A Deadlock-Free Routing Scheme for Interconnection Networks with Irregular Topologies

Hsin-Chou Chi and Chih-Tsung Tang
Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, Taiwan, R.O.C.

Abstract

Interconnection networks with irregular topologies (or irregular networks) are ideal communication subsystems for workstation clusters owing to their incremental scalability. While many deadlock-free routing schemes have been proposed for regular networks such as mesh, torus, and hypercube, they cannot be applied in irregular networks. This paper presents a cost-effective routing architecture, called TRAIN, to solve the routing problem with irregular networks. We show that TRAIN is a deadlock-free scheme. Furthermore, unlike many other routing schemes proposed previously for irregular networks, TRAIN does not require a routing table in the switch. Due to this feature, a TRAIN switch is small and the routing decision can be made rapidly. In order to evaluate the effectiveness of our routing scheme, analysis and event-driven simulation have been performed for various irregular networks. Our results show that TRAIN outperforms other schemes with a higher maximum throughput and lower average latency consistently.

1. Introduction

Multiprocessors and multicomputers achieve high performance using interconnection networks that provide high-bandwidth low-latency interprocessor communication. Point-to-point interconnection networks composed of communication switches have been employed in many experimental and commercial systems. Most of these networks use regular topologies, such as mesh, torus, and hypercube, to connect their switch components. With a regular topology, an interconnection network can be compactly implemented in a circuit board or chassis for massively parallel processors. Many routing algorithms have been developed for regular networks to provide efficient transmission of packets and guarantee deadlock freedom [1, 2, 4, 5, 6, 8].

Lately, workstation clusters have emerged to become a cost-effective approach to build a multiprocessing platform. Traditional local area networks can be used to support inter-node traffic for such parallel systems. However, their performance is not maximized due to the limited bandwidth. Although it is possible to use regular networks originally designed for massively parallel processors, they are not incrementally scalable and cost-effective for workstation clusters. In comparison, an interconnection network with irregular topologies matches the cluster environment better. Such irregular networks consist of many small switches and can scale up the communication performance by adding more switches.

Designing an efficient routing scheme for irregular networks is not trivial. A well designed routing scheme for irregular networks should be deadlock-free and provide high performance for various network topologies. Furthermore, the routing scheme should be efficiently implemented in a communication switch. Several routing architectures have been proposed for irregular networks in recent years [9, 10]. Although these schemes are capable of routing packets in various network topologies and achieve deadlock freedom, they typically rely on a large hardware routing table in the communication switch. With such routing schemes, the size of the routing table in the switch grows proportionally as the size of the network increases. Hence, the implementation of the communication switch becomes relatively complex. Furthermore, packets traversing the switch suffer from long forwarding latency.

We propose a cost-effective routing architecture without a routing table in the switch for irregular networks. The routing scheme is deadlock-free and is adaptive to dynamic network traffic conditions. Since there is no routing table in the switch, the routing decision can be made rapidly and thus provides low latency for packets traversing the network. *Virtual cut-through* switching is assumed in our interconnection network [7]. The routing architecture is based on a spanning tree which is a subset of the network, and hence is called TRAIN (Tree-based Routing Architecture for Irregular Networks). TRAIN uses the links of the spanning tree to form a route which can be followed by a packet from any source node to any destination node. The other links than those in the tree can be used as shortcuts to reduce the number of hops a packet has to traverse, which significantly improve the network performance.

The next section describes the problem of designing a routing system for irregular networks. Previous research work on this issue is surveyed. In Section 3, the TRAIN routing scheme for irregular networks is proposed. It is

0-8186-8227-2/97 $10.00 © 1997 IEEE

shown that the routing algorithm is deadlock-free and can be efficiently implemented in a switch. The performance of TRAIN is evaluated in Section 4. It is compared with other routing schemes based on analysis for unloaded networks and event-driven simulation for loaded networks. Section 5 concludes this work with a summary.

2. The Routing Problem for Irregular Networks

An irregular network consists of an arbitrary number of nodes and has an irregular topology. Figure 1 shows an example of an irregular network consisting of 9 nodes [9]. Each node includes a computing element and an associated communication switch in the network. We assume that a communication switch connects to only one computing element in the rest of the paper. In fact, it is possible to connect more than one computing element to a communication switch.

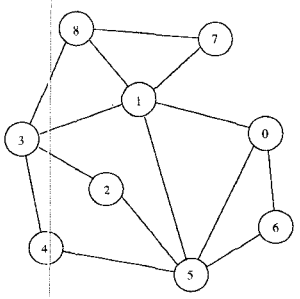

Figure 1: An irregular network.

Figure 2 shows a workstation cluster with an irregular network from Figure 1. A collection of communication switches constitute the interconnection network. Each node in the network has a computing element associated with a communication switch. Note that some ports of the switches are left open in this example. These free ports can be used to connect to other switches or more computing elements. The efficient design and implementation of the communication switch and its routing function have a significant impact on the performance of the interconnection network.

A generic architecture of the communication switch is shown in Figure 3 [11, 12]. Each port in the switch is associated with a pair of input and output channels. The link connected to a port in the switch leads to either a computing element or a communication switch. A communication switch often includes buffers at their input ports for storing incoming packets that cannot be forwarded immediately due to output port contention or blocking. The crossbar in the switch provides unblocking paths between the input ports and output ports. A hardware routing decision module determines which output port an arriving packet will be destined to based on the information in the packet header.

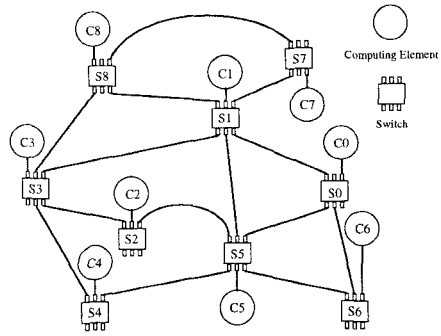

Figure 2: An interconnection network corresponding to the example topology in Figure 1. Each computing element is associated with a communication switch in a node.

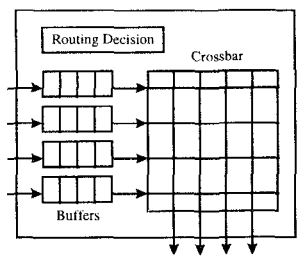

Figure 3: A generic architecture of a communication switch with input buffers.

Without a careful design for the routing scheme, *deadlock* may happen to an interconnection network. When deadlock occurs, a set of messages are blocked forever in the network [4, 5, 8]. Many deadlock-free routing algorithms have been proposed for regular networks such as mesh, torus, and hypercube. Most of these routing algorithms achieve deadlock freedom by avoiding the possibility of deadlock situations. Such *deadlock avoidance* is typically accomplished by removing *channel dependency cycles* for the network [4, 5]. Deadlock avoidance can be implemented in an algorithmic routing function in the communication switch. A well designed routing algorithm should achieve deadlock freedom without significantly reducing the network performance.

While many effective routing schemes have been proposed for regular networks, there have been very few counterparts for irregular networks. Although *deadlock detection and recovery* can be used to resolve the problem, the complexity of the communication switch based on that is significantly increased. Hence, most practical designs proposed recently for deadlock-free routing in irregular networks rely on deadlock avoidance.

The DEC Autonet project has proposed the up*/down* routing algorithm for their network [10]. Autonet is a self-configured, switch-based local area network. In an existing network, a spanning tree is

constructed by a distributed algorithm and each node is assigned an ID during initialization. Each link in the spanning tree is assigned a direction with "up" indicating "toward the root." A link leading to a parent node in the tree is in the "up" direction. A link leading to a node with a smaller ID at the same level is also in the "up" direction. The basic idea of the up*/down* routing is that every packet has to be routed for zero or more hops in the "up" direction, and then for zero or more hops in the "down" direction. A legal route never uses a link in the "up" direction after it has been in the "down" direction. Hence, channel dependency cycle is prohibited and deadlock freedom is achieved.

Note that in Autonet the links connecting nodes at different levels in the tree do not participate in routing packets. The bandwidth of these links is thus wasted. In Autonet, each packet follows the shortest one of all the legal paths, and this is implemented based on a routing table in each switch. Since each switch has an associated routing table, and searching the table is part of the routing function in the switch, the complexity of the switch is relatively high.

Adaptive trail routing is another scheme proposed for irregular networks [9]. The routing algorithm is based on two unidirectional adaptive trails constructed from two opposite unidirectional Eulerian trails. The links not in the two Eulerian trails can be used as shortcuts as long as they do not cause deadlock. The Eulerian trails are determined based on some heuristics during initialization. However, not every network topology has such Eulerian trails. Furthermore, like Autonet routing, the switch for adaptive trail routing requires a routing table, which increases the switch complexity.

3. A Tree-based Routing Architecture for Irregular Networks

Designing a deadlock-free routing scheme for irregular networks is more complex than for regular networks. As discussed in Section 2, Autonet routing and adaptive trail routing have been proposed to solve the problem. Although these two schemes are deadlock-free, they require a routing table built in the switch. As the network grows, the size of the routing table increases significantly. In this section, we propose a tree-based routing scheme which requires no routing table in the switch. This routing scheme is called TRAIN (Tree-based Routing Architecture for Irregular Networks).

3.1. TRAIN Routing

TRAIN is a routing scheme similar to the Autonet routing in that it also uses a subset of the network to construct a tree. A packet sets out from the source node and follows the tree structure to reach any other node in the tree. With TRAIN, however, those links not in the tree can also be used for packet transmission. These links are called *shortcut links* or simply *shortcuts*. The basic idea of TRAIN is that a packet arriving in a switch takes advantage of the shortcuts if they provide a shorter route to the destination node than the "planned" route. A planned route is the one that follows the tree structure. These shortcuts help reduce the number of hops a packet has to traverse from the source node to the destination node. In addition, more bandwidth is provided in the network and the traffic congestion in the area near the root of the tree is mitigated.

When a packet arrives in a switch, the routing decision module of the switch checks if there is a shortcut from there leading to a switch "closer" to the destination node. If such a shortcut exists, a packet is routed to the neighboring switch the shortcut leads to. If such a shortcut does not exist or is currently blocked, the packet simply continues to follow the planned tree route. Note that a packet may utilize shortcuts more than once along the route from the source node to the destination node. A key to the effectiveness of the TRAIN scheme is that distance calculation between two nodes should be simple. Such a distance calculation function should be efficiently implemented in an algorithmic hardware and no routing table is needed in the switch. We have developed a distance calculation algorithm which can be used in TRAIN.

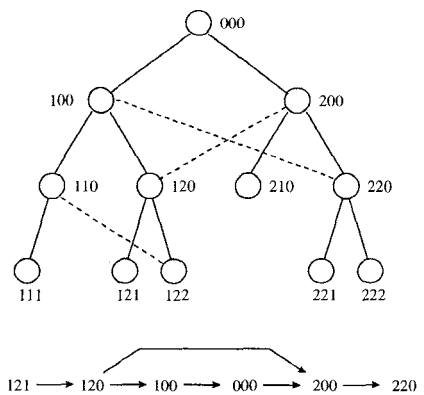

Figure 4: An example network with TRAIN. A packet arrives in node 120 takes the shortcut leading to node 200.

Figure 4 shows an example network with TRAIN. The number beside each node is the node ID used for distance calculation. The labeling method is as follows. The root node is labeled 00...0. The tree is not necessarily a binary tree. The children of the root are labeled 10...0, 20...0, and so on. From source node 121 to destination node 220, for instance, the planned tree route takes five hops. However, when the packet arrives in node 120, it

finds that the shortcut leading to node 200 provides a shorter route. Hence the eventual distance that the packet traverses from node 121 to node 220 is reduced from 5 to 3. Note that in order for the routing decision module to work, a switch is required to store the neighboring nodes' IDs the shortcuts lead to. This information can be set up during initialization.

3.2. Distance Calculation

Distance calculation between two nodes is critical to the performance of the TRAIN scheme. Our solution to this problem is shown in Figure 5, taking the example from Figure 4. The algorithm is described in the following:

Algorithm Distance_Calculation:

Step 1: From the two digit strings of two nodes' IDs, remove the common prefix string from the two node IDs.

Step 2: The distance between two nodes is exactly the total number of non-zero digits left in the two strings.

In the example in Figure 5, for the case on the left there is no common prefix string and hence the distance from node 100 to node 220 is 3 since there are three non-zero digits. The distance from node 200 to node 220 is 1 since the common prefix string "2" is deleted and only one non-zero digit is left.

100	~~2~~00
220	~~2~~20
3	1

Figure 5: Distance calculation example from Figure 4. For the case on the left, there are no common prefix digits and so the distance from node 100 to node 220 is 3 since there are three non-zero digits. The distance from node 200 to node 220 is 1 since the common prefix string "2" is deleted and only one non-zero digit is left.

The TRAIN algorithm can be described in the following algorithm:

Algorithm TRAIN_Routing:

Step 1: For an arriving packet, calculate the possible distance from the current node to the destination node based on the destination node ID of the packet and the stored IDs of the neighboring nodes.

Step 2: If any unblocked shortcut provides a shorter route than the tree route, pick the most profitable unblocked shortcut and route the packet to the neighboring node this shortcut leads to.

Step 3: If all the profitable shortcuts are blocked or there is no profitable shortcut from the current node, route the packet to the neighboring node along the tree route.

Step 4: If all the profitable shortcuts and the tree link are blocked in the current cycle, wait for the next cycle and start from Step 2 again.

In the above algorithm, profitable shortcuts are those shortcut links that can bring the packet closer to the destination.

As mentioned above, each switch can be associated with one or more computing elements. If there is only one computing element coupled with each switch, we can assign a unique node ID for both of them. If there are two or more computing elements connected to a switch, we can treat the computing elements and the switch as independent network nodes and give them different node IDs. The TRAIN scheme can be applied in the above two types of configuration. Figure 6 shows the TRAIN configuration of the interconnection network from the example in Figure 2. Note that six links are used as shortcuts.

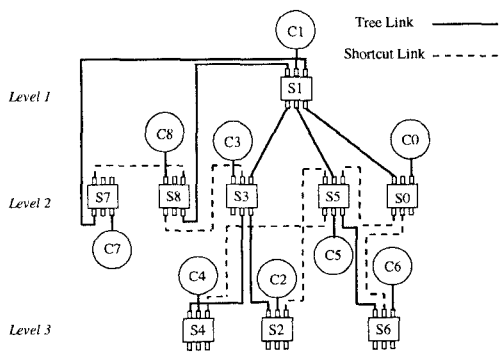

Figure 6: The TRAIN configuration of the interconnection network from the example in Figure 2. Six links are used as shortcuts.

3.3. Deadlock Freedom

The TRAIN scheme is deadlock-free. *Virtual cut-through switching* is used in TRAIN and it is part of the mechanism to prevent deadlock. Virtual cut-through is a switching technique similar to the popular *wormhole switching* in that a packet can start to forward to the next node if the packet header has arrived and the destined output port is decided. However, virtual cut-through switching requires the next switch to have sufficient buffer space to store the packet. When a packet is blocked at the

head, the rest of the packet will keep moving forward and stay in the switch where blocking occurs. Although virtual cut-through switching requires more buffer space than wormhole switching, it is feasible to implement a single-chip switch that can accommodate several whole packets with current VLSI technology. Furthermore, with virtual cut-through switching, no packets can be blocked in place across several nodes waiting for transmission. This helps reduce network contention.

The mechanism for deadlock prevention is simple in TRAIN. Figure 7 shows why deadlock cannot happen in TRAIN. Since there exists no cycle in any tree, at least a shortcut link is required to form a channel dependency cycle. Suppose such a cycle exists and the shortcut link is part of the dependency cycle in the network. At node B, packets coming from node A will never cross the shortcut and go to node D. The reason for this is that when a packet is "going down" (away from the root) in the tree, the destination node must belong to the offspring of the current node (node B in this case). Hence the packet will never go to another tree branch. On the other hand, at node B, packets coming from node C can choose to cross the shortcut or go up to node A, depending on their destination. In the TRAIN scheme, a shortcut leading to a shorter route has a higher priority than the tree link toward the root. However, when the shortcut is blocked, the packet follows the tree link adaptively. Furthermore, for the packets arriving in node B from yet another shortcut, they can also adaptively choose to follow the tree link if the shortcut leading to node D is blocked. A shortcut link thus can never become part of a channel dependency cycle, and deadlock freedom is guaranteed. Note that if wormhole switching is used, deadlock may occur in a network with a routing scheme like TRAIN. The reason for this is that the header of a packet may have crossed the shortcut and then find that it is blocked ahead.

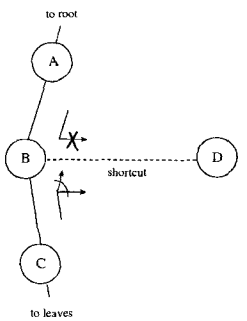

Figure 7: Deadlock prevention in TRAIN.

4. Performance Evaluation

In order to evaluate the effectiveness of the TRAIN scheme, we have analyzed the latency for an average packet in various network topologies. It is assumed that the network is *unloaded* (empty), and packets do not experience any network contention along the route. Simulation for networks under different traffic loads has also been performed to evaluate TRAIN. We have implemented an event-driven simulator which can model different network topologies and network conditions. In this section, analysis and simulation results are reported and their implications are discussed.

4.1. Latency Analysis for Unloaded Networks

We have analyzed the performance of the TRAIN scheme in terms of *base network latency*, which is the latency for a packet traversing an unloaded network. The TRAIN is compared with the Autonet routing, tree routing, and adaptive trail routing (ATR). The tree routing is a scheme that does not utilize any other links than the tree links. The *shortest path routing* scheme is also included to indicate the upper bound of the performance for routing in an unloaded network. The shortest path routing is an optimal scheme which always chooses the route with the shortest distance from the source node to the destination node. It is not a practical scheme without a routing table in the switch. Furthermore, it is not deadlock-free.

Table 1 shows the average base network latency for the shortest path routing, TRAIN routing, Autonet routing, tree routing, and adaptive trail routing. The average base network latency is calculated based on all the possible routes between any two nodes in the network. The network analyzed is the example we described in Section 2. The results indicate that the performance of TRAIN and ATR is superior to that of Autonet routing and tree routing. In addition, the performance of TRAIN and ATR is very close to that of the optimal scheme in this case. Note that the Autonet routing is slightly better than the tree routing owing to the extra links between the nodes at the same level of the tree.

	Shortest Path	TRAIN	Autonet	Tree	ATR
Avg. Latency	1.75	1.78	1.97	2.39	1.79
Variance of Utilization	4.80	6.58	10.86	25.42	18.29

Table 1. Average base network latency and variance of utilization over all the possible routes for the example network in Figure 2 in terms of number of hops. The hops from the source computing element to the switch and from the switch to the destination computing element are not counted.

Table 1 also shows the distribution of link utilization for the network, which can be used to indicate *fairness* of the network. The fairer the network is, the less likely packets will encounter congestion in the network. Two example curves for the distribution of link utilization are

shown in Figure 8. The horizontal dimension in the graph is the count of usage of the links, which is obtained by counting the times each link is "used" from all the possible unique routes in the network. The vertical dimension is the number of links which have a certain count of link usage. Figure 8 shows that the curve indicated by the solid line exhibits better fairness than the curve indicated by the dotted line since the former has a lower variance. The variance of link utilization for the above five schemes is also presented in Table 1. TRAIN has the best variance other than the shortest path scheme in this case.

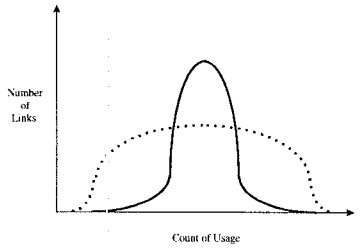

Figure 8: Distribution of link utilization. Number of links vs. count of usage. The curve indicated by the solid line has a lower variance than the one indicated by the dotted line.

The analysis results in Table 1 are obtained based on an example network only, and the conclusions may not hold for other networks. In order to further investigate the performance of these routing schemes, we have implemented a program to analyze them from many randomly generated networks. Table 2 shows the results from 50 randomly generated networks which are composed of 16 nodes and 32 links. The tree for TRAIN, Autonet routing, and tree routing is constructed by a breadth-first spanning algorithm. The average base network latency as well as the average *minimum base network latency* is compared. The average minimum base network latency is obtained by choosing "the best tree" from 16 tree configurations for each related routing scheme. Each of the 16 trees is constructed by using a different node as the root. Since the tree is constructed during initialization, an interconnection network can select the one that delivers the lowest average base network latency.

	Shortest Path	TRAIN	Autonet	Tree
Avg. Latency	1.97	2.31	2.87	3.19
Avg. Min. Latency	1.97	2.26	2.71	3.04

Table 2. Average base network latency and average minimum base network latency over all the possible routes from 50 randomly generated networks in terms of number of hops. Each network is composed of 16 nodes and 32 links.

The analysis results in Table 2 show that TRAIN performs better than the Autonet routing and the tree routing. The difference of the performance between TRAIN and the shortest path routing is also limited. The adaptive trail routing is not included here due to the complexity of generating the deadlock-free routing table automatically.

Table 3 shows the analysis results from 50 randomly generated networks which are composed of 16 nodes and 26 links. Again, TRAIN performs consistently better than the Autonet routing and the tree routing. Note that the average base network latency and the average minimum base network latency are higher than those in Table 2. This is due to the fact that with fewer links, a packet needs to traverse more hops on the average.

	Shortest Path	TRAIN	Autonet	Tree
Avg. Latency	2.31	2.61	3.11	3.41
Avg. Min. Latency	2.31	2.53	2.90	3.12

Table 3. Average base network latency and average minimum base network latency over all the possible routes from 50 randomly generated networks in terms of number of hops. The network is composed of 16 nodes and 26 links.

4.2. Simulation for Loaded Networks

The analysis of the base network latency in Section 4.1 is only valid when the network has no traffic. In order to evaluate the various routing schemes for networks under different traffic loads, a simulator has been developed. The simulator is even-driven and implemented by C language. The shortest path routing scheme is not included in the simulation since deadlock may occur in the network.

The simulations are based on the following properties: 1) packet size is fixed at 32 bytes. 2) during each cycle there is an equal probability of generating a packet at each of the inputs of the network. 3) packet destinations are uniformly distributed over the outputs of the network. We assume that the packet header is 4 bytes. The links are 2-byte wide in each direction. Virtual cut-through switching is assumed. There are two virtual channels at each input port to buffer the arrived packets, and the capacity of each virtual channel is 64 bytes [3]. In addition, each network node consists of a computing element and an associated switch.

The latency for a packet to traverse a switch is as follows. It takes a cycle for two bytes of packet data to be transmitted across the link between two switches. Each input port spends a cycle to buffer the arriving packet. The routing decision takes a cycle. One more cycle is then spent on arbitration of the crossbar. After that, packet data cross the crossbar to the output port, which takes another

cycle. Note that a packet may stay in the input buffer for longer if there is contention at the crossbar or the destined output is blocked.

The performance measures used in our simulation include the *average latency* and the *normalized throughput*. The latency is the number of cycles that elapses from when the first two bytes of a packet generated at an input to when it leaves the network. The normalized throughput is the average number of bytes received by each output per clock cycle. Each simulation run is terminated after 5,000 packets have arrived at their destination outputs. Statistics are gathered only after 2,000 packets in order to remove start-up effects.

Figure 9 shows the simulation results for the example network described in Section 2. This particular network has 9 nodes and 13 links. The performance of TRAIN and ATR is better than that of the Autonet routing with a higher maximum throughput and a lower average latency. However, their performance difference is not significant. With the tree routing, the network reaches saturation at a much lower throughput and the latency is significantly higher.

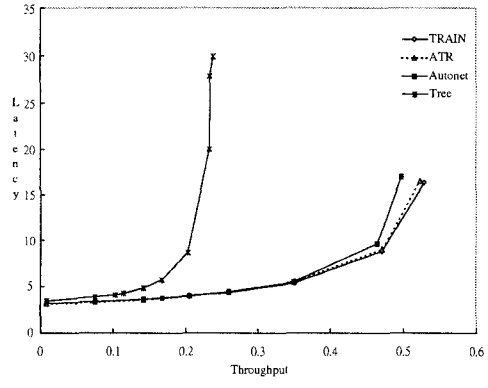

Figure 9: Simulation for the example network in Figure 2. Average latency vs. normalized throughput.

The simulation results in Figure 9 are only for a particular case. To further study the performance comparison for the various routing schemes, we have also simulated tens of network topologies generated randomly. Due to the limited space, only three cases which represent typical results are presented in the following. The adaptive trail routing is not included in these cases due to, again, the complexity of generating the deadlock-free routing table automatically. The three cases have:

1) 16 nodes and 32 links
2) 16 nodes and 26 links
3) 32 nodes and 64 links

The simulation results for case 1 are shown in Figure 10. The tree routing again has the worst performance. However, the performance of the TRAIN scheme outperforms the Autonet routing significantly. The reason for this is that the extra shortcut links between different levels of the tree provide the network with higher bandwidth and lower latency. Although this is only a particular case for such a network size. From the simulation for other random topologies with the same size, we have found that the conclusions hold.

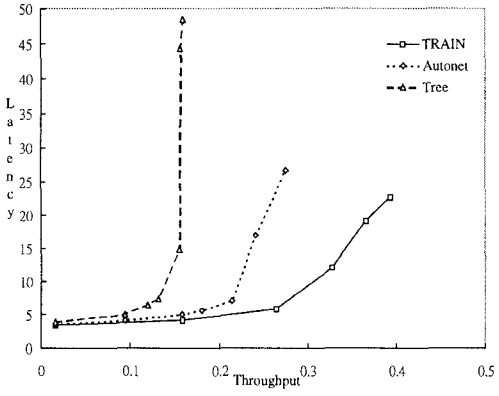

Figure 10: Simulation for a randomly generated network composed of 16 nodes and 32 links. Average latency vs. normalized throughput.

Figure 11 shows the simulation results for case 2. This case has the same number of nodes but fewer links, and hence each routing scheme has a lower maximum throughput and a higher latency than case 1, respectively. The tree routing again has a poor performance. The TRAIN scheme still outperforms the Autonet scheme significantly.

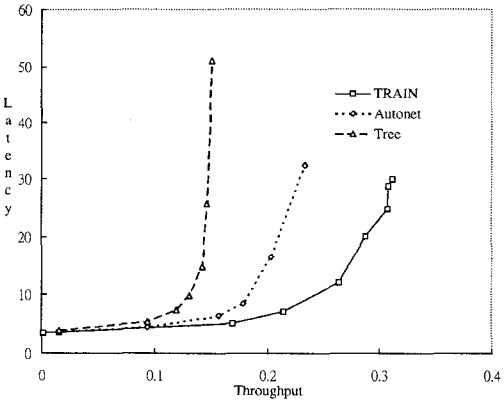

Figure 11: Simulation for a randomly generated network composed of 16 nodes and 26 links.

The performance impact of increasing the network size is shown in Figure 12. The case 3 network simulated

has 32 nodes and 64 links. Compared to the results in Figures 10 and 11, each routing scheme has a lower maximum throughput and a higher latency, respectively. This is due to higher contention in the network. The TRAIN scheme again performs better than both the Autonet and tree schemes consistently.

5. Summary and Conclusions

Interconnection networks with irregular topologies are suitable for workstation clusters since they provide incremental scalability. Deadlock prevention for irregular networks, however, is more difficult than the counterpart for regular networks such as mesh, torus, and hypercube. We have proposed a cost-effective deadlock-free routing scheme called TRAIN for irregular networks. With TRAIN routing, the switch uses an algorithmic routing function to decide the next hop adaptively for the arriving packet. No routing table is required in the switch for the TRAIN scheme, and hence the switch can be small and fast. It is expected that a TRAIN switch can be implemented in a single VLSI chip.

To evaluate the effectiveness of the TRAIN routing scheme, we have analyzed the network latency for TRAIN and other routing schemes in unloaded networks. Our analysis results show that TRAIN outperforms the previously proposed routing schemes and is close to the optimal shortest path scheme. We have also implemented an event-driven simulator to evaluate the TRAIN routing scheme in loaded networks. Many irregular networks with different network sizes have been simulated. Our results show that TRAIN outperforms other routing schemes consistently.

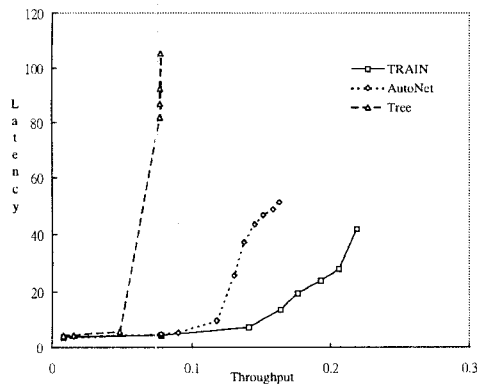

Figure 12: Simulation for a randomly generated network composed of 32 nodes and 64 links.

References

1. R. V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *Proc. Int'l Symp. on Computer Architecture*, pp. 351-360, May 1993.

2. A. A. Chien and J. H. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors," *Journal of ACM*, pp. 91-123, January 1995.

3. W. J. Dally, "Virtual Channel Flow Control," *IEEE Trans. on Computers*, vol. 3, pp. 194-205, March 1992.

4. W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. on Computers*, vol. 36, no. 5, pp. 547-553, May 1987.

5. J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 12, November 1993.

6. C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," *Proc. Int'l. Symp. on Computer Architecture*, pp. 278-286, May 1992.

7. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, vol. 3, pp.267-286, 1979.

8. L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE on Computers*, vol. 26, pp. 62-76, February 1993.

9. W. Qiao and L. M. Ni, "Adaptive Routing in Irregular Networks Using Cut-Through Switches," *Proc. Int'l Conf. on Parallel Processing*, August 1996.

10. M. D. Schroeder et al., "Autonet: a High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," *SRC Research Report 59*, DEC, April 1990.

11. Y. Tamir and G. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communication Switches," *Proc. Int'l. Symp. on Computer Architecture*, pp. 343-354, 1988.

12. Y. Tamir and H. C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication Switches," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13-27, January 1993.