Network and Complex Systems
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol 2, No.4, 2012

A Deadlock – Free Routing Algorithm for Torus Network

Krishnan M.N., Raghunath S., Ajith Pravin Dhas D., and Benny Raj A.M., Prof. M.Pounambal
School of Information Technology and Engineering, VIT University
Vellore-632014, Tamil Nadu, India
kadampuzhas@gmail.com, aphdas@gmail.com, mpounambal@vit.ac.in

Abstract:

TORUS is a n-dimensional network topology. Each dimension will have k nodes. A routing algorithm determines the sequence of channels for a packet to traverse from the source to destination. A new router design that significantly reduces the main drawback of worm hole switching – latency, is presented in this paper. Worm-hole switching is combined with virtual channel to provide better performance. Packet deadlock is avoided by verifying the freeness of the nodes before sending the packets to that node.

The traditional **'wormhole switching'** mechanism for routing in the torus network has the disadvantages such as link contention, message latency, need for large buffer size and finally a massive deadlock may appear. The recently proposed **'clue'** algorithm, has the disadvantages such as difficulty in cut through the link by the packets, says nothing about loss of packets between a hop and storage overhead and complexity in dividing the virtual channels. We proposed an **'Advanced Clue'** algorithm by combining the concepts of clue and flow controlled clue and also overcome the disadvantages of clue. We use two virtual channels and a buffer which gives a combination of clue and flow controlled clue. We also propose conditions that satisfy the reliability of the packet delivery between hops. The packet will be sent to the next hop and buffered in the current hop. The sending hop will set a timer and wait for the acknowledgement. If the acknowledgement is not arrived till the timer expired then, the packet will be resend, and otherwise the packet will be removed from the buffer.

Keywords: Torus, Virtual channels, Cut – through Switching, Wormhole switching.

1. INTRODUCTION

An n-dimensional k-ary torus is a network with k*n nodes. By definition, the n-dimensions of the n-dimensional torus will be referred to as $X_{n-1},\ldots, X_0$. Each node of the torus will be denoted by a tuple $(X_{n-1},\ldots, X_0)$ with ,$0 \leq X_i < k$ for all $0 \leq i < n$, and will be connected to nodes $(X_{n-1}, \ldots, X_{i+1}, (X_{i+1}) \bmod k, X_{i-1}, \ldots, X_0)$, and $(X_{n-1}, \ldots, X_{i+1}, (X_{i-1}) \bmod k, X_{i-1}, \ldots, X_0)$for all $0 \leq i < n$. The link connecting nodes $(X_{n-1}, \ldots, k-1, \ldots, X_0)$ and $(X_{n-1},\ldots, 0, \ldots, X_0)$ along dimension X_i will be called a wrap-around link along dimension X_i [6]. This torus topology is primarily used in Networks-on-Chip technology. NoCs have been emerging technologies for high – performance computer systems and CPUs.

2. RELATED WORKS

Wei Luo and Dong Xiang explained the methods for routing the packets in a torus network using two virtual channels. They also proposed the Virtual cut-through method for forwarding the packets to avoid deadlock [8]. Here they have given two algorithms named Clue and flow-controlled clue and used two virtual channels and the concept

Virtual cut-through switching. Several routing algorithms were proposed for meshes and tori [1], [2], [3]. Based on a scheme called virtual partitioning, in which two virtual networks shares a virtual channel, Xiang proposed a new scheme for 3-dimensional torus networks, called as Channel overlapping [2]. He extended the deadlock-freeness by including an adaptive fault – tolerant routing algorithm. In the dimension order routing for a 2-dimensional mesh network, the packet delivery in Y-dimension will be considered only after all nodes in X – dimension is eliminated [5]. The adaptive bubble router [7] for VCT switched torus is based on Duato's protocol, and implements adaptive routing with minimum path length. It provides default escape channels to ensure that packets will not be blocked at point.

3. WORKING PRINCIPLE

3.1 Advanced Clue Algorithm:

We assume the following conditions in our proposed system:

- The link between nodes is considered as reliable for transferring single bit without loss.
- The minimum number of nodes in each dimension is 3. Because if k=1 it doesn't form a network and
- If k=2 there is no formation of wrap-around link it forms a simple ring topology.
- The number of senders trying to send messages simultaneously must be $\leq 2k$ for $k \geq 4$ and k for k=3..

3.2 Channelization:

Two virtual channels are used in this algorithm, say R1 and R2. Whenever there is a wrap-around link between two nodes then the nodes are added to R2 channel. All other node pairs which do not have a wrap-around link are included in the R1 channel.

3.3 Decision of channel:

For each hop the current node has to decide the channel from which the node for next hop is to be selected. For this the following algorithm is used:

- If |C-D|>k/2 R2 channel is selected
- For 2-dimensional torus network considering the X-dimension if |Cx-Dx|>k/2 then go for horizontal-wrap of node Cx.
- Considering the Y-dimension if |Cy-Dy|>k/2 then go for vertical-wrap of node Cy.
- Otherwise R1 channel is selected
- Considering the X-dimension if Cx<Dx then go for right-hop of node Cx, else go for left-hop of
- node Cx.
- Considering the Y-dimension if Cy<Dy then go for up-hop of node Cy, else go for down-hop of node Cy.

3.4 Checking for free node:

After selecting the channel for the next hop the sender or the current node has to check whether the node selected for the next hop is free. To check this sender has to forward a signal or a single bit called **"Reserve"** to the hop node. When a node receives a **"Reserve"** signal, then it must set its status as busy and sends the *reply* signal. the sender receives the *reply* signal then it knows that the next hop node is free and is reserved for the

3.5 Forwarding or resending packets:

After reserving the path or the node for next hop the sender has to forward the packet to the receiver or the next hop node. Here the condition is that, the sender has to send only a copy of the packet to the next hop. It has to set a timer for acknowledgement and has to wait. On receiving the packet the receiver has to reply to the corresponding sender with an **"Ack"** signal. On receiving the Ack signal the sender has to flush its buffer to receive the next packet. If suppose there is no Ack signal until the timer expires then the sender has to resend the packet. This increases the reliability of the reception of data by the receiver. The nodes have to maintain the reserved state until they receive the last packet. On receiving the last packet the nodes have to change their status to *free* and flush its buffer.

3.6 Routing:

Using the techniques applied in the above module, we can select a channel for the passage of packet to the next hop. Each packet in the message will be forwarded by sending to the selected hop. Forwarding stops when the packet is delivered to its destination. In this way the entire message is delivered from the source hop to the destination hop.

4. RESULTS AND DISCUSSION

The screen shots of the simulation are given below:

1) Fig 1 shows the initial condition of the network. The wrap-around links are denoted by red lines and the neighbouring links by black lines. Here we consider a network of 2-dimension with k = 3.
2) Fig 2 shows checking of freeness of nodes from source (node7) to destination (node0). It chooses the path node7 → node1 → node0. This path is selected on the fact that |Cy – Dy|> k/2 i.e.) |3-0| > (3/2). So vertical wrap-around link to node1 is selected for the next hop. After that node0 is selected by the fact that Cx > Dx. Therefore it traverses to the left node i.e.) node 0.

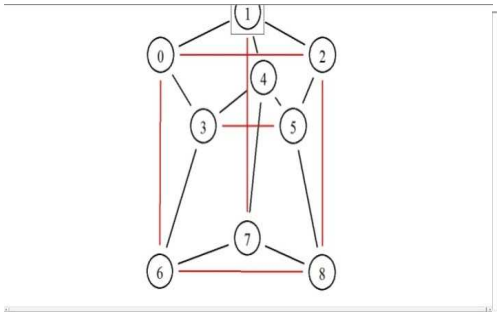

Fig 1

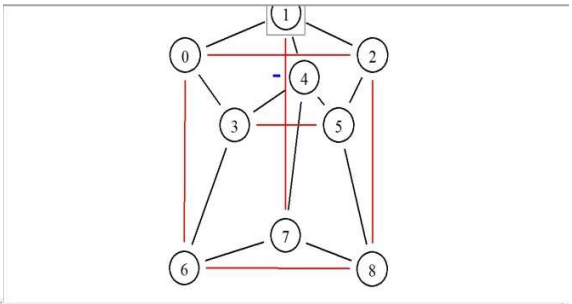

Fig 2

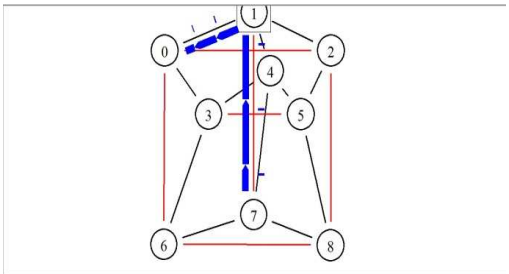

Fig 3

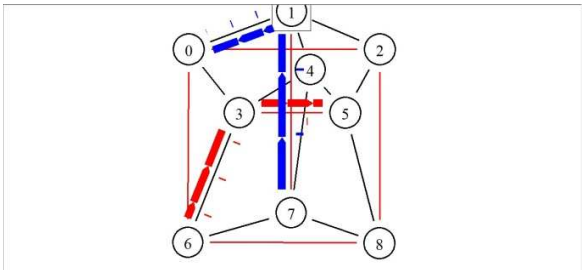

Fig 4

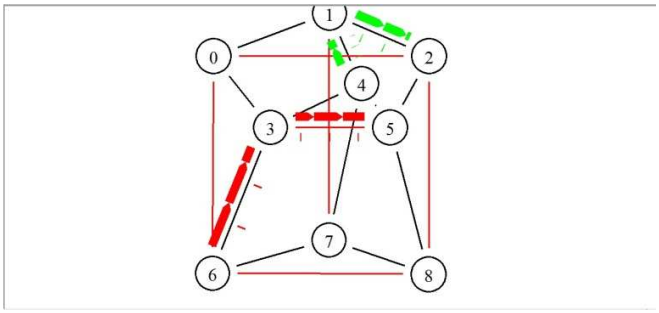

Fig 5

3) Fig 3 shows forwarding of packets from node7 to node0.
4) Fig 4 shows the packet flow from node6 to node5 via the path node6 → node3 → node5 using the algorithm without formation of any deadlocks.
5) Fig 5 shows the packet flow from node4 to node2 via the path node4 → node1 → node2 using the algorithm without the formation of deadlock.

The following snapshots show the algorithm implementation using Java:

1) Fig a shows Initial condition of 2-D 3-ary torus, where the nodes initially all the nodes are free and we assigned a port number for each node.
2) Fig b shows the forwarding path of a message from (0,0) to (2,2) via the path {(0,0), (2,0), (2,2)}. Here the next hop is selected using the algorithm, first the horizontal wrap – around link is selected and then the vertical wrap – around link is selected.
3) Fig c shows that the path (2,0) is made busy or busy with some other packet forwarding and the sender (0,0) chooses an alternative path to reach the destination (2,2) since the path (2,0), is busy via the path {(0,0), (1,0), (1,2), (2,2)}. In this we don't go backwards in the network even there is no node to go forward in the network then it'll wait until the nodes are free.
4) Fig d shows that the sender is checking the node continuously when the destination is busy. It'll reserve the path at once when the route is free.

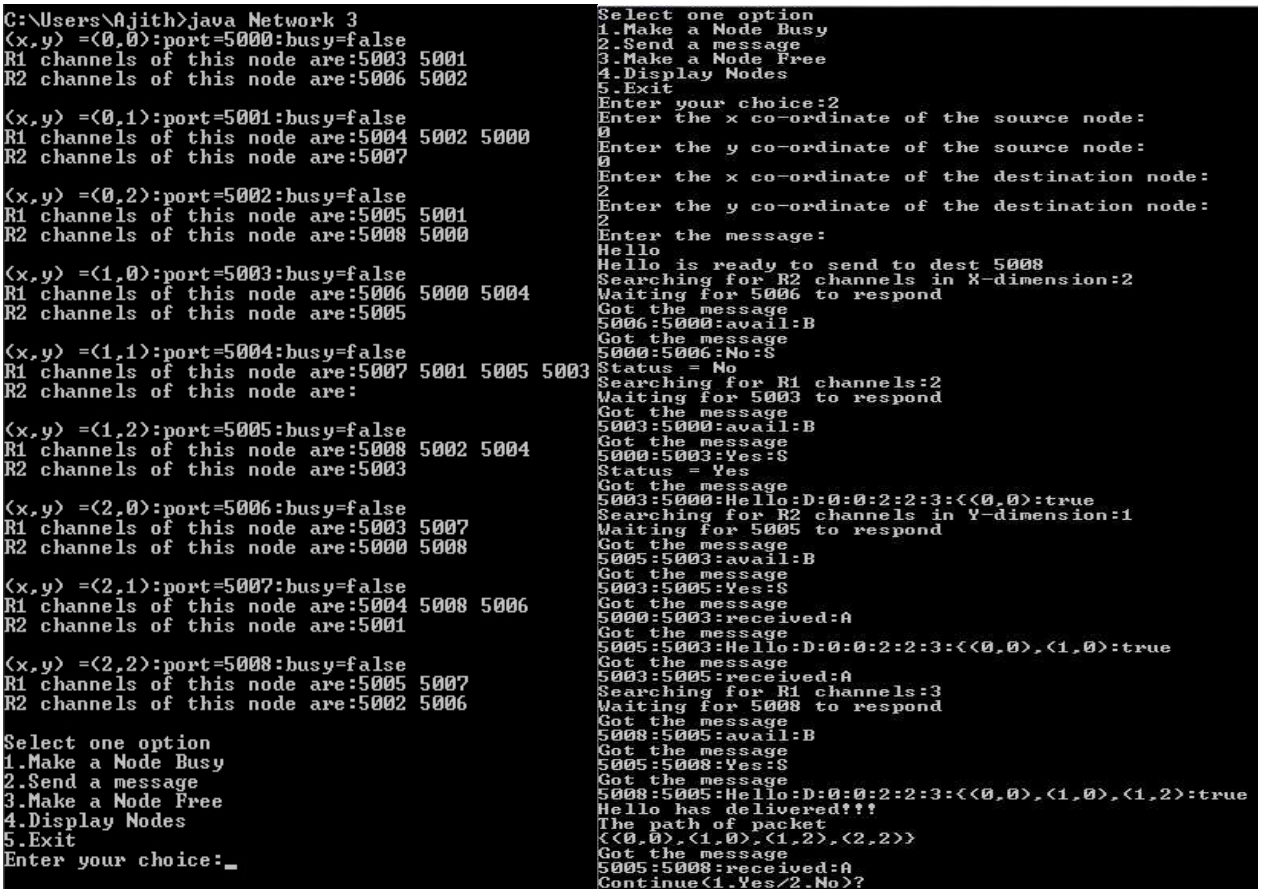
C:\Users\Ajith>java Network 3
(x,y) =(0,0):port=5000:busy=false
R1 channels of this node are:5003 5001
R2 channels of this node are:5006 5002
(x,y) =(0,1):port=5001:busy=false
R1 channels of this node are:5004 5002 5000
R2 channels of this node are:5007
(x,y) =(0,2):port=5002:busy=false
R1 channels of this node are:5005 5001
R2 channels of this node are:5008 5000
(x,y) =(1,0):port=5003:busy=false
R1 channels of this node are:5006 5000 5004
R2 channels of this node are:5005
(x,y) =(1,1):port=5004:busy=false
R1 channels of this node are:5007 5001 5005 5003
R2 channels of this node are:
(x,y) =(1,2):port=5005:busy=false
R1 channels of this node are:5008 5002 5004
R2 channels of this node are:5003
(x,y) =(2,0):port=5006:busy=false
R1 channels of this node are:5003 5007
R2 channels of this node are:5000 5008
(x,y) =(2,1):port=5007:busy=false
R1 channels of this node are:5004 5008 5006
R2 channels of this node are:5001
(x,y) =(2,2):port=5008:busy=false
R1 channels of this node are:5005 5007
R2 channels of this node are:5002 5006
Select one option
1.Make a Node Busy
2.Send a message
3.Make a Node Free
4.Display Nodes
5.Exit
Enter your choice:
Select one option
1.Make a Node Busy
2.Send a message
3.Make a Node Free
4.Display Nodes
5.Exit
Enter your choice:2
Enter the x co-ordinate of the source node:
0
Enter the y co-ordinate of the source node:
0
Enter the x co-ordinate of the destination node:
2
Enter the y co-ordinate of the destination node:
2
Enter the message:
Hello
Hello is ready to send to dest 5008
Searching for R2 channels in X-dimension:2
Waiting for 5006 to respond
Got the message
5006:5000:avail:B
Got the message
5000:5006:No:S
Status = No
Searching for R1 channels:2
Waiting for 5003 to respond
Got the message
5003:5000:avail:B
Got the message
5000:5003:Yes:S
Status = Yes
Got the message
5003:5000:Hello:D:0:0:2:2:3:<(0,0):true
Searching for R2 channels in Y-dimension:1
Waiting for 5005 to respond
Got the message
5005:5003:avail:B
Got the message
5003:5005:Yes:S
Got the message
5000:5003:received:A
Got the message
5005:5003:Hello:D:0:0:2:2:3:<(0,0),(1,0):true
Got the message
5003:5005:received:A
Searching for R1 channels:3
Waiting for 5008 to respond
Got the message
5008:5005:avail:B
Got the message
5005:5008:Yes:S
Got the message
5008:5005:Hello:D:0:0:2:2:3:<(0,0),(1,0),(1,2):true
Hello has delivered!!!
The path of packet
<(0,0),(1,0),(1,2),(2,2)>
Got the message
5005:5008:received:A
Continue(1.Yes/2.No)?

Fig.a **Fig.b**

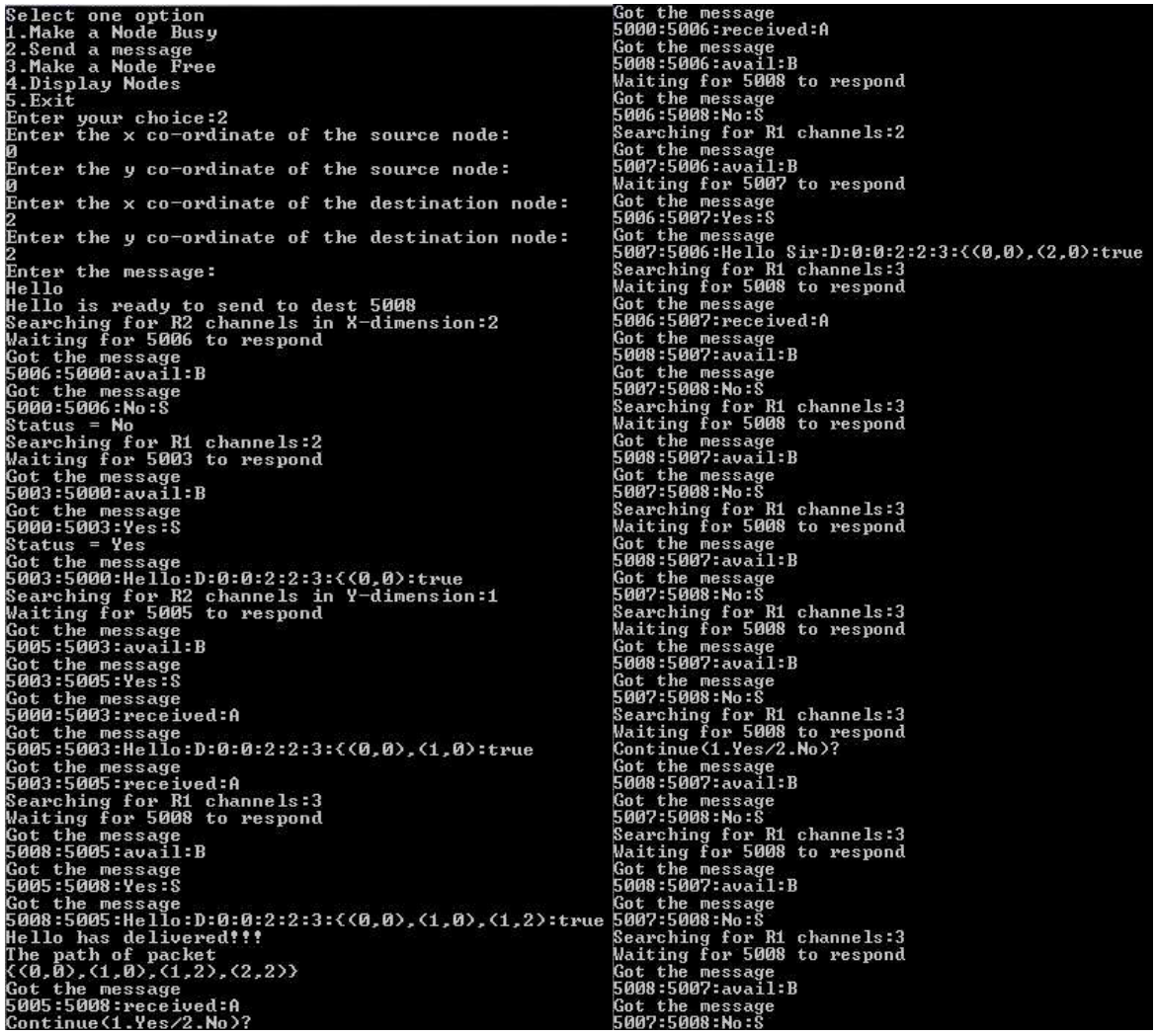

Fig. c **Fig.d**

5. QOS SUPPORT

5.1 Scalability:

The network can be expanded to whatever extend in choice of dimension or in number of nodes in each dimension. This algorithm can be used for routing packets in multi-dimensional torus networks.

5.2 Reliability:

The packet routing is reliable as we are using a method to resend the packets until it reaches the destination. Also we consider that the path is reliable for single bit transaction.

5.3 Maintainability:

As this routing algorithm is modeled for dynamic torus network it can work even there are certain changes in the network structure.

5.4 Deadlock Freeness:

As we are sensing the path for free node, we are sending only when the path is free to send. This will avoid the formation of deadlock. Also we make the nodes holding the packets as busy to avoid congestion. We assumed that the number of senders who can send packets simultaneously is at most $2k$ for $k \geq 4$ and k for $k = 3$. We can also give a round-robin scheduling property to this network to avoid starvation for a link by a node to send packets, which we've not yet implemented.

6. CONCLUSION

Thus we give a solution to avoid deadlock while routing packets in a 2-dimensional torus network. We also take care of reliablity of the packet transfer into condsideration and proposed the method of retransmission in case of packet loss. We also provide a dynamic capablity to our algorithm to make it suitable for working with any number of nodes in each dimension.

7. REFERENCES

[1] Dong Xiang, Ysi Pan, Qi Wang and Zhen Chan, "Deadlock-Free Fully Adaptive Routing in 2-Dimensional Tori Based on New Virtual Network Partitioning Scheme," *28th International Conference on Distributed Computing Systems* 17-20th Jun 2008.

[2] Dong Xiang, "Deadlock – Free Adaptive Routing in Meshes with Fault Tolerance Ability Based on Channel Overlapping" *IEEE Trans. on Dependable and Secure Computing,* vol.8, no.1,pg. 74-88, Jan. 2011.

[3] Ed Anderson, Jeff Brooks, Charles Grassl and Steve Scott, "Performance of the CAY T3E Multiprocessor," *ACM/IEEE SC97 Conference (SC'97).*

[4] Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi and Davide Patti, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip," *IEEE Trans. on Computers,* vol.57, no.6, pg:809-820 June 2008.

[5] Jose Duato, "A Necessary and sufficient condition for Deadlock-free Adaptive Routing in Wormhole Networks." *IEEE Trans. Parallel and Distributed Systems,* vol.6, no.10, pg.1055-1067, October 1995.

[6] Luis Gravano, Gustavo D. Pifarre, Pablo E.Berman and Jorge L. C. Sanz, " Adaptive Deadlock and Livelock-Free Routing With all Minimal Paths in Torus Networks", *IEEE Trans. On Parallel and Distributed Systems,* vol.1, no.12,pg.1233-1251, Dec, 1994.

[7] V. Puente, R. Beivide, J.A. Gregorio, J.M. Prellezo, J.Duato and C. Izu, " Adaptive Bubble Router: A Design to Improve Performance in Torus Networks", *IEEE, International Conference on Parallel Processing*, pg.58 Sept, 1999.

[8] Wei Luo and Dong Xiang, "An Efficient Adaptive Deadlock-free Routing Algorithm for Torus Networks," *IEEE Trans. Parallel and Distributed Systems* vol.pp issue.99 2012.