TMTM - An Efficient Deadlock-Free Tree-Based Multicast Algorithm for 2D Torus Multicomputer

Kadry Hamed

Dept. of CS, Faculty of Computers and Information, Minia University, Egypt.
Dept. of CS, College of Computing and Information Technology, Shaqra University, KSA.

Abstract

Multicomputer system has high computational power and it can perform multiple tasks concurrently. Torus is a commonly used network topology in building multicomputer system because of many features including low bandwidth, fixed degree of nodes and scalability. The performance of multicomputer highly depends on the underlying network communication such as multicast. This paper suggests an efficient wormhole deadlock-free Tree-based algorithm to Multicast the message in 2D Torus Multicomputer, hence it is called TMTM. TMTM algorithm is designed such that the message can be sent to any number of destinations within two start-up communication times. It allows some intermediate nodes that are not destinations to perform multicast operations. This feature can achieve high degree of parallelism and low communication latency over a wide range of traffic loads and therefore improves the performance. Simulation studies on different torus networks are discussed to compare TMTM algorithm with a previous algorithm.

Keywords: Multicomputer, Parallel System, 2D torus network, Multicast, Deadlock-free, Tree-based, wormhole Algorithm.

I. INTRODUCTION

Nowadays, there are many fields of computer applications, such as aircraft control, telephone networks, robotics, etc. that demand millions of computational operations. Multicomputer that refer to a computer system has multiple processors is used to meet these demands [1, 2]. The computational power of multicomputer is very high, so it can perform multiple tasks concurrently. The overall performance of a system can be improved by passing the message in an efficient manner. Consequently, efficient routing of message is critical to the performance of multicomputer [3, 4]. In wormhole-routed networks, a message is divided into a number of small pieces, called flits. A flit is the smallest unit of information that a channel can accept or refuse. All routing information of the message is kept in the header flit(s) and the data elements are kept in the other flits of the message. So, the header flit governs the routing and remaining flits follow the header in a pipeline fashion [4, 5]. Messages can be passed using multicast method. Multicast refers to a collective communication in which allows a source node to send the same message to an arbitrary number of distinct destinations. Existing multicast routings can be classified as unicast-based [6], path-based [7, 8, 9] and tree-based [10, 11]. Multicast can be implemented through multiple unicast (one to one communications), which is not efficient because the message is sent from the source node to each destination near sequentially. The more efficient and effective ways, path-based and tree-based multicast [1, 8, 9, 11, 12, 13], are to use good strategies to disseminate the message from the source to multiple destinations concurrently. These ways overcome the drawbacks of unicast-based multicast. In path-based algorithms, a message that send to a group of destination nodes is equipped in the source node by first sorting the addresses of the destinations in the right order in which they are visited in the path, and then placing this sorted list in the header of the message [14]. Tree-based multicast algorithms attempt to deliver the message to all destination nodes in a single multi-head worm that splits at some routers and replicates the data on multiple output ports [11]. Tree-based routing has an essential feature that it does not require any arranging of header flits. Furthermore and from the source node, the subpaths to each destination are all optimal. The major impediment of tree-based routing is its high deadlock probability, which can negatively affect performance. Deadlock probability is high since all subpaths must be concurrently allocated to a message in order for data flits to be transmitted. In the interconnection network, deadlock happens when a group of messages is blocked forever because each holding one or more resources needed by another message in this group [15]. Two main parameters used to evaluate multicast routing are the latency takes to deliver the message to all destination nodes and the traffic which refers to the total number of channels involved [16].

Finding the multicast tree which is efficient on both latency and traffic is the essential challenge. Experiments exhibit multicast traffic and latency do not seem to be independent from one another. They dispute each other. The problem is that optimization devoted to one will usually negative affect the performance of the other [17]. As a matter of fact, Lin [18] has proved that finding the optimal multicast tree in mesh-connected networks is NP-complete. So, the multicast problem of seeking optimal results on both traffic and latency is NP-hard. Heuristics are needed to solve the problem. A good approach is to minimize one parameter first, and try to reduce the cost of the other one as much as possible. Since our goal is to develop efficient multicast routing algorithms for time-critical multicomputer, we choose the pro-time approach.
that minimizes the time first. Our problem becomes the Optimal Multicast Tree problem [16, 18]. Torus networks are widely used in high performance multicomputers because of their many features 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 than in mesh networks [19]. Torus networks have been implemented in several researches and commercial multicomputer systems such as the Torus Routing Chip [20], Cray T3E [21], and the Cray XT3 system [22].

This research is organized as follows: previous works are given in section 2. The proposed TMTM algorithm is introduced in section 3. The comparison work of TMTM algorithm against existing well known, TASNEM algorithm [23] is presented in section 4. Finally, conclusion and future works are given in section 5.

II. PREVIOUS WORKS

There are many multicast routing algorithms have been proposed for 2D torus networks [1, 12, 13, 14, 23, 24, 25, 26]. In [12, 13, 24, 26], a path-based multicast algorithms are presented. These algorithms define a main path starts from the source node and extended to a special node such that the nodes on the main path can cover all destination nodes of the torus network. In [23], El-Baky proposed a tree-based algorithm (called TASNEM) for multicasting in 2D torus networks. \( T_{\text{min}} \) where \( n, m \) represent the numbers of columns and rows respectively. It uses the vertical wrap-around channels to divide the torus network into nearly two equally size 2D mesh sub-networks. The first mesh sub-network contains the remaining nodes of the torus network. The second mesh sub-network contains the remaining destination nodes of the torus network. TASNEM algorithm 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 sub-network, 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. Fig. 1 shows a 2D torus network, \( T_{\text{sys}} \), with a source node, \( s=(4,1) \) and a set of random distributed destination nodes in gray color. Fig. 2 shows the message paths on the torus network when TASNEM algorithm is used to send a message from the source node to all destination nodes. The solid and dotted paths represent communication paths of the main paths and branched paths, respectively.

Reducing the latency and traffic of multicasting message are important goals of this research. So, an efficient multicast routing algorithm, TMTM, is introduced. The proposed algorithm is compared with TASNEM algorithm [23]. The simulation results show that the introduced algorithm performs better than TASNEM algorithm.

III. THE PROPOSED ALGORITHM

This section introduces an efficient wormhole deadlock tree-based multicast routing algorithm, TMTM, for all-port 2D tours-connected multicomputer. In this algorithm, the source node has to disseminate the same message to all destination nodes by using two startup latency at most. The vertical wraparound channels maybe used by TMTM algorithm to virtually divide the torus network, \( T_{\text{min}} \) into nearly two equally size 2D virtual meshes, \( N_1 \) and \( N_2 \). \( N_1 \) contains the nodes which their y-coordinates are between \( y_i \) and \( y_i + \left\lfloor m/2 \right\rfloor \), where \( y_i \) is y-coordinate of the source node and \( m \) is the number of rows of the torus network. The second mesh sub-network contains the remaining nodes of the torus network. 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 sub-network, 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. In this algorithm, the source node has to disseminate the same message to all destination nodes by using two startup latency at most. The vertical wraparound channels maybe used by TMTM algorithm to virtually divide the torus network, \( T_{\text{min}} \) into nearly two equally size 2D virtual meshes, \( N_1 \) and \( N_2 \). \( N_1 \) contains the nodes which their y-coordinates are between \( y_i \) and \( y_i + \left\lfloor m/2 \right\rfloor \), where \( y_i \) is y-coordinate of the source node and \( m \) is the number of rows of the torus network. The second mesh sub-network contains the remaining nodes of the torus network. 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 sub-network, 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. Fig. 1 shows a 2D torus network, \( T_{\text{sys}} \), with a source node, \( s=(4,1) \) and a set of random distributed destination nodes in gray color. Fig. 2 shows the message paths on the torus network when TASNEM algorithm is used to send a message from the source node to all destination nodes. The solid and dotted paths represent communication paths of the main paths and branched paths, respectively.

This research is organized as follows: previous works are given in section 2. The proposed TMTM algorithm is introduced in section 3. The comparison work of TMTM algorithm against existing well known, TASNEM algorithm [23] is presented in section 4. Finally, conclusion and future works are given in section 5.

II. PREVIOUS WORKS

There are many multicast routing algorithms have been proposed for 2D torus networks [1, 12, 13, 14, 23, 24, 25, 26]. In [12, 13, 24, 26], a path-based multicast algorithms are presented. These algorithms define a main path starts from the source node and extended to a special node such that the nodes on the main path can cover all destination nodes of the torus network. In [23], El-Baky proposed a tree-based algorithm (called TASNEM) for multicasting in 2D torus networks. \( T_{\text{min}} \) where \( n, m \) represent the numbers of columns and rows respectively. It uses the vertical wrap-around channels to divide the torus network into nearly two equally size 2D mesh sub-networks. The first mesh sub-network contains the nodes which their y-coordinates are between \( y_i \) and \( y_i + \left\lfloor m/2 \right\rfloor \), where \( y_i \) is y-coordinate of the source node and \( m \) is the number of rows of the torus network. The second mesh sub-network contains the remaining nodes of the torus network. TASNEM algorithm 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 sub-network, 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. Fig. 1 shows a 2D torus network, \( T_{\text{sys}} \), with a source node, \( s=(4,1) \) and a set of random distributed destination nodes in gray color. Fig. 2 shows the message paths on the torus network when TASNEM algorithm is used to send a message from the source node to all destination nodes. The solid and dotted paths represent communication paths of the main paths and branched paths, respectively.
Kadry Hamed  

TMTM: An Efficient Deadlock-Free Tree-Based Multicast Algorithm for 2D Torus Multicomputer

reduced. This makes each mesh less dependent on the other one.

The Hamiltonian path defines a solitary label for every node of a torus network, where the first node \( (f_u) \) in the path is labeled 0 (set \( L_u = 0 \)) and the last node \( (l_u) \) in the path is labeled to \( mn-1 (set L_u = mn-1) \). So, TMTM algorithm assigns a label for every node based on the place of that node in a Hamiltonian path. According to this node labeling, the torus network is viewed as two subnetworks: a high-channel network and a low-channel network. The high-channel network contains all of the directional channels with the nodes labeled from low to high numbers. The low-channel network contains all of the directional channels with the nodes labeled from high to low numbers. The destination nodes are divided into two sets. One set includes all the destinations that could be reached using the high-channel network, and the other includes the remaining destination nodes that could be reached using the low-channel network.

Each of TASNEM and TMTM algorithms generate the routing decision at each sending node but they have different message propagation. The routing technique used by TMTM algorithm decreases the message transmission time involved between the source node and the destination nodes. 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. Also, it uses Asynchronous replication approach that replicates the data in tree-based schemes [27]. Two messages simultaneously going out from the source node to implement the multicast. To fully utilize the multicast routing paths supplied by TMTM algorithm, some intermediate nodes that are not destination nodes are permitted to perform multicast operations. This features increase flexibility in distributing messages to the destination nodes thereby improving the performance that is evaluated through simulations.

In TMTM algorithm, the multicast message is prepared for delivery to the destination nodes that their addresses are placed in the message header flits. In each virtual mesh, the main message path begins at the source node and vertically forward to the nodes which have labels higher (or lower) than the label of source node. Accordingly to the Hamiltonian path and for the destination nodes that their labels are between the label of the source node and the label of the upper (or lower) node, several horizontal message paths may be branched from the main message path to send the message for these destination nodes. Depending on the position of the source node, there exist the following two cases:

**Case 1:** If the \( y \)-coordinate of the source node is less than \( \left\lfloor \frac{m}{2} \right\rfloor \), then the destination set \( D_1 \) contains the destination nodes whose labels are greater than the label of the source node \( (x_s, y_s) \) and less than or equal to the labels of nodes whose \( y \)-coordinate is equal to \( y_s + \left\lfloor \frac{m}{2} \right\rfloor \) and the destination set \( D_2 \) contains the remaining destination nodes.

**Case 2:** If the \( y \)-coordinate of the source node is greater than or equal to \( \left\lfloor \frac{m}{2} \right\rfloor \), then the destination set \( D_1 \) contains the destination nodes whose labels are less than the label of the source node \( (x_s, y_s) \) and greater than or equal to the labels of nodes whose \( y \)-coordinate is equal to \( y_s - \left\lfloor \frac{m}{2} \right\rfloor \) and the destination set \( D_2 \) contains the remaining destination nodes.

For simplicity, TMTM algorithm will be constructed only for the first case when \( y_s < \left\lfloor \frac{m}{2} \right\rfloor \) and the other case \( (y_s \geq \left\lfloor \frac{m}{2} \right\rfloor) \) is similar. In fig. 3, TMTM algorithm uses two main procedures. The first one UP-NET\((x_u, y_u, D_v, mess, d_x, d_y)\) takes as inputs, the two coordinates of the source node \( (x_u, y_u) \), \( D_v \) that contains the remaining destination nodes, mess, and \( d_x, d_y \) that is equal to \( 1 \) when the message moves from left to right and \( -1 \) when the message moves from right to left (when \( y_u \) is even according to the Hamiltonian path), and \( d_x = 1 \) when the message moves from down to up, and \( d_y = -1 \) when the message move from up to down. This procedure makes the source node sends the message mess to all destinations of \( D_v \). For do this, it uses three sub-procedures, MULTICAST-MOVE(), LAST-ROW(), and SIDE-PATH(). The first two sub-procedures are used to send the message for other rows, and the third is used to multicast the message through branched paths.

The second main procedure DOWN-WRAP-NET\((x,y, D_v, mess, d_x, d_y)\) is similar to UP-NET() procedure but it multicast the message to the destination nodes of \( D_v \). Also, it uses two sub-procedures MULTICAST-MOVE(), and ACROSS-WRAP-CHANNELS(). The last one is used to multicast the message through the wrap-around channels.

There is a function, SEND-MESS()\((x_u, y_u)\), mess, \((x, y)\) that used to make the router of the current node \((x_u, y_u)\) sends the message mess to the neighboring node \((x, y)\).

At receiving the message, the router of each current node locates whether it is a destination node. If so, it is removed from the destination set of the message header and receives the message to its processor. At this point, if the destination set is not empty, the algorithm continues according to this scenario: starting from the source node and each current node on the main path computes the destination set, \( D_{side} \) which contains the horizontal destination nodes which their labels are between label of the current node and its vertical neighboring node label. If the set \( D_{side} \) is empty then the current node forwards the message together with the set of destination nodes to its vertical neighboring node. If \( D_{side} \) is not empty then the current node forwards the message together with the set \( D_{side} \) to its horizontal neighboring node and forwards the message together with the remaining destination nodes to its vertical neighboring node.
Kadry Hamed

TMTM - An Efficient Deadlock-Free Tree-Based Multicast Algorithm for 2D Torus Multicomputer

**TMTM algorithm** ($x_s, y_s, \text{mess}, D, m$)

**Input:** $m$ is the number of rows in a torus network $T_{\text{torus}}$, $x_s$ and $y_s$ are the coordinates of the source node $(s)$, a destination set $D$, and the message size $\text{mess}$

**Output:** $\forall d \in D$, SEND_MESS($s, \text{mess}, d$)

**Begin:**
1. Let $z = \lceil m/2 \rceil$, $d_0 = 1$
2. IF ($y_s$ is even) THEN $d_0 = 1$ ELSE $d_0 = -1$
3. $D_1 = \{ (x, y) \in D : L(x, y) < L(x_s, y_s) \}$
4. $D_2 = D - D_1$
5. UP_NET($x_s, y_s, D_1, \text{mess}, d_0, d_0$)
6. DOWN_NET($x_s, y_s, D_2, \text{mess}, -d_0, -d_0$)

**END TMTM algorithm**

**Procedure UP_NET**($x, y, D_1, \text{mess}, d_0, d_0$)

**Begin:**
1. WHILE($D_1 \neq \emptyset$) DO
2. IF ($y = n$) THEN
   \{ $D_{\text{data}} = \{ (x, y) \in D_1 : L(x, y) < L(x_s, y_s) \}$
   \{ MULTICAST_MOVE($x, y, D_1, D_{\text{data}}, d_0, d_0$) \}
   \{ to multicast mess across row \}
3. ELSE LAST_ROW($x, y, D_1, \text{mess}, d_0, d_0$)

**End WHILE**

**END UP_NET() procedure**

**Procedure DOWN_NET**($x, y, D_2, \text{mess}, d_0, d_0$)

**Begin:**
1. WHILE ($D_2 \neq \emptyset$) DO
2. IF ($y = 0$) THEN
   \{ $D_{\text{data}} = \{ (x, y) \in D_2 : L(x, y) < L(x_s, y_s) \}$
   \{ MULTICAST_MOVE($x, y, D_2, D_{\text{data}}, d_0, d_0$) \}
   \{ to multicast mess across row \}
3. ELSE ACROSS_NET($x, y, D_2, \text{mess}, d_0, n-1$)

**End WHILE**

**END DOWN_NET() procedure**

**Procedure MULTICAST_MOVE**($x, y, D, D_{\text{data}}, d_0, d_0$)

**Begin:**
SIDE_PATH($x, y, \text{mess}, D_{\text{data}}, d_0, d_0$)
SEND_MESS($x, y, \text{mess}, (x, y-d_0)$)
IF ($x+y+d_0 \in D$) THEN $D = D - ((x, y+d_0))$
BEGIN $y = y+d_0$, $d_0 = -d_0$
Return($D, y, d_0$)

**End MULTICAST_MOVE() procedure**

**Procedure LAST_ROW**($x, y, D, \text{mess}, d_0, d_0$)

**Begin:**
$D_{\text{data}} = D$
SIDE_PATH($x, y, \text{mess}, D_{\text{data}}, d_0, d_0$)
$D = \emptyset$
Return($D$)

**End LAST_ROW() procedure**

**Procedure ACROSS_WRAP_CHANNEL**($x, y, D, \text{mess}, d_0, d_0$)

**Begin:**
$D_{\text{data}} = \{ (x, y) \in D : L(x, y) < L(x_s, y_s) \}$
IF ($x+y+d_0 \in D$) THEN $D = D - ((x, y+d_0))$
BEGIN $y = y+d_0$, $d_0 = -d_0$
Return($D, y, d_0$)

**End ACROSS_WRAP_CHANNEL() procedure**

**Procedure SIDE_PATH**($x, y, \text{mess}, D, d_0, d_0$)

**Begin:**
WHILE ($D \neq \emptyset$) DO
SEND_MESS($x, y, \text{mess}, (x+d_0, y)$)
IF ($x+y+d_0 \in D$) THEN $D = D - ((x, y+d_0))$
BEGIN $y = y+d_0$, $d_0 = -d_0$
**End WHILE**

**End SIDE_PATH() procedure**

**Figure 3. Multicast TMTM algorithm**

Fig. 4 illustrates the message paths when TMTM algorithm is used on the torus network, $T_{\text{torus}}$ of Fig. 1. TMTM algorithm is multicastr the message from the source node $(4, 1)$ to all destination nodes. The source node generates two copies of the message one of them is sent to the destination set, $D_1$ in upper subnetwork such that $D_1 = \{ (3, 1), (1, 2), (7, 2), (0, 3), (4, 3), (1, 4), (2, 4), (7, 4), (0, 5), (5, 5) \}$ and the other message is sent to the destination set, $D_2$ in the lower subnetwork such that $D_2 = \{ (2, 0), (7, 0), (1, 7), (5, 7), (3, 6), (6, 6) \}$. The solid lines represent the main message paths in the upper and lower subnetworks. The dotted lines represent the horizontal message paths that branch from the main paths.

The longest path of channels used to deliver the message to the destination nodes in $D_1$ is 8 and to the destination nodes in $D_2$ is 5. The total number of channels used to deliver the message to the destination nodes in $D_1$ is 25 and to the destination nodes in $D_2$ is 15. Hence, the network traffic and latency of TMTM algorithm are 39 and 8, respectively. From Fig. 2, the network traffic and latency of TASNET algorithm are 45 and 20, respectively.

By comparing TMTM algorithm with the previous algorithm, TASNET [23], it is clearly that the network traffic and latency obtained by TMTM algorithm are the lowest.

Lemma 1: TMTM algorithm is deadlock-free

Proof: At the source node position, the technique of TMTM algorithm splits the torus network into two disassemble subnetworks, \( N_1 \) and \( N_2 \). Thus, \( N_1 \cap N_2 = \emptyset \). Then TMTM algorithm is deadlock-free at the two subnetworks. Now, no dependencies within every subnetwork will be proved. At an intermediate node on a main path, say \( g_k = (x_k, y_k) \), in \( N_1 \), TMTM algorithm splits the destination set \( D_1 \) into two disassemble sets:

\[
D_{\text{side}} = \{ g_i \in D_1 \wedge L(x_k, y_k) < L(g_i) < L(x_k, y_k + 1) \}, \quad D'_{\text{side}} \bigcap \{ D_{\text{side}} \} = \emptyset.
\]

It is obvious that \( D_{\text{side}}' \cap \{ D_{\text{side}} \} = \emptyset \), then, no cyclic dependency can be generated among the paths in \( N_1 \). Hence, TMTM algorithm is deadlock-free.

IV. PERFORMANCE EVALUATION AND RESULT DISCUSSIONS

In this section, the performance of the proposed algorithm is evaluated by comparing with the previous multicast tree-based TASNEI algorithm [23]. A simulation in VC++ language was designed and implemented for this study. The system model for the simulation is 2D T40x40 network with various multicast sizes. The multicast size is the number of destination nodes that uniformly distributed in the network. Also, many different types of 2D torus network sizes, TN are generated and used. The parameter TN refers to the torus network size where \( N \) spreads between 5 and 40, for example T10 means that the torus network with 10 columns and 10 rows. All used 2D torus networks contain two virtual channels per each physical channels.

As a common measures for evaluate the performance, the network latency and the network traffic are used. It is assumed that the network latency time between any two adjacent nodes has been set 25ns [23, 28]. In the following two subsections, the network latency and traffic obtained by TMTM and TASNEI algorithms are plotted.

A. Effect of the Multicast Size

The Figs. 5 and 6 show the network latency and the network traffic obtained by the two algorithms versus various multicast sizes, which is ranging between 100 and 1600 destination nodes. In fig. 5, the network latency obtained by TASNEI algorithm decreases when the multicast size increases. Because of TASNEI algorithm is a tree-based, its paths that starts from the source node trying to move for the following row and create a sub-tree by searching for the nearest destination node. Therefore, as the multicast size increase, the destination nodes may become closer to each current node. The network latency obtained by TMTM algorithm is nearly constant when the multicast size expands. This is due to the fact that TMTM algorithm is a tree-based and its main paths that starts from the source node do not search for the nearest destination node but they directly moving to the following row by sending the message to the above(or below) node that is a destination node or not. This makes the lengths of the main paths are shorter. Many sub-trees are created at each intermediate node of the main paths. In fig. 6, as the multicast size increases, the network traffic obtained by the two algorithms increases.

Figure 4. Message multicasting by using TMTM algorithm

Figure 5. Network latency of TMTM and TASNEI vs. multicast size

Figure 6. Network traffic of TMTM and TASNEI vs. multicast size
subnetworks. To decrease the tree paths that branch from the main paths, TMTM algorithm allows some intermediate nodes that are not destinations to perform multicast operations. The performance of TMTM algorithm was evaluated through comparing it with TASNEM algorithm [23]. The results expound that the best performance is gained by TMTM algorithm over different traffic loads and multicast sizes. Our future works will focus on extending the proposed TMTM algorithm to higher dimensional torus networks. Also, another multicast tree-based that using horizontal wraparound channels of the torus network will be studied.

REFERENCES


Figure 7. Network latency of TMTM and TASNEM vs. torus size

Figure 8. Network Traffic of TMTM and TASNEM vs. torus size

B. Effect of the Network Size
Figs. 7 and 8 show the network latency and traffic gained by the two algorithms versus the various sizes of 2D torus network which is ranging between T5 and T40, Pdest =20%, where Pdest is a parameter that refers to the percentage of destination nodes to the total number of nodes in a torus network. Fig. 7 indicates that as the torus size rises, the network latency gained by the two algorithms rises. This is because as the network size rises, the Pdest value rises and hence the latency values rises. It is clearly that the latency value gained by the proposed algorithm rises slowly while the latency value gained by TASNEM algorithm rises rapidly. In fig. 8, as the torus size rises, the network traffic gained by the two algorithms rises.

From the previous figures (5-8), it is clearly that the network latency obtained by TMTM algorithm is lower than that of TASNEM algorithm. Also, as the number of destination nodes rises, the network traffic obtained by the two algorithms rises but TMTM algorithm has the lowest.

V. CONCLUSION AND FUTURE WORK
In this research, a deadlock-free wormhole multicast tree-based TMTM algorithm is proposed for 2D torus network. TMTM algorithm entail at most two communication start-up steps to multicast to any number of destination nodes. It uses the vertical wraparound channels to divide the torus network into nearly two equally size 2D mesh networks, “J Parall Distrib Comput, 45 (2) (1997), pp. 104–121 http://dx.doi.org/10.1006/jpdc.1997.1372


http://dx.doi.org/10.1006/jpdc.1998.1473


http://www.cray.com/sites/default/files/resources/CrayXE6mBrochure.pdf


[31] MG. Darwish, AA. Radwan, MA Abd El-Baky, Kadry Hamed “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/Sessions_3D.htm

