

Designing Deadlock-Free Turn-Restricted Routing Algorithms for Irregular Wormhole-Routed Networks

JENQ-SHYAN YANG AND CHUNG-TA KING*

Acer Communication & Multimedia Inc.

Taipei Business Center

Taipei, Taiwan 114, R.O.C.

E-mail: jsyang@lhit.edu.tw

**Department of Computer Science*

National Tsing Hua University

Hsinchu, Taiwan 300, R.O.C.

E-mail: king@cs.nthu.edu.tw

Irregular networks connected by wormhole-routed switches are becoming increasingly popular for building networks of workstations for cost-effective parallel processing. A primary strategy to achieve deadlock-free routing in such networks is to first configure the links in a network into some specific directions, and then prohibit the turns that a message may traverse. A routing algorithm imposes fewer turn prohibitions will have a higher adaptivity and thus a higher performance. In general, designing such a routing algorithm requires two basic components: (1) assigning link directions, and (2) determining a link-direction-based routing guideline. In this paper we examine various assignment rules and routing guidelines, from which different heuristics and criteria are proposed to construct a good routing algorithm. Their effectiveness in reducing turn prohibitions is investigated, and in most cases the minimum turn prohibitions can be achieved. For a connected network with N switches and M links, the complexity of finding the set of turn prohibitions using our proposed method is $O(N \times M)$.

Keywords: adaptive routing, deadlock freedom, irregular network, routing algorithm, wormhole routing

1. INTRODUCTION

Switch-based networks with wormhole switching [10] are becoming increasingly popular in building networks of workstations for cost-effective parallel processing. In wormhole switching, the switches require only small buffers and the message latency is almost insensitive to the transmission distance in the network [5]. Examples of wormhole switches include DEC GIGAswitch [9] for FDDI network, the HP FCS and Ancor FCS 266/1062 switches [1] for FC (Fibre Channel) networks, the HP EtherTwist LAN switch and the IBM 8271 EtherStream Switch [11] for switched Ethernet, the Myricom Myrinet [7], the DEC Autonet [6], and the Tandem ServerNet [8]. Usually the switches

Received February 11, 1999; revised January 14 & May 11, 2000; accepted June 27, 2000.

Communicated by Jean-Lien C. Wu.

* This work was supported in part by the National Science Council, R.O.C., under Grants NSC-86-2213-E-007-043 and NSC-86-2622-E-009-0102.

are interconnected in irregular topologies in order to provide wiring flexibility for scalable systems and incremental expansion capability.

Message routing in wormhole switch-based networks is usually difficult and prone to deadlock, because a blocked message may hold the buffer resources in several switches. This difficulty is further compounded by the irregularity in the network topology. One solution, as proposed in [11], is to use an *Eulerian Trail* to help maintain a sequence in channel dependence. By routing messages following the order, the resultant routing algorithm is deadlock free.

DEC Autonet [6] adopts another approach in which links connecting the switches are first identified as either the *up* or *down* direction. A legal route must pass through zero or more links in the “up” direction, followed by zero or more links in the “down” direction. In other words, a message which is using a down link cannot make a turn to up links. By imposing the turn prohibitions in this way, all cyclic channel dependencies in the network can be eliminated, while the network remains reachable. The resultant routing algorithm is thus deadlock free. Such a routing method is referred to as the *up*/down** routing.

In [4], the above turn prohibitions are relaxed by properly adding duplicate channels among the spare switch ports. A legal route can now pass through zero or more “new” channels, followed by zero or more links in the “up” direction, and finally followed by zero or more links in the “down” direction. By relaxing the turn prohibitions with a tolerable addition of hardware (duplicated links), the resultant routing algorithm has a higher adaptively, which leads to a lower message latency and a higher effective network bandwidth. In [2] a general methodology was proposed to design deadlock-free routing algorithms which minimize the number of prohibited turns. The methodology is based on recursive partitioning of a graph model of the network and removal of cross-partition edges to eliminate cycles. It is applicable to irregular networks, and [] showed that the minimum number of prohibited turns can be obtained in a few hundred runs, using the example network shown in Fig. 2.



Fig. 2. An example of an irregular network.

In general designing a turn-prohibited routing algorithm like up*/down* routing requires two basic components: assigning link directions and determining a link-direction-based routing guideline. In this paper we examine various direction assignment rules and routing guidelines. Different heuristic rules and criteria are proposed to form the assignment rules and routing guidelines. Their effectiveness in constructing a routing algorithm with minimized turn prohibitions is then studied. In most cases the minimum number of prohibited turns can be achieved. For a connected network with N switches and M links, the complexity of finding the set of prohibited turns using our proposed method is $O(N \times M)$.

The rest of the paper is organized as follows. Section 2 introduces necessary background, including the generic wormhole switch model, a graph representation of irregular networks, and the DEC Autonet. Basic principles for designing deadlock-free turn-prohibited routing algorithms are discussed in Section 3. Different components in the routing algorithms are introduced, and various criteria and heuristic rules for designing good algorithms are described. In Section 4, performance of the resultant routing algorithms is investigated and compared with previous approaches. Finally, Section 5 concludes the paper.

2. ROUTING IN SWITCH-BASED IRREGULAR NETWORKS

2.1 Network Model

The structure of a generic wormhole-routed k -port switch is shown in Fig. 1. Each port is associated with a pair of input and output links (or a *bidirectional* link). Each port can connect to a processor or the port of another switch. It can also be kept open for a future connection. Each port contains an input and an output flit buffer. The input flit buffer is assumed to be large enough to hold at least the header flit of an incoming message for making routing decisions. A $k \times k$ crossbar is used within the switch to provide simultaneous connections between the input and output flit buffers. When a message is blocked because there is no available output flit buffer, it may also occupy the flit buffers in the switches that are along its path.

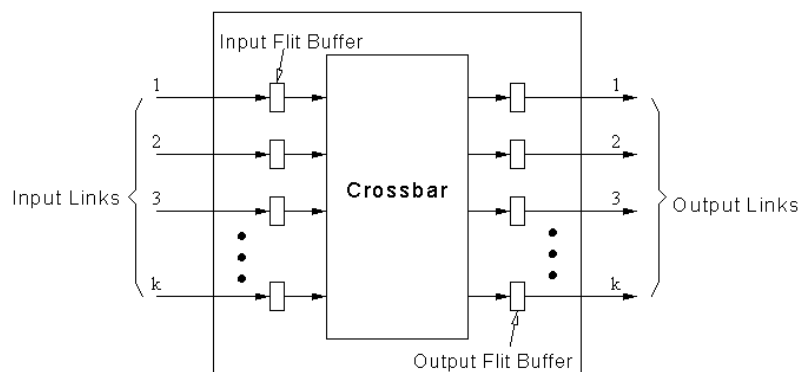


Fig. 1. Structure of a generic wormhole-routed k -port switch.

Fig. 2(a) shows an example switch-based irregular network, which was adopted from [4]. Each switch in the network is a wormhole-routed 8-port switch. In general, the topology of a switch-based network can be represented by a graph $G(V, E)$, where V is the set of switches and E is the set of bidirectional links between switches. The graph corresponding to the irregular network in Fig. 2(a) is shown in Fig. 2(b). Note that all edges in the graph are bidirectional, and that multiple edges between nodes are allowed.

2.2 Message Routing in DEC Autonet

As mentioned in Section 1, links in DEC Autonet are classified as in either the *up* or *down* direction. Then, deadlock can be avoided by prohibiting the messages from taking the *down/up* turns. The task was achieved by assigning directions to communication links and prohibiting some types of turns.

To assign the link directions, Autonet builds a BFS spanning tree rooted at the switch with the smallest ID. Then, links have an up direction if they connect a child switch to a parent switch. They are in the down direction if they connect a parent switch to a child switch. Links which connect switches at the same level are called *cross edges*. A cross edge has a down direction if it links a switch with a lower ID to another with a higher ID. An up cross edge is similarly defined. After the link directions were determined, the down/up turns are chosen to be prohibited.

Fig. 3 illustrates the BFS spanning tree corresponding to the example network in Fig. 2. The tree is rooted at switch 0, i.e., the switch with the smallest ID. There are three levels in the spanning tree. All links in the up direction are shown by solid arrows. For each up link, there is a down link in the opposite direction sharing the same bidirectional link. For the purpose of clarity, they are not shown in the figure. Furthermore, the hollow arrows represent the down/up turns which are prohibited in the up*/down* routing. It can be also seen that there are a total of 18 prohibited turns when applying the above link direction assignment rule to the example topology by prohibiting the down/up turns.

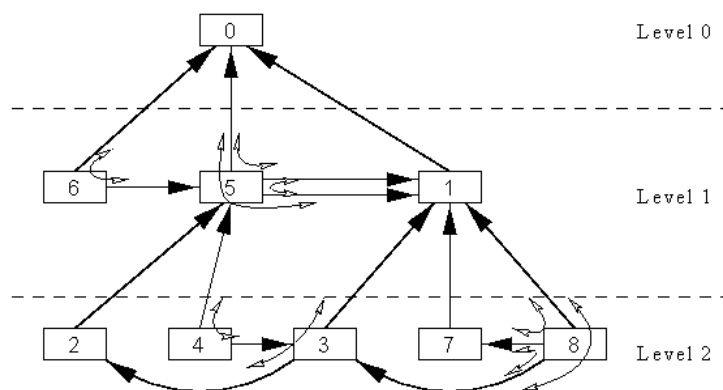


Fig. 3. The BFS spanning tree used for routing in DEC Autonet.

3. DESIGNING TURN-PROHIBITED ROUTING METHODS

3.1 Outline of the Design Methodology

The Autonet's up*/down* routing is a typical example of turn-prohibited routing methods, in which the network is configured as a tree and deadlock-freedom is achieved by prohibiting certain turns in the network. In general designing such a turn-prohibited routing algorithm for irregular networks encompasses two major components: (1) assigning link directions, and (2) determining a link-direction-based routing guideline.

Link direction assignment can be done in three steps: *root selection*, *spanning tree construction* and *cross edge direction assignment*. Root selection is performed based on the *root selection rule*, denoted **R1**. In the case of Autonet, the rule is to choose the switch with the smallest ID. When a root is selected, a BFS spanning tree rooted at the root switch can be obtained. Links across levels in the spanning tree can be assigned up and down directions accordingly.

For cross edges, the cross edge direction assignment rule, denoted **R2**, is used to assign directions. In Autonet, a cross edge is in the down direction if it links a switch with a smaller ID to another with a larger ID. It is in the up direction if it links a switch with a larger ID to another with a smaller ID. Such a rule will be referred to as the **Small ID Parent rule**. The link-direction-based routing guideline dictates how a message is routed through the links in the network based on the directions of the links. In other words, it defines a set of turn prohibitions. For example, the up*/down* routing in Autonet requires that a message be transmitted through zero or more up links, followed by zero or more down links (not to take any down/up turn).

3.2 Link-Direction-Based Routing Guideline with Extended Link Directions

In link-direction-based routing guidelines, messages are routed according to the turn prohibitions. The more the turn prohibitions are, the lower the adaptivity is. In Autonet, there are only two types of turns and one type (down/up) is totally prohibited. To relax the prohibition for more efficient routing methods, we propose to classify the directions of links into four types: *Up*, *Down*, *Lx_Up*, and *Lx_Down*. As a result, the types of turns increase to 12, as shown in Fig. 4.

The method to differentiate links into these four directions is similar to that for the Autonet's. After a root is selected and the BFS spanning tree is built, links across levels will be assigned as in Autonet. However, cross edges are assigned to *Lx_Up* directions if they connect younger siblings to elder siblings, and to *Lx_Down* directions if they connect elder siblings to younger siblings. The definition of younger siblings and elder siblings could change, depending on the *cross edge direction assignment rule*. A simple cross edge direction assignment rule may define younger siblings as switches with a smaller ID, and elder siblings as switches with larger ID. A more detailed discussion of the link direction assignment with four directions will be presented.

Minimizing of prohibited turn types to avoid deadlock

In designing turn-prohibited routing algorithms, we must find a set of prohibited

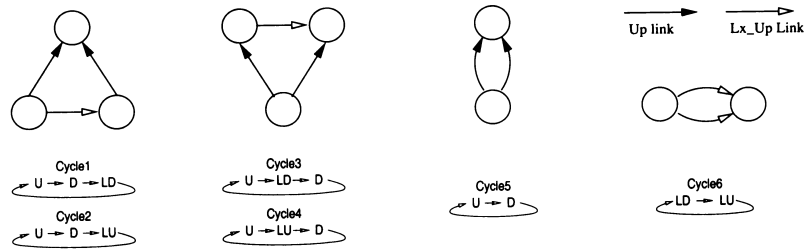


Fig. 4. Different types of turns in irregular networks.

turns so as to make the routing method deadlock free while keeping all switches reachable. To avoid deadlock, we should prohibit at least one turn for each cycle. Fig. 5 shows six different *basic cycles*. Each basic cycle contains a set of turns of particular types. The turns in any other cycle are an union or extension of turns contained in the basic cycles. Therefore, a deadlock free routing method needs to prohibit certain turns so as to break all the basic cycles.

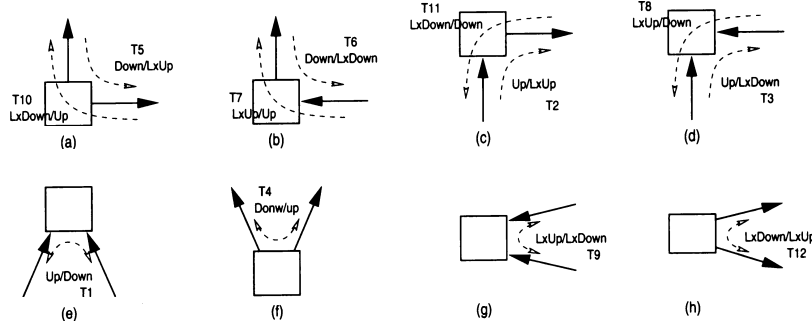


Fig. 5. Possible cycles in irregular networks.

To maintain switch reachability, the Up/Down type of turns should not be prohibited. This is because when there is no cross edge between a pair of switches, the Up and then Down route will be the only route to transmit messages between them. Thus, forbidding Up/Down turns may make the network unreachable. Fig. 6 shows a simple example. If the Up/Down turns are forbidden then there will be no route from switch 1 to switch 2, and vice versa.

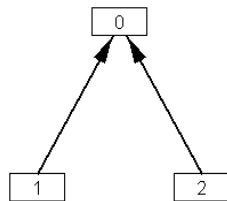


Fig. 6. A network will become unreachable if Up/Down turns are forbidden.

A good routing method will prohibit as few turns as possible. Therefore, the problem is to find the minimum number of types of turns which break all basic cycles while not making the network unreachable. This problem can be modeled as *finding the minimum cover set* to cover the types of turns contained in the basic cycles. The corresponding turn sets for the basic cycles are shown in Fig. 7. We find that there are eight alternative minimum cover sets which cover the turn sets of all basic cycles and do not include the Up/Down (T1) turns which make the network unreachable. These eight minimum cover sets are listed in Fig. 7 and also summarized in Table 1, each prohibits a different set of turns. They will be good candidates which produce minimum turn prohibitions. For each method, a \otimes mark in the table indicates that the corresponding type of turns is prohibited.

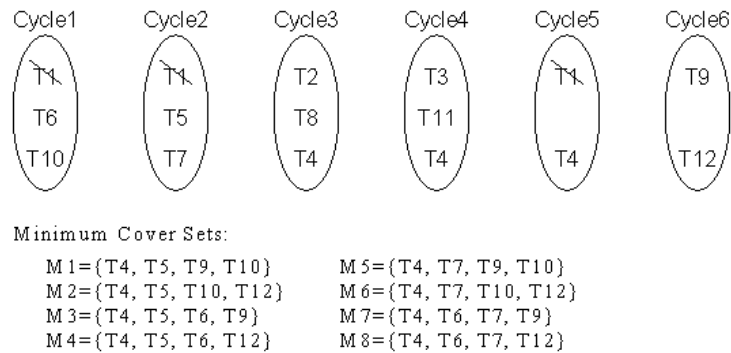


Fig. 7. Six corresponding turn sets for basic cycles.

3.3 Link Direction Assignment

The link-direction-based routing guideline tries to minimize the prohibited turn types while avoiding deadlock. Thus, it will affect the network performance directly. On the other hand, well-assigned link directions may reduce the number of prohibited turns. That is, the link direction assignment will influence the network performance indirectly.

Some observations

From Table 1, we can see that all eight methods prohibit four types of turns: they are the minimum requirement to break the basic cycles. If we can reduce the turns in the prohibited types, then the total number of turn prohibitions can be reduced. In the following, we summarize some observations on attempting to minimize the turns in different prohibited types (\otimes_1 to \otimes_7).

Observation 1 Since the total number of turns in a network is constant, moving the nodes with a higher degree closer to the root will increase the number of *non-prohibited* Up/Down turns, and have a higher possibility of decreasing Down/Up turns. The observation is obtained from symbol \otimes_1 .

Table 1. Routing methods and the set of turns they prohibit.

Different Turn-Types	Methods							
	M1	M2	M3	M4	M5	M6	M7	M8
T1: Up/Down								
T2: Up/LxUp								
T3: Up/LxDown								
T4: Down/Up	\otimes_1	\otimes_1	\otimes_1	\otimes_1	\otimes_1	\otimes_1	\otimes_1	\otimes_1
T5: Down/LxUp	\otimes_2	\otimes_2	\otimes_2	\otimes_2				
T6: Down/LxDown			\otimes_3	\otimes_3			\otimes_3	\otimes_3
T7: LxUp/Up					\otimes_4	\otimes_4	\otimes_4	\otimes_4
T8: LxUp/Down								
T9: LxUp/LxDown	\otimes_5		\otimes_5		\otimes_5		\otimes_5	
T10: LxDown/Up	\otimes_6	\otimes_6			\otimes_6	\otimes_6		
T11: LxDown/Down								
T12: LxDown/LxUp		\otimes_7		\otimes_7		\otimes_7		\otimes_7

After a root has been chosen, the BFS spanning tree can easily be established and the directions of links connecting switches of different levels can be decided. Next we need to decide the directions of cross edges. Again we can make several observations. In the following discussion, the switches of the same level will be called the *siblings*. As mentioned before, an *elder sibling* is defined by the source (destination) of an LxDown (LxUp) link, and a *younger sibling* is defined by the destination (source) of and LxDown (LxUp) link.

Observation 2 In a switch, a Down/LxUp turn comes from one of its parent switches and goes to an elder sibling. Similarly, a LxDown/Up turn comes from an elder sibling and goes to a parent switch. Thus, the number of either Down/LxUp or LxDown/Up turns is equal to the number of parent switches multiplied by the number of elder siblings. The number of parent switches is determined after the BFS tree is established. Therefore, for each cross edge, selecting the end switch with more parent switches as the elder sibling will decrease the number of Down/LxUp and LxDown/Up turns. The observation is obtained from symbols \otimes_2 and \otimes_6 .

Observation 3 In a switch, a Down/LxDown turn comes from one of its parent switches and goes to a younger sibling. Further, a LxUp/Up turn comes from one of its younger siblings and goes to a parent switch. Thus, the number of either Down/LxDown or LxUp/Up turns is equal to the number of parent switches multiplied by the number of younger siblings. The number of parent switches is determined after the BFS tree is established. Therefore, selecting the end switch with more parent switches as the younger sibling will decrease the number of Down/LxDown and LxUp/Up turns. The observation is obtained from symbols \otimes_3 and \otimes_4 .

Observation 4 In a switch, a LxUp/LxDown turn comes from one of its younger siblings and goes to a younger sibling. Thus, the number of LxUp/LxDown turns is equal to $N_y \times (N_y - 1)$, where N_y is the number of younger siblings. Reducing the number of

LxUp/LxDown turns is the same as reducing the number of younger siblings N_y . For each cross edge, selecting the end switch with more younger siblings as the younger sibling will reduce the number of younger siblings of a switch. The observation is obtained from symbol \otimes_5 .

Observation 5 In a switch, a LxDown/LxUp turn comes from one of its elder siblings and goes to an elder sibling. Thus, the number of LxDown/LxUp turns is equal to $N_e \times (N_e - 1)$, where N_e is the number of elder siblings. Reducing the number of LxDown/LxUp turns is the same as reducing the number of elder siblings N_e . For each cross edge, selecting the end switch with more elder siblings as the elder sibling will reduce the number of elder siblings of a switch. The observation is obtained from symbol \otimes_7 .

Now, we can summarize some useful rules for assigning the link directions from the above observations.

- Rule 1** Choose the root as the highest degree switch, and if there is more than one such switch, then choose the switch whose immediate neighbors have a larger total degree. (This rule is derived from Observation 1.)
- Rule 2** When M1 or M2 routing is used, the end switch of a cross edge with more parents should be set as the elder. (This is obtained from Observation 2.)
- Rule 3** When M7 or M8 routing is used, the end switch of a cross edge with more parents should be set as the younger. (This is obtained from Observation 3.)

Note, for routing methods M3, M4, M5, and M6, Observation 2 contradicts Observation 3.

- Rule 4** When M1, M3, M5, or M7 routing is used, the end switch of a cross edge with more younger siblings should be set as the younger. (This is derived from Observation 4.)
- Rule 5** When the M2, M4, M6, or M8 routing method is used, the end switch of a cross edge with more elder siblings should be set as the elder. (This is derived from Observation 5.)

Discussion

From Table 1, we can predict that the routing methods M1, M2, M7, and M8 will have a chance to achieve better performance than others, since each of them adopts three of the above rules to reduce the prohibited turns. On the other hand, the other methods (M3, M4, M5, and M6) only use two of the rules and lead to a contradiction between observations 2 and 3. Let M be any of the eight routing methods, and the function $TR(M)$ returns the number of prohibited turns of method M . Also let the function $Turn(T)$ to return the number of turns for type T . Therefore, we have the following equations:

$$\begin{aligned} TR(M1) &= Turn(T4) + Turn(T5) + Turn(T9) + Turn(T10) \\ TR(M2) &= Turn(T4) + Turn(T5) + Turn(T10) + Turn(T12) \end{aligned}$$

$$\begin{aligned}
TR(M3) &= Turn(T4) + Turn(T5) + Turn(T6) + Turn(T9) \\
TR(M4) &= Turn(T4) + Turn(T5) + Turn(T6) + Turn(T12) \\
TR(M5) &= Turn(T4) + Turn(T7) + Turn(T9) + Turn(T10) \\
TR(M6) &= Turn(T4) + Turn(T7) + Turn(T10) + Turn(T12) \\
TR(M7) &= Turn(T4) + Turn(T6) + Turn(T7) + Turn(T9) \\
TR(M8) &= Turn(T4) + Turn(T6) + Turn(T7) + Turn(T12)
\end{aligned}$$

From Observation 2, we know that the number of Down/LxUp (T5) turns is equal to the number of LxDown/Up (T10) turns. Moreover, from Observation 3, we know that the number of Down/LxDown (T6) turns is equal to the number of LxUp/Up (T7) turns. Therefore, we find that the number of prohibited turns of M3 and M5 are equal, so are M4 and M6.

Link direction assignment rules — some heuristics

The root selection rule R1 and the cross edge direction assignment rule R2 are the two major components in the link direction assignment. Let TD denote the total number of the immediate neighbors of a switch. To select the root, there are several possible heuristics for R1.

- (1) select the switch with the smallest UID, which is an unique identifier for each switch,
- (2) select the switch with the smallest degree, and with the smallest TD if there is more than one switch having the smallest degree,
- (3) select the switch with the smallest degree, and with the largest TD if there is more than one switch having the smallest degree,
- (4) select the switch with the largest degree, and with the smallest TD if there is more than one switch having the largest degree,
- (5) select the switch with the largest degree, and with the largest TD if there is more than one switch having the largest degree.

For cross edge direction assignment, there are also several heuristics for R2. We may set the switch to either (1) a smaller UID, (2) a higher degree, or (3) a smaller degree as the *elder sibling* of the other. If we randomly pick a pair of R1 and R2, then a new link direction assignment policy is formed. We will call the set of link directions assigned by R1 and R2 a *configuration* of the network. In the following, we list different configurations obtained by using different root selection and cross edge direction assignment rules.

- (C1) **SUIDR**: Small UID as Root and Smaller UID as elder
- (C2) **SD_lR-H**: Smallest Degree and Larger TD as Root and Higher degree as elder
- (C3) **SD_sR-H**: Smallest Degree and Smaller TD as Root and Higher degree as elder
- (C4) **SD_lR-H**: Smallest Degree and Larger TD as Root and Lower degree as elder
- (C5) **SD_sR-H**: Smallest Degree and Smaller TD as Root and Lower degree as elder
- (C6) **LD_lR-H**: Largest Degree and Larger TD as Root and Higher degree as elder
- (C7) **LD_sR-H**: Largest Degree and Smaller TD as Root and Higher degree as elder
- (C8) **LD_lR-H**: Largest Degree and Larger TD as Root and Lower degree as elder
- (C9) **LD_sR-H**: Largest Degree and Smaller TD as Root and Lower degree as elder

Note when the M2 routing method is applied to the SUIDR configuration, the resulting set of prohibited turns is the same as that of the up*/down* routing in Autonet. This is because when the four link directions approach is reduced to two, the LxUp direction links are re-assigned as Up direction links, and the LxDown direction links are re-assigned as Down direction links. Therefore, the prohibited Down/Up, Down/LxUp, LxDown/Up, and LxDown/LxUp turns in M2 become the Down/Up turns, which are the prohibited turns of Autonet's up*/down* routing.

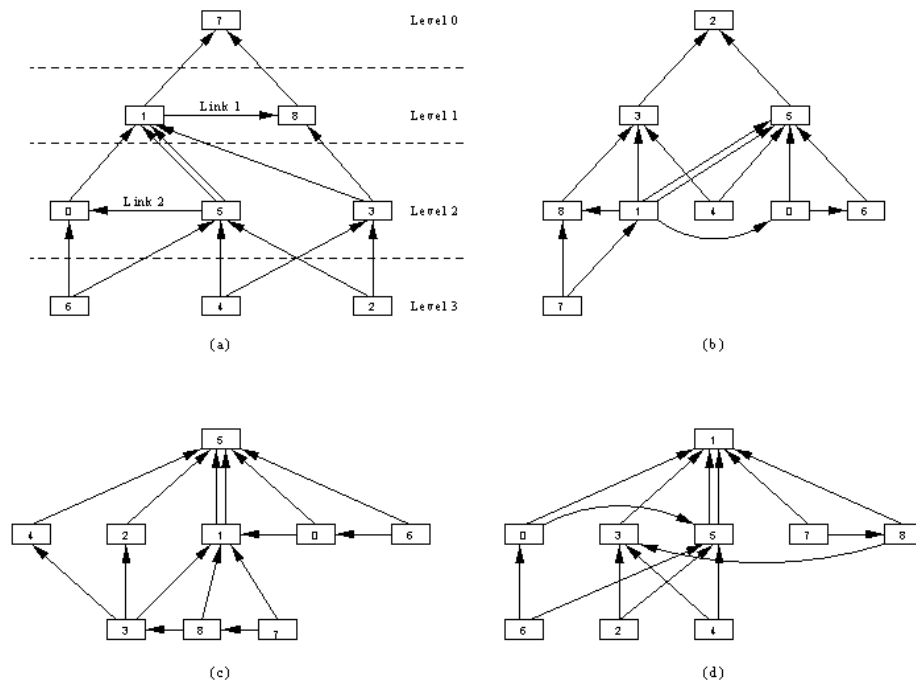


Fig. 8. The configurations produced by using different root selection and cross edge direction assignment rules, (a) SD_sR-L (root = 7), (b) SD_lR-L (root = 2), (c) LD_sR-H (root = 5), and (d) LD_lR-H (root = 1).

Fig. 8 shows the results of four of the above configuration algorithms for the example network. The results achieved by using SD_sR-H , SD_lR-H , LD_sR-H , and LD_lR-H configuration algorithms are shown in part (a), (b), (c), and (d) of Fig. 8, respectively. We can see in Fig. 8(a) that when the SD_sR-L configuration algorithm is used, the switch with smallest degree and smaller TD is chosen as switch 7. The BFS spanning tree is constructed with switch 7 at level 0, switches 1 and 8 at level 1, switches 0, 5, and 3 at level 2, and switches 6, 4, and 2 at level 3. Then the directions of a link between different level are assigned as Up directions if it is directed from a higher level to lower one, and as Down directions if it is directed from a lower level to a higher one. Finally, by applying *lower degree switch as the elder* rule, the cross edge link 1 will be assigned as LxUp direction from switch 1 (higher degree) to switch 8 (lower degree) and so will the cross edge link 2 from switch 5 (higher degree) to switch 0 (lower degree). Again, for

clarity, the Down and LxDown direction links, which share the same bidirectional links of the Up and LxUp direction links, are not shown in the figure. Other configurations shown in Fig. 8(b), (c), and (d) are established in a similar way

4. PERFORMANCE STUDY

In this section, we study the effectiveness of the routing algorithms that have been developed following our methodology for reducing turn prohibitions. A reference for comparison is the scheme proposed in [2], which used graph partitioning to find routing methods with the minimum number of prohibited turns. Pseudo-code for the scheme, referred to as the *partition method*, is shown in Fig. 9.

```

FIND_RESTRICTIONS  $G'(V', E')$ 
Input: graph representation of the network
Output: a set of turn restrictions
1  CYCLE_DETECT ( $G'$ )
   if no cycles in  $G'$  then
       return empty set.
   endif
2  Partition  $V'$  into two sets of vertices  $A$  and  $B$  such that
    $A$  and  $B$  have nearly equal number of vertices
   AND the number of edges from  $A$  to  $B$  is minimal
   AND partition constraints are satisfied.
3  Remove the set of edges  $TR$  from  $A$  to  $B$ 
4   $TR \leftarrow TR + \text{FIND\_RESTRICTIONS}(G_A)$ 
5   $TR \leftarrow TR + \text{FIND\_RESTRICTIONS}(G_B)$ 
6  return  $TR$ 

```

Fig. 9. The scheme proposed in [2] for finding turn prohibitions.

In the partitioning method, a directed graph representation $G'(V', E')$ of networks is used, where vertices represent the *network links*, and directed edges represent every combination of input-output pairs within the switch. To find the turn prohibitions, the set V' is first partitioned into subsets A and B using a nearly equal number of vertices. Then, vertices in A and B are swapped to minimize the number of cross edges from A to B . The resultant set of cross edges is a subset of the turn prohibitions. This procedure is applied to sets A and B recursively. Finally, the union of the sets of cross edges found forms the final set of turn prohibitions.

4.1 Complexity Analysis

The complexity of the partitioning method is determined as follows. Assume that in the initial graph $|V| = N$ and $|E| = M$. The directed graph representation $G'(V', E')$ will have $|V'| = 2 \times M$ and $|E'| \geq 2 \times (M - 1)$, where the low bound of $|E'|$ occurs when there is no cycle in the initial graph $G(V, E)$. Step 1 in Fig. 9 detects whether the given graph has any cycles. This step can be accomplished in $O(M^2)$ time. In Step 2, a two-way uniform partitioning algorithm is used to partition the vertex set of G' into two subsets A

and B , which can be done in $O(M^2 + M^{3/2} \times 4^M)$ time [3]. Step 3 removes the set of edges from A to B , and the recursion continues to the two subgraphs A and B . Since the procedure is run on the set V' recursively, in the worst case the complexity of the whole partitioning algorithm will be $O(\log_2 M \times (M^2 + M^{3/2} \times 4^M))$.

In our approach, a link direction assignment algorithm is first run on the initial graph $G(V, E)$. The complexity of the algorithm is $O(N \times M)$, since root selection requires $O(N)$ time, spanning tree construction takes $O(N + M)$ time, direction assignment for links across different levels requires $O(N \times M)$ time, and direction assignment for links across the same level takes $O(N \times M)$ time. After direction assignment, the complexity for calculating the turn prohibitions is $O(N)$. In all, the complexity using our scheme is $O(N \times M)$, which when compared to the complexity of the partitioning method, has a much lower complexity.

4.2 Comparison Based on the Example Topology

In this subsection, we compare different routing methods in terms of the number of turn prohibitions they produced for the example network in Fig. 2. The results are shown in Table 2. For the partitioning method, 100 different initial partition sets A and B are randomly chosen. Each initial partition set will result in a different set of turn prohibitions. In these 100 runs, the minimum, maximum, and average number of turn prohibitions are listed. On the example network, the partitioning method achieved an average of 17.43 turn prohibitions, with a maximum of 23 and a minimum of 14.

On the other hand, for the turn-prohibited routing algorithm, a set of turn prohibitions can be achieved after the link directions are assigned and the link-direction-based routing guideline is chosen. Each of the combinations of the nine direction assignment methods and eight different link-direction-based routing guidelines are compared. A minimum of 14 turn prohibitions was achieved when the M1 or M2 routing guideline was used with a LD_LR-H (C6) configuration, or M7 routing guideline was used with a LD_LR-L (C8) configuration. This case with 14 turn prohibitions is as good as the minimum value achieved by the partitioning method in 100 runs. More importantly there is no need to perform hundreds of runs in order to get the best result. Note also that this is much better than the 18-turn-prohibition obtained by the M2 routing guideline with a SUIDR configuration (C1). *The combination of M2 and C1 is just the up*/down* routing used by DEC Autonet.* From this result, we conclude that one can develop a very efficient routing method on irregular networks by proper selection of the root switch, better assignment of the link directions, and using a matched routing guideline.

When different root selection policies are compared under the same cross edge assignment rule, we can see that the root selected in the LD_LR-H and LD_LR-L configurations is always better than the others. To explain why, recall that in applying LD_LR-H or LD_LR-L rules the largest node is chosen as root. And, if there is more than one node having the largest degree, then we choose the one that has the largest sum of the degrees of the immediate neighbors, denoted TD . This is just the conclusion made in Rule 1, which is derived from Observation 1. It has the effect of reducing the number of Down/Up turns which are prohibited by all routing guidelines. Therefore, the LD_LR-H and LD_LR-L rules will have fewer turn prohibitions in all eight routing guidelines.

Table 2. The number of turn prohibitions on the example network.

Method	# of Turn Restrictions				
	Max.	Min.	Avg.		
Partition	23	14	17.43		
	SUIDR	SD_LR-H	SD_SR-H	LD_LR-H	LD_SR-H
	root = 0	root = 2	root = 6	root = 1	root = 5
M1	18	18	18	14	16
M2	18	16	18	14	16
M3	18	22	20	15	19
M4	18	20	20	15	19
M5	18	22	20	15	19
M6	18	20	20	15	19
M7	18	26	22	16	22
M8	18	24	22	16	22
		SD_LR-H	SD_SR-H	LD_LR-H	LD_SR-H
		root = 2	root = 6	root = 1	root = 5
M1	None	24	22	16	20
M2		26	22	18	22
M3		20	20	15	19
M4		22	20	17	21
M5		20	20	15	19
M6		22	20	17	21
M7		16	18	14	18
M8		16	18	16	20

Next, let us consider the effects of different routing guidelines. For each of the configurations which assign the cross edge direction in such a way that nodes with a higher degree are designated as the elder sibling, better results are achieved using the M1 and M2 methods. This is because doing so will reduce the Down/LxUp (T5) and LxDown/Up (T10) types of turns which are prohibited in M1 and M2, and thus will reduce the turn prohibitions in M1 and M2. However, when the cross edge direction is assigned to point to nodes with a lower degree, better results are obtained with M7 and M8. The reason is similar, since choosing the lower degree end as elder sibling of cross edges will reduce the Down/LxDown (T6) and LxUp/Up (T7) types of turns which are prohibited in M7 and M8. Therefore, the turn prohibitions in M7 and M8 can be reduced.

4.3 Comparison Based on Randomly Generated Networks

In the previous subsection, we compared different configurations and routing guidelines on the example topology. In this subsection, for a more detailed analysis, we compare different routing guidelines, root selection policies, and cross edge direction assignment rules based on randomly generated irregular networks with 8, 16, and 32 switches. Each switch is assumed to have 10 ports. For each size of the network, one

hundred different connections are randomly generated. In the experiments we make the following assumptions. First, the generated network must be connected so that messages can be transmitted between nodes. Second, each switch uses two to six ports to connect to other switches. Note that at least two ports are needed to make the network connected, and some idle ports (at least four ports in our assumption) are needed to connect the processing nodes to generate communication traffic. Third, duplicated links between two switches are allowed. In this way the switches can accommodate heavier traffic between them.

Effects of different cross edge direction assignment rules

Here we use the cross edge direction assignment rule as a variable and study the effects on the resulting turn prohibitions. Fig. 10(a) shows the turn prohibitions of different configurations, each averaging over eight routing guidelines for 8-switch networks. Let us consider the configurations C2 (SD_LR-H) and C4 (SD_LR-L), where the difference between them is the cross edge direction assignment rule. We can find that C2, which uses the *higher degree elder* rule, has fewer turn prohibitions than C4, which uses the *lower degree elder* rule. Similar results can also be found if we compare configuration pairs (C3, C5), (C6, C8), and (C7, C9). In these pairs, the former configuration adopts the higher degree elder rule, while the latter uses the lower degree elder rule. As we know, fewer turn prohibitions means higher adaptively and better performance of a routing method. Therefore, we can conclude that, on average, the higher degree elder cross edge direction assignment rule is the better choice.

Effects of different root selection rules

Here we use the root selection rule as a variable and study the effects on the resulting turn prohibitions. When considering different root selection policies, configurations choosing the node with the largest degree as the root are better than those choosing the node with the smallest degree. For example, consider Fig. 10 again. The configuration C6 (LD_LR-H) has fewer turn prohibitions than C2 (SD_LR-H). Similar behaviors

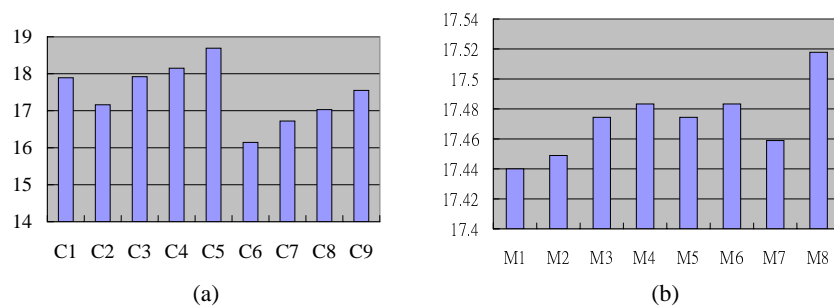


Fig. 10. The number of turn prohibitions using different combinations of (a) nine configurations, each averaging different routing methods, and (b) eight routing guideline and assignment policies on 8-switch networks.

can also be found in configuration pairs (C7, C3), (C8, C4), and (C9, C5), where the only difference between configurations in each pair is that the former choose the largest degree node as root but the latter choose the smallest one.

Furthermore, if there is more than one node with the largest degree, then Fig. 10(a) shows that the configurations using the largest TD , i.e., choosing the node whose immediate neighbors have the largest total degree, are better. Such configurations include C2 ($SD_L R-H$), C4 ($SD_L R-L$), C6 ($LD_L R-H$), and C8 ($LD_L R-L$). From Fig. 10(a), we can see that they have fewer turn prohibitions than their corresponding pair configurations, i.e. C3 ($SD_S R-H$), C5 ($SD_S R-L$), C7 ($LD_S R-H$), and C9 ($LD_S R-L$) respectively. From the figure we can also see that on average the $LD_L R-H$ configuration has the minimum number of turn prohibitions of all routing guidelines. This is because C6 ($LD_L R-H$) satisfies more criteria, as mentioned in Section 3.3, than any other configuration.

Effects of different link-direction-based routing guidelines

In Fig. 10(b), the turn prohibitions of eight routing guidelines averaging over nine configurations are shown. They produce a set of very close values, where the largest average is 17.518 achieved with M8 and the smallest is 17.44 achieved with M1. Therefore, we cannot say that any one particular guideline is superior.

In Fig. 11, the number of turn prohibitions is plotted for each combination of routing guideline and configuration. We can see that within a configuration, there is a wide variety in the number of turn prohibitions for the eight routing guidelines. The best result is 14.9, which occurs for the M2 routing guideline and C6 ($LD_L R-H$) configuration. The average number of turn prohibitions of the generated networks using DEC Autonet is 17.4. By applying the graph partitioning method to the generated networks, each with 100 different initial partition sets, we can obtain the average turn prohibitions to be 17.27, with the minimum being 14.45 and the maximum being 21.87. We can find, again, the combination (M2, C6) gets a very good result, less than Autonet's approach and just a little larger than the minimum value for the graph partitioning method.

If we consider the configurations which use the *higher degree elder* rule (C2, C3, C6, and C7), we find that the number of turn prohibitions produced by different routing guidelines have the following relations: (1) M2 produces the fewest number of turn prohibitions, and (2) $M1 > M2$, $M3 > M4$, $M5 > M6$, and $M7 > M8$. The reason is that when the higher degree elder rule is applied, the resulting configuration may satisfy the statements in Rules 2 and 4. On the other hand, if we consider configurations which use the *lower degree elder* rule (C4, C5, C8, and C9), we find that the number of turn prohibitions produced by different routing guidelines have the following relations: (1) M7 produces the fewest number of turn prohibitions, and (2) $M1 < M2$, $M3 < M4$, $M5 < M6$, and $M7 < M8$. The reason is that when the lower degree elder rule is applied, the resulted configuration may satisfy the statements in Rules 3 and 5.

For networks with 16 and 32 switches, the behavior is quite similar to that of 8-switch networks. The number of turn prohibitions can be found in Figs. 12 and 13. The only difference is that the values obtained with the 16-switch networks are nearly twice that of those with the 8-switch networks. Similarly, those with the 32-switch networks are about four times that of those with the 8-switch networks.

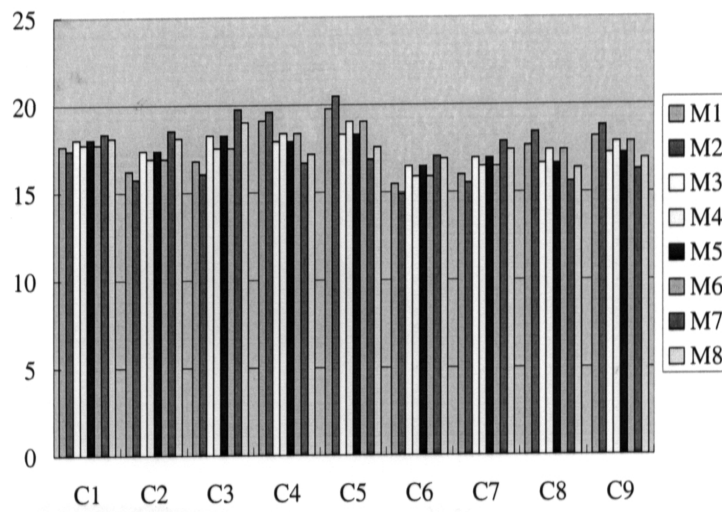


Fig. 11. The number of turn prohibitions using different combinations of routing guideline and assignments policies on 8-switch networks.

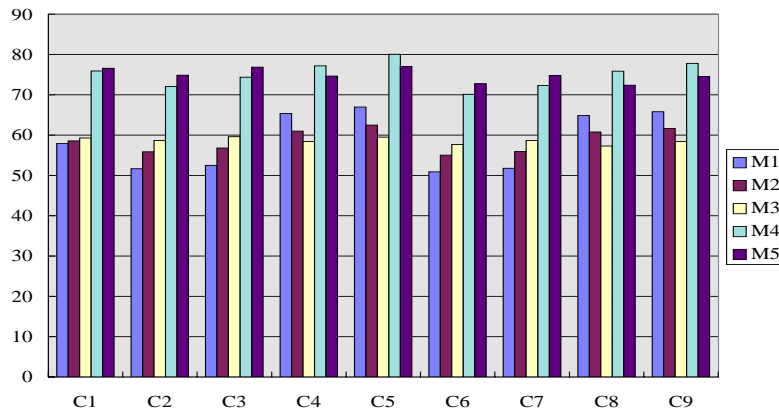
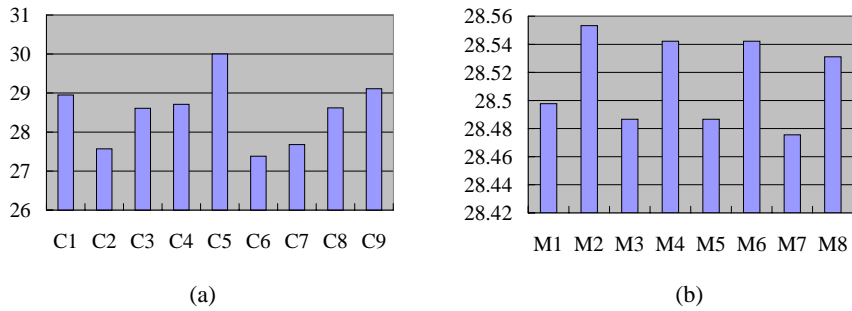


Fig. 12. The number of turn prohibitions using different combinations of routing guideline and assignments policies on 16-switch networks.

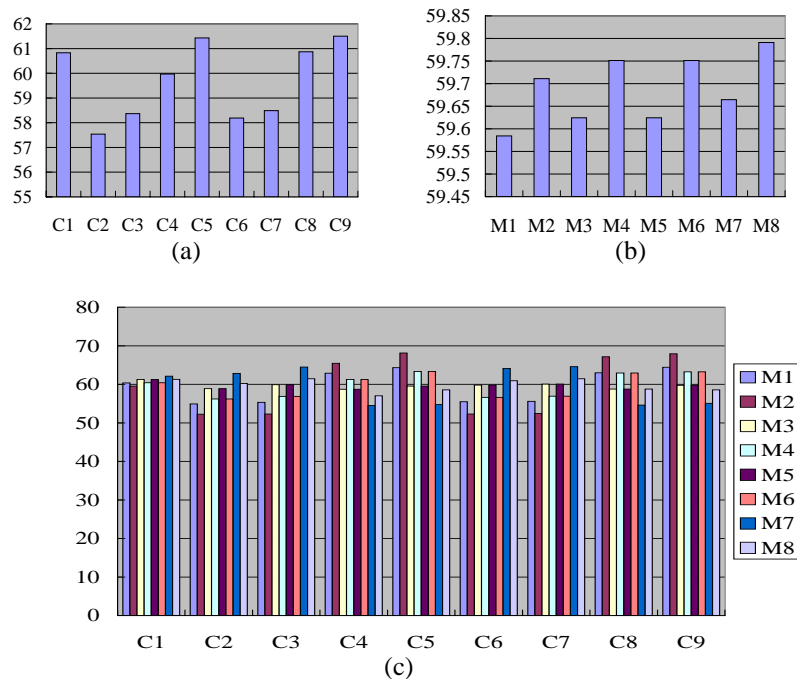


Fig. 13. The number of turn prohibitions using different combinations of routing guideline and assignment policies on 32-switch networks.

5. CONCLUSIONS

With the popularity of irregular networks in parallel processing, efficient routing methods for enhancing message transmission while avoiding deadlocks are becoming important. Turn-prohibited routing algorithms are a primary strategy to achieving such goals. In this paper, we show that turn prohibitions can be reduced by properly assigning link directions. When an example network was considered, our proposed methods — (M1, C6), (M2, C6), and (M7, C8) — achieved the minimum number of 14 turn prohibitions. In addition, there is no need to perform hundreds of runs in order to obtain this result, as was necessary when using the partition method. We have also shown that the complexity of the partition method is much higher than the $O(N \times M)$ complexity using our approach.

We have experimented with randomly generated networks having eight switches. Among the different direction assignment policies, the configuration C6 (LD_LR-H) imposes the fewest turn prohibitions. When comparing the routing guidelines, the average turn prohibitions produced by the eight routing guidelines are very similar. If we look at different configurations individually, we can find many different behaviors. The root selection rule, the rule to break a tie on root selection, and the rule to assign the directions (Lx_Up or Lx_Down) for cross edges will all affect the result in different ways. Overall, if we assign the link directions by C6 (LD_LR-H) and prohibit the types of turns defined in M2 ($Up*/Down*$), then we get the fewest turn prohibitions among all

link-direction-based approaches. This result is better than Autonet's approach and very close to that of the graph partition scheme. From the discussions in this paper, we have shown that good root selection and link direction assignment policies can produce efficient routing algorithms, even with a simple routing guideline such as Up*/Down*.

REFERENCES

1. Ancor Communication Inc. <http://www.ancor.com/cxt.html>.
2. B. Abali, "A deadlock avoidance method for computer networks," in *Workshop on Communication and Architectural Support for Network-based Parallel Computing*, 1997.
3. B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partition graphs," *Bell System Technical Journal*, Vol. 49, No.2, 1970, pp. 291-307.
4. F. Silla, M. P. Malumbres, A. Robles, P. Lopez, and J. Duato. "Efficient adaptive routing in networks of workstations with irregular topology," in *Workshop on Communication and Architectural Support for Network-based Parallel Computing*, 1997.
5. L. M. Ni and P. K. Mckinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, 1993, pp. 62-76.
6. M. D. Schroeder et al., "Autonet: a high-speed self-configuring local area network using point-to-point links," Technical report, Digital Equipment Corporation, 1990. SRC research report 59.
7. N. J. Boden et al., "Myrinet – a Gigabit-per-second local area network," *IEEE Micro*, Vol. 15, No. 1, 1995, pp. 29-36.
8. R. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proceedings of 10th International Parallel Processing Symposium (IPPS'96)*, 1996, pp. 274-280.
9. R. J. Souza et al., "The GIGAswitch system: a high-performance packet switching platform," *Digital Technical Journal*, Vol. 6, No. 1, 1994.
10. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computer*, Vol. C-36, No. 5, 1987, pp. 547-553.
11. W. Qiao and D. M. Ni, "Adaptive routing in irregular network using cut-through switches," in *Proceedings of the 1996 International Conference of Parallel Processing*, Vol. I, 1996, pp. 52-60.



Jenq-Shyan Yang (楊振賢) received the BS degree in Information Science from Tung Hai University, Taiwan, R.O.C., in 1990, and the M.S. and Ph.D. degrees in Computer Science from National Tsing Hua University, Taiwan, R.O.C., in 1992 and 1999 respectively. From 1997 to 1999, he served as an information officer in the Ministry of National Defense, Taiwan, R.O.C., and from 1999 to 2000, he was an assistant professor of Information Management Department at LungHwa Institute of Technology, Taiwan, R.O.C. He is currently a project researcher of Broadband BU at Acer Communications and Multimedia Inc. His research interests include computer architecture, distributed computing environments, and computer networks.



Chung-Ta King (金仲達) received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, R.O.C., in 1980, and the M.S. and Ph.D. degrees in computer science from Michigan State University, East Lansing, Michigan, in 1985 and 1988, respectively. From 1988 to 1990, he was an assistant professor of computer and information science at New Jersey Institute of Technology, New Jersey. In 1990 he joined the faculty of the Department of Computer Science, National Tsing Hua University, Taiwan, R.O.C., where he is currently a professor. His research interests include computer architecture, cluster systems, and web computing. He is a member of the IEEE.