Collective Communications on a Reconfigurable Optical Interconnect[1]

Ahmad Afsahi and Nikitas J. Dimopoulos
Department of Electrical and Computer Engineering
University of Victoria
P.O. Box 3055, Victoria, B.C., Canada, V8W 3P6
{aafsahi, nikitas}@ece.uvic.ca

Abstract. *Free-space optical interconnection is used to fashion a reconfigurable network. Latency hiding techniques are used to enhance its performance.For this network, we present and analyze broadcasting/multi-broadcasting algorithms that utilizes latency hiding and reconfiguration. A combined total exchange algorithm has been proposed based on a combination of the direct, and standard exchange algorithms. Also, known algorithms for other collective communication primitives have been adapted to our network.*

Keywords: Massively parallel computers, collective communications, reconfigurable free-space optical interconnects.

1. Introduction

Message-passing multicomputers are composed of a large number of processor/memory modules that communicate with each other by exchanging messages through their point-to-point interconnection networks. As the communication overhead is one of the most important factors affecting their performance, there has been a great deal of interest in the design of the interconnection networks. In this respect, various types of static interconnection networks, such as complete networks, hypercubes, meshes, rings, and tori have been proposed and some of them have been implemented.

We are interested in having a complete interconnection network, where any computing node can communicate with any other node directly. Complete interconnection networks can be modeled by a complete graph K_N, where all N vertices are linked together and the diameter is one. Each vertex has degree N - 1 and the number of edges is $N(N-1)/2$, far too high to be practical when N is large. These limitations prevent implementing complete networks using metal-based interconnections.

Optics is ideally suited for implementing interconnection networks because of its superior characteristics over electronics [17][9], such as higher interconnection density, higher bandwidth, suitability for reconfigurable interconnects, larger number of fan-in and fan-out, lower error rate, freedom from planar constraints (light beams can easily cross each other), immunity from electromagnetic field and ground loops, lower signal crosstalk etc. It is foreseen that future massively parallel machines will employ optical interconnections in an ever increasing manner.

1. This research was supported through grants from the Natural Sciences and Engineering Research Council of Canada.

Research is focusing in two major directions; the first implements point-to-point interconnection networks [14] but utilizing the advantages of the new emerging optical technologies, while the second attempts to design new interconnection networks [21] based on the constraints of optical devices, such as the optical passive star coupler (OPS). These interconnection networks are modeled by hypergraphs.

Along the first direction, we introduce a complete network using a *reconfigurable free-space optical interconnect*.

Free-space optical interconnects use free-space (vacuum, air or glass) for optical signal propagation and include two basic components: one is the optoelectronic device for photon generation or modulation such as VCSELs and SEEDs, while the other component is the optical beam router to redirect or distribute optical beams. Optical beam routing in a free-space optical interconnection network often employs external optical elements such as holograms, mirrors, prisms, lenses etc.

The study of parallel algorithms have shown some generic communication primitives that appear very often [13][7]. *Collective communications*, are common basic patterns of inter-processor communication that are frequently used as building blocks in a variety of parallel algorithms. Proper implementation of these basic communication operations on parallel architectures is the key to the efficient execution of the parallel algorithms that use them. These collective communication problems include *broadcasting, multi-broadcasting, scattering, gathering, multinode broadcasting,* and *total exchange.* Excellent surveys on collective communication algorithms for different networks can be found in [10][11].

In this work, we shall present efficient algorithms for some collective communication problems for a reconfigurable optical interconnect. Special emphasis will be given to latency hiding to overcome the impact of the reconfiguration costs. Thus, in section two, we define a complete and reconfigurable optical interconnection network using free-space beam routing (to be called OK_N), and then discuss its communication modeling. In section three, we present and analyze a broadcasting/multi-broadcasting algorithm which utilizes the reconfiguration capabilities of this network. Later on in sections four and five, algorithms for scattering and multinode broadcasting are adapted to our network. Finally, a new algorithm for the total exchange operation, to be called *combined total exchange algorithm*, is proposed in section six.

2. A reconfigurable optical interconnection network

In this section, we define an abstract model for our complete free-space optical interconnection network for massively parallel computers, and discuss its characteristics.

Definition A reconfigurable optical network, OK_N, consists of N computing nodes with their own local memory. A node is capable of connecting directly to any other node. These connections are established dynamically by reconfiguring the optical interconnect. The links remain established until they are explicitly destroyed.

A simplified block diagram of the network is shown in Figure 1. Messages are sent using circuit-switching. That is, a connection must be established between the source and destination pair before the message is sent. Each node has the ability to simultaneously send and receive k messages on its k links (the *k-port* model), or exactly one message on one of its links (the *single-port* model). Full-duplex communication where a node can send and receive messages at the same time is supported.

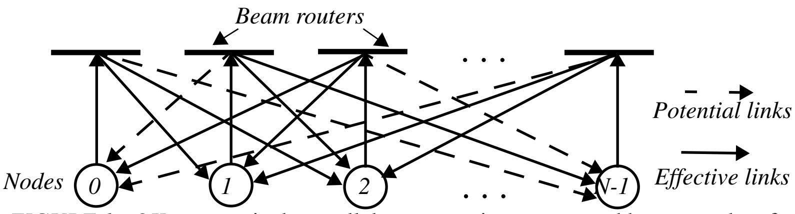

FIGURE 1. OK_N, a massively parallel computer interconnected by a complete free-space optical interconnetion network

Various implementation technologies exist to embody the above abstract model. Such technologies include computer generated holograms [22] and/or deformable mirrors for switching [23], frequency hoping for coding, wavelength tuning for transceivers [2], VCSELs [8] and/or SEEDs [12] for photon generation or modulation. In this paper, we are particularly interested in the abstract model of our proposed network and do not address these implementation issues. We shall assume that one or more of the technologies outlined above will be used to implement the proposed interconnect. Under such an implementation, the various overheads associated with the reconfiguration of the network (such as tuning the receivers, reconfiguring the router, or sending the frequency code in a frequency hoping implementation) are lumped together as the (re)configuration delay d.

An important concern is to model the communication time T required to send a message from one node to another. In the linear model, the communication time depends, among other things, on the length of the message, and it is formulated as $T = t_s + l_m\tau$ where l_m is the length of the message, τ is the per unit transmission time, and t_s is the time required to prepare the message, such as adding a header, a trailer, etc.

For our case, we amend the linear model by explicitly including the reconfiguration delay d (d is an integer) that is necessary for a node to configure a link that would connect directly to its target node. This configuration delay includes, for the case of optical switching, such things as setting up the optical fabric (e.g. beam steering, setting up a computer-generated hologram), the transmission code in the case of frequency hoping etc. The transmission time then becomes $T = d + t_s + l_m\tau$. By incorporating both t_s and $l_m\tau$ into a single message delay $t_m = t_s + l_m\tau$, the message transmission time becomes $T = d + t_m$ For the remaining of the discussion, and without loss of generality, we assume that $t_m = 1$ for a message of unit length. We also assume that $T = d + M$ for a combined message size of M units length which is used in scattering, multinode broadcasting, and total exchange operations.

We shall assume that a node sends a message to another node by first establishing a link to the target (hence the configuration delay d) and then sending the actual message over the established link. It is obvious that if the link is already in place, then the configuration phase does not enter the picture with a commensurate savings in the message transmission time. The main objective of this work is therefore to establish efficient algorithms where the link establishment costs are minimized.

Our model differs from the *postal model* [1] in that we send the message after reconfiguration, while in the postal model a message is sent immediately after the previous one but it encounters a delay before it reaches its destination.

In this paper, we use the notations B_m, MB_m, S_m, G_m, TE_m, to denote both the algorithm and its time complexity for broadcasting, multi-broadcasting, scattering, multinode broadcasting (also called gossiping), and total exchange, respectively. We derive the time complexities of collective communication algorithms in our network under full duplex single-port ($m = F1$) or k-port ($m = Fk$) models.

3. Broadcasting/Multi-broadcasting

Assuming that the (re)configuration delay d is the dominant factor contributing to the message transmission time, we shall concentrate in techniques that could effectively hide it. Such techniques allow a link that is to be used frequently to remain established, or establishing the link prior to its use (under free space propagation, the activation of a link does not impede other ongoing communications).

3.1 Broadcasting

***k-port*:** The easiest algorithm is to let the broadcasting node n_0 inform k new nodes at a step. Clearly, it takes $(d+1)\lceil (N-1)/k \rceil$ time units. In a more efficient algorithm, $B1_{Fk}$, node n_0 sends the message to k other nodes and these k nodes, upon receiving the message, send it to k other nodes each, which are distinct from the nodes that have received the message thus far. Continuing this way, one will terminate the algorithm after $(d+1)(\lceil \log_k(N(k-1)+1) \rceil - 1)$ time units.

Another algorithm, $B2_{Fk}$, allows the nodes that have already been informed, to resend the same message to a different group of nodes. Thus, starting with node n_0, it sends the message to k nodes. At the end of this step, $k + 1$ nodes possess the message which they now send to k nodes each. The algorithm will terminate after $\lceil \log_{k+1} N \rceil$ steps and will require $(d+1)\lceil \log_{k+1} N \rceil$ time units in total.

The algorithms described above are logarithmic in time, but they suffer if one assumes that the reconfiguration delay (d) is large compared to the message transmission delay (t_m). We are interested in devising algorithms that will overcome the existence of the large reconfiguration delay by essentially hiding it. Algorithm $B1_{Fk}$ can be improved if the configuration of all the links forming the tree proceed in parallel. Hence, in this new algorithm, $B3_{Fk}$, the broadcasting message would reach the leaves of the tree in time $d + \lceil \log_k(N(k-1)+1) \rceil - 1$.

However, in a better algorithm $B4_{Fk}$, a node that has sent the message to k other nodes, reconfigures so that it can send the same message to another set of k nodes etc.

This process forms broadcast trees of varying depths. We also assume that the configuration of all the links in the various broadcast trees happen in parallel, as in $B3_{Fk}$. It is understood that while a node is reconfiguring, the message continues to propagate over the previous broadcast tree. Figure 2 depicts the algorithm for a 2-port network with 41 nodes and a reconfiguration delay of 1. As it can be seen, the broadcast tree rooted on node 0 covers only 31 of the total 41 nodes. As soon as node 0 has sent its message to nodes 1 and 2, it starts the reconfiguration process to reach nodes 15 and 16 (and the second broadcast tree) and this is delayed by the (re)configuration delay d=2. While this reconfiguration proceeds, the message continues to propagate over the original broadcast tree. Under these assumptions, a message leaving node n_x will reach $1+k+k^2+\ldots+k^{\lceil d/t_m \rceil} = \frac{k^{\lceil d/t_m \rceil + 1}}{k-1}$ nodes while n_x is reconfiguring.

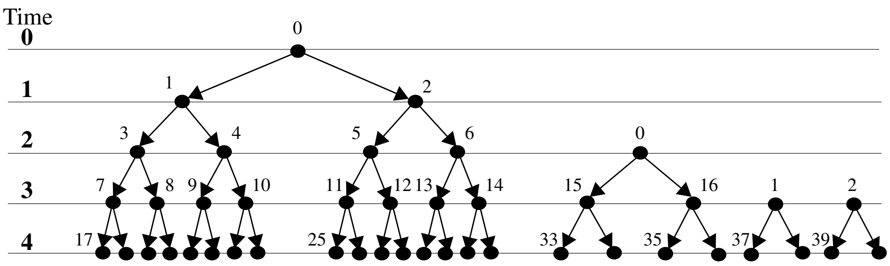

FIGURE 2. Latency hiding broadcasting algorithm for OK_N with 41 nodes, k = 2, d = 1

A configuration phase caches the sets of nodes an arbitrary node needs to connects to. During the configuration phase, the nodes are divided into two groups; the group that has already received the message and the one that has not. A node from the first group will select a set of k nodes from the second, connect to these, and send them the message it already has. Nodes that receive the message are placed in the first group. Every time a node selects a new set of k nodes, it caches the names of the nodes in the previous set together with the iteration number and then it increments the iteration number. The configuration phase terminates when all nodes have received the message. At the end of the configuration phase, each node has cached the sets of nodes it needs to connect to together with their associated iteration numbers. For broadcasting therefore, each node connects to the set of nodes prescribed by the first iteration. As soon as a message to be broadcast reaches a node, it sends it out to the nodes it is already connected to, and then reconfigures to the set of nodes prescribed by the second cached-set etc.

In the example presented in Figure 2, node 0 has cached two sets. The first set comprises nodes 1 and 2, while the second set nodes 15 and 16. Similarly nodes 1 and 2 have two sets cached, while all other nodes have only one set indicating that they need not reconfigure. Thus node 0 will pre-establish the links to nodes 1 and 2 and will wait until it needs to broadcast a message. As soon as a broadcast commences, node 0 will forward the message to nodes 1 and 2, reconfigure to link to nodes 15 and

16, send the same message to these two nodes, and finally reconnect to nodes 1 and 2 waiting for the next broadcast message.

3.1.1 Analysis of the broadcast algorithm

Before presenting the analysis of our broadcast algorithm, we note that the total number of nodes, $N(S)$, informed up to step S follows the recurrence:

for $S \le d+1 : N(S) = kN(S-1)+1$,

for $S > d+1 :\ N(S) = kN(S-1)+N(S-d-1)$, and

$N(0) = 1$.

However, this recurrence cannot be solved for a general d. Therefore, in order to find the time required for the broadcast algorithm to complete, we shall find the number of nodes that will be informed as time progresses, and we shall stop when all nodes N have been informed.

Denote by S the termination time (in units of t_m). Then starting from an arbitrary node n_0, the nodes that will be informed, assuming no reconfiguration, belong to a k-ary tree of depth S rooted at node n_0. There are $N_1 = \frac{k^{S+1}-1}{k-1}$ nodes in this tree, and we shall reference them as belonging to the first generation. Each of the nodes in this tree, once it has broadcast the message to its own children, will reconfigure and will become the root of a new tree over which a new wave of broadcasting will commence and proceed concurrently with the broadcasting in the first generation tree. This can only happen if $S \ge d+2$ ensuring that the first node to be reconfigured (node n_0) will have enough time to reconfigure and broadcast to its k children.

We shall refer to the nodes belonging to the trees rooted at nodes which were included in a generation i tree and reconfigured, as the generation (i+1) nodes. Thus, node n_0 can send its message again at time $d + 1$ after its router has been reconfigured to connect to a set of k new nodes. By sending this new message, n_0 actually embeds a new k-ary tree at depth $d + 1$. The next k nodes at depth 1 of the first generation of trees embed k different k-ary trees at depth $d + 2$. Using this concept, the k^{S-d-2} nodes at depth S-d-2 of the first generation embed the last k^{S-d-2} different trees at depth S - 1 in the second generation. Figure 3 depicts the embedding of the first two generations.

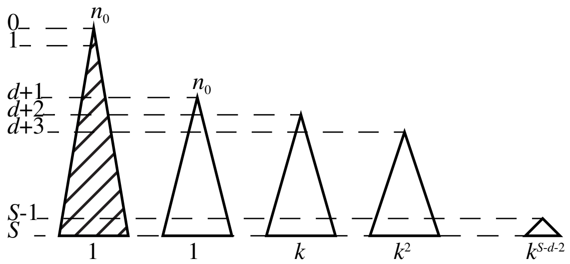

FIGURE 3. First and second generation trees. The numbers underneath each tree denote the number of trees having the same height. These trees are rooted at nodes that were at the same level in the first generation tree.

Denote by N_2 the total number of new nodes in the second generation, and by M_i the total number of new nodes in the trees of the second generation rooted at depth i. Therefore, $M_{d+1} = k + k^2 + \ldots + k^{S-(d+1)} = \frac{k^{S-d} - k}{k-1}$

$$M_{d+2} = k(k + k^2 + \ldots + k^{S-(d+2)}) = \frac{k^{S-d} - k^2}{k-1}.$$

This continues until depth S-1 where we have $M_{S-1} = k^{S-d-2}(k) = \frac{k^{S-d} - k^{S-d-1}}{k-1}$. Therefore, the total number of new nodes in the second generation, N_2, will be: $N_2 = \sum_{i=d+1}^{S-1} M_i = \sum_{j=1}^{S-d-1} \frac{k^{S-d} - k^j}{k-1}$.

The process of reconfiguring the optical interconnects continues by the nodes as soon as they have broadcast the message to their children. Each generation of trees embeds a new generation that commences at depth $d + 1$ from its parent generation. It is clear that the total number of generations is $\lceil S/(d+1) \rceil$.

Let us now count the total number of nodes N_3 in the third generation. The first tree of the third generation is embedded at depth $2(d+1)$ by n_0. We begin with those trees of this generation which are embedded by the nodes of the first tree in the second generation. Let Q_i^1 denote the total number of nodes in these trees rooted at depth i.

$$Q^1_{2(d+1)} = k + k^2 + \ldots + k^{S-2(d+1)} = \frac{k^{S-2d-1} - k}{k-1},$$

$$Q^1_{2(d+1)+1} = k(k + k^2 + \ldots + k^{S-2(d+1)-1}) = \frac{k^{S-2d-1} - k^2}{k-1},$$

and this continues until depth S -1 where $Q^1_{S-1} = k^{S-2d-3}(k) = \frac{k^{S-2d-1} - k^{S-2d-2}}{k-1}$.

Now, consider trees embedded in the third generation by the nodes of the next k trees at depth S - d - 2 in the second generation, and let Q_i^2 denote the total number of nodes in these trees rooted at depth i. Therefore,

$$Q^2_{2(d+1)+1} = k(k + k^2 + \ldots + k^{S-2(d+1)-1}) = \frac{k^{S-2d-1} - k^2}{k-1},$$

$$Q^2_{2(d+1)+2} = k(k(k + k^2 + \ldots + k^{S-2(d+1)-2})) = \frac{k^{S-2d-1} - k^3}{k-1}.$$

This continues until depth S - 1 where $Q^2_{S-1} = k(k^{S-2d-4}(k)) = \frac{k^{S-2d-1} - k^{S-2d-2}}{k-1}$.

We continue with the trees embedded in the third generation by the nodes of the next k^2 trees of depth S-d-3 in the second generation, and let Q_i^3 denotes the total number of nodes in these trees rooted at depth i. Therefore,

$$Q^3_{2(d+1)+2} = k^2(k+k^2+\ldots+k^{S-2(d+1)-2}) = \frac{k^{S-2d-1}-k^3}{k-1},$$

and this continues until depth S - 1 where $Q^3_{S-1} = k^2(k^{S-2d-5}(k)) = \frac{k^{S-2d-1}-k^{S-2d-2}}{k-1}$.

The process of generating trees in the third generation continues up to the trees embedded at depth S - 1, by the nodes of the trees in the second generation, rooted at depth S- d- 2. Let Q^{S-2d-2}_{S-1} denotes the total number of nodes in these trees. Therefore, $Q^{S-2d-2}_{S-1} = k^{S-2d-3}(k) = \frac{k^{S-2d-1}-k^{S-2d-2}}{k-1}$.

Now, we are at the stage to sum the number of nodes at each depth in the third generation. Let Q_i denote the total number of nodes of the trees in the third generation rooted at depth i. Therefore,

$$Q_{2(d+1)} = \frac{k^{S-2d-1}-k}{k-1},\ Q_{2(d+1)+1} = 2\frac{k^{S-2d-1}-k^2}{k-1},$$

$$Q_{2(d+1)+2} = 3\frac{k^{S-2d-1}-k^3}{k-1}\quad, \ldots,\ Q_{S-1} = (S-2d-2)\frac{k^{S-2d-1}-k^{S-2d-2}}{k-1}.$$

Hence, the total number of the new nodes in the third generation, N_3, will be:

$$N_3 = \sum_{i=2(d+1)}^{S-1} Q_i = \sum_{j=1}^{S-2d-2} j\left(\frac{k^{S-2d-1}-k^j}{k-1}\right).$$

In a similar manner, we can compute the number of nodes for the subsequent generations (e.g. $N_4 = \sum_{j=1}^{S-3d-3} \frac{j(j+1)}{2!}\left(\frac{k^{S-3d-2}-k^j}{k-1}\right)$). This process implies lemma 1.

Lemma 1 The number of new nodes in generation i + 1, $i \geq 1$ can be found as:

$$N_{i+1} = \sum_{j=1}^{S-i(d+1)} \binom{j+i-2}{i-1}\left(\frac{k^{S-i(d+1)+1}-k^j}{k-1}\right)$$

Proof. We give a combinatorial argument for its validity. Assume a tree belonging to generation i -1 and rooted at depth (i -1)(d + 1). This tree will produce a number of trees belonging to generation i and rooted at depth i(d + 1). The term $\frac{k^{S-i(d+1)+1}-k^j}{k-1}$ represents the number of new nodes in the first tree of generation i rooted at depth i(d + 1). Subsequent trees in this generation, have a decreasing (by one) number of levels, but since they were produced by nodes that are at lower levels in the parent generation, their numbers grow with the power of k. Therefore, the number of nodes within all the trees at each level, remains the same and equal to $\frac{k^{S-i(d+1)+1}-k^j}{k-1}$.

We have just accounted for the number of trees produced by a single tree in a parent generation. There are though more than one trees of identical depth in the parent generation, and the multiplicative term $\binom{j+i-2}{i-1}$ accounts of this number based on the pascal's triangles [4].

■

The total number of nodes in all generations is equal to $N = N_1 + N_2 + \ldots + N_{\lceil \frac{S}{d+1} \rceil - 1}$,

or $$N = \frac{k^{S+1} - 1}{k-1} + \sum_{i=1}^{\lceil S/(d+1) \rceil - 1} \sum_{j=1}^{S-i(d+1)} \binom{j+i-2}{i-1} \frac{k^{S-i(d+1)+1} - k^j}{k-1}. \qquad \text{(EQ 1)}$$

To determine the termination time S one has to solve equation 1 for S. Unfortunately, no analytical solution has been found. However, this equation can be solved numerically. The subsequent tables provide some numerical examples for the broadcasting time, $B4_{Fk}$, of our latency hiding algorithm, for already known algorithms, $B1_{Fk}$ and $B2_{Fk}$, on our network, and the lower bound $\log_{k+1} N$, obtained from $B2_{Fk}$ when $d = 0$, for a particular number of nodes, N, reconfiguration delay, d, and port modeling, k. It is quite clear that the broadcasting time of our algorithm is much less than the known algorithms.

Table I: Broadcasting time, $k = 2, d = 1$

N	$B2_{Fk}$	$B3_{Fk}$	$B4_{Fk}$	$\log_{k+1} N$
99	10	7	5	5
1393	14	11	8	7
114243	22	17	13	11
1607520	28	21	16	14

Table II: Broadcasting time, $k = 4, d = 3$

N	$B2_{Fk}$	$B3_{Fk}$	$B4_{Fk}$	$\log_{k+1} N$
85	12	6	3	3
1369	20	9	5	5
88633	32	12	8	8
1429100	36	14	10	9

***Single-port*:** In this case, a node can only use one of its links. Therefore, instead of k-ary trees, linear arrays are embedded. Hence, using the same concept as in the k-port modeling, the total number of nodes for generations 1, 2, 3 are: $N_1 = S + 1$, $N_2 = \sum_{j=1}^{S-d-1} S - d - j$, $N_3 = \sum_{j=1}^{S-2(d+1)} j(S - 2d - 1 - j)$.

If we continue in a similar manner to the k-port modeling, then the total number of nodes in all generations, N, would be:

$$N = S + 1 + \sum_{i=1}^{\lceil S/(d+1) \rceil - 1} \sum_{j=1}^{S-i(d+1)} \binom{j+i-2}{i-1} (S - i(d+1) + 1 - j). \qquad \text{(EQ 2)}$$

The subsequent table provides some numerical examples for the broadcasting time, $B4_{F1}$, (in terms of t_m) of our latency hiding algorithm, of the spanning binomial algorithm [20], and for the best case $\log_2 N$ when $d = 0$, for a particular number of nodes,

N, and reconfiguration delay, d. It is clear that the broadcasting time of our algorithm is much less than the known algorithm. Also, the same grouping schema as in k-port modeling can be used.

Table III: Broadcasting time, $d = 3$

N	$(d+1)\lceil \log_2 N \rceil$	$B4_{F1}$	$\log_2 N$
69	28	12	7
1252	44	21	11
82629	68	34	17

4. Scattering

The scattering operation, is used to distribute data to the processors of a parallel computer. The easiest algorithm fimplementing scattering, under single-port modeling, is based on the *sequential tree* [18]. In this case, the source node sends its different message to each of the other nodes sequentially. As the source of communication is the same for the whole scattering operation, this node should reconfigure its optical interconnect after each step. Therefore, the scattering time, $S1_{F1}$, is (N-1)(d+1) time units.

The *spanning binomial tree algorithm* [15] used for broadcasting/multicasting operations can also be used for scattering operation. In this algorithm, the number of informed nodes doubles at each step, and each node stores its own message and forwards the rest of the messages it received, if necessary, to its children. This algorithm has a scattering time, $S2_{F1}$, of $(N-1)+d\log_2 N$ (note that we have neglected the data permutation time at each node). It is easy to see that latency hiding technique does not improve this timing cost in our network.

***k-port*:** The sequential tree algorithm can be extended for k-port modeling. That is, at each step the source node sends its k different messages to k other different nodes. Therefore, $S1_{Fk} = (d+1)((N-1)/k)$. Another algorithm is proposed by F. Desprez et al [6] for scattering operation. In this algorithm, the scattering node n_0, sends k messages of length $N/(k+1)$ each, to its k children. Therefore, there are (k + 1) nodes having $N/(k+1)$ different messages. These nodes, at step 2, communicate each with their k children and send one (k + 1)-th of their initial message to each one. This process continues and all nodes are informed after $\log_{k+1} N$ communication steps. Thus the scattering time, $S2_{Fk}$, takes

$$S2_{Fk} = \sum_{i=1}^{\log_{k+1} N} \left(d + \frac{N}{(k+1)^i}\right) = \frac{N-1}{k} + d\log_{k+1} N \text{ time units.}$$

5. Multinode broadcasting

In multinode broadcasting, also called gossiping [10], all nodes send their unique messages to all other nodes.

***k-port*:** A lower bound for the multinode broadcasting time is $(N-1)/k$ since each node must receive $N-1$ different messages and it can only receive at most k messages at a time. A simple algorithm is based on the extension of the direct algorithm for k-port modeling. That is, at step i, node p sends its message to nodes $(p + (i - 1)k + 1)$ *mod* N, $(p + (i - 1)k + 2)$ *mod* N, ..., $(p + ik)$ *mod* N.

This algorithm has a cost of: $G1_{Fk} = (d+1)\left(\frac{N-1}{k}\right)$.

A better algorithm is to let the nodes combine the messages to reduce the effect of reconfiguration delay. We divide the nodes into $N/(k+1)$ groups of $(k + 1)$ nodes each. Nodes are grouped as $(0, 1, ..., k)$, $(k + 1, k + 2, ..., 2(k +1) - 1)$, ..., $(N - (k + 1), N - (k + 1) + 1, ..., N - 1)$. At step 1, all nodes within a group exchange their messages. At the end of this step, each node has $(k + 1)$ messages. At step 2, node p exchanges all its messages with nodes $(p + (k + 1))$ *mod* N, $(p + 2(k +1))$ *mod* N, ..., $(p + k(k + 1))$ *mod* N. At the end of this step, each node has $(k+1)^2$ messages. Let $s = \log_{k+1}N$. This process continues to the step s, where node p exchanges its messages with the nodes $(p + (k+1)^{S-1})$ *mod* N, $(p + 2(k+1)^{S-1})$ *mod* N, ..., $(p + k(k+1)^{S-1})$ *mod* N. It is clear that at each step i of this algorithm, each node sends $(k+1)^{i-1}$ messages to k other nodes. Hence, this algorithm has a multinode broadcasting time of

$$G2_{Fk} = \sum_{i=1}^{\log_{k+1}N} (d+(k+1)^{i-1}) = \frac{N-1}{k} + d\log_{k+1}N .$$

***Single-port*:** This is derived from the *k-port* model for k=1 with a lower bound of the broadcasting time being $N-1$ while $G2_{Fk} = N-1+d\log_2 N$

6. Total exchange

In total exchange, all nodes send their different messages to all other nodes. A naive algorithm for total exchange is to perform a scattering operation N times in sequence. However, better algorithms exist.

Single-port: In a direct algorithm [19], at step i, node p sends the message destined for node $(p + i)$ *mod* N. The cost of this algorithm, $TE1_{F1}$, is $(N - 1)(d + 1)$.

One may also use a standard exchange algorithm for total exchange similar to the ones used in hypercubes, and meshes [3], where during each step, the complete network is recursively divided into halves, and messages are exchanged across new divisions at each step. Nodes combine messages into larger messages to be transmitted as a single unit. Consider this algorithm for an 8-node multicomputer, as shown in Figure 4. There are $N/2$ messages to be sent by each node at any step in this algorithm. We only describe this for node 0. Node 0 sends all its messages for the nodes at the upper half (that is, nodes 4, 5, 6, and 7) to node 4 at step 1. At the same time, it receives the messages for its half from node 4. At the second step, node 0 sends its messages, along with the messages from node 4, destined to nodes 2 and 3, to node 2. At the same time, it receives the messages from nodes 2, and 6 for itself and node 1. At the third step, node 0 sends its message along with the messages from nodes 2, 4, and 6 to node 1. It is clear that at the end of this step all nodes have exchanged all their messages. Thus, this algorithm, $TE2_{F1}$, has a cost of $\left(d+\frac{N}{2}\right)\log_2 N$.

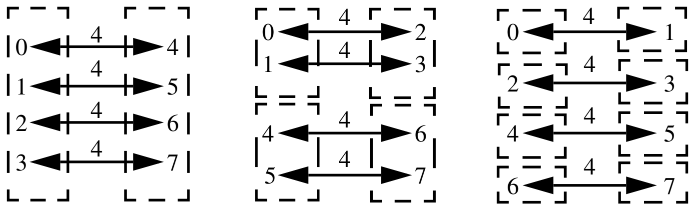

FIGURE 4. Total exchange on an 8-node OKN under single-port modelling

Which algorithm, $TE1_{F1}$ or $TE2_{F1}$, is faster depends on the number of nodes N, and the reconfiguration delay, d. We propose another algorithm, called *combined total exchange algorithm*, $TE3_{F1}$, which is a combination of these two algorithms.

We begin this algorithm by doing some of (or even none of) the steps involved in the standard total exchange algorithm, and then continue with the direct algorithm.

Assuming that we proceed for i steps with the standard total exchange algorithm. After these i steps, there will be 2^i groups of $N/2^i$ nodes each. Each node now will hold messages destined for nodes belonging to the same group which it must exchange using the direct algorithm. Observe that because the preceding standard algorithm, each node has a total $N-1$ messages of which 2^i-1 are destined to the node itself. Thus, each node must send a total of $N-1-(2^i-1) = N-2^i$ messages to the other $N/2^i-1$ nodes in its group, or equivalently $\frac{N-2^i}{N/2^i-1} = 2^i$ messages to each of the nodes. Thus, we can use the direct algorithm in each of the groups but with larger messages, messages of size 2^i. Therefore, the total time for the *combined total exchange* algorithms is computed as the time for the standard exchange part lasting for i steps yielding $i\left(d+\frac{N}{2}\right)$ plus the time for the direct algorithm with messages of size 2^i on groups of $N/2^i$ nodes yielding $(N/2^i-1)(d+2^i)$ for a total time

$$TE3_{F1} = i\left(d+\frac{N}{2}\right)+\left(\frac{N}{2^i}-1\right)(d+2^i).$$

Let us explain this algorithm with $i = 1$ (number of steps doing the standard exchange algorithm) for the example shown in Figure 4. At step 1, the nodes in our complete network are divided in two groups. Each node exchanges 4 messages with its corresponding node at the other half. This takes $d + 4$, and at this point, each of the network halves contain messages destined to the half itself. As a matter of fact, each node now has two messages for each of the nodes in its half. These messages can be distributed to their destinations by using a direct algorithm. There are 4 nodes in each half and 2 messages to be exchanged at a time for a cost of (4 - 1)(d + 2) = 3d + 6. Hence, this algorithm has a total cost of 4d + 10.

It is clear that this algorithm is exactly the same as the direct algorithm when $i = 0$, and the standard exchange algorithm when $i = \log_2 N$.

***k-port*:** The direct algorithm for the k-port modeling requires node p at step i to send its message to nodes $(p + (i - 1)k + 1)$ *mod N*, $(p + (i - 1)k + 2)$ *mod N*, ..., $(p + ik)$ *mod N*. This algorithm has a cost of, $TE1_{Fk} = (d+1)\left(\frac{N-1}{k}\right)$.

The same grouping and algorithm as $G2_{Fk}$ can be used for total exchange with the exception that this time each node sends $N/(k+1)$ messages at a time. Therefore, the cost of this algorithm, $TE2_{Fk} = \sum_{i=1}^{\log_{k+1} N}\left(d+\frac{N}{k+1}\right) = \left(d+\frac{N}{k+1}\right)\log_{k+1} N$. Figure 5 illustrates the above algorithm when N =9 and k = 2.

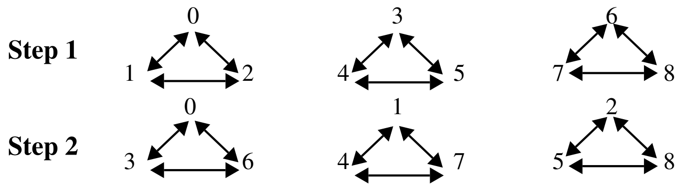

FIGURE 5. Total exchange on a 9 node OK_N under 2-port modelling

Which algorithm, $TE1_{Fk}$ or $TE2_{Fk}$, is faster depends on the number of nodes N, number of input/output channels, k, and the reconfiguration delay, d. Just like the single-port modeling, a *combined total exchange algorithm*, $TE3_{Fk}$, is proposed which is a combination of the above two algorithms with a total exchange time of $TE_{Fk} = i\left(d+\frac{N}{k+1}\right)+\frac{1}{k}\left(\frac{N}{(k+1)^i}-1\right)(d+(k+1)^i)$ and i being the number of steps doing the standard exchange algorithm $TE2_{Fk}$ before switching to the direct algorithm $TE1_{Fk}$.

It is clear that this algorithm is exactly the same as the direct algorithm when $i = 0$, and the standard exchange algorithm when $i = \log_{k+1} N$. We haven't found any mathematical proof that our algorithm is better than the known algorithms. However, in all the numerical examples (more than 150,000) that we have performed for the comparison of these algorithms, we have always found a step, i, for which, our combined total exchange algorithm had a shorter or equal exchange time than both the direct, $(d+1)\left(\frac{N-1}{k}\right)$, and the standard, $\left(d+\frac{N}{k+1}\right)\log_{k+1} N$, exchange algorithms. The above statement is also true for the single-port modeling.

It is a conjecture that the combined total exchange algorithm has a termination time that is earlier than (or at most equal) to either the direct or the standard total exchange algorithms.

Tables 5 and 6 below present some typical examples

Table IV: Total exchange time, $N = 1024$, $k = 1$

d	$TE1_{F1}$	$TE2_{F1}$	$TE3_{F1}$
2	3069	5140	2558 (i = 1, 2)
3	4092	5150	2815 (i = 2)
4	5115	5160	3072 (i = 2, 3)
5	6138	5170	3202 (i = 3)

Table V: Total exchange time, $N = 729$, $k = 2$

d	$TE1_{Fk}$	$TE2_{Fk}$	$TE3_{Fk}$
1	728	1464	728 (i = 0, 1)
2	1092	1470	850 (i = 1)
3	1456	1476	972 (i = 1, 2)
4	1820	1482	1014 (i = 2)

7. Conclusion

Free-space optical interconnects provide attractive alternative ways to achieve high speed communications in a massively parallel computer system. In this paper, we proposed a reconfigurable, complete, free-space optical interconnection network, where we showed how broadcasting, and multi-broadcasting can be done in an efficient way using a latency hiding technique. Our algorithms are much faster the known algorithms.

We also presented a new total exchange algorithm which is a combination of the direct, and standard exchange algorithms. We conjecture that this algorithm has a faster time (or at least equal to) than known algorithms in our network. We also adapted known algorithms for scattering and multinode broadcasting to our network.

We should mention that some of the algorithms in this paper are applicable to massively parallel computers where the number of nodes are a power of 2 or a power of (k + 1) for single-port, and k-port modeling, respectively. We think that finding optimal algorithms for any number of nodes is not an easy task. However, if the number of nodes is not a power of 2 or (k +1), dummy nodes can be assumed to exist until the next power of 2 or (k +1), respectively, with a little performance loss.

In addition to our latency hiding techniques, other techniques such as prefetching [16] can also be used to establish a communication link before it is actually required. Cacheing link information dynamically can also hide the overhead included in link establishment and message start-up.

References

1. A. Bar-Noy and S. Kipnis, "Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems," 4th Annual ACM Symposium on Parallel Algorithms and Architectures, June 1992, pp. 11-22
2. P. Berthome and A. Ferreira, "Communication Issues in Parallel Systems with Optical Interconnections," International Journal of Foundations of Computer Science, 1996, to appear in special issue on interconnection networks

3. S. H. Bokhari and H. Berryman, “Complete Exchange on a Circuit Switched Mesh,” Proceedings of the 1992 Scalable High Performance Computing Conference, Apr. 1992, pp. 300-306

4. P. J. Cameron, Combinatorics: Topics, Techniques, Algorithms, Cambridge University Press, 1994

5. L. J. Camp “Guided-Wave and Free-Space Optical Interconnects for Parallel Processing Systems: a Comparison,” Applied Optics, Vol. 33, No. 26, pp. 6168-6180 (1994)

6. F. Desprez et al., “Efficient Communication Operations on Passive Optical Star Networks,” Proceedings of the First International Conference on Massively Parallel Processing using Optical Interconnections, 1994, pp. 52-58

7. V. Dimakopoulos and N. J. Dimopoulos, "Total Exchange in Cayley Networks," Euro-Par’ 96, Parallel Processing, Lecture Notes in Computer Science, 1996, pp. 341-346

8. L. Fan et al., “Optical Interconnection Networks for Massively Parallel Processors using Beam Steering Vertical Cavity Surface-Emitting Lasers,” Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections, Oct. 1995, pp. 28-34

9. M. R. Feldman et al., “Comparison Between Optical and Electrical Interconnects Based on Power and Speed Considerations,” Appied Optics, 27(9), May 1988, pp. 1742-1751

10. P. Fraigniaud and E. Lazard, "Methods and Problems of Communication in Usual Networks," Discrete Applied Mathematics, Vol. 53, 1994, pp. 79-133

11. S. M. Hedetneimi et al., "A Survey of Gossiping and Broadcasting in Communication Networks," Networks, Vol. 18, 1988, pp. 319-349

12. H. S. Hinton et al., “Free-Space Digital Optical Systems,” Proceedings of IEEE, Vol. 82, No. 11, Nov. 1994, pp. 1632-1649

13. V. Kumar et al., Introduction to Parallel Computing: Design and Analysis of Algorithms, The Benjamin/Cummings Publishing Company, Inc., 1994

14. A. Louri and H. K. Sung, "An Optical Multi-Mesh Hypercube: A Scalable Optical Interconnection Network for Massively Parallel Computing," Journal of Lightwave Technology, Vol. 12, No. 4, 1994, pp. 704-716

15. P. K. McKinley et al., “Unicast-based Multicast Communication in Wormhole-routed Networks,” IEEE Trans. Parallel and Distributed Systems, 5(12): 1252-1265, Dec. 1994

16. T. Mowry and A. Gupta, “Tolerating Latency Through Software-Controlled Prefetching in Shared-Memory Multiprocessors,” Journal of Parallel and Distributed Computing, 12(2), 1991, pp. 87-106

17. R . A. Nordin et al., “A System Perspective on Digital Interconnection Technology,” IEEE Journal of Lightwave Technology, Vol. 10, June 1992, pp. 801-827

18. N. Nupairoj and L. M. Ni, “Benchmarking of Multicast Communication Services,” Technical Report MSU-CPS-ACS-103, Michigan State University, Sept. 1995

19. S. R. Seidel, “Circuit Switched vs. Store-and-Forward Solutions to Symmetric Communication Problems,” Proceedings of the 4th Conference on Hypercube Computers and Concurrent Applications, 1989, pp. 253-255

20. H. Sullivan and T. R. Bashkow, “A Large Scale, Homogeneous, Fully Distributed Parallel Machine,” Proceedings of the 4th Annual Symposium on Computer Architecture, Vol. 5, Mar. 1977, pp. 105-124

21. T. Szymanski, “Hypermeshes: Optical Interconnection Networks for Parallel Computing,” Journal of Parallel and Distributed Computing, 26, 1995, pp. 1-35

22. G. Tricoles, “Computer Generated Holograms: A Historical review,” Applied Optics, Special Issue on Computer Generated Holograms, Vol. 26, No. 20, 1987, pp. 4351-4360

23. T. Yatagai, “Optical Computing and Interconnect, “Proceedings of IEEE, Vol. 84, No. 6, June 1996, pp. 828-852