On the Influence of the Selection Function on the Performance of Fat-trees *

F. Gilabert, M.E. Gómez, P. López and J. Duato
Dept. of Computer Engineering
Universidad Politécnica de Valencia
E-mail: fragivil@gap.upv.es

No Institute Given

Abstract. Fat-tree topology has become very popular among switch manufacturers. Routing in fat-trees is composed of two phases, an adaptive upwards phase, and a deterministic downwards phase. The unique downwards path to the destination depends on the switch that has been reached in the upwards phase. As adaptive routing is used in the ascending phase, several output ports are possible at each switch and the final choice depends on the selection function. The impact of the selection function on performance has been previously studied for direct networks and has not resulted to be very important. In fat-trees, the decisions made in the upwards phase by the selection function can be critical, since it determines the switch reached in the upwards phase, and therefore the unique downwards path to the destination. In this paper, we analyze the effect of the selection function on fat-trees. Several selection functions are defined, compared and evaluated. The evaluation shows that selection function has a great impact on fat-trees.

Keywords: selection function, adaptive routing, fat-tree, interconnection networks

1 Introduction

Clusters of PCs have grown in popularity in the last years due to their excellent cost-performance ratio. The interconnection network has a great impact in the performance of these systems. Several switch-based point-to-point commercial networks are currently available. As long as high degree switches are available, multistage networks (MINs) have become very popular. Among them, the fat-tree topology is the preferred choice (e.g.: Mellanox [13], Myricom [15], Quadrics [14]). Routing is one of the most important design issues of interconnection networks. The routing strategy determines the path that each packet follows between a source–destination pair. Routing is deterministic if only one path is provided

* This work was supported by the Spanish MCYT under Grant TIC2003-08154-C06-01.

for every source–destination pair, or adaptive, if several paths are available. Adaptive routing better balances network traffic, thus allowing the network to obtain a higher throughput. The routing algorithm is implemented by means of the routing and selection functions [3]. The routing function supplies a set of suitable routing options to reach the destination. A choice from this set is made by the selection function based on network status.

Many works [1], [2], [3] has pointed out the great influence that the routing function has on network performance especially for direct networks. There are also some papers [10], [8], [6] that analyze the influence of selection function, showing that it has some small impact on performance. In these networks, the routing function is adaptive along the whole path. However, this is not the case of fat-trees topologies. Routing in fat-trees is performed in two phases, an upwards adaptive one and a downwards deterministic one. The unique path to follow in the downwards phase is determined by the selected upwards path. So, in fat-trees the selection made in the upwards path is responsible of balancing network traffic. Thus, we expect that the selection function will have a greater influence on interconnection network performance.

The rest of the paper is organized as follows. Section 2 presents some background on the fat-tree topology. Section 3 contains references to related work. In Section 4, we propose several selection functions for fat-trees, analyzing their performance in Section 5. Finally, some conclusions are drawn.

2 Fat-trees

A multistage interconnection network (MIN) is a regular topology in which switches are identical and organized as a set of stages. Each stage is only connected to the previous and the next stage using regular connection patterns. Depending on the interconnection scheme employed between two adjacent stages, several MINs have been proposed. In this paper, we focus on the fat-tree topology.

A fat-tree topology is based on a complete tree. Unlike traditional trees, fat-trees get thicker near the root. A set of processors is located at the leaves and each edge of the tree corresponds to a bidirectional channel. However, the degree of the switches increases as we go nearer to the root, which makes the physical implementation unfeasible. Hence, some alternative implementations have been proposed in order to use switches with constant degree, as the k-ary n-trees. A k-ary n-tree is

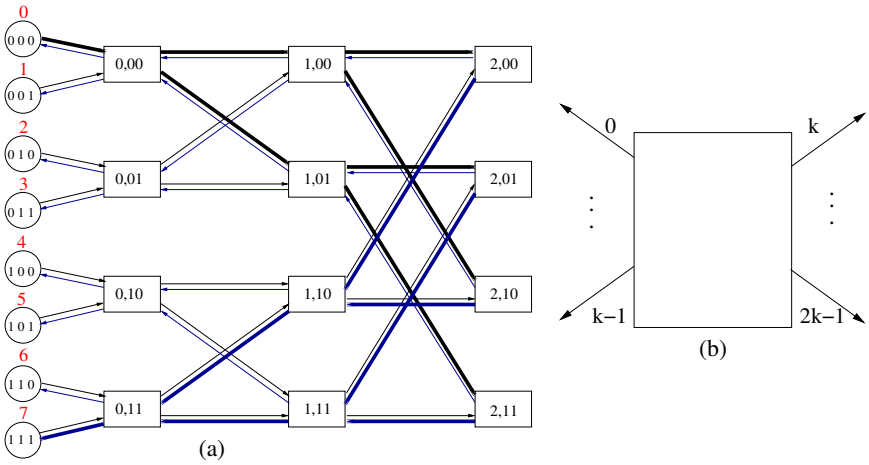

Fig. 1. a) The four possible paths from source 0 to destination 7 in a 2-ary 3-tree. (b) Link numbering in switches of a k-ary n-tree.

composed of $N = k^n$ processing nodes and nk^{n-1} switches, with k input and k outputs ports. In what follows, we will use either the term fat-tree or k-ary n-tree to refer to k-ary n-trees.

Each processing node is represented as a n-tuple $\{0, 1, ..., k-1\}^n$, and each switch is defined as a pair $\langle s, o \rangle$, where s is the stage where the switch is located at, that is $s \in \{0..(n-1)\}$, and o is a $(n-1)$-tuple $\{0, 1, ..., k-1\}^{n-1}$. Figure 1.(a) shows a 2-ary 3-tree, with 8 processing nodes and 12 switches. We consider stage 0 as the closest one to the processing nodes.

In a fat-tree, two switches $\langle s, o_{n-2}, ..., o_1, o_0 \rangle$ and $\langle s', o'_{n-2}, ..., o'_1, o'_0 \rangle$ are connected by an edge if $s' = s + 1$ and $o_i = o'_i$ for all $i \neq s$. On the other hand, there is an edge between the switch $\langle 0, o_{n-2}, ..., o_1, o_0 \rangle$ and the processing node $p_{n-1}, ..., p_1, p_0$ if $o_i = p_{i+1}$ for all $i \in \{n-2, ..., 1, 0\}$. Descending links will be labeled from 0 to k-1, and ascending links from k to $2k-1$ (see Figure 1.(b)).

In fat-trees, minimal routing from a source to a destination can be accomplished by sending packets forward to one of the nearest common ancestors of both source and destination and, from there, backward to destination. When crossing stages in the forward direction, several paths are possible, so adaptive routing is provided. Each switch can select any of its upward output ports. Then, the packet is turned around and sent backwards to its destination. Once the turnaround is crossed, a single path is available up to the destination node. The stage up to which the

packet must be forwarded up is obtained by comparing the source and destination components beginning from the $n-1$ (the most significant one). The fist pair of components that differs indicates the last stage to forward up the packet. For instance, in order to send a packet from the node $p_{n-1}, ..., p_1, p_0$ to the node $p'_{n-1}, ..., p'_1, p'_0$, the packet must be sent up to the stage i, if $p_j = p'_j$ for $j \in \{n-1..i+1\}$ and $p_i \neq p'_i$. Once at stage i, the descending path is deterministic. At each stage, the descending link to choose is indicated by the component corresponding to that stage in the destination n-tuple. In the example, from stage i, the packet must be forwarded through the p'_i link; at stage $i-1$ through link p'_{i-1}, and so on.

For instance in Figure 1.(a), a packet generated at node 0 whose destination is node 2 will be forwarded up to stage number one (through switch 0,00 and choosing either path to 1,00 or 1,01). From any of these switches, the remaining bits of the destination node (10 in our example) correctly forwards the packet.

3 Related Work

Although there is not any work about the impact of selection function in fat-trees, there are some previous works about this issue on other topologies. In [4], Duato proposed a time-dependent selection function for hypercubes, which prevents a message from using certain virtual channels until the time a message has been waiting exceeds some threshold value. In [1], Badr and Podar showed that the *zigzag* selection function is optimal for meshes, in the sense that it maximizes the probability of a message reaching the destination without delay. In [2], [5], and [6] the authors, analyzed the impact of the selection function in the routing algorithm performance in meshes and tori. In [9], Koibuchi et al. evaluated a selection function that is not specific to any topology. Finally, in [10] Martínez et al. analyzed the impact of selection function in the context of irregular topologies.

4 Selection Functions

An adaptive routing algorithm is composed of the routing and selection functions. The routing function supplies a set of output channels based on the current and destination nodes. The selection function selects an output channel from the set of channels supplied by the routing function.

All the selection functions proposed in this paper take into account the state of the output physical link offered by the routing function and then applies some criteria to select one of them. Virtual cut-trough switching with credit-based flow control is assumed. Notice that, if virtual channel multiplexing is available, the selection function does not select the virtual channel of the physical link that will store the packet. The virtual channel will be selected when the packet is transferred through the link, and the first virtual channel with a free buffer will be selected.

The selection functions presented below provide a preferred ascending link (i). When this link is not free, it performs a linear rotative search starting at link $i + 1$ until it finds a free link, if any.

We propose a possible hardware implementation for each selection function. Unless said otherwise, all the selection functions can be implemented in two steps: the first one will obtain the preferred link, and the second implements the linear rotative search by using a programmable priority encoder which takes the preferred link as an extra input. The encoder will give the highest priority to the input represented by this value. The first step changes according to the particular selection function to implement. Taking into account the set of physical links and the physical link that is preferred, it changes the order of the set of physical links in order to put the preferred one in the first position.

We have tested the following selection functions:

First Free (FF). The FF selection function selects the first physical link which has free space. It uses a lineal search, starting at the first ascending physical link (the k^{th}, port according to our notation). FF can be implemented by using a plain priority encoder.

Static Switch Priority (SSP). In a given stage, the SSP selection function assigns the highest priority to a different ascending link at each switch. The idea is to create a disjoint high priority ascending path for each switch of the first stage. Hence, packets coming from different switches at the first stage will reach different switches at the last one, thus balancing the traffic. The high priority physical link for the switch $\langle s, o_{n-2}, ..., o_1, o_0 \rangle$ is the ascending link labeled $k + o_s$. SSP needs the hardware mentioned above, connecting the switch component $\langle o_s \rangle$ to the programmable priority encoder.

Static Destination Priority (SDP). The SDP selection function assigns priorities to physical links at each switch depending only on the packet destination. The preferred physical link is given by the least significant component of the packet destination, which represents the port that the destination is attached to in the first stage. That is, a packet sent

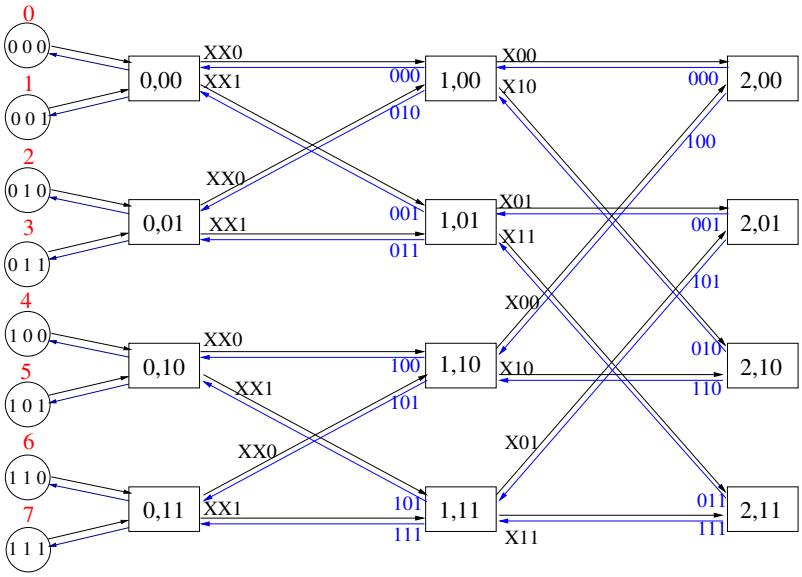

Fig. 2. Preferred links for each destination in SADP for a 2-ary-3-tree

to processing node $p_{n-1}, ..., p_1, p_0$ has as the preferred link the $k + p_0$ link. Thus the ascending paths of two packets with destination nodes $p_{n-1}, ..., p_1, p_0$ and $p'_{n-1}, ..., p'_1, p'_0$ are not disjoint only if $p_0 = p'_0$. SDP uses the last component of the packet destination to control the programmable priority encoder.

Static Origin Priority (SOP). The SOP selection function assigns priorities to physical links depending only on the packet source. The preferred physical link is given by the least significant component of the packet source (which represents the port that the origin is attached to in the first stage). That is, for a packet sent from processing node $p_{n-1}, ..., p_1, p_0$, it is $k + p_0$. The hardware implementation of SOP is the same as SDP, but connecting the last component of the packet source to the programmable priority encoder.

Stage And Destination Priority (SADP). The SADP selection function takes into account both the stage at which the switch belongs to and the component of the packet destination corresponding to that stage, (i.e., a switch located at stage s considers the s^{th} component of the destination address). That is, at the switch $\langle s, o_{n-2}, ..., o_1, o_0 \rangle$, the highest priority physical link for a packet with destination $p_{n-1}, ..., p_1, p_0$ will be $k + p_s$. As a consequence, each switch located at the top of the tree concentrates traffic destined to all processors whose id differ only by the most signi-

ficant digit. Indeed, the paths to these destination are disjoint, as each one is reachable through different output ports of the switch. Figure 2 illustrates this selection function.

Packets are classified according to their destination considering all the components and not only the last one component as SDP does. The main difference between SADP and SDP is that in SDP packets destined to different nodes can have the same preferred links if their destination nodes have the same least significant component (p_0). On the other hand, in SADP, only those packets that share the first switch and are destined to the same node will share all the preferred links along the complete upwards phase.

To implement SADP, we need an additional multiplexer to select a different component of the packet destination at each stage. The output of this multiplexer is connected to the programmable priority encoder.

Cyclic Priority (CP). The CP selection function uses a round robin algorithm to choose a different physical link each time a packet is forwarded. The implementation of CP needs a counter that is incremented each time a packet is routed. The counter is connected to the programmable priority encoder.

More Credits (MC). Since we use credits to implement the flow control mechanism, the MC selection function selects the link which has the highest number of credits available. This number is determined by the sum of the credits available in the all the virtual channels of the physical link. The implementation of MC is more complex, as it needs several comparators to select the link with more available credits.

Random Priority (RP). It selects a random physical link each time a packet is transmitted. This function obtained similar performance results as CP. This is due to the fact that with a high number of packets to transmit, CP and RP selects each physical link the same number of times without considering the source or destination of the packets. The implementation of RP is complex, because it is difficult to obtain by hardware a truly random number. As it obtains similar results to CP, we finally decide to not to consider this selection function in this paper.

5 Performance Evaluation

5.1 Network Model

To evaluate the different selection functions proposed below, a detailed event-driven simulator has been implemented. The simulator models a k-ary n-tree with adaptive routing and virtual cut-through switching. Each

router has a full crossbar with queues both at the input and output ports. We assumed that it takes 20 clock cycles to apply the routing algorithm and the selection function, and switch and link bandwidth has been assumed to be one flit per clock cycle and fly time through the link has been assumed to be 8 clock cycles. These values were used to model Myrinet networks [7]. Credits are used to implement the flow control mechanism. Each physical input port can be multiplexed into up to 3 virtual channels, with space to store two packets. Also, each output port link has a two-packet output buffer.

Packet size is 8 Kb and packet generation rate is constant and the same for all the processors in the network. We have evaluated two different traffic patterns: uniform and complement. In the uniform traffic pattern, message destination is randomly chosen among all the processors in the network, while in the complement traffic pattern each processor sends all its messages to the opposite node. Thus, in a network with N processors the processor i sends messages to the processor $N - i - 1$. The complement traffic patterns has two interesting properties in fat-tree networks. The first one is that all the packets have to reach the upper stage in order to arrive to their destination, hence, the selection function must be applied several times. The second one is that each processor node only sends messages to one destination. This proves useful because with a good selection function, the preferred ascending path of two packets should not cross each other.

5.2 Evaluation Results

We have evaluated a wide range of k-ary-n-tree topologies. We have evaluated from 2-ary-2-tree (4 nodes) to 2-ary-8-tree (256 nodes), from 4-ary-2-tree (16 nodes) to 4-ary-6-tree (4096 nodes), from 8-ary-2-tree (64 nodes) to 8-ary-4-tree (4096 nodes), from 16-ary-2-tree (256 nodes) to 16-ary-3-tree (4096 nodes) and for a 32-ary-2-tree (1024 nodes). Due to space limitations, we show here only a subset of the most representative simulations.

Figure 3.(a) shows results for a very small network (4-ary 2-tree, 16 nodes). The behavior of the selection functions is not very different, with the exception of FF. FF always returns the same preferred ascending link, therefore an ascending path needs to be saturated before another one is selected. Hence, there is a really unbalanced link utilization as those that belong to the preferred ascending paths always have a higher utilization than the others. On the other hand, due to the fact that there are only 2

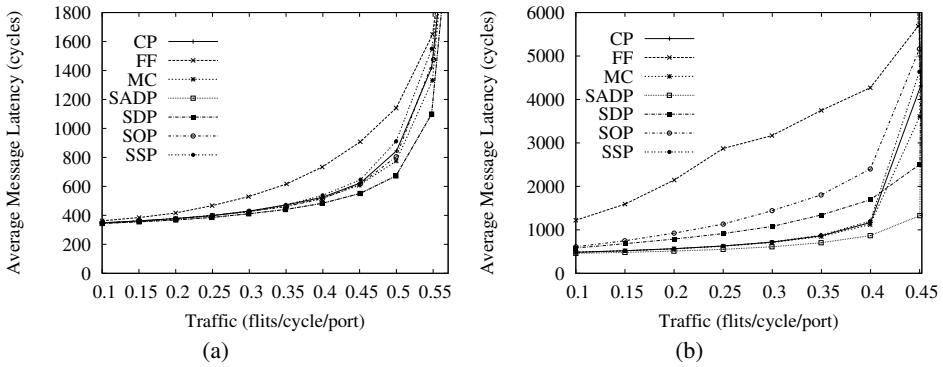

Fig. 3. Average message latency versus traffic with uniform traffic pattern. (a) 4-ary-2-tree. (b) 4-ary-4-tree.

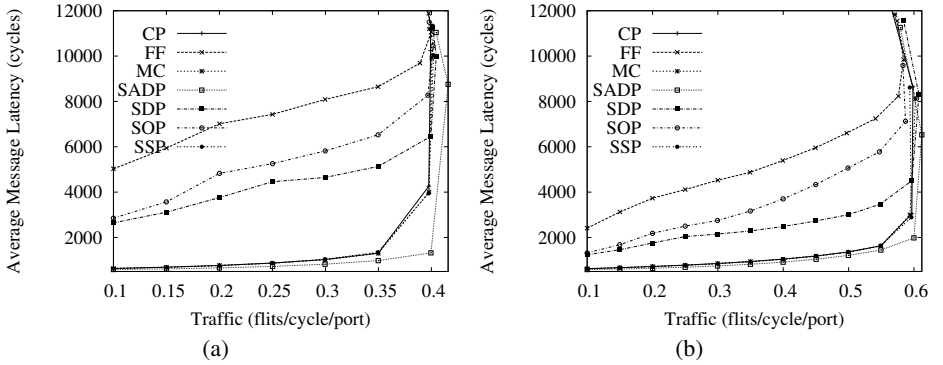

Fig. 4. Average message latency versus traffic with uniform traffic pattern. (a) 4-ary-6-tree with one virtual channel. (b) 4-ary-6-tree with three virtual channels.

stages, the rest of selection functions have almost the same performance, because with a low number of stages there is a low number of different paths that can be chosen to reach any destination.

Figure 3.(b) shows the results for a 4-ary 4-tree (256 nodes). As it can be seen, with 4 stages there are more differences in packet latency for the different selection functions. Despite SOP and SDP achieve a better performance than FF, they still have a high network latency. Their main drawback is that they select paths based only on the p_0 component of the packet origin (SOP) or destination (SDP). Therefore, the probability of obtaining disjoint paths and, thus, an even network utilization is quite low. On the other hand, CP, SSP and MC have almost the same performance, because they do a good job balancing the network utilization. However, in CP and MC there is not any mechanism to try avoiding that the ascending paths of packets cross each other. In SSP, each switch at

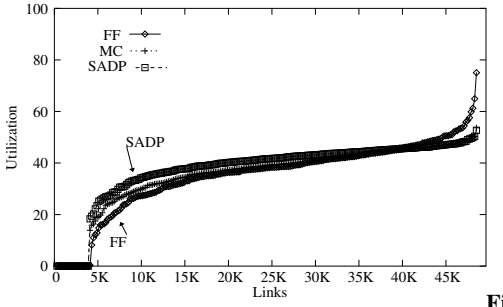

Fig. 5. Link utilization at saturation in a 4-ary-6-tree with uniform traffic for three selection functions.

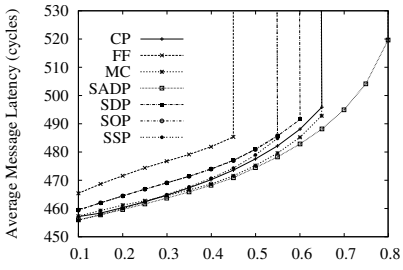

Fig. 6. Average message latency vs. accepted traffic with complement traffic for a 4-ary-4-tree.

each stage tries to send the ascending traffic to a different switch, but it does not take into account the destination of packets. Hence, the ascending paths of packets with different origin and destination processing nodes may cross. SADP achieves the best performance because, assuming that the preferred ascending path is free, only the ascending paths of packets sent to the same destination will cross each other.

Figure 4.(a) shows results for a larger network. In this case, the differences among selection functions are higher. This is due to the fact that with more stages, there are more different ascending paths to choose from, therefore more opportunities to balance traffic. On the other hand, the selection functions with a poor traffic balance achieve even worse results than in the previous examples.

Figure 4.(b) shows the effects of using virtual channel multiplexing. The use of virtual channels reduces the effect of the head-of-line blocking. By using three virtual channels, all the selection functions have a better performance, but it is also important to balance network traffic. Although all the selection functions benefits from the use of virtual channels, the ones that have a balanced use of the links give better results than the other ones.

We have also analyzed the impact of the arity (k) of the tree on performance (not shown). The difference among the evaluated selection functions keeps qualitatively the same. This is due to the fact that the impact of any selection function is greater when there is a high number on stages.

Figure 5 shows the utilization of network links for uniform traffic when injecting traffic at the saturation rate of each selection function. As can be expected, the selection functions that better balances traffic are the ones that obtains the best performance.

Figure 4 show the performance of the selection functions for a medium sized network 4-ary 4-tree (256 nodes). As expected, FF has the

worst performance. SOP and SDP with complement traffic have worse performance than the one obtained with uniform traffic, because they concentrates in the same switch of the last stage all the packets with the same least significant component of the packet source or destination, and in the complement traffic pattern all the packets reach the last stage. SSP with complement traffic also shows worse performance, as in SSP the chosen link does not depend on the source or destination node. Therefore, packets with different destinations crosses each other.

Both CP and MC achieve a good performance with complement traffic. MC has a better performance than CP because it takes into account the current number of credits of the link, and this allows MC to select the links that are less saturated. SADP shows the best performance of all. Remember that, with SADP, the preferred ascending path of two packets only can cross each other if the packets have the same destination and in the complement traffic pattern, every packet sent from a different node has a different destination. As a consequence, it achieves a good balance on link utilization.

We have also analyzed the hot spot traffic pattern. As we expected, all the selection functions proposed here obtain a very low performance, because we do not use any congestion control mechanism. On the other hand, other traffic patterns have been also analyzed (like perfect shuffle and bit reversal) and the overall results are qualitatively similar to the ones presented in this paper.

6 Conclusions

The selection function in fat-trees has a strong impact on network performance. As in the descending phase routing is deterministic, it is very important to choose the ascending path correctly. In this paper, we have proposed several selection functions for fat-trees, which use one of the two following strategies: the first one is to give always priority to one ascending path; the second one is to dynamically balance the link utilization without the use of preferred paths. FF, SDP, SOP, SSP and SADP use the first approach, while MC and CP use the second one. From the results of all the simulations we have performed, we can say that it is important to balance the utilization of the links, that is what CP and MC basically do. But an alternative way of achieving this good traffic balance is by correctly choosing ascending paths (SADP).

SADP provides the best results, because it chooses the preferred path in a manner so that the ascending paths of two packets only can cross

each other if the packets have the same origin switch or destination node. This provides a balance of link utilization even better than the one provided by MC. For example, a 4-ary-6 with uniform traffic using SADP as selection function reaches, for a medium network load, a 24.53% lower latency than the same fat-tree using MC and the latter has a more complex implementation. If we compare SADP with the naivest selection function (FF), SADP decreases latency by a factor of 8.9.

References

1. S. Badr and P. Podar, An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh-Connected Topologies, *IEEE Transactions on Computers*, Oct. 1989.
2. W. J. Dally and H. Aoki, Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels, *IEEE Transactions on Parallel and Distributed Systems*, Apr. 1993.
3. J. Duato, S. Yalamanchili and L. Ni, Interconnection Networks. An Engineering Approach. *Morgan Kaufmann*, 2004.
4. J. Duato, Improving the Efficiency of Virtual Channels with Time-Dependent Selection Functions, *Proc. Parallel Architectures and Languages Europe '92*, 1992.
5. C. Glass and L. M. Ni, The Turn Model for Adaptive Routing, *Journal of the Association for Computing Machinery*, Sep. 1994.
6. W. Feng and K. Shin, Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks, *Proc. ACM International Conference on Supercomputing*, 1997.
7. J. FLich, M.P. Malumbres, P. López, J. Duato, Improving Routing Performance in Myrinet Networks, *Proc. 14th International Parallel and Distributed Processing Symp.*, 2000.
8. A. Funahashi, M. Koibuchi, A. Jouraku, H. Amano, The Impact of Output Selection Function on Adaptive Routing, *Proc. International Conference on Computers And Their Applications*, Mar. 2001.
9. M. Koibuchi, A. Jouraku, H. Amano, MMLRU Selection Function: A Simple and Efficient Output Selection Function in Adaptive Routing, *IEICE Transactions*, 2005.
10. J.C. Martínez, F. Silla, P. López and J. Duato, On the Influence of the Selection Function on the Performance of Networks of Workstation, *Proc. High Performance Computing*, 2000.
11. F. Petrini and M. Vanneshi, k-ary n-trees: High Performance Networks for Massively Parallel Architectures, *Proc. International Parallel Processing Symp., 1997.*
12. F. Petrini and M. Vanneschi, Performance analysis of wormhole routed K-ary N-trees, *International Journal of Foundations of Computer Science*, 1997.
13. *Mellanox homepage http://www.mellanox.com/*
14. *Quadrics homepage http://www.quadrics.com/*
15. *Myricom homepage http://www.myri.com/*