The "Staggering Switch": An Electronically Controlled Optical Packet Switch

Zygmunt Haas, *Senior Member, IEEE*

***Abstract*—This paper presents an architecture and implementation issues of an "almost-all" optical packet switch that does not rely on recirculating loops for storage implementation. The architecture is based on two rearrangeably nonblocking stages interconnected by optical delay lines with different amounts of delay. We investigate the probability of loss and the switch latency as a function of link utilization and of the size of the switch. In general, with proper setting of the number of delay lines, the switch can achieve an arbitrarily low probability of loss. Growability patterns and extension of the design to the dense wavelength division multiplexing (WDM) case are also shown. In particular, we discuss an extension to the architecture whereby, through the use of WDM, the switch capacity may be increased several times, with only minor changes to the switch design. Additionally, issues involving practical implementation of such a switch are discussed. For example, we show a scheme that allows optical packet synchronization for the synchronously-operated switch. Using this scheme, the switch may be a central component in the design of future all-optical, packet-switched networks.**

I. Introduction and Motivation

IN "all-" and "almost-all" optical networks the data path is fully optical. In other words, the data bits (the payload) remain in the optical domain during their journey through the network. In "almost-all" optical networks, the control of the switching operation is done electronically. In contrast, in all-optical networks, the switching control is fully optical. The current state of optical processing and computing does not allow full implementation of an all-optical network. Thus, our work concentrates on the design of an "almost-all" optical network.

In this paper, it is assumed that optical networks have some advantage over their electronic counterpart. One of the features provided by photonics is the transparency, and refers to the fact that, except for the control information (i.e., the packet header), the payload may be encoded in an arbitrary format or at an arbitrary bit rate. This feature may have far-reaching consequences for expanding and upgrading of future networks. Nevertheless, it is beyond the scope of this paper to argue for the superiority of photonics over electronics, especially because many crucial parameters, such as future cost, are still unknown.

The main problem in the implementation of packet-switched optical networks is the lack of random access optical memory ([1]). Some networks cope with this shortcoming by introducing special architectures that either eliminate the need for local buffering or that reduce the buffers' size ([2]). Other networks, especially local area networks, rely on some (electronically based) reservation scheme that again avoids the use of optical buffers ([3]). However, this approach cannot be easily extended to a wider network span.[1] Therefore, the challenge is to propose an optical switch architecture design with the constraint that large optical storage is not feasible.

Several recently proposed switching architectures[2] are not suitable for optical implementation because of the limitation of optical buffering; i.e., lack of random access optical memory. Nevertheless, some come quite close to coping with the above limitation. For example, Starlite ([5]), by using the recirculation lines as (shared) buffers, could be optically implemented. However, with recirculation loop storage, every recirculation loop may require optical amplification because of the relatively large attenuation created by the switching element that controls the insertion (extraction) to (from) the recirculation loop.[3] Here, we present a switch architecture that employs the inherent capability of optical fiber for storage, yet does not rely on recirculation—the Staggering Switch.[4]

The paper is organized as follows: In Section II, the basic concept and elements of the Staggering Switch design are described. Because of the limitation on the length of this paper, detailed performance figures are not fully presented. These will be reported in another publication. (An interested reader is also referred to [6].) Nevertheless, a limited performance analysis is included in Section III. Extensions to the basic architecture of the Staggering Switch are discussed in Section IV. In particular, growability of the switch, both in the space and in the wavelength domains, is addressed. Additionally, a packet synchronization scheme, the packet flipping scheme, is introduced. An example of the control design is shown in Section V. A number of implementation issues are also considered in this section. Finally, some concluding remarks are included in Section VI.

II. The Staggering Switch Architecture

The Staggering Switch is an output-collision-resolution scheme that is based on a set of delay lines of unequal delay and is targeted at the (optical) packet-switched networks.

Manuscript received June 30, 1992; November 3, 1992.
The author is with AT&T Bell Laboratories, Holmdel, NJ 07733.
IEEE Log Number 9207744.

[1] Though, some attempts to do so has been considered ([4]).

[2] The reason for this increased activity is the Asynchronous Transfer Mode effort.

[3] Since an optical amplifier degrades the optical signal recirculation loop memory may hold information only for limited duration.

[4] The switch is named the Staggering Switch because of the structure of its storage elements and their effect on the switched packets.

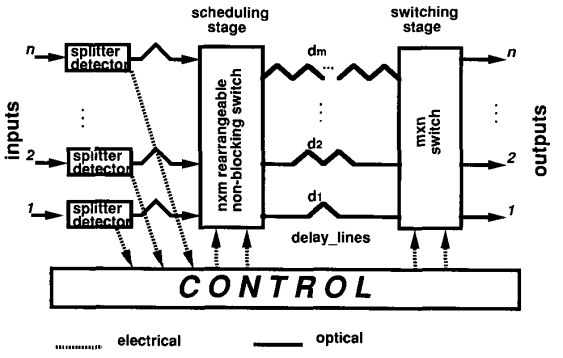


Fig. 1. An example of the Staggering Switch architecture.

A. The Hardware

An example of the the Staggering Switch architecture shown in Fig. 1 is based on two stages: the scheduling stage and the switching stage. Each stage is a reconfigurably nonblocking switching fabric. These stages are implemented with electronically controlled optical devices.[5] The scheduling stage is $n \times m$ and the switching stage is $m \times n$, where $m > n$. The scheduling stage is connected to the switching stage by m delay lines, $d_i, i = 1$ to m. The delay of the d_i delay line equals i packets.[6]

The control part of the switch operates on the "almost-all" optical principle described in [7]. The optical energy of each one of the n switch input lines is split immediately after its arrival; a small fraction of the energy is passed to the detector, converted to an electrical signal, and forwarded to the Control. The Control reads the header bits to determine the required routing for the packet and drives the scheduling and the switching stages of the switch.

Additional delay between the splitter and the scheduling module may be needed to compensate for the electronic Control processing time. It is assumed that:

1. all the packets are of fixed size[7]; τ [s],
2. time is slotted; initially we assume that each slot duration is τ,
3. the arrival of all the packets to the switch is synchronized.

The scheduling stage and the switching stage may be implemented as rearrangeably nonblocking networks (e.g., Beneš networks ([8] and [9])).[8] In rearrangeably nonblocking networks, any permutation of the inputs can be achieved at the outputs without blocking. Dilation of switching fabrics may be used in optical implementation as a way to reduce the crosstalk, especially for large number of inputs/outputs (e.g., the dilated Beneš networks [10]).

Numerous configurations of the Staggering Switch architecture are possible. We discuss here only the basic one.

B. The Operation of the Staggering Switch

The scheduling stage distributes packets to the delay lines in such a way that, in any time slot, no two packets arrive at the switching stage destined for the same output. In other words, the output collisions present at the inputs are resolved by delaying the colliding packets by different numbers of slots, so that when they arrive at the switching module, there are no output collisions.

The scheduling is done by the Control. The Control receives the header information from all the arriving packets and attempts to allocate as many of them as possible into the delay lines, d_i. The scheduling may be done according to the algorithm presented in the next section. To perform this algorithm, the Control needs to maintain knowledge of the "content" of the delay line at any time. Based on this information, the Control ensures that no collisions occur at the switching stage. Because of the statistical properties of the arrival process, some packets cannot be accommodated by the scheduling process, without violating the "no collisions at the switching stage" principle. Thus, some packets will be lost. The fraction of lost packets corresponds to some probability of packet loss. Section III evaluates the loss performance of the switch.

The switching stage permutes the outputs of the delay lines, so that the packets emerge at the required switch output.

C. The Scheduling Algorithm

In its basic form, the scheduling algorithm is as follows[9]:

```
while( ) do
  for input=1 to n do
  if (packet is present on input) then
    begin
    index := m;
    while (index ≥ 1) do
      if (column[index] is busy) then index := index − 1
      else if (no other packet to the same destination
          in the column[index]) then
    begin
    insert input packet in column[index];
    break;
    end;
  end;
end;
```

In other words, scanning the inputs sequentially, for each input packet the algorithm tries to insert the packet in the lowest possible delay line, subject to the two following conditions:

1. that no previous packet was inserted in the delay line in this time slot (i.e., the output is free),
2. that no other packet to the same destination exists in the column in which the packet is to be inserted.

[5] $LiNbO_3$ for example,

[6] We differentiate between packets and slots. A packet is the data that needs to be switched, while a slot is a frame into which the logical packet is inserted. Slot duration need not equal packet duration.

[7] Size, here, being the time duration. However, the number of bits in each packet may vary, depending on the actual data rate of the data field; i.e., the network may operate based on the field coding technique ([7]).

[8] The complexity of routing in large rearrangeably non-blocking networks may be quite high. In these cases, non-blocking networks may be required.

[9] Refer to Fig. 2 for definition of a column.

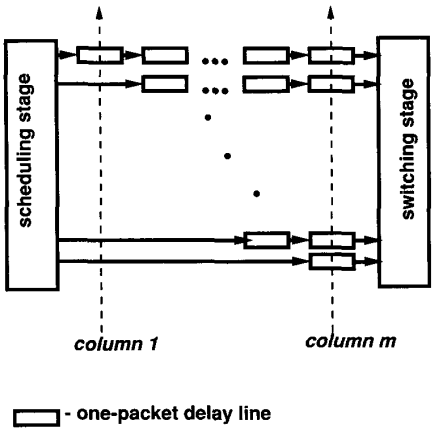


Fig. 2. Definition of scheduling algorithm "data structure."

This algorithm is referred to as the sequential biased assignment, since it gives higher priority to the lower numbered inputs by trying first to accommodate a packet from input 1, then from input 2, etc. Thus, the probability of blocking of input number 1 is 0 (i.e., $P_{\text{block}}(1) = 0$, and $P_{\text{block}}(i) < P_{\text{block}}(i+1)$). A variation of this algorithm is the sequential nonbiased assignment, where in each slot the order in which the inputs are accommodated is random. Other assignment possibilities exist. In particular, there is an optimal assignment based on the knowledge of the past and the future. Some of the more interesting possibilities are discussed in Appendix A.

III. Performance Evaluation

Packet switching structures are compared based on a number of parameters. The main figure of merit is the loss probability or blocking probability (i.e., the probability that a packet cannot be switched because of contention) under some assumptions on traffic intensity and statistics (usually uniform distribution of destinations). The capacity of a switch is the maximum throughput, again under some assumptions on traffic statistics. The switch latency is the average delay of a packet in the switch, when the switch operates under specific traffic load.

In this section, the two major switch performance criteria are considered: probability of blocking[10] and latency[11]. Both probability of blocking and latency depend on input line utilization[12] ρ; both increase with an increase in line utilization. Probability of blocking, P_{block}, is directly related to the normalized switch throughput, T, by: $T = \rho \cdot (1 - P_{\text{block}}(\rho))$. The normalized switch capacity C is its maximum throughput. Thus assuming that the maximum throughput is achieved at $\rho_m, C = (1 - P_{\text{block}}(\rho_m))$.[13]

[10] Probability of blocking is defined as the probability that a randomly chosen packet cannot be scheduled and is dropped.

[11] Latency is defined as the average time (in slots) that a packet is delayed in the switch.

[12] Line utilization is the probability that an input slot contains a packet.

[13] The results of our simulations strongly suggest that under the assumptions outlined in the next subsection, $\rho_m = 1$.

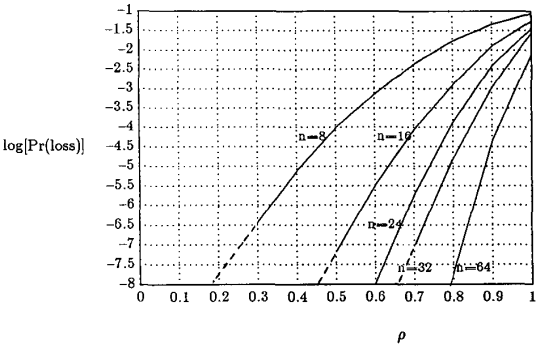


Fig. 3. Simulation results of loss probability versus loading: $n = m, n$ parameter.

A. Probability of Blocking—Throughput

To evaluate the blocking performance of the switch, the following were assumed:

1. The destination addresses are uniformly distributed.
2. The traffic is not time-correlated, i.e., nonbursty traffic. (This restriction is alleviated later.)
3. There is no correlation between packets arriving on any two inputs; no arrival time correlation and no destination correlation.
4. Sequential biased scheduling is performed.
5. All packets have the same routing- and loss-priorities.

Note that since the traffic on each input is assumed to be uniformly distributed and uncorrelated in time, the throughput of the switch in the cases of sequential biased and sequential non-biased assignments is the same.

The probability of blocking is evaluated by simulation as a function of line utilization with the size of the switch n as a parameter. Fig. 3 shows the simulation results for switches of size $8 \times 8, 16 \times 16, 24 \times 24, 32 \times 32$, and 64×64. As is evident from the graph, the larger the switch, the lower the loss probability for a given line utilization. Thus, for example, at $\rho = 0.7, P_{\text{block}} \approx 10^{-4}$ for the 16×16 switch, while $P_{\text{block}} \approx 10^{-7}$ for the 32×32 switch.

An additional parameter is the number of delay lines m. As can be expected, increasing the number of delay lines lowers the loss probability.[14] This is shown in Fig. 4, where n is kept constant at 16 and m is varied from 16 to 32. The effect of the increase in m is quite dramatic. For example, at $\rho = 0.8$, when m is increased from 16 to 32, the P_{block} decreases from $1.2 \cdot 10^{-3}$ to $6 \cdot 10^{-7}$ (i.e., about 3–4 orders of magnitude). Thus m can serve as a very effective design parameter to achieve a desirable level of performance.

Since the Staggering Switch operates in the deflection routing mode, packets are, in fact, not lost but misdirected. In other words, packets are accommodated in the delay lines in such a way as to maximize the number of packets that conform to the rule that "no two packets in the same column are destined for the same output." But, since the scheduling module only permutes the inputs, some packets will be forwarded to delay lines in violation of the above rule. These packets will be

[14] Since as m increases, there is more "buffering" available.

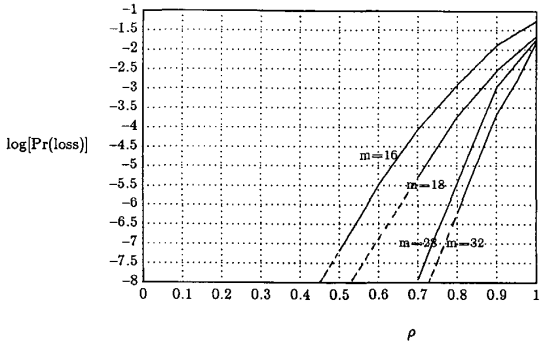


Fig. 4. Simulation results of loss probability versus loading: $n = 16, m$ parameter.

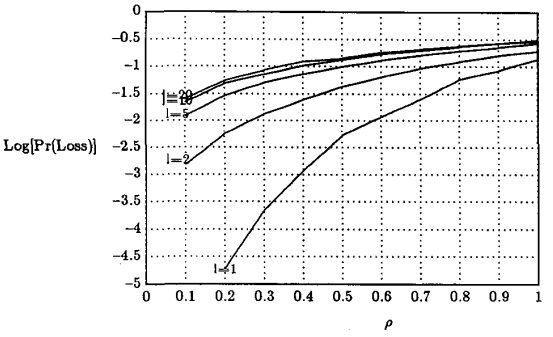


Fig. 5. Probability of loss with time-correlated traffic: $m = n = 4, \Delta = 1$.

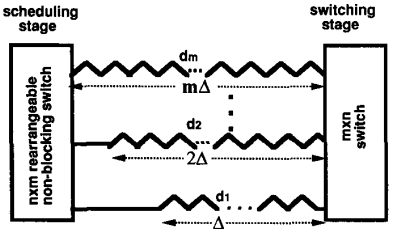


Fig. 6. Change in the delay lines to accommodate bursty traffic.

switched to a wrong output at the switching stage, that again simply permutes the delay lines to the outputs. Thus, even though we consider here such misdirected packets as lost, in practical, multihop networks, these packets may be still correctly delivered after subsequent nodes redirect them back to their original destination.

B. Bursty Traffic

The assumption of nonbursty traffic is unrealistic for future high-speed networks. The following set of simulation results shows the effect of time correlation (i.e., burstiness) on the switch performance. As a function of time, the input traffic is composed of bursts of packets destined to the same output. These bursts are followed by an idle period that can be of length zero. The average burst length is l, and $l = 1$ corresponds to traffic with no time correlation. (Refer to Appendix B for a description of the bursty traffic model.)

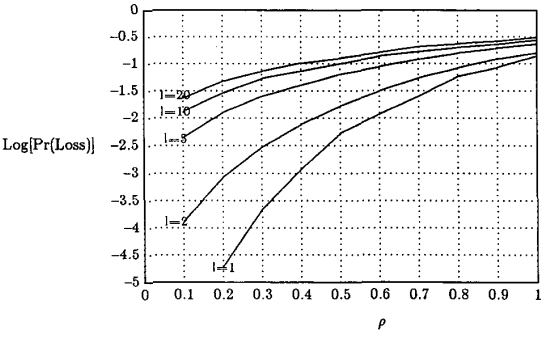


Fig. 7. Probability of loss with time-correlated traffic: $m = n = 4, \Delta = 2$.

Fig. 5 shows the performance of the 4×4 Staggering Switch in bursty traffic, with the burst length as a parameter ($l = 1$, 2, 5, and 10). As can be noted, the performance degrades rapidly with an increase in the burst length. Thus, the basic Staggering Switch architecture is not well suited to time-correlated traffic. The following change to the switch design allows us to reduce the ill-effect of the time-correlated traffic considerably. The change is described in Fig. 6. Let Δ_i define the difference in the length of the i^{th} and the $(i+1)^{\text{th}}$ delay line, i.e.,

$$\Delta_i = d_{i+1} - d_i, \qquad 1 \le i \le m-1. \tag{1}$$

Thus, in the basic Staggering Switch design, $\Delta_i = \Delta = 1$. Increasing Δ, reduces the effect of traffic time correlation. This is shown in Fig. 5 and Figs. 7–9, where $\Delta_i = \Delta$ takes values of 1, 2, 5, and 10. As can be observed from these figures, as Δ increases, the performance of bursty traffic tends more to the performance of the uncorrelated case ($l = 1$). It should be noted that increasing Δ corresponds to an increase in the length of the delay lines but has no effect on the number of the delay lines or the size of the scheduling and the switching modules. This is an important fact, since the hardware cost is determined by the size of the scheduling and switching modules. Thus by properly sizing Δ, the increase in probability of loss due to traffic correlation can be effectively eliminated. The penalties for increasing Δ are: the cost of fiber (which is quite small); the increased attenuation (which is probably negligible because of the dominant attenuation of the $LiNbO_3$ modules); the increased chromatic dispersion for very high-speed design; and the increased switch latency (i.e., the switch latency is Δ times larger than the basic design with $\Delta = 1$). The scheduler operation (i.e., the size of the scheduling) remains uneffected by the change in Δ's, as long as the number of delay lines remains constant.

C. Reduction in Column Dependency

The packet loss probability can be improved if some of the dependency between the columns can be reduced or eliminated. The dependency is created because the system has memory. In other words, if a packet is "rejected" from a column and accepted at a subsequent column, then future packets "rejected" from the first column will be no longer independently "rejected" or "accepted" at the second column.

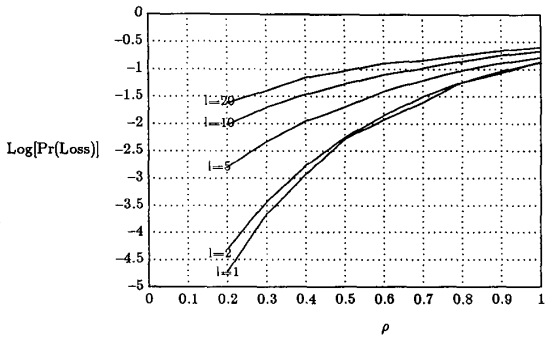


Fig. 8. Probability of loss with time-correlated traffic: $m = n = 4, \Delta = 5$.

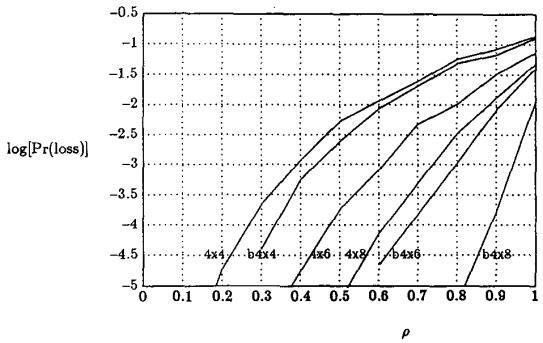


Fig. 10. Comparison of packet loss probability for the $\Delta = 1$ and the reduced-dependency cases.

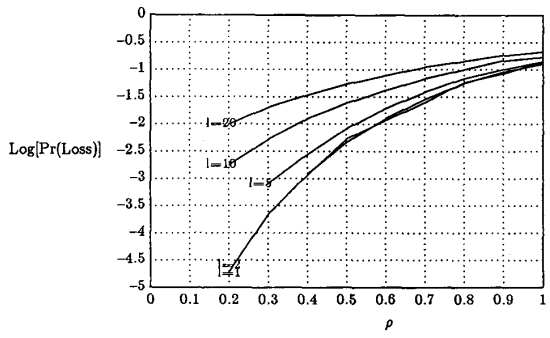


Fig. 9. Probability of loss with time-correlated traffic: $m = n = 4, \Delta = 10$.

The Δ's can be staggered so that only once (during the "life" of the columns) will the same two columns "accept" packets at the same time. Such an arrangement reduces the column dependency and exists, for example, if

$$\Delta_i = 2^i. \tag{2}$$

Other arrangements with smaller Δ's than in formula (2), are possible.

Fig. 10 demonstrates the improvement of the probability of loss for $n = 4$, when the Δ's are arranged according to formula (2) and uncorrelated traffic is assumed. In this figure, the curves are labeled $n \times m$. Prefix b indicates that arrangement (2) was used. As can be observed from the figure, significant improvement can be achieved by the above arrangement, especially for low utilization. It should be pointed out that the improvement is not due to additional storage created by the arrangement of formula 2 since, as can be easily verified, an equal increase in all the Δ's (and thus an increase in the storage capability) provides no improvement in the probability of loss when uncorrelated traffic is assumed.

D. Switch Latency

The latency of the Staggering Switch, D, depends on ρ, n, and m.[15] In general, an increase in ρ, n, and m results in an increase in the average packet delay through the switch. This behavior is demonstrated in Figs. 11 and 12. The delay remains relatively constant as a function of m for small values of ρ. The reason is that, for low utilization, the "extra" delay lines are rarely occupied. However, as ρ increases, the longer delay lines become more and more occupied, and the average latency increases. Note that the latency for smaller values of m is lower. This happens because latency is averaged only for successful packets; also for large values of m and high utilization, many packets are scheduled on longer delay line, contributing to higher latency numbers. The maximum and minimum switch latencies are m and 1, respectively. The average latency at $\rho = 1$ is $(m + 1)/2$.

It should be emphasized that switch latency is not an issue in networks with wide span and small packet size, since the propagation delay of long distance fiber may be several orders of magnitude larger than the actual switching delay. For example, with a 64×64 switch size and 100 km between switching nodes, the propagation delay is 500 μs, while the maximum switch latency is 64 packets or approximately 27 μs, for 53-bytes, 1 Gb/s packets. Of course, for a smaller network span, larger packets, or large values of Δ 's, latency may be an important factor.

IV. Extensions to the Basic Switch Architecture

A. Packet Resequencing

The Staggering Switch may resequence packets. Packets arriving on the same input may end up in the reverse order on the same output, because the packet arriving first may be placed in a longer delay line than the packet arriving later.

Packet resequencing may be solved either in the switch itself or on the higher protocol layers. The transport layer, for example, may assume the function of packet resequencing (e.g., Transmission Control Protocol [11], for example). For some applications, packet sequencing is not an issue, and no action is required.[16] ATM networks require cell order to be preserved, since there is no internal mechanism in ATM to

[15] On Δ's, more generally.

[16] For example, database queries where each request or response is contained in a single packet, or voice packet that undergo build-out procedure to eliminate the jitter in the network delay.

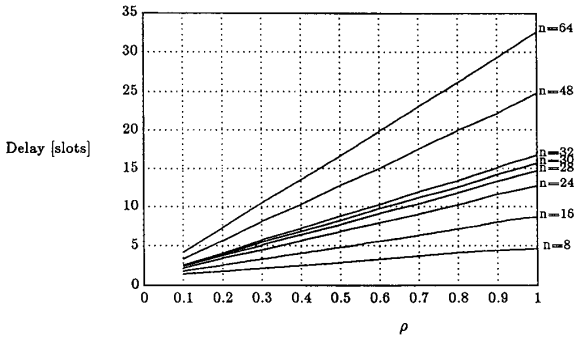


Fig. 11. Simulation results of the Staggering Switch latency: $m = n, n$ parameter.

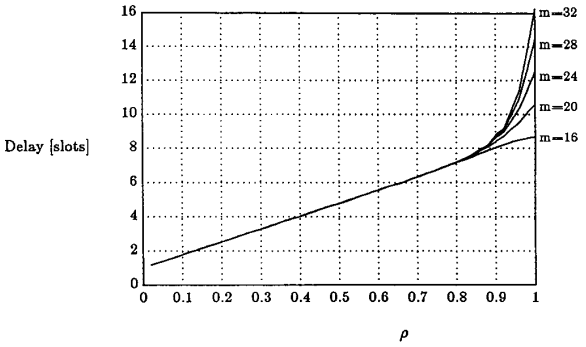


Fig. 12. Simulation results of the Staggering Switch latency: $n = 16, m$ parameter.

restore the order of cells.[17] Thus, for this kind of network, some solution to restore packet order is required.

Packet ordering in the Staggering Switch could be preserved by modifying the scheduling algorithm. For example, if a packet arrived on input k to output j and was scheduled on d_i, then the packet arriving on the same input to the same output in the next slot can be scheduled only on d_h, where $h > i$. In general, if the last packet from input k to output j arrived s slots ago and was placed on d_i, then a packet from the same input arriving in the current slot can be placed on d_h, where

$$\max(1, i - s + 1) \leq h \leq m. \tag{3}$$

Thus, for each input, a pointer must be maintained to indicate the range of the delay lines on which a packet arriving on this input may be placed. Alternatively, a pointer may be kept for each output, in which case condition (3) should be applied to output lines. However, Fig. 13 shows a considerable increase in the probability of loss for the above two schemes (the curves "input" and "output"). When a pointer is maintained for every input-output pair and condition (3) is preserved for each such input-output pair (the curve "inp-out"), the performance degradation is minimal, as shown in the same figure.[18] However, n^2 pointers are required in the latter case. Since, as pointed out before, some traffic may not require packet ordering, the routing may be based on the traffic type—traffic that does not require order preservation may be routed according to the regular scheduling algorithm, while order preserved traffic will be scheduled based on the above pointer-based scheduling scheme.

B. Growing the Staggering Switch in the Space Domain

A large-sized Staggering Switch may not be easy to implemented because of practical problems, such as attenuation and cross-talk. Also, in a large switch, the electronic control has a limited ability to process the large flux of incoming packets. Thus, a scheme is required that allows building of a large switch using relatively smaller modules. One way to grow the Staggering Switch is shown in Fig. 14. An $nq \times nq$ switch is

[17] ATM is a connection-oriented interface and assumes that the underlying subnet routes all the packets the same way.

[18] The curve "4×4" is for regular sequential scheduling in an $n = 4, m = 4$ switch.

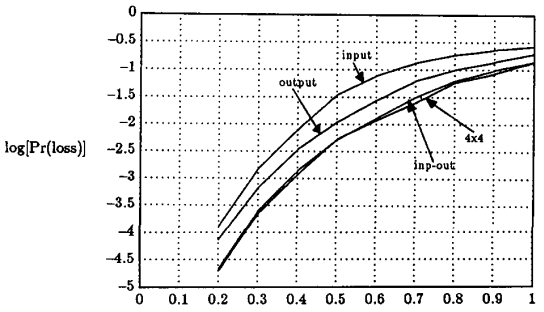


Fig. 13. Performance of order-preserving scheduling algorithms.

built out of $2q$ $n \times n$ switches. Since each path passes through two stages, the total loss probability is given in terms of the loss probability of a single $n \times n$ module, $P_{\text{loss}}^{n \times n}$, by:

$$P_{\text{loss}}^{\text{total}} = 1 - (1 - P_{\text{loss}}^{n \times n})^2. \tag{4}$$

Thus, for example (see Fig. 15), for a 16×16 switch built out of 4×4 with $m = 4$ modules (the "$4 \times (4 \times 4 \times 4)$" curve), the probability of loss is considerably higher than for the single stage 16×16 (the "16×16" curve) switch. However, by increasing m, one can obtain the same blocking probability as the pure 16×16 case. As shown in Fig. 15, the probability of loss for the 16×16 switch built out of 4×4 modules with $m = 10$ (the "$4 \times (4 \times 10 \times 4)$" curve) may be a relatively good approximation of the 16×16 curve, especially for large utilization. Additionally, the probability of blocking for a 16×16 switch composed of 4×4 modules with $m = 16$ (the "$4 \times (4 \times 16 \times 4)$" curve) is considerably lower than the probability of loss of a single-stage 16×16 switch with $m = 16$. Thus, by using the space domain growability scheme, one can build a 16×16 switch by using smaller modules of 4×4 with $m = 10$ and obtain loss performance similar to that of a one-stage 16×16 switch with $m = 16$. Alternatively, one can improve the performance of the one-stage 16×16 switch with $m = 16$, by using 4×4 modules with $m = 16$ modules.

The routing through this $nq \times nq$ switch is quite simple, since there is only a single path between each input and output. Thus

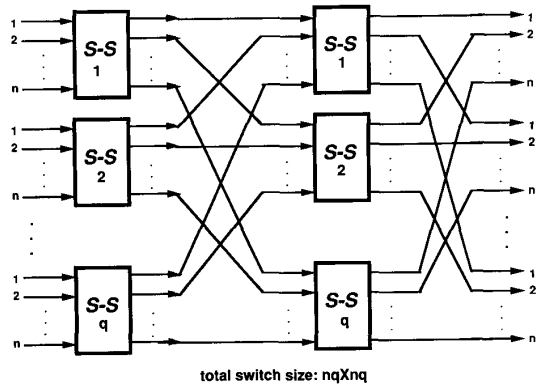


Fig. 14. Growing the Staggering Switch in the space domain.

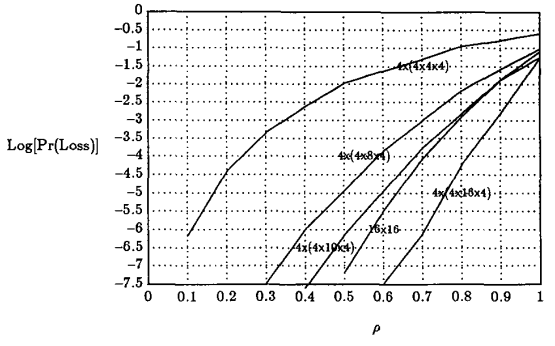


Fig. 15. Comparison of direct implementation with growability in the space domain.

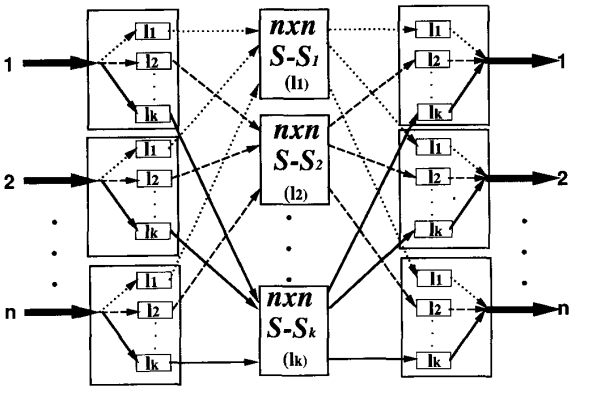


Fig. 16. Expansion of the Staggering Switch architecture in the wavelength domain.

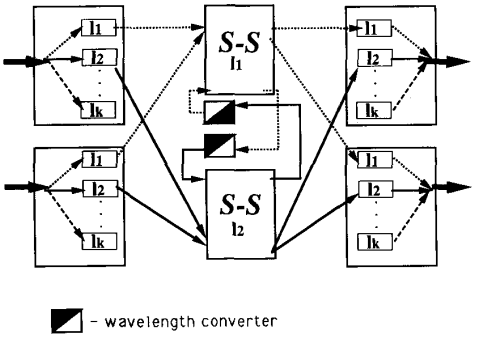


Fig. 17. Inter-wavelength switching.

routing can be performed in a distributed manner (i.e., each switching module may perform the routing locally).

C. Growing the Stagggering Switch in the Wavelength Domain

The Staggering Switch may be extended in the wavelength domain in two ways. First, in the WDM case where each one of the n trunks carries k wavelengths, each trunk is demultiplexed to k channels and the n channels of the same color are grouped together for switching in a single $n \times n$ switch (shown in Fig. 16). Following the switching operation, the outputs of the switches are multiplexed onto n outgoing trunks. In this arrangement, any single switch needs to operate only at a single color. A total of k switches are required. If inter-wavelength switching is required, wavelength converters must be included, as shown in Fig. 17.

Another way to incorporate the wavelength multiplexing in the design of the Staggering Switch is by representing the packet simultaneously on several wavelengths. This scheme is shown in Fig. 18, where four colors are used in parallel to represent a single nibble of information. In this example, each one of the four wavelengths carries a 2.5 Gb/s stream, totaling 10 Gb/s per input line. Since the $LiNbO_3$ optical bandwidth is large enough to accommodate four Dense-WDM wavelengths, parallel 4 bit switching can be performed at the same time. In this way, the capacity of the switch can be quadrupled with no change to the switch design. The required changes are in the peripheral devices that generate the four parallel channels and multiplex/demultiplex them on a single fiber. Practically, much more than four wavelengths can be accommodated, considerably increasing the switch capacity. Such an arrangement can be employed where the use of extremely fast transmitters/receivers is prohibitively expensive.

D. Slot Synchronization

When the Staggering Switch is used as a part of an all-optical network, it is necessary either to ensure that all the inputs to the switch are synchronized (i.e., packets arriving at different switch inputs are aligned at the switch) or to operate the switch in an unsynchronized manner. If the switch is to be asynchronously operated, the switching and scheduling modules should be nonblocking elements, not reconfigurably nonblocking elements as in the synchronized case, since in asynchronous operation reconfiguration is not possible while packets are being forwarded to the delay lines. Thus using reconfigurably nonblocking elements for asynchronously operated switch results in an additional penalty because of the blocking in the scheduling and switching modules. Nonblocking modules with small crosstalk are harder and more expensive to manufacture.

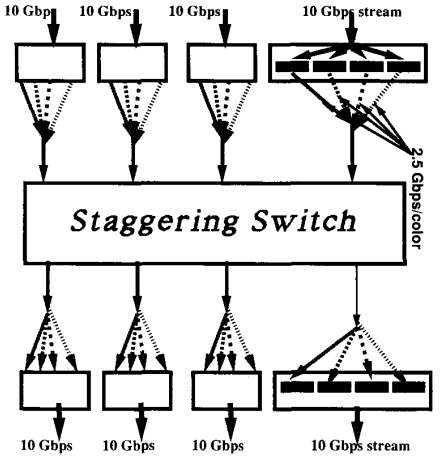


Fig. 18. Expansion in the wavelength domain by parallel transmissions.

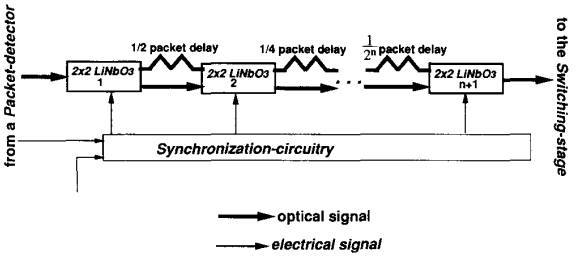


Fig. 19. A scheme for packet synchronization.

When the switch operates synchronously, the scheme in Fig. 19 can be used to synchronize packets arriving at different inputs.[19] In this scheme, delay lines with delay equal to a fractional part of a slot are connected in tandem. The synchronization circuit, after comparing the packet time of arrival with the phase of the local slot clock, $\text{clock}_{\text{slot}}$, generates the appropriate setting of the delay line, so that the input stream is aligned with the local clock. If the delay variations in the network are slow, the adjustments to the delay line settings occur infrequently. This sychronization scheme suffers relatively large attenuation because the optical signal may travel in and out for the $LiNbO_3$ wafer several times.

To overcome this shortcoming, we propose a new synchronization scheme, termed packet flipping. The scheme is based on time slots that are twice as long as the actual duration of the packet. Thus, the network capacity is reduced to 50%.[20] Refer to Fig. 20 for the explanation of the scheme's operation. It is assumed that each switching node in the network has a local clock, $\text{clock}_{\text{frame}}$, with period of 2τ, where τ is the duration of the packet. We call this 2τ-long slot a frame. The clocks at different switches (nodes) are unsynchronized with each other. Moreover, it is assumed that in each switching node (and network interface) only one packet can be placed

[19] Other configurations are possible.

[20] This is in line with the philosophy that some part of the enormous bandwidth can be "wasted" to provide simpler control or operation of the all-optical networks.

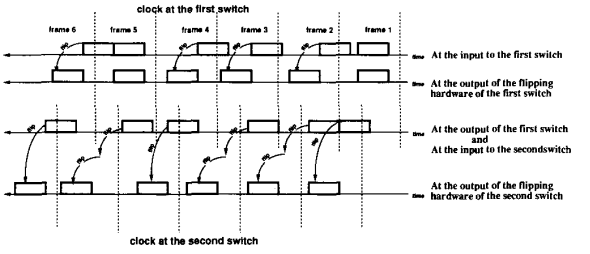


Fig. 20. The packet flipping scheme for packet synchronization.

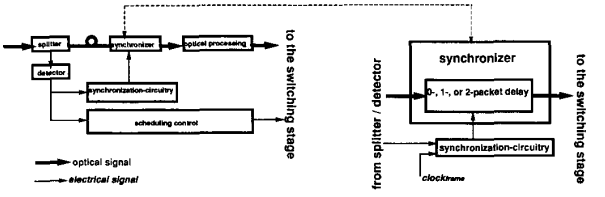


Fig. 21. The components for the packet flipping scheme.

in a (locally defined) frame. Within this frame, however, the packet may float. If such traffic is presented to the Staggering Switch with slot size equal to the frame size (i.e., double the slot size), the switch preserves the "one packet per frame" rule. However, if traffic coming from the output of the switch is received at the next switching node with an unsynchronized local frame clock, the "one packet per frame" rule may be violated, as shown in Fig. 20. To correct this, so that there again is a single packet per frame aligned with the new local frame clock, the hardware shown in Fig. 21 is introduced. The central component is the synchronizer.[21]

The synchronizer can be built from two delay modules, as shown in Fig. 22(a), where each module can either add no extra delay or a delay of τ to the optical signal. Alternatively, the sychronizer can be composed of three 2×2 modules interconnected by two sets of two delay lines; one with delay of τ and one with no delay, as shown in Fig. 22(b). Comparing the two sychronizer designs, the maximum loss of the two-modules design is equivalent to four times the loss of a single 2×2 module, while the three-modules design loss is only three times the single module loss. Moreover, the loss of the two-modules design is not constant and depends on the total delay required. However, the cost of the three-modules synchronizer hardware is considerable higher than the two-modules one.

If a packet does not fall in between two frame clocks, or if there are two packets per frame, an extra delay is added to correct this situation. A proof to show that the two packet delays can restore the "one packet per frame" characteristic is given in Appendix C. Intuitively, when the "original" frame clock of the traffic is offset by less than τ, a single packet delay will restore the traffic "one packet per frame" characteristic; when the offset is longer, two packet delays are required. The

[21] The synchronizer is placed between the splitter and optical processing unit. The purpose of the optical processing unit is to condition the optical signal, so that it is suitable for the $LiNbO_3$ modules; e.g., proper polarization.

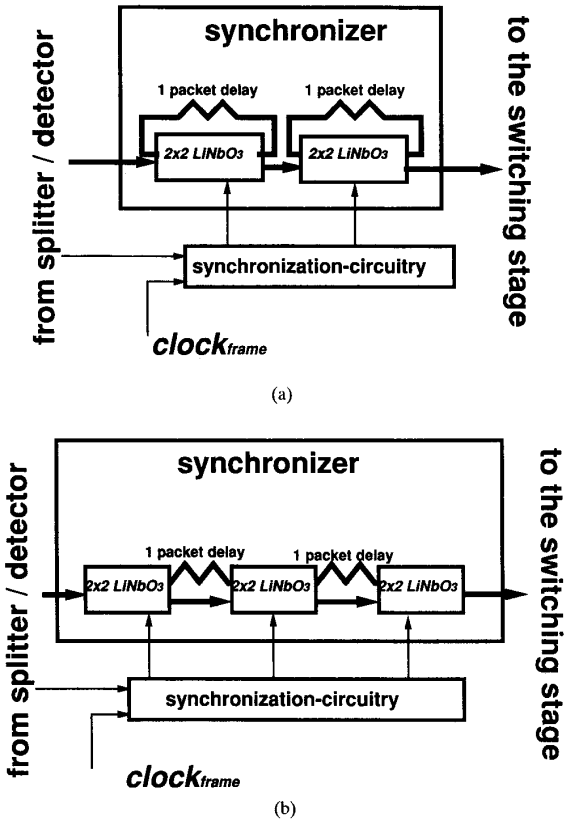


Fig. 22. Two possible designs of the synchronizer.

packet flipping scheme reduces the number of times the optical signal needs to leave the $LiNbO_3$ wafer compared with the scheme in Fig. 19 at the expense of reduced bandwidth.

V. Implementation Issues

A. Some Practical Considerations

As discussed in [12], 16 × 16 rearrangeably non-blocking modules were built with an optical power loss of 13.4 dB per module. Thus, connecting two of these modules[22] yields attenuation of 29 dB. This is a satisfactory power budget at 2.5 Gb/s per line. However, if the switch is only one of several switches that a packet is going to pass through (as in a multi-hop network), some sort of amplification is required. Moreover, since packets arriving at a switch may have traveled on different routes, the header receivers may require considerable dynamic range. A possible solution is a single stage of optical amplification ([13]). Another solution is electrooptic amplification, described in [14], in which an optical signal is detected, amplified as an analog signal, and converted back to light. Threshold amplification is an additional possible feature of the electrooptic amplification technique, reducing the Control receiver's large dynamic range problem. Moreover, the problem of crosstalk accumulation[23] is virtually eliminated by using this amplification technique.

Since the $LiNbO_3$ modules are polarization dependent, either polarization maintaining fibers or polarization controllers inserted at the input to each one of the two modules are required. Alternatively, polarization independent switches may be used ([15], [16]).

The high-speed receivers at the switch output need to be properly designed. The required features are: DC-coupling (or use of another scheme to avoid the DC-wandering problem) and proper dynamic range (to compensate for variations in optical path lengths through the switching modules and through the different delay lines). Burst receivers described in [17] were designed to cope with these problems.

B. Hardware Implementation of the Scheduling Algorithm

An example of the Control implementation is shown in Fig. 23 and is composed of the data-base, the scheduler, the x- and y -converters, and the path hunting memories. The data-base stores the representations of the packets' destinations of the packets that are in the delay lines. It is composed of an array of shift-registers that mimic the flow of the packets in the delay lines. The scheduler, which can be implemented as combinatorial logic or as an array of look-up memories, accepts the destinations of the inputs (vector $\underline{i}$) and the representation of the content of the delay lines from the data-base (through the vector $\underline{c}$). The scheduler generates the vector $\underline{b}$ that contains the representation of the accepted packets and is fed back into the data-base for storage in the shift-registers. The x-converter changes the representation of the sheduling information (vector $\underline{b}$) into a more concise form (vector $\underline{e}$). This information is then used by the path-hunting memory to determine the actual setting of the scheduling module. A similar operation is performed **in parallel** for the switching module, resulting in the vector $\underline{y}$ that contains the setting of this module. For large switch size, the path-hunting information may not fit into reasonably sized memory and may require further hardware to implement the appropriate algorithm.

Because the operation of the Control hardware may require a considerable fraction of a time slot (especially for small, ATM-like packets and high bit-rate), the operation may be pipelined. In the simplest form, the calculation of the setting is done one step ahead (i.e., while the transmission of the current packet takes place, the Control computes the setting for the set of packets in the next slot). Thus, the vector $\underline{i}$ that contains the destination information for the next slot is supplied at the beginning of a slot, and the computation of the next slot setting is completed before the end of the current slot. During the guard-bands, the actual reconfiguration of the modules takes place in parallel with updating the data-base with the information from the vector $\underline{b}$.

[22] For $m = 64$, 1 Gb/s line rate, ATM size packets of 53 bytes, and fiber loss of 0.2 dB/km, the total fiber loss (≈5 km) is 1.0 dB. Adding additional 1 dB for connectors results in total interconnection loss of ≈2 dB.

[23] Based on [12], the extinction ratio of the 16 × 16 switch is better than 20 dB. Thus, it is necessary to ensure that when a packet passes through several switches, the cross-talk does not accumulate to levels that may considerable interfere with the signal detection.

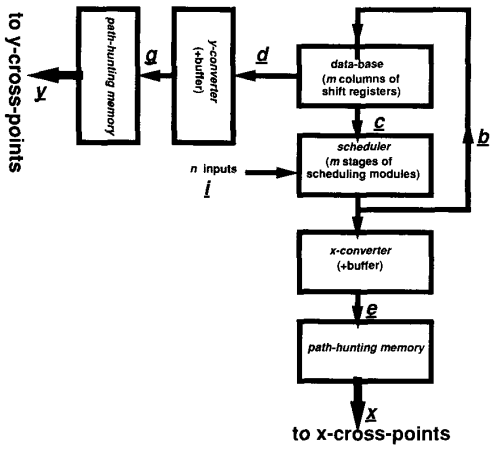


Fig. 23. Implementation of the Staggering Switch Control.

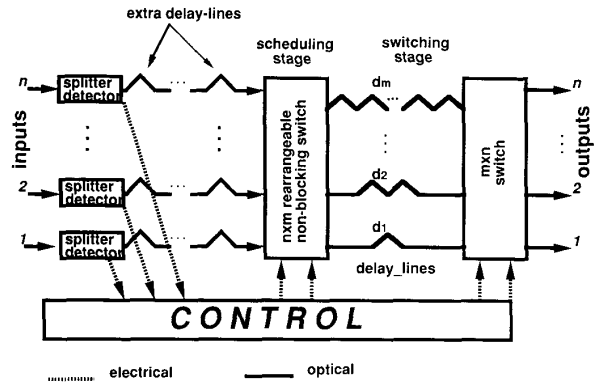


Fig. 24. Hardware configuration for non-casual scheduling algorithm.

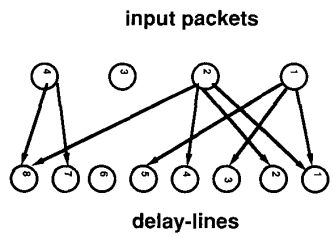


Fig. 25. The scheduling as maximal matching problem.

VI. Concluding Remarks

We have presented here an "almost-all" optical switch architecture that does not rely on recirculating loops for storage implementation. Our architecture is based on two rearrangeably non-blocking stages interconnected by delay lines with different amounts of delay; the Staggering Switch architecture. We have investigated the probability of loss as a function of link utilization and the size of the switch. In general, with proper setting of the number of delay lines, the switch can achieve arbitrarily low probability of loss. The latency characteristics of the switch were also investigated. Moreover, a scheme to reduce the effect of traffic burstiness was presented. Growability in space and wavelength domains were discussed. A packet synchronization scheme was introduced. A possible design of the electronic Control circuit was shown. We believe that the Staggering Switch may become the fundamental building block of future, wide-area, all-optical networks as the network bit-rate increases and packet switching becomes the switching scheme of choice in these networks.

Appendix A
The Effect of the Scheduling Algorithm

The scheduling algorithms may be classified based on the amount of information fed into the algorithm. In particular, algorithms can be causal or non-causal, if the information that the algorithm relies on is restricted to the past information or not, respectively. Thus, if an algorithm has knowledge about future arrivals, it is noncausal. Non-causal algorithms may be implemented by delaying the input information as in Fig. 24.

The scheduling algorithm may have a considerable effect on the performance of the switch, both in the probability of loss and the latency. The scheduling algorithm described in Section II-C, referred to as the sequential algorithm, is a causal algorithm. It is, however, not an optimal algorithm for achieving the lowest probability of loss. The problem of maximizing the number of packets that can be accommodated at a given slot can be viewed as the maximal matching on bipartite graphs, as shown in Fig. 25. In this example, the four left nodes are the four inputs, the eight right nodes are the eight delay lines in the 4×4 Staggering Switch with $m = 8$. There is a link from left node i to right node j, if a packet from input i can be accommodated in column j. Maximizing a matching on this bipartite graph corresponds to the maximum number of packets that can be accommodated in this slot. Finding a maximal matching will not, in general, correspond to the minimal loss probability, since, obviously some maximal matching (i.e., the ones that place the packets on lower numbered delay lines) are better than others. This is shown in Fig. 26, in which the performance of three algorithms is shown: the sequential algorithm, the maximal algorithm which randomly chooses between the maximal matching, and the l-maximal, which gives more weight to the maximal matching with lower numbered delay lines. The l -maximal algorithm may not necessarily be the optimum. More extensive results on the comparison between the various scheduling algorithms will be reported in a separate future publication.

Appendix B
Model for the Bursty Traffic

A realistic traffic model needs to account for some time correlation in the traffic stream. In other words, if a current packet on input i is destined to output j, then there is some probability greater than $1/n$ that the next packet on i is also destined to j, where n is the number of inputs. Thus, as a function of time, traffic on each input is composed of bursts of packets destined to the same output. These bursts are

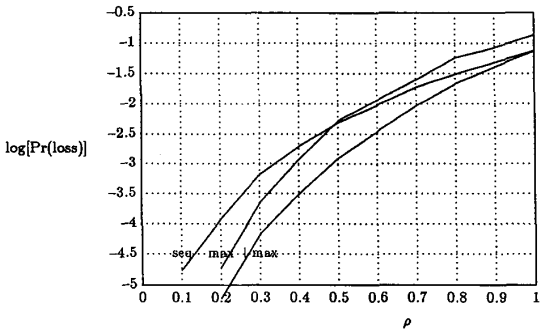


Fig. 26. Comparison of several scheduling algorithms.

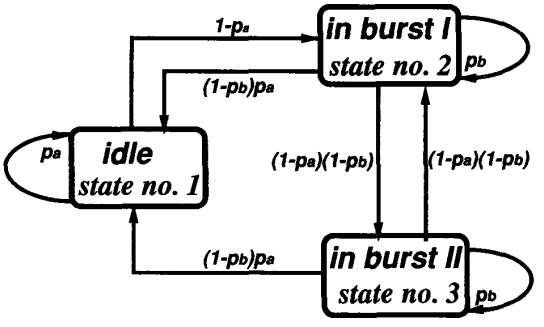


Fig. 27. Markov model for bursty traffic.

followed by an idle period that can be of length zero (i.e., two back-to-back bursts).

Time correlation of traffic on each input is modeled on the Markov chain, shown in Fig. 27. The chain is composed of three states: idle, in burst I, and in burst II. The system is in the idle state when no packet arrives in the current slot. With probability p_a, no packet will also arrive in the next slot, while, with probability $1-p_a$, a new burst will begin and the system will transfer to the state "in burst I." While in this state, with probability p_b, the next arrival will be part of the burst (i.e., destined to the same destination), or the burst will terminate with probability p_a. The termination of the burst can occur in two ways: by starting a new burst (i.e., to another destination), in which case the system will transfer to "in burst II" state, or by going to the idle state. The probability of the first case is $(1-p_b)\cdot(1-p_a)$, and the probability of second case is $(1-p_b)\cdot p_a$. The behavior of the system in the state "in burst II" is similar to that of the state in burst I.

Equation (5), seen at the bottom of the page, corresponds to the steady-state solution of the above Markov chain (π_i is the steady-state probability of the system being in state i).

Solving (5) results in:

$$\pi_1 = \frac{p_a\cdot(1-p_b)}{1-p_a\cdot p_b},\quad \pi_2 = \frac{(1-p_a)}{(1-p_a\cdot p_b)\cdot(2-p_a)},$$
$$\pi_3 = \frac{(1-p_a)^2}{(1-p_a\cdot p_b)\cdot(2-p_a)}. \tag{6}$$

First, we calculate the average link utilization, ρ, which equals the fraction of the time that the system is not in the idle state:

$$\rho = 1-\pi_1 = \frac{1-p_a}{1-p_a\cdot p_b}. \tag{7}$$

The average burst length, l, is calculated by observing that the probability of burst length k is:

$$Pr(k) = (1-p_b)\cdot p_b^{k-1},\quad k\geq 1. \tag{8}$$

Thus the average burst length is:

$$l = \sum_{k=1}^{\infty} k\cdot Pr(k) = \frac{1}{1-p_b}. \tag{9}$$

APPENDIX C
PROOF OF THE PACKET FLIPPING ALGORITHM

In this Appendix, we assume that all clocks are of equal period 2τ. We define s-characteristic of traffic with respect to a specific clock, as having the following property: **All the packets in the traffic stream are positioned between the clock ticks.**

In other words, the traffic might have been generated in such a way that there is only one packet per frame (2τ-long). The packets may, however, "float" within the frame.

In this Appendix, we prove that s-characteristic traffic with respect to any clock can be made s-characteristic with respect to a specific clock by the hardware shown in Fig. 21. It follows, as a conclusion from this proof that the s-characteristic of traffic with respect to local clock can be preserved at each switch by having the hardware in the Fig. 21 in front of each switch.

Consider the time diagram in Fig. 28. This figure (case 1 and case 2) shows a transmission of four packets with two (unsynchronized) clocks A and B. The traffic has the s-characteristic with respect to the frames of clock A (i.e., A_1, A_2, A_3, and A_4), but not with respect to clock B. For clock A, time is measured on axis t, while for clock B, time

$$\begin{bmatrix} p_a & (1-p_b)p_a & (1-p_a)p_b \\ (1-p_a) & p_b & (1-p_a)(1-p_b) \\ 0 & (1-p_a)(1-p_b) & p_b \end{bmatrix}\cdot\begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix} = \begin{bmatrix}\pi_1\\ \pi_2\\ \pi_3\end{bmatrix} \tag{5}$$

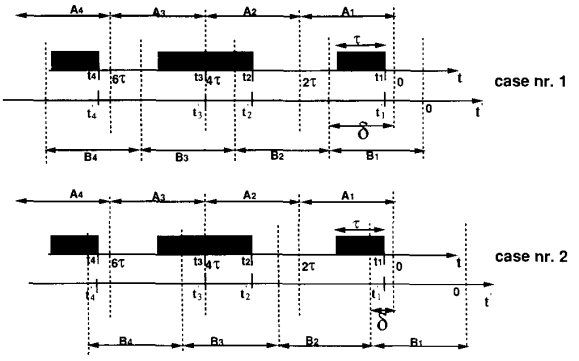


Fig. 28. Time diagram for proving the packet flipping algorithm.

is measured on axis t'. We will show how to convert the traffic so that it exhibits the s-characteristic with respect to clock B.

We label the offset of clock B from clock A by δ (i.e., $t_A - t_B = \delta$, where t_A and t_B are the clock instances of clock A and B, respectively). Assume case 1, where $\tau \leq \delta \leq 2\tau$. In this case, we use one of the delay lines of delay τ to offset the transmission, so that we can now assume that $0 \leq \delta \leq \tau$, as in the case 2. What remains to be shown is that in the case when $0 \leq \delta \leq \tau$, using a single delay line of delay τ restores the s -characteristics of the traffic with respect to clock B (the case nr. 2).

To prove the above, associate each frame of clock A with a frame of clock B, as shown in the figure (i.e., A_1 is associated with B_1, etc.). Now, the claim is that under the above conditions, a packet either fits in a frame of clock B, or can be delayed by τ and now fits in the frame.

If a packet fits in the clock B frame, nothing needs to be done. This will happen when $\delta \leq t_i \leq 2\tau$, where t_i is the arrival time of packet i. On the other hand, if a packet "falls" on the B clock (i.e., when $0 \leq t_i \leq \delta \leq \tau$), the packet needs to be delayed by τ ("flipped"). When this happens, the time of arrival of the packet i on the t' axis will be $t'_i = t_i - \delta + \tau$. But since $0 \leq t_i \leq \delta \leq \tau$, it follows that $0 \leq \tau - \delta \leq t'_i \leq \tau$. Thus, after flipping, the packet will fit into the clock B frame.

The Staggering Switch with frames of size 2τ will preserve the s-characteristic of the traffic with respect to the local clock. Thus, if the input traffic to the first switch in a series of switches is s-characteristic with respect to any clock, by using the flipping hardware in front of every switch the traffic presented to every one of the switches can be made s-characteristic with respect to every local clock.

REFERENCES

[1] P. S. Henry, "What can optics do for networks?" presented at the *IEEE Summer Topical Meet. Optical Multi Access Networks* (Monterey, CA), July 1990.

[2] Z. Haas and D. R. Cheriton, "Blazenet: A packet-switched wide-area network with photonic data path," *IEEE Trans. Commun.,* vol. 38, no. 6, pp. 818–829, June 1990.

[3] N. R. Dono, P. E. Green, Jr., K. Liu, R. Ramaswami, and F. F.-K. Tong, "A wavelength division multiple access network for computer communication," *IEEE J. Select. Areas Commun.,* vol. 8, pp. 983–994, Aug. 1990.

[4] I. Chlamtac, A. Ganz, and G. Karmi, "Lightnet: Lightpath based solutions for wide bandwidth WANs," *Proc. IEEE INFOCOM'90* (San Francisco, CA), pp. 1014–1021, June 1990.

[5] A. Huang and S. Knauer, "Starlite: A wideband digital switch," in *Proc. GLOBECOM'84,* pp. 121–125 (Atlanta, GA), Dec. 1984.

[6] R. Izmailov and Z. Haas, "Peformance evaluation of the optical staggering switch in underload input traffic conditions," in *Proc. Twenty-sixth Annual Conf. Information Sciences and Syst.* (Princeton, NJ), Mar. 1992.

[7] Z. Haas and R. D. Gitlin, "Field coding: A, High-speed "almost-all" optical interconnect," *1991 Conf. Information Sciences and Syst.* (Baltimore, MD), Mar. 1991.

[8] V. E. Beneš, "Permutation groups, complexes, and rearrangeable connecting networks," *The Bell Syst. Tech. J.,* p. 1619, July 1964.

[9] V. E. Beneš, "Optimal rearrangeable multistage connecting networks," *The Bell Syst. Tech. J.,* p. 1641, July 1964.

[10] K. Padmanabhan and A. N. Netravali, "Dilated networks for photonic switching," *IEEE Trans. Commun.,* vol. COM-35, p. 1357, 1987.

[11] Douglas Comer, *Internetworking with TCP/IP,* Prentice Hall, 1988.

[12] T. O. Murthy, C. T. Kemmerer, and D. T. Moser, "A 16 × 16 Ti : $LiNbO_3$ dilated Beneš photonic switch module," *Photon. Switching,* Postdeadline paper (Salt Lake City, UT), Mar. 1991.

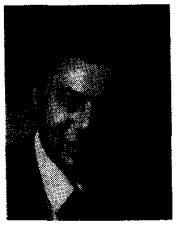
[13] M. J. O'Mahoney, "Semiconductor laser optical amplifier for use in future fiber systems," *J. Lightwave Technol.,* vol. 6, no. 4, Apr. 1988.

[14] Zygmunt Haas, "Optical distribution channel: An "almost-all" optical LAN based on the field-coding technique," *Proc. OE/Fibers'91 Symposium* (Boston, MA), Sept. 1991.

[15] Y. Silberberg, P. Perlmutter, and J. E. Baran, "Digital optical switch," *Appl. Phys. Lett.,* vol. 51, no. 16, p. 19, Oct. 1987.

[16] P. Granestrand *et al.,* "Integrated optics 4 × 4 switch matrix with digital optical switches," *Electron. Lett.,* vol. 26, no. 1, Jan. 1990.

[17] Y. Ota, R. G. Swartz, and V. D. Archer, "DC-1 Gb/s burst mode receiver for optical bus applications," in *Proc. SPIE High-Speed Fiber Networks and Channels* (Boston, MA), Sept. 4–6, 1991.



Zygmunt Haas (S'84–M'88–SM'90) received his B.Sc. degree in EE from Technion/Israel in 1979 and M.Sc. degree in EE from Tel-Aviv University/Israel in 1985, both with "Summa Cum Laude." From 1979 till 1985 he worked for the Government of Israel. In 1988, he earned his Ph.D. from Stanford University researching fast optical packet-switched networks, and subsequently joined AT&T Bell Laboratories in Holmdel NJ, where he is now a Member of Technical Staff in Network Systems Research Department. Dr. Haas is an author of numerous technical papers and holds several patents in the field of optical communication and high-speed networking. He is a Senior Member of IEEE and his interest include: high-speed communication and protocols, lightwave networks, optical switching, mobile communication, and personal communication service.