Adaptive and Global Routing for Multiplexed Optical Communication on Mesh and Torus Networks *

Chihming Chang and Rami Melhem
Department of Computer Science
The University of Pittsburgh
Pittsburgh, PA 15260.
{achang,melhem}@cs.pitt.edu
(412) 624-8426

Corresponding author:
Rami Melhem
219 MIB (Department of Computer Science)
The University of Pittsburgh,
Pittsburgh, PA 15260.
melhem@cs.pitt.edu
(412)-624-8426.

Abstract

Optical interconnection networks have the potential for achieving low latency and high throughput. Moreover, the high bandwidth of an optical link can be shared by many virtual channels through the use of either multiple wavelengths or multiple time slots. In this paper, routing algorithms are studied for mesh-like architectures with path-multiplexed or link-multiplexed communications. Deterministic, adaptive and global routing algorithms are evaluated and compared in terms of the multiplexing degree needed to support random communication requests.

*This work is supported in part by NSF award MIP-9633729

1 Introduction

General purpose parallel and distributed computing have come to a point where communication latency is the bottleneck of execution. Communication typically accounts for as little as 5-10% of dynamic referencing activities, but can be responsible for as much as 80-90% of the total execution time of an application running distributively [12]. Communication latency can be reduced by increasing network bandwidth and throughput. However, there is a tradeoff between reducing network latency and increasing implementation cost and complexity.

Optical networks have the potential for both reducing latency and increasing throughput [6, 8]. Specifically, large latencies usually result from the overhead of controlling the network. Given that optical fibers may be used for transmitting terahertz signals, it is reasonable to sacrifice some of this large bandwidth in return for simple and efficient control. Time-division and wavelength division multiplexing techniques increase the number of connections that can be simultaneously established in the network, thus reducing the frequency of control operations, which in turns reduces the network control overhead [10].

In time-division multiplexing (TDM), a number of virtual channels can be established on each communication link by dividing the time domain into fixed time slots and establishing each virtual channel during a time slot in a cyclic manner. In wavelength-division multiplexing (WDM), virtual channels are established on the same communication link by transmitting and receiving optical signals on different wavelengths.

There are two basic approaches that can be used to establish connections in a point-to-point network multiplexed in time or wavelength [1, 7, 3, 9, 15, 14, 17]. One approach is to establish a connection which spans more than one communication link using possibly different channels on each link. This approach is called *link multiplexing* (LM), and requires wavelength conversion or time-slot interchange capabilities at each node in the network. The other approach is to establish a connection which spans more than one communication link using the same channel on each link. This approach is called *Path multiplexing* (PM). It is less flexible than link multiplexing but requires less hardware complexity and smaller delays at each node.

In this paper, we will focus on the multiplexing degrees (MD) required to make torus and mesh networks rearrangeably nonblocking in both LM and PM. For the same sequence of communication requests, the multiplexing degree of LM is expected to be smaller than that of PM, since for PM, the same channel has to be used on all the links along a path. The multiplexing degree, however, depends on the choice of the paths used to establish the connections. That is, depends on the algorithm used to route a connection in the network. Adaptive and global routing will be presented and compared to the simpler "dimension order" routing.

The rest of the paper is organized as follows. Section 2 specifies the problem of routing in multiplexed networks, Section 3 presents dimension routing algorithms for both PM and LM systems. Section 4 introduces an adaptive solution to the routing problem. Section 5 introduces routing algorithms that use global information to reduce the multiplexing degree and Section 6 discussed a reordering of the connection requests which improves the routing. Simulation results for 2-3D Torus and Mesh architectures are presented in Section 7. Finally, the conclusions are summarized in Section 8.

2 Problem Definition

We will consider 2D and 3D meshes and torii architectures in which each link can support many virtual channels through multiplexing. We will study different algorithms for realizing a set of connection requests, CR, in those architectures, where a connection request is defined as a tuple consisting of a source node, s, and a destination node, d. The realization of a request is the establishment of a connection from s to d which can be used to transmit data. The establishment of such a connection involves the determination of a path for the connection, the determination of the channels used on the links along that path, and the execution of control and signaling protocols to reserve the path and the channels for the connection.

The realization of connection requests in multiplexed mesh-like architectures has been studied in [13, 16, 19], where different protocols are presented for reserving and releasing connections. In these studies, however, the path from a source, s, to a destination, d, was assumed to follow the *dimension order* routing, which is a simple algorithm often used for circuit switching, packet switching and wormhole switching [5, 11, 18]. The main disadvantage of the dimension order routing is that it defines deterministic routes that are independent of the network loading conditions. In order to overcome this shortcoming, adaptive algorithms have been defined for packet and wormhole routing to take network loads into consideration during path determination [11].

The main purpose of this paper is to introduce load-sensitive algorithms for determining the paths for establishing connections in multiplexed networks and to compare these algorithms with the dimension order algorithm. Detailed control and signaling schemes needed to establish the connections will not be discussed since protocols similar to the ones used for the dimension order case can be applied [13, 19]. Such schemes can be broadly classified into two classes; centralized and distributed control. A centralized scheme assumes the presence of a central control agent that has a global picture of the entire system, while a distributed control scheme assume that each node has its own control agent which has only a limited knowledge of the entire system. Although centralized schemes are not scalable to large systems, such schemes are very effective when the communication requirements are known in advance, as for instance is the case in systems with compiled communications [2, 4].

The algorithms presented in the next sections will find, for each request in a given set of connection requests, CR, both the path used for that connection and the virtual channels that should be used on that path. For a connection request $(s, d) \in CR$, the path from s to d will be specified by a set

$$path(s, d) = \{ \quad < s, n_1 >, < n_1, n_2 >, < n_2, n_3 > \ldots < n_{m-1}, n_m >, < n_m, d >)\}$$

where $< n_i, n_{i+1} >$ denotes the link between nodes n_i and n_{i+1}. For each link $< n_i, n_{i+1} >$ in $path(s, d)$ the virtual channel used for the connection will be specified by a function VC_{sd}

$$VC_{sd} : path(s, d) \rightarrow \{1, 2, \ldots\}$$

where virtual channels are identified by integers. For LM, a connection (s, d) can use different virtual channels $VC_{sd}(< s, n_1 >), \ldots, VC_{sd}(< n_m, d >)$ on different links, while for PM, a connection (s, d) should use the same virtual channel $VC_{sd}(< s, n_1 >) = \ldots = VC_{sd}(< n_m, d >)$ on all links.

The multiplexing degree, MD, needed to realize CR is the maximum number of virtual channels used on the links of the system. In case of WDM, the MD represents the number of wavelengths needed to realize the required connections, while in case of TDM it represents the number of time slots needed. In the former case, larger MD means more fixed wavelength receivers and transmitters or more tuning capabilities, while in the latter case, larger MD represents longer delays. In either case, it is desirable to use routing algorithms that choose paths for the connections in CR in a way that reduces MD.

In the next sections, we will present different routing algorithms and compare the MD resulting when these algorithms are applied to randomly generated communication requests. All algorithms will establish shortest path connections. In order to be more specific, consider a source nodes, $s = (x_s, y_s)$, and a destination node, $d = (x_d, y_d)$, in a 2D mesh, and define $R_{s,d}$ to be the smallest rectangular region that includes both s and d. That is $R_{s,d}$ includes all the nodes (x, y) such that x is between x_s and x_d and y is between y_s and y_d. The routing algorithms will find a path from s to d within $R_{s,d}$. Note that for a 2D torus, there may be more than one possible smallest rectangular regions for each (s, d) pair if $|x_s - x_d| = \frac{N_x}{2}$ and/or $|y_s - y_d| = \frac{N_y}{2}$, where N_x and N_y are the number of nodes along the X and the Y directions, respectively. In such cases, $R_{s,d}$ is chosen among these regions randomly. Similarly, for 3D meshes and torii, a cubic region $R_{s,d}$ can be defined for each pair (s, d) of nodes.

For each node, $n = (x, y)$, in a 2D region, $R_{s,d}$, let $\bar{n}^x$ be the node in $R_{s,d}$ which is closer to d along the x direction. That is $\bar{n}^x = (x+1, y)$ if $x_d > x$, and $\bar{n}^x = (x-1, y)$ if $x_d < x$. If $x_d = x$, then $\bar{n}^x$ is undefined, which is denoted by $\bar{n}^x = \uparrow$. The definition of $\bar{n}^y$ is similar, and so are the definitions of $\bar{n}^x$, $\bar{n}^y$ and $\bar{n}^z$ in 3D cubic regions.

The description of the algorithms presented in the next three sections will be given for 2D architectures for simplicity and clarity. Extending the algorithms to 3D architectures should be straight forward. The algorithms presented in Sections 3 and 4 can be applied in either static or dynamic environments using either centralized or distributed control. They will be presented in a form that is readily applicable to any of the two control schemes. The algorithm presented in Section 5 chooses paths according to global load information and thus is more suitable to centralized control.

3 The Dimension Order Routing (XY Routing)

Given any source node $s = (x_s, y_s)$ and destination node $d = (x_d, y_d)$ in a 2D region $R_{s,d}$, the XY routing establishes a connection from s along the X-axis until it reaches column y_d, then along the Y-axis to d. Given any set of connection requests, CR, the XY algorithm establishes the connections requested in CR in a sequential manner.

We will first discuss the XY algorithm for link multiplexing, LM. For each link, $< n_1, n_2 >$, the number of virtual channels allocated to connections established on that link is recorded in $alloc(n_1, n_2)$. Hence, a new connection can be established on $< n_1, n_2 >$ using channel $alloc(n_1, n_2)+1$. In the following algorithm, after a node, n, is determined to be on $path(s, d)$, the next node, $next$, on the path is determined according to a fixed rule which is independent of the number of channels already used on link $< n, next >$.

Algorithm XY_LM

- $alloc(n_1, n_2) = 0$, for all the links in the architecture,
- For each request (s, d) in CR do
- $path(s, d) = \emptyset$, the empty set
- $n = s$
- While ($n \neq d$) do
- If $\bar{n}^x \neq \uparrow$ Then $next = \bar{n}^x$ Else $next = \bar{n}^y$
- add $< n, next >$ to $path(s, d)$
- set $VC_{sd}(< n, next >) = alloc(n, next) + 1$
- $alloc(n, next) = alloc(n, next) + 1$
- $n = next$
- MD = the maximum of $alloc(n_1, n_2)$ for all links.

For path multiplexing, the same channel has to be used on all the links forming a connection. This restriction implies that the algorithm may assign a channel, c, on a link to a connection even if a channel $c' < c$ is not allocated to other connections. Hence, the simple integer, $alloc(n_1, n_2)$, that keeps track of the channels allocated on link $< n_1, n_2 >$ should be replaced by a set $ALLOC(n_1, n_2)$. A new connection can be established on $< n_1, n_2 >$ using any channel not in $ALLOC(n_1, n_2)$.

Although in XY routing, the path, $path(s, d)$, from a source, s, to a destination, d, is fixed, a channel which is free (not yet allocated) on all the links of the path should be found. A set, $BLKD$ is used to compute the union of the sets $ALLOC()$ of channels that are not free (blocked) on the links forming $path(s, d)$. The smallest channel number that is free on all the links (the smallest channel not in $BLKD$) is used for the connection.

Algorithm XY_PM

- $ALLOC(n_1, n_2) = \emptyset$ for all the links,
- For each request (s, d) in CR do

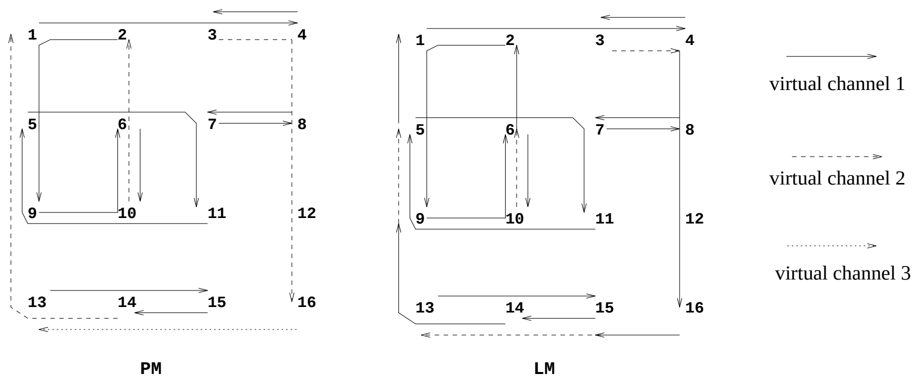


Figure 1: Routing paths for XY_PM and XY_LM on a 4×4 Mesh

- $path(s, d) = \emptyset$ and $BLKD = \emptyset$
- $n = s$
- While ($n \neq d$) do
- If $\bar{n}^x \neq\uparrow$ Then $next = \bar{n}^x$ Else $next = \bar{n}^y$
- add $< n, next >$ to $path(s, d)$
- $BLKD = BLKD \cup ALLOC(n, next)$;
- $n = next$
- let $slot$ be the smallest channel number not in $BLKD$
- For every link, $< n_1, n_2 > \in path(s, d)$ do
- set $VC_{sd}(< n_1, n_2 >) = slot$
- $ALLOC(n_1, n_2) = ALLOC(n_1, n_2) \cup \{slot\}$
- MD = the maximum slot in $ALLOC(n_1, n_2)$ for all links.

For example, consider the following set of communication requests for a 4×4 Mesh:

$$CR = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 4 & 9 & 16 & 3 & 11 & 10 & 8 & 7 & 6 & 2 & 5 & 12 & 15 & 1 & 14 & 13 \end{pmatrix}$$

The top row represents the source nodes and the bottom row represents the destination nodes. For instance, the first request is for a connection from node 1 to node 4. In Figure 1, we show the routing paths for both PM and LM resulting from the XY routing. To illustrate the difference between LM and PM, consider the last request, which is from node 16 to node 13. For path multiplexing, we need to use channel 3 to realize the request because channel 1 has been used on link $< 15, 14 >$ and channel 2 has been used on link $< 14, 13 >$. In other words, the multiplexing degree for PM is 3. However, for link multiplexing, we only need 2 channels since we can use channel 1 on link $< 16, 15 >$ and channel 2 on both links $< 15, 14 >$ and $< 14, 1 >$ along the paths (see Figure 1).

4 Adaptive Linear Routing Algorithms

In the XY routing, after a node is selected to be on the path from s to d, the next node on the path is selected in a predetermined way. In this section, we present a heuristic in which the next node is selected depending on local load information. Specifically, given an intermediate node on a path, the next node on the path is selected to be the one with the least communication load. For multiplexed networks, an obvious measure of the communication load is the number of allocated virtual channels. The routing is greedy and establishes a connection in a time linear to the distance between the source and destination. Thus, it is called *adaptive linear routing.*

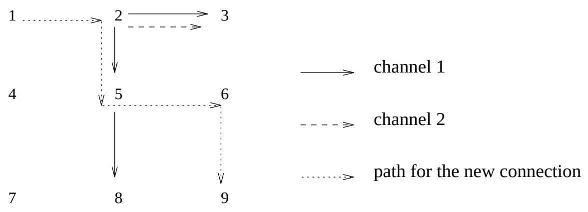


Figure 2: Example of adaptive linear routing for LM

The algorithm for LM is very similar to XY_LM except that at an intermediate node, n, the next node, $next$, is selected to minimize $alloc(n, next)$, with ties broken in favor of XY routing.

Algorithm Adaptive Linear_LM

- $alloc(n_1, n_2) = 0$, for all the links in the architecture,
- For each request (s, d) in CR do
- $path(s, d) = \emptyset$
- $n = s$
- While ($n \neq d$) do
- If $\bar{n}^x \neq \uparrow$ and $(alloc(n, \bar{n}^x) \leq alloc(n, \bar{n}^y))$ Then $next = \bar{n}^x$ Else $next = \bar{n}^y$
- add $< n, next >$ to $path(s, d)$
- $VC_{sd}(< n, next >) = alloc(n, next) + 1$
- $alloc(n, next) = alloc(n, next) + 1$
- $n = next$
- MD = the maximum of $alloc(n_1, n_2)$ for all links.

For example, consider the request (1,9) in Figure 2 and assume that channels 1 and 2 are occupied on link $< 2, 3 >$ and that channel 1 is occupied on links $< 2, 5 >$ and $< 5, 8 >$. That is $alloc(2, 3) = 2$ and $alloc(2, 5) = alloc(5, 8) = 1$. According to Linear_LM, the path from 1 to 9 will first proceed to node 2 (favors XY routing when $alloc(1, 2) = alloc(1, 4) = 0$), then node 5 ($alloc(2, 5) < alloc(2, 3)$), then node 6 ($alloc(5, 6) = 0 < alloc(5, 8)$), and finally to node 9.

The adaptive algorithm for PM uses an idea similar to that of the LM adaptive routing for finding the path from a source to a destination. The next node on a path is selected such that the number of channels that cannot be used to establish the connection is locally minimized. However, as in the case of XY_PM, channels are not assigned to connections as the path is constructed. Rather, information about channel availability is collected in $BLKD$ and the smallest channel not in $BLKD$ is allocated to the connection on all the links in the chosen path.

Algorithm Adaptive Linear_PM

- $ALLOC(n_1, n_2) = \emptyset$ for all the links,
- For each request (s, d) in CR do
- $path(s, d) = \emptyset$ and $BLKD = \emptyset$
- $n = s$
- While ($n \neq d$) do
- If $\bar{n}^x = \uparrow$ then $next = \bar{n}^y$
- Elseif $\bar{n}^y = \uparrow$ then $next = \bar{n}^x$
- Else $BLKD_x = BLKD \cup ALLOC(n, \bar{n}^x)$
- $BLKD_y = BLKD \cup ALLOC(n, \bar{n}^y)$
- If $|BLKD_x| < |BLKD_y|$ then $next = \bar{n}^y$
- Elseif $|BLKD_y| < |BLKD_x|$ then $next = \bar{n}^x$
- Else select $next$ randomly from $\{\bar{n}^x, \bar{n}^y\}$.
- add $< n, next >$ to $path(s, d)$

- $BLKD = BLKD \cup ALLOC(n, next)$;
- $n = next$
- let $slot$ be the smallest channel number not in $BLKD$
- For every link, $< n_1, n_2 >$ in $path(s, d)$ do
- set $VC_{sd}(< n_1, n_2 >) = slot$
- $ALLOC(n_1, n_2) = ALLOC(n_1, n_2) \cup \{slot\}$
- MD = the maximum slot in $ALLOC(n_1, n_2)$ for all links.

Consider the same example shown in Figure 2 where $ALLOC(2, 3) = \{1, 2\}$, $ALLOC(2, 5) = ALLOC(5, 6)$ $= \{1\}$ and $ALLOC$ for all other links is empty. When establishing a connection from 1 to 9, algorithm Linear_PM will chose the same path shown in the figure and will have $BLKD = \{1\}$, which means that channel 1 cannot be used for that connection. Thus channel 2 is used. Note that this is different from the channel assignment resulting from Linear_LM, where the connection from 1 to 9 will be established using channel 1 on link $< 1, 2 >$, channel 2 on link $< 2, 5 >$, and channel 1 on links $< 5, 6 >$ and $< 6, 9 >$.

5 Global Quadratic Routing Algorithms

In this section, we present heuristic routing algorithms that take advantage of global load information to establish a connection. The algorithm uses dynamic programming to find the least loaded path from a source to a destination. It is quadratic in the distance between the source and destination, and thus is called "Global Quadratic".

As in the previous algorithms, the information about channel utilization will be stored in the arrays $alloc$ and $ALLOC$, for LM and PM, respectively. The selection of a path from a source, s, to a destination, d, however, will not be made incrementally at intermediate nodes in R_{sd}. Instead, the load on all possible shortest paths from s to d will be used to select $path(s, d)$. For this purpose, a set $BLKD(n)$ will be defined for each node n in R_{sd} to contain the channels that cannot be used to connect the source node s to n using any shortest path from s to n.

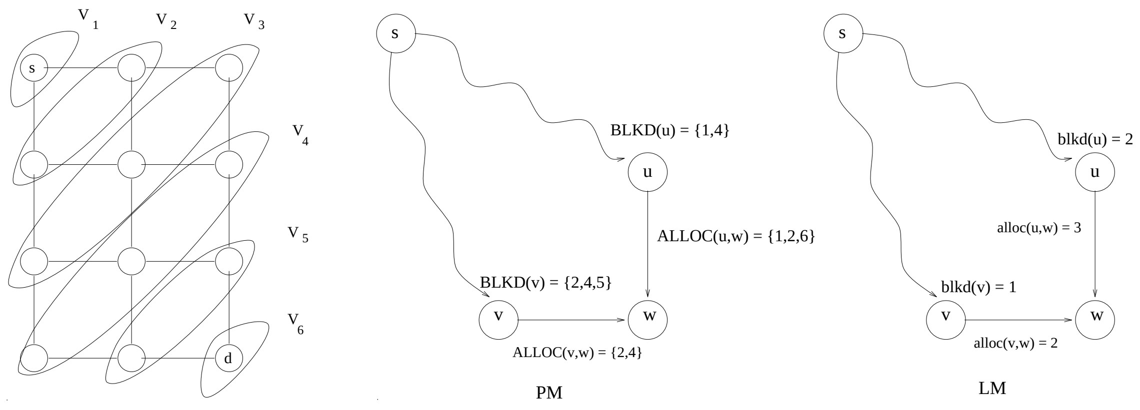
In order to use dynamic programming to compute $BLKD(n)$ for the nodes in a region, $R_{s,d}$, that region is viewed as a layered graph in which the vertices are partitioned into $k \geq 2$ disjoint sets V_i, $1 \leq i \leq k$ such that: $V_1 = \{s\}$ and $V_{i+1} = \{\bar{n}^x, \bar{n}^y | n \in V_i\}$, for $i = 1, 2, \ldots$. Figure 3(a) shows a 3×4 region and its corresponding 6-layer graph. Dynamic programming will be used to compute $BLKD(n)$ for the nodes in level $i + 1$ from $BLKD(n)$ of the nodes in level i. Finally, $BLKD(d)$ will contain the time slots that cannot be used to establish the connection from s to d and thus, any time slot not in $BLKD(d)$ can be used to establish the connection through some path in R_{sd}.

We start by illustrating the technique for PM using Figure 3(b). $BLKD(u) = \{1, 4\}$ means that there are no shortest path from s to u that can use channels 1 or 4. Given that $ALLOC(u, w) = \{1, 2, 6\}$, then any path from s to w that goes through u cannot use any of the channels in $BLKD(u) \cup ALLOC(u, w)$ $= \{1, 2, 4, 6\}$. Similarly, any path from s to w that goes through v cannot use any of the channels in $BLKD(v) \cup ALLOC(v, w) = \{2, 4, 5\}$. Hence, all but channels 2 and 4 can be used to connect s to w. That is

$$BLKD(w) = (BLKD(u) \cup ALLOC(u, w)) \cap (BLKD(v) \cup ALLOC(v, w)).$$

After finding $BLKD(d)$, and selecting a channel to be used for the connection, the global PM algorithm visits the nodes backward from d to s determining the path and setting the channel used for the connection. This is different from the XY_PM and the Linear_PM where the path was determined first, and then the channel was selected for the connection. Following is the complete algorithm.

Algorithm Global Quadratic_PM
- $ALLOC(n_1, n_2) = \emptyset$ for all the links,
- For each request (s, d) in CR do



(a) A 6-layer graph

(b) Computation of $BLKD$ and $blkd$

Figure 3: The dynamic programming algorithm

- $path(s,d) = \emptyset$ and $BLKD(n) = \emptyset$ for all $n \in R_{s,d}$
- For each level V_i, $i = 2, \ldots, k$ Do
- For each node $n \in V_i$ Do
- Let G_n be the set of nodes in V_{i-1} that are neighbors to n,
- $BLKD(n) = \bigcap_{u \in G_n} \{BLKD(u) \cup ALLOC(u,n)\}$
- let $slot$ be the smallest time slot not in $BLKD(d)$
- $n = d$
- While $n \neq s$ do
- let $prev$ be a node in G_n such that $slot$ is not in $BLKD(prev) \cup ALLOC(prev, n)$
- add $< prev, n >$ to $path(s,d)$
- set $VC_{sd}(< prev, n >) = slot$
- $ALLOC(prev, n) = ALLOC(prev, n) \cup \{slot\}$
- $n = prev$
- MD = the maximum slot in $ALLOC(n_1, n_2)$ for all links.

For LM, contiguous channels can be allocated on each link, and thus $BLKD(n)$ can be replaced by an integer, $blkd(n)$, which contains the largest allocated channel. The union and intersection operations can then be replaced by the maximum and minimum operations. For example, consider Figure 3(b), where $blkd(u) = 2$ means that s cannot be connected to u using only channels 1 and 2. If $alloc(u,w) = 3$, then s cannot be connected to w through u using only channels 1, 2 and 3, as expressed by $max\{blkd(u), alloc(u,w)\} = 3$. Similarly s cannot be connected to w through v using only channels 1 and 2. Hence, s cannot be connected to w using only channels 1 and 2 as expressed by:

$$min\{max\{blkd(u), alloc(u,w)\}, max\{blkd(v), alloc(v,w)\}\}.$$

The complete global algorithm for LM is given next.

Algorithm Global Quadratic_LM

- $ALLOC(n_1, n_2) = \emptyset$ for all the links,
- For each request (s,d) in CR do
- $path(s,d) = \emptyset$ and $blkd(n) = 0$ for all $n \in R_{s,d}$
- For each level V_i, $i = 2, \ldots, k$ Do
- For each node $n \in V_i$ Do
- Let G_n be the set of nodes in V_i that are neighbors to n,
- $blkd(n) = min_{u \in G_n} max\{blkd(u), alloc(u,n)\}\}$

- $slot = blkd(d) + 1$
- $n = d$
- While $n \neq s$ do
- let $prev$ be a node in G_n such that $slot > max\{blkd(prev), alloc(prev, n)\}$
- add $< prev, n >$ to $path(s, d)$
- set $VC_{sd}(< prev, n >) = alloc(prev, n) + 1$
- $alloc(prev, n) = alloc(prev, n) + 1$
- $n = prev$
- MD = maximum of $alloc(n_1, n_2)$ for all links.

6 Reordering Connections

In all the algorithms presented above, the path for a connection from s to d is chosen to lie in a rectangular (or cubic) region, R_{sd}. The number of possible shortest paths from s to d depend on the area of R_{sd}. The larger the area of R_{sd}, the larger the number of shortest paths. If we know all the communication requests that are to be established, that is we know CR, then it is possible to change the order by which the connections are established. Specifically, it is possible to sort the requests in CR according to the area of the region, R_{sd}, that each request (s, d) forms. The rational for this sorting is that for connections with small regions, we have only a few choices for selecting a path, and thus these connections should be established first. The connections with larger regions can be established last since more possible paths are available for these connections to choose from.

For example consider Figure 4 where the two connections (3,4) and (2,1) are to be established using Linear_PM. Clearly $R_{2,1}$ is smaller than $R_{3,4}$, and connection (2,1) can be established using only one possible path, while (3,4) can be established using any one of three different paths. If (3,4) is established first (Figure 4(a)), then two channels are needed since (2,1) does not have alternate path choices. Alternatively, if (2,1) is established first (Figure 4(b)), then (3,4) can find an alternate path that will not require the use of a second channel.

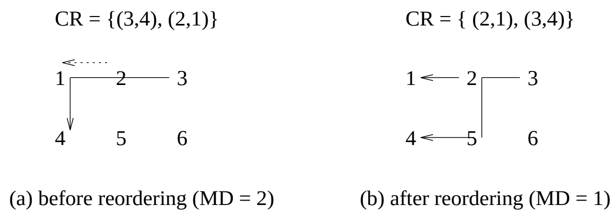


Figure 4: Effect of reordering in PM

It should be noted that, for XY(Z) routing algorithm with LM, the routing paths are predefined and the number of channels used on each link does not depend on the order of connection establishment. Hence, sorting the connections will not have any effect on the MD resulting from XY(Z)_LM. The global quadratic algorithm should benefit most from the sorting since requests with larger R_{sd} can better explore all the possible alternative paths.

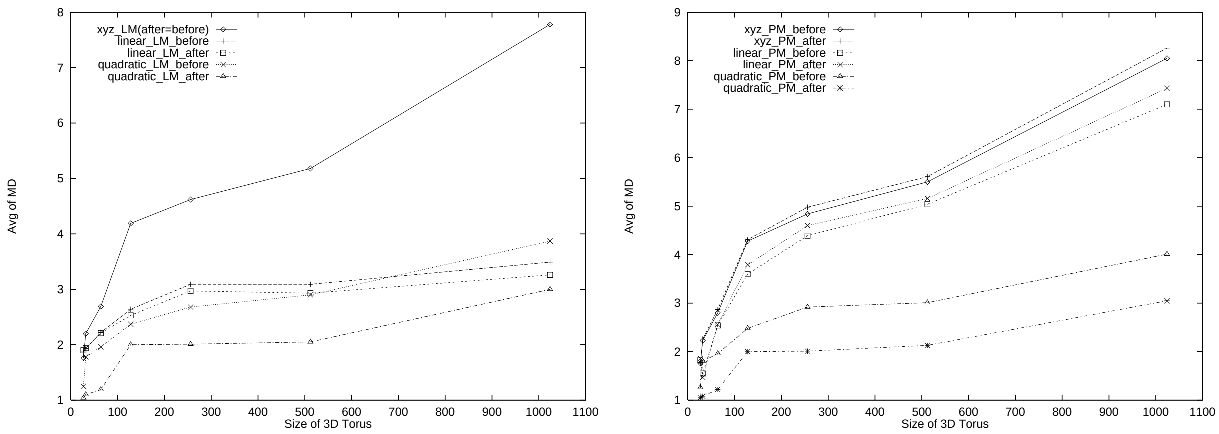
Sorting the requests in CR is not possible if distributed control is used to establish dynamically arriving requests. Noting that the global algorithms are also not suitable when distributed control is used, we can summarize the type of environment for which each routing algorithm is most suitable in the following table.

Environment	static (centralized)	dynamic (distributed)
XY routing	sorted or unsorted	unsorted
Adaptive linear	sorted or unsorted	unsorted
Global quadratic	sorted or unsorted	not suitable

7 Comparison of the Different Algorithms

In this section, we compare the average multiplexing degree, MD, resulting from simulating each routing algorithm on randomly generated connection request sets. We first restrict our attention to request sets that specify permutations among the nodes in 2D and 3D Mesh and Torus architectures. In such cases, each connection request set, CR, contains N requests, where N is the size of the system (the total number of nodes). We then consider request sets that contain arbitrary numbers of randomly generated connections.

We start by considering request sets that represent randomly generated permutations, and we study the effect of reordering (sorting) the connection requests in each CR as explained in Section 6. Figure 5 shows the average multiplexing degree resulting from applying the LM and PM algorithms on 3D Torii architectures with and without connection reordering. For each metwork size, the MD shown is the average over 100 request sets. As described in Section 6, reordering does not have any effect on the XYZ routing algorithm for LM. Moreover, the effect of reordering on the XYZ algorithm for PM and on the adaptive linear algorithm for both LM and PM is minimal. For the global quadratic algorithm, reordering the requests reduces MD by at least 30%.



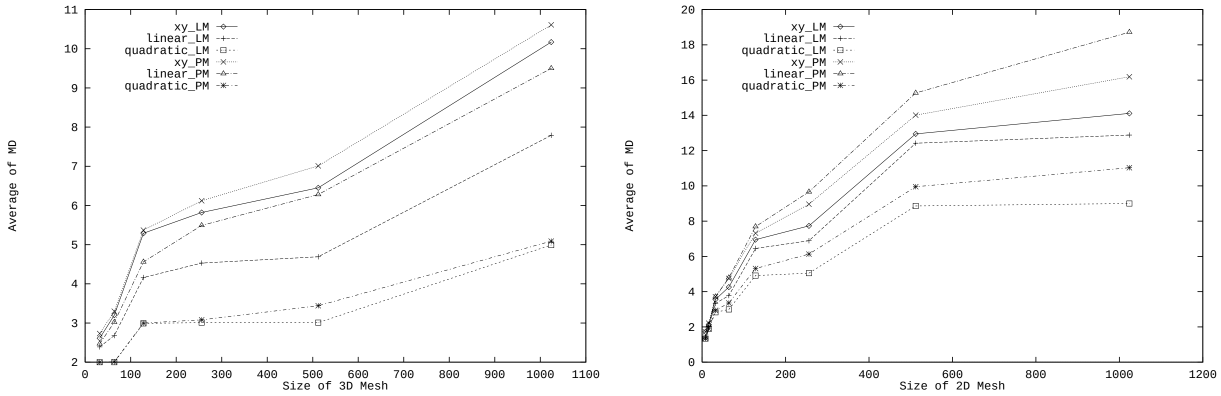
(a) using LM

(b) using PM

Figure 5: Effect of reordering on MD for 3D Torii

Given that reordering is the sorting of the N requests in CR by the area, R_{sd}, of each request (s, d), and that such area can assume one of N possible values, bin sorting can be used to reorder the requests in $O(N)$ time. This complexity is less that the $O(N^2)$ time complexity of the global quadratic algorithm in which each of the N requests in CR is established in $O(N)$ time. In other words, reordering does not increase the complexity of the global quadratic algorithm, and thus, only the results after reordering the requests will be presented in the rest of this section.

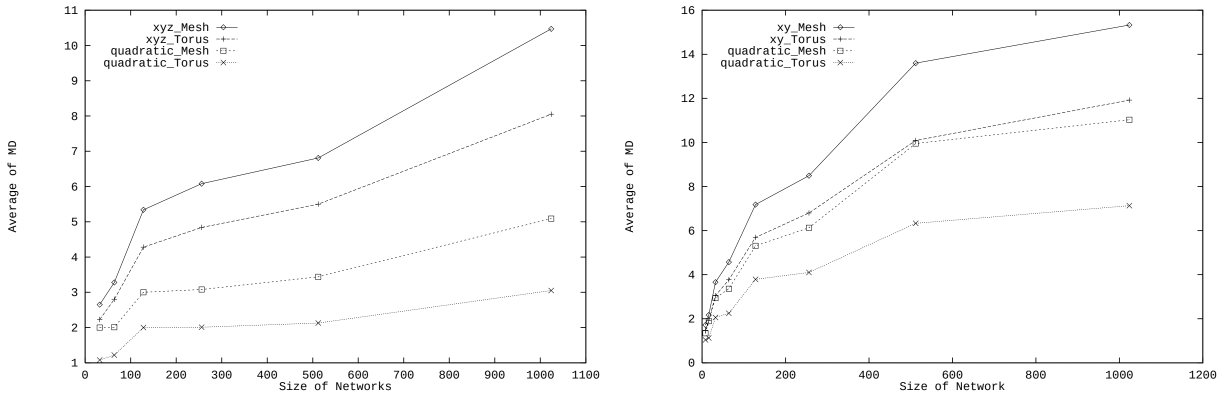
The average MD resulting from applying the six algorithms on 2D and 3D torii and meshes of different sizes are presented in Figures 5 and 6. As expected, the average MD of LM is smaller than that of PM for the same routing algorithm. Also, the global quadratic algorithms produce better results than the adaptive linear algorithms, which, in turn, produce better results than the XY(Z) algorithms. These observations are consistent for all topologies.



(a) 3D mesh

(b) 2D mesh

Figure 6: Average MD of different algorithms for Meshes



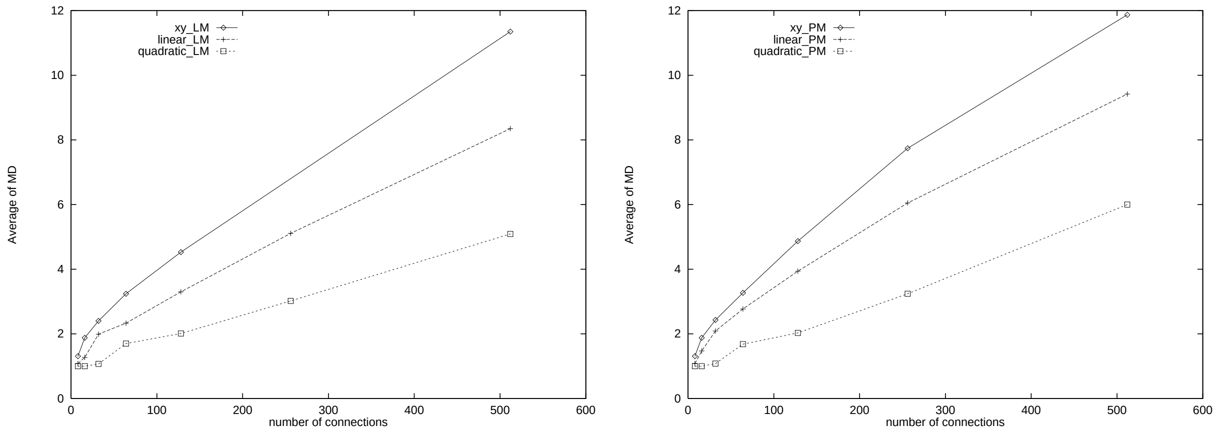
(a) 3D architectures

(b) 2D architectures

Figure 7: Effects of the wrap-around connections when PM is used

Figure 7 shows the improvement that the global algorithm realize over the XY(Z) algorithm for PM. The quadratic algorithm decreases the multiplexing degree by 30-40% over the XY(Z) algorithm. The figures can also be used to compare meshes with torii architecture to estimate the effect of adding the wrap-around connections to meshes. Those connections are shown to drastically reduce the multiplexing degree.

So far, we have shown results for request sets that are random permutations. In practical applications, a node may communicate with more than one other node while some nodes may not communicate at all. Figure 8 shows the results of applying the routing algorithms to establish non-permutation connection request sets in 64-node 3D torus architectures. The communication requests in each set are generated randomly and the number of requests in each set is changed in the range from 32 to 512. To be concise, we only show the results for $4 \times 4 \times 4$ torii. However, the trends observed are consistent for all other sizes and topologies. They show that the same type of results obtained for permutation sets apply to non-permutation sets as well.



(a) for LM (b) for PM

Figure 8: Non-permutation results on $4 \times 4 \times 4$ Torii

8 Conclusions

The overhead of controlling all-optical interconnection networks can be greatly reduced by multiplexing. Path Multiplexing and Link Multiplexing are two different approaches for establishing connections in a multiplexed network. Path multiplexing is especially suitable to all-optical connections since it does not require time-slot interchanging (in case of TDM) or wavelength conversion (in case of WDM).

We have described and compared many algorithms for finding the paths and the channels needed to establish a set of connections, CR, in multiplexed torus and mesh networks. The dimension order routing, sometimes called the XY(Z) routing, is the most common technique used in mesh-like networks. For 2D and 3D architectures, the time complexity of the XY(Z) algorithms are $O(N^{\frac{1}{2}}M)$, and $O(N^{\frac{1}{3}}M)$, respectively, where N is the size of the network and M is the number of requests in CR. Algorithm that adaptively determine the paths for connections depending on local load conditions have the same time complexity of the XY(Z) algorithms but result in lower multiplexing degrees. Finally, global algorithms that run in time $O(NM)$ give the least multiplexing degrees.

A scheme for reordering the requests in CR according to the number of different possible paths that can be used to establish the connections is presented. It is shown to be effective in reducing the multiplexing degree, especially when the global algorithms are used. Finally, we note that all algorithms only generate shortest paths. It is possible sometimes to reduce the multiplexing degree further by exploring the establishment of connections using longer paths. We have not explored this alternative in this paper.

References

[1] R. Barry and P. Humblet. Models of blocking probability in all-optical networks with and without wavelength changers. In *Proc. of INFOCOM'95*, pages 402–412, 1995.

[2] F. Cappello and C. Germain. Toward high communication performance through compiled communications on a circuit switched interconnection network. In *proceddings of the Int'l Symp. on High Performance Computer Architecture*, pages 44–53, Jan. 1995.

[3] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: An approach to high bandwidth optical wan's. *IEEE Trans. on Communications*, 40(7), July 1992.

[4] R. Cypher, A. Ho, S. Konstatinidou, and P. Messina. Architectural requirements of parallel scientific applications with explicit communication. In *Proceddings of the Int'l Symp. on Computer Architecture*, pages 2–13, 1993.

[5] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computers.*, pages 36:547–553, May 1987.

[6] M. Feldman, S. Esener, C. Guest, and S. Lee. Comparsion between optical and electrical interconnects based on power and speed considerations. *Applied Optics*, 29:1742–1751, 1990.

[7] D. Hunter and D. Smith. New architectures for optical tdm switching. *IEEE/OSA J. of Lightwave Technology*, 11:495–511, March 1993.

[8] P. Lalwaney, L. Zenou, A. Ganz, and I. Koren. Optical interconnects for multiprocessors: Cost performance analysis. In *Proceddings of the Symp. on Frontiers of Massively Parallel Computation*, pages 278–285, 1992.

[9] K. Lee and V. Li. A wavelength-convertable optical network. *IEEE/OSA J. of Lightwave Technology*, 11(5/6):962–970, 1993.

[10] R. Melhem. Time-multiplexing optical interconnection networks; why does it pay off?. In *Proc. of ICPP Workshop on Challenges for Parallel Processing*, 1995.

[11] L. Ni and P. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computers.*, pages 26(2):62–76, Feb. 1993.

[12] T. Pinkston and M. Flynn. Parallel processor memory reference behavior and its application to interconnect architecture. In *The 5th Distributed Memory Computing Conference Proceddings*, April 1990.

[13] C. Qiao and Y. Mei. Wavelength reservation under distributed control. In *Proc. of IEEE/LEOS summer topical meeting: Broadband Optical Networks*, 1996.

[14] C. Qiao and R. Melhem. Reconfiguration with time-division multiplexed mins for multiprocessor communication. *IEEE Trans. on Parallel and Distributed Systems*, pages 5(4):337–352, 1994.

[15] C. Qiao and R. Melhem. Reducing communication latency with path multiplexing in optically interconnected multiprocessor systems. In *Proceddings of the International Symposium on High Performance Computer Architecture*, pages 34–43, January 1995.

[16] R. Ramaswami and A. Segall. Distributed control for wavelength routed optical networks. In *Proc. of INFOCOM'96*, 1996.

[17] S. Subramanian, M. Azizoglu, and A. Somani. Connectivity and sparse wavelength conversion in wavelength-routing networks. In *Proc. of INFOCOM'96*, 1996.

[18] H. Sullivan and T. Bashkow. A large scale, homogeneous, fully distributed parallel machine. In *Proceddings of the 4th Symposium on Computer Architecture*, pages 105–124, Mar. 1977.

[19] X. Yuan, R. Gupta, and R. Melhem. Distributed control in optical wdm networks. In *Proc. of IEEE Conf. on Military Communications (MILCOM)*, 1996.