Distributed Path Reservation Algorithms for Multiplexed All-Optical Interconnection Networks

Xin Yuan, Rami Melhem, and Rajiv Gupta

Abstract—In this paper, we study distributed path reservation protocols for multiplexed all-optical interconnection networks. The path reservation protocols negotiate the reservation and establishment of connections that arrive dynamically to the network. These protocols can be applied to both *wavelength division multiplexing* (WDM) and *time division multiplexing* (TDM) networks. Two classes of protocols are discussed: forward reservation protocols and backward reservation protocols. Simulations of multiplexed two-dimensional torus interconnection networks are used to evaluate and compare the performance of the protocols and to study the impact of system parameters, such as the multiplexing degree and the network size, speed, and load, on both network throughput and communication delay. The simulation results show that, in most cases, the backward reservation schemes provide better performance than their forward reservation counterparts.

Index Terms—Optical interconnection networks, routing protocols, mesh-like networks, path reservation, wavelength-division multiplexing, time-division multiplexing, distributed control.

1 Introduction

With the increasing computation power of parallel computers, interprocessor communication has become an important factor that limits the performance of supercomputing systems. Due to their capabilities of offering large bandwidth, optical interconnection networks, whose advantages have been well demonstrated on wide and local area networks [1], [7], are promising networks for future supercomputers.

Directly connected networks, such as meshes, tori, rings, and hypercubes, are commonly used in commercial supercomputers. By exploiting space diversity and traffic locality, they offer larger aggregate throughput and better scalability than shared media networks such as buses and stars. Optical direct networks can use either multihop packet routing (e.g., Shuffle Net [2]), or *deflection routing* [5]. The performance of packet routing is limited by the speed of electronics since buffering and address decoding is needed at intermediate nodes. Thus, packet routing cannot efficiently utilize the potentially high bandwidth that optics can provide. With deflection routing, buffering of the optical data signals at intermediate nodes is avoided by routing a packet through alternate paths when multiple packets compete for the same output link. While deflection routing requires simple network nodes and minimal buffering, a mechanism is necessary to guarantee bounded transfer delays within the network.

In order to fully explore the potential of optical communication, optical signals should be transmitted in a pure circuit-switching fashion in the optical domain. No buffering, routing, or protocol decoding should be performed at intermediate nodes during data transmission since these operations require optical-to-electronic and electronic-to-optical conversions, as well as electronic processing. Moreover, multiplexing techniques should be used to fully utilize the large bandwidth of optics and to provide multiple *virtual channels* on each communication link. Two techniques can be used for multiplexing optical signals on a fiber-optics link, namely *time-division multiplexing* (TDM) [10], [12], [15] and *wavelength-division multiplexing* (WDM) [6], [7], [9], [18]. In TDM, a link is multiplexed by having different virtual channels communicate in different *time slots*, while in WDM, a link is multiplexed by having different virtual channels communicate using different *wavelengths*. Both multiplexing techniques increase the number of connections that can be simultaneously established in the network.

All-optical communication requires that a connection which spans more than one link uses the same channel on all the links, thus avoiding wavelength conversion or time-slot interchange at intermediate nodes. Centralized control mechanisms for wavelength assignment [14] or time slot assignment [15] in multiplexed networks are not scalable to large networks. It is, therefore, essential to develop distributed path reservation protocols for all-optical communication in large scale multiplexed networks. Such protocols are studied in this paper.

Two types of protocols are considered and evaluated in the following sections, namely *forward reservation* and *backward reservation* protocols. In the forward reservation protocols, the source node starts allocating the virtual channels along the links toward the destination node once it receives a connection request. In the backward reservation protocols, a *probe* packet travels through the network collecting the virtual channel usage information before the actual allocation starts. These protocols are generalizations of control protocols in nonmultiplexed circuit-switching networks [13]. Multiplexing, however, introduces additional complexity, which requires a careful consideration of many factors and parameters that affect the efficiency of the protocols.

Most studies of all-optical multiplexed networks assume virtual channel assignments [4], [14], few works consider the on-line control mechanisms needed to find these assignments. In [15], a distributed control algorithm to establish connections in multiplexed multistage networks is proposed. In [17], the performances of dynamic reservation protocols are compared while taking into consideration the signaling overheads. The protocols described in the above work fall into the *forward reservation* category. *Backward reservation* schemes for multiplexed networks have not been described and evaluated before. Our study shows that, although backward reservation protocols are slightly more complex than the forward reservation protocols, they consistently provide better performance, except for networks with large connectivities.

The rest of the paper is organized as follows: In Section 2, we discuss the problem of path reservation in multiplexed networks. In Sections 3 and 4, we discuss the distributed control protocols. In Section 5, we present the results of the simulation study and, in Section 6, we give some concluding remarks.

2 Path Reservation in Multiplexed Networks

We consider directly connected networks consisting of switches with a fixed number of input and output ports. All but one input port and one output port are used to interconnect with other switches, while one input port and one output port are used to connect to a local processing element. Each link in the network is multiplexed to support multiple virtual channels. The 5-node linear network shown in Fig. 1 is an example of such networks.

We use Figs. 1 and 2 to illustrate path multiplexing in TDM networks, where two virtual channels are supported on each link by dividing the time domain into time slots and using alternating time slots for the two channels $c0$ and $c1$. Fig. 1 shows four established connections using the two channels, namely connections $(0, 2)$ and $(2, 1)$ that are established using channel $c0$ and

- *X. Yuan is with the Department of Computer Science, Florida State University, Tallahassee, FL 32306. E-mail: xyuan@cs.fsu.edu.*
- *R. Melhem is with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260. E-mail: melhem@cs.pitt.edu.*
- *R. Gupta is with the Department of Computer Science, University of Arizona, Tucson, AZ 85721. E-mail: gupta@cs.arizona.edu.*

For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 110678.

0018-9340/99/$10.00 © 1999 IEEE

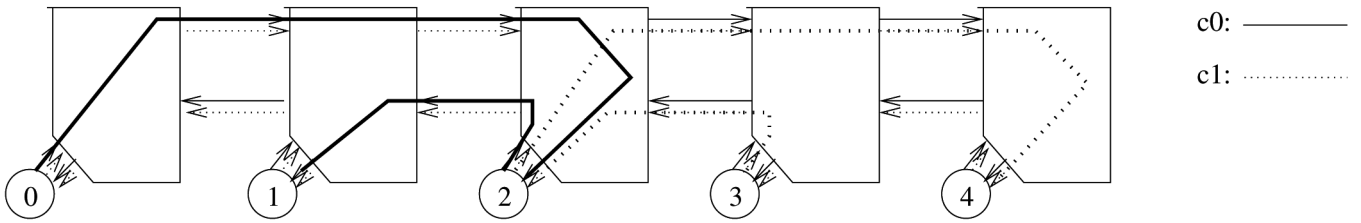

Fig. 1. Path multiplexing in a linear array.

connections $(2,4)$ and $(3,2)$ that are established using channel $c1$. Here, (u,v) denotes a connection from node u to node v. The switches are globally synchronized at time slot boundaries and each switch is set to alternate between the two states that are needed to realize the established connections. For example, Fig. 2 shows the two states that the 3×3 switch attached to processor 2 must realize for the establishment of the connections shown in Fig. 1. Note that each switch can be an electro-optical switch (Ti:$LiNbO_3$ switch, for example [11]) which connects optical inputs to optical outputs without optical/electronic conversion. The state of the switch is controlled by setting electronic bits in a *switch state register*.

The duration of a time slot is typically equal to the duration of several hundred bits [3]. For synchronization purposes, a guard band at each end of a time slot must be used to allow for changing the state of the switches (shifting a shift register) and to accommodate possible drifting or jitter. For example, if the duration of a time slot is $276ns$, which includes a guard band of $10ns$ at each end, then $256ns$ can be used to transmit data. If the transmission rate is $1Gb/s$, then a packet of 256 bits can be transmitted during each time slot. Note that the optical transmission rate is not affected by the relatively slow speed of changing the state of the switches ($10ns$.) since that change is performed only every $276ns$.

Fig. 1 can also be used to demonstrate the establishment of connections in WDM networks, where two different wavelengths are used for the two channels. In such networks, each switch should have the capability of switching signals with different wavelengths independently [19]. Moreover, transmitters and receivers at each node should be tunable to any of the two wavelengths used to implement the channels. Alternatively, two transmitters and two receivers may be used at each node for the different wavelengths.

In order to support a distributed control mechanism for connection establishment, we assume that, in addition to the optical data network, there is a logical *shadow network* through which all the control messages are communicated. The shadow network has the same physical topology as the data network. The traffic on the shadow network, however, consists of small control packets and, thus, is much lighter than the traffic on the data network. The shadow network operates in packet switching mode; routers at intermediate nodes examine the control packets and update local bookkeeping information and switch states accordingly. The shadow network can be implemented as an electronic network or, alternatively, a virtual channel on the data network can be reserved exclusively for exchanging control messages. We also assume that a node can establish multiple virtual channels simultaneously.

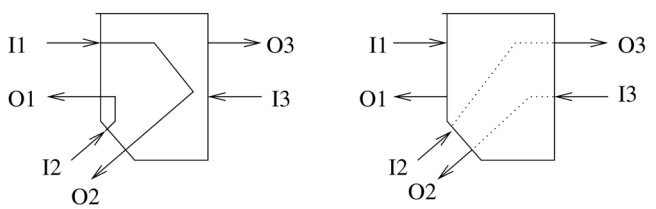

Fig. 2. Changing the state of a switch in TDM. (a) Time slot 0. (b) Time slot 1.

A path reservation protocol ensures that the path from a source node to a destination node is reserved before the connection is used. There are many variations for path reservation which are discussed next.

- *Forward reservation* versus *backward reservation*. Locking mechanisms are needed by the distributed path reservation protocols to ensure the exclusive usage of a virtual channel for a connection. This variation characterizes the timing at which the protocols perform the locking. Under forward reservation, the virtual channels are locked by a control message that travels from the source node to the destination node. Under backward reservation, a control message travels to the destination to probe the path, then virtual channels that are found to be available are locked by another control message which travels from the destination node to the source node.
- *Dropping* versus *holding*. This variation characterizes the behavior of the protocol when it determines that a connection establishment does not progress. Under the dropping approach, once the protocol determines that the establishment of a connection is not progressing, it releases the virtual channels locked on the partially established path and informs the source node that the reservation fails. Under the holding approach, when the protocol determines that the establishment of a connection is not progressing, it keeps the virtual channels on the partially established path locked for some period of time, hoping that, during this period, the reservation will progress. If, after this timeout period, the reservation still does not progress, the partial path is then released and the source node is informed of the failure. Dropping can be viewed as holding with holding time equal to zero.
- *Aggressive* reservation versus *conservative* reservation. This variation characterizes the protocol's treatment of each reservation. Under the aggressive reservation, the protocol tries to establish a connection by locking as many virtual channels as possible during the reservation process. Only one of the locked channels is then used for the connection, while the other channels are released. Under the conservative reservation, the protocol locks only one virtual channel during the reservation process.

The options of holding and dropping are commonly used in the context of call establishment protocols and the concept of backward reservation was previously used in [13] for nonmultiplexed networks.

2.1 Deadlock

Deadlock in the control network can arise from two sources. First, with a limited number of buffers, a request loop can be formed within the control network. Second, deadlock can occur when a

request is holding (locking) virtual channels on some links while requesting other channels on other links. This second source of deadlock can be avoided by the dropping or holding mechanisms described above. Specifically, a request will give up all the locked channels if it does not progress within a certain timeout period.

Many deadlock avoidance or deadlock prevention techniques for packet switching networks proposed in the literature [8] can be used to deal with deadlock caused by the limited buffer capacity. Moreover, the control network is under light traffic and each control message consists of only a single packet of small size. Hence, it is feasible to provide a large number of buffers in each router to reduce or eliminate the chance of deadlock. In the simulations presented in Section 5 for comparing the reservation schemes, we will nullify the effect of deadlock in the control network by assuming an infinite number of control packet buffers at each node. This will allow us to concentrate on the effect of the reservation protocols on the efficiency of the multiplexed data network.

2.2 States of Virtual Channels

The control network router at each node maintains a state for each virtual channel on links connected to the router. A virtual channel, V, on link L, can be in one of the following states:

- $AVAIL$: indicates that the virtual channel V on link L is available and can be used to establish a new connection,
- $LOCK$: indicates that V is locked by some request in the process of establishing a connection.
- $BUSY$: indicates that V is being used by some established connection to transmit data.

For a link, L, the set of virtual channels that are in the $AVAIL$ state is denoted as $Avail(L)$. When a virtual channel, V, is not in $Avail(L)$, an additional field, CID, is maintained to identify the connection request locking V, if V is in the $LOCK$ state, or the connection using V, if V is in the $BUSY$ state.

3 Forward Reservation Schemes

In the connection establishment phase of the protocols, each connection request is assigned a unique identifier, id, which consists of the identifier of the source node and a serial number issued by that node. Each control message related to the establishment of a connection carries its id, which becomes the identifier of the connection, when successfully established. It is this id that is maintained in the CID field of locked or busy virtual channels on links. Four types of packets are used in the forward reservation protocols to establish a connection.

- *Reservation packet* (RES), used to reserve virtual channels. In addition to the connection id, a RES packet contains a bit vector, $cset$, of size equal to the number of virtual channels in each link. The bit vector $cset$ is used to keep track of the set of virtual channels that can be used to satisfy the connection request carried by RES. These virtual channels are locked (their states are changed from $AVAIL$ to $LOCK$) at intermediate nodes, while the RES message progresses toward the destination node (unused channels will be released later). The switch states are also set to connect the locked channels on the input and output links.
- *Acknowledgment packet* (ACK), used to inform the source of the success of a connection request. An ACK packet contains a $channel$ field which indicates the virtual channel selected for the connection. As an ACK packet travels from the destination to the source, it changes the state of the virtual channel selected for the connection to $BUSY$ and unlocks (changes from $LOCK$ to $AVAIL$) all other virtual channels that were locked by the corresponding RES packet (all virtual channels with state $LOCK$ and $CID = ACK.id$).
- *Fail or Negative ack packet*($FAIL/NACK$), used to inform the source of the failure of a connection request. While traveling back to the source node, a $FAIL/NACK$ packet unlocks all virtual channels that were locked by the corresponding RES packet.
- *Release packet*(REL), used to release a connection. A REL packet traveling from a source to a destination changes the state of the virtual channel reserved for that connection from $BUSY$ to $AVAIL$.

The protocols require that a control packet from a destination, d, to a source, s, follows the same path (in the opposite direction) as the packet from s to d. We will denote the fields of a packet by $packet.field$. For example, $RES.id$ denotes the id field of the RES packet.

The forward reservation with dropping works as follows: When the source node wishes to establish a connection, it composes a RES packet with $RES.cset$ equal to the virtual channels that the node may use. This message is then routed to the destination. When an intermediate node receives the RES packet from a link L_i, it determines the next outgoing link, L_o, on the path to the destination, and updates $RES.cset$ to $RES.cset \cap Avail(L_o)$. If the resulting $RES.cset$ is empty, the connection cannot be established and a $FAIL/NACK$ message is sent back to the source node. The source node will retransmit the request after some period of time. This process of failed reservation is shown in Fig. 3a. Note that if $Avail(L_o)$ is represented by a bit-vector, then $RES.cset \cap Avail(L_o)$ is a bit-wise "AND" operation.

If the resulting $RES.cset$ is not empty, the router reserves all the virtual channels in $RES.cset$ on link L_o by changing their states to $LOCK$ and updating $Avail(L_o)$. The router will then set the switch state to connect the virtual channels in the resulting $RES.cset$ of L_o to the corresponding virtual channels in L_i. Maintaining the states of outgoing links is sufficient for these two tasks. The RES message is then forwarded to the next node on the path to the destination. This way, as RES approaches the destination, the path is reserved incrementally. Once RES reaches the destination with a nonempty $RES.cset$, the destination selects from $RES.cset$ a virtual channel to be used for the connection and informs the source node that the channel is selected by sending an ACK message with $ACK.channel$ set to the selected virtual channel. The source can start sending data once it receives the ACK packet. After all data is sent, the source node sends a REL packet to tear down the connection. This successful reservation process is shown in Fig. 3b. Note that, although in the algorithm described above, the switches are set during the processing of the RES packet, they can instead be set during the processing of the ACK packet.

3.1 Holding

The protocol described above can be modified to use the holding policy instead of the dropping policy. Specifically, when an intermediate node determines that the connection for a reservation cannot be established, that is, when $RES.cset \cap Avail(L) = \phi$, the node buffers the RES packet for a limited period of time. If, within this period, some virtual channels in the original $RES.cset$ become available, the RES packet can then continue its journey. Otherwise, the $FAIL/NACK$ packet is sent back to the source. Implementing the holding policy requires each node to maintain a holding queue and to periodically check that queue to determine if any of the virtual channels has become available. In addition, some timing mechanism must be incorporated in the routers to timeout held control packets. This increases the hardware and software complexities of the routers.

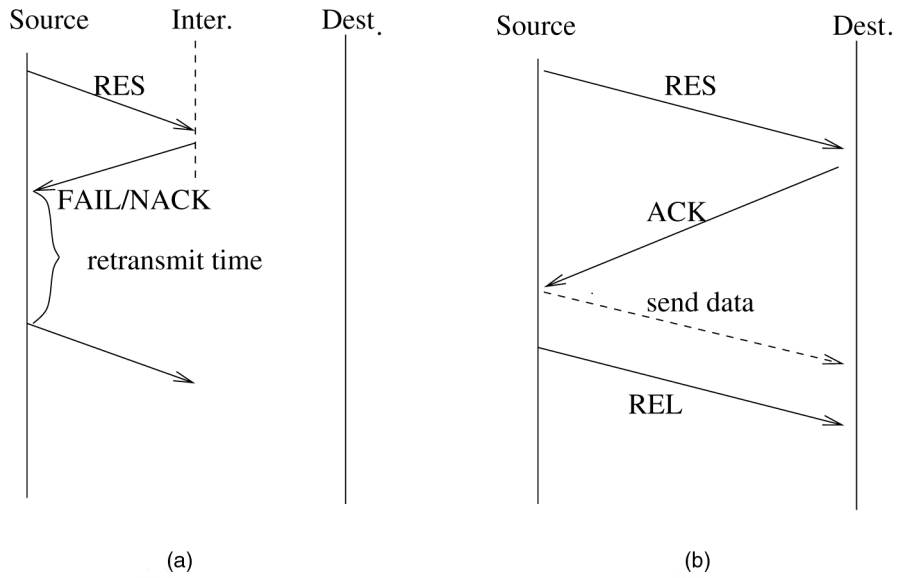

Fig. 3. Control messages in forward reservation.

3.2 Aggressiveness

The aggressiveness of the reservation is reflected in the size of the virtual channel set, $RES.cset$, initially chosen by the source node. In the most aggressive scheme, the source node initializes $RES.cset$ to $\{0, \ldots, N-1\}$, where N is the number of virtual channels in the system. This ensures that the reservation will be successful if there exists an available virtual channel on the path. On the other hand, the most conservative reservation initializes $RES.cset$ to include only a single virtual channel. In this case, the reservation can be successful only when the virtual channel chosen by the source node is available in all the links on the path. Clearly, the aggressive scheme has some advantages over the conservative scheme. Specifically, it reduces the probability that a RES packet will be blocked. This reduction, however, comes at a price of overly locking the virtual channels in the system. In heavily loaded networks, overly locking the channels may decrease the overall throughput. To obtain optimal performance, the aggressiveness of the protocol should be chosen appropriately between the most aggressive and the most conservative extremes.

The retransmit time is another protocol parameter. In traditional nonmultiplexed networks, the retransmit time is typically chosen randomly from a range [0, MRT], where MRT denotes some maximum retransmit time. In such systems, MRT must be set to a reasonably large value to avoid live-lock. However, this may increase the average message latency time and decrease the throughput. In a multiplexed network, the problem of live-lock only occurs in the most aggressive scheme (nonmultiplexed circuit switching networks can be considered as having a multiplexing degree of 1 and using aggressive reservation). For less aggressive schemes, the live-lock problem can be avoided by changing the virtual channels selected in $RES.cset$ when RES is retransmitted. Hence, for these schemes, a small retransmit time can be used.

4 Backward Reservation Schemes

In the forward locking protocol, the initial decision concerning the virtual channels to be locked for a connection request is made in the source node without any information about network usage. The backward reservation scheme tries to overcome this handicap by probing the network before making the decision. In the backward reservation schemes, a forward message is used to probe the availability of virtual channels (see Fig. 4). After that, the locking of virtual channels is performed by a backward message. The backward reservation scheme uses six types of control packets, all of which carry the connection id, in addition to other fields as discussed next:

- *Probe packet* ($PROB$), that travels from source to destination gathering information about virtual channel usage without locking any virtual channel. A $PROB$ packet carries a bit vector, $init$, to represent the set of virtual channels that are available to establish the connection.
- *Reservation packet* (RES), similar to the RES packet in the forward scheme, except that it travels from destination to source, locking virtual channels as it goes through intermediate nodes, and setting the states of the switches accordingly. A RES packet contains a $cset$ field.
- *Acknowledgment packet* (ACK), similar to the ACK packet in the forward scheme except that it travels from source to destination. An ACK packet contains a $channel$ field.
- *Fail packet* ($FAIL$), to unlock the virtual channels locked by the RES packet in case of connections establishment failure.
- *Negative acknowledgment packet* ($NACK$), used to inform the source of reservation failure.
- *Release packet* (REL), used to release connections after the communication is completed.

Note that a $FAIL/NACK$ message in forward schemes performs the functions of both a $FAIL$ message and a $NACK$ message in backward schemes. Specifically, in backward reservation, when a RES packet is blocked on its route from the destination to the source, a $FAIL$ should be sent to the destination to unlock the channels that were locked by the RES and a $NACK$ should be sent to the source to reinitiate the circuit establishment. In forward reservation, RES travels from the source to the destination and, thus, the $NACK$ and $FAIL$ messages may be combined.

The backward reservation with dropping works as follows: When the source node wishes to establish a connection, it composes a $PROB$ message with $PROB.init$ set to contain all virtual channels in the system. This message is then routed to the destination. When an intermediate node receives the $PROB$ packet from an incoming link L_i, it determines the next outgoing link, L_o, on the forward path to the destination and updates $PROB.init$ to

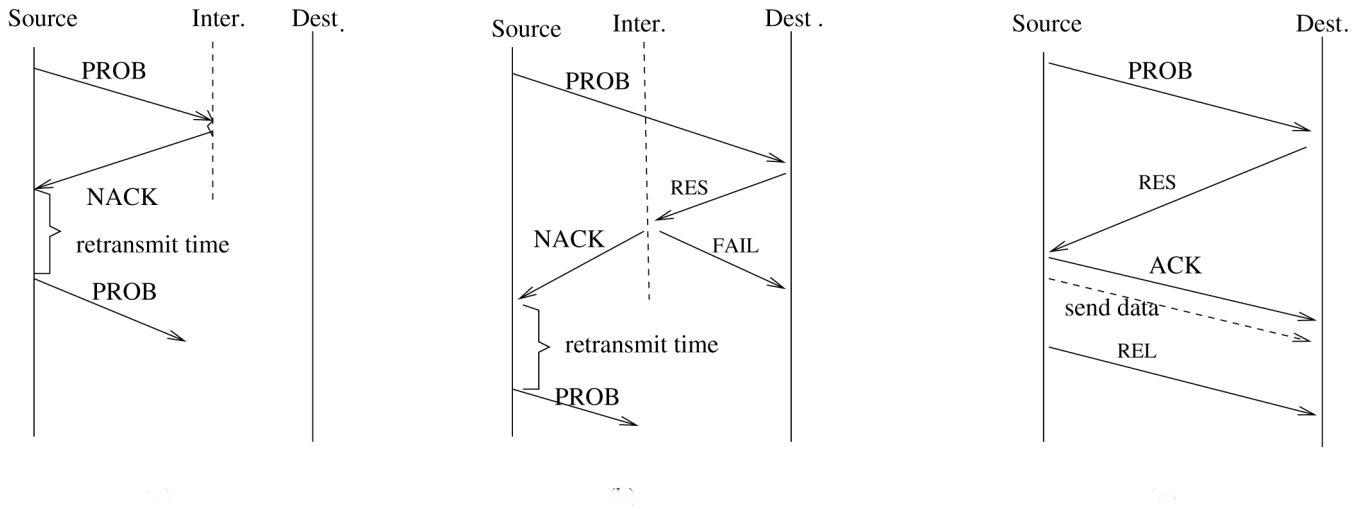

Fig. 4. Control messages in backward reservation.

$PROB.init \cap Avail(L_o)$. If the resulting $PROB.init$ is empty, the connection cannot be established and a $NACK$ packet is sent back to the source node. The source node will try the reservation again after a certain retransmit time. Fig. 4a shows this failed reservation case.

If the resulting $PROB.init$ is not empty, the node forwards $PROB$ on L_o to the next node. This way, as $PROB$ approaches the destination, the virtual channels available on the path are recorded in the $init$ set. Once $PROB$ reaches the destination, the destination forms a RES message with $RES.cset$ equal to a selected subset of $PROB.init$ and sends this message back to the source node. When an intermediate node receives the RES packet, it determines the next link, L_b, on the backward path to the source, and updates $RES.cset$ to $RES.cset \cap Avail(L_b)$. If the resulting $RES.cset$ is empty, the connection cannot be established. In this case, the node sends a $NACK$ message to the source node to inform it of the failure, and sends a $FAIL$ message to the destination to free the virtual channels locked by RES. This process is shown in Fig. 4b.

If the resulting $RES.cset$ is not empty, the virtual channels in $RES.cset$ are locked, the switch is set accordingly, and RES is forwarded on L_b to the next node. When RES reaches the source with a nonempty $RES.cset$, the source selects a virtual channel from the $RES.cset$ for the connection and sends an ACK message to the destination with $ACK.channel$ set to the selected virtual channel. This ACK message unlocks all the virtual channels locked by RES, except the one in $channel$. The source node can start sending data as soon as it sends the ACK message. After all data is sent, the source node sends a REL packet to tear down the connection. The process of successful reservation is shown in Fig. 4c.

4.1 Holding

Holding can be incorporated in the backward reservation scheme as follows: In the protocol, there are two cases that cause the reservation to fail. The protocol may determine that the reservation fails when processing the $PROB$ packet. In this case, no holding is necessary since no resources have yet been locked. When the protocol determines that the reservation fails during the processing of a RES packet, a holding mechanism similar to the one used in the forward reservation scheme may be applied.

4.2 Aggressiveness

The aggressiveness of the backward reservation protocols is reflected in the initial size of $cset$ chosen by the destination node. The most aggressive approach sets $RES.cset$ equal to $PROB.init$, while the most conservative approach sets $RES.cset$ to contain a single virtual channel from $PROB.init$. Note that if a protocol supports only the most conservative scheme, the ACK messages may be omitted and, thus, only five types of messages are needed. As in the forward reservation schemes, the aggressiveness can be chosen between the two extremes. Also, as in the forward schemes, the retransmit time is a parameter in the backward schemes.

5 Performance Evaluation

In the following discussion, we will use F to denote forward reservation, B to denote the backward reservation, H for holding, and D for dropping schemes. For example, FH means the forward holding scheme. We have implemented a network simulator with various control mechanisms including FH, FD, BH, and BD. In addition to the options of backward/forward reservation and holding/dropping policy, the simulation uses the following protocol parameters:

- *Initial cset size*: This parameter determines the initial size of $cset$ in the reservation packet. It restricts the set of virtual channels under consideration for a reservation.
- *Timeout value*: This value determines how long a reservation packet can be put in a waiting queue and can be a function of the end-to-end delay. The dropping scheme can be considered as a holding scheme with timeout time equal to 0.
- *Maximum retransmit time* (MTR): This specifies the period after which a node will retry a failed reservation. As discussed earlier, this value is crucial for avoiding live-lock in the most aggressive schemes. The actual retransmit time is chosen randomly between 0 and $MRT - 1$.

Other system parameters used in the simulation are listed below.

- *System size*: This specifies the size of the network. All the simulation results presented in Sections 5.1 and 5.2 are for torus topologies assuming X–Y routing [8]. Results for other topologies are given in Section 5.3.
- *Multiplexing degree*. This specifies the number of virtual channels supported by each link.
- *Message size*: This specifies the number of packets in each message.
- *Request generation rate at each node (r)*: This specifies the traffic in the network. The connection requests at each node is assumed to have a Poisson interarrival distribution

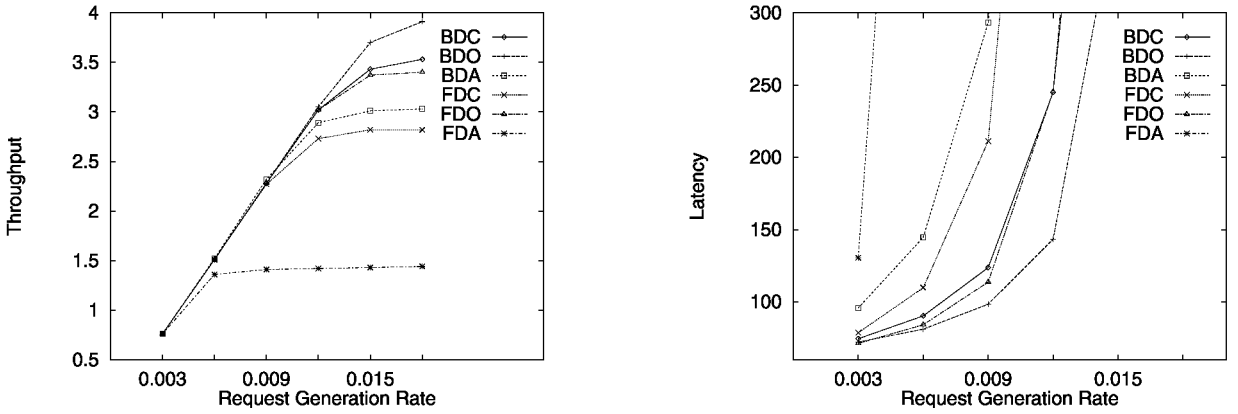

Fig. 5. Comparison of the reservation schemes with dropping.

with an average of r messages generated every time unit. When a request is generated at a node, the destination of the request is generated randomly among the other nodes in the system. When a generated request is blocked, it is put into a queue, waiting to be retransmitted.

- *Control packet processing and propagation time* (γ): This specifies the speed of the control network. The control packet processing time is the time for an intermediate node to process a control packet. The control packet propagation time is the time for a control packet to be transferred from one node to the next. We assume that all the control packets have the same processing and propagation time.

We use the average latency and throughput to evaluate the protocols. The latency is the period between the time when a message is ready and the time when the first packet of the message is sent. The throughput is the number of messages received at the destination nodes per time unit. Under light traffic, the performance of the protocols is measured by the average message latency, while, under heavy traffic, the throughput is used as the performance metric. The simulation time is measured in time slots, where a time slot is the time to transmit an optical data packet between any two nodes in the network. Note that, in multiprocessing applications, nodes are physically close to each other (a few feet apart) and, thus, signal propagation time is very small (one foot per nsec) compared to the time it takes to transmit a packet (256ns to transmit 256 bits at 1Gb/s).

The results presented in this section are for TDM networks. The performance of WDM networks can be approximated by an equivalent TDM network in which the bandwidth of the data links are multiplied by the multiplexing degree. Specifically, the throughput of a TDM network with multiplexing degree k and control packet processing and propagation time γ is approximately equal to $\frac{1}{k}$ of the throughput of a WDM network with multiplexing degree k and control packet processing and propagation time equal to $\frac{\gamma}{k}$.

Fig. 5 depicts the throughput and average latency as a function of the request generation rate for six protocols that use the dropping policy in a 16×16 torus. The multiplexing degree is taken to be 32, the message size is assumed to be eight packets, and γ is assumed to be two time units. For each of the forward and backward schemes, three variations are considered with varying aggressiveness: the conservative variation in which the initial $cset$ size is 1, the most aggressive variation in which the initial set size is equal to the multiplexing degree, and an optimal variation in which the initial set size is chosen (by repeated trials) to maximize the throughput. The letters C, A, and O are used to denote these three variations, respectively. For example, FDO means the forward dropping scheme with optimal $cset$ size.

From the figure, it is clear that FDA is the least efficient, which is due to the unnecessary locking of all the channels. Also, FDC is more efficient than FDA, but yet suffers from a high blocking probability because reservation is attempted on only one channel. In general, backward reservations result in a larger bandwidth and smaller latency than the corresponding forward reservations. This is mainly due to the probing of the available channels before locking them. Finally, the use of the optimal $cset$ size (larger than 1 and smaller than 32) reduces the delay in addition to increasing the throughput.

Fig. 5 reveals that, when the request generation rate, r, is small, for example, $r = 0.003$, the network is under light traffic and all the protocols achieve the same throughput, which is equal to r times the number of processors. In this case, the performance of the network should be measured by the average latency. In the rest of the performance study, we will use the maximum throughput (at saturation) and the average latency (at $r = 0.003$) to measure the performance of the protocols. We perform three sets of experiments. The first set evaluates the effect of the protocol parameters on the network throughput and delay, the second set evaluates the impact of system parameters on performance, and the third set evaluates the effect of network connectivity on performance.

5.1 Effect of Protocol Parameters

In this set of experiments, we study the effect of the initial $cset$ size, the holding time, and the retransmit time on the performance of the protocols. The system parameters for this set of experiment are chosen as follows: System size = 16×16, message size = 8, and γ = 2 time units.

Fig. 6 shows the effect of the initial $cset$ size on the forward holding scheme with different multiplexing degrees, namely 1, 2, 4, 8, 16, and 32. The holding time is taken to be 10 time units and the MTR is 5 time units for all the protocols with initial $cset$ size less than the multiplexing degree and 60 time units for the most aggressive forward scheme. Large MTR is used in the most aggressive forward scheme because we observed that small MTR often leads to live-lock in that scheme. We show only the protocols with the holding policy since using the dropping policy leads to similar patterns. Fig. 7 shows the results for the backward schemes with the dropping policy.

From Fig. 6, we can see that, when the multiplexing degree is larger than 8, both the most conservative protocol and the most aggressive protocol do not achieve the best throughput. Fig. 6 also shows that these two extreme protocols do not achieve the smallest latency either. The same observation applies to the backward schemes in Fig. 7. The effect of choosing the optimal initial $cset$ is significant on both throughput and delay. That effect, however, is more significant in the forward scheme than in the backward scheme. For example, with multiplexing degree = 32, choosing a

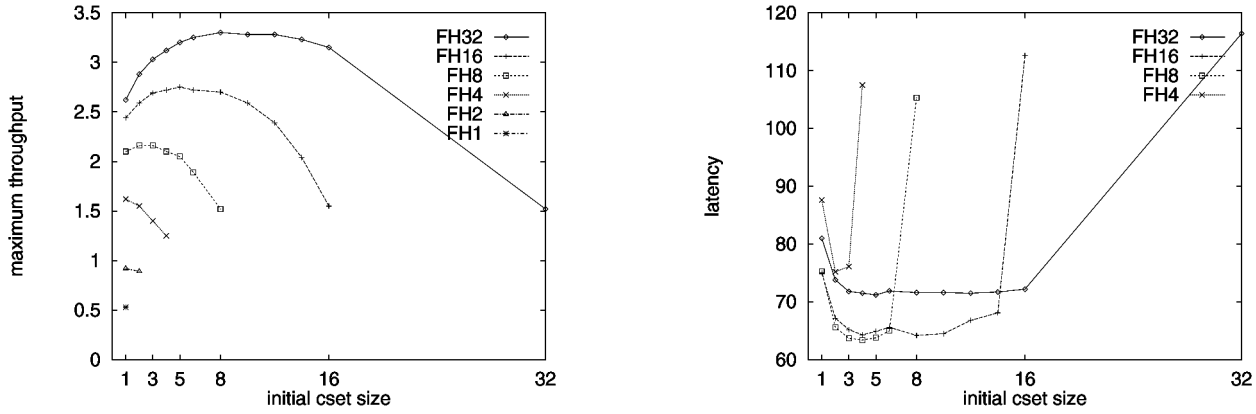

Fig. 6. Effect of the initial $cset$ size on forward schemes.

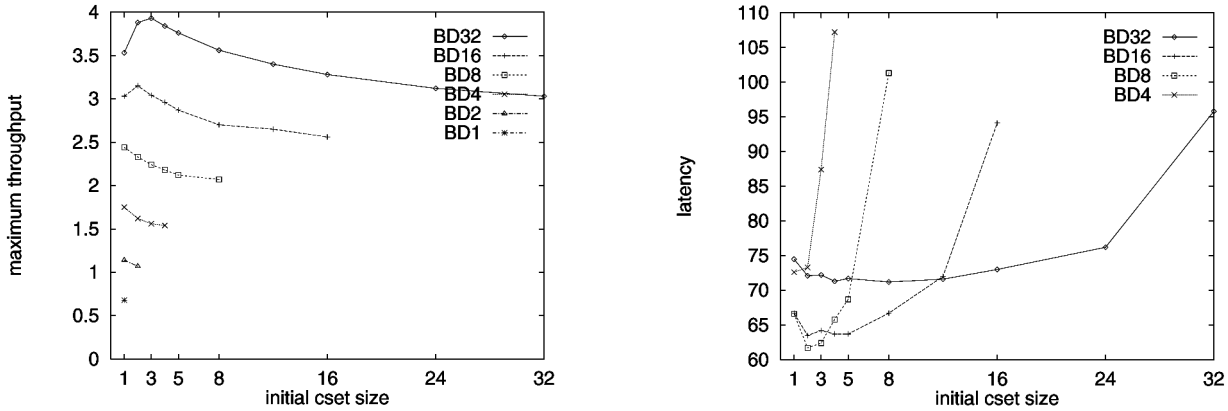

Fig. 7. Effect of the initial $cset$ size on backward schemes.

nonoptimal $cset$ size may reduce the throughput by 50 percent in the forward scheme and only by 25 percent in the backward scheme. In general, the optimal initial $cset$ size is hard to find. Note that the optimal $cset$ size is small in the backward scheme because the information gathered by the $PROB$ packet is used when forming $cset$. In fact, a $cset$ size larger than one is only needed to cope with the probability that the state of a virtual channel has changed after the $PROB$ recorded it.

We have conducted experiments to study the effect of the holding time on the performance of the protocols. The results (see [20]) indicate that the holding time has little effect on the maximum throughput. It slightly increases the performance for the aggressive schemes. Since holding requires extra hardware support compared to dropping, we conclude that holding is not cost-effective for the reservation protocols.

We have also conducted experiments to study the effect of the retransmit time (MRT) on the performance and found that increasing MRT results in performance degradation in all the schemes except FDA, in which the performance improves with the MRT [20]. This confirms that the MRT value is important to avoid live-lock in the network when aggressive reservation is used. In other schemes, this parameter is not important because, when retransmitting a failed request, virtual channels different than the ones that have been tried are included in $cset$. This result indicates another drawback of the forward aggressive schemes: In order to avoid live-lock, the MRT must be a reasonably large value, which decreases the overall performance.

In the next section, we will only consider dropping schemes with MRT equal to five time units for all schemes except FDA, whose MRT is set to be 60.

5.2 Effect of Other System Parameters

This set of experiments focuses on studying the performance of the protocols under different multiplexing degrees, system sizes, message sizes, and control network speeds.

Fig. 8 shows the performance of the protocols for different multiplexing degrees, assuming a 16×16 torus, eight packets per message, and $\gamma = 2$ time units. When the multiplexing degree is small, BDO and FDO have the same maximum bandwidth as BDC and FDC, respectively. That is, the conservative schemes are optimal. When the multiplexing degree is large, BDO and FDO offer better throughput. In addition, for all multiplexing degrees, BDO is the best among all the schemes. As for the average latency, aggressive schemes have significantly larger latency than all other schemes. Also, FDO and BDO have the smallest latencies. We can see from this experiment that the backward schemes always provide the same or better performance (both maximum throughput and latency) than their forward reservation counterparts for all multiplexing degrees considered.

Fig. 9 shows the effect of the network size (when the message size = 8 packets) and the effect of the message size (when network size is 256) on the performance of the protocols. For this experiment, the multiplexing degree = 16 and $\gamma = 2$. We can see from the figure that all the protocols, except the aggressive ones, scale nicely with the network size. This indicates that the aggressive protocols cannot take advantage of the spatial diversity of the communication. This is a result of excessive reservation of channels. When the network size is small, there is little difference in the performance of the protocols. When the network size is larger, the backward schemes show their superiority.

When comparing the performance for different message sizes in Fig. 9, we normalize the throughput to reflect the number of

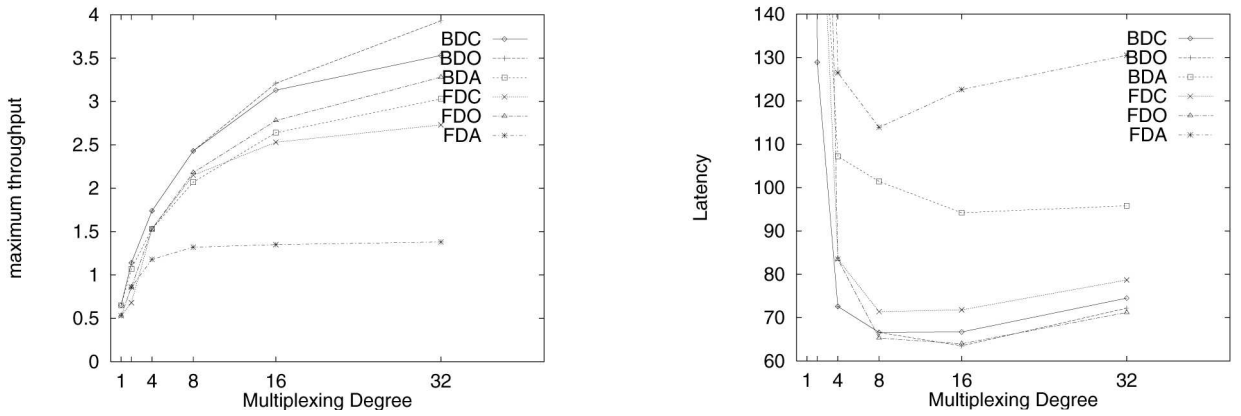

Fig. 8. The performance of the protocols for different multiplexing degree.

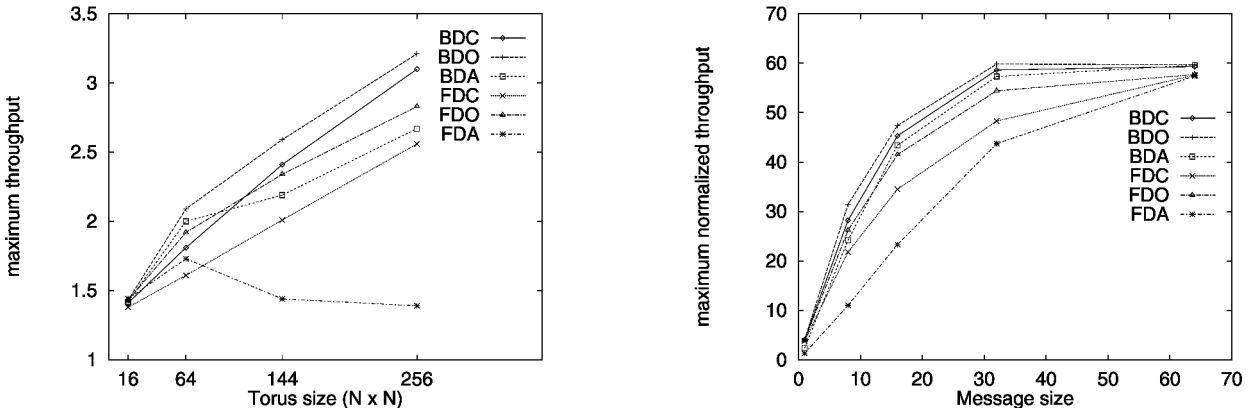

Fig. 9. Effect of the network size and the message size on Throughput.

TABLE 1
Maximum Throughput for Different Topologies

Topology	FDC	BDC	FDO	BDO	FDA	BDA
256–node ring	0.032	0.120	0.032(1)	0.120(1)	0.010	0.040
16×16 torus	2.39	2.99	2.91(8)	3.41(3)	1.04	2.56
$8 \times 8 \times 4$ torus	3.02	4.67	4.42(16)	4.76(2)	2.90	3.90
256–node hypercube	5.45	6.05	7.57(22)	7.63(5)	6.89	5.66

packets that pass through the network, rather than the number of messages. That is,

$$normalized\ throughput\ = msg\ size \times throughput.$$

Both the forward and backward locking schemes achieve higher throughput for larger messages. When messages are sufficiently large, the signaling overhead in the protocols is small and all protocols have almost the same performance. However, when the message size is small, BDO achieves higher throughput than the other schemes. This indicates that BDO incurs less overhead in the path reservation than the other schemes.

Finally, we have studied the effect of the control network speed on performance [20]. The speed of the control network is determined by the time for a control packet to be transferred from one node to the next node and the time for the control router to process the control packet. Simulation results indicate that, when the control speed is slower, the maximum throughput and the average latency degrade. The most aggressive schemes in both forward and backward reservations, however, are more sensitive to the control network speed. Hence, it is important to have a reasonably fast control network when these reservation protocols are used.

5.3 Effect of Network Connectivity

In this section, we consider the effect of network connectivity on the performance of the reservation protocols. Specifically, we show, in Table 1, the maximum throughput obtained by applying the protocols to a 256-node ring, a 16×16 two-dimensional torus, an $8 \times 8 \times 4$ three-dimensional torus, and a 256-node hypercube. This experiment assumes that the multiplexing degree = 32 and the message size = 8. The numbers inside the parentheses in the columns for FDO and BDO are the optimal $cset$ size.

The results of this experiment indicate that, when the network connectivity increases, the advantage of the backward schemes over the forward schemes decreases. The reason is that backward schemes use more control messages than the forward schemes in order to chose the most appropriate channels and to avoid unnecessarily locking channels. As the network connectivity and, thus, the number of available channels, increases, the advantages of carefully choosing and locking channels diminish and the overhead of using more control messages negatively affects the

performance. When the benefits of smarter channel selection in the backward schemes cannot offset the extra overheads, the backward schemes can perform worse than the forward schemes as is the case in the aggressive schemes on the hypercube topology.

6 Concluding Remarks

In this paper, we have described various protocols for virtual path reservation in directly connected, multiplexed, all-optical networks. The protocols are classified into two categories: forward reservation and backward reservation. Extensive experiments were carried out to compare these two classes of protocols. More specifically, we studied the effect of the initial $cset$ size (which determines the number of channels that are locked in each reservation attempt), the holding time, and the retransmit time on the protocols. We found the following results about the protocols: First, the initial $cset$ size largely affects the performance. For large multiplexing degree, the optimal $cset$ size generally lies between the two obvious extremes of locking one channel and locking all the channels. Choosing the optimal $cset$ size can improve the performance by about 50 percent in the forward reservation schemes and 25 percent in the backward schemes. Second, the holding mechanism, which requires additional hardware, does not improve the performance of the protocols in a tangible way and, thus, is not cost-effective. Third, although the retransmit time is an important factor for nonmultiplexed networks, it does not affect the performance of a multiplexed network except when the forward aggressive scheme is used.

We also studied the effect of the system parameters, such as the multiplexing degree, the network size, the message size, and the relative speed of the control network on the performance of the protocols. We found that, for large message sizes and fast control networks, the control overhead is small compared to the data transmission time and, thus, all the protocols exhibit the same performance. When the control overhead is significant, the backward schemes always offer better performance than their forward counterparts. Irrespective of the control protocol, the results show that multiplexing the network always increases its throughput and, up to a certain multiplexing degree, always decreases the average message delay.

There are two main advantages for multiplexing optical networks. First, multiplexing increases the number of connections that can be simultaneously established in the network, thus increasing the chance of successfully establishing a connection. This reduces the traffic in the control network, which in turn reduces the control overhead. The second advantage of multiplexing optical networks is to bridge the gap between the large bandwidth of optical transmission and the low data generation rate at each node, especially if transmitted data is to be fetched from memory. In other words, if data cannot be fetched from memory fast enough to match the optical transmission bandwidth, then dedicating an optical path to one connection will waste communication bandwidth. In such cases, multiplexing allows the large optical bandwidth to be shared among multiple connections. The effect of such a mismatch between memory speed and optical bandwidth is studied in [21].

Acknowledgments

A preliminary version of this paper appeared in the *Proceedings of the Third International Symposium on High Performance Computer Architecture* (HPCA 3), 1997. This work was supported in part by U.S. National Science Foundation awards CCR-9157371 and MIP 9633729.

References

[1] H.R. As, "Media Access Techniques: The Evolution towards Terabit/s LANs and MANs," *Computer Networks and ISDN Systems,* vol. 26, pp. 603-656, 1994.

[2] A.S. Acampora and M.J. Karol, "An Overview of Lightwave Packet Network," *IEEE Network Magazine,* vol. 3, no. 1, pp. 29-41, 1989.

[3] R. Ballart and Y. Ching, "SONET: Now It's the Standard Optical Network," *IEEE Comm. Magazine,* vol. 26, no. 3, pp. 8-15, 1989.

[4] R.A. Barry and P.A. Humblet, "Models of Blocking Probability in All-Optical Networks with and without Wavelength Changers," *Proc. IEEE Infocom,* pp. 402-412, Apr. 1995.

[5] J. Brassil, A.K. Choudhury, and N.F. Maxemchuk, "The Manhattan Street Network: A High Performance, Highly Reliable Metropolitan Area Network," *Computer Networks and ISDN Systems,* vol. 26, nos. 6-8, pp. 841-858, 1994.

[6] C.A. Brackett, "Dense Wavelength Division Multiplexing Networks: Principles and Applications," *IEEE J. Selected Areas of Comm.,* vol. 8, pp. 948-964, Aug. 1990.

[7] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath Communications: An Approach to High Bandwidth Optical WAN's," *IEEE Trans. Comm.,* vol. 40, no. 7, July 1992.

[8] W. Dally and C. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers,* vol. 36, no. 5, May 1987.

[9] P. Dowd, K. Bogineni, and K. Ali, "Hierarchical Scalable Photonic Architectures for High-Performance Processor Interconnection," *IEEE Trans. Computers,* vol. 42, no. 9, pp. 1,105-1,120, Sept. 1993.

[10] A. Ganz and Y. Gao, "A Time-Wavelength Assignment Algorithm for WDM Start Networks," *Proc. of IEEE INFOCOM,* 1992.

[11] H.S. Hinton, "Photonic Switching Using Directional Couplers," *IEEE Comm. Magazine,* vol. 25, no. 5, pp. 16-26, 1987.

[12] R. Melhem, "Time-Multiplexing Optical Interconnection Network; Why Does it Pay Off?" *Proc. 1995 ICPP Workshop Challenges for Parallel Processing,* pp. 30-35, 1995.

[13] S. Nugent, "The iPSC/2 Direct-Connect Communications Technology," *Proc. Third Conf. Hypercube Concurrent Computers and Application,* vol. 1, Jan. 1988.

[14] R. Ramaswami and K. Sivarajan, "Optimal Routing and Wavelength Assignment in All-Optical Networks," *Proc. IEEE Infocom,* vol. 2, pp. 970-979, June 1994.

[15] C. Qiao and R. Melhem, "Reconfiguration with Time Division Multiplexed MIN's for Multiprocessor Communications," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 4, Apr. 1994.

[16] C. Qiao and R. Melhem, "Reducing Communication Latency with Path Multiplexing in Optically Interconnected Multiprocessor Systems," *Proc. Int'l Symp. High Performance Computer Architecture,* 1995.

[17] C. Qiao and Y. Mei, "Wavelength Reservation under Distributed Control," *Proc. IEEE/LEOS Summer Topical Meeting: Broadband Optical Networks,* Aug. 1996.

[18] S. Subramanian, M. Azizoglu, and A. Somani, "Connectivity and Sparse Wavelength Conversion in Wavelength-Routing Networks," *Proc. INFOCOM '96,* 1996.

[19] R.E. Wagner, R.C. Alferness, A.A.M. Saleh, and M.S. Goodman, "MONET: Multiwavelength Optical Networking," *IEEE/OSA J. Lightwave Technology,* vol. 14, no. 6, pp. 1,349-1,355, 1996.

[20] X. Yuan, "Dynamic and Compiled Communication in Optical Time-Division Multiplexed Point-to-Point Networks," PhD dissertation, Dept. of Computer Science, Univ. of Pittsburgh, 1998.

[21] X. Yuan, R. Gupta, and R. Melhem, "Does Time-Division Multiplexing Close the Gap between Memory and Optical Communication Speeds," *Proc. Workshop Parallel Computing, Routing, and Comm. (PCRCW 97),* Atlanta, Ga., June 1997.