Broadcasting on Wormhole-Routed 2D Tori with Arbitrary Size[*]

Yomin Hou[a], Chien-Min Wang[b], Ming-Jer Tsai[b], and Lih-Hsing Hsu[a]
[a]Department of Computer and Information Science
National Chiao-Tung University
Hsinchu, Taiwan, ROC
Email:{gis81556,lhhsu}@cis.nctu.edu.tw
[b]Institute of Information Science, Academia Sinica
Taipei, Taiwan, ROC
Email: {cmwang,mjtsai}@iis.sinica.edu.tw

Abstract

For distributed memory parallel computers, broadcast operations are widely used in a variety of applications. In this paper, we propose an efficient algorithm for broadcasting on an all-port wormhole-routed 2D torus with arbitrary size. The underlying network is assumed to support only the dimension-ordered unicast. By taking the advantage of the all-port model and the distance insensitivity of the wormhole routing, the proposed algorithm can greatly reduce the number of message-passing steps. In addition, it can be proved to be depth contention-free. The performance study in this paper clearly shows the advantage of the proposed algorithm.

1. INTRODUCTION

A *massively parallel computer* (MPC) consists of a large number of identical processing elements and an interconnection network. Each processing element has its own processor, local memory, and other supporting devices. Processors in a MPC communicate by sending messages through the interconnection network. One of the most fundamental communication operations is *broadcast*, in which the same message is delivered from a source node to all nodes in the network. Efficient broadcast communication is useful in message-passing applications, and is also necessary in several other operations, such as replication and barrier synchronization [1], which are supported in data parallel languages.

Early systems that used the store-and-forward switching usually adopted the hypercube topology because its relatively dense interconnection network resulted in shorter message paths. However, many new-generation wormhole-routed MPCs use low-dimensional mesh and torus topologies. These topologies are simpler and more easily to construct than hypercubes. Although they exhibit larger internode distances, the relative distance insensitivity of wormhole routing obviates this problem. Dally [2] had shown that low-dimensional networks have lower latency and higher hot-spot throughput than high-dimensional networks with the same bisection width.

Most existing MPCs support broadcast in software. In these environments, broadcast must be implemented by sending multiple unicast messages. The simplest way is to send a separate copy of the broadcast message directly from the source to every other node. However, this strategy is unacceptable for its poor performance. An alternative approach is to use a *broadcast tree* to improve the performance. In each message-passing step of a broadcast tree, each node holding a copy of the message forwards it to some subset of the other nodes that have not yet received it. The number of messages a node can send out concurrently is determined by the system's *port model*. In an *all-port* system, each node can send and receive messages to and from all its neighbors at the same time.

In this paper, we address the problem of broadcasting on an all-port wormhole-routed two-dimensional (2D) torus with arbitrary size. Our method is intrigued from the idea of recursively decomposing a 2D torus into a number of smaller blocks. Following the decomposition, the broadcast message is also forwarded to every node. The proposed algorithm can be proved to be depth contention-free, and can greatly reduce the number of message-passing steps.

The remainder of the paper is organized as follows.

[*] This work was supported by National Science Council, Republic of China, under Grant NSC88-2213-E-001-008.

Section 2 describes the specific architectural characteristics of the systems considered in this paper, and Section 3 illustrates the issues and the problems involved in supporting efficient broadcast communication in such systems. In Sections 4, related works are briefly discussed. Section 5 gives the details of the proposed broadcast algorithm. The performance of the proposed algorithm is studied in Section 6. Finally, Section 7 concludes this paper.

2. The system model

The time required to move data between nodes is critical to the system performance. It can be evaluated by *communication latency*, which is the interval from the time the source node begins to send out a message until the destination node has received the message. Communication latency is composed of two parts: *software latency* and *network latency*. The software latency is the time required for the system to handle the message at both the source and destination nodes. The network latency equals the elapsed time after the head of a message has entered the network at the source until the tail of the message emerges from the network at the destination. In addition to the time for transmitting a message through channels, network latency also contains the *blocking time*, which includes all possible delays encountered during the lifetime of a message. For example, there may be delays due to *channel contention*, i.e., some channel is required by two or more unicasts simultaneously.

For software-supported broadcast, the total latency, called *broadcast latency*, is the interval from the time the source node begins to send the message until the last node has received the message. Since several message-passing steps may be required for broadcasting, broadcast latency is severely affected by the number of message-passing steps and the communication latency in each step. The way to minimize broadcast latency depends on the particular system architecture. The system architectures under consideration in this paper are characterized by four properties described below.

First, their topologies are 2D tori. A 2D torus $T_{n0\times n1}$ is an undirected graph of $n_0\times n_1$ nodes. Each node is denoted as (x, y), where $0\le x<n_0$ and $0\le y<n_1$. Node (x, y) has an edge connecting to $((x\pm1) \bmod n_0, y)$ along dimension 0, and an edge connecting to $(x, (y\pm1) \bmod n_1)$ along dimension 1. Each edge consists of two directed channels pointing in opposite directions. Without loss of generality, we may assume that the source node for broadcasting is at the center of the network, i.e., $(\lceil n_0/2\rceil-1, \lceil n_1/2\rceil-1)$, since the torus is a node symmetric topology.

Second, the *wormhole routing switching strategy* [8] is used. With wormhole routing, each message is divided into a number of flits. The header flit(s) carries the address information and governs the route while the remaining flits of the message follow in a pipeline fashion. One of the attractions of wormhole routing is that only small flit buffers are required. Another attraction is that, in the absence of channel contention, the network latencies of wormhole routing are relatively independent of the distance between the source and destination nodes.

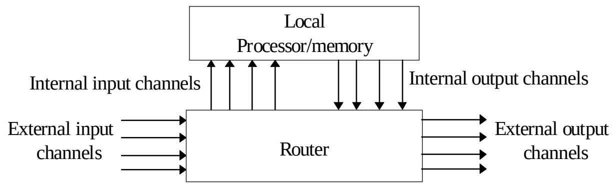


Fig. 1. The all-port node architecture.

Third, the *all-port* architecture is utilized. In wormhole-routed MPCs, communications among nodes are handled by a separate *router*, as shown in Fig. 1. The *external channels* connect the router to neighboring routers, and the *internal channels* connect to its local processor. The *port model* refers to the number of internal channels in each node. In the case of an *all-port* system, every external channel has a corresponding internal channel, thus allowing the node to send and receive messages to and from all its neighbors concurrently.

Finally, these systems use the deterministic *dimension-ordered routing algorithm* [8], which reserves links in a strictly increasing order of dimensions when sending messages. To provide shortest routing paths, *virtual channels* have to be added to prevent deadlock in a torus network. Dally[3] had illustrated the method to implement deadlock-free message routing with virtual channels. By accounting for the merits of the dimension-ordered routing algorithm, the designer of unicast-based collective operations may be able to eliminate channel contention so that the performance can be improved.

3. The problem

The most important issue for an efficient broadcast algorithm is to minimize the broadcast latency. To implement the broadcast operation in a network with N nodes, $(N-1)$ unicasts have to be generated so that every node can receive the broadcast message. If all these $(N-1)$ unicasts are sent by the source node, then the broadcast latency will be the sum of the communication latencies of the $(N-1)$ unicasts in the worst case. Obviously, this strategy is unacceptable for its poor performance. By organizing these unicasts as a *broadcast tree*, better performance may be obtained. Fig. 2 shows the difference of these two implementations for broadcasting on a 4×4 torus. In Fig. 2(a), (0,0) generates 15 unicasts to send the broadcast message. In Fig. 2(b), an example of a broadcast tree is illustrated.

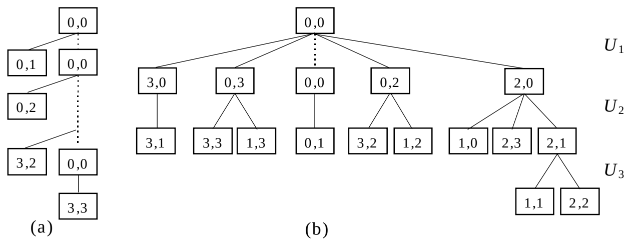


Fig. 2. Examples of broadcasting on a 4×4 torus.

The performance of a broadcast tree depends on the system's architecture, especially the switching strategy, the port model, and the unicast routing algorithm. By exploiting the distance insensitivity of wormhole routing, the communication latencies of the unicasts in a broadcast tree are approximately the same in the absence of channel contention. For this reason, in the absence of channel contention, the broadcast latency is approximate the time for performing the longest sequence of unicasts in the broadcast tree. For example, the longest sequence of unicasts in Fig. 2(b) is first from (0, 0) to (2, 0), then from (2, 0) to (2, 1), and finally from (2, 1) to either (1, 1) or (2, 2). However, if channel contention happens in some unicasts, their communication latencies may increase, and the broadcast latency may also increase and becomes unpredictable. From the above discussion, it is obvious that an efficient broadcast algorithm should minimize the number of unicasts in the longest sequence, i.e., minimize the number of message-passing steps, and prevent channel contention. To formally define the requirements of an efficient implementation for broadcasting on a dimension-ordered wormhole-routed network, some definitions are given below and several theorems for contention-free broadcast algorithms are also presented.

A unicast operation in a broadcast tree can be denoted as an ordered quadruple $(u, v, p(u, v), t)$ [7], where u and v are the source and destination nodes, respectively, $p(u, v)$ is a path based on dimension-ordered routing, and t is the message-passing step of the broadcast at which the unicast is performed. Two unicasts $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$ are *contention-free* if they will not contend for the same channel at the same time. In the following discussion, we shall use $C_i(p(u, v))$ to denote the ith channel in $p(u, v)$.

Definition 1: An implementation $I(B)$ of a broadcast B is a sequence of unicast sets $U_1, U_2, \ldots, U_k$, satisfying the following conditions.

1) For every unicast $(u, v, p(u, v), t) \in U_1$, $u=s_0$, where s_0 is the source node of B.
2) For every unicast $(u, v, p(u, v), t) \in U_t$, $u \neq s_0$ and $1<t\leq k$, there must exist a set U_j with $j<t$ which has $(w, u, p(w, u), j)$ as a member for some node w.
3) For any two unicasts $(u, v, p(u, v), t)$ and $(x, y, p(x, y), t)$ in U_t, $1\leq t\leq k$, if $u=x$, then $C_1(p(u, v)) \neq C_1(p(x, y))$.
4) For every node d_i, $d_i \neq s_0$, there exist exactly one node w such that $(w, d_i, p(w, d_i), j)$ appear in U_j for some integer j, $1\leq j\leq k$.

The first condition in Definition 1 states that only the source node is sending messages in the first step. The second condition guarantees that a node, except the source node, has received the message before it may forward the message to other nodes. The third condition implies that the messages sent by the same node in a step must use different output channels. Finally, the last condition ensures that every node receives the broadcast message exactly once. In the example illustrated in Fig. 2(b), there are three unicast sets in the implementation. A broadcast implementation is said to be *stepwise contention-free* if the elements in each unicast set U_i are pairwise contention-free, and *depth contention-free* if any two unicasts in the broadcast implementation are contention-free. Theorem 1 and Theorem 2 give sufficient conditions for two unicasts in an implementation to be contention-free, and Theorem 3 gives sufficient conditions for an implementation to be depth contention-free.

Definition 2: Given a broadcast implementation $I(B)=\{U_1, U_2,\ldots, U_k\}$, a node v is in the reachable set of node u, denoted as R_u, if and only if $v=u$ or there exists j, $1\leq j\leq k$, such that $(w, v, p(w, v), j)\in U_j$ for some node $w\in R_u$.

Theorem 1: Given a broadcast implementation $I(B)$, if at least one of the following three conditions holds for any two unicasts $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$ in $I(B)$, where $t\leq\tau$, then they are contention-free.

1) $x\in R_v$.
2) $p(u, v)$ and $p(x, y)$ are arc-disjoint.
3) $u=x$ and $C_1(p(u, v)) = C_1(p(x, y))$.

Proof: We shall show that channel contention does not arise in these three conditions.

1) If $x\in R_v$, then $(u, v, p(u, v), t)$ must be completed before $(x, y, p(x, y), \tau)$ begins, as shown in Fig. 3(a). Clearly, they are contention-free. (Note: $u\notin R_y$ since $t\leq\tau$.)
2) If $p(u, v)$ and $p(x, y)$ are arc-disjoint, then, by definition, the two unicasts must be contention-free.
3) If $u=x$ and $C_1(p(u, v)) = C_1(p(x, y))$, then we can derive $t<\tau$ since u can send only one message through the same output channel at a time as defined in Definition 1. Fig. 3(b) illustrates this situation. Note that u sends the

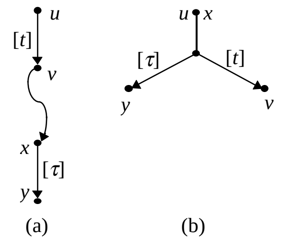


Fig. 3. Conditions 1 and 3 in Theorem 1.

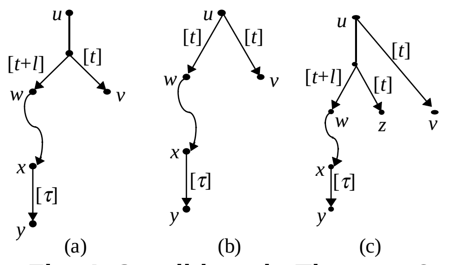


Fig. 4. Conditions in Theorem 2.

message to v before sending it to y. Even if $\tau=t+1$ and the sending latency is 0, no channel contention will happen. □

Theorem 2: Given a broadcast implementation $I(B)$, if at least one of the following three conditions holds for any two unicasts $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$ in $I(B)$, where $t\leq\tau$, and there is no channel contention between $(u, v, p(u, v), t)$ and any other unicast, then they are contention-free.

1) $x\in R_w$, $(u, w, p(u, w), t+l)\in I(B)$ for some node w and positive integer l, and $C_1(p(u, v)) = C_1(p(u, w))$.
2) $x\in R_w$, $(u, w, p(u, w), t)\in I(B)$ for some node w, and $(u, v, p(u, v), t)$ is sent out before $(u, w, p(u, w), t)$ is sent out by the same node u.
3) $x\in R_w$, $(u, w, p(u, w), t+l)\in I(B)$ for some node w and positive integer l; $(u, z, p(u, z), t)\in I(B)$ for some node z and $C_1(p(u, z)) = C_1(p(u, w))$; and $(u, v, p(u, v), t)$ is sent out before $(u, z, p(u, z), t)$ is sent out by the same node u.

Proof: We shall show that channel contention does not arise in these three conditions.

1) This condition is shown in Fig. 4(a). As proved in Condition 3 of Theorem 1, node u must complete sending the message to node v before it can start to send the message to node w. Since node w is either an ancestor of x or x itself and there is no channel contention between $(u, v, p(u, v), t)$ and any other unicast, clearly, node v will receive the message prior to node x. This prevents channel contention between $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$.
2) Fig. 4(b) illustrates this condition. Let c be a common channel of $p(u, v)$ and $p(x, y)$. Suppose c is from node z to node z'. Since z is a node in $p(u, v)$ and dimension-ordered routing is used, the path from u to z in $p(u, v)$ is the shortest path from u to z. Hence, without contending any channel with other unicast, $(u, v, p(u, v), t)$ must have passed through c completely before $(x, y, p(x, y), \tau)$ reserves it because $(u, v, p(u, v), t)$ is sent out before $(u, w, p(u, w), t)$. This prevents channel contention between $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$.
3) As shown in Fig. 4(c), this condition is a combination of Conditions 1 and 2. From the proof of Conditions 1 and 2, there is no channel contention between $(u, v, p(u, v), t)$ and $(x, y, p(x, y), \tau)$. □

Theorem 3: Given a broadcast implementation $I(B)$, if at least one of the conditions in Theorem 1 or Theorem 2 holds for every pair of unicasts in $I(B)$, then $I(B)$ is depth contention-free.

Proof: Let $I(B)= \{U_1, U_2,\ldots, U_m\}$. We shall prove that, for any two unicasts in $\{U_1, U_2,\ldots, U_i\}$, $1\leq i\leq m$, there is no channel contention. The proof is by mathematic induction on i.

For $i=1$, since all the unicasts in U_1 are generated by the source node, they must be arc-disjoint. Hence, there is no channel contention between any two unicasts in U_1. Suppose there is no channel contention between any two unicasts in $\{U_1, U_2,\ldots, U_k\}$, $1\leq k<m$. We shall prove that, for any two unicasts in $\{U_1, U_2,\ldots, U_{k+1}\}$, there is no channel contention.

Consider those unicasts in U_{k+1}. Any two of them must be arc-disjoint because no other condition in Theorem 1 or Theorem 2 can hold for them. Therefore, there is no channel contention between any two unicasts in U_{k+1}. From the assumption, there is no channel contention between any two unicasts in $\{U_1, U_2,\ldots, U_k\}$. Hence, if there is channel contention between two unicasts in $\{U_1, U_2,\ldots, U_{k+1}\}$, then one of them should be in U_{k+1}, and the other one should be in $\{U_1, U_2,\ldots, U_k\}$.

Let $(u, v, p(u, v), t) \in \{U_1, U_2,\ldots, U_k\}$ and $(x_0, y_0, p(x_0, y_0), \tau_0) \in U_{k+1}$ be two unicasts that contend for the same channel. If any condition in Theorem 1 holds for them, they must be contention-free. Hence, only those conditions in Theorem 2 may hold for them. Moreover, there must exist another unicast $(x_1, y_1, p(x_1, y_1), \tau_1)$, $k<\tau_1$, which contends for some channel with $(u, v, p(u, v), t)$ before the time the channel contention between $(u, v, p(u, v), t)$ and $(x_0, y_0, p(x_0, y_0), \tau_0)$ occurs. Since the two unicasts $(u, v, p(u, v), t)$ and $(x_1, y_1, p(x_1, y_1), \tau_1)$ are not contention-free either, similar situation happens again. Thus, for any $(x_i, y_i, p(x_i, y_i), \tau_i)$, $k<\tau_i$, which contends for some channel with $(u, v, p(u, v), t)$, there must exist another unicast $(x_{i+1}, y_{i+1}, p(x_{i+1}, y_{i+1}), \tau_{i+1})$, $k<\tau_{i+1}$, which contends for some channel with $(u, v, p(u, v), t)$ before the time the channel contention between $(u, v, p(u, v), t)$ and $(x_i, y_i, p(x_i, y_i), \tau_i)$ occurs. Since this infinite situation could never happen, there must be no channel contention between $(u, v, p(u, v), t)$ and $(x_0, y_0, p(x_0, y_0), \tau_0)$. Therefore, there is no channel contention between any two unicasts in $\{U_1, U_2,\ldots, U_{k+1}\}$, $1\leq k<m$. Hence, by mathematic induction, $I(B)$ is depth contention-free. □

4. Related works

The problem of broadcasting on wormhole-routed networks has been studied extensively. A recent survey can be found in [6]. In terms of the all-port model, Ho and Kao [4] developed an optimal broadcast algorithm for hypercubes. On meshes and tori, Tsai and Mckinley [9], [10] proposed the *extended dominating nodes* (*EDN*) approach for broadcasting. Their algorithm requires d steps for broadcasting on a $2^d\times2^d$ torus, and is proved to be stepwise contention-free. However, channel contention may happen between two unicasts in different steps. To eliminate the channel contention, we had proposed a depth contention-free broadcast algorithm [5], which also requires only d steps for broadcasting on a $2^d\times2^d$ torus.

For broadcasting on an all-port $n_0\times n_1$ torus, $n_0\leq n_1$, Tseng [11] proposed a *dilated-diagonal-based scheme*. This algorithm required $2\lceil\log_5 n\rceil+1$ steps for a square $n\times n$ torus. For a nonsquare $n_0\times n_1$ torus, $n_0<n_1$, $(\lceil\log_5 n_0\rceil + \lceil\log_5(n_0/2)\rceil + \lceil\log_5(n_1/n_0)\rceil + t)$ steps are required, where t=2 if n_0 is even, and t=3 if n_0 is odd. However, this algorithm did not consider broadcasting on an $n_0\times n_1$ torus, where $n_0>n_1$. Furthermore, channel contention may happen in Tseng's algorithm.

5. The proposed algorithm

In this section, we shall propose a broadcast algorithm for all-port 2D tori with arbitrary size that is both efficient and depth contention-free.

5.1 Straight pattern and Turn pattern

The proposed broadcast algorithm is based on two broadcast patterns, *Straight pattern* (S-pattern) and *Turn pattern* (T-pattern). Fig. 5 shows these two patterns. Following each pattern, a node can send four unicasts concurrently if the size of the network is large enough. At the first step of broadcasting, a pattern will be chosen by the source node to sent the broadcast message. According to the pattern, the torus network can be divided into five smaller blocks as shown by dashed lines in Fig. 5. A *block* is a connected sub-network of the torus network. After the first step finished, there are five nodes holding the broadcast message. Each of them is in different blocks and will take the responsibility of broadcasting the message to all nodes of the block. From Fig. 5, it can be observed that, by appropriately choosing the sizes of the blocks, the five blocks will be independent and every node will belong to exactly one block no matter which pattern is used. Recursively applying the above concept, a torus will be decomposed into a set of smaller blocks at each step and the message will be forwarded to every node following the decomposition. In the following paragraphs, we will show how to decide the pattern to be used at each step and the sizes of the blocks.

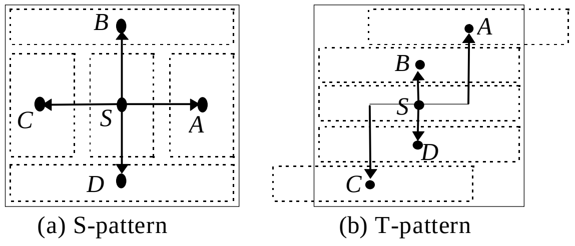
Let $W(n_0, k)$ be the maximum n_1 such that the broadcast operation can be finished in k steps for an $n_0\times n_1$ torus. We can give it a recursive definition that

$$W(n_0, k)= \mathrm{MAX}\{\ W_S(n_0, k),\ W_T(n_0, k)\ \},\ \text{where}$$
$$W_T(n, k)= 5W(n, k\text{-}1),\ \text{and}$$
$$W_S(n, k)= 2W_S(n, k\text{-}1)+ \mathrm{MIN}\{\ W_S(\lfloor n/3\rfloor, k\text{-}1),\ W_S(\lceil n/3\rceil, k\text{-}1)\ \}.$$

If $W(n_0, k)= W_S(n_0, k)$, then S-pattern is adopted for broadcasting. Otherwise, T-pattern is adopted. Note that the five blocks after applying T-pattern are out of alignment. Therefore, T-pattern can only be adopted by a block whose size along dimension 0 is the same as that of the torus since there exist wraparound channels in torus networks. Also note that MIN{ $W_S(\lfloor n/3\rfloor, k\text{-}1)$, $W_S(\lceil n/3\rceil, k\text{-}1)$ } is considered in the definition of $W_S(n, k)$. This ensures that all the nodes in a block will be in some smaller block after S-pattern is applied.

Suppose $S= (X_S, Y_S)$ is the source node for broadcasting on an $n_0\times n_1$ torus, a pre-computed table can be consulted to find k' such that $W(n_0, k'\text{-}1)< n_1\leq W(n_0, k')$. The pattern to be used can also be determined. Fig. 6 shows the block sizes for these two patterns, and Fig. 7 shows the addresses of the nodes that receive the broadcast message from node S. In Fig. 6 and Fig. 7, n_0 is assumed to be divisible by 3 and n_1 is assumed to be divisible by 5. If indivisible condition happens, the even policy is used such that the difference between any two parts is no more than one. Consider the example of broadcasting on a 3×8 torus. T-pattern will be adopted and the torus will be divided into 5 blocks at the first step. Since 8 is indivisible by 5, the even policy is used and the sizes of the 5 blocks are 3×1, 3×2, 3×2, 3×2 and 3×1, respectively.

For successfully applying the recursive decomposition



(a) S-pattern (b) T-pattern

Fig. 5. Straight pattern and Turn pattern.

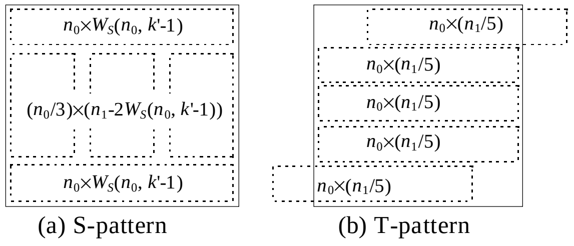

$n_0\times W_S(n_0, k'\text{-}1)$
$(n_0/3)\times(n_1\text{-}2W_S(n_0, k'\text{-}1))$
$n_0\times W_S(n_0, k'\text{-}1)$
$n_0\times(n_1/5)$
$n_0\times(n_1/5)$
$n_0\times(n_1/5)$
$n_0\times(n_1/5)$
$n_0\times(n_1/5)$
(a) S-pattern (b) T-pattern

Fig. 6. Block sizes for S-pattern and T-pattern.

$$\begin{tikzcd} S=(X_S,\ Y_S) \arrow[r] \arrow[dr] \arrow[ddr] \arrow[dddr] & A=(X_S+n_0/3,\ Y_S) \\ & B=(X_S,\ Y_S+(\lceil n_1/2\rceil-\lceil W_s(n_0,\ k'-1)/2\rceil)) \\ & C=(X_S-n_0/3,\ Y_S) \\ & D=(X_S,\ Y_S-(\lfloor n_1/2\rfloor-\lfloor W_s(n_0,\ k'-1)/2\rfloor)) \end{tikzcd}$$

(a) S-pattern.

$$\begin{tikzcd} S=(X_S,\ Y_S) \arrow[r] \arrow[dr] \arrow[ddr] \arrow[dddr] & \mathrm{A}=(X_S+1,\ Y_S+2n_1/5) \\ & B=(X_S,\ Y_S+n_1/5) \\ & C=(X_S-1,\ Y_S-2n_1/5) \\ & D=(X_S,\ Y_S-n_1/5) \end{tikzcd}$$

(b) T-pattern.

Fig. 7. Destination nodes for S-pattern and T-pattern.

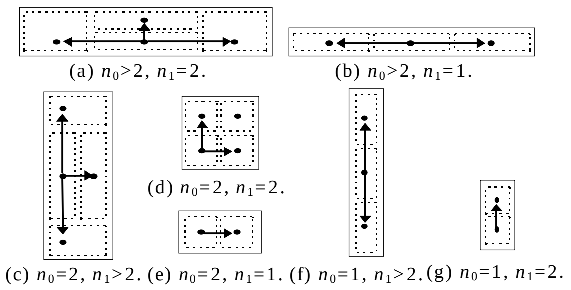


Fig. 8. The degenerated patterns.

process for broadcasting, the degenerated patterns for small blocks should also be determined. Because neither S-pattern nor T-pattern can be performed completely for an $n_0 \times n_1$ block, where n_0<3 or n_1<3, a degenerated pattern should be used for broadcasting. Fig. 8 shows the degenerated patterns considered in the proposed algorithm. When broadcasting, a torus is recursively decomposed into a set of smaller blocks. Finally, each of the blocks consists of one node only, and the broadcast message can be forwarded to every node following the decomposition.

5.2 The broadcast algorithm

The above subsection has illustrated the concept and the requirements for successfully applying S-pattern and T-pattern to broadcast on a 2D torus. Fig. 9 gives the detail of the broadcast algorithm. The `while-loop` of Fig. 9 is the main part of the algorithm. Once a node receives the broadcasting message, it becomes the source node of the $n_0 \times n_1$ block, and begins to broadcast the message. In the `while-loop`, the pattern to be used is determined by `look_table()`, then `pattern_send()` is called to send out the message and set the block sizes. The `while-loop` ends when the block size is 1×1, i.e., only one node in the block. This algorithm can broadcast a message to every node correctly. The proof is presented in Theorem 4. It can also be proved that the algorithm is depth contention-free as shown in Theorem 5.

```
/* Broadcasting on a n0×n1 torus. */
Procedure dcf_broadcast(int n0, int n1)
{
 char pre_ptn; /* Pattern  been  used  for  sending
                  message to this processor. */
 char ptn; /* Pattern to be used for broadcasting */
 if( node_id == bcst_source_node ){
     /* I am source node */
     Prepare the broadcast message B_MSG;
     pre_ptn = ' ';
 }else{
     b_receive(&n0,&n1,&pre_ptn,&B_MSG);
    /* Receive the broadcast message, the previously
       used pattern, and the block size n0×n1. */
 }
 while( n0 > 1 || n1 > 1){
     ptn= look_table(n0, n1, pre_ptn).
     /*look_table()return a pattern to be used. */
     pattern_send( ptn, &n0, &n1, B_MSG);
     /*Follow ptn to send out messages. Set the block
       sizes for each block. */
     pre_ptn= ptn;
 }
}
```

Fig. 9. The proposed broadcast algorithm.

Theorem 4. The proposed algorithm delivers a message exactly once to every node in an $n_0 \times n_1$ torus.

Proof: It can be observed that an $n_0 \times n_1$ torus is also a block. From the algorithm, if a block consists of more than one nodes, it will be decomposed into a set of independent blocks following one of the patterns shown in Fig. 5 and Fig. 8. The size of each block for the patterns in Fig. 5 is determined as shown in Fig. 6, and the block sizes for the patterns in Fig. 8 can be obtained similarly. Therefore, after each step, every node will belong to exactly one of the blocks, and only one node in each block will hold the broadcast message. Moreover, each node holding the message will take the responsibility of broadcasting in the block and will not send the message to any node in other blocks. Therefore, following the decomposition, the message will be forwarded to every node exactly once. □

Lemma 1. Suppose S-pattern is adopted for broadcasting in block **B**. Let *u* be one of the four unicasts of S-pattern and *v* be any unicast in **B**. At least one condition in Theorem 1 or Theorem 2 holds for *u* and *v*.

Proof: Let the five smaller blocks after applying S-pattern be $\mathbf{B}_S$, $\mathbf{B}_A$, $\mathbf{B}_B$, $\mathbf{B}_C$, and $\mathbf{B}_D$, and the source nodes of these blocks be *S*, *A*, *B*, *C*, and *D*, respectively. Note that *S* is also the source node of block **B**. Suppose S-pattern is adopted for broadcasting in block **B** at step *t*. The four unicasts of S-pattern should be (*S*, *A*, *p*(*S*, *A*), *t*), (*S*, *B*, *p*(*S*, *B*), *t*), (*S*, *C*, *p*(*S*, *C*), *t*), and (*S*, *D*, *p*(*S*, *D*), *t*). Clearly, all these four unicasts are arc-disjoint. Consider the unicast (*S*, *A*, *p*(*S*, *A*), *t*).

1. Because it passes $\mathbf{B}_S$ and $\mathbf{B}_A$ only, for any unicast *v* in $\mathbf{B}_B$, $\mathbf{B}_C$ and $\mathbf{B}_D$, (*S*, *A*, *p*(*S*, *A*), *t*) and *v* are arc-disjoint. Hence, Condition 2 of Theorem 1 holds.

2. For each unicast (x, y, $p(x, y)$, τ) in $\mathbf{B}_A$, $x \in R_A$, Condition 1 of Theorem 1 must hold.
3. For each unicast (x, y, $p(x, y)$, τ) in $\mathbf{B}_S$, if it passes some channel of $p(S, A)$, then one of following conditions holds.
   (1) $x=S$. Since node S uses S-pattern at step t, from the algorithm, it will use S-pattern or the degenerated patterns at the following steps. Therefore, y must be a node in $p(S, A)$ and $C_1(p(x, y)) = C_1(p(S, A))$. Hence, Condition 3 of Theorem 1 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).
   (2) There exists an ancestor node w of x, and w receives the message directly from S at step $t+l$ for some positive integer l. As we had mentioned above, node S must use S-pattern or one of the degenerated patterns at step $t+l$. Therefore, w must be a node in $p(S, A)$ and $C_1(p(S, w)) = C_1(p(S, A))$. Hence, Condition 1 of Theorem 2 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).

From the above discussion, if v is any unicast in **B,** at least one condition in Theorem 1 or Theorem 2 holds for (S, A, $p(S, A)$, t) and v. Similar proof can be applied to (S, B, $p(S, B)$, t), (S, C, $p(S, C)$, t), and (S, D, $p(S, D)$, t). This completes the proof. □

Lemma 2. Suppose T-pattern is adopted for broadcasting in block **B**. Let u be one of the four unicasts of T-pattern and v be any unicast in **B**. At least one condition in Theorem 1 or Theorem 2 holds for u and v.

Proof: Let the five smaller blocks after applying T-pattern be $\mathbf{B}_S$, $\mathbf{B}_A$, $\mathbf{B}_B$, $\mathbf{B}_C$, and $\mathbf{B}_D$, and the source nodes of these blocks be S, A, B, C, and D, respectively. Note that S is also the source node of block **B**. Suppose T-pattern is adopted for broadcasting in block **B** at step t. The four unicasts of T-pattern should be (S, A, $p(S, A)$, t), (S, B, $p(S, B)$, t), (S, C, $p(S, C)$, t), and (S, D, $p(S, D)$, t). Clearly, all these four unicasts are arc-disjoint. Consider the unicast (S, A, $p(S, A)$, t).

1. Because it passes $\mathbf{B}_S$, $\mathbf{B}_A$ and $\mathbf{B}_B$ only, for any unicast v in $\mathbf{B}_C$ and $\mathbf{B}_D$, (S, A, $p(S, A)$, t) and v are arc-disjoint. Hence, Condition 2 of Theorem 1 holds.
2. For each unicast (x, y, $p(x, y)$, τ) in $\mathbf{B}_A$, $x \in R_A$, Condition 1 of Theorem 1 must hold.
3. For each unicast (x, y, $p(x, y)$, τ) in $\mathbf{B}_B$, $x \in R_B$. Since (S, A, $p(S, A)$, t) is sent out before (S, B, $p(S, B)$, t), Condition 2 of Theorem 2 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).
4. For each unicast (x, y, $p(x, y)$, τ) in $\mathbf{B}_S$, if it passes some channel of $p(S, A)$ along dimension 0, then $x=S$ because there is only one channel $p(S, A)$ along dimension 0. Hence, Condition 3 of Theorem 1 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).

   If (x, y, $p(x, y)$, τ) passes some channel of $p(S, A)$ along dimension 1, then there exists an ancestor node w of x, w receives the message directly from S and one of following conditions holds.
   (1) $C_1(p(S, w)) = C_1(p(S, A))$. Hence, Condition 1 of Theorem 2 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).
   (2) $C_1(p(S, w)) = C_1(p(S, B))$. Since ($S$, A, $p(S, A)$, t) is sent out before (S, B, $p(S, B)$, t), Condition 3 of Theorem 2 holds for (S, A, $p(S, A)$, t) and (x, y, $p(x, y)$, τ).

From the above discussion, if v is any unicast in **B,** at least one condition in Theorem 1 or Theorem 2 holds for (S, A, $p(S, A)$, t) and v. Similar proof can be applied to (S, C, $p(S, C)$, t). The proof for (S, B, $p(S, B)$, t) and (S, D, $p(S, D)$, t) is similar to the proof in Lemma 1. This completes the proof. □

Lemma 3. Suppose one of the seven patterns in Fig. 8 is adopted for broadcasting in block **B**. Let u be a unicast of the pattern and v be any unicast in **B**. At least one condition in Theorem 1 or Theorem 2 holds for u and v.

Proof: It can be observed that each pattern in Fig. 8 is a special case of S-pattern. Hence, this lemma can be proved from Lemma 1. □

Theorem 5. The proposed algorithm for broadcasting on an $n_0 \times n_1$ torus is depth contention-free.

Proof: As we have proved in Theorem 4, an $n_0 \times n_1$ torus is a block and will be decomposed recursively following the patterns in Fig. 5 and Fig. 8. Since all the blocks in a step are independent, from Lemmas 1~3, it can be easily proved that at least one condition in Theorem 1 or Theorem 2 holds for any two unicasts in the proposed algorithm. Therefore, from Theorem 3, the proposed algorithm is depth contention-free. □

6. Performance study

In this section, the performance of the proposed algorithm is studied. In an all-port 2D torus, each node can send messages to at most four nodes simultaneously. Thus, there exists a theoretical lower bound, $\lceil \log_5(n_0 \times n_1) \rceil$, on the number of message-passing steps for broadcasting on

$N=n_0 \times n_1$	$\lceil \log_5(n_1) \rceil$	Tseng $n_0 \le n_1$	DCF $n_0 \le n_1$	DCF
9 ~ 25	2	4.90	2.70	2.82
26 ~ 125	3	6.17	3.84	3.95
126 ~ 625	4	7.19	4.85	5.10
626 ~ 3125	5	8.19	5.89	6.24
3126 ~ 15625	6	9.20	6.95	7.37
15626 ~ 78125	7	10.19	7.95	8.49
78126 ~ 390625	8	11.20	8.98	9.62
390626 ~ 1953125	9	12.19	10.02	10.74
1953126 ~ 9765625	10	13.20	11.03	11.87

Table 1. Average number of steps for broadcasting on $n_0 \times n_1$ tori.

an $n_0 \times n_1$ torus. However, this lower bound may not be achievable because it is not always possible for every node to send the message to four new nodes in every step under the constrain of dimension-ordered routing. For Tseng's algorithm on an $n_0 \times n_1$ torus, the number of message-passing steps can be represented as follows.

$$\begin{cases} 2\lceil \log_5 n_0 \rceil + 1 & \text{, if } n_0 = n_1, \\ \lceil \log_5 n_0 \rceil + \left\lceil \log_5 \frac{n_0}{2} \right\rceil + \left\lceil \log_5 \frac{n_1}{n_0} \right\rceil + 2 & \text{, if } n_0 < n_1 \text{ and } n_0 \text{ is even,} \\ \lceil \log_5 n_0 \rceil + \left\lceil \log_5 \frac{n_0}{2} \right\rceil + \left\lceil \log_5 \frac{n_1}{n_0} \right\rceil + 3 & \text{, if } n_0 < n_1 \text{ and } n_0 \text{ is odd,} \\ \text{not considered} & \text{, if } n_0 > n_1. \end{cases}$$

The analysis of the proposed algorithm on an $n_0 \times n_1$ torus is discussed as follows. At each step, the nodes holding the broadcast message may use different patterns for broadcasting. Which pattern to use in a block depends on the size of the block. The number of steps k' for the proposed algorithm is decided by $W(n_0, k')$ such that $W(n_0, k'-1) < n_1 \le W(n_0, k')$. Since it is difficult to give an analytical solution for $W(n_0, k')$, k' is unable to be formulated by a general equation.

To compare Tseng's algorithm with the proposed algorithm, the number of steps for broadcasting on a torus with N nodes is computed. For tori with N nodes, the average numbers of steps for $5^i \le N < 5^{i+1}$, $1 \le i \le 9$, are shown in Table 1. Only those tori that consist of at least 3 nodes along each dimension are considered. In the first column of Table 1, the range of N is illustrated. Column 2 shows the theoretical lower bound for broadcasting on a torus with N nodes. The average number of steps for Tseng's algorithm is shown in column 3. Note that only those tori with $n_0 \le n_1$ are considered for Tseng's algorithm. In order to compare with Tseng's algorithm, we also consider only those tori with $n_0 \le n_1$ for the proposed algorithm in column 4. It can be observed that the proposed algorithm requires about 2 fewer steps than Tseng's algorithm. Moreover, no channel contention will occur for the proposed algorithm. Broadcasting on the tori with $n_0 > n_1$ is not considered in Tseng's algorithm. The proposed broadcast algorithm does not have such problem and can be used for arbitrary tori. Column 5 shows the average numbers of steps for all the tori with arbitrary size by using the proposed algorithm.

7. Conclusion

In this paper, we propose a new algorithm for broadcasting messages on all-port wormhole-routed 2D tori with arbitrary size. The underlying network is assumed to support only the dimension-ordered unicast. When broadcasting, each node holding the broadcast message follows one of the two patterns, S-pattern and T-pattern, to send out the message. The torus network is recursively decomposed into a set of smaller blocks at each step such that exactly one node in each block holds the message. Following the decomposition, the message is forwarded to every node. The performance study shows that the proposed algorithm requires about 2 fewer steps than the Tseng's algorithm in average. Furthermore, the proposed algorithm achieves the goal of being depth contention-free and being applicable to a 2D torus with arbitrary size.

References

[1] Tim S. Axelrod, "Effects of synchronization barriers on multiprocessor performance," *Parallel Computing*, Vol. 3, No. 2, May. 1986, pp. 129-140.

[2] Willian J. Dally, "Performance Analysis of *k*-ary *n*-cube Interconnection Networks," *IEEE Transactions on Computers*, Vol. 39, No. 6, Jun. 1990, pp. 775-785.

[3] Willian J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, Vol. C-36, No. 5, May. 1987, pp. 547-553.

[4] Ching-Ten Ho and Ming-Yang Kao, "Optimal Broadcast in All-Port Wormhole-Routed Hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 2, Feb. 1995, pp. 200-204.

[5] Yomin Hou, Chien-Min Wang and Lih-Hsing Hsu, "Depth Contention-Free Broadcasting on Torus Networks," *International Conference on Supercomputing*, Jul. 1998, pp. 393-400.

[6] Philip K. McKinley, Yih-jia Tsai, and David F. Robinson, "Collective Communication in Wormhole-routed Massively Parallel Computers," *IEEE Computer*, Vol. 28, No. 12, Dec. 1995, pp. 39-50.

[7] Philip K. McKinley, Hong Xu, Abdol-Hossein Esfahanian, and Lionel M. Ni, "Unicast-Based Multicast Communication in Wormhole-Routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 12, Dec. 1994, pp. 1252-1265.

[8] Lionel M. Ni and Philip K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, Vol. 26, Feb. 1993, pp. 62-76.

[9] Yih-jia Tsai and Philip K. McKinley, "An Extended Dominating Node Approach to Collective Communication in Wormhole-Routed 2D Meshes," *Proc. Scalable High Performance Computing Conf.*, May. 1994, pp. 199-206.

[10] Yih-jia Tsai and Philip K. McKinley, "A Broadcast Algorithm for All-Port Wormhole-Routed Torus Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 8, Aug. 1996, pp. 876-885.

[11] Yu-Chee Tseng, "A Dilated-Diagonal-Based Scheme for Broadcast in a Wormhole-Routed 2D Torus," *IEEE Transactions on Computers*, Vol. 46, No. 8, Aug. 1997, pp. 947-952.