Appendix

In this section, we introduce some techniques for implementing the beam routing, and collision avoidance in the recievers. Each node has a beam router, such as a computer generated hologram or a deformable mirror, which is capable of redirecting k (or 1 in the single-port model) optical beams concurrently. The reconfiguration (programming the computer generated hologram, or deforming the mirror) takes d time steps (d is an integer) while the actual transmission takes only one step. By current technology, programming a computer generated hologram takes a few microseconds [2].

Each node has a transmitter for sending an optical beam toward the beam router to be directed or distributed to the receivers at the other nodes. Also, each node has a number of receivers at its input ports. Some of these input ports may be used for collective communications while others can be used for other pair-wise communications. The number of receivers can also be degenerated to 1 if, for example, more than one operation (either collective or pair-wise communication) is supported.

Path-setup phase can be done by sending an encoded light beam to the beam router to reprogram the computer generated hologram, or to deform the mirror so that the actual message can be delivered to the destination(s) directly. We propose two alternative ways for path setup phase in the circuit switching between any two nodes. A node that wants to reconfigure its optical interconnect may include the destination address in the header of its message. The router at the computer generated hologram or the deformable mirror upon receiving the message stores it in a buffer and then configures its output links so that it can forward the message to the destination node(s). This approach needs a buffer for the entire message at each beam router which is of high cost. The better approach may be to send a single beam including only the destination address to the beam router for path setup phase. Then, d steps later, the second beam containing the actual message can be sent to the already configured beam router.

Several beams may arrive at a destination node at the same time. If we want to assign only one fixed wavelength to these beams, resembling the address of the receiving node (that is the transmitters are tunable, and the receivers are fixed and tuned to the wavelengths resembling their node address), then there would be a collision at the receiving node. We can make sure that this situation does not happen if we put a constraint on our collective communication algorithms that no two signals can enter a node at the same time. However, if the receivers are tunable, the above limitation can be overcome. There is a tuning overhead ρ_r [2] that it takes for the receiver to tune to a new wavelength. Another time of interest is the stopping time δ [2], which is the time required to stop receiving information on previous tuned wavelength. Other parameters of interest is the number of available wave

lengths Λ, for the receiver to tune to and the code used in case of frequency hoping. Moreover, in case of limited number of wavelengths, one can utilize spread-spectrum techniques where each transmitter sends its information changing the wavelength in a pseudorandom fashion. The receiver can reconstruct the transmitted message if it is aware of the pseudo-random code used for encoding the sequence of wavelengths used during the transmission.

17. V. Kumar et al., Introduction to Parallel Computing: Design and Analysis of Algorithms, The Benjamin/ Cummings Publishing Company, Inc., 1994
18. 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
19. P. K. McKinley and D. F. Robinson, “Collective Communication in Wormhole-Routed Massively Parallel Computers,” IEEE Computer, Dec. 1995, pp. 39-50
20. P. K. McKinley et al., “Unicast-based Multicast Communication in Wormhole-routed Networks,”IEEE Transactions on Parallel and Distributed Systems, 5(12): 1252-1265, Dec. 1994
21. 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
22. R . A. Nordin et al., “A System Perspective on Digital Interconnection Technology,” IEEE Journal of Lightwave Technology, Vol. 10, June 1992, pp. 801-827
23. N. Nupairoj and L. M. Ni, “Benchmarking of Multicast Communication Services,” Technical Report MSU-CPS-ACS-103, Michigan State University, Sept. 1995
24. 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
25. R. Thakur and A. Choudhary, “All-to-all Communication on Meshes with Wormhole Routing,” Proceedings of the 1994 International Parallel Processing Symposium, 1994, pp. 561-565
26. 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
27. T. Szymanski, “Hypermeshes: Optical Interconnection Networks for Parallel Computing,” Journal of Parallel and Distributed Computing, 26, 1995, pp. 1-35
28. G. Tricoles, “Computer Generated Holograms: A Historical review,” Applied Optics, Special Issue on Computer Generated Holograms, Vol. 26, No. 20, 1987, pp. 4351-4360
29. T. Yatagai, “Optical Computing and Interconnect, “Proceedings of IEEE, Vol. 84, No. 6, June 1996, pp. 828-852

We also presented a new total exchange algorithm which is a combination of the direct, and standard exhange 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 modelling, 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 [21] 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. H. Bourdin et al., “A Comparative Study of One-to-Many WDM Lightwave Interconnection Networks for Multiprocessors,” Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections, 1995, pp. 257-263
5. P. J. Cameron, Combinatorics: Topics, Techniques, Algorithms, Cambridge University Press, 1994
6. L. J. Camp, “Guided-Wave and Free-Space Optical Interconnects for Parallel Processing Systems: a Comparison,” Applied Optics, Vol. 33, No. 26, Sept. 10 1994, pp. 6168-6180
7. 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
8. 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
9. 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
10. 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
11. P. Fraigniaud and E. Lazard, "Methods and Problems of Communication in Usual Networks," Discrete Applied Mathematics, Vol. 53, 1994, pp. 79-133
12. S. M. Hedetneimi et al., "A Survey of Gossiping and Broadcasting in Communication Networks," Networks, Vol. 18, 1988, pp. 319-349
13. H. S. Hinton et al., “Free-Space Digital Optical Systems,” Proceedings of IEEE, Special Issue on Optical Computing Systems, Vol. 82, No. 11, Nov. 1994, pp. 1632-1649
14. S. L. Johnsson, “Communication in Network Architectures,” in VLSI and Parallel Computation, ed. R. Suaya and G. Birtwistle, Morgan Kaufmann, 1990
15. S. L. Johnsson and C.-T. Ho, “Optimum Broadcasting and Personalized Communication in Hypercubes,” IEEE Transactions on Computers, Vol. C-38, Sept. 1989, pp. 1249-1268
16. F. E. Kiamilev, “Performance Comparison between Optoelectronic and VLSI Multistage Interconnection Networks,” Journal of Lightwave Technology, Vol. 9, No. 12, Dec. 1991, pp. 1674-1692

The same grouping and algorithm as $G2_{Fk}$ can be used for total exchange with the exception that this time each node sends $\frac{N}{k+1}$ messages at a time. Therefore, the cost of this algorithm, $TE2_{Fk}$, is: $\sum_{i=1}^{\log_{k+1}N}\left(d+\frac{N}{k+1}\right)=\left(d+\frac{N}{k+1}\right)\log_{k+1}N$. Figure 10 illustrates the above algorithm when N =9 and k = 2.

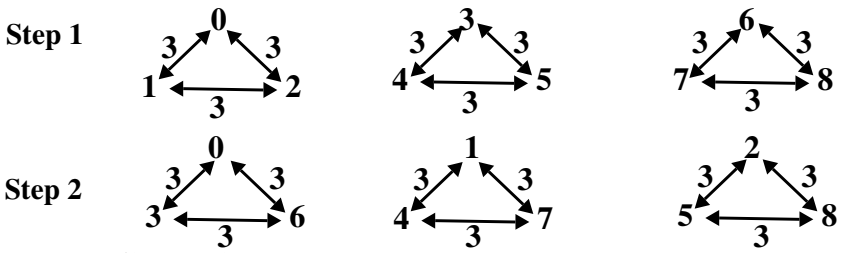


FIGURE 10. Total exchange on a 9 node OK_N under *2*-port modelling

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 modelling, a *combined total exchange algorithm*, $TE3_{Fk}$, is proposed which is a combination of the above two algorithms.

Lemma 3 The *combined total exchange algorithm* under k-port modelling on OK_N takes $i\left(d+\frac{N}{k+1}\right)+\frac{1}{k}\left(\frac{N}{(k+1)^i}-1\right)\left(d+(k+1)^i\right)$ time units, where i is the number of steps to do the standard exchange algorithm before switching to the direct algorithm.

Proof. In the combined total exchange algorithm, under k-port modelling, each time a standard exchange algorithm step is done a cost of $d+\frac{N}{k+1}$ is added. This brings up the term $i\left(d+\frac{N}{k+1}\right)$. The first part of the second term is for the number of nodes in the goups doing the direct algorithms simultaneously. The second part stands for the delay associated with the transfer of messages. ■

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 hundred thousand examples) 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 modelling. Therefore, It is a conjecture that our algorithm is better than (or at least equal to one of the algorithms) both known algorithms. Tables 5 and 6 below summarize some typical example.

Table 5: Total exchange time, N = 1024

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 6: 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.

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 message, 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 the nodes 2, and 6 for itself and the node 1. At the third step (actually, $\log_2 N$ steps), node 0 sends its message along with the other messages from the nodes 2, 4, and 6 to the node 1. It is clear that at the end of this step all nodes have exhanged all their messages. Thus, this algorithm, $TE2_{F1}$, has a cost of $\left(d+\frac{N}{2}\right)\log_2 N$.

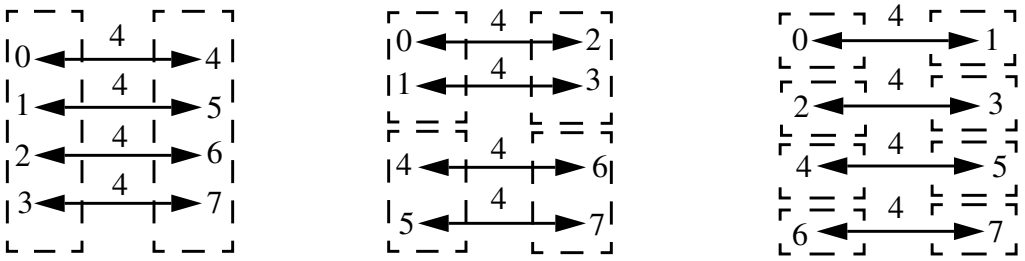


FIGURE 9. Total exchange on an 8-node OK_N under single-port modelling

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 with doing some of (or even none of) the steps involved in the standard total exchange algorithm, and then continue with the direct algorithm. That is, we divide the nodes in our complete network into halves and do the steps involved in the standard total exchange algorithm up to a point(s) that there is no gain in continuing so. Therefore, from that step(s) on, the direct algorithm is used for all the nodes in each of the created subgroups at the same time. Actually, the goal is to find the number of steps, or a bound for the number of steps, before switching to the direct algorithm such that the time associated with this algorithm is less than (or at least equal to) the other two (direct, and standard exchange) algorithms.

Let us explain this algorithm with $i = 1$ (number of doing the standard exchange algorithm) for the example shown in Figure 9. At the step 1, the nodes in our complete network are divided in halves. 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 $4d + 10$.

Lemma 2 The *combined total exchange algorithm* under single-port modelling on OK_N takes $i\left(d+\frac{N}{2}\right)+\left(\frac{N}{2^i}-1\right)\left(d+2^i\right)$ time units, where i is the number of steps to do the standard exchange algorithm before switching to the direct algorithm.

Proof. In the combined total exchange algorithm, each time a standard exchange algorithm step is done a cost of $d+\frac{N}{2}$ is added. This brings up the term $i\left(d+\frac{N}{2}\right)$. The first part of the second term is for the number of nodes in the goups doing the direct algorithms simultaneously. The second part stands for the delay associated with the transfer of messages which is doubled at each steps. ■

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 modelling requires node p at step i to send its message to the 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, $TE1_{Fk} = (d+1)\left(\frac{N-1}{k}\right)$.

communications and the length of messages at each step for multinode broadcasting on an 8 node message-based multicomputer. Unfortunately, latency hiding cannot improve this cost.

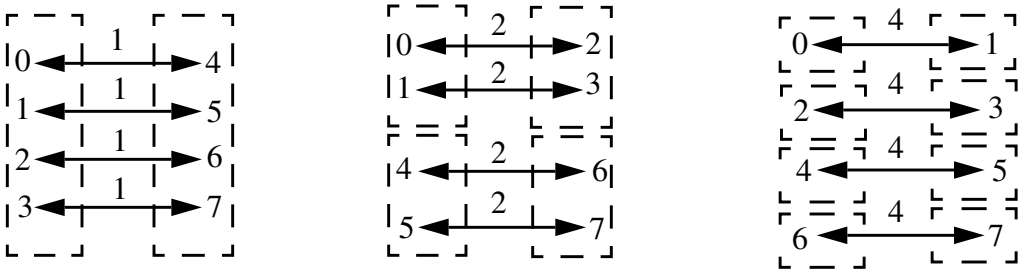


FIGURE 7. Multinode broadcasting on an 8-node OK_N under single-port modelling

***k-port*:** A simple algorithm is based on the extension of the direct algorithm for *k*-port modelling. That is, at step *i*, node *p* sends its message to the 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 $\frac{N}{k+1}$ 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 $G2_{Fk} = \sum_{i=1}^{\log_{k+1}N}\left(d + (k+1)^{i-1}\right) = \frac{N-1}{k} + d\log_{k+1}N$. Figure 8 illustrates the above algorithm when $N = 9$ and $k = 2$.

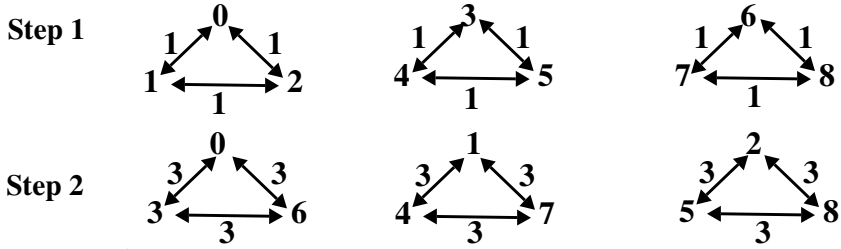


FIGURE 8. Multinode broadcaasting on a 9 node OK_N under *2*-port

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 the direct algorithms [24][25], at any step *i*, a node *p* sends the message to destined node $(p + i)$ *mod N*. Clearly, the cost of this algorithm, $TE1_{F1}$, is $(N - 1)(d + 1)$.

One may also use the standard exchange algorithm for total exchange similar to the ones used in hypercubes, and meshes [15][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 large messages to be transmitted as a single unit. Consider this algorithm for an 8-node multicomputer, as shown in Figure 9. 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

Now let us consider if latency hiding technique can have any effect on the implementation of the spanning binomial algorithm for scattering operation. We let all nodes have their optical interconnect fixed so that nodes can immediately send their messages, if applicable, as soon as they receive messages from their parents. Figure 6 shows the communication cost, and implementation of this algorithm for our example. It is evident that nodes receive their messages at diferent time steps. Meanwhile, it is seen that latency hiding technique does not improve the communication cost, and it is still equal to $(N-1)+d\log_2 N$. In general, spanning binomial algorithm has a much better performance than the sequential algorithm for our network (except for the trivial case, $N = 2$, where they have the same cost).

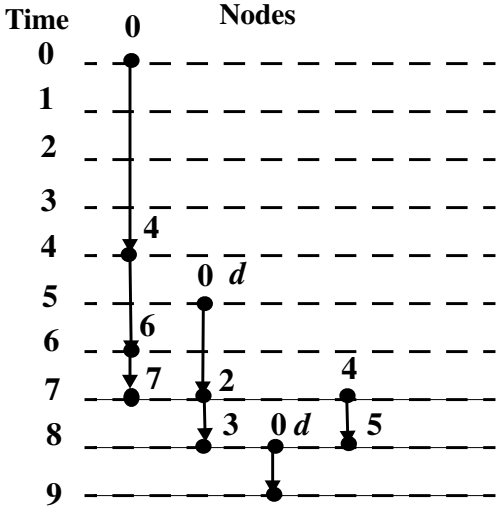


FIGURE 6. Spanning binomial algorithm with latency hiding technique for scatterng operation, $d = 1$

***k-port*:** Sequential tree algorithm can be extended for k-port modelling. That is, at each step the source node sends its k different messages to k other different nodes. Therefore, $S1_{Fk} = (d+1)\left(\frac{N-1}{k}\right)$. Another algorithm is proposed by F. Desprez et al [7] for scattering operation. In this algorithm, the scattering node n_0, sends k messages of $\frac{N}{k+1}$ length each, to its k children. Therefore, there are $(k + 1)$ nodes having $\frac{N}{k+1}$ 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 $\sum_{i=1}^{\log_{k+1} N} \left(d + \frac{N}{(k+1)^i}\right) = \frac{N-1}{k} + d\log_{k+1} N$ time units.

5. Multinode broadcasting

In multinode broadcasting, also called gossiping [11], all nodes send their unique messages to all other nodes, and this is basically used in parallel algorithms when all nodes need to exchange their data. The simplest algorithm for multinode broadcasting is to use our latency hiding broadcasting algorithm N times, one for each node. Another algorithm is to consider the multinode broadcasting as a degenerate case of total exchange, to be discussed in the next section. However, better algorithms exist.

***Single-port*:** In the direct algorithms [24][25], at any step i, a node p sends its message to node $(p + i)$ *mod* N. Clearly, the cost of this algorithm, $G1_{F1}$, is $(N\text{-}1)(d + 1)$.

One may use a better algorithm, just like the standard exchange algorithms for total exchange operation [15][3], where during each step, the complete network are recursively divided into halves, and messages are exchanged across new divisions at each step. This algorithm combines messages into large messages to be transmitted as a single unit. Actually, each node sends its message along with the other messages it received at the previous steps. Hence, the multinode broadcasting cost, $G2_{F1}$, is $(N-1)+d\log_2 N$ for a total of $\log_2 N$ steps. Figure 7 shows pairwise

cost is $d + M(\lceil \log_k N(k-1) + 1 \rceil - 1)$ time units. The following table compares these two algorithms.

Table 4: Multi-broadcasting time, k = 4, d =3, M = 10

N	$MB1_{Fk}$	$MB2_{Fk}$
85	60	33
1369	80	63
22703	100	83
88633	110	93

In what follows, we shall assume that the number of nodes in the OK_N is a power of 2, or a power of (k + 1) under single-port and k-port modelling, respectively.

4. Scattering

The scattering operation, is used basically to distribute data to the processors of a massive parallel computer. The easiest algorithm for scattering operation under single-port modelling is based on the *sequential tree* [23]. In this case, the source node sends its different message to each of the other nodes sequentailly, as shown in Figure 4. 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.

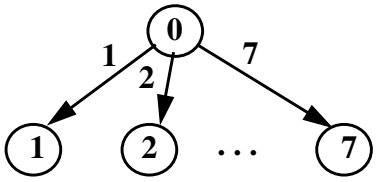
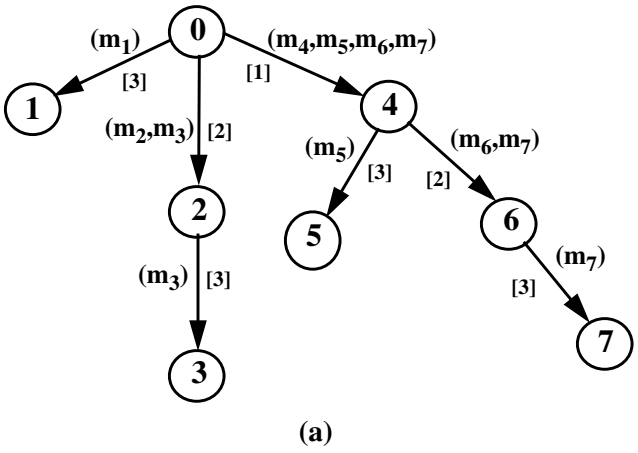


FIGURE 4. Sequential tree algorithm

Spanning binomial tree algorithm [20] 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. As illustrated in Figure 5, the source node sends its messages for the upper half of the nodes to the node 4. In the second step, nodes 0 and 4 are responsible for sending messages to the nodes in their halves. That is, to the node 2 (messages for nodes 2, and 3), and node 6 (messages for nodes 6, and 7), respectively. In the third step, all nodes send the remaining messages to the remaining nodes. These three steps (actually $\log_2 N$ steps) takes each (d + 4), (d +2), and (d + 1) time units, respectively. Generally, 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).



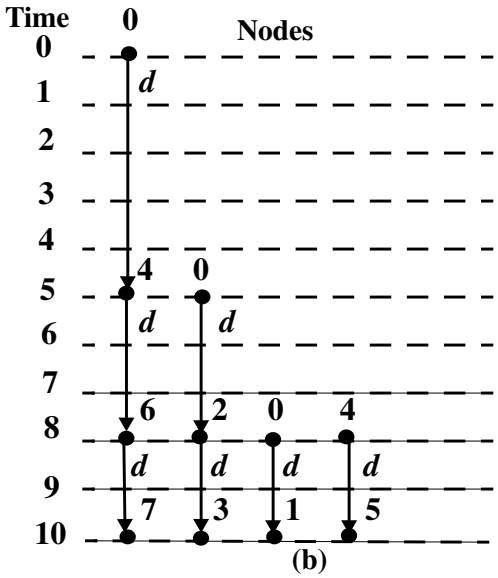


FIGURE 5. (a) Binomial tree algorithm, (b) its implementation on OK_N with d = 1

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

N	$B1_{Fk}$	$B2_{Fk}$	$B3_{Fk}$	$B4_{Fk}$	$\log_{k+1}N$
99	12	10	7	5	5
1393	20	14	11	8	7
19601	28	18	15	11	9
114243	32	22	17	13	11
1607520	40	28	21	16	14

Table 2: Broadcasting time, $k = 4$, $d = 3$

N	$B1_{Fk}$	$B2_{Fk}$	$B3_{Fk}$	$B4_{Fk}$	$\log_{k+1}N$
85	12	12	6	3	3
1369	24	20	9	5	5
22703	32	28	11	7	7
88633	36	32	12	8	8
1429100	44	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 modelling, the total number of nodes for generations 1, 2, 3, 4 are: $N_1 = S + 1$, $N_2 = \sum_{j=1}^{S-d-1} S-d-j$, $N_3 = \sum_{j=1}^{S-2(d+1)} j(S-2d-1-j)$, $N_4 = \sum_{j=1}^{S-3(d+1)} \frac{j(j+1)}{2!}(S-3d-2-j)$. If we continue in a similar manner to the k-port modelling, then the total number of nodes in all generations, N, would be:

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

The subsequent table provides some numerical examples for the broadcasting time, B_{F1}, (in terms of t_m) of our latency hiding algorithm, of the spanning binomial algorithm [26], 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 modelling can be used.

Table 3: Broadcasting time, $d = 3$

N	$(d+1)\lceil \log_2 N \rceil$	$\log_2 N$	B_{F1}
69	28	7	12
1252	44	11	21
8657	56	14	27
82629	68	17	34

3.2 Multi-broadcasting

If there are M messages to be broadcast by a node to all other nodes, the simplest algorithm is to use the above latency hiding broadcasting algorithm M times in sequence. This algorithm, denote it by *MB1*, takes $M \times (d + B4_{Fk})$, and $M \times (d + B_{F1})$ time units under k-port and single-port modelling, respectively.

One may think of another algorithm, *MB2*, where the first message embeds a broadcasting tree (first generation tree) rooted at node n_0; Each of the subsequent messages use this embedded tree to broadcast bypassing thus the reconfiguration costs that the first message incurred. Thus, the first message will incur a delay of $d + (\lceil \log_k N(k-1) + 1 \rceil - 1)$ time units to broadcast over all N nodes and to embed the broadcast tree, while the second and subsequent messages would only incur a broadcast delay of $\lceil \log_k N(k-1) + 1 \rceil - 1$ each. Therefore, the total-

$Q^3_{S-1} = k^2\left(k^{S-2d-5}(k)\right) = \frac{k^{S-2d-1}-k^{S-2d-2}}{k-1}$.

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 denotes the total number of nodes of the trees in the third generation rooted at depth i. Therefore, $Q_{2(d+1)} = \frac{k^{S-2d-1}-k}{k-1}$, $Q_{2(d+1)+1} = 2\frac{k^{S-2d-1}-k^2}{k-1}$, $Q_{2(d+1)+2} = 3\frac{k^{S-2d-1}-k^3}{k-1}$, ..., $Q_{S-1} = (S-2d-2)\frac{k^{S-2d-1}-k^{S-2d-2}}{k-1}$. 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 fourth and fifth generations as $N_4 = \sum_{j=1}^{S-3d-3} \frac{j(j+1)}{2!}\left(\frac{k^{S-3d-2}-k^j}{k-1}\right)$, and $N_5 = \sum_{j=1}^{S-4d-4} \frac{j(j+1)(j+2)}{2!}\left(\frac{k^{S-4d-3}-k^j}{k-1}\right)$. 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 the 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 [5]. ■

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

$$N = \frac{k^{S+1}-1}{k-1} + \sum_{i=1}^{\left\lceil \frac{S}{d+1} \right\rceil - 1} \sum_{j=1}^{S-i(d+1)} \binom{j+i-2}{i-1} \frac{k^{S-i(d+1)+1}-k^j}{k-1} \quad \text{(EQ 1)}$$

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 best case $\log_{k+1}N$ when $d = 0$, for a particular number of nodes, N, reconfiguration delay, d, and port modelling, k. It is quite clear that the broadcasting time of our algorithm is much less than the known algorithms.

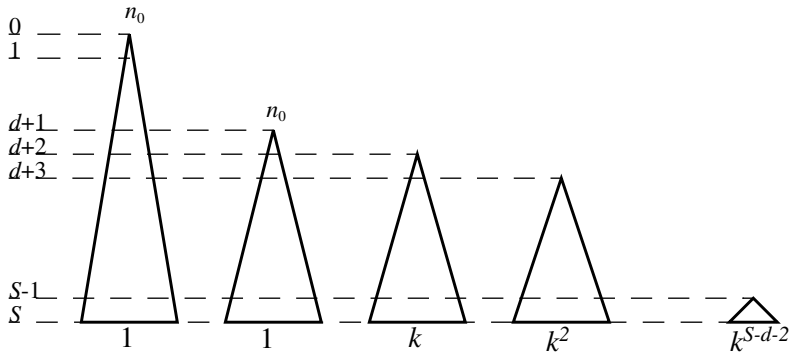


FIGURE 3. First and second generation trees. The numbers underneath each tree denote the number of trees having the same height. These trees are rooted at nodes that were at the same level in the first generation tree.

$M_{d+2} = k\left(k + k^2 + \ldots + k^{S-(d+2)}\right) = \frac{k^{S-d} - k^2}{k-1}$, and $M_{d+3} = k^2\left(k + k^2 + \ldots + k^{S-(d+3)}\right) = \frac{k^{S-d} - k^3}{k-1}$. 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 $\left\lceil \frac{S}{d+1} \right\rceil$.

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 denotes the total number of nodes in these trees rooted at depth i. Therefore $Q_{2(d+1)}^1 = k + k^2 + \ldots + k^{S-2(d+1)} = \frac{k^{S-2d-1} - k}{k-1}$, $Q_{2(d+1)+1}^1 = k\left(k + k^2 + \ldots + k^{S-2(d+1)-1}\right) = \frac{k^{S-2d-1} - k^2}{k-1}$, and $Q_{2(d+1)+2}^1 = k^2\left(k + k^2 + \ldots + k^{S-2(d+1)-2}\right) = \frac{k^{S-2d-1} - k^3}{k-1}$. This continues until the depth S -1 where we have: $Q_{S-1}^1 = k^{S-2d-3}(k) = \frac{k^{S-2d-1} - k^{S-2d-2}}{k-1}$.

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 denotes the total number of nodes in these trees rooted at depth i. Therefore, $Q_{2(d+1)+1}^2 = k\left(k + k^2 + \ldots + k^{S-2(d+1)-1}\right) = \frac{k^{S-2d-1} - k^2}{k-1}$, $Q_{2(d+1)+2}^2 = k\left(k\left(k + k^2 + \ldots + k^{S-2(d+1)-2}\right)\right) = \frac{k^{S-2d-1} - k^3}{k-1}$, and $Q_{2(d+1)+3}^2 = k\left(k^2\left(k + k^2 + \ldots + k^{S-2(d+1)-3}\right)\right) = \frac{k^{S-2d-1} - k^4}{k-1}$. This continues until the depth S - 1 where we have $Q_{S-1}^2 = k\left(k^{S-2d-4}(k)\right) = \frac{k^{S-2d-1} - k^{S-2d-2}}{k-1}$.

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, $Q_{2(d+1)+2}^3 = k^2\left(k + k^2 + \ldots + k^{S-2(d+1)-2}\right) = \frac{k^{S-2d-1} - k^3}{k-1}$, $Q_{2(d+1)+3}^3 = k^2\left(k\left(k + k^2 + \ldots + k^{S-2(d+1)-3}\right)\right) = \frac{k^{S-2d-1} - k^4}{k-1}$, $Q_{2(d+1)+4}^3 = k^2\left(k^2\left(k + k^2 + \ldots + k^{S-2(d+1)-4}\right)\right) = \frac{k^{S-2d-1} - k^5}{k-1}$. This continues until the depth S - 1 where we have

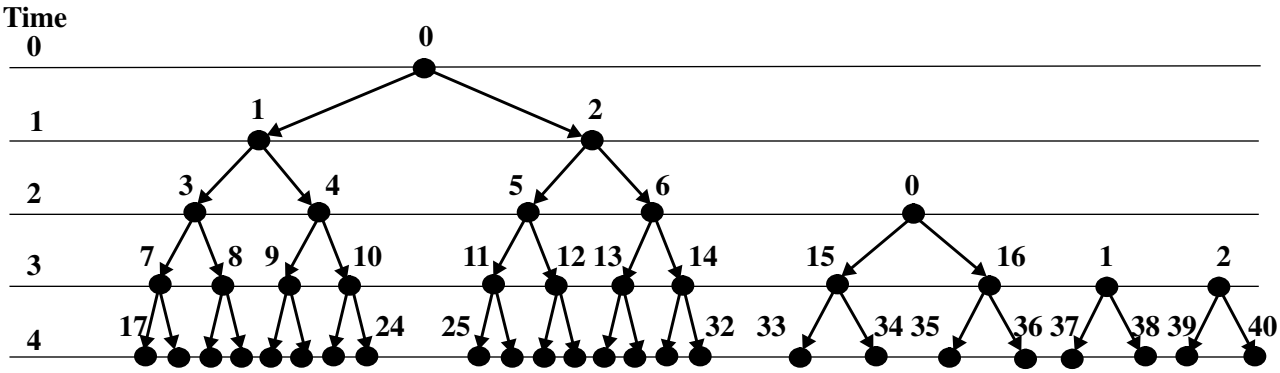


FIGURE 2. Latency hiding broadcasting algorithm for OK_N with 41 nodes, $k = 2, d = 1$

configured, and each node can now reconfigure to the first iteration connection cached, cycling through the next, cached, set of connections.

The above grouping schema can also be used for the multi-broadcasting case, to be covered later. In this case, the first message is used to establish the links to be used during the operation. As soon as the broadcasting of the first message is done, all the nodes reconfigure each connecting to the first group of nodes cached. They are now ready to participate in the broadcasting of the subsequent messages. In the susequent, we shall assume that the initial configuration steps have already been taken.

3.1.1 Analysis of the greedy algorithm

Before presenting the analysis of the greedy algorithm, it is worth noting that it can be shown that the total number of nodes, $N(S)$, informed up to step S follows the recurrence relations:

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, these relations cannot be solved for a general d. Therefore, we present another approach to find a closed formula for the total number of nodes, $N(S)$, up to the step S.

The problem we shall endeavor to solve is to find the time required for the greedy algorithm to complete. We shall approach the analysis constructively, that is, 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 and assuming no reconfiguration, belong to a k-ary tree rooted at node n_0 and of depth S. 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 the first generation tree and reconfigured, as the second generation 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 of the nodes.

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

ality, we shall assume that $t_m = 1$ for a message of unit length. Also, we shall 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 should mention that our modelling 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 , for broadcasting time, multi-broadcasting time, scattering time, multinode broadcasting (also called gossiping) time, and total exchange time, respectively. We derive time complexities of collective communication algorithms in our network under the model m, where $m = \{F1, Fk\}$. $F1$ and Fk stands for single-port, and k-port modelling under full-duplex communication, to be defined later in section two.

3. Broadcasting/Multi-broadcasting

Given that the reconfiguration delay d is the dominant factor contributing to message delay, we shall concentrate in techniques that could effectively hide it in our broadcasting/multi-broadcasting algorithms. Such techniques include latency hiding whereby a link that is to be used frequently remains established (our network utilizes free space propagation techniques, and hence 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 $(d+1)\left\lceil \frac{N-1}{k} \right\rceil$ 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 $\lceil \log_k (N(k-1)+1) \rceil - 1$ steps, while in terms of elapsed time, the algorithm will require $(d+1)\,(\lceil \log_k (N(k-1)+1) \rceil - 1)$.

Obviously, one can do better than this if one 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. Proceeding this way, this algorithm, $B2_{Fk}$, 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 because of the large reconfiguration delay that each node incurs. We are interested in devising algorithms that will overcome the existence of the large reconfiguration delays by essentially hiding it. The 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, a better algorithm is proposed here. We adopt a greedy algorithm, $B4_{Fk}$, where the broadcasting node reconfigures to reach k children which lead to a pre-configured tree of an appropriate $O(\log_k N)$ depth. As soon as the broadcasting node has finished sending its message, it reconfigures its links to reach another predefined tree. It is understood that while this node is reconfiguring (this takes d steps time units), nodes that have already been configured and are in possession of the message send it to k neighbors each. This process repeats at each node every time it sends the message. Potentially, the message, starting at 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 before node n_x be able to reconfigure. Figure 2 depicts the algorithm for a 2-port network with 41 nodes and a reconfiguration delay of 1.

A simple mechanism of counting the number of times the message is sent by a node and caching the links ensures the correct formation of the trees needed. 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 k nodes from the second group, 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 k nodes, it caches the names of the previous group together with the iteration number. This process continues until all nodes have received the message. With this technique, all the reconfigurations are done in parallel. Once the second group of nodes is empty, it means that the network has been

In section two of this paper, we define a complete and reconfigurable optical interconnection network using free-space beam routing (to be called OK_N), and then discuss its communication modelling. 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, known algorithms from literature for scattering and multinode broadcasting are adapted to our network. Then, a new algorithm for 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 memeory. The nodes are connected by a complete graph. By a complete network, we mean that any node has the capability to connect to any other node directly in a single hop. The links can be established dynamically on demand between any two nodes by reconfiguring the optical interconnect.

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 with another node at the same time is supported.

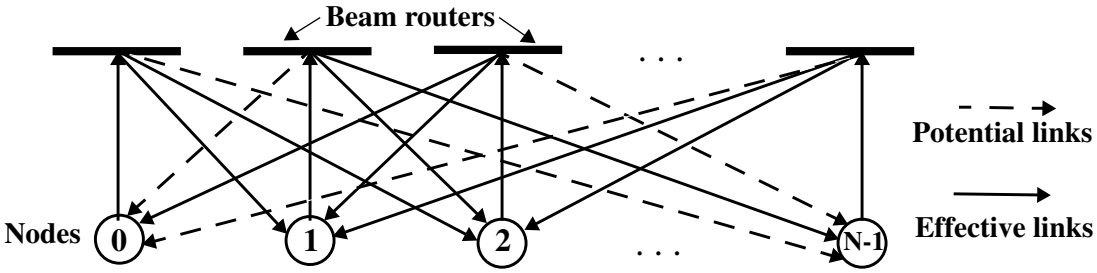


FIGURE 1. OK_N, a massively parallel computer interconnected by a complete free-space optical interconnetion network

Various implementation technologies exist to embody the above abstract model. Such technologies include computer generated holograms [28] and/or deformable mirrors for switching [29], frequency hoping for coding, wavelength tuning for transceivers [2], VCSELs [9] and/or SEEDs [13] 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 and we leave it for future works. However, a very brief appendix is presented at the end of the paper discussing such possible techniques. For our purposes, we shall consider that one of the techniques will have been implemented, and the various overheads associated with tuning the receivers, reconfiguring the optical beam router or sending the frequency code information are lumped together in the reconfiguration delay d to be discussed later in this section.

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, error correcting codes 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 etc.), the transmission code in the case of frequency hoping etc. The transmission time then becomes $T = d + t_s + l_m\tau$. We incorporate both t_s and $l_m\tau$ into a single message delay $t_m = t_s + l_m\tau$. Thus the message transmission time becomes $T = d + t_m$. For the remaining of the discussion, and without loss of gener-

1. Introduction

Message-based multicomputers, also called *massively parallel computers*, 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 the performance of massively parallel computers, there has been a great deal of interest in the design of their 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 in a single-hop. Complete interconnection networks can be modeled by a complete graph, K_N, where the computing nodes and communication links are mapped onto the vertices and edges of the graph. In a complete graph, 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 of practical interest when N is large. These limitations prevent implementing complete networks using metal-based interconnections as there is a fixed physical link between any two nodes.

Optics is ideally suited for implementing interconnection networks because of its superior characteristics over electronics [22][10][16], such as higher interconnection density, higher bandwidth, suitability for reconfigurable interconnects, larger number of fan-ins and fan-outs, 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. Consequently, future massively parallel machines will employ optical interconnections in an ever increasing manner.

Research is focusing in two major directions; One is to implement traditional, and design new novel, point-to-point interconnection networks [18] but utilizing the advantages of the new emerging optical technologies to overcome the constraints of metal-based communication, while the second direction is to design new interconnection networks [4][27] based on the constraints of optical devices, such as optical passive star coupler (OPS) interconnection networks which can be modeled by hypergraphs.

Along the first direction, we introduce a complete network using a *reconfigurable free-space optical interconnect*. Free-space optical interconnections result in much denser interconnection networks than OPS-based and guided-wave interconnections [6][18], and has the potential to solve the problems associated with implementing complete networks due to its ability to reconfigure the interconnections. It can easily exploit the third spatial dimension which dramatically increases the available communication bandwidth.

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 computer generated holograms, mirrors, prisms, lenses etc.

The study of parallel algorithms have shown some generic communication primitives that appear very often [17][14][8]. *Collective communications*, in which more than two processes are involved, are common basic patterns of interprocessor communication that are frequently used as buliding 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, and hence, on the overall performance of the parallel computers. These collective communication problems include *broadcasting, multi-broadcasting, scattering, gathering, multinode broadcasting,* and *total exchange.* In broadcasting, a node sends its unique message to all other nodes. It is used in a variety of linear algebra algorithms [14], such as matrix-vector multiplication, matrix-matrix multiplication, LU-factorization, and Householder transformations. In multi-broadcasting, a node broadcasts a number of messages to all other nodes. In scattering, a node sends a different messsage to all other nodes. It is basically used for distribution of data among the processors. In gathering, a node receives a different message from all the other nodes. Gathering is the exact reverse of scattering, so we will not discuss it as a seperate operation. In multinode broadcasting, all nodes send their unique messages to all other nodes. In total exchange, all nodes send their different messages to all other nodes. *Personalized communication*s (scattering, gathering, and total exchange) are used, for instance, in transposing a matrix, or in neural network simulations. Excellent surveys on collective communication algorithms in different networks can be found in [11][12][19].

Collective Communications on a Reconfigurable Optical Interconnect[1]

Ahmad Afsahi Nikitas J. Dimopoulos

Technical Report ECE-97-2

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

September, 1997

Abstract. Free-space optical interconnection is used to fashion a reconfigurable network. Since network reconfiguration is expensive compared to message transmission, we utilize *latency hiding techniques* to increase performance. For this network, we present and analyze broadcasting/multi-broadcasting algorithms that utilizes latency hiding and reconfiguration in order to speed these operations. 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. This research was supported through grants from the Natural Sciences and Engineering Research Council of Canada.