Communications in Binary Fat Trees*

Vassilios V. Dimakopoulos Nikitas J. Dimopoulos
Department of Electrical and Computer Engineering
University of Victoria, P.O. Box 3055,
Victoria, B.C., CANADA, V8W 3P6
E-mail: {dimako,nikitas}@ece.uvic.ca

Abstract

Fat trees are built around complete b-ary trees but have processing nodes only at the leaf level and may have different branch capacities in different levels. In this paper we study the communication capabilities of binary fat trees (including the simple binary tree) with respect to five major communication operations: broadcasting, multinode broadcasting, scattering, gathering and total exchange. We present and analyse optimal and nearly optimal algorithms for the five operations.

Keywords: *binary trees, collective communications, fat trees, interconnection networks.*

1 Introduction

Recently there has been an increased interest in designing efficient algorithms to handle the communication requirements of distributed memory multiprocessors. Such algorithms seek to schedule regular communication patterns and find their way to standard library routines so that effectively most of the interconnection network details are hidden from the user.

The need for efficient communication was realized quite early, especially in the context of parallel numerical algorithms [4, 5] where a variety of such regular communication patterns was observed. Although the terminology is not standard yet, five major patterns have been identified [1, 6, 10]

- *broadcasting* where a certain node sends the same message to all nodes in the network
- *scattering* where a certain node sends different messages to the other nodes
- *gathering* where a certain node receives a message from every other node
- *multinode broadcasting* which involves simultaneous broadcastings from all nodes
- *total exchange*, which involves simultaneous scatterings (or gatherings) from all nodes.

It is worth noting that other related operations have been studied; for example multicasting [8] is a generalized form of broadcasting where the receiving nodes may form a proper subset of the nodes in the network.

The above communication modes occur in a variety of situations although numerical algorithms and especially matrix-related ones form the best paradigm. Total exchange for example is associated with FFT algorithms and matrix transposition while multinode broadcasting arises naturally in iterative algorithms [1]. Scattering and gathering are considered dual operations. An algorithm for one of the problems can be transformed to an algorithm for the other by simply reversing the data paths.

With the advent of Thinking Machines' CM-5 [12], fat tree networks have received increased attention. Fat trees are hierarchical networks built around complete trees but have processing nodes only at the leaves. The capacities of the branches in a fat tree may not remain constant in all levels, but rather increase in an unspecified manner towards the root. In this paper we address the algorithmic nature of the five communication operations in fat trees. Experimental results on the subject have been reported in [9].

This work concentrates on binary fat trees although the results can be easily extended to other fat trees as well. In the next section we give the necessary definitions, the model we will follow and lower bounds on the time requirements of the five communication problems. The actual algorithms and analyses appear in sections 3 – 5. Section 6 concludes the work.

2 Preliminaries

A b-ary fat tree [7] is a complete b-ary tree whose branches get thicker (i.e. increase in capacity) as one moves from the leaves to the root. Level 0 of the tree is the leaf level and level $\log_b n$ is the level of the root node as shown in Fig. 1; n (a power of b) is the number

*This research was supported in part through grants by NSERC, IRIS and the University of Victoria

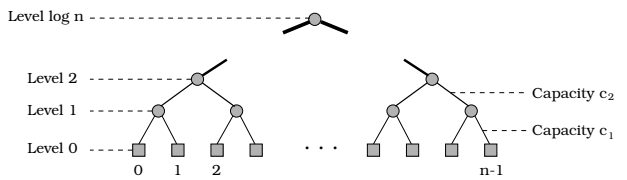

Figure 1 A fat tree

	Lower Bounds
Broadcasting	$2\log n$
Scatter/Gather	$n+1$
Multi. Broadcast.	$n+1$
Total Exchange*	$\max\{n+1, n^2/4c_{\log n}\}$

*(bound is not tight)

Table 1 Lower bounds

of leaves, which are numbered from left to right as 0, 1, ..., $n-1$. The leaves correspond to processing nodes while the other levels include only routing nodes. We are going to concentrate in binary fat trees (BFT) and use the notation $\log n$ to represent the logarithm of n in base $b = 2$. The branches between levels $i-1$ and i have capacity $c_i \geq 1$ and if $j > i$ then $c_j \geq c_i$. The branch capacity corresponds to the number of physical links included in the branch.

Although the results presented here cover any capacities arrangement, two capacity patterns will be of interest: constant and exponential which represent the two extremes in interconnection size. In the first case $c_i = 1$ for all levels $i = 1, 2, \ldots, \log n$ and the resulting fat tree (CBFT) is identical to the simple complete binary tree with processing nodes only at the lowest level. The exponential-capacity tree (EBFT) has $c_i = 2^{i-1}$ and as a result the total number of physical links at each level is equal to n.

In the following it is assumed that the network is packet-switched. Each of the routing nodes is equipped with packet queues (although this will not always be necessary) and is able to utilize all its incident links simultaneously. This is usually refered to as the multiport model. Messages are assumed to consist of a single packet and to be of the same size so that transferring a message between two neighbors needs a constant amount of time which we take as one time unit (or step). The links are assumed to be bidirectional and fully duplex; half-duplex links where only one direction can be accommodated at a time cause a slowdown by at most a factor of two.

2.1 Lower bounds

Broadcasting under the multiport model requires D steps in a network with diameter equal to D. In our case this means that $2\log n$ steps are enough; since the path between any pair of nodes is unique in the tree, broadcasting is easily accomplished by setting all routing nodes to a broadcast mode whereby the received message is replicated towards all directions. Broadcasting will not be further discussed here.

Scattering and gathering in general trees of processors were considered in [2]. Although related, our case differs in that the significant nodes are confined to the leaves of the tree. The observation that the source node has to send or receive $n-1$ different messages over its single incident link leads to a simple lower bound of $n-1$ steps. As a matter of fact the exact bound is $n+1$ steps and can be seen as follows. In gathering $n-1$ messages at processor 0, in the first four steps we can at most receive only two messages since node 1 is in distance two and either of nodes 2 or 3 is in distance four away. Consequently there will be at least two steps with no message reception by node 0. The same bound applies to the case of multinode broadcasting as every node will send and receive $n-1$ different broadcast messages over its bidirectional incident link.

The same argument though for the case of total exchange does not always yield a tight bound. Consider a BFT and observe that there are $(n/2)^2$ messages to cross the root node on their way from the left to the right subtree. If the capacity of the branches incident to the root node is $c_{\log n}$ then at least $n^2/4c_{\log n}$ steps are needed for all $(n/2)^2$ messages to get across. At the extremes, this renders total exchange as an $\Omega(n^2)$ operation for CBFTs, and an $\Omega(n)$ operation for EBFTs. It should be clear that this simple argument does not yield tight bounds. In fact, a tighter lower bound for EBFTs will be proven in section 5. The bounds are summarized in Table 1.

In the next sections we show that scattering, gathering and multinode broadcasting can be accomplished in the minimum number of steps by the simple CBFT. As a consequence, increasing the capacities of branches offers no advantage to these operations. Total exchange is the only case where branch capacities matter and is treated in section 5.

3 Scattering

Since a gathering algorithm can be had from a scattering algorithm (and vice versa) by simply reversing the data paths, we only consider scattering here.

The lower bound of $n+1$ steps can be achieved in the CBFT (hence in any other fat tree) using the *furthest-first* scheduling discipline whereby the source node gives priority to messages that have to travel the

furthest.

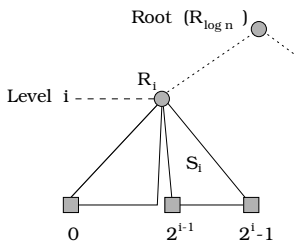

Figure 2

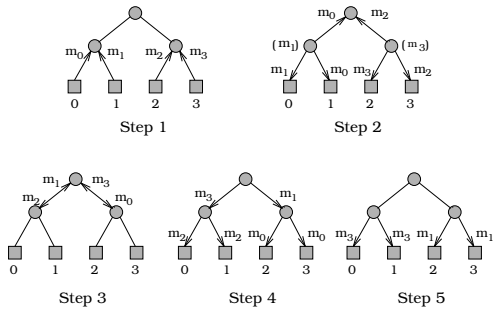

Figure 3 Multinode broadcasting in a 4-node CBFT

Theorem 1 *The furthest-first discipline results in an optimal scattering algorithm for BFTs.*

Proof. Assuming without loss of generality that node 0 of a CBFT is the source node, consider the routing node R_i at level i that is the root of the subtree containing leaf nodes 0 to $2^i - 1$, as shown in Fig. 2. Let S_i be the right subtree of R_i which contains nodes 2^{i-1} to $2^i - 1$. The last message destined for a leaf of S_i leaves node 0 just before the first message for the left subtree of R_i does, according to the furthest-first regimen; consequently the last message for a leaf of S_i is dispatched at time $n - 2^{i-1}$, for all $i = 1, 2, \ldots, \log n$. Since the leaves of S_i are in distance $2i$ from node 0, the last message for S_i will reach its destination at time

$$T_i = n - 2^{i-1} + 2i - 1.$$

Consequently the algorithm will finish at time

$$T = \max_{1 \le i \le \log n} \{T_i\}.$$

It can be seen that T_i is a decreasing function of i for $i \ge 2$. The maximum value for integer i can thus occur only for $i = 1$ or 2. Since $T_1 = n$ and $T_2 = n + 1$, we see that $T = n + 1$ as claimed. ■

4 Multinode broadcasting

We show here that in a simple CBFT where all nodes begin broadcasting at the same time, and all nodes have enough buffers, all messages will have been received at time $n + 1$, i.e. multinode broadcasting can be achieved in time equal to the lower bound. The algorithm is in effect a *flooding* procedure [11] where received messages are replicated to all directions (expect the one they came from) when received by intermediate nodes. An example for the case of $n = 4$ nodes is shown in Fig. 3.

Theorem 2 *Multinode broadcasting can be performed in the minimum number of steps in a CBFT.*

Proof. The case of 4 nodes was covered in Fig. 3. Assuming as an induction hypothesis that for $n \ge 4$ we need exactly $n + 1$ steps, we shall show that $2n + 1$ steps are enough when $2n$ nodes are involved.

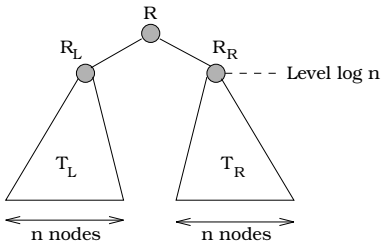

Figure 4

Consider two CBFTs T_L and T_R with n leaves each and root nodes R_L and R_R correspondingly. A $2n$-leaf CBFT includes an extra node adjacent to both R_L and R_R as shown in Fig. 4. At time $\log n$ (when the first two messages of the leaves of T_L reach node R_L) all the other messages of T_L are pipelined below node R_L so that they can cross the node one by one in the following steps. Analogous is the situation in T_R.

For multinode broadcasting to finish within T_L there are another $n + 1 - \log n$ steps remaining, based on the induction hypothesis. But the first message of T_R will either be queued behind a T_L message on its way to the leaves of T_L or will arrive at the first level of T_L in the next $\log n + 1$ steps, the rest of T_R messages following behind it. In the second scenario, since $n \ge 4$ it is seen that $n + 1 - \log n \ge \log n + 1$. Consequently, in both cases the messages from the leaves of T_R have enough time to arrive before multinode broadcasting is finished within T_L. As a result n more steps after time $n + 1$ are enough to deliver the n messages from T_R to the leaves of T_L, proving the claim. ■

4.1 Queueing considerations

The flooding algorithm analyzed above achieves the lower bound but bares the cost of excessive queueing

requirements. Consider any BFT and assume that the *last* message to be received by node 0 is the message m from a node in distance $2i$ away. Normally m would arrive at time $2i$ but due to contention it arrives at time $n+1$ instead, hence it was delayed by $n+1-2i$ steps. Assuming FIFO queues, this means that the *sum* of the queue sizes along its path reached the value $n+1-2i$; on the average the queue size per node was $(n+1-2i)/2i$. The minimum value for the last expression occurs for $i = \log n$, and it shows that at best the average queue size will be $O(n/\log n)$. For CBFTs it can be shown that the exact queue sizes needed at the outgoing links of a level-i routing node reach the value 2^{i-1} (see [3]).

It should be clear that the queueing requirements of the algorithm are excessive. The routing nodes should be kept as low in complexity and cost as possible; the presence of queueing plus the large size of the queues do not contribute to that effect. It is worth noting that there exists an $(n + 2\log n - 2)$-step algorithm that eliminates the queues completely for any BFT [3].

5 Total exchange

Total exchange represents the densest form of communication as each node has a different message to send to every other node. The branch capacities now control the complexity of the algorithms as shown in Table 1.

For a CBFT a simple solution consists of n consecutive scatterings one by each node in turn. Such an algorithm has the advantage that no queueing capability is required from the routing nodes. Based on our previous results the n scatterings will take $n(n+1)$ steps. This time can be reduced in half by observing that two scatterings can be performed simultaneously with no contention.

We now give an algorithm that further reduces the time requirements, induces no queueing, and is applicable to any capacities pattern. The algorithm runs in time close to the lower bounds in Table 1 and is in principle similar to total exchange algorithms for hypercubes [1]. It can be recursively stated as follows: initially the left and right subtrees of the root node exchange their $2(n/2)^2$ messages meant for the opposite sides. Then, the two subtrees perform internally a total exchange in parallel. Iteratively, the algorithm can be stated as:

```
For all i = 0 to log n − 1
  /* Phase i */
  Do in parallel for all level log n − i nodes
    Transfer all messages of the left subtree
    meant for the right subtree and vice versa.
```

During the ith phase a node at level $h_i = \log n - i$ has to pass $(2^{h_i-1})^2$ messages in each direction over its incident branches which have capacity c_{h_i}. This means that only c_{h_i} messages can cross at a time. To avoid contention while maintaining the maximum speed, exactly c_{h_i} leaves should dispatch messages at a single step, and the messages should be appropriately chosen so that their destinations are distinct.

Consider the ith phase and the jth node from the left at level h_i, $R^j_{h_i}$, $0 \le j \le 2^{i-1}-1$. In Fig. 2 node R_i is actually node R^0_i under our notation. Node $R^j_{h_i}$ is the root of the subtree that contains leaves $j2^{h_i}$ to $(j+1)2^{h_i}-1$. In the case of CBFTs $c_{h_i} = 1$, i.e. only one message should be dispatched at a time. We divide phase i in 2^{h_i-1} periods. In each period, one node from each of the two subtrees of node $R^j_{h_i}$ sends its 2^{h_i-1} messages one by one. Consequently, phase i can be implemented as follows:

```
/* Phase i for CBFTs */
Do in parallel for all j = 0 to 2^{i−1} − 1
  For k = 0 to 2^{h_i−1}  /* 2^{h_i−1} periods */
    For l = 0 to 2^{h_i−1}  /* 2^{h_i−1} messag. to send */
      Node j2^{h_i} + k sends to j2^{h_i} + 2^{h_i−1} + l and
      Node j2^{h_i} + 2^{h_i−1} + k sends to j2^{h_i} + l.
```

In the case of EBFTs, all leaves must dispatch messages simultaneously in order to achieve the maximum speed. Consider leaf k in the subtree with $R^j_{h_i}$ as the root node. If $\oplus$ is the bitwise XOR operation, then $k \oplus 2^{h_i-1}$ inverts the (h_i-1)th bit in the binary representation of k and consequently, if k belongs to the left subtree of $R^j_{h_i}$ then $k \oplus 2^{h_i-1}$ is a node belonging to the right subtree and vice versa. Thus we can let phase i consist of 2^{h_i-1} steps. During step l any node k sends its message destined to node $k \oplus 2^{h_i-1} \oplus l$ and there will be no contention between messages:

```
/* Phase i for EBFTs */
Do in parallel for all leaf nodes k = 0 to n − 1
  For l = 0 to 2^{h_i−1}  /* 2^{h_i−1} messages to send */
    Node k sends to node k ⊕ 2^{h_i−1} ⊕ l.
```

Timing analysis: During phase i there are $(2^{h_i-1})^2$ messages to cross a level $h_i = \log n - i$ node in each direction, and only c_{h_i} messages can do that at a time. Accounting also for the initial delay of $2h_i$ steps for the first message to reach the opposite side, phase i needs

$$T_i = \left\lceil \frac{(2^{h_i-1})^2}{c_{h_i}} \right\rceil + 2h_i - 1.$$

The total time needed for the algorithm is thus

$$T = \sum_{i=0}^{\log n-1} T_i = \sum_{i=1}^{\log n} \left\lceil \frac{4^{i-1}}{c_i} \right\rceil + (\log n)^2. \qquad (1)$$

Pipelining the phases: Instead of executing the phases serially, we can pipeline them appropriately.

Consider phase i exactly $h_i - 1$ steps before it finishes. This is the time when the last message from a level h_i node is transferred to one of its children. Phase $i+1$ could have safely started $h_i - 2$ steps before that time instant, and no contention would occur because no message of that phase would have reached the children of the level h_i node yet. Consequently, phases i and $i+1$ can have $2h_i - 3$ steps overlapping, leading to a savings of $\sum_{i=0}^{\log n - 2}(2h_i - 3) = (\log n)^2 - 2\log n + 1$ steps. Using (1), we conclude that the total number of steps for the new algorithm is

$$T = \sum_{i=1}^{\log n} \left\lceil \frac{4^{i-1}}{c_i} \right\rceil + 2\log n - 1. \qquad (2)$$

Simplifying further, for the CBFT we have $c_i = 1$ and

$$T_{\mathrm{CBFT}} = \sum_{i=1}^{\log n} 4^{i-1} + 2\log n - 1 = \frac{n^2 - 1}{3} + 2\log n - 1.$$

For EBFTs, $c_i = 2^{i-1}$ and (2) gives

$$T_{\mathrm{EBFT}} = \sum_{i=1}^{\log n} 2^{i-1} + 2\log n - 1 = n + 2\log n - 2. \quad (3)$$

5.1 The exponential capacities case

The above analysis shows that the presented total exchange algorithm is suboptimal with respect to the lower bounds of Table 1 by a very small multiplicative factor in the worst case. It should be clear that the given bound is not tight since in deriving it not all message transmissions where considered. We show below that for the case of EBFTs a tighter lower bound exists which guarantees that our algorithm is very close to optimal (within $2\log\log n$ steps) for such trees.

If at any given step of an algorithm a node did not receive a message we say that a *hole* occurred in its receptions. Notice that if the average number of holes per node was h then the algorithm takes time $T \geq n - 1 + h$; the equality holds if all nodes had the same number of holes (h).

Consider the graph G_i which is derived from the EBFT by collapsing its last i levels into a single vertex. We then have a collection of 2^i subtrees $T_1, T_2, \ldots, T_{2^i}$ each having $n/2^i$ nodes and their roots have a single common parent. In addition we introduce new edges between every pair of leaves in each of the 2^i subtrees. We also impose the restriction that *only one message may be received by a leaf at any time* so that the new edges cannot be used simultaneously. It is seen that any total exchange algorithm for the original EBFT is a total exchange algorithm for G_i (but not vice versa) — the message transfers within the last i levels of the EBFT are substituted by no-ops in G_i and the additional edges in G_i are not utilized. As a result, the graph G_i can be used to derive a lower bound on total exchange algorithms for the EBFT.

A message will be called *internal* to subtree T_j if both the source and the destination lie in T_j otherwise the message is *external*. The crucial observation is that holes will start appearing in G_i after the first external message is dispatched. Assume that T_0 is the one to send the first e external messages at time t. Then at time $t+1$ there will be e holes in the nodes of T_0. In fact, due to the $2(\log n - i + 1)$-step delay external messages suffer before reaching their destinations, no messages will have arrived from other subtrees before time $t + 2(\log n - i + 1)$. Consequently the number of holes in T_0 between times t and $t + 2(\log n - i + 1)$ will be equal to the total number of external messages it sent during this period.

We will now find the minimum number of holes that will occur in every node in G_i. Consider any total exchange algorithm for G_i and partition time in $2(\log n - i + 1)$-step periods. If there are p such periods in total, the algorithm needs time $T \geq 2p(\log n - i + 1)$. Assume that at the sth period there were $k_s^{(j)}$ holes in T_j. Hence in the first period the total number of holes was

$$k_1 = k_1^{(1)} + k_1^{(2)} + \cdots + k_1^{(2^i)}.$$

This is exactly the number of external messages sent (in total) during the first period. During the second period the total number of holes was

$$k_2 = k_2^{(1)} + k_2^{(2)} + \cdots + k_2^{(2^i)}$$

and as a result the total number of external messages sent during the second period was *at most* equal to $k_1 + k_2$. In general, during the jth period the maximum number of external messages was $k_1 + k_2 + \cdots + k_j$. Hence after p periods the total number of external messages observed was at most

$$\begin{aligned} \sum_{j=1}^{p}(k_1 + k_2 + \cdots + k_j) &= \sum_{j=1}^{p}(p - j + 1)k_j \\ &= p\sum_{j=1}^{p} k_j - \sum_{j=1}^{p}(j-1)k_j \\ &= pH - Q \end{aligned}$$

where $H = \sum k_j$ is the total number of holes and Q is a non-negative term.

Each node must sent in total $n - 1$ messages out of which the $n - n/2^i$ will be external, so that the total number of external messages will be $n(n - n/2^i)$.

This means that $pH \geq n(n - n/2^i)$. Notice that the average number of holes per node is $h = H/n$ and since $T \geq 2p(\log n - i + 1)$, the last inequality yields

$$hT \geq 2n(1 - \frac{1}{2^i})(\log n - i + 1). \quad (4)$$

Since (4) holds for any i, it holds for $i = \log\log n + 1$ as well, which gives

$$\begin{aligned} hT &\geq 2n(1 - \frac{1}{2\log n})(\log n - \log\log n) \\ &= 2n\log n - 2n\log\log n - n + n\frac{\log\log n}{\log n} \\ &\geq 2n\log n - 2n\log\log n - n \quad (5) \end{aligned}$$

From our earlier discussion, if h is the average number of holes per node then $T \geq n - 1 + h$, the equality holding if all nodes have exactly h holes. Assume now that $h < q = 2\log n - 2\log\log n - 1$. Then (5) gives

$$\begin{aligned} T &\geq \frac{2n\log n - 2n\log\log n - n}{2\log n - 2\log\log n - 2} \\ &= n + \frac{n}{2\log n - 2\log\log n - 2} \end{aligned}$$

and after a bit of algebra, we get $T > n - 1 + q$. On the other hand, if $h > q$ then $T \geq n - 1 + h > n - 1 + q$. Consequently, the minimum time can be had only for $h = q$ and in that case

$$T = n - 1 + q = n + 2\log n - 2\log\log n - 2 \text{ steps.} \quad (6)$$

Lemma 3 *Any optimal total exchange algorithm for EBFTs needs at least* $n + 2\log n - 2\log\log n - 2$ *steps.*

Proof. This is an immediate consequence of the facts that a total exchange algorithm for EBFTs is a total exchange algorithm for G_i and that for G_i the lower bound is given by (6). ■

Intuitively, an optimal total exchange algorithm for graph G_i requires strictly less time than an optimal algorithm for the EBFT. We conjecture here that there exists no total exchange algorithm for EBFTs that requires less than $n + 2\log n - 2$ steps.

6 Conclusion

We have studied the implementation and performance of communication operations in fat trees where the processing nodes are confined to the leaf level. Although we concentrated on binary fat trees, the results can be generalized to b-ary fat trees easily. Broadcasting, scattering and multinode broadcasting can be performed optimally in trees with minimal branch capacities. Increased capacities on the other hand are beneficial in such operations as total exchange which involve a large number of messages.

There are a number of issues to be considered. Multinode broadcasting has excessive queueing requirements if it is to be performed in the minimum number of steps. What is the lower bound under the constraint of no queueing? The total exchange algorithm we presented induces no queueing and is very close to optimal especially in the case of EBFTs. Improved bounds, though, for this problem need to be found as the straightforward are not tight.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewoods Cliffs: Prentice - Hall, 1989.

[2] S. N. Bhatt et al., "Scattering and gathering messages in networks of processors," *IEEE Trans. Comput.*, Vol. 42, No. 8, pp. 938–949, Aug. 1993.

[3] V. V. Dimakopoulos and N. J. Dimopoulos, "Leaf communications in trees and fat trees," Technical Report ECE-94-7, University of Victoria, Sept. 1994.

[4] D. B. Gannon and J. van Rosendale, "On the impact of communication complexity on the design of parallel numerical algorithms," *IEEE Trans. Comput.*, Vol. C-33, No. 12, pp. 1180–1194, Dec. 1984.

[5] S. L. Johnsson, "Communication efficient basic linear algebra computations on hypercube architectures," *J. Parallel Distrib. Comput.*, Vol. 4, pp. 133–172, 1987.

[6] S. L. Johnsson, C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, Vol. 38, No. 9, pp. 1249–1268, 1989.

[7] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Comput.*, Vol. C-34, No. 10, pp. 892–901, Oct. 1985.

[8] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," in *Proc. 1990 Int'l Conf. Parall. Proc.*, 1990, pp. 114–118.

[9] R. Ponnusamy, R. Thakur, A. Choudhary and G. Fox, "Scheduling regular and irregular communication patterns on the CM-5," in *Proc. Supercomputing '92*, Minneapolis, Nov. 1992, pp. 394–402.

[10] Y. Saad and M. H. Schultz, "Data communications in parallel architectures," *Parallel Comput.*, Vol. 11, pp. 131–150, 1989.

[11] D. M. Topkins, "Concurrent broadcast for information dissemination," *IEEE Trans. Softw. Eng.*, Vol. 11, No. 10, pp. 1107–1112, Oct. 1985.

[12] C. E. Leiserson et al., "The network architecture of the Connection Machine CM-5," in *Proc. 4th ACM Symp. Parall. Algor. Arch.*, June 1992, pp. 272–285.