Optimizing Up*/Down* Routing By Minimal Paths

Lalit Kishore Arora
Assistant Professor,MCA Dept,
AKG Engg College,
Ghaziabad, UK, India

Rajkumar
Reader,CSE Dept,
Gurukul Kangri Vishva Vidyalaya,
Haridwar, UK, India

ABSTRACT
Networks of workstations (NOWs) often uses irregular interconnection patterns. Up*/down* is the most popular routing scheme currently used in NOWs with irregular topologies. One of the main problem with up*/down* routing is difficult to route all packets through minimal paths. Several solutions have been proposed in order to improve the up*/down* routing scheme. In this paper we discussed those solutions which provide minimal paths to route most the packets to improve the performance of the up*/down* routing.

General Terms
Computer Architecture, Parallel and distributed Computing, Embedded Systems

Keywords
Networks Of Workstations, Irregular Topologies, Routing Algorithms, Minimal Path, Spanning Tree.

INTRODUCTION
Networks of workstations (NOWs) are becoming increasingly popular as a cost-effective alternative to parallel computers. In these machines, the network connects processors using irregular topologies, providing the wiring flexibility, scalability, and incremental expansion capability required in this environment. Also, when performance is the primary concern, these network products are being used to build large commodity clusters with regular topologies [13]. Some commercial interconnects for NOWs are Myrinet [11], Servernet II [1], Autonet [10], Gigabit Ethernet [14], and InfiniBand [4]. And several high-performance interconnects have been recently introduced for NOWs, including the Quadrics QsNet [2], and QsNet II [12], and Sun Fire Link [16].

In some of these networks, packets are delivered using source routing. In this kind of networks, the path to destination is built at the source host and it is written into the packet header before it is transmitted. Switches route packets through the fixed path found at the packet header. One example of network with source routing is Myrinet [11].

Usually, NOWs are arranged as switch-based networks whose topology is defined by the customer in order to provide wiring flexibility and incremental expansion capability. Often, due to building constraints, the connections between switches do not follow any regular pattern leading to an irregular topology. The irregularity in the topology makes the routing and deadlock avoidance quite complicate. In particular, a generic routing algorithm suitable for any topology is required.

Up*/Down* [1] is the most popular routing algorithm used in the NOW environment. In this paper we discussed the up*/down* routing and the solutions to improve the performance of up*/down* routing. Section II we discussed the up*/down* routing and its drawbacks. Section III to VI explain the methodologies to improve the performance of up*/down* routing via route the maximum packets through minimal paths.

1. UP*/DOWN* ROUTING
Up*/down* routing is the most popular routing scheme currently used in commercial networks, such as Myrinet [11]. It is a generic deadlock-free routing algorithm valid for any network topology.

Up*/down* is a distributed deadlock-free routing algorithm that provides partially adaptive routing in irregular networks. In order to fill the routing tables, a breadth-first spanning tree (BFS) on the graph of the network is computed first using a distributed algorithm. Routing is based on an assignment of direction labels (“up” or “down”) to the operational links in the network by building a BFS spanning tree. To compute a BFS spanning tree a switch must be chosen as the root. Starting from the root, the rest of the switches in the network are arranged on a single spanning tree [10].

After computing the BFS spanning tree, the “up” end of each link is defined as: 1) the end whose switch is closer to the root in the spanning tree; 2) the end whose switch has the lowest identifier, if both ends are at switches at the same tree level. The result of this assignment is that each cycle in the network has at least one link in the “up” direction and one link in the “down” direction. To avoid deadlocks while still allowing all links to be used, this routing scheme uses the following up*/down* rule: a legal route must traverse zero or more links in the “up” direction followed by zero or more links in the “down” direction. Thus, cyclic channel dependencies [15] are avoided because a packet cannot traverse a link in the “up” direction after having traversed one in the “down” direction.

When a message arrives at a switch, the routing algorithm is computed by accessing the routing table. The address of the table entry is obtained by concatenating the input port number with the

address of the destination node stored in the message header. If there are several suitable outgoing ports, one of them is selected.

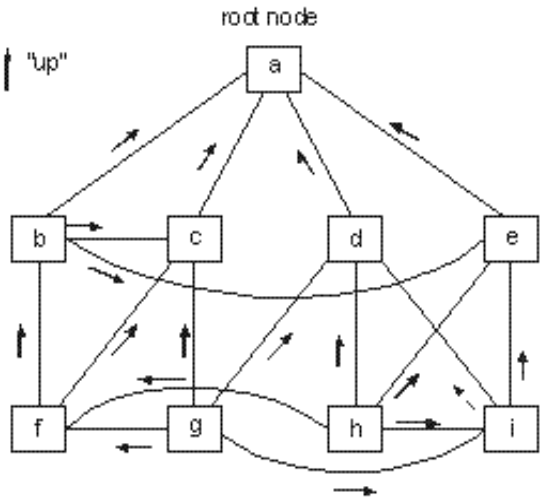

Figure 1 BFS spanning tree and assignment of directions to links for a 9 switch network

The main advantage of using up*/down* routing is the fact that it is simple and easy to implement. However, there exist several drawbacks that may noticeably reduce network performance. First of all, this routing scheme does not guarantee all the packets to be routed through minimal paths. This problem becomes more important as network size increases. In general, up*/down* concentrates traffic near the root switch, often providing minimal paths only between switches that are allocated near the root switch [8], [7]. Additionally, the concentration of traffic in the vicinity of the root switch causes a premature saturation of the network, thus obtaining a low network throughput and leading to an uneven channel utilization.

There fore the main drawbacks of up*/down* routing are the unbalanced channel utilization and the difficulties to route most packets through minimal paths, which negatively affects network performance.

Several solutions have been proposed in order to improve the up*/down* routing scheme, such as the In-transit Buffer [5], the DFS methodology [9] , Adaptive-trail routing [15] , and Smart routing [19].

2. DFS METHODOLOGY

The DFS methodology [9] is a new methodology to compute the up*/down* routing tables that makes a different assignment of direction ("up" or "down") to links in order to increase the number of minimal paths followed by the messages. This methodology is based on obtaining a depth-first search spanning tree (DFS) instead of the BFS spanning tree used in the original methodology of up*/down* routing.

Like in the up*/down routing with BFS spanning tree, an initial switch must be chosen as the root before starting the computation of the DFS spanning tree. The selection of the root is made by using heuristic rules [8]. For instance, the switch with the highest average topological distance to the rest of the switches will be selected as the root node. The rest of the switches are added to the DFS spanning tree following a recursive procedure. Unlike the BFS spanning tree, adding switches is made by using heuristic rules [8]. Starting from the root switch, the switch with the highest number of links connecting to switches that already belong to the tree is selected as the next switch in the tree. In case of tie, the switch with the highest average topological distance to the rest of the switches will be selected first. Next, in order to assign directions to links, switches in the network must be labeled with positive integer numbers.

When assigning directions to links, the "up" end of each link is defined as the end whose switch has a higher label. Figure 2 shows the new link direction assignment for the same network graph depicted in Figure 1. It has been shown that the DFS methodology [9] provides more minimal paths than the BFS one, resulting in a significant increase in network performance [8].

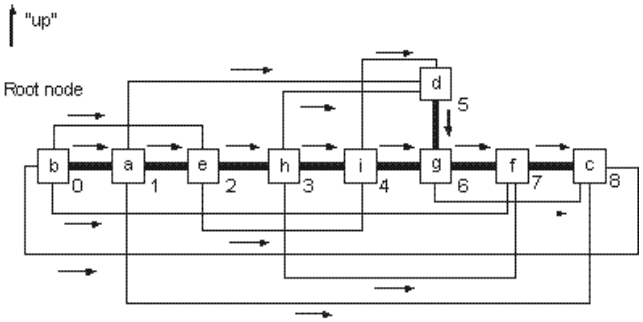

Figure 2 DFS spanning tree and assignment of directions to links for a 9 switch network

3. ADAPTIVE-TRAIL ROUTING

Adaptive-trail routing algorithm [15] is applicable to any network topology with Eulerian trails. Adaptive-trail routing (ATR) algorithm is based on computing an Eulerian trail. The basic idea of the ATR is to find two opposite unidirectional Eulerian trails to provide reasonable routing paths and control the order of channel dependency. The Eulerian trail is a sequence of channels, which visits each channel once and exactly once so that it can maintain the order of channel dependency. In order to maximize channel utilization and allow more and shorter routing paths, shortcuts are added to the two unidirectional Eulerian trails. The two unidirectional trails with shortcuts are called adaptive trails. To avoid deadlock, some shortcuts have to be removed or used in a restricted way based on the channel dependencies along the adaptive trails. However, a dependency cycle is allowed as long as there is an escape channel for that cycle. The allowed paths between pairs along the two adaptive trails define all legal routes. A static routing table is maintained in each switch to carry routing information.

The main drawback of Adaptive-trail routing is that it is impossible to compute an Eulerian trail in some irregular topologies, since all the switches must have even degree or exactly two switches must have odd degree. This limited applicability becomes important when the network dynamically changes its topology, which is quite frequent in a LAN environment because some links may fail or some components

may be added/removed. In addition, this routing algorithm can only be applied in networks with distributed routing.

4. SMART ROUTING

Up*/down* routing is fast and simple and guaranteed to find a deadlock-free routing. However, it tends to concentrate traffic at the root node, which restricts the performance. Since balancing the traffic in an irregular topology requires a relatively expensive solution to a multi-commodity flow problem, it seems reasonable to use a more complex deadlock-free routing algorithm that does not suffer from the root-node congestion problem.

The paper [19] attempting this point using Smart routing. Rather than break the buffer cycles by arbitrarily picking a root node and performing a search from that node, instead building an explicit buffer dependency graph and search it for cycles. For each cycle, algorithm break the dependency that minimizes some heuristic cost function. The procedure terminates when the buffer dependency graph has no cycles. The routing is represented implicitly by the buffer dependency graph: it is the paths of connected buffers in the buffer dependency graph that lead from the source node to the destination node.

Thus, smart routing is a greedy technique guided by a heuristic function. Ideally, the heuristic cost function would be the actual topology throughput. Since computing this requires a multi-commodity flow solution, putting this in the inner loop of the routing search would be prohibitively costly. Instead, algorithms use a much simpler heuristic: the average path length. A secondary heuristic attempts to distribute the cuts among the various switches in the topology.

The Smart routing algorithm [19] is based on a linear programming solver to balance traffic while it tries to break the deadlock cycles. Although the Smart routing algorithm can be applied to both source and distributed routing, this routing algorithm is impractical due to its high computational overhead, especially in large networks. Smart routing balances channel utilization assuming a uniform traffic. However, in real networks, non-uniform traffic is commonly observed.

5. IN-TRANSIT BUFFER MECHANISM

In the In-transit Buffer mechanism [5], all the minimal paths are allowed by absorbing the messages in those intermediate nodes of the path where there is a forbidden transition ("down" → "up") according to the up*/down* routing algorithm.

Basically, this mechanism avoids routing restrictions by ejecting packets at intermediate hosts and later re-injecting them. This mechanism can be easily implemented in Myrinet by modifying the network control program at the network interface card without changing the network hardware. This mechanism was originally proposed to provide minimal routing to up*/down*. In this routing algorithm, ITBs are put in all the down-up transitions. The mechanism has been extensively evaluated for both irregular [5] and regular networks [6] under different traffic patterns, network topologies, network sizes, and different message sizes. Overall, this mechanism improves on the performance achieved by up*/down*. Moreover, as network size increases, more benefits are obtained since the up*/down* routing does not scale well.

The basic idea of the mechanism is to break cyclic dependences with host buffering. The paths between source-destination pairs are computed following any given rule and the corresponding CDG is obtained. Then, the cycles in the CDG are broken by splitting some paths into sub-paths. To do so, an intermediate host inside the path is selected and used as an in-transit buffer (ITB); at this host, packets are ejected from the network as if it were their destination. The mechanism works similarly to the cut-through switching technique. Therefore, packets are re-injected into the network as soon as possible to reach their final destination. Notice that the dependences between the input and output channels of the switch are completely removed because, in the case of network contention, packets will be completely ejected from the network at the intermediate host. The CDG is made acyclic by repeating this process until no cycles are found. Notice that more than one intermediate host may be needed for a particular path [17].

As an example [17], Fig. 3.a shows a network and the assignment of link directions following the up*/down* rule. Although there is a minimal path between switch 4 and switch 1 (4 → 6 → 1), it is forbidden because it uses an up link after a down link at switch 6.

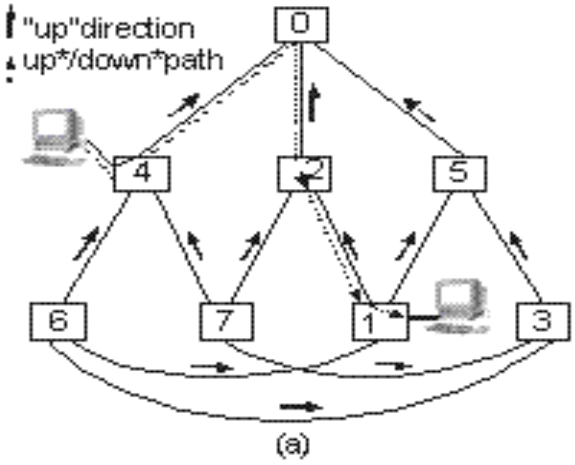

Figure 3 (a): Link direction assignment and use of the ITB mechanism for an irregular network.

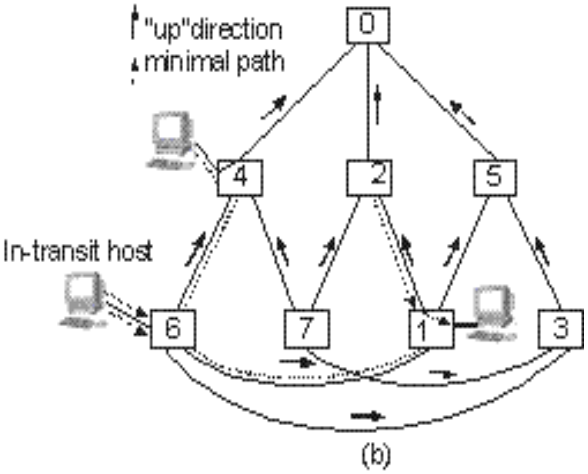

Figure 3 (b): Link direction assignment and use of the ITB mechanism for an irregular network.

However, with the ITB mechanism (see Fig. 3.b), this path is allowed by using one host at switch 6 as an in-transit host to break the dependence. By using ITBs, minimal routing can be guaranteed while keeping deadlock freedom.

Although this mechanism can be applied in networks with both source and distributed routing, it requires large enough buffers to store the ejected packets, DMA support, processors at the NICs in order to manage the in-transit messages, and the use of a new message format in the network to distinguish the in-transit messages. Moreover, it requires at least one host to be attached to every switch in the network.

6. CONCLUSION

Several solutions are proposed to improve up*/down* routing, but we find and discussed best solutions. The In-transit Buffer [5], Smart routing [19] and the DFS methodology [9] increase the number of minimal paths. But the In-transit Buffer mechanism requires large enough buffers, DMA support and processors at NICs. Where Smart routing balances channel utilization assuming a uniform traffic but maximum time non-uniform traffic observed in real networks. Therefore, DFS methodology [9], unlike other approaches that require specific hardware support, provide a low-cost alternative to improve the performance of the Up*/down* routing algorithm.

In the next step we are designing a simulation environment where we compare above approaches at different traffic patterns. Above approaches may perform best at specific parameters but the cost and performance are to observed at any type regular or irregular topologies.

7. REFERENCES

[1] Horst, R.,1996, "ServerNet deadlock avoidance and fractahedral topologies", in Proc. of the Int. Parallel Processing Symp. .

[2] Petrini,F. et.al, 2003, "Performance Evaluation of the Quadrics Interconnection Network", Journal of Cluster Computing, pp. 125-142.

[3] Silla, F. and Duato,J., 1997,"Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology", Int. Conference on High Performance Computing.

[4] InfiniBandTM Trade Association, InfiniBandTM architecture. Specification Volume 1. Release 1.0.a. Available at http://www.infinibandta.com.

[5] Flich, J. et.al,2000, "Performance Evaluation of a New Routing Strategy for Irregular Networks with Source Routing", Proc. Int'l Conf. Supercomputing.

[6] Flich, J. et.al, 2000, "Improving the Performance of Regular Networks with Source Routing", Proc. Int'l Conf. Parallel Processing.

[7] Flich, J. et.al, 2000, "Combining In-Transit Buffers with Optimized Routing Schemes to Boost the Performance of Networks with Source Routing", Proc. of Int. Symp. on High Performance Computing.

[8] Sancho, J. and Robles, A.,2000, "Improving the Up*/Down* Routing Scheme for Networks of Workstations", in Proc. of Euro-Par.

[9] Sancho, J. et.al,2000, "New Methodology to Compute Deadlock-Free Routing Tables for Irregular Networks", in Proc. of 4thWorkshop on Communication, Architecture and Applications for Networkbased Parallel Computing.

[10] Schroeder, M. et al.,1990, "Autonet: A high-speed, self-configuring local area network using point-to-point links", SRC research report 59.

[11] Boden,N.J. et al.,1995, "Myrinet - A gigabit per second local area network", IEEE Micro, vol. 15.

[12] Quadrics. Available: http://www.quadrics.com.

[13] Riesen, R.et al,1999, "CPLANT", in Proc. of the 2nd. Extreme Linux Workshop, June 1999.

[14] Sheifert, R., 1998, "Gigabit Ethernet", Addison-Wesley.

[15] Qiao, W. and Ni, L.M.,1996, "Adaptive routing in irregular networks using cut-through switches," in Proc. of the 1996 International Conference on Parallel Processing.

[16] Qian, Y. et.al, 2004, "Performance Evaluation of the Sun Fire Link SMP Clusters", 18th International Symposium on High Performance Computing Systems and Applications, HPCS 2004, pp. 145-156.

[17] Flich, J. et.al,2003, "Applying In-Transit Buffers to Boost the Performance of Networks with Source Routing", IEEE Transactions On Computers, Vol. 52, No. 9.

[18] Silla,F. et.al,1997, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," in Workshop on Communications and Architectural Support for Network-based Parallel Computing.

[19] L. Cherkasova, V. Kotov, and T. Rockicki,1996, "Fibre Channel Fabrics: Evaluation and Design," Proc. 29th Hawaii Int'l Conf. System Sciences, Jan. 1996.