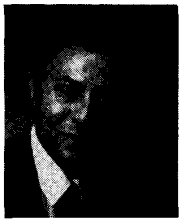
[8] OGW 2X2 Switch Data Sheet, Crystal Technol., Inc., 1060 East Meadow Circle, Palo-Alto, CA 94303.

[9] Z. Haas, "Packet-switching in future high-performance wide-area networks," Ph.D. dissertation, Elec. Eng. Dep., Stanford Univ., May 1988.

[10] Z. Haas and D. R. Cheriton, "A case for packet-switching in high-

performance wide-area networks," in Proc. SIGCOMM '87 Workshop, Stowe VT, Aug. 11-13, 1987.
[11] ——, "Blazenet: A high-performance wide-area packet-switched network using optical fibers," in Proc. IEEE Pacific RIM Conf. Commun., Comput. Signal Processing, Victoria, B.C., June 4-5, 1987.
[12] D. R. Cheriton, "Versatile message transaction protocol (VMTP)," SRI Network Inform. Cent., RFC1045, Feb. 1988.
[13] P. R. Prucnal, "All-optical ultra-fast networks," *SPIE Fiber Telecommun. Comput. Networks*, vol. 715, 1986.
[14] P. R. Prucnal, D. J. Blumenthal, and P. A. Perrier, "Photonic switch with optically self-routed bit switching," *IEEE Commun. Mag.*, vol. 25, May 1987.
[15] K. P. Jackson, S. A. Newton, B. Moslehi, M. Tur, C. C. Cutler, J. W. Goodman, and H. J. Shaw, "Optical fiber delay-line signal processing," *IEEE Trans. Microwave Theory Techniques*, vol. MTT-33, Mar. 1985.
[16] B. Moslehi, J. W. Goodman, M. Tur, and H. J. Shaw, "Fiber-optic lattice signal processing," *Proc. IEEE*, vol. 72, July 1984.
[17] T. E. Bell, "Optical computing: A field in flux," *IEEE Spectrum*, vol. 23, Aug. 1986.
[18] *Proc. IEEE*: Special Issue on Optical Computing, vol. 72, July 1984.
[19] S. D. Smith, "An introduction to optically bistable devices and photonic logic," *Phil. Trans. R. Soc. Lond.*, Great Britain, A-000-000, 1984.
[20] A. L. Lentine, D. A. B. Miller, J. E. Henry, and J. E. Cunningham, "Photonic ring counter differential logic gate using the symmetric self-electrooptic effect device," in Proc. CLEO, Anaheim, CA, Apr. 25-28, 1988.
[21] M. Jinno and T. Matsumoto, "All-optical timing extraction using a 1.5 μm self pulsating multielectrode DFB LD," *Electron. Lett.*, vol. 24, no. 23, Nov. 10, 1988.
[22] M. Sakaguchi and K. Kaede, "Optical switching devices techniques," in Proc. Int. Sem. Digital Commun., New Directions in Switching and Networks, Zürich, Switzerland, Mar. 11-13, 1986.
[23] H. S. Hinton, "Photonic switching using directional couplers," *IEEE Commun. Mag.*, vol. 25, May 1987.
[24] S. D. Personick, "Photonic switching: Technology and applications," *IEEE Commun. Mag.*, vol. 25, May 1987.
[25] W. A. Payne and H. S. Hinton, "Design of lithium niobate based photonic switching systems," *IEEE Commun. Mag.*, vol. 25, May 1987.

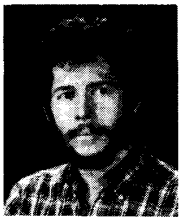
★



Zygmunt Haas (S'83–M'88) received the B.Sc. degree in E.E. from Technion in 1979 and M.Sc. in E.E. from Tel-Aviv University in 1985, both with "Summa Cum Laude." From 1979 till 1985 he worked for the Government of Israel. He received the Ph.D. degree in 1988 from Stanford University.

In 1988, he joined AT&T Bell Laboratories, Holmdel, NJ, where he is now a member of Technical Staff in Network Systems Research Department. His interests include high-speed communication, lightwave networks, and traffic integration.

★



David R. Cheriton (S'76–M'78) received the Ph.D. degree in computer science from University of Waterloo in 1978.

He is an associate professor of computer science at Stanford University. His research interests include distributed systems, computer communication, parallel programming and architectures, and office automation.

Dr. Cheriton is a member of the ACM.