Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

A tree-based algorithm for multicasting in 2D torus networks

M.A. Abd El-Baky [*]

Department of Mathematics, Faculty of Science, Fayoum University, Egypt

Received 11 March 2014; revised 17 December 2014; accepted 20 December 2014
Available online 9 January 2015

KEYWORDS

Multicast routing;
Torus network;
Tree-based algorithms;
Communication latency

Abstract Torus network has become increasingly important to multicomputer design because of its many desirable properties including low bandwidth and fixed degree of nodes. Also, torus networks can be partitioned into mesh networks. The multicast pattern, in which one source node sends the same message to multiple destination nodes, is the essential pattern in a wide variety of applications. This paper proposes a multicast routing scheme in 2D torus networks. The proposed scheme is a Tree-based Algorithm which Splits torus Networks into two Equally Meshes, hence it is called TASNEM. TASNEM algorithm is a tree-based technique, in which the router simultaneously sends incoming flits on more than one outgoing channel. It requires at most two start-up times, one for each mesh subnetwork. For each mesh subnetwork, the source node delivers a message to the destination nodes along one main path and different horizontal paths branched from the main path. TASNEM algorithm can achieve high degree of parallelism and low communication latency over a wide range of traffic loads. Performance results of a simulation study on torus networks are discussed to compare TASNEM algorithm with some previous algorithms.

1. Introduction

An interconnection network system involves connecting multiple independent computers, often called nodes, interconnected by a message-passing network. Each node has a processor, a set of input/output devices and a large local memory. New-generation interconnection networks use hardware routers to enter, leave, and pass messages, where each node is attached to each router. Several pairs of channels connect each router to neighboring routers; the pattern in which they are connected defines the network topology [1]. Torus networks are widely used in high performance parallel computers because of their many properties including constant node degree, constant length channel wires, higher channel bandwidth, lower contention latency, and can be partitioned into meshes. In torus networks, wraparound channels have been added to connect each edge node to the corresponding node on the opposite edge. Under random traffic, the symmetry of torus networks leads to a more balanced utilization of communication channels

[*] Tel.: +2 0846344264.
E-mail addresses: maa49@fayoum.edu.eg, mabaky@yahoo.com

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

than in mesh networks [2]. Torus networks have been implemented in several researches and commercial multicomputers systems such as the Torus Routing Chip [3], Cray T3E [4], and the Cray XT3 system [5].

The messages interchanged among nodes across the network following paths achieving the rules of a routing algorithm. A routing algorithm is a process of transmitting messages from the source node to the destination node in a given system. The advance of messages will be in either wormhole [1,2,6] or virtual cut-through switching [7,8]. In wormhole switching, a message is decomposed into a number of small pieces, called flits. A flit is the smallest unit of information that a channel can accept or refuse. The header flit governs the routing and remaining flits follow the header in a pipeline fashion. The switches only require very small buffers and message latency is almost insensitive to the *transmission* distance in the network. In virtual cut-through switching, a packet is stored at an intermediate node only if the next required channel is busy. In case the header cannot proceed, it waits in the current router and all the following flits subsequently draw in, possibly releasing the channels occupied so far [7].

The multicast communication, in which one source node sends the same message to multiple destination nodes, is essential in a wide variety of parallel applications. An effective multicast routing should minimize communication latency which consists of startup latency, network latency, and blocking latency [9]. The startup latency is the amount of time incurred by the operating system in preparing the message for injection into the network. The network latency consists of channel propagation and router latencies. The blocking latency accounts for all delays associated with contention for routing resources among the various worms in the network. Existing multicast routings can be classified as unicast-based [10], path-based [1,11,12] and tree-based [7,13]. In unicast-based algorithms, the multicast operation is performed by sending a separate copy of a message from a source to every destination or, alternatively, by sending unicast messages to subset of destinations. The drawback of this scheme is the fact that multiple copies of the same message are injected into the network, leading to increased traffic in the network. Furthermore, each copy of the message loses considerable startup latency at the source.

In path-based algorithms, a source node prepares a message for delivery to a set of destinations by first sorting the addresses of the destinations in the correct order in which they are visited in the path, and then placing this sorted list in the header of the message [14]. They use a simple hardware mechanism to allow routers to absorb flits on internal channels (to the local processor) while simultaneously forwarding copies of the flits on output channels enroute to the remaining destinations. When the header entered a router with address A, the router checked to see if A is the next address in the header. If so, the address A is removed from the message header and a copy of data flits will be forwarded both to the local core and the next node on the path. Otherwise, the message is forwarded only to the next node on the path. The path-based multicast algorithms are inefficient because the network has to be traversed multiple times, and flits of the message have to be copied and forwarded by the network interface associated with the nodes.

Tree-based algorithms attempt to deliver the message to all destinations in a single multi-head worm that splits at some routers and replicates the data on multiple output ports [13]. In the literature, two approaches have been proposed for the replication of data in tree-based schemes. Synchronous replication requires that all branches of the multi-head worm proceed in lock-step. Thus any branch of the multidestination that is blocked can block all other branches. In asynchronous replication, different heads of a multi-head worm can progress independently through the network. Bubble flits are inserted where necessary obviating the need for a hardware synchronization mechanism [13]. Asynchronous replication may be preferred for a practical implementation because blocked branches do not block other branches. Many current day routers already provide relatively large buffers to prevent deadlock issue.

The critical issue to implement multicast techniques is deadlock. Deadlock in the interconnection network occurs when a set of messages is blocked forever because each message in the set holds one or more resources needed by another message in this set. The development of theories and methodologies for deadlock-free routing in wormhole networks has been a fruitful research area for many years. Duato [15] defined a necessary and sufficient condition guaranteeing the absence of deadlock. The methodology associated with this condition starts from a deadlock-free routing function and allows extensions provided that messages eventually are routed back to and stay in the deadlock-free subnetwork.

This paper is organized as follows. Section 2 presents related research. Section 3 presents the proposed multicast algorithm. Section 4 compares the performance of the proposed algorithm to some well known existing multicast algorithms. Finally, conclusion is given in Section 5.

2. Related research

Various multicast routing algorithms have been proposed for torus network topologies [2,6,8,11,16–23]. Some multicast routing algorithms for 2D torus networks [18–23] are described in this section. In [18,19], two path-based multicast algorithms for 2D torus networks, General Torus Two-Phase Multicast, GTTPM and Torus Two-Phase Multicast, TTPM, were proposed. The two algorithms use some communication services to forward a message from a source node to some destination nodes. They divide the 2D torus network into two 2D meshes depending on the position of the source node. TTPM algorithm uses the vertical wraparound channels of the torus network while GTTPM algorithm uses the horizontal wraparound channel of the torus network. TTPM algorithm cannot include all the destination nodes if the x-dimension is larger than the y-dimension of the torus network while GTTPM algorithm overcomes this problem. The two algorithms require at most two startup times. At the first startup time, the message is sent to a set of nodes such that all destination nodes can be reached in the first or the second phase of communication. They define a path called the Main Path (MP) that is constructed in the first phase starting from the source node to the end node (a special node is determined according to special rules) and contains two parts at most (horizontal and vertical parts). At the second startup time, some of the intermediate nodes along MP retransmit messages to the rest of the multicast destinations; hence all other destination nodes are guaranteed to be covered by the intermediate nodes during the second startup time. The Main Path (MP) is selected such that it conforms to the base routing algorithm and covers as many destinations as possible. This feature allows the multicast operations to be completed in at most two communication steps in 2D Torus. A message is forwarded by a special routing function to a neighboring node if the current node is not a destination node. If the current node is a destination, the associated multicast flag is used to determine the multicast operation to be performed.

In [20], a path-based multicast algorithm for 2D torus networks, Torus with Two Wraparound channels, T2W, was proposed. T2W algorithm defines a path called the horizontal main path starts from the source node to the last node such that the nodes on the horizontal main path can cover all destination nodes on columns of the torus network. In the first startup step, the horizontal main path is selected such that it may use the horizontal wraparound channels to cover as many destinations as possible. The message is sent from the source node to the last node of the horizontal main path according to a deterministic routing function which supplies a unique minimal path. In the second startup step, some intermediate nodes along the horizontal main path (HMP) retransmit the message to the remaining destination nodes through vertical paths. In T2W algorithm, the multicast is divided into sub-multicasts that can be carried out in parallel by many independent paths which branch from HMP.

Some multicast routing algorithms for 2D torus networks based on the Hamiltonian path model were proposed [21–23]. In [21], two multicast routing algorithms, the uniform routing and the fixed routing, were proposed. The two algorithms use the same underlying routing function while they are different in the message preparation algorithms. The message preparation for the uniform routing easily balances two routing paths on which the two messages are routed. The destination nodes are partitioned into two subsets containing a near equal number of destination nodes according to their positions in the Hamiltonian cycle. The message preparation for the fixed routing involves restricting the two routing paths with a maximum path length. The performance of the fixed and uniform routing algorithms is almost the same in 2D torus networks.

In [22], the multipath-HCM routing algorithm divides the torus network into two subnetworks. The high-channel network (low-channel network) contains all of the directional common channels with nodes labeled from low to high (from high to low) and the directional boundary channels with nodes labeled from high to low (from low to high). The algorithm generates four outgoing messages to implement a multicast. So, the destination nodes are partitioned into four destination subsets each of them contained in the header of the message. Two messages are assigned to route along the high-channel network (to up) and the other two messages are assigned to route along the low-channel network (to down). The first flit of the message contains the offset of the next destination. The second flit contains the number of remaining destinations. The later flits contain the offsets for each of these destinations. The algorithm uses a routing function in terms of the (x, y)-coordinates of the current node and destination node, rather than in terms of node labels.

In [23], a multicast routing algorithm, RG, for 2D torus networks based on the Hamiltonian path model was proposed. RG algorithm divides the 2D torus network into disjoint subnetworks and divides the destination nodes into many groups. It requires two communication startup steps to multicast the same message to multiple destination nodes. In the first startup step, the nearest destination node to the source node is selected to become the leader of its destination group. The message is sent from the source node to the leaders such that all the destination nodes can be reached in the first or the second startup of communication. In the second startup step, the leader in each destination group retransmits the message to all remaining destination nodes in its own group.

These previous multicast algorithms are path-based which have been shown to lead to a deadlock in unidirectional multistage interconnection networks. Also, the path length becomes a dominant factor in these algorithms, leading to high latency. To fully utilize the channels uniformly and reduce the latency, this paper presents a new tree-based multicast algorithm for 2D torus networks, in which the router simultaneously sends the incoming flits on more than one outgoing channel. The proposed scheme is a Tree-based Algorithm which Splits 2D torus Networks into two Equally Meshes, hence it is called TASNEM.

3. The proposed algorithm (TASNEM)

In this section, the proposed algorithm, TASNEM, is explained. A common problem associated with most existing multicast algorithms is that they can overload the selected multicast path and hence cause traffic congestion. To avoid this problem, TASNEM algorithm uses the vertical wraparound channels to divide the torus network into nearly two equally size 2D mesh subnetworks. The first mesh subnetwork contains the nodes which their y-coordinates are between y_s and $y_s \pm \lceil n/2 \rceil$, where y_s is y-coordinate of the source node and n is the number of rows of the torus network. The second mesh subnetwork contains the remaining nodes of the torus network. It entails shifting the origin of torus network so that the source node always appears to be in (or closer to) the center of torus network. This is possible because the torus network has a symmetric topology. TASNEM algorithm uses the advantage of this structure to distribute the traffic load over these meshes. These meshes, in turn, implement the multicast independently in a parallel fashion, and nearly balance the traffic load in the network to avoid the congestion problem, and thus considerably reduce the overall communication latency. This leads to an important issue of making each mesh subnetwork less dependent on the other one.

TASNEM algorithm is a tree-based multicast routing technique for 2D torus networks. It delivers the message from the source node to all destination nodes in a single multi-head worm that splits at some routers and replicates the data on multiple output ports. It allows two messages simultaneously going out from the source node to implement the multicast. The source node prepares the message for delivery to the destination nodes and places their addresses in the header flits of the message. In each mesh subnetwork, the main message path starts at the source node and forwards to the nodes which have labels higher (or lower) than the label of source node. At the nodes which their vertical neighboring nodes are destinations, several horizontal message paths may be branched from the main message path to deliver the message to the destination nodes in the same horizontal level.

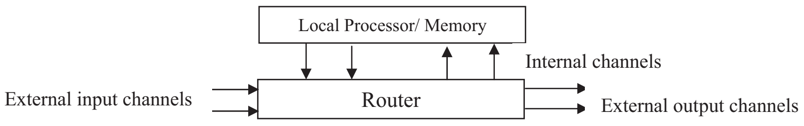

Figure 1 Router architecture.

3.1. System model

In this work, the nodes of torus networks contain dedicated routers to decouple computation and communication functionality in order to improve the performance of them. The basic router architecture of a 2D torus is illustrated in Fig. 1. One or more pairs of internal channels connect a router to its local processor/memory. One channel of each pair is for input and the other for output. Several pairs of external channels connect the router to neighboring routers. The router provides communication services to the host processor. The incoming/ outgoing internal channels to/from the router are usually referred as injection/consumption channels. The number of injection/consumption channels implies the number of messages that can be sent/received concurrently by the processor. In addition, two messages may be transmitted simultaneously in opposite directions between neighboring routers. Also, it is assumed that the torus network adopts all-port wormhole switching and supports an intermediate reception (IR) [12]. In the all-port model, routers are able to transmit multiple messages simultaneously. It is also assumed that there is an internal channel for each external channel, and that all the internal channels are used simultaneously to send and/or receive messages. The IR capability allows a router to deliver an incoming message to the local processor while simultaneously forwarding it to another router. In this manner, a single packet can have several destination addresses in header flits and every intermediate router whose address matches a destination address in the header performs a forward-and-absorb operation: it forwards the pipeline of flits to the next intermediate router while copying simultaneously the data flits into the local processor memory. The router must also remove the corresponding address flit from the header. If the leading destination address does not match the local router address, it just forwards the flits toward the destination. The last destination router absorbs the flit stream and removes it from the network by storing it in the local memory [22].

A 2D $m \times n$ torus network is denoted by $T_{m\times n}$, where m represents the number of columns and n represents the number of rows. Two nodes $p_i = (x_i, y_i)$ and $p_k = (x_k, y_k)$ are called neighboring nodes (i.e., connected by a communication channel) if $|x_k - x_i| + |y_k - y_i| = 1$. The path-based algorithms are established as the Hamiltonian path strategy. The Hamiltonian path is an undirected path that visits each node in a graph exactly once [1]. In this strategy, a node labeling function is used to construct a Hamiltonian path in a 2D torus network. The Hamiltonian path starts at node with label 0, follows the node with labels $1, 2, \ldots$, to the node with label $mn - 1$. The labeling function assigns a unique number to each node. It is given by the following:

$$Q(p) = \begin{cases} y \times n + x & \text{if } y \text{ is even} \\ y \times n + n - x - 1 & \text{if } y \text{ is odd} \end{cases} \forall\, p = (x, y) \in T_{m\times n}, 0 \le x < m, \quad 0 \le y < n \tag{1}$$

According to this node labeling, the torus network is viewed as two subnetworks. A high-channel subnetwork contains all of the channels whose direction is from lower-labeled nodes to higher-labeled nodes, and a low-channel subnetwork contains all of the channels whose direction is from higher-labeled nodes to lower-labeled nodes. The destinations are placed into two groups. One group contains all the destinations that could be reached using the high-channel subnetwork, and the other contains the remaining destinations that could be reached using the low-channel subnetwork. TASNEM algorithm uses the node labels and (x, y)-coordinates of the nodes to divide the set of destination nodes into two subsets.

3.2. Destination preprocessing algorithm

In this subsection, the destination preprocessing algorithm is explained. The source node prepares the message body and a list of destination nodes and executes the destination preprocessing algorithm, *Dest_Prep*, Fig. 2. The algorithm *Dest_Prep* uses the vertical wraparound channels to divide the set of destination nodes into two subsets which are later sent to the router of the source node. It uses the procedure *Divide_Dest1* to divide the set of destination nodes, D, into two lists D_1 and D_2 according to the terms of node labels of the source node and destination nodes.

According to the position of the source node, there exist the following two cases:

Case 1: If y-coordinate of the source node is less than $\lfloor n/2 \rfloor$, then the list D_1 contains the destination nodes with labels are between the label of source node (x_s, y_s) and the label of node $(m - 1,\ y_s + \lceil n/2 \rceil)$ and the list D_2 contains the remaining destination nodes.

Case 2: If the y-coordinate of the source node is greater than or equal to $\lfloor n/2 \rfloor$, then the list D_1 contains the destination nodes with labels are between the label of node (0, $y_s - \lceil n/2 \rceil$) and the label of source node (x_s, y_s) and the list D_2 contains the remaining destination nodes.

The *Dest_Prep* algorithm uses the function *SendS*(M, D, d_x, d_y) to make the processor of source node sends the message body, the list of destination nodes, the horizontal direction d_x, and the vertical direction d_y, to its router. The proposed multicast routing algorithm allows two packets simultaneously going out from the source node. So, the *Dest_Prep* algorithm calls the function *SendS* twice, one for each mesh subnetwork.

```
Procedure Divide_Dest1 (D, x_b, y_b, x_e, y_e):
Input: A destination list D, coordinates of two nodes b and e.
Output: Two destination lists, D_1 and D_2.
Begin
let D_1=Φ, D_2=Φ      // empty lists
for (i=0; i < D.length; i++)
 if (Q(x_b, y_b)< Q(D[i])<Q(x_e, y_e) then add(D_1, D[i]) //add item no. i in D to end of D_1
else add(D_2, D[i])  //add item no. i in D to end of D_2
End.
Dest_Prep Algorithm:
Input: A source node s=(x_s,y_s), a message M, a destination set D.
Output: The processor of source node sends the message at most twice to its router.
Begin:
Let the torus dimensions are m, n.
1- if (y_s is even) then d_x = 1 else d_x = -1
2- if (y_s<⌊n/2⌋ then
        d_y =1
        Divide_Dest1(D, x_s, y_s, m-1, y_s+⌈n/2⌉)
   else
        d_y =-1
        Divide_Dest1(D, 0, y_s-⌈n/2⌉, x_s, y_s)
   endif
3- if (D_1≠Φ) SendS(M, D_1, d_x, d_y)
4- if (D_2≠Φ) SendS(M, D_2, d_x, -d_y)
End.
```

Figure 2 The source node algorithm.

3.3. The message forwarding algorithm

This subsection explains the message forwarding algorithm, *Mess_Forward*, Fig. 3. It is used to determine the next node(s) of the message path and to forward the message to it (them). The *Mess_Forward* algorithm is executed at the router of the source node at most twice. One execution is to forward the message to the destination nodes in the list D_1. The other execution is to forward the message to the destination nodes in the list D_2. Also, the algorithm is executed at the router of each intermediate node of the message path. The *Mess_Forward* algorithm uses the previous procedure *Divide_Dest1*(D, x_b, y_b, x_e, y_e) to divide the set of destination nodes D into two lists D_1 and D_2. The list D_1 contains the destination nodes with labels are between the labels of nodes b (x_b, y_b) and e (x_e, y_e) and the list D_2 contains the remaining destination nodes.

Also, the *Mess_Forward* algorithm uses the procedure *Divide_Dest2*(D) to divide the set of destination nodes D into two lists D_1 and D_2 according to terms of the (x, y)-coordinates of the current node and destination nodes, rather than in terms of node labels. The list D_1 contains the destination nodes in the boundary rows (nodes with y-coordinates equal to 0 or $n-1$) with x-coordinates are less than x-coordinate of the current node. The list D_2 contains the remaining destination nodes. The algorithm uses the function $Send(M, D, d_x, d_y, u)$ to make the router of current node sends the message body, the set of destination nodes, the horizontal direction d_x, and the vertical direction d_y, to the router of node u. Finally, the algorithm uses the following function to determine y-coordinate of the vertical neighboring node of the current node:

$$R(y_c, d_y) = \left\{ \begin{array}{ll} 0 & \text{if } (d_y = 1) \wedge (y_c = n-1) \\ n-1 & \text{if } (d_y = -1) \wedge (y_c = 0) \\ y_c + 1 & \text{if } (d_y = 1) \wedge (y_c < n-1) \\ y_c - 1 & \text{if } (d_y = -1) \wedge (y_c > 0) \end{array} \right\} \quad (2)$$

Upon receiving the message, the router of each current node determines whether it is a destination node. If so, it is removed from the list of destination nodes and sends the message to its processor. At this point, if the set of destination nodes is not empty, the algorithm continues according to this scenario:

Each current node in the message path determines its vertical neighboring node and computes the set D_1 which contains the horizontal destination nodes which are between the current node and its vertical neighboring node. If the set D_1 is empty then the current node forwards the message together with the set of destination nodes to its vertical neighboring node. If the set D_1 is not empty then the current node examines its vertical neighboring node. In case where the vertical neighboring node is not a destination, the current node forwards the message together with the set of destination nodes to its horizontal neighboring node. If the vertical neighboring node is a destination, then the current node forwards the message together with the set of destinations D_1 to its horizontal neighboring node and forwards the message together with the remaining destination nodes to its vertical neighboring node.

Lemma. *The time complexity of TASNEM algorithm is $O(3n)$.*

Proof. TASNEM multicast routing is completed by two algorithms: *Dest_Prep* and *Mess_Forward*. Assume n is the number of destination nodes. Hence,

```
Procedure Divide_Dest2(D):
Input: A destination list D.
Output: Two destination lists, $D_1$ and $D_2$
Begin
Let x-coordinate of the current node =$x_c$ and the vertical torus dimension= $n$
Let $D_1=\Phi$, $D_2=\Phi$
for (i=0; i < D.length; i++)
if ($x_{D[i]}<x_c$) ^ ($y_{D[i]}=0$ or $y_{D[i]}=n-1$)) then add($D_1$,D[i])
 else add ($D_2$, D[i])
End.
Mess_Forward Algorithm:
Input: A message M, a destination list D, a horizontal direction $d_x$, a vertical direction $d_y$.
Output: The router of the current node sends the message to at most two its neighboring node(s).
Begin:
Let the current node c=($x_c$, $y_c$) and the torus dimensions are $m$, $n$.
1- if (c $\in$ D) then          // search
     Remove(D, c)          // remove the address of the node c from the destination list
     Send the message M to the processor
   endif
2- if (D=$\Phi$) then stop.
3- $y_v$ = R($y_c$, $d_y$)
4- if ($d_y$=1) then   // (high-channel subnetwork)
       if ($y_c=n-1$) then Divide_Dest2(D)
       else Divide_Dest1(D, $x_c$, $y_c$, $x_c$, $y_v$)
  else    // $d_y$=-1 (low-channel subnetwork)
      if ($y_c$=0) then Divide_Dest2(D)
      else Divide_Dest1(D, $x_c$, $y_v$, $x_c$, $y_c$)
  endif
5- if ($D_1=\Phi$) then Send(M, D, -$d_x$, $d_y$, ($x_c$, $y_v$))
   else   // $D_1 \neq \Phi$
   if(($x_c$, $y_v$)$\notin$ D) then Send(M, D, $d_x$, $d_y$, ($x_c+d_x$,$y_c$))
    else
     Send(M, $D_1$, $d_x$, $d_y$, ($x_c+d_x$, $y_c$))
     if ($D_2\neq\Phi$) then Send(M, $D_2$, -$d_x$, $d_y$, ($x_c$, $y_v$))
    endif
End.
```

Figure 3 The message forwarding algorithm.

(a) Compute the time complexity of the Dest_Prep algorithm
1. The number of operations in step 1 = 1
2. The number of operations in step 2 = $n + 1$
3. The number of operations in step 3 = 1
4. The number of operations in step 4 = 1

The total number of operations = $n + 4$.
Then, the time complexity is the *Dest_Prep* algorithm O(n). (a.1)

(b) Compute the time complexity of the Mess_Forward algorithm
1. The number of operations in step 1 = n
2. The number of operations in step 2 = 1
3. The number of operations in step 3 = 1
4. The number of operations in step 4 = n
5. The number of operations in step 5 = n

The total number of operations = 3 n + 2.
Then, the time complexity is the *Mess_Forward* algorithm O(3n). (b.1)

From (a.1) and (b.1), the time complexity is the TASNEM algorithm O(3n).

Notice, the size of the destination set D is decreased by 1 at each destination node. That is, the size of the destination set D at the first destination node is n, at the second destination node is $n - 1$, at the third destination node is $n - 2$, and at the last destination node is 1. Then, the average value of the size of the destination set D equals $(1 + 2 + \cdots + n)/n = (n + 1)/2$. So, the time complexity of TASNEM algorithm is O(3) at the last destination node, is O(3(n + 1)/2) at the middle destination node, and is O(3n) at the first destination node. Therefore, the time complexity of TASNEM algorithm is O(3n) in the worst case. □

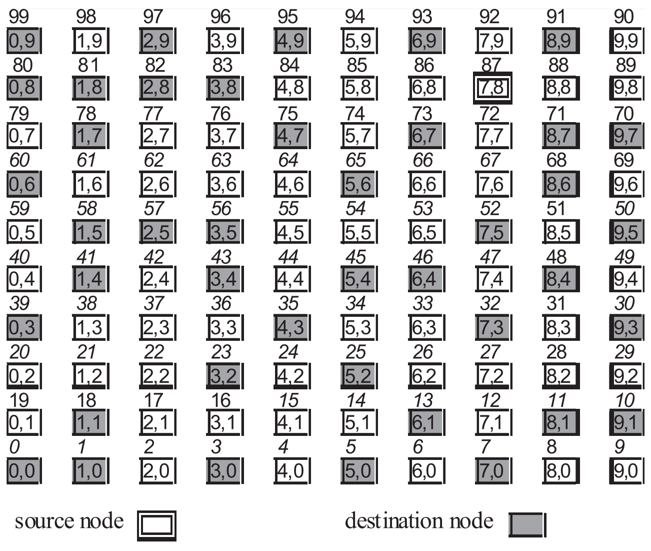

Figure 4 An example of 2D torus network, $T_{10\times10}$.

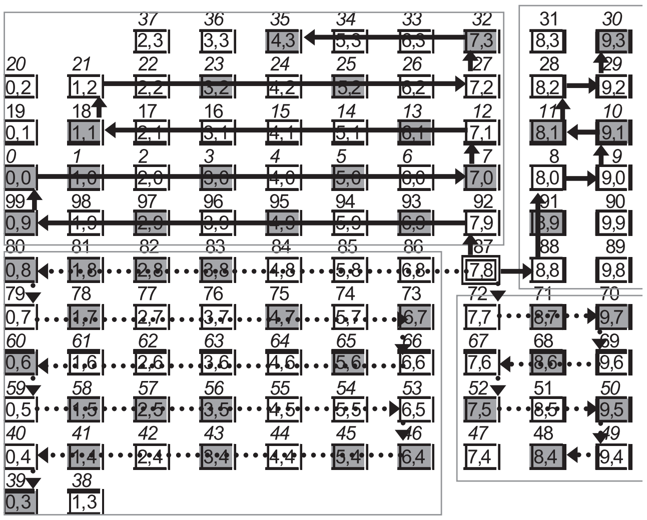

Figure 5 Message multicasting by using HCM algorithm.

3.4. A case study

This subsection introduces a case study to compare TASNEM algorithm with the previous algorithms, multipath-HCM algorithm [22] and T2 W algorithm [20]. Two commonly parameters, the network latency and the network traffic are used. The network latency is equal to the longest message transmission path involved. The network traffic is equal to the number of channels used to deliver all messages involved. The torus network, $T_{10\times10}$, in Fig. 4 is considered and for simplicity, the torus networks will be drawn without channels. Fig. 5 illustrates the message multicasting by using the multipath-HCM algorithm. The source node (7,8) generates four copies of the message each of them contains one of the destination sets, D^{h1}, D^{h2}, D^{l1}, and D^{l2}. The number of channels used to deliver the message to the destination nodes in D^{h1} is 9, to the destination nodes in D^{h2} is 34, to the destination nodes in D^{l1} is 36, and to the destination nodes in D^{l2} is 11. The maximum path length is max(9,34,36,11) = 36. Then, the network latency and traffic of the multipath-HCM algorithm are 36 and 90, respectively.

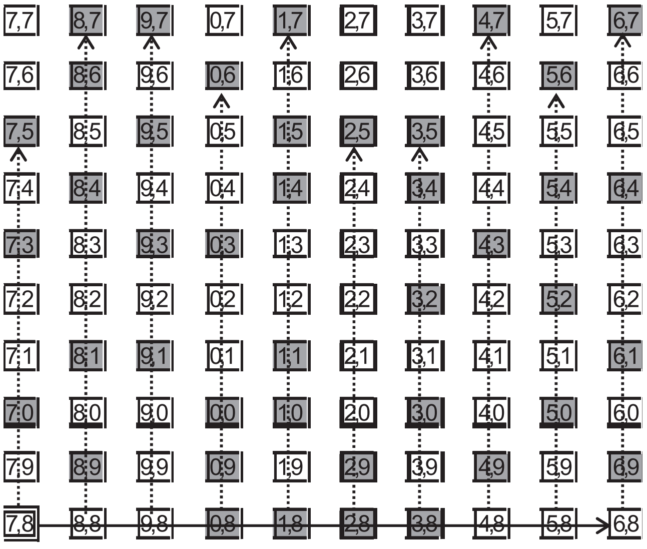

Figure 6 Message multicasting by using T2W algorithm.

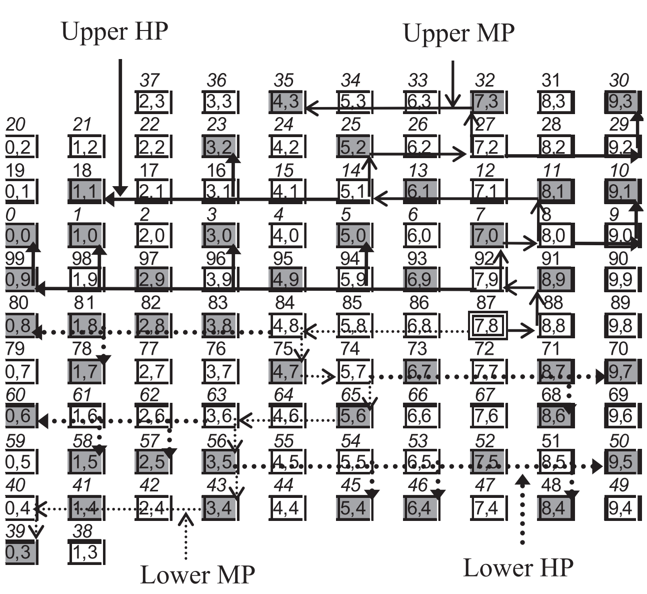

Figure 7 Message multicasting by using TASNEM algorithm.

Fig. 6 illustrates the message multicasting by using T2 W algorithm. In the first startup step, the message is sent from the source node to the last node of the horizontal main path (6,8). In the second startup step, the nodes along the horizontal main path retransmit the message to the remaining destination nodes through vertical paths. The solid and dotted lines represent the message paths in the first and the second phases, respectively. The network latency and the network traffic of T2W algorithm are 18 and 82, respectively.

Fig. 7 illustrates the message multicasting by using TASNEM algorithm. The source node generates two copies of the message each of them contains one of the destination sets, D_1 and D_2. The solid and dotted lines represent the message paths in the upper and lower subnetworks, respectively.

The thin line represents the main message path and the thick (bold) lines represent the horizontal message paths in each subnetwork. The longest path of channels used to deliver the message to the destination nodes in D_1 is 16 and to the destination nodes in D_2 is 15. The total number of channels used to deliver the message to the destination nodes in D_1 is 37 and to the destination nodes in D_2 is 38. Hence, the network latency and the network traffic of TASNEM algorithm are 16 and 75, respectively.

Clearly, the network latency and the network traffic obtained by TASNEM algorithm are less than the network latency and the network traffic obtained by HCM and T2W algorithms.

4. A performance evaluation

In this section, the performance of TASNEM algorithm is compared with the previous multicast algorithms T2W algorithm [20], Uniform algorithm [21], and the multipath-HCM algorithm [22]. The network latency time between any two

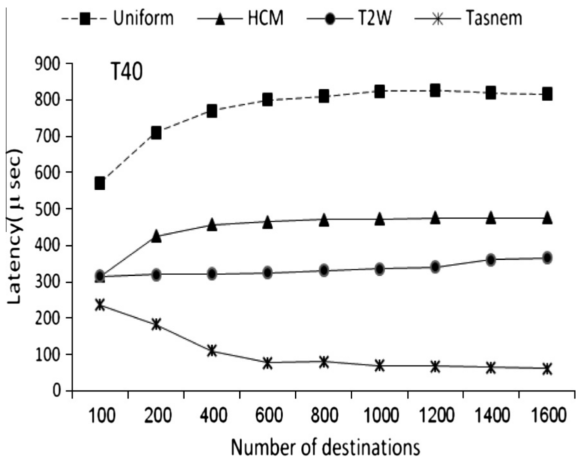

Figure 8 Network latency versus no. of destinations.

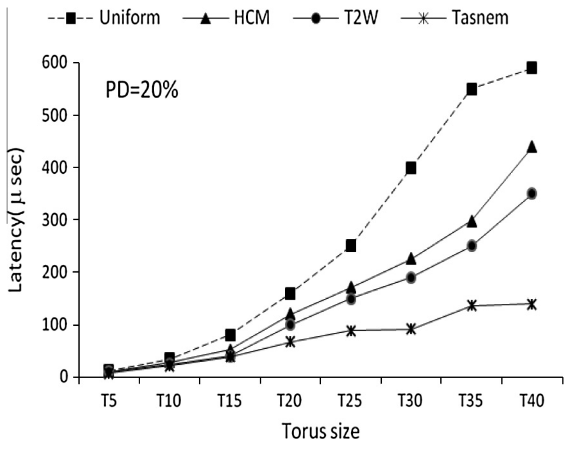

Figure 10 Network latency versus Torus size.

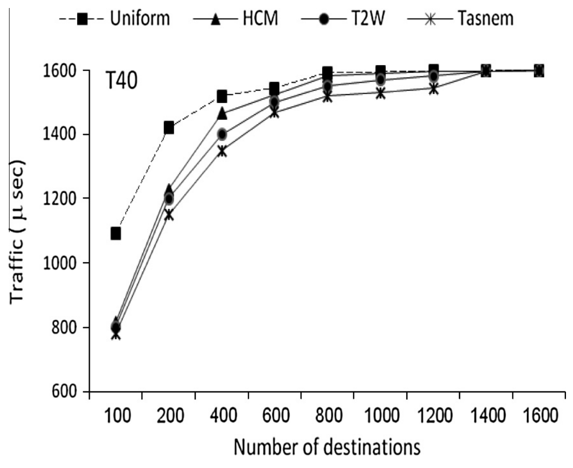

Figure 9 Network traffic versus no. of destinations.

adjacent nodes has been set at 25*ns* [21]. Various numbers of random networks that contain two virtual channels per each physical channel are used. In the following two subsections, the network latency and the traffic obtained by of the four algorithms are plotted versus the next parameters. The parameter Tn refers to the torus network size, and *n* is ranging between 5 and 40, for example, T5 means that the torus network with 5 columns and 5 rows. The parameter PD refers to the percentage of destination nodes to the total number of nodes in the torus network.

4.1. Effect of the number of destinations

Figs. 8 and 9 plot the network latency and the network traffic obtained by the algorithms versus various values of the number of destination nodes, which is ranging between 100 and 1600. In Fig. 8, the network latency obtained by TASNEM algorithm decreases when the number of destinations nodes increases. This is due to the fact that TASNEM algorithm is a tree-based and it searches for the nearest destination node of each current node. Therefore, as the number of destination nodes increases, the destination nodes may become closer to each current node. The network latency obtained by the other algorithms increases when the number of destination nodes increases. This is due to the fact that these algorithms are path-based and visit all destination nodes to reach the last one in each subnetwork. Therefore, as the number of destination nodes increases, the last destination node becomes far away from the sender. In Fig. 9, as the number of destination nodes increases, the network traffic obtained by all algorithms increases.

4.2. Effect of the network size

Figs. 10 and 11 plot the network latency and the network traffic obtained by the algorithms versus the size of torus network which is ranging between 25 and 1600 nodes. The percentage of destinations (PDs) is equal to 20%. In Fig. 10, as the torus size increases, the network traffic obtained by all algorithms increases. This is because as the network size increases, the PD value increases and hence the latency value increases. Cleary, the increasing rate of the traffic curve obtained by TASNEM algorithm is very less than the increasing rate of the traffic curves obtained by the other algorithms. In Fig. 11, as the torus size increases, the network traffic obtained by all algorithms increases.

Cleary, in all tested cases, the network latency and the network traffic obtained by TASNEM algorithm are less than the

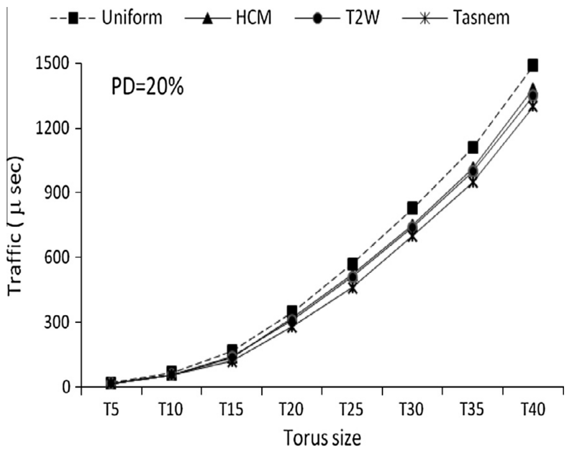

Figure 11 Network traffic versus Torus size.

network latency and the network traffic obtained by the other algorithms. Also, the network latency and the network traffic obtained by Uniform algorithm [21] are more than the network latency and the network traffic obtained by the other algorithms.

5. Conclusion

In this paper, a multicast tree-based routing in 2D torus networks TASNEM was proposed. TASNEM algorithm uses the vertical wraparound channels to divide the torus network into nearly two equally size 2D mesh subnetworks. It requires at most two communication start-up steps to multicast to any member of destinations. The main message path of TASNEM algorithm starts at the source node and forwards to the nodes which have labels higher (or lower) than the label of source node. At the nodes which their vertical neighboring nodes are destinations, horizontal message paths may be branched from the main message path to deliver the message to the destination nodes in the same horizontal level. The performance of TASNEM algorithm was evaluated through comparing it with the other algorithms, T2W algorithm [20], Uniform algorithm [21], and the multipath-HCM algorithm [22]. The simulation results showed that the multicast performance of TASNEM algorithm is better than that of the other algorithms. The further study will focus on extending the proposed multicast algorithm to 3D torus networks.

References

[1] Lin X, McKinley PK, Ni LM. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. IEEE Trans Parall Distrib Syst 1994;5(8):793–804. http://dx.doi.org/10.1109/71.298203.

[2] Robinson DF, McKinley PK, Cheng BHC. Optimal multicast communication in wormhole-routed torus networks. IEEE Trans Parall Distrib Syst 1995;6(10):1029–42. http://dx.doi.org/10.1109/71.473513.

[3] Dally WJ, Seitz CL. The torus routing chip. J Distrib Comput 1986;1(3):187–96.

[4] Anderson E, Brooks J, Grassl C, Scott S. Performance of the cray T3E multiprocessor. In: Proceedings of the 1997 ACM/IEEE conference on supercomputing; 1997. p. 1–17 http://dx.doi.org/10.1109/SC.1997.10043.

[5] Cray Research Inc, "CRAY XT3 scalable parallel processing system," Cray Research Inc.; 2005. < http://www.cray.com-/products/xt3/index.html > .

[6] Verbeek F, Schmaltz J. A decision procedure for deadlock-free routing in wormhole networks. In: IEEE transactions on parallel and distributed systems, July 2013. IEEE computer Society Digital Library. http://dx.doi.org/ieeecomputersociety.org/10.1109/TPDS.2013.121.

[7] Sivaram R, Panda DK, Stunkel CB. Multicasting in irregular networks with cut-through switches using tree-based multidestination worms. In: Proceeding PCRCW'97, 35-48 (June, Georgia, 1997). http://dx.doi.org/10.1007/3-540-69352-1_4.

[8] Luo W, Xiang D. An efficient adaptive deadlock-free routing algorithm for torus networks. IEEE Trans Parallel Distrib Syst 2012;23(5):800–8. http://dx.doi.org/10.1109/TPDS.2011.145.

[9] Imre KM, Baransel C, Artuner H. Efficient and scalable routing algorithms for collective communication operations on 2D all-port torus networks. Int J Parallel Prog 2011;39(6):746–82. http://dx.doi.org/10.1007/s10766-011-0169-2.

[10] McKinely P, Xu H, Esfahanian AH, Ni L. Unicast-based multicast communication in wormhole-routed networks. IEEE Trans Parall Distrib Syst 1994;5:1252–65. http://dx.doi.org/10.1109/71.334899.

[11] Robinson DF, Mckinley PK, Cheng BHC. Path-based multicast communication in wormhole-routed unidirectional torus networks. J Parall Distrib Comput 1997;45(2):104–21. http://dx.doi.org/10.1006/jpdc.1997.1372.

[12] Fleury E, Fraigniaud P. Strategies for path-based multicasting in wormhole routed meshes. J Parall Distrib Comput 1998;53:26–62. http://dx.doi.org/10.1006/jpdc.1998.1473.

[13] Hadas RL, Mazzoni D, Rajagopalan R. Tree-based multicast in wormhole-routed irregular topologies. In: Proceeding of the First IPPS/SPDP; April 1998; p. 244–249. http://dx.doi.org/10.1109/IPPS.1998.669919.

[14] Daneshtalab M, Ebrahimi M, Xu TC, Liljeberg P, Tenhunen H. A generic adaptive path-based routing method for MPSoCs. J Syst Architect 2011;57(1):109–20. http://dx.doi.org/10.1016/j.sysarc.2010.08.002.

[15] Duato J. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. IEEE Trans Parallel Distrib Syst October 1995;6(10):1055–67. http://dx.doi.org/10.1109/71.473515.

[16] Oral S, George Alan D. Multicast performance modeling and evaluation for high-speed unidirectional torus networks. Microprocess Microsyst 2004;28:477–89. http://dx.doi.org/10.1016/j.micpro.2004.04.003.

[17] Hafizur Rahmana MM, Horiguchib Susumu. Routing performance enhancement in hierarchical torus network by link-selection algorithm. J Parall Distrib Comput 2005;65(11):1453–61. http://dx.doi.org/10.1016/j.jpdc.2005.05.024.

[18] Darwish MG, Radwan AA, Abd El-Baky MA, Hamed K. GTTPM –An efficient deadlock-free multicast wormhole algorithm for communication in 2D torus multicomputers. In: Proceeding of the 17th IASTED international conference parallel and distributed computing and systems, phoenix, AZ, USA, November 2005 < http://www.actapress.com/Content_Of_Proceeding.aspx?ProceedingID = 328 > .

[19] Darwish MG, Radwan AA, Abd El-Baky MA, Hamed K. TTPM – an efficient deadlock-free algorithm for multicast communication in 2D torus networks. J Syst Architect 2008;54(10):919–28. http://dx.doi.org/10.1016/j.sysarc.2008.03.004.

[20] Radwan AA, Hamed K, Darwish MG, Abd El-Baky MA. T2W: a multicast routing algorithm for 2D torus networks with horizontal main path model. Int J Intell Comput Inform Sci 2010;10(2):171–82, < http://net2.shams.edu.eg/cis/ijicis/views/abstract.php?cid = 243 > .

[21] Wang N-C, Yen C-P, Chu C-P. Multicast communication in wormhole-routed symmetric networks with Hamiltonian cycle model. J Syst Architect 2005;51(3):165–83. http://dx.doi.org/10.1016/j.sysarc.2004.11.001.

[22] Wang N-C, Hung Yi-Ping. Multicast communication in wormhole-routed 2D torus networks with hamiltonian cycle model. J Syst Archit 2009;55(1):70–8. http://dx.doi.org/10.1016/j.sysarc.2008.09.002.

[23] Darwish MG, Radwan AA, Abd El-Baky MA, Hamed K. Ready groups: a path-based multicast algorithm for 2D torus networks. In: The 7th international conference on informatics and systems (INFOS 2010), NGBN(98-106), Faculty of Computers and Information, Cairo University, Egypt; 2010 < http://infos2010.fci.cu.edu.eg/Sessions_3rd_day/Session_3D.htm > .