Optimal Total Exchange in Linear Arrays and Rings*

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

In this paper we consider the problem of total exchange (or multi-scattering) in the context of linear arrays and rings. Such a communication mode occurs when each node has a distinct message to send to every other node in the network. The problem has been studied extensively, although no optimal algorithm has been proposed for the two networks of interest. We present simple algorithms for the two topologies and prove their optimality.

1 Introduction

Linear arrays and rings are among the simplest and cheapest types of interconnection networks for multiprocessors. Despite their simplicity they can be quite powerful and cost-effective for the solution of various problems. They also form the basis for such multidimensional structures as meshes and tori. They have been extensively studied by researchers; the interested reader is referred to F.T. Leighton's text [7].

Communications between processors (or *nodes*) of any interconnection network form the basis for higher level task interactions and for parallel algorithm design. There have been identified five important modes of communication, namely

- *broadcasting* where a single message from a specific node has to be given to the other nodes in the network
- *multinode broadcasting* which involves simultaneous broadcasting from every node
- *gathering* where a separate message from each node has to reach a specific node
- *scattering*, the dual problem of gathering, where a certain node sends a distinct message to every other node
- *total exchange*, a multiple scattering operation, where every node has a distinct message to send to every other node.

The above terminology follows Bertsekas et al [2, 1]. Multinode broadcasting is also known as 'gossiping' or 'all-to-all communication' [10]. Scattering is also known as 'one-to-all personalized communication' [6] and total exchange is termed 'multi-scattering' in [8].

Communication algorithms for various networks have appeared in [8, 9, 6, 2, 3, 1]. In particular, references [8, 6, 1] deal with hypercubes. Other networks have been considered in [9, 2], while in [3] the scattering algorithm for rings proposed in [9] was proven to be optimal. For all but the total exchange problem, optimal algorithms for linear arrays and rings can be found in [2].

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

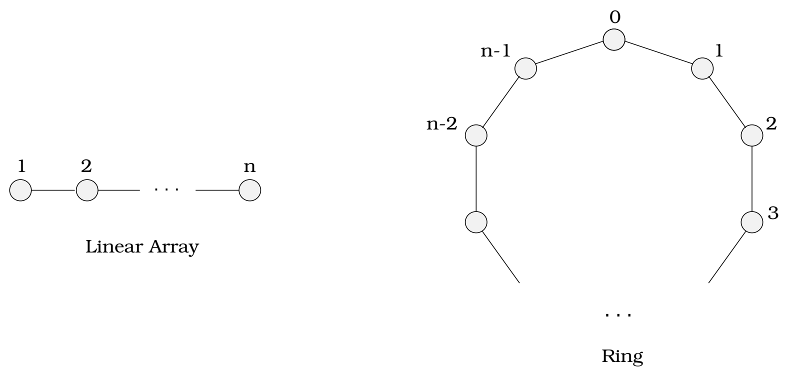

Figure 1: Linear array and ring

Efficient communication is essential for a variety of parallel algorithms. Its impact in basic and numerical linear algebra algorithms has been realized in [4, 5, 2]. In particular, total exchange is, among other things, associated with matrix transposition. Consider for example n nodes where each node i stores the ith row of an $n \times n$ matrix. In transposing the matrix, the ith node has to receive one element of the matrix from every other node in order to form the ith column. We recognize this as the total exchange problem.

This work presents optimal total exchange algorithms for linear arrays and rings and is organized as follows: section 2 introduces the basic assumptions and the lower bounds for the time complexity of total exchange algorithms in the two networks; section 3 gives the algorithm for linear arrays and the proof of its optimality; section 4 deals with rings and includes optimal algorithms for the case of an odd or an even number of nodes.

2 Model and lower bound

A linear array and a ring with n nodes each are shown in Fig. 1. In the linear array nodes are labeled 1 to n from left to right and node i is connected through a *bidirectional* link with node $i+1$, $1 \leq i \leq n-1$. In the ring, nodes are labeled 0 to $n-1$ in a clockwise fashion and node i is adjacent to nodes $i \pm 1$ mod n, $0 \leq i \leq n-1$.

The networks are assumed to be store-and-forward switched so that there is a packet (or message) queue associated with each node. For our purposes the terms packet and message are considered equivalent in that each message consists of a single packet. Following the model of Bertsekas et al [1] we take as a unit of time (or *step*) the time required to transmit a message over an incident link. The multi-port availability assumption is also made whereby every node can utilize all its input and output links simultaneously; each node can receive messages from all its neighbors and send messages to them in a single time unit. The assumption is not crucial; the algorithms can be easily transformed for the case of half-duplex links with a slowdown factor of two.

In the total exchange problem node i has $n-1$ distinct messages to send, one for each of the other nodes in the network. The lower bound for the time complexity of total exchange algorithms can be obtained as follows. Consider the linear array and form a bisection, keeping the first $\lfloor n/2 \rfloor$ nodes in the first half and the rest in the other half. Any total exchange algorithm has to pass the messages from the first half to the second half. There are in total $\lfloor n/2 \rfloor (n - \lfloor n/2 \rfloor)$ such messages and there is only one link that separates the two halves, namely the link from node $\lfloor n/2 \rfloor$ to node $\lfloor n/2 \rfloor + 1$. At least $\lfloor n/2 \rfloor (n - \lfloor n/2 \rfloor)$ steps are hence needed to complete this movement, so that

$$T_{LA}(\text{ total exchange }) \geq T_{opt(LA)} = \left\lceil \frac{n^2-1}{4} \right\rceil . \tag{1}$$

The same reasoning applied to an n-node ring gives

$$T_R(\text{ total exchange }) \geq T_{opt(R)} = \left\lceil \frac{n^2-1}{8} \right\rceil. \qquad (2)$$

For linear arrays, a simple total exchange algorithm was suggested in [2] consisting of n consecutive scatterings. An optimal scattering from node i can be completed in max$\{i-1, n-i\}$ steps [2]. After some algebra we can see that the time needed for this total exchange scheme is $3T_{opt} - \lfloor n/2 \rfloor$ which is about three times slower than the optimum. For rings, rotation of the messages was suggested in [9], dropping one message at each stop. We show that this algorithm is optimal only for odd n while a modification is required to achieve the optimum for even n.

3 Total exchange in linear arrays

We first observe that messages that travel towards the right end of the array do not interfere with messages traveling towards its left end since they use links of opposite direction. Contention occurs only between messages traveling in the same direction. Consequently the problem can be thought as two subproblems, each involving message transmissions in one direction, and they can be solved simultaneously. Thus, without loss of generality, we can concentrate only on the rightward messages.

Initially, node i, $1 \leq i \leq n-1$, has $n-i$ messages to send to its right. In addition, there are $i-1$ messages in total meant for node i, residing at nodes $1, 2, \ldots, i-1$. When a message is received by its destination, it gets consumed immediately, i.e. it does not join the node's queue.

The algorithm we propose is quite simple:

do in parallel for all nodes $i = 1, 2, \ldots, n-1$, all the time:
if node i has any messages in its queue, it selects the message that has to travel the largest distance and sends it to its right. Ties are broken arbitrarily.

That is, we follow a *furthest-first* type of scheduling. The way ties are broken is clearly immaterial for the simple reason that the time the last message, out of the tied messages, leaves node i is independent of the identity of messages; it depends only on the number of tied messages. Notice also that there is no synchronization required among nodes, although for our purposes we will assume that all nodes start at the same time and are synchronized by a common clock. An example for the case of $n = 6$ nodes is given in Fig. 2.

For illustration purposes, we view the evolution of the algorithm as a sequence of logical phases. Phase i starts at the time node $i-1$ gets emptied, i.e. has no more messages to send. Our basic argument is that by the time node i gets emptied, i.e. by the end of phase i, all messages addressed to node $n-i+1$ have reached their destination. By the end of every phase the leftmost and the rightmost of the active nodes become idle, reducing the number of active nodes by two.

As mentioned above, in the case of ties (two or more messages in the same node have to travel the same distance) all tie-breaking protocols yield the same running time. We can thus concentrate in a specific protocol without loss of generality. This protocol breaks the ties in favor of the message that has *already* traveled the furthest distance, or in other words, in favor of the source with the smallest address.

Let $m_k(i)$ stand for the message of node k destined for node i. The message has to go through all nodes j, $1 \leq k \leq j \leq i-1$, before it reaches node i. Let $t^{(j)}(m_k(i))$ be the time $m_k(i)$ leaves node j. Then $t^{(j-1)}(m_k(i))$ is the time $m_k(i)$ arrives at node j.

For the special case of messages destined to node n observe that $m_j(n)$ leaves node j at time 1, for all $j \leq n-1$ and it moves towards n without any delays. Specifically, message $m_1(n)$ is the last message to reach node n and $t^{(j)}(m_1(n)) = j$. We have the following lemma.

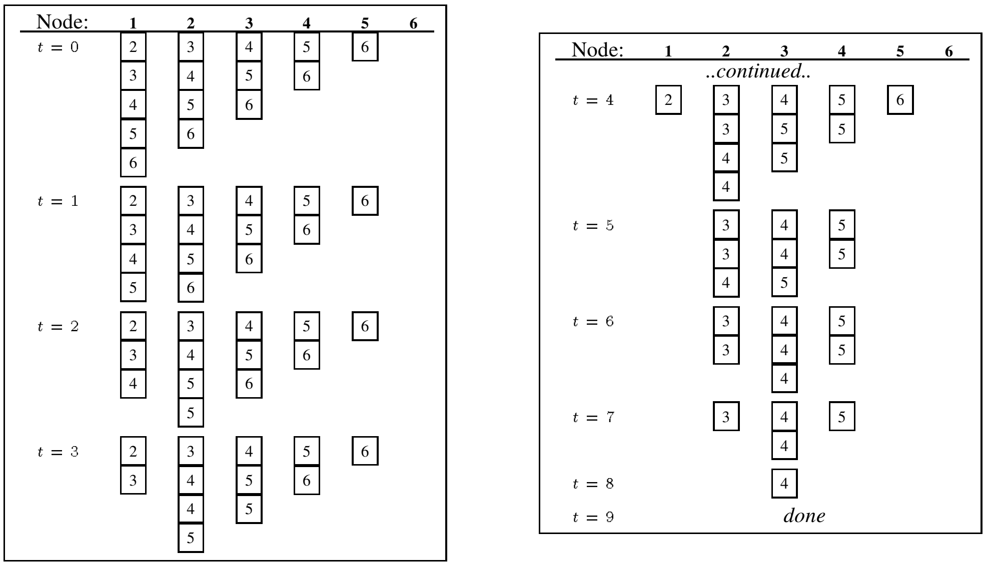

Figure 2: A 6-node linear array example (only the destinations of messages shown)

Lemma 1 *If* $1 \leq j < i \leq n-1$, *then*

$$t^{(j)}(m_1(i)) = \begin{cases} t^{(j)}(m_j(i+1)) + 1, & i < n-1 \\ j+1, & i = n-1 \end{cases} \tag{3}$$

$$t^{(j)}(m_{k+1}(i)) = t^{(j)}(m_k(i)) + 1, \;\; 1 \leq k \leq j-1 \tag{4}$$

Proof. We prove the lemma using double induction on i and j.

Consider first the case of $i = n-1$. Messages destined for node n are not delayed anywhere, so that $m_1(n)$ leaves node j at time j. Message $m_1(n-1)$ follows behind $m_1(n)$ and because of our scheduling discipline, it is not delayed anywhere, as well. Hence it leaves node j at time $j+1$, proving thus (3). Also, up to that time, no message $m_j(x)$ has left node j except for $x = n$. We prove (4) by induction on j. Clearly, $m_2(n-1)$ leaves node 2 immediately after $m_1(n-1)$. If (4) also holds for $j = 2, 3, \ldots, J-1$, then

$$t^{(J-1)}(m_{k+1}(n-1)) = t^{(J-1)}(m_k(n-1)) + 1.$$

Consequently, messages from nodes 1, 2, …, $J-1$ arrive at node J one after the other starting at time $J+1$, and since $m_J(n-1)$ has not left yet (as of time $J+1$), they will also be scheduled to leave node j one after the other, proving thus (4).

Now assume that (3) and (4) hold for all $i = n-1, n-2, \ldots, I+1$. We will show that they also hold for $i = I$, using separate inductions on j.

For $j = 1$, (3) trivially holds. If it also holds for all $j = 1, 2, \ldots, J-1$, then

$$t^{(J-1)}(m_1(I)) = t^{(J-1)}(m_{J-1}(I+1)) + 1,$$

that is, $m_1(I)$ arrives at node J right after $m_{J-1}(I+1)$. But from induction hypothesis for $i = I+1$, message $m_J(I+1)$ has not left node J yet since from (4) is has to leave after $m_{J-1}(I+1)$. Thus $m_1(I)$ must be scheduled to leave immediately after $m_J(I+1)$, concluding the induction on j.

Finally, a similar induction on j can be used to prove (4) for $i = I$. For, since $m_2(I+1)$ leaves node 2 one time unit after $m_1(I+1)$, message $m_1(I)$ has arrived on time at node 2, proving (4) for $j = 2$. Assuming that the equation holds for $j = 1, 2, \ldots, J-1$, we get

$$t^{(J-1)}(m_{k+1}(I)) = t^{(J-1)}(m_k(I)) + 1.$$

Thus $m_{k+1}(I)$ leaves node $J-1$ right after $m_k(I)$, i.e. it arrives at node J just after $m_k(I)$ does. Because of our scheduling discipline, it will leave node J immediately after $m_k(I)$. ■

Equations (3) and (4) state that there is no idle time between message dispatches from node j, for all $j < i$. In particular, (4) guarantees that all messages destined for node i have arrived at node j at appropriate times, to be scheduled one after the other. The last message, hence, to leave j is message $m_j(i)$ which leaves $j-1$ time units after $m_1(i)$. Equation (3) serializes the phases; only after *all* messages for node $i+1$ have left node j, will messages for node i start departing from j.

Using (3) and (4) repeatedly one can find the exact expression for $t^{(j)}(m_k(i))$ for any i, j, k, $1 \leq k \leq j < i \leq n-1$:

$$t^{(j)}(m_k(i)) = jn - ji + k. \tag{5}$$

Lemma 2 *Phase $j \leq n/2$ ends at time $T_j = jn - j^2$ at which time node j sends its last message and node $n-j+1$ receives its last message.*

Proof. Phase j ends when node j gets emptied. This occurs at time $T_j = t^{(j)}(m_j(j+1))$ since from lemma 1, $m_j(j+1)$ is the last message to leave node j. Equation (5) gives

$$T_j = t^{(j)}(m_j(j+1)) = jn - j^2.$$

On the other hand, node $n-j+1 \leq n-1$ receives its last message at $t^{(n-j)}(m_{n-j}(n-j+1))$. Working exactly as above, we get

$$t^{(n-j)}(m_{n-j}(n-j+1)) = jn - j^2 = T_j.$$

The lemma also holds for the case of node n since it receives its last message ($m_1(n)$) at time $n-1$, at the end of phase 1. ■

Theorem 1 *The algorithm terminates at time $T_{opt(LA)}$.*

Proof. After each phase the number of active nodes is reduced by two as lemma 2 shows. The algorithm terminates at time T, after phase $n/2$ or phase $(n-1)/2$ depending on whether n is even or odd. Using lemma 2, in the first case $T = T_{n/2} = n^2/4$ and in the second $T = T_{(n-1)/2} = (n^2-1)/4$ or, for any n, $T = \lceil (n^2-1)/4 \rceil = T_{opt(LA)}$. ■

4 Total exchange in rings

In the case of rings it is beneficial, in terms of speed, to send each message to its destination through the shortest of the two paths available. As in the case of linear arrays we only concentrate in one direction, specifically, clockwise as counterclockwise messages do not interfere; except now every node has at most $\lfloor n/2 \rfloor$ messages to send. Because of some irregularities, the cases of odd n and even n are discussed separately. For our purposes, we use the following notation:

- If x and y are integers, then $x \oplus y \stackrel{\text{def}}{=} x + y \bmod n$ and $x \ominus y \stackrel{\text{def}}{=} x - y \bmod n$.
- If $m_j(i)$ is the message of node j destined for node i, then $m_j(i) \oplus x \stackrel{\text{def}}{=} m_{j\oplus x}(i \oplus x)$.
- If Q is a set of messages, then $Q \oplus x \stackrel{\text{def}}{=} \{m \oplus x \mid m \in Q\}$.

4.1 Odd number of nodes

In the case of odd n we show that any "reasonable" algorithm for the total exchange problem is optimal. First, notice that every node has $(n-1)/2$ messages to send both clockwise and counterclockwise. Concentrating on the clockwise direction, node i has messages for nodes $i \oplus 1$, $i \oplus 2, \ldots, i \oplus (n-1)/2$. Consequently, the messages of node i have to travel a total distance of

$$D_i = \sum_{j=1}^{(n-1)/2} j = \frac{n^2-1}{8}.$$

For the whole ring the total distance is

$$D = \sum_{i=0}^{n-1} D_i = \frac{n(n^2-1)}{8}.$$

To achieve the lower bound of (2), which in the case of odd n is exactly $(n^2-1)/8$, we need a decrease of $D/T_{opt(R)} = n$ in total distance D at *every* step of the algorithm. Since a message transferred to the next node decreases D by 1, we see that a decrease of n in D is possible if and only if all nodes are busy, all the time. Any algorithm achieving this is thus optimal.

Let $Q_i(t)$ be the message queue of node i at time $t \geq 0$. Assume that the *same* algorithm f runs at every node, which given $Q_i(t)$ decides that the next message to leave node i is message $f(Q_i(t))$.

Definition 1 An algorithm f is *node-invariant* if for non-empty message queue it always schedules some message to leave next, and $f(Q_0(t) \oplus i) = f(Q_0(t)) \oplus i$.

Lemma 3 *Any node-invariant algorithm guarantees that, for all* $t \geq 0$, $Q_i(t) = Q_0(t) \oplus i$.

Proof. Notice that initially we have $Q_i(0) = Q_0(0) \oplus i$. If we assume as an induction hypothesis that $Q_i(t) = Q_0(t) \oplus i$ then we get

$$Q_i(t+1) = Q_i(t) \cup \{f(Q_{i\ominus 1}(t))\} \setminus \{f(Q_i(t))\},$$

where $\setminus$ is the set-theoretic difference. The above can be written as

$$\begin{aligned} Q_i(t+1) &= Q_0(t) \oplus i \cup \{f(Q_0(t)) \oplus i \ominus 1\} \setminus \{f(Q_0(t) \oplus i)\} \\ &= \Big\{Q_0(t) \cup \{f(Q_{0\ominus 1}(t))\} \setminus \{f(Q_0(t))\}\Big\} \oplus i \\ &= Q_0(i+1) \oplus i, \end{aligned}$$

proving the lemma. ∎

Theorem 2 *Any node-invariant algorithm is an optimal total exchange algorithm for odd rings.*

Proof. Consider the first time that some node was observed to be idle, i.e. its queue was empty, under some node-invariant algorithm. Because the algorithm is node-invariant, lemma 3 applies to show that all nodes have the same queue size, i.e. they are all empty. Consequently all nodes become idle at the same time; based on our earlier discussion, the algorithm is optimal. ∎

Most reasonable algorithms are node-invariant, e.g. the simple furthest-first and closest-first types of scheduling with any consistent (across the ring) tie-breaking protocol. The same holds for the *message-shift* algorithm where at every node messages join and leave the queue in a FIFO manner. In effect this algorithm rotates the messages around the ring as was initially proposed in [9]. An example is shown in Fig. 3(a). It should also be noted that the FIFO nature of message-shift allows for the most effective implementation; in contrast with the furthest- or closest-first disciplines, the nodes do not have to sort their message queues every time they receive a new message.

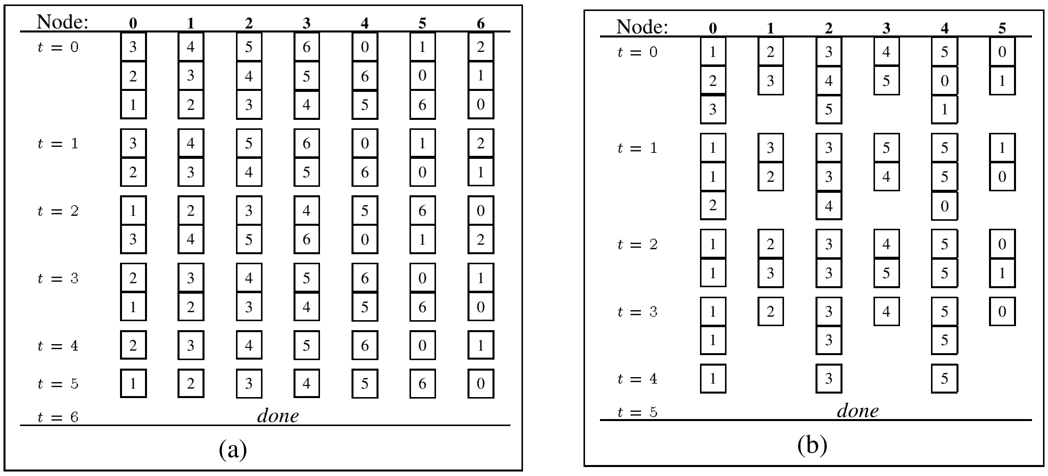

Figure 3: Examples in a 7-node ring (a) and a 6-node ring (b)

4.2 Even number of nodes

The peculiarity of this case arises from the imbalance between clockwise and counterclockwise messages. Specifically, the furthest node from node i is node $i \oplus n/2$, so that if $m_i(i \oplus n/2)$ is scheduled for the clockwise direction, then in the counterclockwise direction there is one less message to be sent. If all nodes send $n/2$ messages in the clockwise direction, then it can be shown that any node-invariant algorithm is suboptimal by an additive factor of $O(n)$. It is thus necessary to have some of the nodes send $n/2 - 1$ messages clockwise.

We show next that a slightly modified message-shift algorithm is optimal if for all even i, $0 \leq i \leq n-2$, node i sends $n/2$ messages and node $i \oplus 1$ sends $n/2 - 1$ messages clockwise. The modification requires that in the first steps, where all messages $m_i(j)$ leave node i, the queue of node i is organized in a furthest-first order. After that we proceed in the normal FIFO manner. That is, if messages leave the queue from the right end and join the queue at the left end, then

$$Q_i(0) = \{m_i(i \oplus 1), m_i(i \oplus 2), \ldots, m_i(i \oplus n/2)\}.$$

For node $i \oplus 1$ the analogous ordering should be had. An example is shown in Fig. 3(b).

Concentrate on nodes i and $i \oplus 1$, where i is even. Initially,

$$\begin{aligned} Q_i(0) &= \{m_i(i \oplus k) \mid k = 1, 2, \ldots, n/2\} \\ Q_{i\oplus 1}(0) &= \{m_{i\oplus 1}(i \oplus 1 \oplus k) \mid k = 1, 2, \ldots, n/2 - 1\}. \end{aligned}$$

A simple induction shows that if at time t

$$\begin{aligned} Q_i(t) &= \{m_{i\ominus 2j}(i \ominus 2j \oplus k) \mid k = 2j+1, \ldots, n/2\} \\ Q_{i\oplus 1}(t) &= \{m_{i\ominus(2j-1)}(i \ominus (2j-1) \oplus k) \mid k = 2j+1, \ldots, n/2 - 1\} \end{aligned}$$

then after $2|Q_{i\oplus 1}(t)|$ steps, i.e. after $T_j = n - 4j - 2$ steps,

$$\begin{aligned} Q_i(t + T_j) &= \{m_{i\ominus(2j+2)}(i \ominus (2j+2) \oplus k) \mid k = 2j+3, \ldots, n/2\} \\ Q_{i\oplus 1}(t + T_j) &= \{m_{i\ominus(2j+1)}(i \ominus (2j+1) \oplus k) \mid k = 2j+3, \ldots, n/2 - 1\}. \end{aligned}$$

Theorem 3 *The algorithm terminates at time $T_{opt(R)}$.*

Proof. Notice that at time t, $|Q_i(t)| = n/2 - 2j$. If $n/2$ is even then for $j = n/4$, $|Q_i(t)| = 0$; otherwise, for $j = (n-2)/4$, $|Q_i(t)| = 1$ and all queues become empty at the next step (since the last message in $Q_i(t)$ will be destined for node $i \oplus 1$). We obtain

$$t = \sum_{k=0}^{j-1} T_k = \sum_{k=0}^{j-1} (n - 4k - 2) = jn - 2j^2.$$

If $n/2$ is even ($j = n/4$), we finish at time $t = n^2/8$ otherwise ($j = (n-2)/4$) we finish at time $t = (n^2+4)/8$; in both cases t is equal to $\lceil (n^2-1)/8 \rceil = T_{opt(R)}$. ■

5 Conclusion

We presented total exchange (or multi-scattering) algorithms for linear arrays and rings. For linear arrays we showed that the furthest-first message scheduling discipline yields the optimum performance. In rings with an odd number of nodes any "consistent" distributed algorithm was shown to be optimal. This does not hold when the number of nodes in the ring is even. In that case a special arrangement of the message-shift algorithm was found to achieve the lower bound.

References

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

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

[3] P. Fraigniaud, S. Miguet and Y. Robert, "Scattering on a ring of processors," *Parallel Comput.*, Vol. 13, No. 3, pp. 377–383, 1990.

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

[7] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Diego, CA: Morgan Kaufmann, 1992.

[8] Y. Saad and M.H. Schultz, "Data communications in hypercubes," *J. Parallel Distrib. Comput.*, Vol. 6, pp. 115–135, 1989.

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

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