L-turn Routing:
An Adaptive Routing in Irregular Networks

Michihiro Koibuchi† Akira Funahashi‡ Akiya Jouraku† Hideharu Amano†
†Dept. of Information and Computer Science
Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, 223-8522 Japan
{koibuchi,jouraku,hunga@am.ics.keio.ac.jp}
‡Dept. of Information Technology
Mie University
1515 Kamihamacho, Tsu-City,
Mie 514-8507, Japan
funa@arch.info.mie-u.ac.jp

Abstract

Network-based parallel processing using commodity personal computers has been widely developed. Since such systems require high degree of flexibility and scalability of wiring, a high-speed network with an irregular topology is often needed. In traditional routing algorithms for irregular networks, available paths are considerably restricted in order to avoid deadlocks.

In this paper, we propose a novel routing algorithm called Left-up-first turn routing (L-turn routing), which makes a better traffic balancing in irregular networks by building a specific spanning tree. Result of simulations shows that L-turn routing achieves better performance than traditional ones with each topology.

1 Introduction

Network-based parallel processing using commodity components, such as personal computers(PC) or workstations(WS), has received attention as an economical high performance parallel-computing environment [10],[11],[4],[12].

In such systems, a free topology high-speed network is often required in order to cope with flexibility and scalability of wiring. Since irregularity introduces difficulty for deadlock-free routing, the performance of irregular networks tends to be insufficient compared with regular networks [3].

In order to make the best use of inherent bandwidth, adaptive routing which can avoid the congestion is essential for irregular networks. However, in traditional routing algorithms for irregular networks, available paths are considerably restricted in order to avoid deadlocks [10],[9].

Consequently, traditional routing algorithms which can not use all of links effectively often cause unbalanced traffic and increase average hops of packets.

In this paper, we propose a novel adaptive routing algorithm called Left-up-first turn routing (L-turn routing) for irregular networks. L-turn routing is based on a logical spanning tree like traditional routing algorithms. However, it distributes the traffic by building a novel directed-graph called the L-R directed-graph.

Consequently, all channels are effectively used in various types of topologies.

L-turn routing also mitigates a traditional problem by which a traffic concentrates around the root node in up/down routing.

The following sections are organized as follows. In Section 2, traditional up/down routings in irregular networks are introduced. In Section 3, we will propose a new routing algorithm in irregular networks called L-turn routing. Section 4 shows the performance evaluation results. Section 5 is a conclusion and future work.

2 Up/down based routings

Recently, various routing algorithms in irregular networks have been proposed [9],[4],[7] and up*/down* routing has already been implemented on Autonet [10].

In this section, two typical traditional routing algorithms in irregular networks are introduced.

2.1 Up*/down* routing in Autonet

Up*/down* routing in Autonet [10] is known as a simple routing algorithm which can be used as a basis of various
routing algorithms for irregular networks since it is an effective method of deadlock avoidance.

At first, a target irregular network is mapped into a directed-graph based on the BFS (Breadth First Search) spanning tree, and the routing algorithm is built using the layered structure of the tree. Based on the structure of a spanning tree, a channel that faces toward the root node is called up channel and a channel that goes away from the root node is called down channel.

Routing algorithm is quite simple. The packet is transferred by down channel (if any) after it has been transferred by up channel (if any). Since the cyclic dependency is broken between down channels and up channels, up*/down* routing is deadlock-free.

2.2 Prefix routing

Prefix routing proposed by Wu and Sheng[9] is similar to up*/down* routing in Autonet but channels which are out of a spanning tree are utilized as short-cuts called "cross channels".

Prefix routing is based on a labeling scheme that assigns a label into each node and channel. \(L(v)\) and \(L(u, v)\) are used as labels for node \(v\) and channel from node \(u\) to node \(v\), respectively.

Assignment of labels to nodes and channels is done as follows. At first, the label of the root is 1, i.e., \(L(r) = 1\). If \(u\) is the \(k\)th child of \(v\), then assign \(L(u) = L(v) \| k\), where \(\|\) represents a concatenation operation. If node \(v\) is the parent of node \(u\), then \(L(v, u) = L(u)\) and \(L(u, v) = e\), and in the other case, \(L(v, u) = L(u), L(u, v) = L(v)\). where \(e\) represents an empty string label (see Figure 2).

All channels are classified into up channel, down channel and cross channel. The cross channel is out of spanning tree.

Routing algorithm is as follows:

- At an intermediate node \(v\) (including source node \(s\)), neighbor \(u\) is selected as the forwarding node if \(L(v, u)\) is a prefix \(L(d)\), where \(d\) is the destination.

- If such a neighbor does not exist, select a neighbor \(w\) such that \(L(v, w) = e\.

In the example shown by Figure 2, when a packet is transferred from \(e(112)\) to \(a\) with the up*/down* routing, a path can be select from 2 candidates \((e \rightarrow b \rightarrow a)\) and \((a \rightarrow c \rightarrow a)\). On the other hand, when Prefix routing is used, the path takes one as \((e(112) \rightarrow b(11) \rightarrow a(1))\).

Since a path is decided only by looking up the prefix, Prefix routing has an advantage that the routing table is smaller than other up/down based routings, and deadlock-free is also guaranteed since cyclic dependency is broken between down channel and up channel.

3 L-turn routing

3.1 Motivation

In up/down based routings, all channels are classified into up or down channel, and a strict rule between them is applied. So they cannot use all of links effectively, and often cause unbalanced traffics. Especially, the heavy traffic
around the root node causes a congestion which degrades the total performance.

Silla and Duato proposed an efficient adaptive routing algorithm which is called minimal adaptive routing[4]. It guarantees deadlock-free without excluding cyclic channel dependency although physical channels are duplicated or split into two virtual channels called an original channel (escape paths) and a new channel. However, the minimal adaptive routing cannot be applied to algorithms based on source routing.

Here, in order to distribute the traffic, the structure of directed-graph itself is modified, and L-R tree and L-R directed-graph are defined first. Then, left/right routing, which is a fundamental routing on the L-R directed-graph, is introduced. Finally, an advanced method called Left-up-first turn routing or L-turn routing is introduced.

3.2 Building an L-R tree

First, an irregular network is mapped into a tree called L-R tree in which horizontal direction (left or right) is assigned to each channel between any two nodes.

An L-R tree is built as follows.

a. Building a BFS spanning tree

A BFS spanning tree is built with the same way in traditional up/down based routings.

Figure 3.a shows a spanning tree used in traditional up/down based routings. Note that a link provides bidirectional channels.

Here, an arrow with a link shows the up direction in the spanning tree.

b. Assignment of the width to every node

An increasing number (width) is assigned to each node in the order that nodes are visited by the pre-order traversal starting from the root node.

In the example shown by Figure 3.b, the order of the traversal is \((0,1,3,7,4,5,2,6,8)\). Width represents a horizontal distance from the root node.

c. Assignment the horizontal directions to all channels

Horizontal direction is assigned to each channel between any two nodes by comparing their widths as follows:

- **Left direction** is assigned to the channels whose destination nodes have the larger widths, and **right direction** is assigned to the other channels.

d. Assignment the vertical directions to all channels

Vertical direction (up or down direction) is assigned to each channel between any two nodes by comparing their depths (distances from the root node in vertical direction) as in up*/down* routing.

In Figure 3.b, an arrow with a thick line shows the left direction, and an arrow with a dotted line shows the up direction. Unlike Figure 3.a, a L-R tree has two dimensional directions: up/down and left/right.

A L-R tree does not include any channels which are out of spanning tree in the original irregular network, and the
L-R directed-graph is built by adding such channels into L-R tree. Thus, the L-R directed-graph is topologically equivalent to the original irregular network.

Channels with the dotted lines in Figure 3 are added in both cases that an up/down directed-graph and L-R directed-graph are built. Eliminating them makes an up/down tree or L-R tree in each case.

Note that different L-R trees can be generated from an original network depending on the assignment of width to the nodes in a BFS spanning tree, and the L-R tree is similar to traditional up/down tree but it has not only vertical direction (up or down) but also horizontal direction (left or right) between any two nodes.

3.3 Left/right routing algorithm

For introducing the left/right routing, the left channel and right channel are defined as follows.

**Definition 1 (left/right channel)** A channel that faces toward the left direction is called left channel, and a channel that faces toward the right direction is called right channel.

If the target node is 4 in Figure 3.b, right channel is a channel from node 4 to node 2 and left channel is a channel from node 4 to node 1.

**Definition 2 (left/right routing)** The only restriction of the left/right routing algorithm is as follows in L-R directed-graph.

- Don’t use left channel after using right channel.

Since any route which satisfies an above restriction can be selected, left/right routing is a non-minimal partially adaptive routing. It requires no virtual channel for deadlock avoidance in each link.

**Theorem 1** Left/right routing is deadlock-free.

**Proof**: A packet is routed by right channels (if any) after routing by left channels (if any). Since left/right routing brakes the cyclic channel dependency between left channels and right channels, it is deadlock-free.

**Theorem 2** Left/right routing guarantees any paths between any pair on nodes.

**Proof**: Each left channel and right channel which belongs to the L-R tree corresponds to each up channel and down channel which belongs to the corresponding up/down tree respectively. Consequently, the path between any pair of nodes is guaranteed by selecting a sequence of left channels (if any) followed by a sequence of right channels (if any).

To prevent a livelock in left/right routing, a packet must select the path whose length is equal to or shorter than the one using only channels which belong to L-R tree.

The restriction in left/right routing is different from the one in up/down based routings. For example, in Figure 3.b, a packet from node 3 to node 8 is transferred with 3 hops(3 → 7 → 6 → 8). On the other hand, in Prefix routing or up*/down* routing, a packet from node 3 to node 8 is transferred by 5 hops(3 → 1 → 0 → 2 → 6 → 8). Further, in left/right routing, this 5 hops route can also be selected.

Next, although a packet from node 7 to node 2 is transferred with 2 hops route (7 → 6 → 2) in up*/down* routing, a packet from node 7 to node 2 in left/right routing is transferred with 4 hops route (7 → 3 → 1 → 0 → 2 or 7 → 3 → 1 → 4 → 2). Left/right routing takes a disadvantage in this case but a packet in left/right routing can be transferred to a route which does not pass the root node.

3.4 L-turn routing algorithm

For introducing L-turn routing, left channels and right channels are further classified as follows.

**Definition 3 (LU/LD/RU/RD channel)**

- A left channel that faces toward the up direction is called left-up channel, and a left channel that faces toward the down direction is called left-down channel in L-R directed-graph.

- A right channel that faces toward the up direction is called right-up channel, and a right channel that faces toward the down direction is called right-down channel in L-R directed-graph.

In L-turn routing, if two connected nodes are at the same depth, the one at the right side is assumed to be closer to the root node in vertical direction.

An example of a position at the same depth is shown in Figure 4.

From definition 3, all channels in L-R directed-graph are classified into left-up, left-down, right-up and right-down channel.

**Definition 4 (L-turn routing)** The restrictions of the L-turn routing algorithm are as follows in L-R directed-graph.

- Don’t use left-up channel after using the channel except left-up channel.

- If a cyclic channel dependency which does not include any left-up channels exists, the turn from left-down channel to right direction in the cyclic channel dependency is not allowed.
The former restriction does not guarantee deadlock-free in only cases that a cyclic channel dependency which does not include any left-up channels may exist.

All possible cyclic channel dependencies which don’t include any left-up channels are based on one of the four patterns in Figure 7, and always include at least one turn from left-down channel to right-up or right-down channel. So, the latter restriction breaks all cyclic channel dependencies which don’t include any left-up channels.

For example, the latter restriction is applied to the cases as shown in Figure 8, but not applied to the cases as shown in Figure 9.

In order to apply the latter restriction, an algorithm which detects the above two turns is required. The outline of the algorithm is described as follows.

1. All nodes which satisfy one of the below conditions are selected:
   (a) two or more right-up channels exist (generating the turn from left-down to right-up), or
   (b) one or more right-up channels and one or more right-down channels exist (generating the turn from left-down to right-down).

2. For each selected node, in the order of the depth-first search, it is checked whether there are cyclic channel dependencies which include selected node and don’t include any left-up channels.

3. If the selected node satisfies 1-(a), the check is executed as follows. At first, an adjacent node which can move from the selected node by using right-up channel is visited. The output channel used for the visit is marked. Then, an adjacent node is visited by using available output channels recursively. If there are no available output channels or cyclic channel dependency is detected at a visiting node, the checking process returns to the previous node. An available output channel satisfies below conditions:
   (a) not left-up channel,
   (b) not marked, and
   (c) doesn’t make turn which is already prohibited by previous search.

If the selected node is visited again through left-down channel, the cyclic channel dependency which includes the turn from left-down to right-up, and no left-up channels is detected. Then, the above turn is prohibited. The check continues until there are no available output channels or all possible cyclic channel dependencies are detected.

If the selected node satisfies 1-(b), the check is executed in almost the same way except for below conditions.

(1) The check starts by using right-down channel.
(2) In order to reduce redundant prohibited turns, output channel which makes turn from left-down to right-up is not available.

This algorithm decreases the redundant prohibited turns as shown in Figure 5 because of its 3-(2). The computation cost of this algorithm is $O(n^2)$ where $n$ is the number of nodes.

By definition 2 and 4, L-turn routing and left/right routing restrict the turns as shown in Figure 6.a and 6.b respectively. In Figure 6, the turns with thin dotted lines are not allowed and the turns with thick dotted lines are not allowed only when it is prohibited by the above algorithm.

As shown in Figure 6.b, the turns which are opposite to each other are prohibited in left/right routing. This makes the concentration of prohibited turns at the same node and may cause unbalanced traffic. Similar disadvantage also applies to up/down based routing. On the other hand, L-turn routing
Figure 6. L-turn routing and left/right routing

(a) Left-up-first turn routing  (b) Left/right routing

Figure 7. Cyclic channel dependencies in specific cases

Figure 8. Cases in which turns from left-down channel to right direction are prohibited

prevents the above situation by distributing the prohibited turns as shown in Figure 6.a. Moreover, by definition 3, prohibited turns at two connected nodes at the same depth are distributed uniformly as shown in Figure 4. The distribution of prohibited turns in L-turn routing will contribute to better traffic balancing and so high throughput.

**Theorem 3** L-turn routing is deadlock-free.

Proof: L-turn routing breaks all possible cyclic channel dependencies by prohibiting at least one of turns which belongs to each cyclic channel dependency by definition 4. Consequently, L-turn routing is deadlock-free.

**Theorem 4** L-turn routing guarantees any paths between any pair on nodes.

Proof: Each left-up channel and right-down channel which belongs to the L-R tree corresponds to each up channel and down channel which belongs to the corresponding up/down tree respectively. Consequently, the path between any pair of nodes is guaranteed by selecting a sequence of left-up channels (if any) followed by a sequence of right-down channels (if any).

To prevent a livelock in L-turn routing, a packet must select the path whose length is equal to or shorter than the one using only channels which belong to L-R tree.

L-turn routing has the following useful property:

**Flexibility for various irregular topologies:** Since various irregular topologies must be treated in cluster computing, the routing algorithms should work well in such networks.

In general, up/down routing tends to cause a congestion around the root node because of its tree structure and assignment of a direction. It is difficult to distribute such traffic based on up and down direction [10],[9] although some methods[7],[8],[6] improved the situation.

Since L-turn routing distributes the prohibited turns as shown in Figure 6.a, the traffic in L-turn routing is more uniformly distributed compared with one in left/right routing and up/down based routings in many cases, and thus, it will work well in various types of irregular networks.

In an example of mesh, if an edge node is selected as a root node and L-R directed-graph is built as shown in Figure 11, L-turn routing is the same as west-first turn model[2]. Consequently, L-turn routing achieves same performance as one for regular network in this case, even though some adaptive routings in irregular networks do not work well in regular networks.
Recently, up*/down* routing computing a depth-first search (DFS) spanning tree with heuristic rules instead of BFS spanning tree is proposed [6]. This methodology achieves better performance by reducing prohibited turns and average distance.

But up*/down* routing essentially has pairs of prohibited turns which are opposite to each other at the same node as shown in Figure 10.b. On the other hand, L-turn routing does not have any such pair of prohibited turns as shown in Figure 6.a, and Figure 10.a.

4 Performance evaluation

In this section, performance of L-turn routing and left/right routing are evaluated by the computer simulation, and compared with others (Prefix and up*/down* in Autonet).

4.1 Network model

A flit-level simulator written in C++ was developed for analysis. Topology, network size, and packet length are selected just by changing parameters. Each node consists of two processors, request queues and a switching fabric which provides four bidirectional ports. Four neighboring nodes are connected using bidirectional links through ports. Here, a simple model consisting of channel buffers, crossbar, link controller and control circuits is used for the switching fabric.

As target topologies, small (16 nodes) and large (64 nodes) irregular networks are evaluated. For each scale, 10 different irregular networks are generated and analyzed. Each topology is generated as follows: first, an acyclic graph is generated by connecting each node with \(\frac{N}{2}\) links randomly so that its connectivity is maintained. Next, \(\frac{N}{2} + 1\) links are added between arbitrary nodes until all four ports of a node are connected. To evaluate regular network which is still used in cluster computing, \(8 \times 8\) 2D torus is also used.

4.2 Simulation parameters

A destination node of a packet is determined by a traffic pattern used in the simulation. Here, the uniform traffic in which all destination nodes are selected randomly is used. Simulation parameters are set as shown in Table 1.

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. We ignored the first 10,000 clocks for the evaluation because the network isn’t stable in that period.

Although L-turn routing, left/right routing and up*/down* routing are non-minimal adaptive routings which can select any possible paths, it is assumed that they select a path from the shortest possible paths in the simulation.

An adaptive routing is generally influenced by its output selection function [1],[5]. Here, a simple selection function
Table 1. Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time</td>
<td>2,000,000 clocks (ignore the first 10,000 clocks)</td>
</tr>
<tr>
<td>The number of virtual channels</td>
<td>1 (no virtual channel)</td>
</tr>
<tr>
<td>Packet length</td>
<td>128 flits</td>
</tr>
<tr>
<td>Switching tech.</td>
<td>wormhole</td>
</tr>
<tr>
<td>Traffic pattern</td>
<td>uniform</td>
</tr>
<tr>
<td>Flit transfer time</td>
<td>3 clocks</td>
</tr>
</tbody>
</table>

which selects port number in increasing order is applied.

4.3 Irregular networks with 16 nodes

Table 2 demonstrates that L-turn routing achieves 17% improvement in average throughput compared with up*/down* routing. Moreover, L-turn routing has the smallest SD of throughput. It shows that the throughput of L-turn routing is stable in various irregular topologies.

<table>
<thead>
<tr>
<th>Table 2. Avg. throughput for 16 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. throughput(SD)</td>
</tr>
<tr>
<td>Prefix</td>
</tr>
<tr>
<td>up/down</td>
</tr>
<tr>
<td>left/right</td>
</tr>
<tr>
<td>L-turn</td>
</tr>
</tbody>
</table>

Table 3 shows the average number of total prohibited turns for 10 topology, the average standard deviation of the prohibited turns per node, and the average distance for all pairs of nodes.

Table 3 and Figure 12 show that the average SD of prohibited turns per node gives influence on the throughput. In L-turn routing, both the number of prohibited turns and the SD of prohibited turns per node are the smallest. When the prohibited turns are reduced and uniformly distributed in the whole network (that is, the sum of prohibited turns and the SD of prohibited turns per node become smaller), efficient routing is possible. Thus, the average distance between nodes can be reduced, and consequently the throughput is enhanced.

4.4 64 nodes irregular networks

Figure 13 shows simulation results of irregular networks with 64 nodes. The condition of simulation is identical to that of with 16 nodes except the scale of network.

Table 4 shows the average throughput and its standard deviation(SD) of 10 topologies shown in Figure 13.

As shown in Figure 13, L-turn routing always achieves better throughput than others. Compared with results of network with 16 nodes, the difference between L-turn routing and other routing algorithms becomes larger. This comes
from that L-turn routing keeps lower average number of total prohibited turns and average SD of prohibited turns per node. Especially, the influence of the average SD of prohibited turns per node on the throughput is large.

Table 4 demonstrates that L-turn routing achieves 33% improvement in average throughput compared with up*/down* routing. Like results with 16 nodes, L-turn routing has the smallest SD, and it demonstrates that its throughput is not so dependent on topologies.

Table 5. Avg. distance and prohibited turn for 64 nodes

<table>
<thead>
<tr>
<th></th>
<th>prohibited turn(total)</th>
<th>prohibited turn SD(per node)</th>
<th>average distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix</td>
<td>515.8</td>
<td>3.92</td>
<td>5.15</td>
</tr>
<tr>
<td>up/down</td>
<td>200.6</td>
<td>3.93</td>
<td>3.94</td>
</tr>
<tr>
<td>left/right</td>
<td>202.0</td>
<td>3.97</td>
<td>3.99</td>
</tr>
<tr>
<td>L-turn</td>
<td>188.3</td>
<td>2.56</td>
<td>3.82</td>
</tr>
</tbody>
</table>

From Table 5 and Figure 13, the same tendency with 16 nodes is observed, but the difference between other networks becomes great.

4.5 8×8 torus

Figure 14 shows simulation results on 8×8 torus under the same simulation conditions. As shown in Figure 14, L-turn routing achieves 41% improvement with respect of up*/down* routing. This comes from that L-turn routing has better flexibility not only for irregular networks but also the regular network.

Figure 15 shows the distribution of channel utilization on 8×8 torus when the throughput shown in Figure 14 is obtained. Here, a node whose node number is (0, 0) on the 2D torus is selected as the root node.

Note that, the uniform traffic is used in this simulation. Although 8×8 torus is a uniform topology, Prefix routing and up*/down* routing tend to gather many packets around the root node. Although left/right routing mitigates the traffic concentration around the root node, other hot spots are generated. On the other hand, in L-turn routing, traffic is well balanced as shown in the figure.

5 Conclusion

We proposed L-turn routing which is an efficient adaptive routing with high degree of flexibility for various spanning trees.

Since it prohibits one of turns which forms cyclic dependencies, L-turn routing is deadlock free. It also distributes the prohibited turns, and thus makes balanced traffic and decreases the number of average hops of packets. Result of simulations shows that L-turn routing achieves better performance than traditional ones both on irregular and regular topologies.

Our future work is to establish efficient distributed algorithms to build L-R tree and output selection function[5],[1]. Using well considered distributed algorithm, the performance can be increased in various irregular networks.

References

Figure 15. Distribution of channel utilization in 8×8 torus

(a) Prefix routing

(b) up*/down* routing in Autonet

(c) left/right routing

(d) L-turn routing