# Distributed path reservation algorithms for multiplexed all–optical interconnection networks *

X. Yuan, R. Melhem and R. Gupta
Department of Computer Science
The Univ. of Pittsburgh
{xyuan,melhem,gupta}@cs.pitt.edu

**Abstract**

*In this paper, we study distributed path reservation protocols for multiplexed all–optical interconnection networks. In such networks, a path for a connection is reserved such that transmitted data remains in the optical domain until it reaches its destination. The path reservation protocols negotiate the reservation and establishment of connections that arrive dynamically to the network. They can be applied to both* wavelength division multiplexing *(WDM) and* time division multiplexing *(TDM), which are two techniques that allow the large optical bandwidth to be shared among multiple connections. Two classes of protocols are discussed: forward reservation protocols and backward reservation protocols. Simulations of multiplexed 2-dimensional torus interconnection networks are used to evaluate and compare the performance of the protocols, and to study the impact of system parameters on both network throughput and communication delay. The simulation results show that the backward reservation schemes provide better performance than their forward reservation counterparts.*

## 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 (WAN and LAN) [1, 6], 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 multi-hop packet routing (e.g.Shuffle Net [2]), or *deflection routing* [4]. The performance of packet routing is limited by the speed of electronics since buffering and address decoding usually requires electronic-to-optics and optics-to-electronic conversions at intermediate nodes. Thus, packet routing cannot efficiently utilize the potentially high bandwidth that optics can provide. While deflection routing requires simple network nodes and minimal buffering, a mechanism is necessary to guarantee bounded transfer delays within the network. As pointed out in [1], although direct optical networks have intrinsically high aggregate throughput, this advantage comes at the expense of additional control complexity in the form of routing and congestion control. New solutions should exploit the inherent flexibility of dynamic reconfiguration of logical topologies.

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 and optical-to-electronic or electronic-to-optical conversions should be needed at intermediate nodes. 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) [7, 9, 12] and *wavelength–division multiplexing* (WDM) [5, 6, 15]. 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*.

Regardless of the multiplexing technique, two approaches can be used to establish connections in multiplexed networks, namely *link multiplexing* (LM) and *path multiplexing* (PM). In LM, a connection which spans more than one communication link is established by using possibly different channels on different links. In PM, a connection which spans more than one communication link uses the same channel on all the links. In other words, PM uses the same time-slot or the same wavelength on all the links of a connection, while LM can use different time-slots or different wavelengths, thus requiring time-slot interchange or wavelength conversion capabilities at each intermediate node.

Centralized control mechanisms for wavelength assignment [11] or time slot assignment [12] in multiplexed networks, are not scalable to large networks. It is, therefore, essential to develop distributed path reservation protocols for all–optical communication in

*This work was supported in part by NSF awards CCR-9157371 and MIP-9633729.

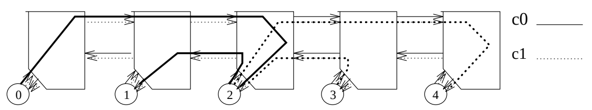

Figure 1: Path multiplexing in a linear array

large scale multiplexed networks. Such protocols are studied in this paper. For simplicity of the presentation, the protocols will be presented for path multiplexing. Similar, and in fact somewhat simpler, protocols can be designed for link multiplexing by removing the restriction that the same virtual channel should be used on all the links forming a connection.

Two types of protocols are considered and evaluated in the following sections, namely *forward reservation* and *backward reservation* protocols. These protocols are generalizations of control protocols in non-multiplexed circuit–switching networks [10]. Multiplexing, however, introduces additional complexity which requires a careful consideration of many factors and parameters that affect the efficiency of the protocols.

Most studies on all–optical multiplexed networks assume virtual channel assignments [3, 11], but only a few works consider the on-line control mechanisms needed to find these assignments. In [12], a distributed control algorithm to establish connections in multiplexed multistage networks is proposed. In [14], the performances of PM and LM are compared while taking into consideration the signaling overhead in the protocols. The protocols described in the above works fall into the *forward reservation* category. *Backward reservation* schemes for multiplexed networks have not been described and evaluated before.

The rest of the paper is organized as follows. In section 2, we discuss the problem of path reservation in multiplexed networks. In Section 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 conclude the paper.

## 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.

We use Figures 1 and 2 to illustrate path multiplexing in TDM networks, where two virtual channels are used on each link by dividing the time domain into time slots, and using alternating time slots for the two channels $c0$ and $c1$. Figure 1 shows four established connections using the two channels, namely connections $(0, 2)$ and $(2, 1)$ that are established using channel $c0$, and connections $(2, 4)$ and $(3, 2)$ that are established using channel $c1$, where $(u, v)$ is used to denote 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

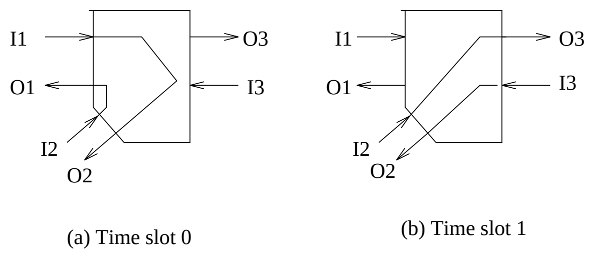

Figure 2: Changing the state of a switch in TDM

two states that are needed to realize the established connections. For example, Figure 2 shows the two states that the $3 \times 3$ switch attached to processor 2 must realize for the establishment of the connections shown in Figure 1. Note that each switch can be an electro-optical switch (Ti:LiNbO$_3$ switch, for example [8]) which connects optical inputs to optical outputs without optical/electronic conversion.

The duration of a time slot is typically equal to the duration of several hundred bits. 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 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 $276ns$.

Figure 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 to switch signals with different wavelengths independently. 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 send or receive messages through different virtual channels simultaneously.

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 options 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 0.

• *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 others are released. Under the conservative reservation, the protocol locks only one virtual channel during the reservation process.

### Deadlock

Deadlock in the control network can arise from two sources. First, with 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 can be used to deal with deadlock within the control network (the first source of deadlock). Moreover, the control network is under light traffic, and each control message consists of only a single packet of small size (4 bytes). 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.

### States of Virtual Channels

The control network router at each node maintains a state for each virtual channel on links connected to the router. For forward reservation, the control router maintains the states for the outgoing links, while in backward reservation, the control router maintains the states for the incoming links. As discussed later, this setting enables the router to have the information needed for reserving virtual channels and updating the switch states. 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 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 packets* ($RES$), 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 at intermediate nodes while the $RES$ message progresses towards the destination node. The switch states are also set to connect the locked channels on the input and output links.

• *Acknowledgment packets* ($ACK$), used to inform source nodes of the success of connection requests.

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.

• *Fail or Negative ack packets* ($FAIL/NACK$), used to inform source nodes of the failure of connection requests. While traveling back to the source node, a $FAIL/NACK$ packet unlocks all virtual channels that were locked by the corresponding $RES$ packet.

• *Release packets* ($REL$), used to release connections. 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 control packets from a destination, $d$, to a source, $s$, follows the same paths (in opposite directions) as packets 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 $RES.cset$ set 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, it determines the next outgoing link, $L$, on the path to the destination, and updates $RES.cset$ to $RES.cset \cap Avail(L)$. 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 Figure 3(a). Note that if $Avail(L)$ is represented by a bit-vector, then $RES.cset \cap Avail(L)$ is a bit-wise "$AND$" operation.

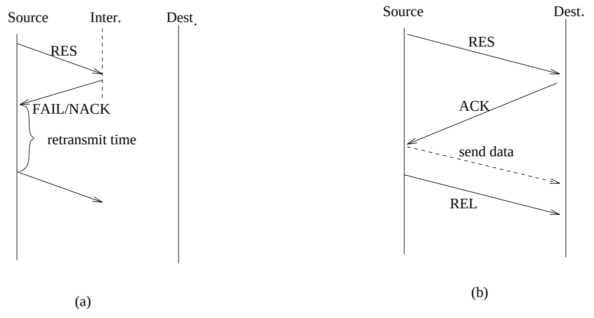

Figure 3: Control messages in forward reservation

If the resulting $RES.cset$ is not empty, the router reserves all the virtual channels in $RES.cset$ on link $L$ by changing their states to $LOCK$ and updating $Avail(L)$. The router will then set the switch state to connect the virtual channels in the resulting $RES.cset$ of the corresponding incoming and outgoing links. 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 non-empty $RES.cset$, 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 Figure 3 (b). 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.

**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.

**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 sets $RES.cset$ to $\{0, ..., N-1\}$, 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 assigns $RES.cset$ 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. Although the aggressive scheme seems to have advantage over the conservative scheme, it results in overly locking the virtual channels in the system. Thus, in heavily loaded networks, this is expected to 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 non–multiplexed 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 (non–multiplexed 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.

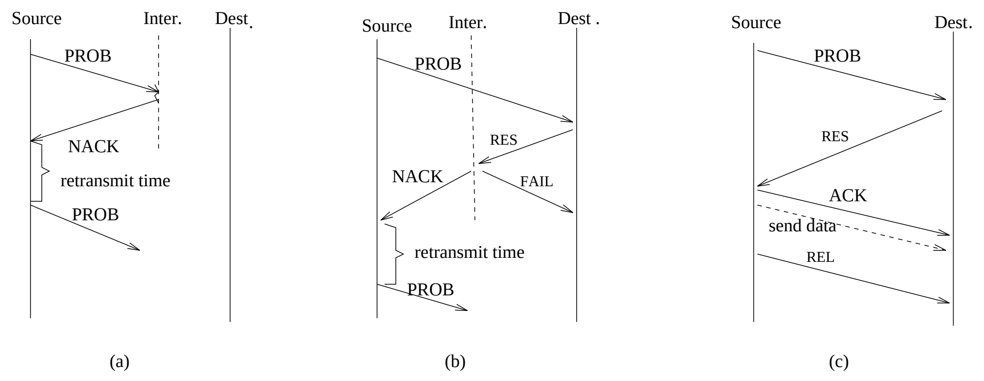

Figure 4: Control messages in backward reservation

## 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. 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 packets* ($PROB$), that travel from sources to destinations 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 packets* ($RES$), similar to the $RES$ packets in the forward scheme, except that they travel from destinations to sources, locking virtual channels as they go through intermediate nodes, and setting the states of the switches accordingly. A $RES$ packet contains a *cset* field.
- *Acknowledgment packets* ($ACK$), similar to $ACK$ packets in the forward scheme except that they travel from sources to destinations. An $ACK$ packet contains a *channel* field.
- *Fail packets* ($FAIL$), to unlock the virtual channels locked by the $RES$ packets in cases of failures to establish connections.
- *Negative acknowledgment packets* ($NACK$), used to inform the source nodes of reservation failures.
- *Release packets* ($REL$), used to release connections after the communication is completed.

Note that a $FAIL/NACK$ message in the forward scheme performs the functions of both a $FAIL$ message and a $NACK$ message in the backward scheme.

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, it determines the next outgoing link, $L_f$, on the forward path to the destination, and updates $PROB.init$ to $PROB.init \cap Avail(L_f)$. 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. Figure 4(a) shows this failed reservation case.

If the resulting $PROB.init$ is not empty, the node forwards $PROB$ on $L_f$ 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 Figure 4(b).

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 non-empty $RES.cset$, 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 Figure 4(c).

**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.

**Aggressiveness**: The aggressiveness of the backward reservation protocols is reflected in the initial size of *cset* chosen by the destination node. The aggressive approach sets $RES.cset$ equal to $PROB.init$, while the conservative approach sets $RES.cset$ to contain a single virtual channel from $PROB.init$. Note that if a protocol supports only the conservative scheme, the $ACK$ messages may be omitted, and thus only five types of messages are needed. As in the forward reservation 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. Although the simulator can simulate both WDM and TDM torus networks, only the results for TDM networks will be presented in this paper. The results for WDM networks follow similar patterns. The simulation uses the following parameters.

- *initial cset size*: This parameter determines the initial size of *cset* in the reservation packet. For FD and FH, the initial *cset* is chosen when the source node composes the RES packet. Assuming that $N$ is the multiplexing degree in the system, an $RES.cset$ of size $s$ is chosen by generating a random number, $m$, in the range [0,N$-$1], and assigning $RES.cset = \{m \ mod \ N, m + 1 \ mod \ N ..., N + s - 1 \ modN\}$. In the backward schemes, the initial *cset* is set when the destination node composes the $ACK$ packet. An $ACK.cset$ of size $s$ is generated in the following manner. If the available set, $RES.INIT$, has less available channels than $s$, the $RES.INIT$ is copied to $ACK.cset$. Otherwise, the available channels are represented in a linear array and the method used in generating the *cset* in the forward schemes is used.

- *timeout value*: This value determines how long a reservation packet can be put in a waiting queue. 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 $MRT - 1$.

- *system size*: This specifies the size of the network. All our simulations are done on torus topology.

- *multiplexing degree*. This specifies the number of virtual channels supported by each link. In our simulation, the multiplexing degree ranges from 1 to 32.

- *message size*: This directly affects the time that a connection is kept before it is released. In our simulations, fixed size messages are assumed.

- *request generation rate at each node (r)*: This specifies the traffic on the network. The connection requests at each node is assumed to have a Poisson inter-arrival distribution. When a request is generated at a node, the destination of the request is generated randomly. When a generated request is blocked, it is put into a queue, waiting to be re-transmitted.

- *control packet processing and propagation time*: This specifies the speed of the control networks. The control packet processing time is the time for an intermediate node to process a control packet. The 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 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, and thus signal propagation time is very small (1 foot per nsec) compared to the length of a message. Finally, deterministic XY–routing is assumed in the torus topology.

Figure 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 $16 \times 16$ torus. The multiplexing degree is taken to be 32, the message size is assumed to be 8 packets and the control packets processing and propagation time is assumed to be 2 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. Note that the use of the optimal *cset* size reduces the delay in addition to increasing the throughput. Note also that the network saturates when the generation rate is between 0.006 and 0.018, depending on the protocol used.

Figure 5 also 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 two sets of experiments. The first set evaluates the effect of the protocol parameters on the network throughput and delay, and the second set evaluates the impact of system parameters on performance.

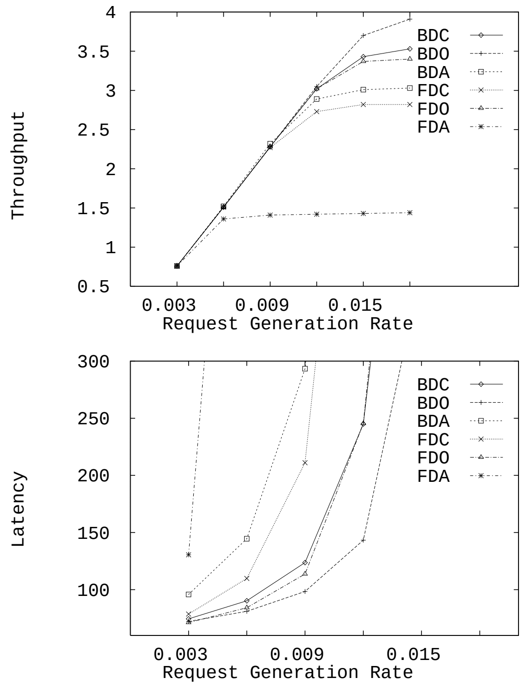

Figure 5: Comparison of reservations with dropping

## 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 = $16 \times 16$, message size = 8 packets, control packet processing and propagation time = 2 time units.

Figure 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. The effect of holding/dropping will be considered in a later figure. Figure 7 shows the results for the backward schemes with dropping.

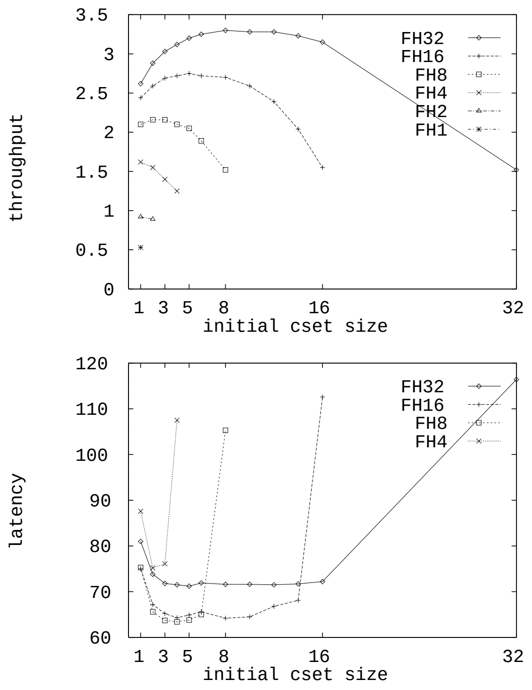

Figure 6: Effect of the *cset* size on forward schemes

From Figure 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. Figure 6 shows that these two extreme protocols do not achieve the smallest latency either. The same observation applies to the backward schemes in Figure 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 non-optimal *cset* size may reduce the throughput by 50% in the forward scheme and only by 25% in the backward scheme. In general, the optimal initial *cset* size is hard to find. A rule of thumb arrived at experimentally to approximate the optimal *cset* size is to use 1/3 and 1/10 of the multiplexing degree for forward schemes and backward schemes, respectively.

Figure 8 shows the effect of the holding time on the performance of the protocols for a multiplexing degree of 32. As shown in Figure 8, the holding time has little effect on the maximum throughput. It slightly increases the performance for the FA (forward aggressive) and BA (backward aggressive) schemes. As for the average latency at light working load, the holding time also has little effect except for the FA scheme, where the latency time decreases by about 20% when the holding time at each intermediate node increases from 0 to 30 time units. Since holding requires extra hardware support compared to dropping, we conclude that holding is not cost–effective for the reservation protocols. In the rest of the paper, we will only consider protocols with dropping policies.

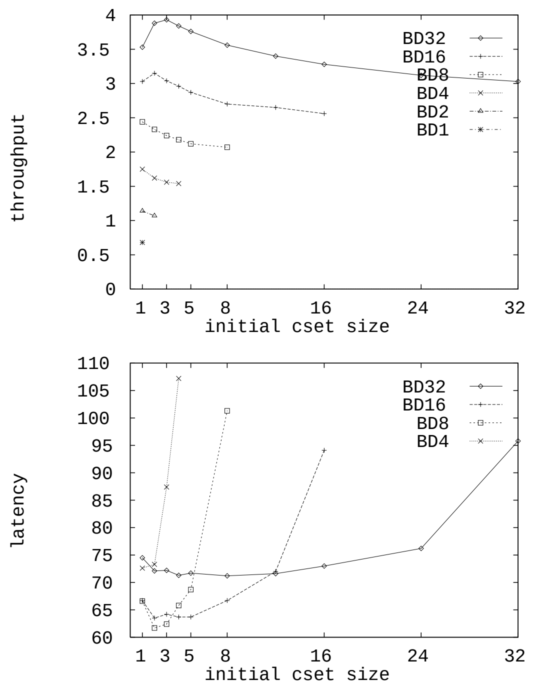

Figure 7: Effect of the *cset* size on backward schemes

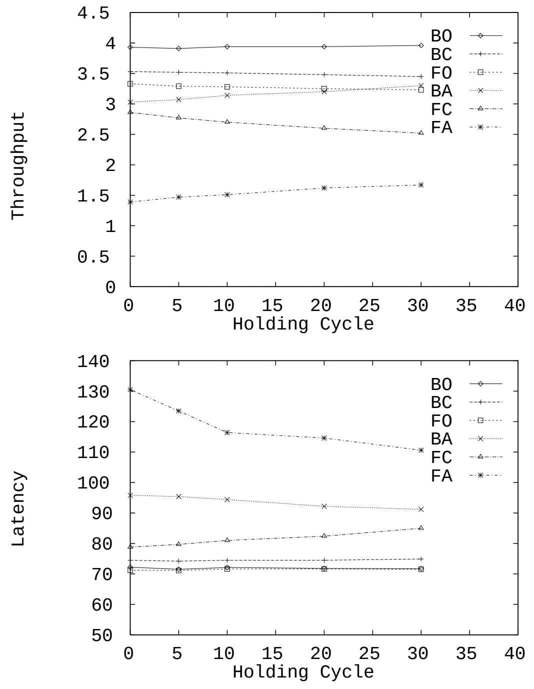

Figure 8: Effect of holding time

In other experiments, we have studied the effect of the maximum retransmit time (MRT) on the performance. We have found that increasing MRT results in a slight performance degradation in all the schemes except FA, in which the performance improves with the MRT. 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 may be 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.

The results of the above set of experiments may be summarized as follows:

- With proper protocols, multiplexing results in higher maximum throughput. Multiplexed networks are significantly more efficient than non–multiplexed networks.

- Both the most aggressive and the most conservative reservations cannot achieve optimal performance. However, the performance of the forward schemes is more sensitive to the initial *cset* size than the performance of the backward schemes.

- The value of the holding time in the holding schemes does not have significant impact on the performance. In general, however, dropping is more efficient than holding.

- The retransmit time has little impact on all the schemes except the FA scheme.

In the next section, we will only consider dropping schemes with MRT equal to 5 time units for all schemes except FA, whose MRT is set to 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. Only one parameter is changed in each experiment, with the other parameters set to the following default values (unless stated otherwise): network size = $16 \times 16$ torus, multiplexing degree = 16, message size = 8 packets, control packet processing and propagation time = 2 time units.

Figure 9 shows the performance of the protocols for different multiplexing degrees. When the multiplexing degree is small, BO and FO have the same maximum bandwidth as BC and FC, respectively. When the multiplexing degree is large, BO and FO offers better throughput. In addition, for all multiplexing degrees, BO is the best among all the schemes. As for the average latency, both FA and BA have significantly larger latency than all other schemes. Also, FO and BO 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.

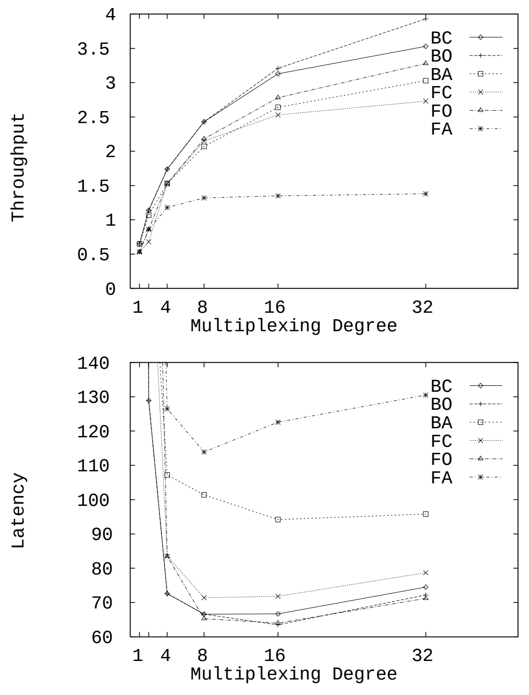

Figure 9: Effect of the multiplexing degree

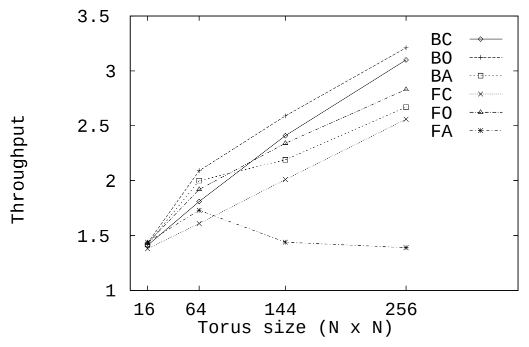

Figure 10: Effect of the network size

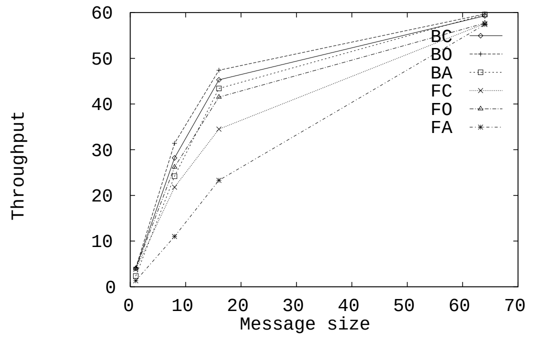

Figure 11: Effect of the message size

Figure 10 shows the effect of the network size on the performance of the protocols. 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.

Figure 11 shows the effect of the message size on the protocols. The throughput is normalized to reflect the number of packets that pass through the network, rather than the number of 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, the BO scheme achieves higher throughput than the other schemes. This indicates that BO incurs less overhead in the path reservation than the other schemes.

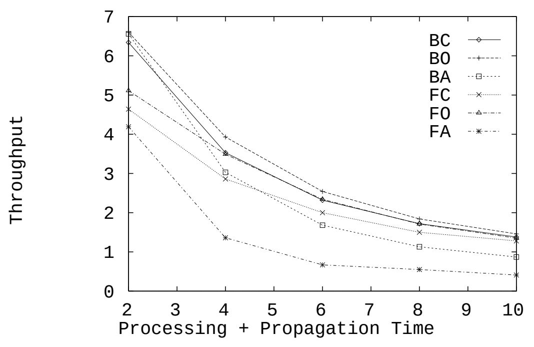

Figure 12: Effect of the speed of the control network

Figure 12 shows the effect of the control network speed on performance. The multiplexing degree in this experiment is 32. 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.

The results of the above set of experiments may be summarized as follows:

- The performance of FA is significantly worse than other protocols. Moreover, this protocol cannot take advantage of both larger multiplexing degree and larger network size.
- The backward reservation schemes provide better performance than the forward reservation schemes for all multiplexing degrees.
- The backward schemes provide better performance when the message size is small and when the network size is large. When the message size is large or the network size is small, all the protocols have similar performance.
- The speed of the control network affects the performance of the protocols greatly.

## 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 in torus networks. 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 100% in the forward reservation schemes and 25% 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 non–multiplexed 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 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 turns 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. In the simulations presented in this paper, we did not consider the effect of such a mismatch between memory speed and optical bandwidth. This effect is being currently studied and will be presented in another forum.

## References

[1] H.R. As, "Media Access Techniques: the Evolution towards Terabit/s LANs and MANs." *Computer Networks and ISDN Systems*, 26(1994) 603–656.

[2] A.S. Acampora and M.J. Karol, "An Overview of Lightwave Packet Network." *IEEE Network Mag.* 3(1), pages 29-41, 1989.

[3] R.A. Barry and P.A. Humblet. "Models of Blocking Probability in All–optical Networks with and without Wavelength Changers." In *Proceeding of IEEE Infocom*, pages 402-412, April 1995.

[4] 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*, 26(6-8), pages 841-858, 1994.

[5] C. A. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," *IEEE Journal on Selected Areas of Communications*, Vol. 8, pp. 948-964, Aug. 1990.

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

[7] A. Ganz and Y. Gao, "A Time-Wavelength assignment algorithm for WDM Start Networks", *Proc. of IEEE INFOCOM*, 1992.

[8] H. Scott Hinton, "Photonic Switching Using Directional Couplers", *IEEE Communication Magazine*, Vol 25, no 5, pp 16-26, 1987.

[9] R. Melhem, "Time–Multiplexing Optical Interconnection Networks; Why Does it Pay Off?" In *Proceedings of the 1995 ICPP workshop on Challenges for Parallel Processing*, pages 30–35, 1995.

[10] S. Nugent, "The iPSC/2 direct–connect communications technology." In *Proceedings of the 3rd conference on Hypercube Concurrent Computers and Application*, Volume 1, Jan. 1988.

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

[12] C. Qiao and R. Melhem, "Reconfiguration with Time Division Multiplexed MIN's for Multiprocessor Communications." *IEEE Trans. on Parallel and Distributed Systems*, Vol. 5, No. 4, April 1994.

[13] C. Qiao and R. Melhem, "Reducing Communication Latency with Path Multiplexing in Optically Interconnected Multiprocessor Systems", *Proc. of HPCA-1*, 1995.

[14] C. Qiao and Y.Mei, "Wavelength Reservation Under Distributed Control." to *Proc. of IEEE/LEOS summer topical meeting: Broadband Optical Networks*, August 1996.

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