BTL - An Efficient Deadlock-Free Multicast Wormhole Algorithm to Optimize Traffic in 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.

Mohamed A. El-Sayed
Dept. of Math, Faculty of Science, Fayoum University, Egypt.
Dept. of CS, College of Computer and Information Technology, Taif University, KSA.

ABSTRACT
Multicast communication, in which the same message is sending from a source node to a set of destination nodes, is being increasingly demanded in multicomputer systems. It can be used to support several other collective communication operations. 2D torus network has many features. So, it has become increasingly important to multicomputer design. This paper presents an efficient multicast wormhole deadlock-free algorithm that Balance Traffic Load on 2D torus network; hence the name BTL algorithm. BTL algorithm handles multicast operation with a fixed number of message-passing steps irrespective of the network size. Also, it is designed such that can send messages to any number of destinations within two communication phases. Results from extensive comparative analysis show that BTL algorithm exhibit superior performance advantages over a well-known algorithm.

General Terms
Network, Multicomputer, and Algorithms.

Keywords
Deadlock-Free, Multicast communication, 2D torus topology, Multicomputer.

1. INTRODUCTION
For Multicomputers, Optimizing the performance of the interprocessor communication depends on many factors as selection of the interconnection network, switching technique and routing algorithm. Two dimensional (2D) torus network is frequently utilized on top-performing multicomputers. It has been one of the most important communication networks due to its desirable properties, such as scalability, recursive structure, ease of implementation, constant node degree, constant length channel wires, higher channel bandwidth, low contention latency, and more [1, 2, 3]. Also, it offers edge connectivity and can be partitioned into meshes. Much recent interest in multicomputer systems is therefore concentrated on 2D torus networks. Such technology has been adopted in [4, 5, 6, 7]. Wormhole switching technique is widely used in interconnection networks due firstly to its low buffering requirements, allowing for efficient router implementation. Secondly, and more importantly, it makes latency almost independent of the message distance in the absence of blocking [1, 8, 9].

In this paper, 2D torus networks with bidirectional channels are used. For simplicity, the torus network will draw without channels. Multicasting is an important primitive among collective communication operations. Multicasting allows a source node to send the same message to a group of destination nodes. If the set of destination nodes contains only one node, the multicast called unicast. If the set of destination nodes contains all of the computational nodes in the system, the multicast called broadcast [10]. Many multicast algorithms have been proposed in the literature [2, 11, 12]. The performance of multicast communication is measured in terms of its latency in delivering a message to all destinations. In wormhole-routed networks, the communication latency consists of three parts, start-up latency, network latency and blocking latency [2]. The start-up latency is the time incurred by the operating system when preparing a message for injection into the network. The network latency is a combination of propagation delay, router delay, and contention delay. The blocking latency accounts for all delays associated with contention for routing resources among the various worms in the network.

In this study, an efficient deadlock-free wormhole multicast routing algorithm for 2D torus multicomputer is proposed. This algorithm is multicasting a message to all destination nodes through two phases at most. In this scheme, the objective is to utilize the channels uniformly and reduce the path length of message worms, making multicasting more efficient in 2D torus networks.

The remainder of this paper is organized as follows. Section 2 summaries the related works. The proposed multicast algorithm and a new routing function are presented in section 3. Section 4 evaluates the performance of the proposed algorithm to an existing well known, T2W multicast algorithm [9]. Finally, section 5 concludes this study.

2. RELATED WORKS
In multicast routing, the path selection procedure is very important to maximize the efficiency of the message delivery process. Various path selection techniques have been proposed in the literature. In general, the algorithms for multicast routing can be classified into three types, unicast-based, path-based, and tree-based. In unicast-based routing, identical messages are sent to the destination nodes recursively [4, 13]. This technique suffers from performance inefficiently as well as excessive power consumption [12].

The tree-based routing [14, 15, 16, 17] tries to construct a tree rooted at the source node in order to deliver a multicast message to destination nodes along the paths on the formed tree.

In path-based routing [2, 9, 12], an ordered sequence of destination addresses that must be delivered in the specific order is stored in the header of a message. When a multicast message reach an intermediate destination, the top address is removed from the header and it can be copied to the local memory. The message is routed to the next destination specified in the sorted list. The last destination node removes the message from the network.

There are two types of multicast operation, single-phase (one startup) [18] and multi-phase (there is more than one startup) [9, 19]. In multi-phase multicasting, a message is required more than one step to reach all destination nodes. In [9], Darwish et al. proposed a path-based wormhole multicast algorithm in 2D torus network terms as T2W. In this algorithm, some intermediate nodes that are non-destination nodes are allowed to perform multicast operations. This feature increases flexibility in distributing messages to the destination nodes thereby improving performance that is evaluated through simulations. T2W algorithm can send a message to a set of destinations within two startup communication phases. It can use both horizontal and vertical wraparound channels of a torus network. In the first phase, T2W algorithm defines a horizontal main path starts from the source node and directed to a special node called end node.

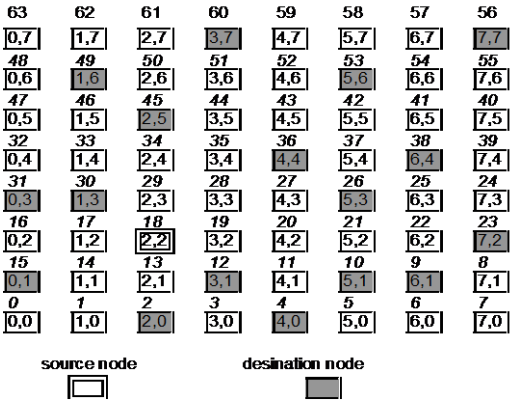

Fig 1: 2D $T_{8\times8}$ torus network

The horizontal main path is selected such that may use the horizontal wraparound channels to cover as many destinations as possible on its path. Also, the nodes on it can cover all remaining destinations that exist on columns of the torus network. In the second phase, some intermediate nodes along horizontal main path retransmit the message to the remaining destination nodes through vertical paths that branch from one side of the main path. In this technique, the multicast routing is divided into sub-multicasts that can be carried out in parallel by many independent paths that branch from the horizontal main path. Fig 1 shows a 2D torus network, $T_{8\times8}$, with a source node, s= (2, 2) and a set of random distributed destination nodes in gray color.

A torus network and message paths become as in fig 2 when T2W algorithm is used to send a message from the source node to all destination nodes. A solid and dotted lines represent communication paths in the first and second phases, respectively.

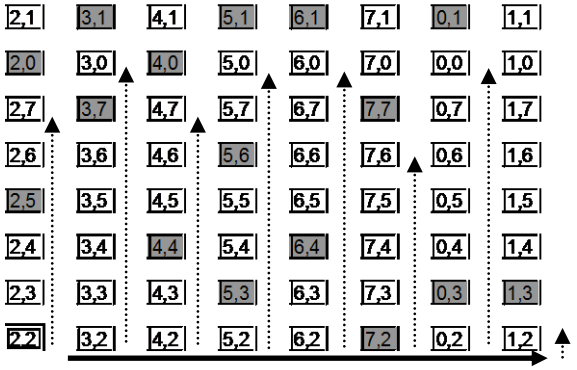

Fig 2: Message multicasting by using T2W algorithm

Reducing the latency and traffic of multicasting message are main important aims of this paper. So, an efficient multicast wormhole routing algorithm with special routing function is introduced (BTL). The proposed algorithm will be compared with T2W algorithm [9]. The simulation results show that the proposed algorithm performs better than T2W algorithm.

3. THE PROPOSED MULTICAST ALGORITHM

This section introduces an efficient multicast wormhole deadlock-free algorithm, BTL for 2D torus network. BTL algorithm uses the concept of virtual partitioning of the torus network into meshes and divides the torus into nearly equally two size horizontal partitions. Each partition represents a 2D mesh. The basic idea behind the introduced algorithm is that, during the first phase, the message is sent to a set of nodes such that all the destinations can be reached in the first or the second phase of multicast communication. In the first phase, BTL algorithm like T2W algorithm where it defines a horizontal path named Horizontal Main Path (HMP) which begins from the source node and may use the horizontal wraparound channels to cover as many destinations as possible. The message is sent to the end node of HMP according to a deterministic routing function which supplies a unique minimal path.

In the second phase, the technique of BTL algorithm is differing from its T2W algorithm. BTL algorithm may use the vertical wraparound channels and divides the torus, $T_{n\times m}$, into nearly equally two virtual meshes. Some intermediate nodes along HMP retransmit the message to the remaining destinations through nearly equally vertical paths. The technique of BTL algorithm shows that the multicast is divided into sub-multicasts that branch from the two sides of HMP. So, the long of paths in BTL algorithm are shorter than of T2W algorithm (nearly half). Also, the multicast is divided into sub-multicasts carried out in parallel fashion by many independent paths. Fig 3 illustrates the message paths when BTL algorithm is used on torus, $T_{8\times8}$ of fig 1. The solid and dotted lines represent communication paths in the first and second phases, respectively.

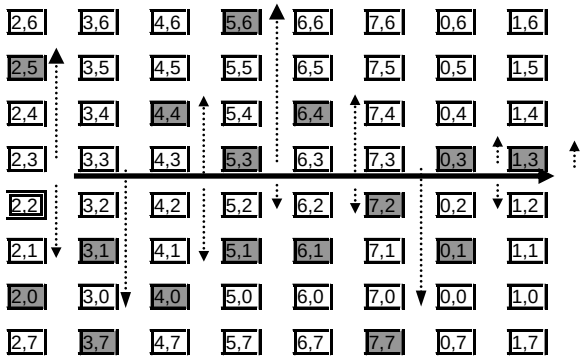

Fig 3: Message multicasting by using BTL algorithm

3.1 Partitioning Algorithm

BTL algorithm begins by determination the best move direction and the optimum last node, $e = (x_e, y_s)$ of *HMP*. This is performing by using algorithm1: **BTL_Balancing_Partitions**. According to the source node position, the technique of algorithm1 uses both of horizontal and vertical wraparound channels and rotates the columns and the rows respectively of a given torus network. So, there are two phases as the following:

Phase 1: algorithm1 may use the horizontal wraparound channels since it selects two destination nodes such that one of them has the farthest *x-coordinate*, x_R, to x_s from the right direction (the right direction starts from the source node column and directed right to reach at the last column that previous the source node column). The other destination node has the farthest *x-coordinate*, x_L, to x_s from the left direction (the left direction is similar to the right direction but directed to left side). So, a variable called *Move_HMP* is used to refer the horizontal moving direction of *HMP* where *Move_HMP* =1 refers to the right direction and *Move_HMP* =0 refers to the left direction. According to *Move_HMP*, the *x-coordinate* of end node, x_e is determined which equals to x_R or x_L.

Phase 2: according to the source node position, algorithm1 may use the vertical wraparound channels to divide the torus, $T_{n\times m}$, into two virtual meshes, M_1 and M_2. So, there are two cases:

Case 1: If the *y-coordinate* of the source node is less than $\lceil m/2 \rceil$, then M_1 contains the nodes whose *y-coordinates* are between *y-coordinate* of the source node and *y-coordinate* of the source node plus $\lfloor m/2 \rfloor$, M_2 contains the remaining nodes.

Case 2: If the *y-coordinate* of the source node is greater than or equal to $\lceil m/2 \rceil$, then M_1 contains the nodes whose *y-coordinates* are between *y-coordinate* of the source node and *y-coordinate* of the source node minus $\lfloor m/2 \rfloor$, M_2 contains the remaining nodes.

For any $T_{n\times m}$ with a destination set $D = \{(x_0,y_0), (x_1,y_1),\dots, (x_k,y_k)\}$. Let $D_x = \{x: (x, y) \in D\}$, $D_y = \{y: (x, y) \in D\}$ are the two sets of *x-coordinates* and *y-coordinates* respectively of *D*. Let $L_x=\{x_i: 0 \le x_i < x_s, x_i \in D_x\}$, and $R_x=D_x-L_x$ are the two sets of *x-coordinates* of *D* which exist on the left columns and the right columns respectively of the column that contains the source node.

Fig 4 illustrates algorithm1 which produces x_e , Move_HMP, and N_HMP that is the number of nodes on HMP.

Algorithm1: BTL_Balancing_Partitions

Input: source node $s= (x_s, y_s)$, the sets *Lx, Rx*

Output: x_e , *Move_ HMP* , *N_HMP*

Begin:

1- *IF* $(Lx \neq \Phi)$ *THEN*

$x_R = Max\{Lx\}$, $R_{dist}=n-|x_s-x_R|$

ELSE $x_R = Max\{Rx\}$, $R_{dist}=|x_s-x_R|$.

2- $R_{X1}=Rx-\{x_s\}$, $L_{X1}=Lx \cup \{x_s\}$

3- *IF* $(R_{X1} \neq \Phi)$ *THEN*

$x_L = Min\{R_{X1}\}$, $L_{dist}=n-|x_s-x_L|$

ELSE $x_L = Min\{L_{X1}\}$, $L_{dist}=|x_s-x_L|$.

4- *IF*$(R_{dist} \geq L_{dist})$ *THEN*

$x_e = x_R$, *Move_ HMP =1* , $N\_HMP = R_{dist}+1$

ELSE

$x_e = X_L$, *Move_ HMP =0* , $N\_HMP = L_{dist}+1$

5- Return (x_e , *Move_ HMP* and *N_HMP*)

End BTL_Balancing Partitions algorithm

Fig 4: Determination HMP

3.2 A Formal Routing Function

A special path routing function, R_{BTL}, is needed to determine the next node for which the path of BTL algorithm will visit in a 2D torus network, $T_{n\times m}$. R_{BTL} is the same as *XY-routing* [**20**] with some conditions which deal with the horizontal and vertical wraparound channels and prevent deadlock to occur. So, a routing path by R_{BTL} is decided first along the *X-dimension* before choosing a path along the *Y-dimension*. Clearly, R_{BTL} routing is deadlock-free for one-to-one communication because it is impossible for a cyclic channel dependency to arise if channels are acquired in *XY* order. The direction variable, *Move_HMP* which is decided in previous subsection will be considered. To send a message from a current node $u= (x_u, y_u)$ to a destination node $v= (x_v, y_v)$, the horizontal neighbor node of node *u* denoted *h*_node(*u*) is determined as follows:

$$h\_node(u) = \begin{cases} (0, yu) & if \quad xu = n-1 \ \wedge \ Move\_hor = 1 \\ (n-1, yu) & if \quad xu = 0 \ \wedge \ Move\_hor = 0 \\ (xu+1, yu) & if \quad Move\_hor = 1 \\ (xu-1, yu) & if \quad Move\_hor = 0 \end{cases} \quad (1)$$

The path routing function of BTL algorithm is a function R_{BTL}: (P × P ➔ P) that maps a (current node, destination node) pair to a neighbor node of the current node. It is defined as follows: $R_{BTL}(u, v) = hv\_node$, where

$$hv\_node = \begin{cases} h\_node(u) & if \quad xu \neq xv \\ (xu, n-1) & if \quad yu = 0 \wedge yv > ys + \lfloor m/2 \rfloor \\ (xu, 0) & if \quad yu = n-1 \wedge yv < ys - \lfloor m/2 \rfloor \\ (xu, yu-1) & if \quad yu > yv \\ (xu, yu+1) & if \quad yu < yv \end{cases} \quad (2)$$

3.3 Message Processing of BTL Algorithm

In this subsection, two ordered subsets (*Olist1* and *Olist2*) of the destination nodes are constructed. One of them is constructed and sorted from the destinations in M_1 and the other is constructed and sorted from the destinations in M_2.

Fig 5 describes the components of algorithm2: BTL_Message_Processing, which constructs *Olist1* and *Olist2* according to the position of the source node where there are four cases. So, there are many *FOR* loops that organize the building of *Olist1* and *Olist2* such that all destinations (are not on *HMP*) receive a message through vertical paths branch from *HMP*. The function *FILL_LIST()* is used to fill *Olist1* and *Olist2* with destination nodes.

Algorithm2: BTL_Message_Processing

Inputs*: s, D, e, N_HMP*

Output: *Olist1 and Olist2*

Begin: let $D=D \cup \{s\}$, $Olist1=\Phi$, $Olist2=\Phi$, $b=s$

FOR k=1 TO N_HMP

{*1. u=b, OL1=*Φ*, OL2=*Φ

2. IF $(y_s < \lceil m/2 \rceil)$ *THEN*

FOR $i=y_u+1$ *TO* $y_u+\lfloor m/2 \rfloor$

IF $(x_u, i) \in D$ *THEN FILL_LIST* $((x_u,i), OL1)$

FOR $i= y_u - 1$ *TO 0 step* –1

IF $(x_u, i) \in D$ *THEN FILL_LIST* $((x_u,i), OL2)$

FOR i=n–1 TO y_u+⌊m/2⌋+1 step -1

IF (x_u, i) ∈D THEN FILL_LIST ((x_u,i), OL2)

3. ELSE

FOR i= y_u – 1 TO y_u –⌊m/2⌋ step -1

IF (x_u, i) ∈D THEN FILL_LIST ((x_u,i), OL1)

FOR i= y_u + 1 TO n–1 step +1

IF (x_u, i) ∈D THEN FILL_LIST ((x_u,i), OL2)

FOR i=0 TO y_u-⌊m/2⌋-1 step -1

IF (x_u, i) ∈D THEN FILL_LIST ((x_u,i), OL2)

4. IF (OL1 ≠ Φ ∨ u∈D) THEN

Olist1= Olist1 ||{u}|| OL1

5. IF (OL2 ≠ Φ ∨ u∈D) THEN

Olist2= Olist2 ||{u}|| OL2

6. b= R_{BTL} (u,e)

}

RETURN (Olist1, Olist2)

END BTL_Message_Processing algorithm

Fig 5: Message processing algorithm

So, the source node constructs two messages, one containing *Olist1* as part of the header and the other contain *Olist2* as part of the header. The source node sends two messages into two disjoint subnetworks M_1 and M_2.

Next, the BTL algorithm uses a distributed routing method in which the routing decision is made at each intermediate node. Upon receiving the message, each intermediate node determines whether its address matches that of the first destination node in the message header. If so the address is removed from the message header, the message is copied and sent together with its header to the neighboring node using the routing function R_{BTL}. In case where the intermediate node is not a destination, it sends the message together with its header to the neighboring node using the routing function R_{BTL}. If the sets of the destination nodes are not empty, the algorithm continues according to the previous method.

Theorem 1: BTL Algorithm is Deadlock-Free.

Proof: as explained in previous subsection 3.1 and according to the position of the source node, BTL algorithm divides the 2D torus network into two disjoint sub-networks, M_1 and M_2. This is obvious since $M_1 \cap M_2 = \Phi$. Then BTL algorithm is deadlock-free at M_1 and M_2. The technique of BTL algorithm uses both of the horizontal and vertical wraparound channels and rearranges columns and rows to deal with a torus as a virtual mesh network. It makes the source node nearly in the middle of the first or last column. In the first phase, there is a main horizontal path (HMP) starts from the source node and extend to last column contains destination nodes. HMP sends the message to the destination nodes that pass on. All other destinations receive the message through vertical paths branch from HMP (up or down in M_1 or M_2). So, all paths are parallel and as follows, there is no intersection between any paths. Then no cyclic dependency can be created among the channels. So, BTL algorithm is deadlock-free.

4. PERFORMANCE EVALUATION

In this section, the performance of the proposed algorithm, BTL is compared with well-known multicast T2W algorithm [9]. A common metric used to evaluate the performance of an interconnection network system is the communication latency, which is approximated by T_S+T_n [21]. T_S is the startup time and consists of two parts T_{S1} and T_{S2}. T_{S1} is made at the source node in the first phase. T_{S2} is made at each intermediate node along HMP in the second phase to retransmit the message to rest of the multicast destinations. It is clear that $T_{S2} < T_{S1}$ because the generation of succeeding messages should take less time than the generation of the first message. The network time, T_n, is the total time spent between the injection of the message into the network until the message is drained away. T_n is different for the two algorithms, T2W and BTL. So, it is used to compare them. Each multicast message can be expressed as sequences of serially forwarded unicast messages from root to destinations [22]. So, the time of multicast message is expressed as follow:

$$T_{multicast} = Max\,[T_s + T_n] \quad \text{over} \quad \forall_{paths} \qquad (3)$$

Where *Max* [ ] operation yields the total multicast latency for deepest path over all paths, and $T_{multicast}$ is the time interval between the initiation of the multicast and the last destination's reception of the message. The network traffic is another parameter and is defined as the number of channels used to deliver all messages involved.

The two metrics, the network latency and the network traffic are calculated for two algorithms to compare the performance. A routing model for each algorithm is used as path processes to determine the channels on which each message should be transmitted. A simulation in VC++ language was designed and implemented for performance evaluation. Many random 2D torus networks that contain two virtual channels per physical channel were used. Each network contains a source node (x_s, y_s) and a set of destination nodes that are uniformly distributed through each network. The networks were generated with different numbers of processors ranging from 25 to 3200. It is assumed that the network latency time between any two nodes is 30 ns, T_{S1} is set to 1μs, and T_{S2} is set to 240 ns. Figs. 6-8 show the results of these algorithms.

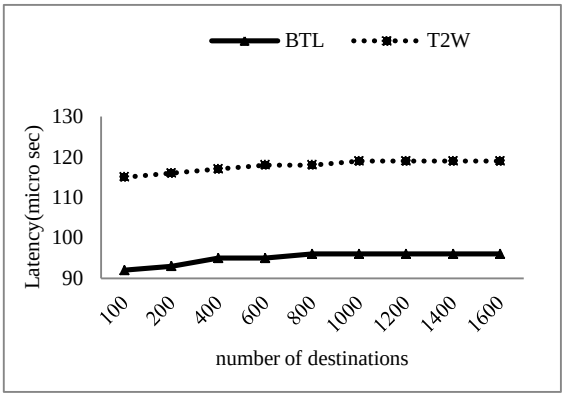

Fig 6: network latency of BTL and T2W vs. no. of destinations

Fig 6 plots the multicast latency obtained by the two algorithms on $T_{40\times40}$ versus various values of the number of destination nodes, ranging from 100 to 1600. The source node

is nearly in the center of the network. It is clear that, as the number of destination increases, the latency values obtained by all algorithms increase. BTL algorithm however, is less sensitive to the increased load than T2W algorithm. This is due to the fact that BTL routing uses shorter paths; so resources are held for shorter time periods, leading to higher throughput.

Fig 7 plots the multicast latency obtained by the two comparison algorithms versus the different sizes of torus networks within range from 25 to 1600 nodes, pd=20% where pd is the percentage of destination nodes out of the total number of nodes in the network, and the source node is nearly in the center of the network. It is clear that, as the torus size increases, as the latency values increase. However, the latency obtained by BTL algorithm increases slowly. It is obvious that, BTL algorithm outperforms T2W algorithm.

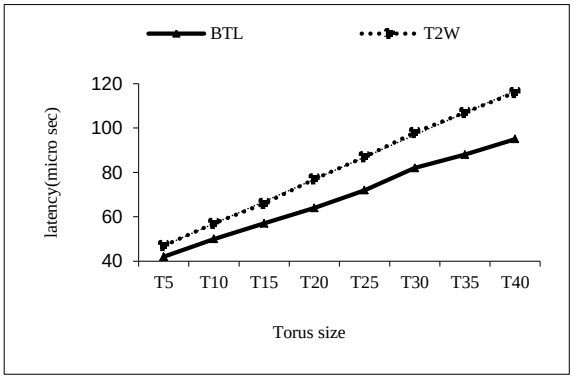

Fig 7: Network latency of T2W and BTL vs. torus size

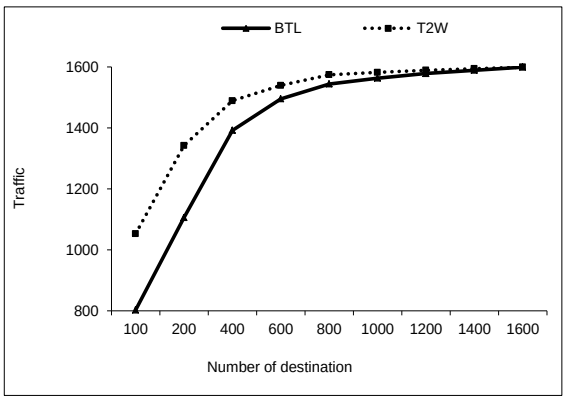

Fig 8: Network traffic of T2W and BTL vs. no. of destinations

Fig 8 plots the network traffic obtained by the two algorithms on $T_{40\times40}$ versus various values of the number of the destinations ranging from 100 to 1600. For the two algorithms, as the number of destinations increases, the traffic curves increase until they meet at a certain point. This happens when Pd =100%, i.e., at the broadcast pattern. But the increasing rate of traffic curve of BTL algorithm is lowest. Also, at small number of destinations, the increasing rate of traffic curves is large, nearly 25%, after this ratio, the Increasing rate is small. Generally, from the previous figures, the following notes can be observed:

- BTL algorithm performs better than T2W algorithm.
- As the number of destinations increases, latency values obtained by the two algorithms increases but BTL routing is less sensitive to the increased load than T2W algorithm. This is due to the fact that it's BTL routing uses shorter paths, figs 6, 7.
- Also, the number of destinations increases, the traffic of the two algorithms increase but BTL algorithm has the lowest, fig 8.

Finally, BTL algorithm is efficiently used in 2D torus multicomputer.

5. CONCLUSION AND FUTURE WORK

In this study, a deadlock-free wormhole multicast algorithm in 2D torus multicomputer, BTL, was presented. This algorithm used a path-based facility and is shown to be deadlock-free. BTL routing uses both horizontal and vertical wraparound channels to send a message to a set of destinations within two phases at most. Also, a routing function, R_{BTL} is designed and is used as a base for the proposed algorithm. The performance of BTL algorithm was evaluated through comparing it with T2W algorithm [9]. The results show that the best performance is obtained by BTL algorithm over different traffic loads and destination set sizes. Our future works will focus on extending the proposed BTL algorithm to higher dimensional torus networks. Also, another multicast partitioning and routing strategy will be studied to enhance the overall system performance.

6. ACKNOWLEDGMENTS

We are using this opportunity to express our gratitude to everyone who supported us throughout completing this research. We also thank all members of Shaqra University and our rector for continues support.

7. REFERENCES

[1] Dally, W.J., and Seitz, C.L. 1986. The torus routing chip. Journal of Distributed Computing 1 (3) 187-196.

[2] Darwish, M. G., Radwan, A. A. A., Abd El-Baky, M. A., and Hamed, Kadry. 2008. TTPM-An Efficient Deadlock-Free Algorithm for Multicast communication in 2D Torus Networks. Journal of Systems Architecture, (October 2008) Volume 54, Issue 10, 919-928.

[3] Wang, Neng-Chung and Hung, Yi-Ping. 2009. Multicast communication in wormhole-routed 2D torus networks with hamiltonian cycle model. Journal of System Architecture 55, 70-78.

[4] Robinson, D.F., McKinley, P.K., and Cheng, B.H.C. 1995. Optimal Multicast Communication in Wormhole-Routed Torus Networks. IEEE Trans. Parallel and Distributed Systems, (Oct. 1995), vol. 6, no. 10, 1029-1042.

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

[6] Fowler, A., Mariantoni, M., Martinis, J., and Cleland, A. 2012. A primer on surface codes: Developing a machine language for a quantum computer. arXiv:1208.0928.

[7] Devitt, S.J., Nemoto, K., and Munro, W.J. 2013. Quantum error correction for beginners. Reports on Progress in Physics. (Aug. 2013), 76, 8.

[8] Ni, L. M., and McKinley, P. K. 1993. A survey of routing techniques in wormhole Networks. IEEE Computer, (Feb. 1993), vol. 26, no. 2, 62-76.

[9] Darwish, M. G., Radwan, A. A. A., Abd El-Baky, M. A., and Hamed, Kadry. 2010. T2W:A Multicast Routing Algorithm For 2D Torus Networks with Horizontal Main Path Model. International Journal of Intelligent Computing and Information Science, (July 2010), v. 10, no. 2.

[10] Moharam, H., Abd El-Baky, M. A., and Nassar, S. M. M. 2000. YOMNA-An efficient deadlock-free multicast wormhole algorithm in 2-D mesh multicomputers. Journal of Systems Architecture, (October 2000), Volume 46, Issue 12, 1073-1091.

[11] El-Obaid, Amnah and Li-Zuo, Wan. 2008. An Efficient Path-Based Multicast Algorithm for Minimum Communication Steps. Inform. Technol. J., 7(1): 32-39.

[12] Moosavi, S.R., Rahmani, A.-M., Liljeberg, P., Plosila, J., and Tenhunen, H. 2013. An Efficient Implementation of Hamiltonian Path Based Multicast Routing for 3D Interconnection Networks. 21st Iranian Conference on Electrical Engineering, (May 2013), (ICEE), p(6).

[13] Mckinley, P., Xu, H., Esfahanian, A-H., and Ni, L. 1994. Unicast-Based Multicast Communication in Wormhole-Routed Networks. IEEE Trans. on Parallel and Distributed Systems, (Dec. 1994), vol. 5, no. 12, 1252-1265.

[14] Malumbres, M. P., Duato, J., and Torrellas, J. 1996. An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors. Proc. Of the 8th IEEE Symp. On Parallel and Distributed Processing, (Oct. 1996), 186-189.

[15] Wang, Honge and Blough, Douglas M. 1998. Tree-Based Multicast in Wormhole-Routed Torus Networks. International Conference on Parallel and Distributed Processing Techniques and Applications. 702-709.

[16] Wang, Nen-Chung and Chu, Chih-Ping. 2005. An Efficient Tree-Based Multicasting Algorithm on Wormhole-Routed Star Graph Interconnection Networks Embedded with Hamiltonian Path. The Journal of Supercomputing 34(1), 5-26.

[17] Wang, Nen-Chung., Chen, Young-Long., Chen, Chin-Ling., Chen, Ying-Shiou. 2011. A Dual-Tree-Based On-Demand Multicast Routing Protocol for Mobile Ad Hoc Networks. SNPD, 128-132.

[18] Lin, X., McKinley, P. K., and Ni, L. M. 1994. Deadlock-free multicast wormhole routing in 2D mesh multicomputers. IEEE Transactions on Parallel and Distributed Systems, (August 1994), vol.5, 793-804.

[19] Darwish, M. G., Radwan, A. A. A., Abd El-Baky, M. A., and Hamed, Kadry. 2010. Ready Groups: A Path-Based Multicast Algorithm for 2D Torus Networks. The 7th International Conference on Informatics and Systems (March 2010), (INFOS 2010) - 28-30.

[20] Lin, X., McKinley, P. K., and . Ni, L. M. 1994. Deadlock-free multicast wormhole routing in 2D mesh multicomputers. IEEE Transactions on Parallel and Distributed Systems, (August 1994), vol.5, 793-804.

[21] Darwish, M. G., Radwan, A. A. A., Abd El-Baky, M. A., and Hamed, Kadry. 2005. 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 14-16, 2005.

[22] Oral, S. and George, A. 2003. Multicast Performance Modeling and Evaluation for High-Speed Unidirectional Torus Networks. HCS Research Lab, U. of Florida, submitted (June 2003).