Terabit Burst Switching

Jonathan S. Turner
jst@cs.wustl.edu

wucs-98-17

July 17, 1998

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

Abstract

Demand for network bandwidth is growing at unprecedented rates, placing growing demands on switching and transmission technologies. Wavelength division multiplexing will soon make it possible to combine hundreds of gigabit channels on a single fiber. This paper presents an architecture for *Burst Switching Systems* designed to switch data among WDM links, treating each link as a shared resource rather than just a collection of independent channels. The proposed network architecture separates burst level data and control, allowing major simplifications in the data path in order to facilitate all-optical implementations. To handle short data bursts efficiently, the burst level control mechanisms in burst switching systems must keep track of future resource availability when assigning arriving data bursts to channels or storage locations. The resulting *Lookahead Resource Management* problems raise new issues and require the invention of completely new types of high speed control mechanisms. This paper introduces these problems and describes approaches to burst level resource management that attempt to strike an appropriate balance between high speed operation and efficiency of resource usage.

[0] This work is supported by the Defense Advanced Research Projects Agency and Rome Laboratory (contract F30602-97-1-0273).

Terabit Burst Switching

Jonathan S. Turner
jst@cs.wustl.edu

1. Introduction

By some estimates, bandwidth usage in the Internet is doubling every six to twelve months [4]. For the first time, data network capacities are surpassing voice network capacities and the growing demand for network bandwidth is expected to continue well into the next century. Current networks use only a small fraction of the available bandwidth of fiber optic transmission links. The emergence of WDM technology is now unlocking more of the available bandwidth, leading to lower costs which can be expected to further fuel the demand for bandwidth.

We now face the near-term prospect of single fibers capable of carrying hundreds of gigabits per second of data. Single optical fibers have the potential for carrying as much as 10 terabits per second. This leads to a serious mismatch with current switching technologies which are capable of switching at rates of “only” 1-10 Gb/s. While emerging ATM switches and IP routers can be used to switch data using the individual channels within a WDM link (the channels typically operate at 2.4 Gb/s or 10 Gb/s), this approach implies that tens or hundreds of switch interfaces must be used to terminate a single link with a large number of channels. Moreover, there can be a significant loss of statistical multiplexing efficiency when the parallel channels are used simply as a collection of independent links, rather than as a shared resource.

Proponents of optical switching have long advocated new approaches to switching using optical technology in place of electronics in switching systems [2, 6, 9]. Unfortunately, the limitations of optical component technology [7, 8, 12] have largely limited optical switching to facility management applications. While there have been attempts to demonstrate the use of optical switching in directly handling end-to-end user data channels, these experiments have been disappointing. Indeed they primarily serve to show how crude optical components remain and have done little to stimulate any serious move toward optical switching.

This paper proposes an approach to high performance switching that can more effectively exploit the capabilities of fiber optic transmission systems and could facilitate a transition to switching systems in which optical technology plays a more central role. *Burst Switching* is designed to make best use of optical and electronic technologies. It uses electronics to provide dynamic control of system resources, assigning individual user data bursts to channels of a WDM link. The control mechanisms are designed to efficiently handle data bursts as short as a kilobyte or as long as many megabytes. Burst switching is designed to facilitate switching of the user data channels entirely in the optical domain. While current optical components remain too crude for

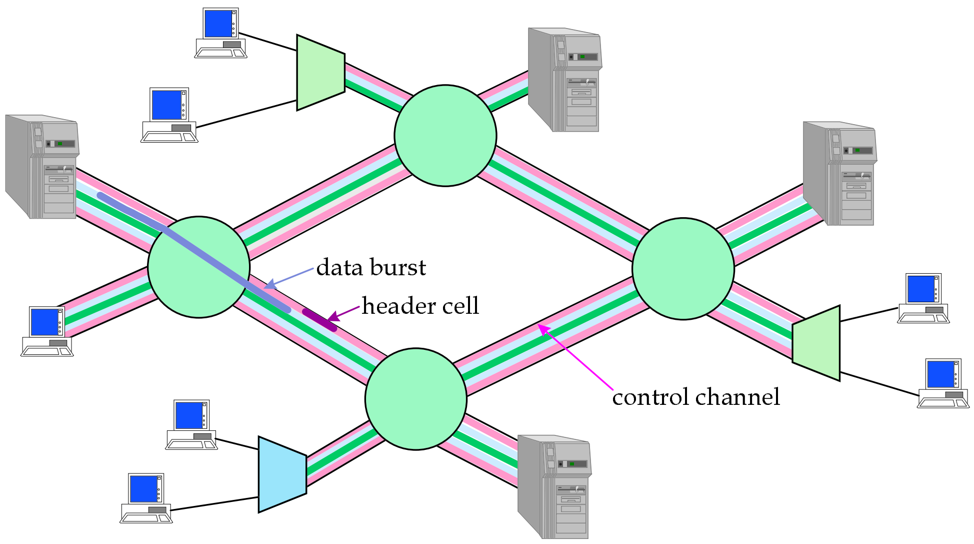


Figure 1: Terabit Burst Network Concept

this to be practical, anticipated improvements in integrated optics should ultimately make optical-domain switching feasible and economically sensible. In the meantime, the data paths of burst switching systems can be implemented in electronics with a substantial advantage in cost, relative to conventional ATM or IP switches.

2. Network Architecture

Figure 1 shows the basic concept for a terabit burst network. The transmission links in the system carry multiple *channels*, any one of which can be dynamically assigned to a user data burst. The channels can be implemented in any of several ways. Wavelength Division Multiplexing (WDM) shows the greatest potential at this time, but Optical Time Division Multiplexing (OTDM) [10] is also possible. Whichever multiplexing technique is used, one channel on each link is designated a *control channel*, and is used to control dynamic assignment of the remaining channels to user data bursts. In addition to WDM and OTDM, there is a third alternative, which is not to multiplex channels on a single fiber, but to provide a separate fiber for each channel in a multi-fiber cable. Integrated optical device arrays using optical ribbon cables that are now becoming commercially available [11], make this option particularly attractive for LAN applications where cost is a predominant concern. In fact, even in a wide area network using WDM or OTDM on inter-switch trunks, access links may be most cost-effectively implemented using this approach.

The format of data sent on the data channels is not constrained by the burst switching system. Data bursts may be IP packets, a stream of ATM cells, frame relay packets or raw bit streams from remote data sensors. However, since the burst switching system must be able to interpret the information on the control channel, a standard format is required there. In this paper, ATM cells are assumed for the control channel. While other choices are possible, the use of ATM cells makes it possible to use existing ATM switch components within the control subsystem of the burst switch, reducing the number of unique components required.

When an end system has a burst of data to send, an idle channel on the access link is selected, and the data burst is sent on that idle channel. Shortly before the burst transmission begins, a *Burst Header Cell* (BHC) is sent on the control channel, specifying the channel on which the burst is being transmitted and the destination of the burst. A burst switch, on receiving the BHC, selects an outgoing link leading toward the desired destination with an idle channel available, and then establishes a path between the specified channel on the access link and the channel selected to carry the burst. It also forwards the BHC on the control channel of the selected link, after modifying the cell to specify the channel on which the burst is being forwarded. This process is repeated at every switch along the path to the destination. The BHC also includes a length field specifying the amount of data in the burst. This is used to release the path at the end of the burst. If, when a burst arrives, there is no channel available to accommodate the burst, the burst can be stored in a buffer.

There are two alternative ways to route bursts. In the *Datagram mode,* BHCs include the network address of the destination terminal, and each switch selects an outgoing link dynamically from among the set of links to that destination address. This requires that the switch consult a routing database at the start of each burst, to enable it to make the appropriate routing decisions. In the *Virtual Circuit mode*, burst transmissions must be preceded by an end-to-end route selection process, similar to an ATM virtual circuit establishment. During route selection, the forwarding information associated with a given end-to-end session is stored in the switches along the path. BHCs include a Virtual Circuit Identifier (VCI), which the switches use in much the same way that ATM switches use an ATM VCI. Note that while the route selection fixes the sequence of links used by bursts in a given end-to-end session, it does not dictate the individual channels used by bursts. Indeed, channels are assigned only while bursts are being sent.

To handle short data bursts efficiently, burst switching systems must maintain tight control over the timing relationships between BHCs and their corresponding data bursts. Uncertainty in the precise timing of the beginning and ending of bursts leads to inefficiencies, since burst switching systems must allow for these uncertainties when setting up and tearing down paths. For efficient operation, timing uncertainties should be no more than about 10% of the average burst duration. For efficient handling of bursts with average lengths of 1 KB or less on 2.4 Gb/s channels, we need to keep the end-to-end timing uncertainty in a burst network to no more than 333 ns.

3. Key Performance Issues

There are two key performance concerns for a burst switching network. The first, is the statistical multiplexing performance. In particular, we need to know understand to engineer a burst-switched network so that it can be operated at reasonably high levels of utilization, with acceptably small probability that a burst is discarded due to lack of an available channel or storage location.

Consider a multiplexor that assigns arriving bursts to channels in a link with h available data channels and storage locations for b bursts. An arriving burst is diverted to a storage location if

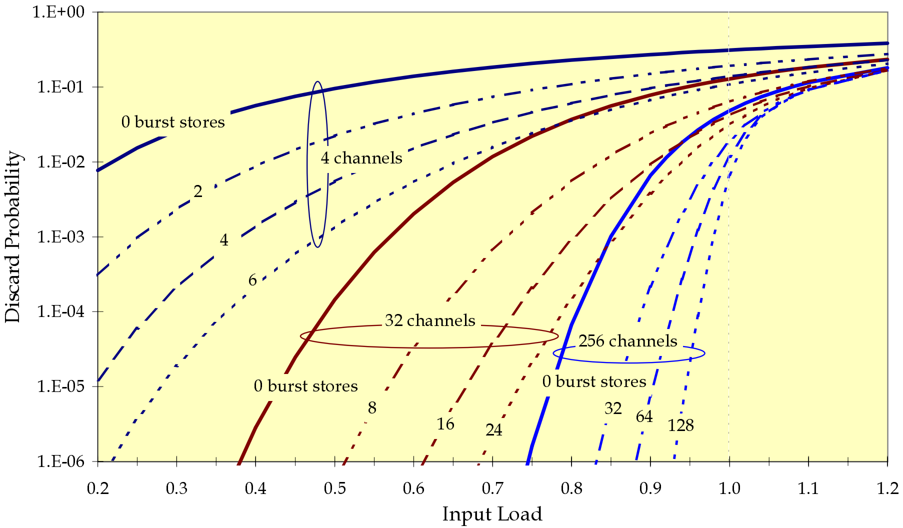


Figure 2: Burst Discard Probability

when it arrives, all h data channels are in use. A burst will be discarded if it arrives when all h channels are busy and all b burst storage locations are being used by bursts that have not yet been assigned a channel. However, once a stored burst has been assigned to an output channel, its storage location becomes available for use by an arriving burst, since the stored burst will vacate space at the same rate that the arriving burst occupies it.

The multiplexor can be modeled by a birth-death process with $b + h + 1$ states, where the state index i corresponds to the number channels in use plus the number of stored bursts that are waiting to be assigned a channel. The transition rate from state i to state $i + 1$ is λ for all values of $i < bh$, where λ is the burst arrival rate. The transition rate from state i to state $i - 1$ is $\min\,\{i, h\}\mu$ where μ is the channel data rate divided by the burst length. If we let π_i be the steady state probability that the system is in state i, then it is easy to show that

$$\pi_i = \begin{cases} \left(\frac{\lambda}{\mu}\right)^i (\pi_0/i!) & 0 \le i \le h \\ \left(\frac{\lambda}{\mu}\right)^i (\pi_0/h!h^{i-h}) & h < i \le h + b \end{cases}$$

Using $\pi_0 + \cdots + \pi_{h+b} = 1$ and solving for π_0 yields

$$1/\pi_0 = 1 + \sum_{i=1}^{h} \left(\frac{\lambda}{\mu}\right)^i \frac{1}{i!} + \sum_{i=h+1}^{h+b} \left(\frac{\lambda}{\mu}\right)^i \frac{1}{h^{i-h}h!}$$

The probability that an arriving burst is discarded is simply π_{h+b}. Figure 2 shows the burst discard probability for links with 4, 32 and 256 channels, with several different choices for the number of burst stores. Note that when the number of channels is large, reasonably good statistical multiplexing performance can be obtained with no burst storage at all. With 32 channels, 8 burst stores is sufficient to handle bursty data traffic at a 50% load level (an average of 16 of the 32 channels are busy), with discard probabilities below 10^{-6}. With 256 channels and 128 burst stores, loading levels of over 90% are possible with low loss rates.

The second key performance issue for burst switches is the overhead created by timing uncertainty. When a user terminal has a burst to send, it first sends a BHC on the control channel of its access link and shortly afterwards, it sends the burst. The BHC includes an *offset* field that specifies the time between the transmission of the first bit of the BHC and the first bit of the burst. When BHCs are processed in burst switches, they are subject to variable processing delays, due to contention within the control subsystem of the burst switch. To compensate for the delays experienced by control cells, the data bursts must also be delayed. A fixed delay can be inserted into the data path by simply routing the incoming data channels through an extra length of fiber. To maintain a precise timing relationship between control and data, BHCs are *time-stamped* on reception. When a switch is preparing to forward a BHC on the output link, it compares the time stamp to the current time, in order to determine how much delay the BHC has been subjected to in its passage through the burst switch. Knowing the (fixed) delay of the data path, it can then recalculate the offset between the BHC and the data. It can also impose some additional delay on the BHC if necessary, to keep the offset from becoming excessively large. If the burst was buffered in the burst switch, the queueing delay must also be taken into account, when recomputing the offset.

As a burst progresses through a burst network, there is some inevitable loss of precision in the offset information. Timing uncertainty arises from two principal sources. First, signals using different wavelengths of a WDM link travel at different speeds down the fiber. While the absolute magnitudes of these differences are not large (under 1 μs for links of 1000 km), they are significant enough to be a concern. Fortunately, they can be compensated at the receiving end of a link, if the approximate length of the fiber is known. Given the length information, the difference in the delay between the control channel and each of the data channels can be calculated and the receiving control circuitry can adjust the offset associated with an arriving burst, accordingly. Note, that this does not require inserting optical delays in the data channel. The control subsystem, simply adjusts the offset value in the BHC to reflect the differences in delay across channels. While this compensation is not perfect, it can be made accurate enough to keep timing uncertainties due to this cause under 100 ns on an end-to-end basis in a wide area network.

Another source of timing uncertainty is clock synchronization at burst switches along the path in a network. When data is received on the control channel at a burst switch, the arriving bit stream must be synchronized to the local timing reference of the burst switch. There is some inherent timing uncertainty in this synchronization process, but the magnitude of the uncertainty can be reduced to very small values using synchronizers with closely-spaced taps. Even with fairly simple synchronizer designs, it can be limited to well under 10 ns per switch. In an end-to-end connection with ten switches, this results in an end-to-end timing uncertainty of 100 ns.

Within switches, it's possible to avoid significant loss of timing precision, since all control operations within a switch can be synchronized to a common clock, and since the data path delays in a switch are fixed and can be determined with high precision. Thus, the end-to-end timing uncertainty is essentially the sum of the uncertainty due to uncompensated channel-to-channel delay variation on links and the uncertainty due to synchronization. If the end-to-end uncertainty is limited to 200 ns then bursts with durations of 2 μs can be handled efficiently. This corresponds to 600 bytes at 2.4 Gb/s.

It's possible to avoid the necessity for tight end-to-end timing control if the switches interpret the information sent on the data channels so that they can explicitly identify the first and last bit

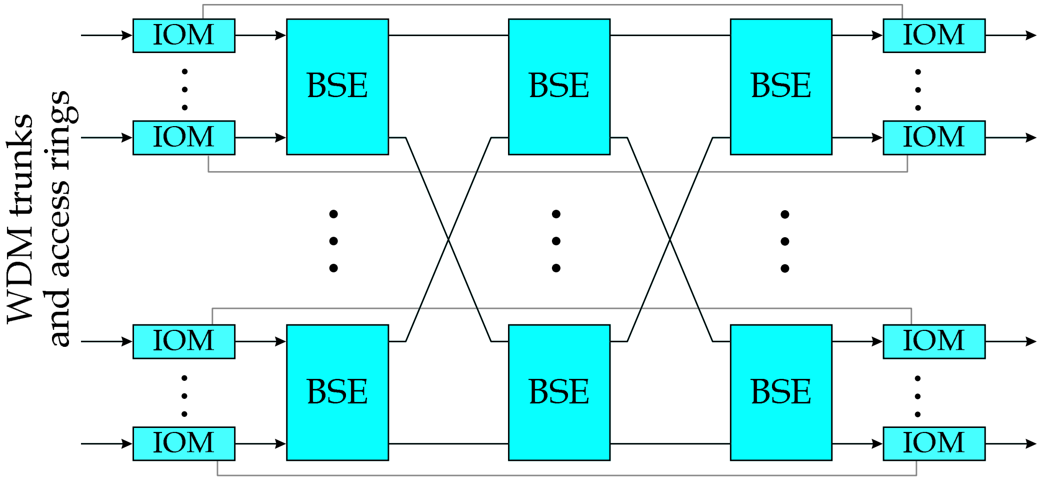


Figure 3: Scalable Burst Switch Architecture

of a burst. While this is reasonable to do in electronic implementations, it is more difficult in systems with optical data paths. In addition, it constrains the format of data that can be sent on the data paths, at least to some degree.

4. Scalable Burst Switch Architecture

Figure 3 illustrates a scalable burst switch architecture consisting of a set of *Input/Output Modules* (IOM) that interface to external links and a multistage interconnection network of *Burst Switch Elements* (BSE). The interconnection network uses a Benes topology, which provides multiple parallel paths between any input and output port. A three stage configuration comprising d port switch elements can support up to d^2 external links (each carrying many WDM channels). The topology can be extended to 5,7 or more stages. In general, a $2k + 1$ stage configuration can support up to d^k ports, so for example, a 7 stage network constructed from 8 port BSEs would support 4096 ports. If each port carried 128 WDM channels at 2.4 Gb/s each, the aggregate system capacity would exceed 1,250 Tb/s.

The operation of the burst switch is illustrated in Figure 4. An arriving burst is preceded by a *Burst Header Cell* (BHC) carried on one of the input channels dedicated to control information. The BHC carries address information that the switch uses to determine how the burst should be routed through the network, and specifies the channel carrying the arriving data burst. The IOM uses the address information to do a routing table lookup. The result of this lookup includes the number of the output port that the burst is to be forwarded to. This information is inserted into the BHC, which is then forwarded to the first stage BSE. The data channels pass directly through the IOMs but are delayed at the input by a fixed interval to allow time for the control operations to be performed.

When a BHC is passed to a BSE, the control section of the BSE uses the output port number in the BHC to determine which of its output links to use when forwarding the burst. If the required

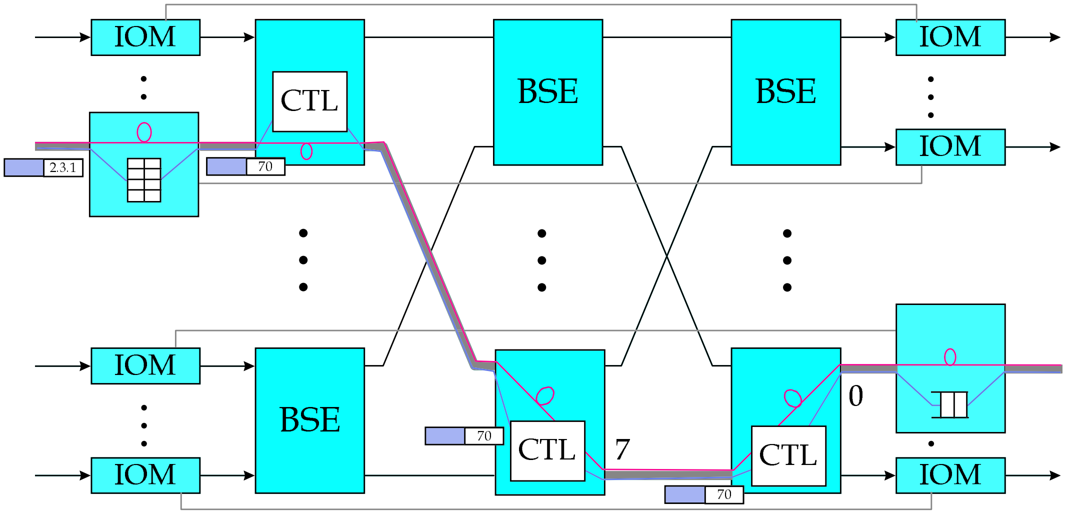


Figure 4: Operation of Burst Switch

output link has an idle channel available, the burst is switched directly through to that output link. If no channel is available, the burst can be stored within a shared *Burst Storage Unit* (BSU) within the BSE.

In the first stage of a three stage network, bursts can be routed to any one of a BSE's output ports. The port selection is done dynamically on a burst-by-burst basis to balance the traffic load throughout the interconnection network. The use of *dynamic routing* yields optimal scaling characteristics, making it possible to build large systems in which the cost per port does not increase rapidly with the number of ports in the system.

At the output port of a burst switch, the BHC is forwarded on the outgoing link and the offset field is adjusted to equal the time delay between the transmission of the first bit of the BHC and the first bit of the burst.

4.1. Multicast Burst Switching

There are several approaches that can be used to implement multicast switching in an architecture of this type. However, the most efficient approach implements multicast in multiple passes through the network, with binary copying in each pass. To implement this method, a subset of the system's inputs and outputs are connected together in a loopback configuration. Ports connected in such a loopback configuration are called *recycling ports*.

The routing table in an IOM can specify a pair of output ports that a burst is to be forwarded to, and the multistage interconnection network uses this information to forward the burst to both specified outputs. If either or both of the specified outputs is a recycling port, the corresponding copy of the burst will pass through the system again, allowing it to be forwarded to two more outputs. In this way, a burst can be forwarded to four outputs in two passes through the network, to eight outputs in three passes, etc.

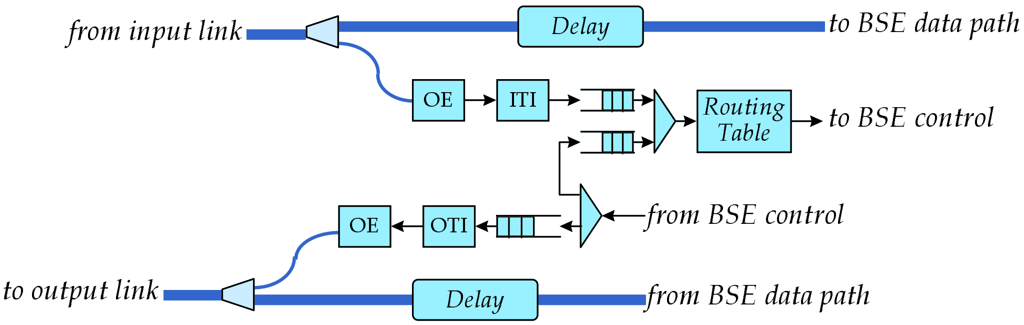


Figure 5: I/O Module

The total bandwidth consumed by multicast traffic on the recycling ports is strictly less than the total bandwidth consumed on output links that are not recycling ports. If δ is the fraction of the outgoing traffic that is multicast, then a system with m output links will need δm recycling ports. Since δ is typically fairly small (say .1), the incremental cost of multicast is also small. In any case, the incremental cost of multicast switching is just a (small) constant factor over the cost of a comparable system that only supports unicast. All known single pass multicast methods, on the other hand, lead to a multiplicative cost increase that grows at least as fast as the logarithm of the number of ports. The multipass approach also makes it easy to add endpoints to or remove endpoints from a multicast, and can be used to provide scalable many-to-many multicast, as well as one-to-many. This method has been adapted from the ATM switch architecture described in [3]; further details can be found there and in references [13, 14].

4.2. IO Module

Figure 5 shows the components of the I/O module. On the input side, data channels are subjected to a fixed delay, while the control channel is converted to electronic form by an *Optoelectronic Receiver* (OE) and decoded by the *Input Transmission Interface* (ITI). BHCs extracted from the control channel are time-stamped and placed in a queue, which is synchronized to the local clock. The routing table uses address information in BHCs to select routing table entries, which specify the output ports that bursts are to be sent to. The offset fields in forwarded BHCs are updated by the input control circuitry to reflect variable processing delays within the IOM, using the timestamps added to the BHCs on reception. As the BHCs are forwarded to the first BSE, the timestamp fields are also updated to the current time.

On the output side, bursts are again subjected to a fixed delay, while the control information is processed. There is little real processing needed on the output side, but the output circuitry must still account for variable queueing delays and adjust the offset fields of BHCs accordingly.

The input and output sections of each IOM are linked by a loopback path, which allows special configuration cells to be sent to any input port. These configuration cells are used to update the routing tables in IOMs, as well as providing access to other system configuration registers.

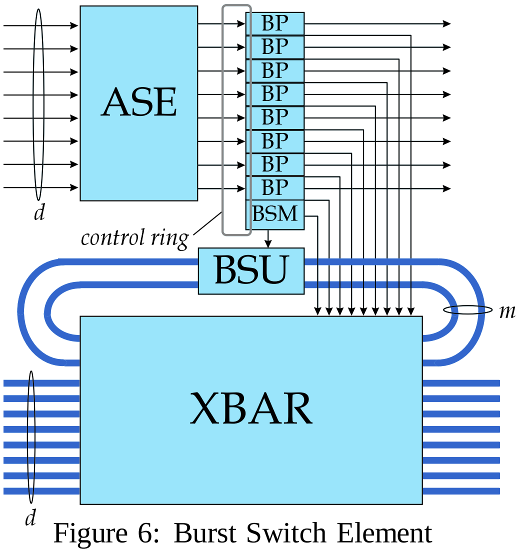


Figure 6: Burst Switch Element

4.3. Burst Switch Element

Figure 6 shows how the BSE can be implemented. In this design, the control section consists of a d port *ATM Switch Element* (ASE), a set of d *Burst Processors* (BP), and a *Burst Storage Manager* (BSM). The data path consists of a crossbar switch, together with the *Burst Storage Unit* (BSU). The BSU is connected to the crossbar with m input links and m output links. The crossbar is capable of switching a signal on any channel within any of its input links to any channel within any of its output links; so in a system with d input and output links and h data channels per link, we require the equivalent of a $(d + m)h \times (d + m)h$ crossbar. Often, the implementation can be simplified by decomposing it into $d + m$ separate $(d + m)h \times h$ crossbars. Each BP is responsible for handling bursts addressed to a particular output link. When a BP is unable to switch an arriving burst to a channel within its output link, it requests use of one of the BSU's storage locations from the BSM, which switches the arriving burst to an available storage location (if there is one). Communication between the BSEs and the BSM occurs through a local *control ring* provided for this purpose.

To enable effective control of dynamic routing, BSEs provide flow control information to their upstream neighbors in the interconnection network. This can take any of several forms. Since some BSE outputs may have channels available, even when other outputs are congested, it is most useful to supply flow control information that reports the status of different outputs. Each output that has sufficiently many spare channels can enable transmission of new bursts by the upstream neighbors, by sending the appropriate flow control indication to the upstream neighbors. This communication takes place in two steps. First, each BP in a BSE shares its flow control information with the other BPs in the same BSE (using the local control ring). Then each BP in the BSE sends its own flow control information and that of the other BPs to one of the BSE's upstream neighbors. This communication takes place using a set of upstream control signals not shown in the figure.

The *Burst Processor* (BP) is perhaps the key component of the BSE. A block diagram for the

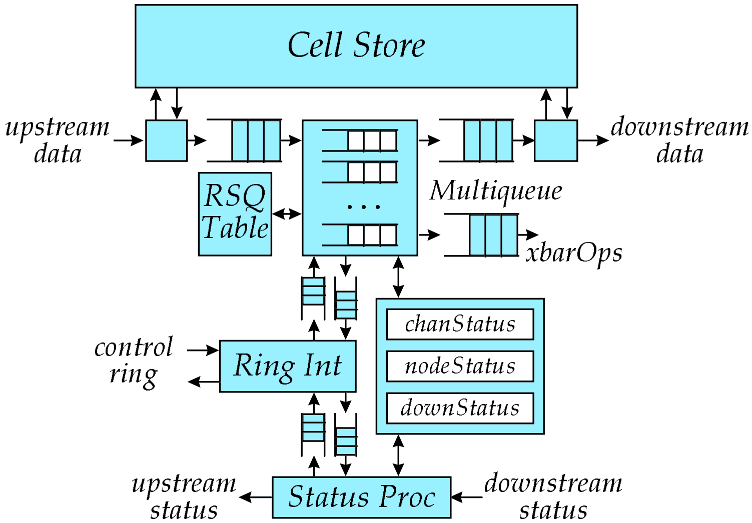


Figure 7: Burst Processor

BP is given in Figure 7. The BP includes a *Cell Store* which is used to store cells as they are processed by the BP. On reception, selected fields of BHCs are extracted and sent through the control portion of the BP, along with a pointer to the BHC's storage location in the cell store. (For simplicity of description, the following text describes the control operation in terms of BHCs moving through various parts of the BP's control circuitry. It should be understood that this refers to just selected information from BHCs, not the entire cell contents.)

On arrival, BHCs are forwarded to a general queueing block called the *Multiqueue*. In the case of a BP that is not in the last stage, the multiqueue implements d queues, one for each output of the downstream BSE. If the next stage BSE is in one of stages k through $2k-1$ of a $2k-1$ stage network, then the address information in the BHC is used to select which queue it should go into. If the next stage is in one of stages 2 through $k-1$, the queue is selected using a dynamic load distribution algorithm that seeks to balance the load, while taking into account the status of the downstream BSE's outputs. If the burst for an arriving BHC cannot be assigned to an outgoing channel immediately, a request is sent to the BSM through the control ring. If the BSM cannot store the burst, the BHC is marked *deleted* (BHCs for deleted bursts are forwarded to the output IOMs where they are counted).

The scheduler for the multiqueue forwards BHCs to the output queue when there is an available channel and when the status information from the downstream neighbor shows that it can accept the burst. At the same time it schedules the appropriate crossbar operation to make the necessary connection to the selected output channel.

The *Status Processor* at the bottom of the figure receives status information from downstream neighbors and other BPs in the same BSE (through the control ring) and updates the stored status information accordingly. It also forwards status information to one of the neighboring upstream BSEs and the other BPs in the same BSE.

The use of dynamic routing in the interconnection network means that bursts corresponding to a single end-to-end session can be routed along different paths through the interconnection network, creating the possibility that they get out of order. Burst switches could simply ignore this issue and leave the problem or resequencing bursts to terminal equipment. Alternatively, burst switches can include mechanisms to ensure that bursts belonging to common user data streams are propagated in the same order as they arrive. In ATM switches using dynamic routing, time-based resequencing is typically used to ensure cells are forwarded in order (see [3] for example). Time-based resequencing is problematical in burst switches because in burst switches (unlike ATM switches), time-based resequencing greatly increases the data storage requirements and data storage is a fairly expensive resource (particularly, in possible future implementations with all-optical data paths).

To minimize storage requirements, resequencing should be done on the basis of individual user data streams. For bursts using virtual circuit routing, this can be done by assigning sequence numbers to BHCs on arrival at input IOMs and using these sequence numbers in the last stage of the interconnection network to reorder bursts as necessary to compensate for varying delays in the interconnection network. In the last stage, the multiqueue block implements a collection of resequencing buffers that are used for reordering bursts, as necessary. The *Resequencing Table* maps VCIs of received BHCs to the proper resequencing buffer.

5. Lookahead Resource Management

For purposes of exposition, the previous sections have purposely glossed over one important issue associated with the burst switch control. Varying queueing delays in the control portion of a burst switch can be as much as 1-10 μs. For this reason, it's important to maintain an offset of at least 10 μs between BHCs and their corresponding bursts (since otherwise, a burst might arrive at a given point in the system before a switch path has been configured for it). If we are to efficiently handle bursts that may be as short as 1 μs in duration, this means that the mechanisms for managing resources in a burst switch must have the ability to project resource availability into the future and make decisions based on future, rather than current resource availability. This requires the use of resource allocation mechanisms that are quite different from those used in conventional systems.

5.1. Channel Scheduling

Consider first the problem of managing the channels of a link leaving a Burst Processor. Suppose we receive a BHC and using the offset and timestamp information in the cell, we determine that the burst will arrive in 10 μs. If we assign the burst to a channel that is available now, then we fragment the resources of the channel, leaving a 10 μs idle period which may be difficult to make use of. To avoid this kind of resource fragmentation, we would prefer to assign the burst to a channel that will become available just shortly before the burst arrives. Ideally, we want to assign the arriving burst to the channel that becomes available as late as possible before the deadline, so as to minimize the idle period created by our scheduling decision and maintain maximum flexibility for handling later arriving bursts.

To support high burst processing rates (millions of BHCs per second), we need to keep the scheduling algorithm as simple as possible. One approach is to maintain a single *scheduling horizon* for each channel on the link. The scheduling horizon for a channel is defined as the latest time at which the channel is currently scheduled to be in use. Given this information, the procedure for assigning a burst to a channel becomes obvious. Simply select from among the channels whose scheduling horizons precede the burst's arrival time and select the channel from this set with the latest scheduling horizon. Once a channel has been selected, we recompute the scheduling horizon to be equal to the time when the burst is due to be completed (based on our knowledge of the arrival time and burst duration).

The simple horizon-based scheduling algorithm can waste resources since it only keeps track of a channel's latest idle period. While it attempts to minimize the size of the gaps between successive bursts, its failure to maintain information about those gaps means that it cannot schedule short bursts within the gaps. More sophisticated scheduling algorithms which maintain partial or complete information about a channel's planned usage are certainly possible. Additional study is needed to evaluate the costs and benefits of such alternatives.

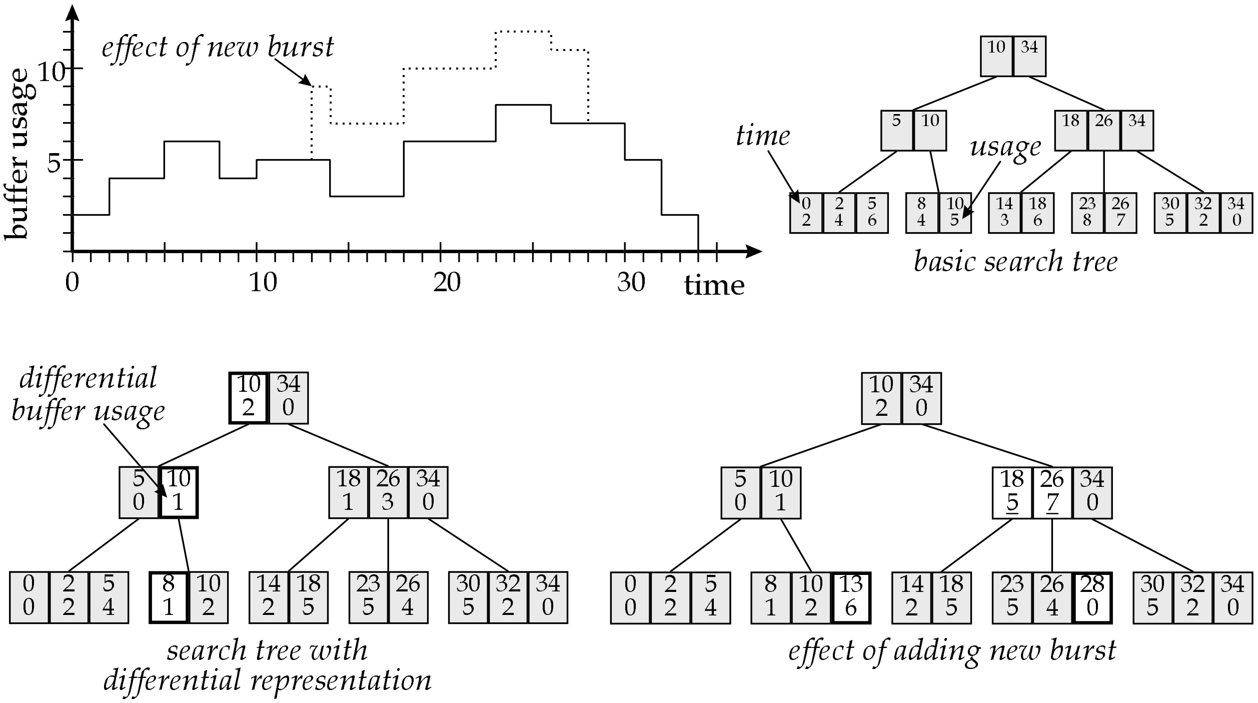
5.2. Flow Control

In a system that uses horizon-based scheduling for its links, it is natural to use a similar *horizon-based flow control* to mange the flow of bursts between stages in a multistage interconnection network. In horizon-based flow control, a BP tells its upstream neighbors at what time in the future it will be prepared to accept new bursts from them. A BP with d upstream neighbors can use its per channel scheduling horizon information to determine the earliest time at which it will have d idle channels and tell its upstream neighbors that it can send bursts at this time, but not before. Alternatively, a different threshold can be used (larger or smaller than d). For small choices of the threshold we may occasionally get more bursts arriving than the output link can handle at once, forcing some of the arriving bursts to be buffered. The proper choice of threshold will be determined by the burst length characteristics and the delay in the delivery of the flow control information to the upstream neighbor.

5.3. Burst Storage Scheduling

In addition to the link resources, a burst switch must also manage the allocation of burst storage resources. This is somewhat more complex because there are two dimensions to consider, time and storage space. The upper left portion of Figure 8 shows a plot of buffer usage vs. time for a burst storage unit that is receiving and forwarding bursts. As new bursts come in, they consume additional storage space releasing the storage later as they are forwarded to the output. The plot shows how a new burst arrival modifies the buffer usage curve. One way to schedule a burst storage unit is to maintain a data structure that effectively describes the buffer usage curve for the burst store. For high performance, we need a data structure that supports fast incremental updating in response to new burst arrivals.

This can be done by extending a search tree data structure such as a 2-3 tree (2-3 trees are a special case of B-trees [5]). The upper right hand portion of Figure 8 shows a 2-3 tree that



basic search tree

search tree with differential representation

effect of adding new burst

Figure 8: Burst Store Occupancy vs. Time

represents the buffer usage curve at the top left. The leaves in this search tree contain pairs of numbers that define the left endpoints of the horizontal segments of the buffer usage curve. Each interior node of the search tree contains a time value equal to the largest time value appearing in any of its descendant leaves. These time values allow one to determine the range of times represented by a given subtree without explicitly examining all the nodes in the subtree.

To allow fast incremental updating of the 2-3 tree when adding a new burst, the basic structure needs to be extended by using a *differential representation* for the buffer usage values. This is illustrated at the bottom left of Figure 8 which shows a search tree that represents the same buffer usage curve. In this search tree, buffer usage must be computed by summing the values along the path from the root of the tree to a value-pair at a leaf. Consider for example, the three value-pairs that are highlighted in the figure. The sum of the buffer usage values along this path is equal to 4, which equals the buffer usage of the corresponding value-pair at the "bottom" of the original search tree.

The differential representation is useful because it allows rapid incremental updating in response to a new burst arrival. This is illustrated at the bottom right of Figure 8. Here, two new value-pairs (highlighted) have been added to reflect the addition of the burst indicated in the top left portion of the figure. Note that two buffer usage values in the righthand child of the tree root have also been modified (these values are underlined) to reflect the addition of the new burst. Using this sort of differential representation, the data structure can be modified to reflect an additional burst in $O(\log n)$ time, as opposed to $O(n)$ time with the original representation.

Unfortunately, to test if a newly arriving burst can be stored without exceeding the available storage space, we may still need examine a large portion of the search tree, resulting in unacceptably large processing times. However, this extra processing time can be avoided if we are willing to accept some loss in the efficiency with which the buffer is used. Let t_1 be the minimum time between the arrival of a BHC and its corresponding burst and let t_2 be the maximum time between a BHC and its corresponding burst. For scheduling purposes, we can ignore the portion of the buffer usage curve that precedes the current time plus t_1. More significantly, the buffer usage curve is monotonically non-increasing for all values of t greater than the current time plus t_2. If $t_1 = t_2$ then we can completely ignore the non-monotonic part of the buffer usage curve and focus only on the monotonic part (so in the example curve in Figure 8, we can ignore the portion before time 28 if the minimum and maximum offsets are equal).

These observations suggest a scheduling algorithm that maintains a monotonic approximation to the buffer usage curve and uses this approximation to make scheduling decisions. The use of this approximation will occasionally cause bursts to be discarded that could safely be stored. However, so long as the gap between the minimum and maximum offsets is small compared to the typical "dwell time" of stored bursts, these occurrences should be rare.

With a monotonic buffer usage curve it is easy to determine if an arriving burst can be accommodated. If there will be enough storage space to handle the "head" of the burst then we are guaranteed that there will be enough to handle the remainder (since the approximate buffer usage curve never increases). Thus, we only need to perform a single comparison to check if an arriving burst can be handled. Once we've determined that a burst can be safely handled, the incremental updating can occur in $O(\log n)$ time, as described above.

To achieve gigabit processing rates for large numbers of stored bursts, 2-3 tree may be too slow. Substantial additional speedup can be obtained using a general B-tree. In a B tree [5], each node can have between B and $2B-1$ children, allowing a substantial reduction in the depth of the search tree. In a Burst Storage Manager using a B-tree to manage the storage space, hardware mechanisms can be used to enable parallel processing of individual B-tree nodes.

6. Implications for Implementation Technologies

Burst switching carries some implications for implementation technologies that may not be obvious at first glance. Given the long-term desirability of all optical implementations, it is important to examine these implications. Consider a burst switch in which the data path is implemented entirely using passive optical components with no re-timing of the data. A terminal receiving a burst through such a switch, must synchronize its receiving circuitry to the arriving data stream, at both the bit level and the burst level. That is, it must recover clock from the arriving bit stream and use whatever embedded formatting is provided by the transmission system to allow it to separate actual user data bits from non-data bits.

At the moment the burst switch makes the connection between the transmitter and the receiver, the frequency and phase of the signal seen at the receiver changes abruptly. The receiving circuitry requires some time to lock into the timing information of the received signal following this transition. After the receiver locks onto the signal's bit level timing, it must identify embedded formatting information in the arriving bit stream to separate data from non-data.

Transmission systems use a variety of different methods for encoding data for transmission. The choice of methods has a very large impact on the time required for a receiver in a burst-switched network to achieve synchronization with the incoming data stream. To handle short duration bursts, the synchronization time needs to be very small, preferably less than 100 ns. This effectively rules out the use of complex transmission formats such as SONET, which inherently require a minimum of 125 μs for synchronization (due to the 125 μs frame period built into the SONET format). The 8B/10B line code used in Fibre Channel and similar serial data links lends itself to much more rapid synchronization. This is a direct consequence of redundancy built into these data formats.

The 8B/10B line code also facilitates more rapid bit level synchronization, since it provides strict limits on the time between bit transitions, which is not true for formats like SONET that rely on pseudo-random scrambling of the data. However, there is an inherent trade-off between rapid timing acquisition and long term stability in receiver circuits that has important consequences for the achievable bit error rate on optical links. Today, receiver circuits typically have fairly long acquisition times (over a millisecond in many cases). While faster acquisition times are certainly possible, improvements in acquisition time come at the cost of higher bit error rates in the presence of noise-induced timing jitter. It seems very unlikely that acquisition times could reduced to under 100 ns without unacceptably high bit error rates.

If receivers cannot be made to acquire timing information rapidly we must either sacrifice the ability to handle short bursts or seek an implementation strategy that does not require such fast acquisition times. If we pursue the latter alternative, we must discard implementations based on passive optical switching. The burst switch data paths must include components that are capable of retiming the data stream as it passes through the burst switch. If retiming elements are placed immediately before and after switching points, the timing skew between them can be kept small enough to eliminate the need for fast timing acquisition.

The introduction of re-timing elements also leads to a much more robust network design that can operate at high data rates in complex networks spanning large distances and many switches. While passive end-to-end optical data paths have a lot of appeal, they do not appear to be a realistic option in large scale networks.

The use of retiming elements throughout the burst network means that there must be transmission components at the inputs to burst switches that recover timing information from the received data streams and use the recovered timing information to regenerate the data and forward it using a local timing reference. Since the local timing reference may differ in frequency from the frequency of the incoming data, the transmission components must be able to make adjustments by inserting or removing padding information in the data stream. With data formats like the Fiber Channel 8B/10B line code, these adjustments are straightforward for electronic implementations. (In fact, for electronic implementations, complex transmission formats like SONET remain an option, although an expensive one.) For all optical implementations, there are no fundamental obstacles to providing these capabilities, although practical implementations must await advances in integrated optics. For such implementations, simple line codes are essential (it's unlikely that SONET transmission components will ever be implemented optically). In fact, it probably makes sense to use even simpler line codes (such as Manchester encoding), trading off per-channel transmission bandwidth for simplicity of implementation and the ability to handle much larger numbers of channels per link.

7. Closing Remarks

Burst switching is not an entirely new concept. The ideas presented here are similar to fast circuit switching concepts developed in the early eighties [1]. Fast circuit switching was developed originally to support statistical multiplexing of voice circuits, but was also suitable for data communication at moderate data rates. Fast circuit switching was largely superseded by fast packet and cell switching in the middle and late eighties. These technologies can be implemented using hardware of comparable complexity and provide greater flexibility and superior statistical multiplexing performance.

Given fast circuit switching's limited success in the eighties, one must question why burst switching should be a viable approach to high performance data communication now. To answer this, one must consider the differences in underlying technologies now and in the eighties. At that time, fast circuit switching was implemented using electronic time-division multiplexing to provide distinct channels. It was implemented using essentially the same underlying technology base as its principal competition, ATM. The circuit complexities of the two technologies were roughly comparable. While fast circuit is conceptually somewhat simpler, complete implementations were not significantly less complex than ATM systems. The data paths in fast circuit and ATM systems were very similar in performance and placed essentially the same demands on the underlying implementation technology. As a result, fast circuit switching had no area in which it could maintain an advantage relative to ATM and so, the superior flexibility of ATM gave it a competitive edge.

The situation now is qualitatively different. Optical fibers have bandwidths that far exceed what can be achieved with electronics. If we are to exploit a larger fraction of the optical bandwidth, there is no choice but to multiplex multiple data streams onto single fibers. Furthermore, there are distinct advantages in both cost and flexibility, that can be obtained if the data can be maintained in optical form on an end-to-end basis. In this context, it makes sense to carry control information in a separate but parallel control channel, allowing the data path to be kept very simple. Furthermore, it is best to avoid queueing as much as possible, since both optical and electronic memories become expensive at gigabit data rates. For this reason, it is most efficient to achieve good statistical multiplexing performance using fiber's ability to carry many independent data channels in parallel, rather than relying on queueing.

A possible alternative to burst switching is some form of optical packet switching, in which the capacity of a fiber is dedicated to the transmission of a single data packet for a period of time, rather than dividing the fiber's capacity among multiple parallel streams. Reference [9] describes the principal architectural alternatives that have been studied for optical packet switching. Fundamentally, all of these systems require substantial packet level buffering, making them difficult to implement in a cost-effective way. Burst switching avoids most of this buffering through its use of parallel data channels. In addition, the explicit separation of control and data facilitates the implementation of the data path using technologies capable of only very limited logic operations (including both optics and very high speed electronics).

Looking ahead into the next decade, we can anticipate continuing improvements in both electronics and optical technology. In the case of electronics, there will be continuing dramatic improvements in circuit density and substantial but smaller improvements in circuit speed. Improvements in circuit speeds will be more limited because of the qualitative differences that arise when bit times become comparable to or smaller than the propagation times between circuit components. In this domain, the mechanisms for sending and receiving data reliably are far more complex than in lower speed systems.

In the optical domain, we can also expect to see substantial improvements. Optoelectronic transmitter and receiver arrays will become widely deployed in commercial applications and the technology to integrate semiconductor optical amplifiers along with passive waveguide structures should mature sufficiently to find application in a variety of systems. Growing demand for WDM transmission systems should stimulate development of components that are more highly integrated and more cost-effective. These trends in electronic and optical technology all appear to favor network technologies like burst switching. This suggests that such technologies could play an important role in the communication networks of the twenty-first century.

Acknowledgements. Several colleagues and students at Washington University have played an important part in helping to define the architecture of burst switching systems and identifying the key issues that must be addressed in order to realize effective implementations. The key contributors include Toshiya Aramaki, Tom Chaney, Yuhua Chen, Will Eatherton, Andy Fingerhut and Fred Rosenberger. Special thanks to Bert Hui of DARPA for challenging me to take a serious look at new switching architectures capable of exploiting emerging WDM technologies in more fundamental ways His encouragement and support are deeply appreciated.

References

[1] Amstutz, Stanford. “Burst Switching — An Introduction,” *IEEE Communications*, 11/83.

[2] Barry, Richard A. “The ATT/DEC/MIT All-Optical Network Architecture,” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[3] Chaney, Tom, J. Andrew Fingerhut, Margaret Flucke and Jonathan Turner. “Design of a Gigabit ATM Switch,” *Proceedings of Infocom*, 4/97.

[4] Coffman, K. G. and A. M. Odlyzko. “The Size and Growth Rate of the Internet,” unpublished technical report, available at http:// www.research.att.com/ amo/doc/networks.html, 1998.

[5] Cormen, Thomas, Charles Leiserson, Ron Rivest. *Introduction to Algorithms,* MIT Press, 1990.

[6] Gambini, Piero. “State of the Art of Photonic Packet Switched Networks,” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[7] Gustavsson, Mats. “Technologies and Application for Space Switching in Multi-Wavelength Networks,” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[8] Ikegami, Tetsuhiko. “WDM Devices, State of the Art,” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[9] Masetti, Francesco. “System Functionalities and Architectures in Photonic Packet Switching” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[10] Prucnal, Paul and I. Glesk. “250 Gb/s self-clocked Optical TDM with a Polarization Multiplexed Clock,” *Fiber and Integrated Optics*, 1995.

[11] Siemens Semiconductor Group. “Parallel Optical Link (PAROLI) Family,” http://w2.siemens.de /semiconductor/products/37/3767.htm, 1998.

[12] Stubkjaer, K. E., et. al. “Wavelength Conversion Technology,” In *Photonic Networks*, Giancarlo Prati (editor), Springer Verlag 1997.

[13] Turner, Jonathan S., “An Optimal Nonblocking Multicast Virtual Circuit Switch,” *Proceedings of Infocom,* June 1994.

[14] Turner, Jonathan S. and the ARL and ANG staff. “A Gigabit Local ATM Testbed for Multimedia Applications,” Washington University Applied Research Lab ARL-WN-94-11.