Leaf Communications in Complete Trees*

Vassilios V. Dimakopoulos *Nikitas J. Dimopoulos*

Technical Report ECE-95-6

Dept. of Electrical and Computer Engineering, University of Victoria
P.O. Box 3055, Victoria, B.C., CANADA, V8W 3P6.
Tel: (604) 472-4289, 721-8902, 721-8773, *Fax:* (604) 721-6052,
E-mail: {dimako, nikitas}@ece.uvic.ca

October, 1995

Abstract

One of the crucial issues associated with loosely-coupled multiprocessors based on interconnection networks is information dissemination. A number of regular communication patterns seem to cover a large portion of information dissemination requirements in practice: broadcasting, scattering/gathering, multinode broadcasting and total exchange. In this work we consider tree networks where the processing nodes are confined to the leaves of the tree. This type of hierarchical topology is also the basis of fat tree networks. We present and analyze algorithms and their timing requirements for the aforementioned problems under two models of communication capabilities.

Keywords: collective communications, complete trees, fat trees, interconnection networks, packet-switched networks.

*This research was supported in part through grants by NSERC and the University of Victoria. This is a revised and expanded version of Technical Report ECE-94-7.

1. Introduction

Distributed memory multiprocessors are based on a collection of independent processing nodes integrated through an interconnection network. The presence of locally generated data spawns the need for information dissemination: a process through which "knowledgeable" nodes inform the rest of the network about the items of information they possess.

A number of information dissemination problems appear frequently in practice and their effective solution is of paramount importance to system performance. They include:

- *broadcasting* where a particular node needs to send the same message to all the other nodes in the network
- *multinode broadcasting* where every node performs a broadcasting
- *scattering (gathering)* where a particular node needs to send (receive) different messages to (from) all the other nodes
- *total exchange* or *multiscattering* where every node performs scattering or gathering; that is, every node has distinct messages to send to the other nodes in the network.

It is worth noting that the above list does not exhaust all communication possibilities. Other operations have also been studied; for example multicasting [11] is a generalized form of broadcasting where the receiving nodes may form a proper subset of the nodes in the network.

The above communication problems are also known as *collective communications*. The need for their efficient solution was realized quite early, especially in the context of parallel numerical algorithms [5, 7]. Broadcasting is an essential operation in almost every parallel algorithm. Multinode broadcasting arises naturally in iterative algorithms [2]. Total exchange is a communication pattern occurring for example in FFT algorithms and is usually identified with matrix transposition.

Saad and Schultz [12], Johnson and Ho [8] and Bertsekas *et al* [1] are some of the researchers that studied the above communication problems for hypercubes. Since then there has been a considerable amount of work on communication algorithms for networks other than hypercubes and for a number of different models of communication; see for example [13]. Two excellent surveys on the subject were given by Hedetniemi, Hedetniemi and Liestman [6] and Fraigniaud and Lazard [4].

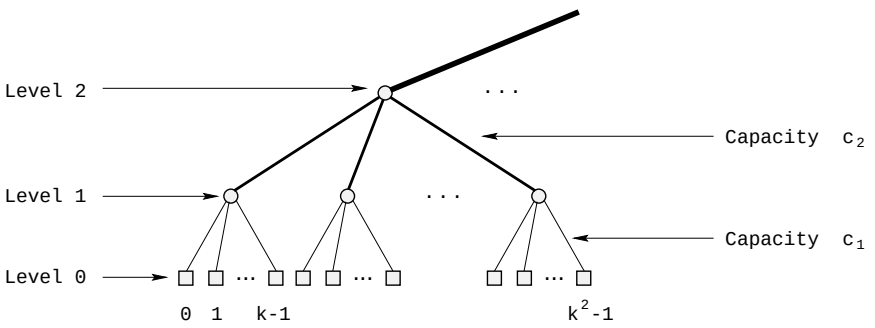

Figure 1. *A k-ary tree*

In this work we consider the complete tree topology where processors are confined to the leaf level. The well-known *fat trees*, introduced by Leiserson [9] and utilized in Thinking Machine's CM-5 multiprocessor [10], are also based on the complete tree topology. The tree is termed "fat" because the capacities of the branches do not remain constant but rather increase towards the root. Originally Leiserson studied binary trees but CM-5 is based on quaternary ones. Here we will consider k-ary trees where k is any integer greater than one. We will study the above communication problems in the context of such a topology and determine the time requirements for solving them in the optimum time.

It will be assumed that the network operates in packet switched mode and we will consider two different communication capability models for the routing nodes, based on whether they can utilize all their outgoing links simultaneously or not. After some introductory definitions, we concentrate on the single-port model in Section 2, where routing nodes are able to use only one outgoing link at a time. Communications under the assumption that all outgoing links can be used in a single step are considered in Section 3. Finally, Section 4 concludes the paper.

1.1. Preliminaries

Portion of a complete k-ary tree is shown in Fig. 1. Each node has k children, except for the leaves. The tree consists of $\log_k n + 1$ levels of nodes. Level 0 of the tree is the leaf level and level $h = \log_k n$ is the level of the root node; n is a power of k and denotes the number of leaves, which will be numbered from left to right as 0, 1, ..., $n-1$, as shown in the figure. The leaves correspond to *processing* nodes while all the other levels

include only *routing* nodes. We will assume that the tree has at least three levels, i.e. that $h \geq 2$ or equivalently, $n \geq k^2$ (if $h = 1$ the graph degenerates to a star which consists of n vertices adjacent to an additional central vertex).

A *fat tree* is based on the same topology only branches get thicker (i.e. increase in capacity) as one moves from the leaves to the root. 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 capacity arrangement, two capacity patterns will be of more interest in later sections: constant and exponential. These two patterns represent two extremes in interconnection size. In the first case $c_i = 1$ for all levels $i = 1, 2, \ldots, h$ and the resulting tree is the simple k-ary tree. In the second case, $c_i = k^{i-1}$. Every level of such a tree has n links in total, being able to carry messages from all leaves at the same time.

The network is assumed to be packet-switched. Messages are of the same size and consist of a single packet, 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 in some of the algorithms. Each of the routing nodes is equipped with packet queues (although this will not always be necessary for the algorithms we will discuss). Finally, we will consider two models of communication capability: in the first one, nodes will be able to utilize only one of their output links at a time (*single-port* model). In the *multiport* model, all links incident to a node can be utilized simultaneously.

2. Communications under the single-port model

It should be clear that if every node is incapable of sending more than one message at every time unit, branch capacities have no effect at all. Any fat tree behaves exactly like the corresponding complete k-ary tree. In this section we will derive lower bounds for the communication problems we consider and will provide algorithms that achieve the lower bounds. Notice that we allow any node to receive messages from *all* its neighbors simultaneously but it can only send one message at each step.

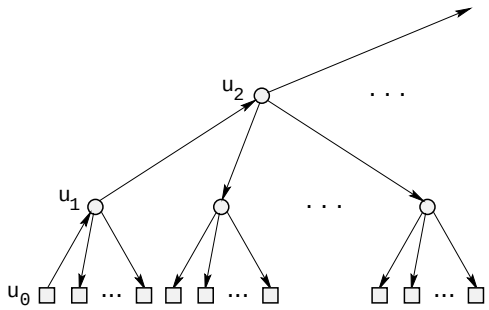

Figure 2. *Broadcasting under the single-port model*

2.1. Broadcasting

Let us first consider broadcasting from the root v of an arbitrary tree T, where v has d children v_i, $i = 1, 2, \ldots, d$. Let T_{v_i} be the subtree rooted at v_i and assume that broadcasting in T_{v_i} needs $b(T_{v_i})$ steps. Let us further assume for convenience that the trees have indices reversely to their broadcast times, i.e. if $i \leq j$ then $b(T_{v_i}) \geq b(T_{v_j})$. Since the root can send the message to only one child at a time it is clear that the best plan is to send the message to v_1 first, then to v_2, etc. As a result, the time needed to broadcast from v in T is

$$b(T) = \max_{1 \leq i \leq d} \{b(T_{v_i}) + i\}. \tag{1}$$

If T is a complete k-ary tree with height ℓ then the k subtrees rooted at the children of root v are identical: they are all complete k-ary trees with height $\ell - 1$. Let $B_r(\ell)$ be the broadcast time from the root in T. Then the broadcast time for any of the k subtrees is $B_r(\ell - 1)$, and from (1), setting $d = k$, $b(T) = B_r(\ell)$, $b(T_{v_i}) = B_r(\ell - 1)$, we obtain

$$B_r(\ell) = B_r(\ell - 1) + k \Rightarrow B_r(\ell) = k\ell. \tag{2}$$

since $B_r(1) = k$.

Our problem, however, is to broadcast from a leaf of a complete k-ary tree. Let, without loss of generality, the leftmost leaf be the source node, labeled u_0 in Fig. 2. Starting from v_0's only parent (labeled u_1) and following the upward edges we observe the following: below the ith node u_i in this path there are $k - 1$ complete k-ary trees of height $i - 1$. If u_i gets informed about the message then all the leaves in these $k - 1$ subtrees can be informed in $B_r(i-1)+k-1$ steps, the earliest. Of course, u_i also has to inform its parent.

In order to determine the exact broadcast time for v_0 we will start from the end of the upward path, calculate the minimum time needed for broadcasting from that node and then work our way down to u_0, using (1) at each step. We let $b(T_{u_i})$ be the broadcast time remaining after u_i receives the message.

Node u_h in the path is the root node of our original tree. It can only be informed through its leftmost child. When u_h receives the message it has to inform its $k-1$ remaining children as we mentioned above and for this $B_r(h-1)+k-1$ steps will be needed. Consequently,

$$b(T_{u_h}) = B_r(h-1) + k - 1.$$

Next, consider u_{h-1}. This node has to inform its $k-1$ subtrees lying below plus its parent, u_h. For the subtrees below $B_r(h-2)+k-1$ steps are needed, while for broadcasting from u_h, $B_r(h-1)+k-1$ steps are necessary as we just saw. According to (1), the best plan is to send the message to u_h first and to the subtrees below next. Consequently, we need time:

$$b(T_{u_{h-1}}) = \max\{b(T_{u_h})+1, B_r(h-2)+k\} = b(T_{u_h})+1.$$

Proceeding downwards in this manner it is seen that any node u_i must first inform its parent (u_{i+1}) and then its $k-1$ remaining children, and

$$b(T_{u_i}) = \max\{b(T_{u_{i+1}})+1, B_r(i-1)+k\} = b(T_{u_h})+h-i,$$

giving

$$b(T_{u_0}) = b(T_{u_h}) + h = B_r(h-1) + k - 1 + h.$$

Since from (2) $B_r(h-1) = k(h-1)$, we have the following result (recall that $h = \log_k n$):

Theorem 1 *Broadcasting in fat trees under the single-port model requires* $(k+1)\log_k n - 1$ *steps.*

Fig. 3 shows an algorithm that achieves this lower bound; it is directly derived from the preceding analysis.

2.2. Scattering

Scattering and gathering in general trees of processors were considered in [3]. Although related, our case differs in that the significant nodes are confined to the leaves of the tree.

Source Node:

1 Send message to parent;

Every Routing Node:

1 Send message to parent (if any);

2 Send message to each child in turn, except to the one the message came from;

Figure 3. *Optimal broadcasting algorithm under the single-port model*

The observation that the source node has to send or receive $n-1$ different messages over its incident link leads to a simple lower bound of $S(n) = G(n) \geq n-1$ steps. As a matter of fact the exact bound is n or $n+1$ steps (depending on k) and can be seen as follows. In gathering $n-1$ messages at processor 0, in the first two steps we can at most receive one message since the closest leaf is at distance two. Consequently there will be at least one step with no message reception by node 0, i.e. $S(n) = G(n) \geq n$. If the tree is binary ($k = 2$) there will be one extra step of no reception since the next closest leaves (2 and 3) are both at distance four from node 0, i.e. $S(n) = G(n) \geq n+1$ if $k = 2$.

Since a gathering algorithm can be had from a scattering algorithm (and vice versa) by simply reversing the data paths, we will only consider scattering here. The lower bound of n or $n+1$ steps can be achieved using the *furthest-first* scheduling discipline whereby the source node gives priority to messages that have to travel the furthest.

Theorem 2 *The furthest-first discipline results in an optimal scattering algorithm.*

Proof. Assuming without loss of generality that leaf node 0 is the source node, consider the routing node R_i at level i which is the root of the subtree containing nodes 0 to $k^i - 1$, as shown in Fig. 4. Let S_j be the subtree rooted at the jth child of R_i from the left (the source node belongs to S_0). The last message destined for any of the subtrees $S_1, S_2, \ldots, S_{k-1}$ will leave node 0 before any message destined for a leaf within S_0 does, according to the furthest-first regimen; that is, it will leave at time $n - k^{i-1}$. Since this message is destined for a node at distance $2i$ from node 0, the last message for any of the

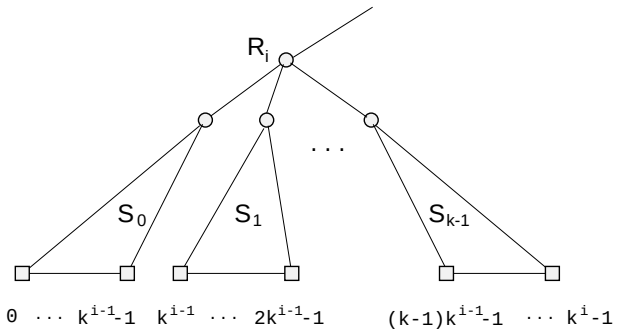

Figure 4.

aforementioned $k-1$ subtrees of R_i will be received at time

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

Consequently, the algorithm will finish at time

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

It can be seen that T_i is a decreasing function of i for $i \ge 2$. The maximum value of T_i for integer i can thus occur only for $i = 1$ or $i = 2$. Since $T_1 = n$ and $T_2 = n - k + 3$, we see that $T = n$ if $k \ge 3$ and $T = n + 1$ if $k = 2$, as claimed. ■

2.3. Multinode broadcasting

In multinode broadcasting every node broadcasts its own message. If all nodes start broadcasting at the same time contention will be observed sooner or later at the routing nodes. An optimal multinode algorithm schedules the traffic in each routing node so that all leaves receive all messages at the minimum possible time. The algorithm we present next will be proven to be optimal. First, however, we derive the lower bound for multinode broadcasting algorithms.

Theorem 3 *Any multinode broadcasting algorithm under the single-port model requires at least* $kn + (k+1)(\log_k n - 2) + 1$ *steps.*

Proof. Consider the k children v_i, $i = 1, 2, \ldots, k$, of the root node. Node v_1 is at level $h-1$ and consequently has k^{h-1} leaves lying below. Each of the leaves will generate a

broadcast message and this message must be sent through a child of v_1 to v_1 itself. Then v_1 must send it to its remaining $k-1$ children and to the root of the tree. Thus v_1 must make $kk^{h-1} = k^h$ transmissions in total.

In addition, v_1 will receive all messages coming (through the root) from v_2, v_3, ..., v_k. Each of these messages must be broadcast by v_1 to all its k children. Thus each of these messages also forces v_1 make k transmissions. Since each v_i will send k^{h-1} messages, v_1 will receive $(k-1)k^{h-1}$ messages from the other children of the root. The total number of transmissions for v_1, including the k^h transmissions from the previous paragraph, will thus be $k(k-1)k^{h-1} + k^h = k^{h+1} = kn$. Under the single-port model v_1 can only send one message at a time. Thus it will have to be busy for at least kn steps.

The *first* message will arrive at v_i at time $h-1$ the earliest since the closest leaf is at distance $h-1$. The *last* message to leave v_i will be sent to one of its children. But this child will have to broadcast this message to the subtree it roots, which is a complete k-ary tree with height $h-2$. In other words, after the last message leaves v_i it will need another $B_r(h-2)$ steps, as given by (2), in order for the last leaf to receive it. Consequently, any multinode broadcasting algorithm needs time

$$\begin{aligned} MB(n) &\geq kn + h - 1 + B_r(h-2) \\ &= kn + h - 1 + k(h-2) \\ &= kn + (k+1)(\log_k n - 2) + 1. \end{aligned}$$

■

In Fig. 5 we give an algorithm for the multinode broadcasting problem and in Fig. 6 we demonstrate its operation for $n = 4$ leaves. After some initial delay, a routing node starts receiving messages from its subtrees and from its parent. "Phase" A in the algorithm keeps forwarding all messages from the children to the parent but also retains a copy of them in its queue (in Fig. 6 phase A for level-1 nodes ends in the 3rd step). In phase B messages in the queue (and any other messages coming from the parent) are transmitted to the appropriate children. Notice that the children *take turns* in receiving messages, i.e. each child receives one message per k steps.

Lemma 1 *By the time a routing node sends the last message to its parent (i.e. phase A ends), all its k children will have no messages in their queues.*

Every Leaf Node:

Send message to parent;

Every Routing Node: (except the root)

A Receive all messages from children, sending a copy of them upwards, one-by-one;

B Until the last message arrives from parent

Send one message to each child in turn;

Root Node:

B While receive all messages from children, keep sending one message to each child in turn;

Figure 5. *Optimal multinode broadcasting algorithm under the single-port model*

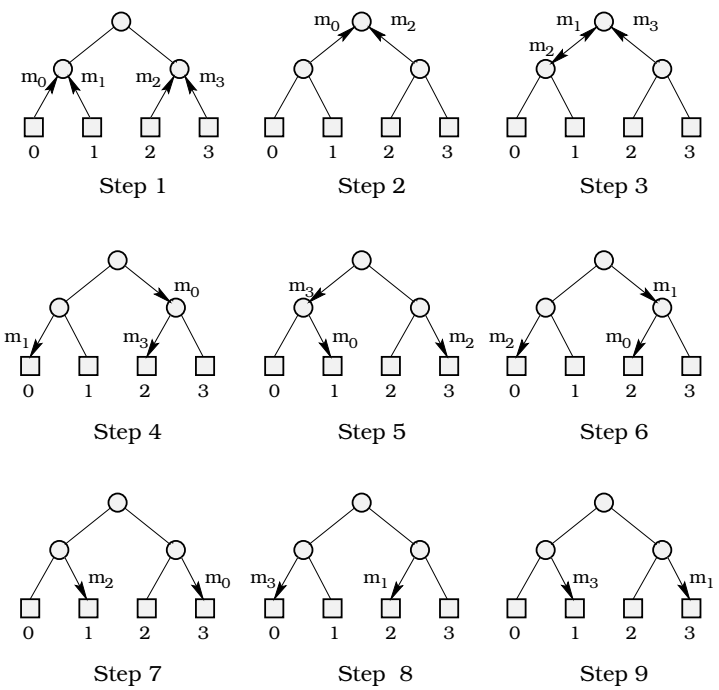

Figure 6. *Multinode broadcasting in 4 leaves under the single-port model*

Proof. Let R_i be a node at any level $i \leq h-2$ and let R_{i+1} be its parent. Node R_i sends the last message to its parent at time $i + k^i$ since it will receive k^i messages from the leaves below and the first message will arrive at the ith step. Similarly, R_{i+1} will send the last message to its parent at time $i + 1 + k^{i+1}$. Now, R_i needs exactly $(k-1)k^i$ time units to exchange the messages between its children (since it has received k^i messages and each one must be given to $k-1$ children). Consequently, its queue becomes empty at time $i + k^i + (k-1)k^i = i + k^{i+1}$. That is, R_i's queue becomes empty one step before R_{i+1} finishes sending messages to its parent. ■

Theorem 4 *The algorithm presented in Fig. 5 is optimal.*

Proof. Consider any routing node R_i at level i and let T_A and T_B be the time this node needs for phase A and B correspondingly. Notice that since R_i is in level i, phase A will start i steps after the time multinode broadcast commences.

During phase B each message sent to a child of R_i has to be broadcast to the corresponding subtree, which has height $i-1$. Consider R_{i-1}, a child of R_i. When R_{i-1} receives the first message from R_i its queue is empty as Lemma 1 shows. For every received message, R_{i-1} needs k steps to broadcast it to its k children. From the way phase B is described above, R_{i-1} receives one message every k steps from R_i. Consequently, at the time R_{i-1} finishes broadcasting a message, it receives a new one from its parent. As a result, at the moment any message arrives from R_i, it finds R_{i-1}'s queue empty. The conclusion is that there is *no contention* at any node during phase B. Thus, after R_i sends the last message to its children (i.e. phase B ends), we only need $B_r(i-1)$ steps to finish.

Concentrating on a node R_{h-1}, it is seen that the time needed for the algorithm is $T = (h-1) + T_A + T_B + B_r(h-1)$. This is because R_{h-1} will receive its first message at time $h-1$, it will be busy for $T_A + T_B$ steps and $B_r(h-1)$ steps will be enough to deliver the last message it sends in phase B since, according to the above discussion, this message will not compete with any other message. Clearly, $T_A = k^{h-1}$ since k^{h-1} messages are received from its children. These messages must also be exchanged among R_{h-1}'s children and this will take $(k-1)k^{h-1}$ steps. In addition, R_{h-1} will receive $(k-1)k^{h-1}$ messages from the root node that have to be sent to all its children. Since only one transmission occurs at a time, $T_B = (k-1)k^{h-1} + k(k-1)k^{h-1}$. Finally, $B_r(h-1) = 2(h-2)$. The total time needed is seen to be $T = k^{h+1} + (k+1)(h-2) + 1$ which is the best possible

according to the lower bound derived previously. ■

One undesirable characteristic of the above algorithm is that it requires routing nodes to be equipped with large queues to buffer the incoming messages. We defer a detailed analysis of queueing for the section dealing with the multiport model of communication since the problem arises there, too.

2.4. Total exchange

Total exchange causes the densest form of contention on the network. Consider the subtree rooted at a routing node R_i in level i. A leaf u in this subtree will generate $n-1$ messages in total, out of which $n-k^{i-1}$ will pass through R_i as follows. There will be $n-k^i$ "upward" messages, that is messages to be sent through the parent of R_i to a subtree other than the one R_i roots. There will also exist messages to be given from the subtree u lies in, to the other $k-1$ subtrees of R_i. Those messages will be $(k-1)k^{i-1}$ in number, giving a total of $n-k^{i-1}$ messages of u that go through R_i. Under the single-port model we may easily determine the lower bound of total exchange algorithms for messages generated at the leaves.

Theorem 5 *Any total exchange algorithm under the single-port model requires at least* $n^2(2k+1)(k-1)/k^3 + 2\log_k n - 3$ *steps.*

Proof. Consider any total exchange algorithm and concentrate on some child v of the root node as we did in Theorem 3. Node v will receive $k^{h-1}(n-k^{h-2})$ different messages from its k^{h-1} leaves below. In addition, v will receive $(k-1)k^{h-1}k^{h-1}$ messages from the root of the tree, originating at the other subtrees of the root. All these messages must be transmitted to the appropriate neighbors of v and under the single-port model only one transmission is possible at a time. Accounting for the initial delay for the first message to arrive at v ($h-1$ steps) and the smallest possible delay for the last message leaving v to arrive to its destination (at best $h-2$ steps *after* the departure instant), we obtain

$$\begin{aligned} TE(n) &\geq k^{h-1}(n-k^{h-2}) + (k-1)k^{h-1}k^{h-1} + h - 1 + h - 2 \\ &= n^2\frac{(2k+1)(k-1)}{k^3} + 2h - 3. \end{aligned}$$

■

Every Leaf Node:

Send the $n-1$ messages in a furthest-first order;

Every Routing Node: (except the root)

A While there exist upward messages from children
    Send one message to parent;

B Until the last message arrives from parent
    Send one message to any (appropriate) child;

Root Node:

B While receiving all messages from children, keep sending one message to any (appropriate) child;

Figure 7. *Optimal total exchange algorithm under the single-port model*

We now present an algorithm that achieves the lower bound of Theorem 5. Every leaf node transmits its $n-1$ messages under the *furthest-first* discipline as in the case of scattering and every routing node sends all possible messages to its parent before it starts sending any messages to its children. The algorithm is given in Fig. 7.

Since leaves follow a furthest-first rule, "upward" messages arrive in routing nodes first, followed by messages destined to within the subtrees rooted at the routing nodes. In a similar fashion to Lemma 1 we have the following.

Lemma 2 *By the time a routing node sends the last message to its parent (i.e. phase A ends), all its k children will have no messages in their queues.*

Proof. Let R_i be a node at any level $i \leq h-2$, and let R_{i+1} be its parent. Since node R_i receives $k^i(n-k^i)$ "upward" messages, it will send the last message to its parent at time $i + k^i(n-k^i)$; similarly, R_{i+1} will send the last message to its parent at time $t = i+1+k^{i+1}(n-k^{i+1})$. Now, R_i will receive an additional $k^i(k-1)k^{i-1}$ messages to be exchanged between its k subtrees. Thus, its queue will become empty at time $i+k^i(n-k^i)+k^i(k-1)k^{i-1} = i+k^i(n-k^{i-1})$. It is easily seen that $i+k^i(n-k^{i-1}) < t$ for $k \geq 2$. Thus R_i's queue becomes empty before R_{i+1} finishes sending messages to its

Problem	**Time**
Broadcasting	$(k+1)\log_k n - 1$
Scattering/Gathering	n (if $k \geq 3$) , $n+1$ (if $k=2$)
Multinode Broadcasting	$kn + (k+1)(\log_k n - 2) + 1$
Total Exchange	$n^2(2k+1)(k-1)/k^3 + 2\log_k n - 3$

Table 1. *Time requirements for communications under the single-port model*

parent. ■

Theorem 6 *The algorithm presented above is optimal.*

Proof. The proof is similar to the proof of Theorem 4 so the details are omitted. Using Lemma 2 it is seen that during phase B in any routing node's schedule there is no contention of messages. Calculating the time needed for phases A and B leads to the the desired result. ■

2.5. Discussion

We summarized the results of the previous analyses in Table 1. It is interesting to compare the relative performance of different trees given that the number of leaves (n) is fixed. The broadcasting time is a decreasing function of k for $2 \leq k \leq 4$ and increases with k for $k \geq 4$; quaternary trees need thus the minimum time. For multinode broadcasting though, binary trees give the best performance as $MB(n)$ is an increasing function of k for $k \geq 2$. Finally, for the total exchange problem, ignoring the logarithmic term, it is seen that larger values of k are needed; at the extreme where $k = \sqrt{n}$ the network consists of only three levels and total exchange can be performed in $\Theta(n\sqrt{n})$ steps.

3. Communications under the multiport model

We will now assume that a node is able to utilize all its incident links simultaneously. This model of communication is affected by the capacity arrangement on the branches of the tree since now the number of messages allowed to cross a link can be greater than one. In order to compare the performance of fat trees with that of the simple complete k-ary

tree, we will assume that the processor (leaf) connections to/from the routing network are fixed, and consist of exactly one link. In other words, the capacity of level-1 branches will be $c_1 = 1$.

3.1. Single-source communications

We have already seen that if $e(v)$ is the eccentricity of node v in any network then broadcasting from v under the multiport model takes time equal to $e(v)$. In our case, since the furthest node from a source is a leaf at distance $2h$, broadcasting will require $B(n) = 2h = 2\log_k n$ steps. It is easily accomplished by setting the routing nodes to a broadcast mode whereby the received message is replicated towards all directions.

Scattering and gathering, under our assumptions, is governed by the same bounds as in the single-port case; there is only one link available from a leaf to a routing node, forcing only one message to be send or received at a time by a leaf. Consequently, $S(n) = G(n) \geq n$ (or $n+1$ if $k = 2$). Had we allowed $c_1 > 1$, other lower bounds could have been derived in an obvious way.

3.2. Multinode broadcasting

The same argument used for deriving bounds for scattering/gathering algorithms can be used to determine lower bounds for multinode broadcasting in the multiport model since every leaf has to receive $n-1$ different broadcast messages. The exact bounds are $MB(n) \geq n$ if $k \geq 3$ or $MB(n) \geq n+1$ if $k = 2$. It is seen from this bound (and from the fact that the bound will be shown to be tight) that capacities within the routing network make no difference at least in terms of speed.

We show next that in a complete k-ary tree where all nodes begin broadcasting at the same time, and all nodes have enough buffers, all messages will have been received at time n, i.e. multinode broadcasting can be performed in time equal to the lower bound. The algorithm we will follow is in effect a *flooding* procedure [14] where received messages are replicated to all possible directions (expect the one they came from) when received by intermediate nodes.

Theorem 7 *Multinode broadcasting can be performed in time equal to the lower bound.*

Proof. The proof is by induction on the height h of a complete k-ary tree. We will first assume that $k \geq 3$. Consider only two levels (i.e. $h = 1$ and there exist k leaves in total). It is clear that after the first step the root node knows all messages. Furthermore it is easy to see that the root may distribute them to the appropriate leaves in another $k-1$ steps using the following schedule: during step $i = 2, 3, \ldots, k$ it sends the message of leaf $i-1$ to leaves 0, 1, ..., $i-2$ and the message from leaf $i-2$ to leaves $i-1$, i, ..., $k-1$. The algorithm is thus completed in k steps.

Assume now as an induction hypothesis that the theorem holds for height $h-1 \geq 1$ (i.e. for k^{h-1} leaves we need k^{h-1} steps). We will show it holds for h levels, that is the algorithm requires $n = k^h$ steps.

An complete tree with height h consists of k trees $T_1, T_2, \ldots, T_k$ each with height $h-1$, rooted at nodes $v_1, v_2, \ldots, v_k$, plus an additional node v adjacent to $v_1, v_2, \ldots, v_k$. Concentrate on T_j without loss of generality. Let t_f be the time the first message(s) arrive at v_j. The rest of the messages from the leaves of T_j are pipelined below v_j and reach v_j in subsequent steps. Let also t_l be the time the last message from a leaf in T_j leaves v_j downwards, that is, the last message to complete multinode broadcasting within T_j. The first message from v_j will be available at some other node v_m at time $t_f + 2$ since it has to go through the root v. This also means that v_j itself will start receiving messages from other trees T_m $(m \neq j)$ continuously beginning at time $t_f + 2$. According to the induction hypothesis the algorithm operates in k^{h-1} steps within T_j. Since v_j will receive a total of $(k-1)k^{h-1}$ messages from other trees, and since these messages will leave v_j downwards after the k^{h-1} messages from within T_j do, the algorithm will operate in $k^{h-1} + (k-1)k^{h-1} = k^h = n$ steps; the only requirement is that the last message of T_j leaving v_j downwards is *immediately* followed by the additional messages from the other trees. In other words, all we have to prove is that $t_f + 2 \leq t_l$.

Clearly, the first message arrives in v_j at time $t_f = h - 1$. Node v_j will receive in total k^{h-1} messages from the leaves of T_j, each of which has to be transmitted to $k-1$ children of v_j; that is, there must be $(k-1)k^{h-1}$ transmissions towards its k children. Since only k transmissions may occur at any step, the last such message to leave v_j will do so at time $t_l = t_f + (k-1)k^{h-1}/k$. In order to show that $t_f + 2 \leq t_l$, all we have to show is that $(k-1)k^{h-2} \geq 2$, which is true since we have assumed that $k \geq 3$ and $h - 1 \geq 1$. This completes the induction.

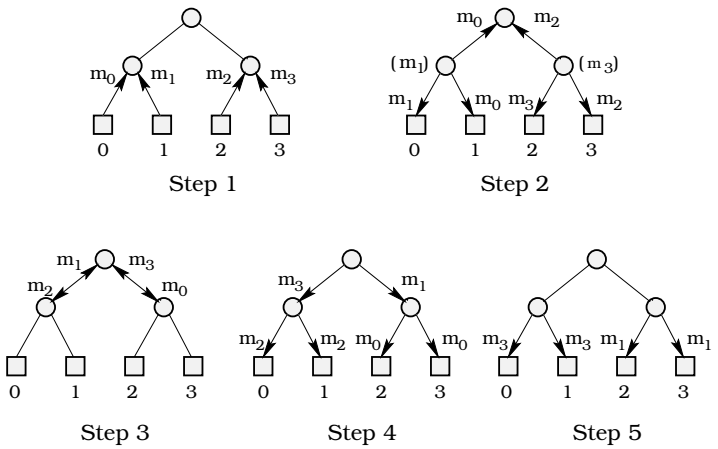

Figure 8. *Multinode broadcasting in a 4-leaf binary tree*

Quite similar arguments can be used to prove that for $k = 2$ the algorithm takes $n+1$ steps. The only difference is that the induction basis is for a tree with 3 levels. Fig. 8 shows that the algorithm takes 5 steps in this case. ■

3.2.1. Queueing considerations

The flooding algorithm analyzed above achieves the lower bound but bares the cost of excessive queueing requirements. Concentrate on a routing node other than the root node, i.e. a node R_i at level i, $1 \leq i \leq h-1$, of a simple k-ary tree. The node must maintain a queue at each outgoing link, for a total of $k+1$ queues. When a message is received, it is deposited to all the output queues (except one). Consider first the queue of the link between R_i and its parent. Node R_i will be continuously receiving k messages from its children but can only forward one of them at a time to its parent. Since this will occur for k^{i-1} units and R_i will have to send k^i messages to its parent, the queue of the link will reach the size of $k^i - k^{i-1} = (k-1)k^{i-1}$ messages.

On the other hand, if $i \geq 3$, there will be $n - k^i$ consecutive messages between (and including) steps $i+2$ and $i+n-k^i+1$, coming from the parent of R_i and intending to go through the link between R_i and one of its children. The same link will be used by the $(k-1)k^{i-1}$ messages of the other $k-1$ children, between steps i and $i+(k-1)k^{i-1}-1$. As a result, the queue of the link will reach the size of $\min\{i+(k-1)k^{i-1}-1, i+n-k^i+$

$1\} - (i+2) + 1$ messages; since $i < h$, it is seen that the exact size will be $(k-1)k^{i-1} - 2$ messages.

Although increasing the branch capacities intuitively seems to reduce the queueing requirements, this is not the case for all links. Assume that the last message to be received by node 0 is the message m from a node at distance $2i$ away. Normally m would arrive at time $2i$ but due to contention it arrives at time n ($n+1$ if the tree is binary) instead, hence it was delayed for $n - 2i$ steps. Assuming FIFO queues, this means that the *sum* of the queue sizes along its path reached the value $n-2i$; on the average the queue size per node was $(n-2i)/2i$. The minimum value for the last expression occurs for $i = h$, and it shows that at best the average queue size will be $O(n/\log_k n)$. In conclusion, increased capacity allows more messages to move upwards simultaneously but does not reduce queueing on their way down to the leaves.

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. Next we present an $(n + 2\log_k n - 2)$-step algorithm that eliminates the queues completely.

3.2.2. Eliminating the queues

We show here that for multinode broadcasting it is possible to eliminate the queues at every node at the expense of $2\log_k n - 2$ extra steps. The algorithm induces no contention between messages; nodes should only be capable of buffering one message from every incoming link. It operates as follows. Initially, every node in the k subtrees of the root node broadcasts its message *only* to the other $k-1$ subtrees. In the normal broadcast algorithm we discussed in Section 3.1, when a message is received by a routing node it is replicated towards all directions. Now, however, we insist that such a node sends it only to its parent; only if the message comes from the parent will it be broadcast to the children.

The k^{h-1} broadcastings from each subtree of the root do not start simultaneously but in different time steps to avoid contention. After all messages have been transferred, we are left with k subtrees which need to perform a multinode broadcasting within themselves. The algorithm proceeds recursively in the same manner until we are left with one-node subtrees. Iteratively, the algorithm can be stated as shown in Fig. 9.

During phase i the routing nodes involved are roots of trees with k^{h-i} leaves, and

```
1   For all i = 0, 1, ..., h − 1
        /* Phase i */
2       Do in parallel for all level h − i nodes
3           Broadcast the messages from each of the k subtrees to
            the other k − 1 subtrees;
```

Figure 9. *Multinode broadcasting algorithm that eliminates queues*

consequently each one receives k^{h-i} messages from the leaves below. In each step such a node receives k messages from its children. It is possible to exchange them among the children in $k-1$ steps as was shown in the proof of Theorem 7. Consequently, to avoid contention, there must be only k leaves transmitting every $k-1$ steps. This means that phase i of the algorithm requires time

$$T_i = (k-1)k^{h-i-1} + 2(h-i) - 1,$$

accounting for the fact that the first message(s) arrives in the routing node at step $h-i$ and that the last message(s) to leave this node needs $h-i$ steps to reach the appropriate leaves. In total, the algorithm requires time

$$T = \sum_{i=0}^{h-1} T_i = k^h + h^2 - 1. \tag{3}$$

Instead of executing the phases serially, it is advantageous to 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 since 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 $2h-2i-3$ steps overlapping, leading to a savings of

$$\sum_{i=0}^{h-2}(2h-2i-3) = h^2 - 2h + 1 \quad \text{steps.}$$

Using (3), we conclude that the total number of steps for the new algorithm is

$$T = k^h + 2h - 2 = n + 2\log_k n - 2.$$

```
1    For all i = 0 to h − 1
         /* Phase i */
2        Do in parallel for all level h − i nodes
3            Transfer all messages from each of the k subtrees
             to the other k − 1 subtrees;
```

Figure 10. *A total exchange algorithm with no contention*

3.3. Total exchange

Lower bounds for the total exchange problem can be derived by considering the messages that go through a level $h-1$ routing node. A particular node at this level will receive $k^{h-1}(n-k^{h-1}) = n^2(k-1)/k^2$ "upward" messages from the leaves of the subtree it roots; those messages will go through the other nodes in level $h-1$. If the branch capacities are c_h then only c_h messages can be transferred at a time from our node to the root of the tree, so that

$$TE(n) \geq n^2 \frac{k-1}{k^2 c_h} + 2\log_k n - 1. \tag{4}$$

A simple total exchange algorithm that works for any capacity pattern and induces no queueing can be derived from the multinode broadcasting algorithm of the previous section. Initially, the k subtrees of the root node exchange their $n(n-k^{h-1})$ messages meant for each other. Then, the k subtrees perform internally a total exchange in parallel. Iteratively, the algorithm can be stated as in Fig. 10.

During the ith phase a node at level $h-i$ has to pass $k^{h-i}(k^{h-i}-k^{h-i-1})$ messages over its k incident branches which have capacity c_{h-i}. This means that a maximum of kc_{h-i} messages can cross towards the node at a time. To avoid contention while maintaining maximum speed, exactly kc_{h_i} leaves should dispatch messages at a single step, and the messages should be appropriately chosen so that their destinations are distinct. We omit any implementation details since we are only interested in the time requirements of the algorithm. Phase i clearly requires

$$T_i = \left\lceil \frac{k^{h-i}(k^{h-i}-k^{h-i-1})}{kc_{h-i}} \right\rceil + 2(h-i) - 1,$$

where we also considered the delays for the first message to arrive at a level $h-i$ node

and for the last message leaving this node to arrive at the appropriate leaf. The total time needed for the algorithm is

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

Pipelining the phases, as in the case of multinode broadcasting, will save in total h^2-2h+1 steps, giving a final number of steps

$$T = \sum_{i=1}^{h} \left\lceil \frac{(k-1)k^{2i-2}}{c_i} \right\rceil + 2h - 1. \tag{5}$$

In particular, if the network is the simple k-ary tree ($c_i = 1$), the algorithm will take

$$T = (n^2-1)\frac{k-1}{k^2-1} + 2h - 1.$$

If the capacities follow the exponential rule, i.e. $c_i = k^{i-1}$, we obtain from (5)

$$T = k^h + 2h - 2 = n + 2h - 2.$$

It is seen from (4) that the time needed for our algorithm is slightly suboptimal by a small constant factor. For the exponential capacities case the bound of (4) is not tight since it predicts $TE(n) \geq n - n/k + 2h - 2$. It should be clear that $TE(n) \geq n$ (or $TE(n) \geq n+1$ if the tree is binary) is a tighter bound, since multinode broadcasting can be performed in at most as many steps as total exchange [2]. According to this last bound, the algorithm we presented above is suboptimal by $2\log_k n - 2$ steps. It is interesting to see whether the last bound is tight or not. In the next section we show that actually there exists a tighter lower bound for trees with exponential capacities.

3.3.1. A tighter bound for exponential capacities trees

We show here that for the case of trees with exponential capacities arrangement, a bound tighter than $TE(n) \geq n$ exists which guarantees that the total exchange algorithm we presented is very close to optimal (within $2\log_k \log_k n$ steps) for such trees.

If at any given step of a total exchange algorithm a leaf did not receive a message we say that a *hole* occurred in its receptions. Notice that since each leaf must receive $n-1$ messages, if the average number of holes per leaf node was w then the algorithm takes time $T \geq n - 1 + w$; the equality holds if all leaves had the same number of holes (w).

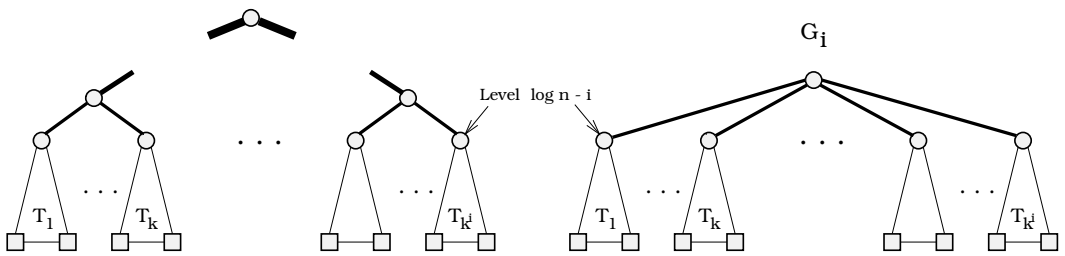

Figure 11. *Collapsing the last i levels of the tree gives graph G_i*

Consider the graph G_i which is derived from our original tree by collapsing its last i levels into a single vertex (see Fig. 11). We then have a collection of k^i subtrees $T_1, T_2, \ldots, T_{k^i}$ each having n/k^i leaves and their roots have a single common parent. The height of the new graph is $h - i + 1$. In addition we introduce new edges between every pair of leaves within each of the k^i subtrees so that messages within a subtree can be transferred in a single step. In order to avoid those edges begin used simultaneously, we also impose the restriction that *only one message may be received by a leaf at any time* as in the original tree. It is seen that any total exchange algorithm for the original tree is a total exchange algorithm for G_i (but not vice versa) — the message transfers within the last i levels of our tree 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 our original tree.

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_1 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 leaves of T_0. In fact, due to the $2(h - i + 1)$-step delay external messages suffer before reaching their destinations, no messages will have arrived from any other subtree before time $t + 2(h - i + 1)$. Consequently the number of holes in T_0 between times t and $t + 2(h - i + 1)$ 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 leaf in G_i. Consider any total exchange algorithm for G_i and partition time in $2(h - i + 1)$-step periods. If there are p such periods (plus possibly a last, shorter one), the algorithm needs

time $T \geq 2p(h-i+1)$. Assume that at the sth period there were $w_s^{(j)}$ holes in T_j. Hence in the first period the total number of holes was

$$w_1 = w_1^{(1)} + w_1^{(2)} + \cdots + w_1^{(k^i)}.$$

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

$$w_2 = w_2^{(1)} + w_2^{(2)} + \cdots + w_2^{(k^i)}$$

and as a result the total number of external messages sent during the second period was *at most* equal to $w_1 + w_2$. In general, during the jth period the maximum number of external messages was $w_1 + w_2 + \cdots + w_j$. Hence after p periods the total number of external messages observed was at most

$$\begin{aligned} \sum_{j=1}^{p}(w_1 + w_2 + \cdots + w_j) &= \sum_{j=1}^{p}(p-j+1)w_j \\ &= p\sum_{j=1}^{p} w_j - \sum_{j=1}^{p}(j-1)w_j \\ &= pW - Q \end{aligned}$$

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

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

$$pW \geq n(n-n/k^i).$$

Notice that the average number of holes per node is $w = W/n$ and since $T \geq 2p(h-i+1)$, the last inequality yields

$$wT \geq 2n(1-\frac{1}{k^i})(h-i+1). \tag{6}$$

From our earlier discussion, if w is the average number of holes per leaf then

$$T \geq n-1+w, \tag{7}$$

the equality holding if all leaves have exactly w holes. Inequalities (6) and (7) can be combined to determine the minimum value of T. Since (6) holds for any value of i, we can

choose i so that we obtain the best bound on T. For example if $i = 1$, it can be seen that T cannot be less than $n + h - 1$. A tighter bound can be had if we select $i = \log_k h + 1$; we obtain from (6):

$$\begin{aligned} wT &\geq 2n(1 - \frac{1}{kh})(h - \log_k h) \\ &\geq 2n(1 - \frac{1}{2h})(h - \log_k h) \\ &= 2nh - 2n\log_k h - n + n\frac{\log_k h}{h} \\ &\geq 2nh - 2n\log_k h - n. \end{aligned} \tag{8}$$

If $w < q = 2h - 2\log_k h - 1$, then (8) gives

$$\begin{aligned} T &\geq \frac{2nh - 2n\log_k h - n}{2h - 2\log_k h - 2} \\ &= n + \frac{n}{2h - 2\log_k h - 2}. \end{aligned}$$

Noting that $h = \log_k n$, and after a bit of algebra we obtain $T > n - 1 + q$. On the other hand, if $w > q$ then from (7) $T \geq n - 1 + w > n - 1 + q$. Consequently, the minimum time can be had only for $w = q$ and in that case

$$T = n - 1 + q = n + 2h - 2\log_k h - 2 \text{ steps.} \tag{9}$$

Theorem 8 *An optimal total exchange algorithm for exponential capacity trees needs at least* $n + 2\log_k n - 2\log_k \log_k n - 2$ *steps.*

Proof. This is an immediate consequence of the facts that a total exchange algorithm for the trees of interest is a total exchange algorithm for G_i and that for G_i a lower bound is given by (9). ■

3.4. Discussion

The results for the multiport model are summarized in Table 2. The preceding analyses were based on the assumption that $c_1 = 1$. It is expected that some of the problems will require less steps if the first level branches consisted of more than one links; it would be interesting to see what is the exact effect of c_1 in this case.

For the total exchange problem, the bound given in Table 2 is general but provably not tight in some cases. What we have shown here is that for complete k-ary trees, total

Problem	Time
Broadcasting	$2\log_k n - 1$
Scattering/Gathering	n (if $k \geq 3$) , $n+1$ (if $k = 2$)
Multinode Broadcasting	n (if $k \geq 3$) , $n+1$ (if $k = 2$)
Total Exchange*	$\geq n^2(k-1)/(k^2 c_h) + 2\log_k n - 1$

*see text

Table 2. *Time requirements for communications under the multiport model*

exchange can be performed in

$$n^2\frac{k-1}{k^2} + 2\log_k n - 1 \leq TE(n) \leq (n^2-1)\frac{k-1}{k^2-1} + 2\log_k n - 1,$$

and for trees with exponentially growing capacities,

$$n + 2\log_k n - 2\log_k \log_k n - 2 \leq TE(n) \leq n + 2\log_k n - 2.$$

4. Conclusion

We have studied the implementation and performance of communication operations in k-ary complete trees where the processing nodes are confined to the leaf level, considering both the single port and the multiport models. The results can be easily generalized to the case where nodes in different levels can have a different number of children, that is level i nodes having k_i children, where $n = k_1 k_2 \cdots k_h$. Under the multiport model, branch capacities as used in fat tree networks have a beneficial role in high-volume communications such as total exchange. If the processing nodes are connected through a single link with the routing network, capacities are of no importance in the rest of the communication problems we studied.

There are still 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. The second algorithm we presented for this problem avoids contention but requires $2\log_k n - 2$ more steps. It is still an open question whether this is optimal or contention can be avoided in less steps. Another issue for consideration is total exchange under the multiport model. The algorithm we presented is close to optimal especially in the case of exponential capacities, where we showed that it is within $2\log_k \log_k n$ steps from the lower bound. Improved

bounds though for the total exchange problem need to be found as the straightforward one does not seem to be tight.

References

[1] Bertsekas, D. P., Ozveren, C., Stamoulis, G. D., Tseng, P., and Tsitsiklis, J. N. Optimal communication algorithms for hypercubes. *J. Parallel Distrib. Comput.* **11** (1991), 263–275.

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

[3] Bhatt, S. N., Pucci, G., Ranade, A., and Rosenberg, A. L. Scattering and gathering messages in networks of processors. *IEEE Trans. Comput.* **42**, 8 (Aug. 1993), 938–949.

[4] Fraigniaud, P., and Lazard, E. Methods and problems of communication in usual networks. *Discrete Appl. Math.* **53** (1994), 79–133.

[5] Gannon, D. B., and van Rosendale, J. On the impact of communication complexity on the design of parallel numerical algorithms. *IEEE Trans. Comput.* **C-33**, 12 (Dec. 1984), 1180–1194.

[6] Hedetniemi, S. M., Hedetniemi, S. T., and Liestman, A. L. A survey of gossiping and broadcasting in communication networks. *Networks* **18** (1988), 319–349.

[7] Johnsson, S. L. Communication efficient basic linear algebra computations on hypercube architectures. *J. Parallel Distrib. Comput.* **4** (1987), 133–172.

[8] Johnsson, S. L., and Ho, C. - T. Optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Comput.* **38**, 9 (1989), 1249–1268.

[9] Leiserson, C. E. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* **C-34**, 10 (Oct. 1985), 892–901.

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

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

[12] Saad, Y., and Schultz, M. H. Data communications in hypercubes. *J. Parallel Distrib. Comput.* **6** (1989), 115–135.

[13] Saad, Y., and Schultz, M. H. Data communications in parallel architectures. *Parallel Comput.* **11** (1989), 131–150.

[14] Topkins, D. M. Concurrent broadcast for information dissemination. *IEEE Trans. Softw. Eng.* **11**, 10 (Oct. 1985), 1107–1112.