Routing Algorithms Based on 2D Turn Model for Irregular Networks

Akiya Jouraku Michihiro Koibuchi Hideharu Amano
Department of Computer Science, Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522 Japan.
E-mail: {jouraku,koibuchi,hunga}@am.ics.keio.ac.jp

Akira Funahashi
Department of Information Technology, Mie University
1515 Kamihamacho, Tsu-City, Mie 514-8507 Japan.
E-mail: funa@arch.info.mie-u.ac.jp

Abstract

In order to solve traffic unbalancing caused by up/down* routing for irregular networks, two-dimensional direction is introduced into a spanning tree, and novel routing algorithms based on two-dimensional turn model are proposed. The proposed algorithms improve traffic balancing by selecting prohibited turns for deadlock-free carefully. Simulation results demonstrate that proposed algorithms improve both the performance and stability.*

1 Introduction

High performance distributed computing systems using commodity components, such as personal computers, interconnected with a high-speed network [1],[2],[3] have been widely developed because of their high performance per cost.

In such systems, switch-based irregular networks, such as Myrinet[1], are often required for high degree of flexibility and scalability of wiring. Irregularity of interconnection introduces difficulty on guarantee of connectivity and deadlock-free packet transfer, and spanning tree based routing algorithms[2] which use the connectivity and acyclicity of spanning trees are noticed as practical solutions.

Up*/down* routing is the most popular spanning tree based routing algorithm[2] and can be easily applied for both regular and irregular networks. By restricting packet transfer direction based on the turn model[4] on one-dimensional layered structure of a spanning tree, cyclic dependencies between channels are removed, and deadlock can be avoided. However, it is difficult to utilize network bandwidth effectively, since traffic balancing is hard to be cared by a simple one-dimensional turn model.

In this paper, we propose new routing algorithms which are based on two-dimensional turn model using a specific spanning tree called H/V tree.

2 Up*/Down* Routing

Up*/down* routing is the most popular deadlock-free routing algorithm for irregular networks, and has been used in Autonet[2] and Myrinet[1]. It can be easily applied for both regular and irregular networks because of its simplicity. In order to guarantee connectivity and deadlock-free for irregular networks, it exploits a spanning tree based directed graph in which *up* or *down* direction is assigned to each network channel.

The traditional spanning tree used in Autonet is the BFS (Breadth-First Search) spanning tree. The MDST (Minimum Depth Spanning Tree)[2] is one of the BFS spanning tree. In the MDST, topological distance of each node from the root is always minimum.

After building the spanning tree, a directed graph is constructed by assigning up or down direction to each network channel based on one-dimensional layered structure of the spanning tree as follows. The “up” end of each link is defined as: (1) the end whose switch is closer to the root in the spanning tree; (2) the end whose switch has the lower ID, if both ends are at the switches with the same tree level.

Up*/down* routing avoids the deadlock using the turn model[4]. In this model, all directions of packet turns and their cycles in the target network are analyzed. Then, just enough of the turns to break all of the cycles are prohibited.

Since only one-dimensional direction: up or down is included in up*/down* directed graph, only two turns and one cycle are detected. Thus, cyclic dependencies between channels are broken based on a simple restriction: a packet must be transferred by using channels with the up direction (if needed) followed by channels with the down direction (if needed). This restriction prevents a packet turning from the down direction to the up direction.

Although deadlock is avoided with this simple restriction, there is no flexibility about selecting prohibited turns in this method. Thus, a pair of prohibited turns between two links is always formed as shown in Figure 1. In Figure 1,a pair of prohibited turns is formed at node B and three pairs of prohibited turns are formed at node A. This concentration of prohibited turns in up*/down* routing may cause an un-

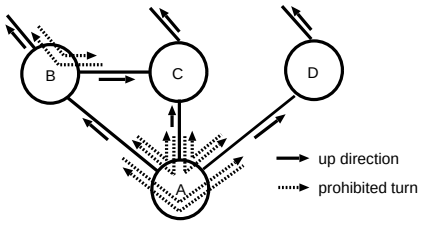

Figure 1. A pair of prohibited turns in up*/down* routing

balanced traffic that degrades effective network bandwidth.

3 2D Turn Model Based Routing Algorithms

3.1 Construction of H/V Graph

Here, we extend the one-dimensional directed graph used in up*/down* routing, and introduce two-dimensional directed graph which is called H/V graph.

H/V graph is constructed based on a spanning tree as follows.

1. Building a BFS spanning tree

First, the BFS spanning tree is built as the same manner for up*/down* routing, such as the MDST. After building a spanning tree, *depth* is assigned to each node. The depth is minimal distance from the root node in the vertical direction and used to determine the vertical direction(*up* or *down*) of each channel like up*/down* routing.

For example, Figure 2(a) shows the assignment of the depth to a 9-switch irregular network. As shown in Figure 2(a), the same depth can be assigned to different nodes.

2. Assignment of width to each node

In order to construct two-dimensional directed graph, we assign *width* to each node for introducing the horizontal direction: *left* or *right*.

The width of each node is determined by pre-order traversal from the root node. An ascending integer is assigned to each node in the order of visiting during the traversal. After the assignment of the width, two-dimensional coordinates of each node are determined. Coordinates of the node N are represented as $C_N = (x, y)$ where x and y are the width and the depth of the node N respectively.

For example, the assignment of the width to the network in Figure 2(a) is shown in Figure 2(b). As shown in the figure, the width of each node is always unique by the pre-order traversal and so the two-dimensional coordinates of each node are also unique.

Since two or more children nodes can be selected as the next visit node in the pre-order traversal, several selection rules can be applied. Thus, various H/V graphs can be built from the same target network.

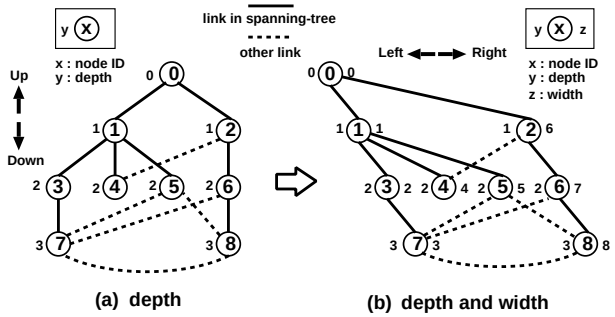

Figure 2. Assignment of depth and width

3. Assignment of directions to each channel

Here, the vertical direction and the horizontal direction are assigned to each channel based on the two-dimensional coordinates of each node.

First, the horizontal direction is assigned to each channel. The horizontal direction for the channel from node A to node B is determined by the following conditions. When the coordinates of node A and B are $C_A = (x_A, y_A)$ and $C_B = (x_B, y_B)$ respectively:

1. if $x_A > x_B$, then *left* direction is assigned, and
2. if $x_A < x_B$, then *right* direction is assigned.

Next, the vertical direction is assigned to each channel. The vertical direction for the channel from node A to node B is determined by the following conditions:

1. if $(y_A > y_B) \cup ((y_A = y_B) \cap (x_A > x_B))$, then *up* direction is assigned, and
2. if $(y_A < y_B) \cup ((y_A = y_B) \cap (x_A < x_B))$, then *down* direction is assigned.

Finally, one of the four H/V directions is assigned to each channel based on the assigned horizontal and vertical direction. H/V direction of each channel $HV(h, v)$ is defined with a pair of horizontal direction (h) and vertical direction (v). That is, the direction of $HV(left, up)$ channel is called *left-up(LU)* direction. Similarly, the direction of $HV(left, down)$ channel, $HV(right, up)$ channel and $HV(right, down)$ channel are called *left-down(LD)* direction, *right-up(RU)* direction, and *right-down(RD)* respectively. In this paper, we call a channel with dir H/V direction as dir channel. These H/V directions are used for the definition of our turn model based routings. After the assignment of the H/V direction to each channel, H/V graph is constructed.

Figure 3 shows the H/V graph for the network in Figure 2. The subgraph of H/V graph, which consists of only channels in the spanning tree, is called H/V tree.

3.2 Definition of 2D Turn Model Based Routing Algorithms

Deadlock-free routing algorithms are defined by applying the two-dimensional turn model to the H/V graph as follows.

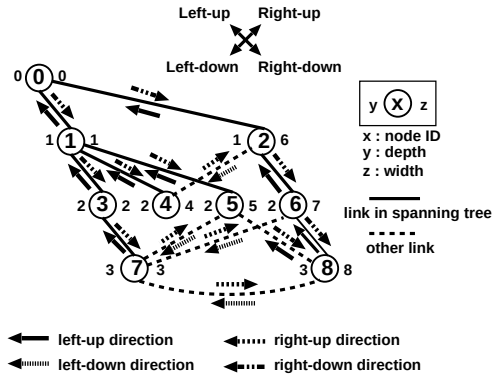

Figure 3. H/V Graph

1. Identifying all possible turns

First, we represent a packet turn from a direction $prev$ to another direction $next$ as $T_{prev,next}$.

Since there are four H/V directions in H/V graph, the number of possible turns is twelve as shown in Figure 4. Figure 4 shows all possible turns from one H/V direction to another H/V direction in H/V graph. In Figure 4, there exist eight 90-degree turns and four 180-degree turns.

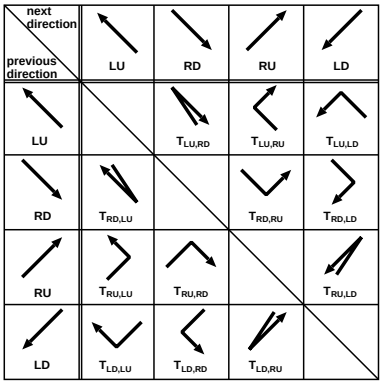

Figure 4. All possible turns in H/V Graph

2. Identifying all possible cycles and choosing the prohibited turns

We identify all possible cycles formed by the turns shown in Figure 4, and prohibit only turns enough to break all cycles. For better traffic balancing, prohibited turns are chosen so as to avoid the concentration of prohibited turns as possible.

Here, a dependency from turn T_i to T_j is represented as $Dep(T_i, T_j)$ if T_j can be formed after T_i. For example, $Dep(T_{up,down}, T_{down,up})$ is formed in up*/down* routing. Assume that a set of turns $\{T_0, T_1, ..., T_{n-1}\}$, where n is the number of turns, forms a cycle in H/V graph. The cycle $\{Dep(T_i, T_j) \mid j = (i+1) \bmod n, i = 0, 1, ..., n-1\}$ is represented as $L(T_0, T_1, ..., T_{n-1})$.

First, we identify four simple cycles in H/V graph. Since every pair of nodes is connected through links belong to the spanning tree in H/V tree, if one link which doesn't belong to H/V tree ("other link") connects some two nodes, then two cycles through the two nodes and their ancestor node are always formed by the "other link" and the spanning tree links.

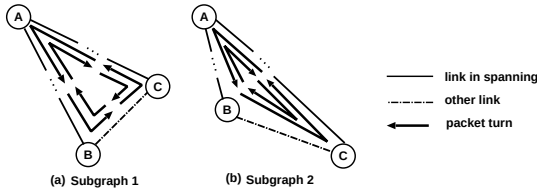

Figure 5. Four possible cycles in subgraphs of H/V graph

Figure 5(a),(b) show two simple subgraphs which introduce such two cycles. In Figure 5(a),(b), node B and C are connected with a link which does not belong to the spanning tree, and node A is their ancestor node. The difference between the two subgraphs is the relative position of node B and C. As shown in the figures, there exist two cycles in each subgraph. The cycles in Figure 5(a) are $L_a(T_{LU,RD}, T_{RD,RU}, T_{RU,LU})$ and $L'_a(T_{LU,RD}, T_{RD,LD}, T_{LD,LU})$. On the other hand, the ones in Figure 5(b) are $L_b(T_{LU,RD}, T_{RD,LU})$ and $L'_b(T_{LU,RD}, T_{RD,LU})$. Since L_b and L'_b are logically equivalent, only L_b will be considered.

In order to break the four cycles, one of the turns in each cycle is prohibited based on the following two policies: (1) $T_{LU,RD}$ must not be prohibited to guarantee the connectivity, because $T_{LU,RD}$ can be formed when a packet transfers between spanning tree channels. (2) Selected prohibited turns should not form the concentration as shown in Figure 1 as possible. Considering these policies, $\{T_{LD,LU}, T_{RU,LU}\}$ or $\{T_{RD,RU}, T_{RD,LD}\}$ are chosen as prohibited turns to break L_a and L'_a.

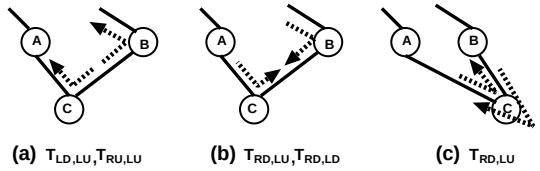

Figure 6. Prohibited turns for H/V graph

As shown in Figure 6(a) and (b), the distribution of prohibited turns is achieved. However, in order to break L_b, we are obliged to prohibit $T_{RD,LU}$, which introduces the concentration of prohibited turns as shown in Figure 6(c), because $T_{LU,RD}$ can't be prohibited.

Eventually, one of the following two turn sets, P_1 or P_2, can be chosen for breaking the four cycles.

$$P_1 = \{T_{LD,LU}, T_{RU,LU}, T_{RD,LU}\}$$
$$P_2 = \{T_{RD,RU}, T_{RD,LD}, T_{RD,LU}\}$$

Next, we identify the remaining cycles which don't include the above prohibited turns. Here, it is assumed that P_1 is chosen as prohibited turns in the following discussion.

Although three turns in P_1 are prohibited, the other nine turns in Figure 4 can form cycles. The nine turns can be classified into $Q_1 = \{T_{LU,i} \mid i \neq LU, i \in H\}$ and $Q_2 = \{T_{i,j} \mid i \neq LU, j \neq LU, i \neq j,\ i,j \in H\}$ where $H = \{LU, LD, RU, RD\}$.

Theorem 1 *Cycles including a turn in Q_1 includes one in P_1* □

Proof Assuming that a cycle $L_c(T_0, T_1, ..., T_x, ..., T_{n-1})$ which includes a turn $T_x(x = 0, 1..., n-1)$ in Q_1 but not the turn in P_1 can be formed. Since T_x is formed when a packet transfers from *LU* direction to another direction, a turn which can be formed just before T_x must be the one in $\{T_{i,LU} \mid i \neq LU, i \in H\}$. However, the turn set is equivalent to P_1, and this contradicts the above assumption. Therefore, cycles including a turn in Q_1 includes one in P_1. □

From Theorem 1, there are no requirement to prohibit Q_1 , since cycles include Q_1 are broken by P_1.

As a result, all possible cycles including a turn with *LU* direction are broken. Thus, the remaining possible cycles are only ones consisting of turns in Q_2 which include no *LU* direction. In order to identify such cycles, we show TDG(Turn Dependency Graph) for Q_2 as shown in Figure 7.

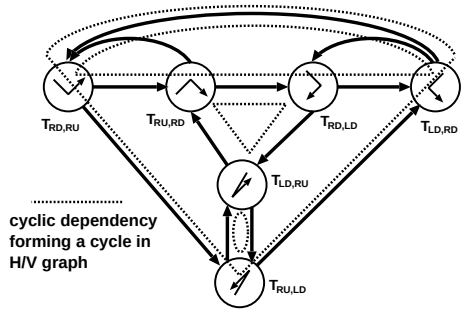

Figure 7. Turn Dependency Graph for Q_2

In Figure 7, each node represents one of the turns in Q_2 and each arrow represents a dependency between two turns. All possible cycles by the turns in the TDG are based on one of the four cycles as shown in Figure 7 with dotted cycles.

The four cycles are represented as follows.

1. $L_1(T_{RU,RD}, T_{RD,LD}, T_{LD,RU})$
2. $L_2(T_{RD,RU}, T_{RU,LD}, T_{LD,RD})$
3. $L_3(T_{LD,RU}, T_{RU,LD})$
4. $L_4(T_{RD,RU}, T_{RU,RD}, T_{RD,LD}, T_{LD,RD})$

Figure 8 shows these four cycles.

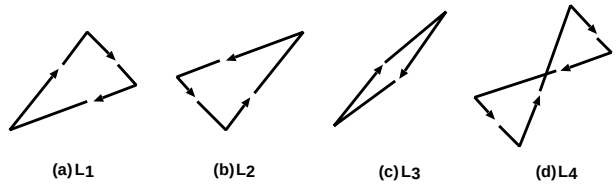

Figure 8. Possible four cycles formed by turns in Q_2

For breaking the four cycles, one of the following two turn sets, P_1' or P_1'', is prohibited based on the policies mentioned above.

$$P_1' = \{T_{LD,RU}, T_{LD,RD}\}$$
$$P_1'' = \{T_{RU,LD}, T_{RU,RD}\}$$

Finally, a turn set: $P_{1a} = P_1 + P_1'$ or $P_{1b} = P_1 + P_1''$ is chosen as prohibited turns.

Since all possible cycles with a turn including *LU* direction are broken by P_1 and the other possible cycles are broken by P_1' or P_1'', all possible cycles are broken. In almost the same way, the turn set $P_2' = \{T_{LD,RU}, T_{LU,RU}\}$ or $P_2'' = \{T_{RU,LD}, T_{LU,LD}\}$ is prohibited if P_2 is chosen instead of P_1. As a result, a turn set: $P_{2a} = P_2 + P_2'$ or $P_{2b} = P_2 + P_2''$, is alternative for breaking all cycles.

Theorem 2 *Deadlock-free is guaranteed by prohibiting one of the turn sets, P_{1a}, P_{1b}, P_{2a} and P_{2b}.* □

Proof All possible cycles are broken by prohibiting one of the turn sets. Therefore, deadlock-free is guaranteed. □

Theorem 3 *Connectivity between every pair of nodes is guaranteed when prohibiting one of the turn sets, P_{1a}, P_{1b}, P_{2a} and P_{2b}.* □

Proof The possible turn in H/V tree is only $T_{LU,RD}$ because it includes only *LU* or *RD* channels. Since $T_{LU,RD}$ is not prohibited, a packet transfer between every pair of nodes connected by H/V tree is allowed. Therefore, the connectivity between every pair of nodes is guaranteed. □

3. Cycle detection algorithm

In the following discussion, it is assumed that $P_1' = \{T_{LD,RU}, T_{LD,RD}\}$ is chosen as prohibited turns.

We defined four alternative sets of prohibited turns for breaking all possible cycles in H/V graph. However, the two turns in P_1', which are prohibited for breaking four cycles shown in Figure 8, don't always form cycles. Thus, if turns in P_1' are prohibited statically, some of the turns may be prohibited unnecessarily.

In order to alleviate the problem, a traversal search on the H/V graph is performed for detecting the above four cycles. By employing the cycle detection, only the turns in P_1' which form each detected cycle in the H/V graph are prohibited.

Here, the traversal algorithm for cycle detection is introduced. The traversal starts from all nodes which meet one of the following conditions.

(a). One or more output *RU* channels and one or more output *RD* channels are connected(forming $T_{LD,RD}$ in P_1')

(b). Two or more output *RU* channels are connected (forming $T_{LD,RU}$ in P_1')

The procedure of the traversal consists of the following two steps.

STEP1: Traversal from the node meeting the condition (a)

First, the traversal process visits an adjacent node which can be reached through an output *RD* channel from the target node. Once a channel is used for visit, it is marked so as not to be used again. Then, if there exists an available output channel, adjacent nodes are recursively visited in the way of the depth-first search. A channel is available if it meets the below conditions:

1. not the *LU* channel(doesn't form a turn in P_1),
2. not marked, and
3. doesn't form a turn which was prohibited by the previous traversal.

If there are no available output channels, the traversal process returns to the previous visited node. The target cycle is detected if the traversal process returns to the target node through an output *LD* channel from an adjacent node. Then, the turn $T_{LD,RD}$ from the *LD* channel to the first used *RD* channel is prohibited. The traversal process continues until there are no available output channels.

The above procedure is performed for each output *RD* channel of the target node in turn.

STEP2: Traversal from the node meeting the condition (b)

This traversal is performed in almost the same way except for the following conditions:

1. a channel used for the first visit is an output *RU* channel,
2. when a cycle is detected, the turn $T_{LD,RU}$ is prohibited.

In almost the same way, the traversal algorithm can be employed for P_1'', P_2', P_2'', respectively.

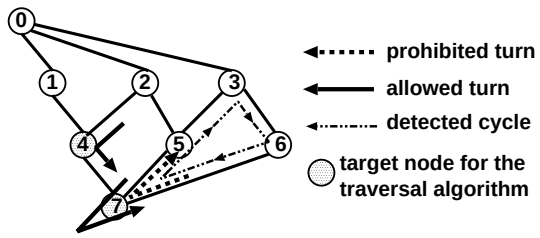

Figure 9. A detected cycle by the traversal algorithm

Figure 9 shows an example of a detected cycle by the traversal algorithm. In Figure 9, there exist two target nodes for the algorithm and three turns in P_1'. As shown in the figure, only one turn in P_1' is prohibited because the other turns form no cycle.

The computation cost for the algorithm is $O(ln^2)$, where n is the number of nodes and l is the number of links per node.

4. Definition of routing algorithms

Here, we define four routing algorithms based on the above four prohibited turn sets.

The routing algorithms based on the prohibited turn set P_1 are called **L-turn routing(Left-up first turn routing)**. Since all turns to the *LU* direction are prohibited in L-turn routing, a packet must start out in that direction in order to reach the destination node in *LU* direction. Especially, the routing algorithm which prohibits turns in P_1 statically and ones in $P_1'(P_1'')$ dynamically based on the cycle detection algorithm is called **L-turn/α(L-turn/β)**.

Similarly, the routing algorithms based on the prohibited turn set P_2 are called **R-turn routing(Right-down last turn routing)**. Especially, the routing algorithm which prohibits turns in P_2 statically and ones in $P_2'(P_2'')$ dynamically based on the cycle detection algorithm is called **R-turn/α(R-turn/β)**.

Since a packet can travel on any path without a prohibited turn, nonminimal paths are available. However, a hotspot is more likely to be formed when nonminimal paths are allowed in irregular networks. Thus, only minimal paths among all possible paths should be available in the proposed routing algorithms.

4 Performance evaluation

In this section, performance of the proposed routing algorithms is evaluated by computer simulation, and compared with the BFS based up*/down* routing. A flit-level simulator written in C++ was used for the simulation.

4.1 Network model

We evaluate each routing algorithms on both irregular and regular networks. Irregular networks include 64 switches, and 10 different topologies are randomly generated. For each topology, the spanning tree which provides minimum crossing paths is built. In case of tie, the spanning tree which provides the smaller average distance is used. Crossing paths are the maximum number of routing paths crossing through any network channel, and average distance is the average number of hops in minimal routing paths between all pairs of switches.

As a regular network, 8×8 2D torus is used. Each network switch has 8 ports. 4 ports are used for connecting other switches, and the rest are connected to processing elements.

4.2 Simulation parameters

Simulation parameters are as follows.

Each simulation is performed in $1,000,000$ clock cycles and the first $50,000$ clock cycles are ignored for the evaluation. Packet length is 128 flits and the traffic pattern is *uniform*. For switching technique, virtual cut-through is used. In the switch, flit transfer requires 3 clocks, that is, one for routing, one for transferring a flit from an input channel to output channel through a crossbar, and the rest for transferring the flit to the next node. Each routing algorithm selects only minimal paths among all possible paths.

4.3 Irregular networks

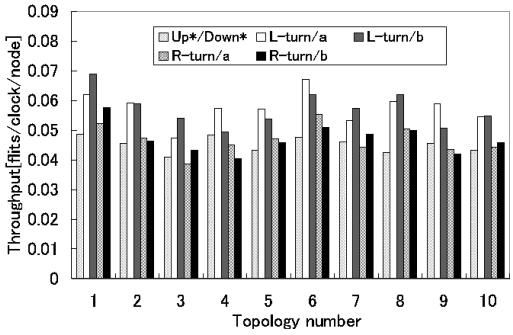

Figure 10. Throughput of 5 routing algorithms on 10 irregular networks with 64 switches

Figure 10 shows the throughput of 5 routing algorithms on 10 different irregular networks with 64 switches. As shown in Figure 10, two L-turn routings achieve higher throughput, approximately 27% in average, than up*/down* routing in each topology. On the other hand, two R-turn routings achieve only a small improvement, approximately 4%, compared with up*/down* routing.

Table 1. The average value of the performance metrics, Throughput, *PT*(the number of prohibited turns), *SDPT*(the standard deviation of the number of prohibited turns per node), *AD*(average distance) on 10 irregular networks with 64 switches

Routing algorithm	Throughput	*PT*	*SDPT*	*AD*
Up*/Down*	0.04518	193.2	3.669	3.844
L-turn/α	0.05763	184.0	2.225	3.793
L-turn/β	0.05717	185.4	2.269	3.789
R-turn/α	0.04684	177.0	2.006	3.703
R-turn/β	0.04705	184.5	2.310	3.731

Table 1 shows the average value of the performance metrics for each routing algorithm on 10 topologies. *SDPT* shows how uniformly the prohibited turns are distributed. Since the turn model based routings provide smaller *SDPT* than up*/down* routing as shown in Table 1, it can be said that the turn model based routings distribute the prohibited turns more uniformly. The turn model based routings also provide smaller value about *PT* and *AD* than up*/down* routing. However, the throughput of each R-turn routing is poor compared with each L-turn routing although they provide almost equal value about the other three metrics. The reason can be explained as follows. R-turn routings allow all turns toward LU direction except $T_{RD,LU}$, which leads packet toward the root node. On the other hand, they also prohibit all turns from RD direction, which restrain packet transfer toward the leaf nodes. Thus, traffic tends to concentrate around the root node and degrades the performance.

4.4 8 $\times$ 8 2D torus

Figure 11 shows the relation between average latency and accepted traffic of 5 routing algorithms on 8$\times$8 torus.

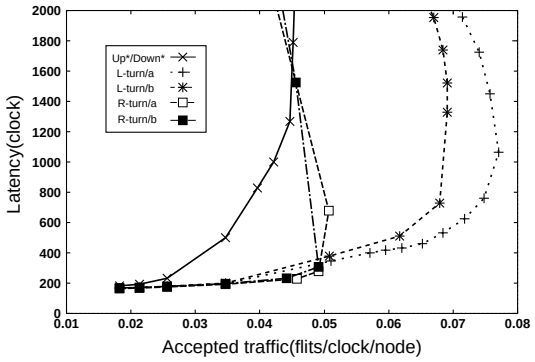

Figure 11. Latency versus accepted traffic for 8$\times$8 2D torus

As shown in Figure 11, two L-turn routings achieve much higher throughput than the other routing algorithms. Especially, L-turn/α achieves approximately 70% improvement compared with up*/down* routing. The performance of two R-turn routings is still poor compared with L-turn routings.

5 Conclusion

Four routing algorithms based on two-dimensional turn model using specific spanning tree called H/V tree are proposed and evaluated.

The concept of the proposed routing algorithms is to distribute the prohibited turns for better traffic balancing. In order to avoid prohibiting turns unnecessarily, the cycle detection algorithm is also proposed.

Result of simulations shows that proposed routing algorithms achieve better performance than traditional up*/down* routing both on irregular and regular topologies. Especially, L-turn routings offer much better performance than other ones.

References

[1] N.J.Boden et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–35, 1995.

[2] M.D. Schroeder et al. Autonet: A high-speed, selfconfiguring local area network using point-to-point links. *Technical Report SRC research report 59,DEC*, April 1990.

[3] T.Kudoh, S.Nishimura, J.Yamamoto, H.Nishi, O.Tatebe, and H.Amano. RHiNET: A network for high performance parallel computing using locally distributed computing. In *Proc. Innovative Architecture for Future Generation High-Performance Processors and Systems*, pages 69–73, November 1999.

[4] C.J.Glass and L.M.Ni. Maximally Fully Adaptive Routing in 2D Meshes. In *Proc. International Symposium on Computer Architecture*, pages 278–287, 1992.