ONE-TO-ALL PERSONALIZED COMMUNICATION IN TORUS NETWORKS

Weizhen Mao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187, USA
wm@cs.wm.edu

Jie Chen and William Watson, III
The High Performance Computing Group
Jefferson Lab
Newport News, VA 23606, USA
{chen, watson}@jlab.org

ABSTRACT
Given a multicomputer system of parallel processors connected in a torus network, the one-to-all personalized communication is to send from the root processor unique data to each of the other processors in the network. Under the assumptions of same-size data to each processor, store-and-forward routing, and all-port processors, we formulate the one-to-all personalized communication problem as an optimization problem with the goal to minimize the total elapsed time (measured in the number of time steps) for all data to reach their respective destinations. We design an optimal algorithm based on partitioning the torus network into disjoint subnetworks. We also present a heuristic algorithm based on a greedy strategy. We implement the algorithms on two Linux clusters with Gigabit Ethernet torus connection, currently in use at the Jefferson National Lab and configured as a 2-dimensional 8×8 torus and a 3-dimensional $4 \times 8 \times 8$ torus, respectively. We analyze the performance of the algorithms using data collected in experiments.

KEY WORDS
One-to-all personalized communication, scatter and gather, parallel computing, torus network, algorithm, optimization.

1 Introduction

A multicomputer is a multiple processor system with distributed memory. It contains parallel running processors connected in a network topology by message passing links. To run a parallel job on a multicomputer involving all processors, inter-processor communication is often a frequent issue to deal with. For examples, a certain processor needs to exchange data with another processor (i.e., one-to-one communication), input data which originally reside at a root processor need to be moved to other processors (i.e., one-to-all communication), or results obtained by each processor need to be sent to every other processor (i.e., all-to-all communication). The communication between processors is often the main obstacle to increasing the performance of a multicomputer since it is not only dependent on the network topology (e.g., how many links are available at a processor to send or receive data to or from other processors) but also restricted by the capability of processors (e.g., how many simultaneous activities of sending and receiving a processor can engage in). Thus, it is commonly agreed that inter-processor communication is an important bottleneck.

In general, an inter-processor communication problem can be formulated by specifying the following parametric factors:

- The network topology of the multicomputer;
- The communication pattern of the application running on the multicomputer;
- The routing method for sending data from source to destination; and
- The number of ports available at each processor in the multicomputer.

The network topology is how the processors are connected in a multicomputer. Conventionally , such a network is viewed as a graph, where each node presents a processor and each edge represents a link. The multi-dimensional torus (which is a multi-dimensional grid with wrap-around edges in all dimensions) is the most common topology used. Communication patterns include one-to-one transfer (a single node sending its own data to another node), one-to-all broadcast (a single node sending its own data to all other nodes), one-to-all personalized communication (a single node sending unique data to each of other nodes), all-to-all broadcast (each node sending its own data to all other nodes), and all-to-all personalized communication (each node sending unique data to each of other nodes). Note that personalized communication is also called scattering for the one-to-all pattern and gathering for the all-to-one pattern. The most commonly used routing method is the store-and-forward method, where data move from source to destination one hop at a time, and at each intermediate node, the data are stored before starting the next hop, although wormhole routing and circuit-switched routing are also used. The number of ports that a node can monitor simultaneously gives an upper bound to the number of sending and receiving activities that the node can engage in simultaneously. A node can be of single-port, multi-port, or all-port (with the number of ports equal to the number of edges adjacent to the node, which is $2D$ in a D-dimensional torus).

A few years ago, two production Linux clusters (multicomputers) with Gigabit Ethernet torus connection of

sizes 8×8 and $4 \times 8 \times 8$ were deployed at the Jefferson National Lab (a research lab under the Department of Energy). The primary mission of the clusters is to carry out Lattice Chromodynamics (LQCD) calculations to study strong interactions among quarks. A typical LQCD calculation involves all nodes, which in our case are connected into a 2-dimensional or 3-dimensional torus. Each node needs its own input data to carry out its specific calculation. However, the data are initially stored at the root node (any specific processor in the cluster). Before the parallel calculation begins at each node, a major preparation step is to dispatch in the store-and-forward fashion all input data from the root node to the nodes where they belong to. This is actually the one-to-all personalized communication (scattering) problem. Since the input data needed by each node are huge in LQCD calculations, they have to be broken into small-size messages to be sent from the root node. On average, an LQCD calculation involves sending messages (small-size data) from the root node to the messages' respective destinations a total of 250,000 times. This makes it necessary and beneficial to design optimal or near-optimal algorithms to handle the one-to-all personalized communication problem.

This paper is organized as follows. In Section 2 we give a precise definition of our one-to-all personalized communication problem on torus networks of all-port processors with store-and-forward routing. We also give a brief discussion of related work. In Section 3, we discuss our algorithms, OPT and SDF. For the OPT algorithm, we also prove its optimality. In Section 4, we present some data collected in experiments conducted on two clusters available at the Jefferson National Lab to evaluate the average performance of our algorithms. Finally, we make conclusions in Section 5.

2 Problem Definition and Related Work

The communication problem that arises in LQCD calculation uses a multi-dimensional torus network of all-port processors and the communication pattern is one-to-all scattering by store-and-forward routing. Assume that in the torus there are p nodes (processors), indexed by $0, \ldots, p-1$. Obviously, the number of messages is the same as the number of nodes, p. Also assume that the messages are named according to the destinations where they will later be sent, i.e., $0, 1, \ldots, p-1$, where message i is destined to node i. Initially, all messages reside at the root node 0. Assume that all messages are of the same size and will take one time step to traverse a link. The goal of our one-to-all personalized communication problem is to scatter, in the store-and-forward fashion, all messages from the root node to their respective destinations in the minimum number of time steps (hops). At the beginning of the communication, message i is at the root node 0, but at the end of the communication message i will be at node i, for $i = 0, 1, \ldots, p-1$. We should mention that once the calculation at each processor is completed, the result may need to be sent back to the root node. This follows the reversed communication pattern of scattering. The resulting all-to-one personalized communication problem is called gathering and can be solved by the reverse of any scattering algorithm. For this reason, we will focus of one-to-all scattering in this paper.

A general discussion on communication problems in multicomputers, including both one-to-all and all-to-all patterns of broadcasting and personalized communication, can be found in [1]. Focusing on the store-and-forward routing method and personalized communication, we review some of the results relevant to our work. Being perhaps the most difficult among all, the all-to-all personalized communication pattern has received the most attention from the research community. Optimal and approximation algorithms for all-to-all personalized communication have been proposed for various network topologies, such as hypercubes [2], 2-dimensional tori [3], fully-connected networks [4], multistage interconnection networks [5], and wavelength-division-multiplexed (WDM) rings [6]. All-to-all communication (including broadcasting and personalized communication) is implemented on the IBM BlueGene/L systems using the MPI programming model [7]. For the one-to-all personalized communication pattern, however, the research is rather sparse. Both lower and upper bounds on the number of steps to scatter messages from the root node to their respective destinations on an arbitrary network topology are given in [8]. Algorithms are given for tree-shaped networks in [9]. A dispatching strategy called Furthest-Distance-First is studied for linear arrays (rings) with set-up times in [10].

This paper extends and details our earlier work in [11]. The exact one-to-all personalized communication problem like ours, with the torus topology, store-and-forward routing, and all-port processors, is studied in [12] and [13]. But both use different scattering algorithms from ours. In [12], messages are sent one dimension at a time. For example, in a 2-dimensional torus with each node referenced by the corresponding x and y coordinates, a message originally at the root node of $(0, 0)$ and with a destination of $(i, -)$, will be first sent to the intermediate node $(i, 0)$ along the x-axis and then will be sent to the destination along the y-axis. In [13], a tree-like partition of the torus network is used in scattering the messages from the root node. Our first algorithm that will be described in the next section also uses a partition scheme, but it differs from the tree partition in [13] in that each subnetwork in our partition has a node that is a neighbor of the root node while the partition in [13] may have subnetworks that are many hops away from the root node.

3 Algorithms for One-to-All Personalized Communication

We now describe two algorithms for the one-to-all personalized communication problem for torus networks. Although we have implemented our algorithms on a 2-

dimensional 8×8 torus and a 3-dimensional $4 \times 8 \times 8$ torus already in place and functioning at the Jefferson National Lab, the algorithms work for higher dimensions as well. From now on, we will name a node in the torus with its coordinates, e.g., node i is (x_i, y_i, z_i) in the 3-dimensional case. Assume that the root node is the origin of the coordination system with zero coordinates in all dimensions, i.e., $(0, 0, 0)$ in the 3-dimensional case. Also, we will name a message with the coordinates of the message's destination, e.g., message i is (x_i, y_i, z_i). In general, an algorithm for the one-to-all personalized communication problem must contain two components. The first component is the selection of messages to send for each time step. In a multicomputer system, the number of sending and receiving activities that a node can engage in is bounded by the number of ports the node can monitor simultaneously (e.g., single-port, multi-port, and all-port). When the number of messages waiting at a node is more than the number of allowable ports, a message or a set of messages has to be selected to be sent in the next time step. The second component is the actual routing. After a message is selected, a direction (among four in the 2-dimensional case and six in the 3-dimensional case) needs to be chosen to send the message in the next time step.

3.1 Optimal Algorithm OPT

Our first algorithm achieves optimality in minimizing the total time steps needed to scatter all p messages to their respective destinations. So we call it OPT. In one-to-all personalized communication, the root node is the bottleneck for delays since all messages are initially at the root node and have to be sent one by one (in the single-port mode) or group by group (in the multi-port and all-port modes) toward their respective destinations. The optimality of OPT is achieved in several ways.

First, each message is guaranteed to travel the shortest distance possible. For example, in the 3-dimensional case, if the message is (x_i, y_i, z_i) then the distance (measured by the number of hops) it will travel using OPT is $d(i) = \min\{x_i, xdim - x_i\} + \min\{y_i, ydim - y_i\} + \min\{z_i, zdim - z_i\}$, where $xdim$, $ydim$, and $zdim$ are sizes of the torus in dimensions x, y, and z, respectively. Second, messages are selected by OPT using the Furthest-Distance-First principle [10]. That is, a message that will travel the furthest distance to reach its destination will be among the first to be sent from each node. This way, collisions among messages could be entirely eliminated. Third, and most importantly, the torus is first partitioned into equal-size regions (i.e., containing roughly the same number of nodes in each region), with one region corresponding to one link leaving the root node and all nodes in the region are accessible from the link in the smallest number of hops possible. Consequently, all messages are also divided into groups, with one group corresponding to one region and with all messages of destinations falling into the region being put in the corresponding group. The partition

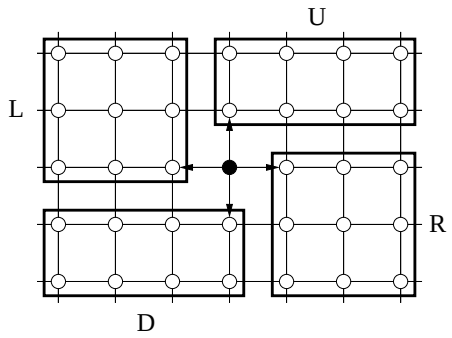

Figure 1. Partition of a 7×5 torus into four equal-size regions

establishes a 1-1-1 correspondence among a link leaving the root node, a region of nodes, and a group of messages. To spread out the congestion at the root node, messages in group i will be sent via link i to leave the root node to enter region i. And from then on, the message will stay within region i and travel to its destination without any delay in the smallest number of hops.

We explain the OPT algorithm using 2-dimensional tori as examples, although the method extends to higher dimensions. We first consider an example, where the network topology is a 2-dimensional torus of size 7×5. The partition of the torus is illustrated in Figure 1, where the solid black node is the root node with coordinate $(0, 0)$ and the four regions in the partition are bordered by darkened boundaries. For cleanliness, the wrap-around edges in the torus are not drawn in the figure. Note that in a higher dimension of D, the number of regions (or subnetworks, to be more precise) to be created in the partition will be $2D$. Once an equal-size partition is determined, the algorithm works pretty intuitively. Messages with destination nodes in regions R, L, U, and D will be sent out from the root node via the right, left, upper, and down links (marked by arrows), respectively. Among the messages that will use a certain one of the four links to enter the region of its destination, the message with the most number of hops to reach its destination will have the highest priority to traverse the link.

The example illustrated in Figure 1 can be easily generalized to any 2-dimensional torus of size $(2a+1) \times (2b+1)$, resulting a partition with both the R and L regions to be of rectangular size $a \times (b+1)$ and both the U and D regions to be of rectangular size $(a+1) \times b$. However, most multicomputers currently in use around the world have even sizes in all dimensions. So assume the torus is of size $2a \times 2b$. Using a similar partition method with the $odd \times odd$ case, we can partition the torus into the R, L, U, and D regions of rectangular sizes $(a-1) \times (b+1)$, $a \times b$, $a \times (b-1)$, and $(a+1) \times b$, respectively. This partition is clearly not an equal-size partition. See Figure 2 for an example of partitioning an 8×8 torus.

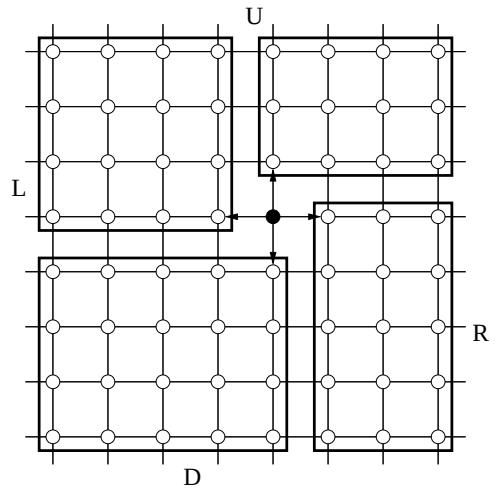

Figure 2. Partition of an 8×8 torus into four non-equal-size regions

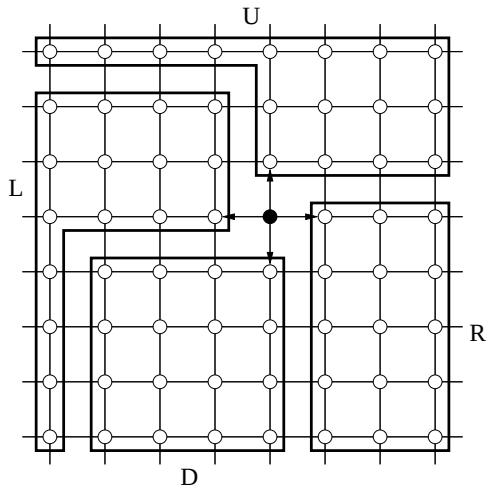

Figure 3. Partition of an 8×8 torus into four equal-size regions

To partition an $even \times even$ torus into equal-size regions, we use an ad-hoc approach. That is, given a torus topology with even sizes in all dimensions, we partition the torus into equal-size regions by hand and then hard-code the partition by defining each region one by one with assignment statements in the implementation. Although spotting an equal-size partition is not terribly hard, the process of hard-coding the partition is tedious. Fortunately, for each torus network on which one-to-all personalized communication is to be carried out, the partition of the network needs to be done only once. Figure 3 gives an equal-size partition of the 8×8 torus.

3.2 Proof of Optimality of OPT

Let p be the number of nodes in the D-dimensional torus. Since we use the all-port mode, $2D$ is thus the number of ports that a node can monitor simultaneously. Therefore, at any time, the root node will send out $2D$ messages using the $2D$ links adjacent to it. Then the root node will need at least $\lceil (p-1)/(2D) \rceil$ time steps to send out all $p-1$ messages, excluding the one destined to the root node itself. Let T_{OPT} be the number of time steps that OPT takes to dispatch all $p-1$ messages to their destinations. Then

$$T_{OPT} \geq \lceil (p-1)/(2D) \rceil.$$

We next prove the optimality of OPT by showing that OPT indeed takes $\lceil (p-1)/(2D) \rceil$ time steps to dispatch all $p-1$ messages to their destinations.

The partition in OPT creates $2D$ equal-size regions, $R_1, \ldots, R_{2D}$, with the size of a region to be no greater than $\lceil (p-1)/(2D) \rceil$. Since the movement of messages in different regions is parallel, we consider an arbitrary region R_j. For any node $i \in R_j$, let $d(i)$ be the minimum number of hops that message i (which is the message with destination i) travels from the root node to its destination. Recall that $d(i)$ is decided by all coordinates of node i as defined earlier. Also, let $w(i)$ be the number of time steps that message i waits at the root node before being sent to leave the root node. Assume $t(i)$ is the number of time steps that OPT takes for message i to reach its destination, then

$$t(i) = d(i) + w(i).$$

Note that after waiting for $w(i)$ time steps, message i enters its region and from then on it travels within the region without any further delay in the fastest way until reaching the destination. For easy explanation, assume that the nodes in R_j are relabeled with indices, $1, \ldots, |R_j|$, such that $w(i_1) < w(i_2)$ for $i_1 < i_2$. Therefore, $w(1) = 0, w(2) = 1, \ldots, w(|R_j|) = |R_j| - 1$. That is, message i in R_j leaves the root node at time $i-1$ and enters R_j at time i. Under this labeling scheme, we observe that messages with smaller indices wait less time to leave the root node but have longer distances to travel. So $d(1), \ldots, d(|R_j|)$ form a non-increasing sequence, with $d(i+1) = d(i)$ or $d(i+1) = d(i) - 1$ for any i and $w(1), \ldots, w(|R_j|)$ form a strictly increasing sequence, with $w(i+1) = w(i) + 1$ for any i. Adding the two sequences, we have a non-decreasing sequence $t(1), \ldots, t(|R_j|)$, with $t(i+1) = t(i)$ or $t(i+1) = t(i) + 1$ for any i. Therefore,

$$\begin{aligned} \max_{\forall i \in R_j} \{t(i)\} &= t(|R_j|) \\ &= d(|R_j|) + w(|R_j|) \\ &= 1 + (|R_j| - 1) \\ &= |R_j|. \end{aligned}$$

We then have the following definition for T_{OPT}.

$$\begin{aligned} T_{OPT} &= \max_{\forall j} \max_{\forall i \in R_j} \{t(i)\} \\ &= \max_{\forall j} \{|R_j|\} \\ &= \lceil (p-1)/(2D) \rceil. \end{aligned}$$

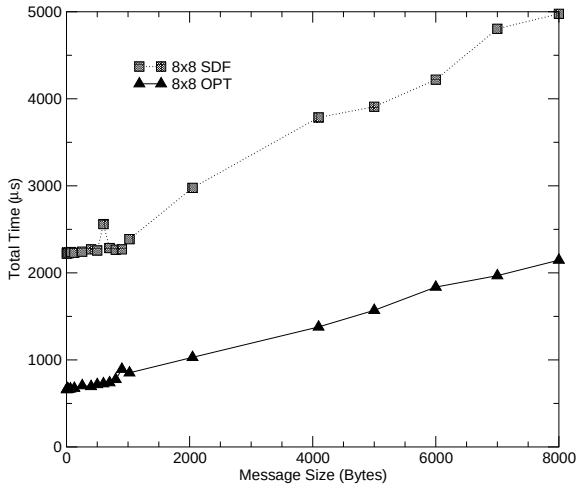

Figure 4. Performance data for the 8×8 cluster

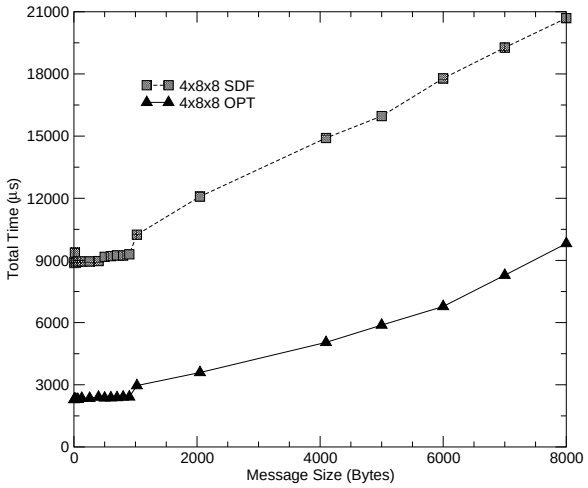

Figure 5. Performance data for the $4 \times 8 \times 8$ cluster

Matching the lower bound of T_{OPT}, the OPT algorithm is thus optimal.

3.3 Heuristic Algorithm SDF

Our second algorithm is a greedy heuristic, called Shortest-Direction-First (SDF), which does not go through the preprocessing phase of partitioning the torus. At any time step when there are nodes (including the root node) in the torus holding messages that have not reached their destinations, each of these nodes selects $k \leq 2D$ (in the case of k ports available at each processor) earliest arriving messages to send according to the First-Come-First-Serve principle and then send them along a direction (dimension) chosen by the Shortest-Direction-First principle, i.e., the direction in which a message has the smallest number of remaining hops will be chosen. The algorithm does not dispatch messages in the smallest number of time steps, thus is not optimal. But its simple design makes the implementation relatively easy without the partition overhead of the OPT algorithm.

4 Experiments

To compare the performance of OPT and SDF in practice, we have implemented both algorithms for an 8×8 configuration and a $4 \times 8 \times 8$ configuration of the torus connection clusters available at the Jefferson National Lab. Figure 4 presents the time results of the OPT and SDF algorithms on the 8×8 cluster for various message sizes and Figure 5 presents the time results of OPT and SDF on the $4 \times 8 \times 8$ cluster. SDF is easier to implement and OPT requires more overhead in obtaining the partition beforehand. However, on average, OPT dispatches messages to their destinations almost four times faster than SDF does for either cluster configuration across all tested message sizes. In addition, our OPT algorithm scales very well from the 8×8 configuration to the $4 \times 8 \times 8$ configuration for most part of tested message sizes except for the large sizes where simultaneous sending of large messages from the root node using all links requires more memory in practice and thus becomes difficult to achieve.

5 Conclusions

In this paper, we study the one-to-all personalized communication (scattering) problem for torus networks of all-port parallel processors with store-and-forward routing. We formulate the problem as an optimization problem with the goal to minimize the number of time steps needed to send same-size messages from the root node to each of the other nodes in the network. We give two algorithms, one of which uses a partition scheme and is proved to be optimal in that it takes the smallest number of time steps to dispatch all messages to their respective destinations, and the other of which is a heuristic algorithm. We present data collected from experiments conducted on two clusters (a 2-dimensional 8×8 torus and a 3-dimensional $4 \times 8 \times 8$ torus) available at the Jefferson National Lab. The data show that both algorithms are able to scatter messages in a torus network in a reasonably small amount of time although the heuristic algorithm is about four times slower than the optimal algorithm.

Our design of the optimal algorithm is based on partitioning the network into equal-size subnetworks. In the current implementation of the algorithm, the partition is done by hand first and then hard-coded with assignment statements. We are interested in developing a general partition algorithm which takes in as input the dimension sizes of a torus and produces as output an equal-size partition of the torus, which can then be used in routing messages to their destinations.

6 Acknowledgment

The authors wish to thank Robert Edwards in the Theory Group at the Jefferson National Lab for the discussion that motivated this research.

References

[1] J. Duato, S, Yalmanchili, and L. M. Ni, *Interconnection networks; An engineering approach*, IEEE Computer Society Press, 1997.

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

[3] S. Hinrichs, C. Kosak, D. R. O'Hallaron, T. M. Stricker, and R. Take, An architecture for optimal all-to-all personalized communication, *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1994, 310–319.

[4] J. Bruck, C.-T. Ho, S. Kipnis, and D. Weathersby, Efficient algorithms for all-to-all communications in multi-port message-passing systems, *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1994, 298–309.

[5] A. Massini, All-to-all personalized communication on multistage interconnection networks, *Discrete Applied Mathematics*, 128(2-3), 2003, 435–446.

[6] X. Zhang and C. Qiao, On scheduling all-to-all personalized connections and cost-effective designs in WDM rings, *IEEE/ACM Transactions on Networking*, 7(3), 1999, 435–445.

[7] G. Almasi, C. J. Archer, C.C. Erway, P. Heidelburger, X. Martorell, J. E. Moreira, B. Steinmacher-Burow, and Y. Zhang, Optimization of MPI collective communication on BlueGene/L systems, *Proceedings of the International Conference on Supercomputing*, 2005, 253–262.

[8] D. Barth and P. Fraigniaud, Approximation algorithms for structured communication problems, *Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures*, 1997, 180–188.

[9] S. Bhatt, G. Pucci, A. Ranade, and A. Rosenberg, Scattering and gathering messages in networks of processors, *IEEE Transactions on Computers*, 42(8), 1993, 938–948.

[10] Y.-D. Lyuu, On the furthest-distance-first principle for data scattering with set-up, *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing*, 1993, 633–640.

[11] J. Chen, W. Watson III, R. Edward, and W. Mao, Message passing for Linux clusters with Gigabit Ethernet connections, *Proceedings of the Fifth International Workshop on Communication Architecture for Clusters* (held in conjunction with the Nineteenth IEEE International Parallel and Distributed Processing Symposium (IPDPS05)), 2005, 1–8.

[12] Y. Saad and M. H. Schultz, Data communication in parallel architectures, *Parallel Computing*, 11(2), 1989, 131–150.

[13] E. Chan, R. van de Geijn, W. Gropp, and R. Thakur, Collective communication on architectures that support simultaneous communication over multiple links, *Proceedings of the ACM Symposium on Principles and Practice of Parallel Programming*, 2006, 2-11.