Fibre Channel Fabrics: Evaluation and Design

Ludmila Cherkasova, Vadim Kotov, Tomas Rokicki
Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94303
e-mail: {cherkasova,kotov,rokicki}@hpl.hp.com

Abstract

We consider how Fibre Channel switches can be cascaded to form a Fibre Channel fabric. We begin with an analytical model of topology performance that provides a theoretical upper bound on fabric performance and a method for the practical evaluation of fabric topologies. We then consider the prevention of buffer-cycle deadlock in Fibre Channel networks. We show that though some topologies implicitly do not deadlock, these topologies have inferior throughput, and we present a technique for optimizing performance that takes into account both deadlock and traffic balancing.

1 Introduction

The Fibre Channel standard provides a mechanism for interconnecting heterogenous systems containing peripheral devices and computer systems through optical fiber media. In this paper, we consider how Fibre Channel switches can be cascaded to form a Fibre Channel fabric with the maximum throughput. Much work has been done on asymptotic limits on large networks. Our concern, instead, is how to obtain the highest performance from a network consisting of a small number of switches.

Each switch has a fixed number of ports. Each trunk link connecting two switches requires two ports. We shall assume that each port is connected to either a trunk link or a terminal port. The task is to build a network with the maximum number of terminal nodes, minimum number of switches, and highest throughput.

Fibre Channel switches have certain characteristics that affect the problem.

- The switches tend to be relatively large and expensive, unlike single-chip routing nodes used in MPPs. Thus, it is important to minimize the number of switches.
- The routing is oblivious; it is done with lookup tables in each switch; every source and destination pair can use a different route, but every source and destination pair always use precisely the same route. Thus, there is a neglible cost (other than wiring complexity) to using an irregular topology rather than a symmetrical topology.
- Finally, all packets are either delivered or else notification is sent to the sender that the packet could not be delivered. This distinguishes Fibre Channel from ATM, where a certain (low) packet loss is considered an acceptable solution to congestion control and deadlock.

The paper contains two major parts. The first part deals with Fabric topologies and their throughput, ignoring the possibility of deadlock. The second part discusses deadlock and its ramifications on Fibre Channel Fabrics.

In the first section, we present methods for analyzing and comparing topologies, the results of a computer search for good topologies, and an analytic model that fits the results well and that can be used to predict the Fabric performance.

The analytical model proposed in this paper does not depend on any specific characteristics of Fibre Channel and can be applied to many other types of networks. It provides a bound on the traffic throughput supported by any topology given specific parameters, and is useful for quickly exploring design alternatives or evaluating a specific topology against what might be achievable.

Over the past few decades, many hundreds of papers have been published on various specific interconnect topologies [AK89, ODH94, SS88, Sen89]. Some work has attempted a broad characterization and comparison of their performance [AJ75].

We evaluated and compared both specific symmetrical and general irregular topologies for networks of up to twenty-five switches. We found that the irregular topologies generally performed better than the more regular, symmetrical topologies. The result of this work is a topology database which can answer the following questions, among others:

- What is the best known topology with S switches each with c ports to support N terminal nodes?
- What is the minimal number of switches necessary to connect N terminal nodes and to support g traffic in a fabric, and what is the corresponding topology?
- What are the best known topologies that have N terminal nodes, expressed as a two-dimensional graph showing the number of switches versus the supportable throughput?

The second part of the paper discusses deadlock. Fibre Channel fabrics are particularly vulnerable to buffer-cycle deadlock. Only a small subset of topologies based of tree structure or complete graphs are innately free from deadlocks. Most optimal and high-performance fabric topologies will deadlock if not routed intelligently.

The only well-known deadlock-avoidance technique for arbitrary topologies is up*/down* [OK92], which has a substantial negative impact on the performance of some Fabric topologies [OK92]. Deadlock is prevented by restricting the set of paths that can be used between certain sources and destinations. The performance impact is highly topology dependent; some Fabric topologies are subject to a tremendous loss in performance; others suffer no loss.

A specific deadlock-free routing usually introduces inbalance in link utilization, so the traffic must be carefully assigned given a particular deadlock-free routing. This is true of routings ignoring the possibility of deadlock as well, but the restricted paths of a deadlock-free routing makes it even more essential.

In this paper we present a technique for finding a deadlock-free routing for an irregular topology that yields better throughput than up*/down*. We call the combination of this technique with traffic balancing "Smart Routing"; for most networks, there exists a smart routing that has very nearly the throughput of the unrestricted routing. The static routing tables in Fibre Channel switches allow simple static traffic-balancing.

2 Optimal Interconnect Topologies

The first consideration in cascading Fibre Channel switches to form fabrics is the topology to use. This section presents both a theoretical upper bound on fabric performance over the range of topologies and a practical evaluation of most current popular interconnect topologies.

We define the following fixed parameters:

- S is the number of switches in the network.
- N is the total number of nodes in the network.
- c is the number of ports on each switch. For simplicity, we assume the switches are all the same.

Our goal is to build a network with N nodes out of S switches with c ports. We shall assume uniform traffic from each node. We determine a lower bound on the number of switches required to construct such a network without reducing the bandwidth available from each node to every other node.

We also define the following variables:

- L is the number of trunk links in the fabric.
- $\bar{p}$ is the average path length between terminal nodes in hops. (Each trunk link traversed is one "hop"; the path length between a pair of terminal nodes connected to the same switch is considered to be zero.)
- p_{max} is the maximum over the minimum path length from one node to any other node.
- g is the amount of traffic generated by each node as a fraction of the port bandwidth.

The relationship between the number of terminal nodes and the number of trunk links is expressed by

$$2L + N = cS$$

2.1 Analysis of a Particular Topology

The throughput g of a particular topology is a function of that topology and the switch parameters. This section defines how that throughput is calculated.

We assume that each terminal port generates traffic for every other terminal port in a uniform manner. We also assume that each switch has enough buffering capacity and routing speed so that the main bottleneck is the bandwidth of the trunk links. With these assumptions, the maximum terminal port flow can be stated as a set of linear inequalities expressing a multicommodity flow problem. One set of inequalities constrains the amount of traffic on trunk links to lie below the trunk link capacity. Another set of inequalities reflects the traffic injected into and ejected from the fabric, for each particular destination. The resulting set of inequalities can then be solved by a linear programing solver in polynomial time. The resulting maximum concurrent throughput is what we report as the throughput of the topology. This value is a theoretical upper bound on the traffic supported assuming a uniform random traffic flow and infinite buffering. Where engineering compromises in the switch decrease performance, the slowdown can be measured against the optimal value.

In general, throughout this paper we concentrate on 16-port switches, since this is the size of the switch we are most concerned with. The techniques and analysis should apply equally well to other sizes of switch. We focused on topologies with twenty-five or fewer switches, for local clusters of computers.

2.2 Regular Topologies

We initially considered ring networks, star networks, complete graphs, two and three dimensional mesh networks, chordal ring networks, and a few special-purpose graphs such as Petersen, the 13-node mesh, and the E3 hexagonal mesh. We allowed trunk links to have arbitrary multiplicity. We also considered simple algebraic operations on graphs, including the Cartesian product of two graphs and induced graphs.

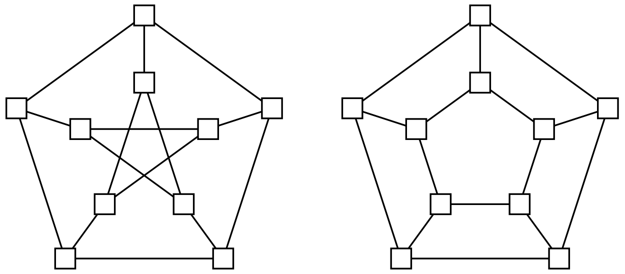

Figure 1: Petersen graph (on left) and a 2×5 wrapped mesh.

Given the range of parameters above, even with this small subset of relatively symmetrical topologies there exists thousands of topologies to consider.

We define a network graph G to be *better* than another network graph G' if $c = c'$, $S \leq S'$, $N \geq N'$, and $g \geq g'$, and at least one of the inequalities is strict.

For instance, the Petersen graph, pictured in Figure 1, supports 130 terminal nodes at a traffic rate of 15.4% with an average path length of 1.5, using ten switches. The 2×5 wrapped mesh, shown next to it, also supports 130 terminal nodes with ten switches, but only at 12.8% and with an average path length of 1.7. Thus, the Petersen graph is better than the 2×5 wrapped mesh.

The result of our consideration of the regular, symmetrical topologies was a database of topologies, each one appropriate for a particular number of switches, number of nodes, and traffic rate. Using these charts it is easy to determine the "best" network of the set that, for instance, supports 100 terminal nodes at a traffic rate of 30% with the smallest number of 16-port switches. In this case, a ten-switch Petersen network with doubled trunk links would be the best; it would support a maximum of 100 nodes at a traffic rate of 40% and an average path length of 1.5 hops. If only nine switches could be used, but 100 terminal nodes were definitely required, the best traffic rate would be 25%, supplied by a three by three wrapped mesh with single links in all four directions; the average path length would then be 1.33 hops. If we could only afford nine switches but were unwilling to compromise on the traffic rate, then the best network would be the same three by three network, but populated to only 90 terminal nodes.

2.3 Irregular Topologies

The set of optimal symmetrical topologies we considered left our topology database somewhat sparse, with gaps between topologies and large jumps in terminal node count and throughput, shown by the jagged line in Figure 2. To improve this, and to answer the question of how good the symmetrical topologies are, we performed a computer search for high-performance irregular topologies.

We were able to use exhaustive search to find the best performing topologies on all graphs with 11 or fewer nodes. For between 12 and 25 nodes, we performed a non-exhaustive computer search. Figure 2 shows how much the set of irregular topologies improved our results. The resulting curve is surprisingly smooth.

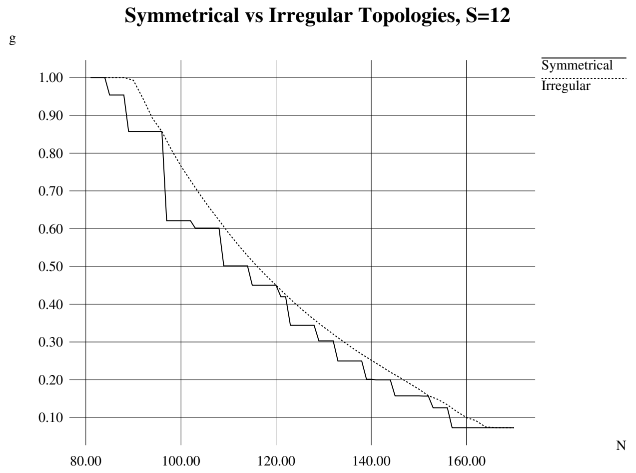

Figure 2: Symmetrical and irregular topologies.

With the topology database extended to the irregular topologies, we generate different answers for the topology questions presented in the previous section. The best network that supports 100 terminal nodes at 30% is an irregular topology requiring 9 switches that yields a throughput of 32.6%. The best 10-switch network that supports 100 terminal nodes supplies a throughput of 50% at an average path length of 1.2 hops, an improvement of 25% over the result supplied by symmetrical topologies.

2.4 Analytic Model

The smoothness of the curve in Figure 2 seemingly implies that the throughput of the optimal topology might fit some mathematical model. In this section, we present the derivation of such an analytical model.

Let us imagine what the best possible topology looks like. Such a topology will have the maximum attainable throughput, minimum average path length, and maximum number of nodes.

Consider bandwidth requirements. The total bandwidth for link traffic on a switch must be the average path length $\bar{p}$ times the traffic rate g times the number of nodes generating traffic. Each link carries traffic in both directions, so overall it carries twice the port bandwidth in traffic, yielding this minimum number of links:

$$L \geq \frac{g\bar{p}N}{2}$$

If the network is connected, then

$$L \geq N - 1.$$

The above equation underscores the importance of the average path length. It is obvious that the average path length is closely tied to the average packet latency; this equation shows that the average path length is also directly related to the attainable throughput. In particular, as the average path length increases, so does the number of ports per switch that need to be assigned to trunk links rather than terminal nodes, so the cost for switches per terminal node rises with the average path length.

2.5 Limits on Average Path Length

We next explore a lower bound on $\bar{p}$. Consider a matrix D such that D_{ij} is the distance in hops between nodes i and j. D_{ii} is always zero, and D_{ij} is 1 if and only if there is at least one trunk link between nodes i and j. All other values of D must be greater than or equal to 2.

For our bound on $\bar{p}$, we shall assume that each trunk link connects a different pair of switches, and that $L \leq \binom{S}{2}$.

Another matrix P contains the number of terminal nodes on each switch. P must satisfy the following equations:

$$N = \sum_{0 \leq i < S} P_i$$

$$2L = \sum_{0 \leq i < S} (c - P_i)$$

The average path length for the topology is

$$\bar{p} = \frac{1}{N^2} \sum_{0 \leq i,j < S} D_{ij} P_i P_j$$

The matrix D contains N zeros and $2L$ ones. The average path length is minimized if all other values are 2, that is, if $p_{max} = 2$. A violation of this would only increase $\bar{p}$, loosening our bound. Nonetheless it is enlightening to explore how realistic this assumption is.

This assumption is valid for $S \leq c + 1$ and for any L that allows a connected topology. We simply use $S - 1$ edges to connect the nodes in a broad single-level tree, and then add the additional nodes any way we wish.

Similarly, a graph formed by the Cartesian product of two complete graphs each with n nodes forms a graph with maximum path length 2, $c \geq 2n - 2$, $S = n^2$, and $L = n^2(n - 1)$. Other constructions yield other points.

On the other hand, it is never possible to build a topology with $S > c^2 + 1$ switches with $p_{max} \leq 2$, since at most c switches are reachable in one hop and $c(c - 1)$ in two.

As we shall show, assuming $p_{max} \leq 2$ leads to a fairly accurate value for $\bar{p}$ for most small optimal fabric topologies with less than a few dozen switches.

Since the only values in the D matrix are 0, 1, and 2, we can express $\bar{p}$ (assuming $p_{max} \leq 2$) as

$$\bar{p} = \frac{1}{N^2} \sum_{0 \leq p \leq 2} \left(p \sum_{\substack{0 \leq i,j < S \\ D_{ij} = p}} P_i P_j \right)$$

It is easiest to work with the values of i and j for which the path length is 0 and 1, since these represent the identity function and adjacency matrix, respectively:

$$\bar{p} = 2 - \frac{1}{N^2} \left(2 \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 0}} P_i P_j + \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} P_i P_j \right)$$

The first sum can be simplified by remembering that $D_{ij} = 0$ only if $i = j$.

$$\bar{p} = 2 - \frac{1}{N^2} \left(2 \sum_{0 \leq i < S} P_i^2 + \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} P_i P_j \right)$$

Next, we wish to separate our path length bound into two parts, one that is independent of and one that is dependent on the distribution of values in P. We define a new matrix P' such that $P_i = P'_i + r$, where $r = N/S$; that is, r is the average number of terminal ports per switch, and P' represents the difference between a regular topology and the hypothetical "optimal" topology. The P' matrix satisfies

$$0 = \sum_{0 \leq i < S} P'_i.$$

Let us first consider the summation concerning the zero hop paths. This expands to

$$2 \left(\sum_{0 \leq i < S} r^2 + 2 \sum_{0 \leq i < S} r P'_i + \sum_{0 \leq i < S} {P'_i}^2 \right).$$

The first term is simply Sr^2. The middle term is zero. So the term for the zero-length paths sums to

$$2 \left(Sr^2 + \sum_{0 \leq i < S} {P'_i}^2 \right).$$

Next we work with the final term representing the distance one pairs. This expands to

$$\sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} r^2 + \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} r(P'_i + P'_j) + \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} P'_i P'_j$$

The first term sums to $2Lr^2$. The second term is symmetrical and can be rewritten

$$2r \sum_{\substack{0 \leq i,j < S \\ D_{ij} = 1}} P'_i$$

If each trunk link connects a different pair of switches, then there are $c - P_i$ values of j for which $D_{ij} = 1$, so the above equation becomes

$$2r \sum_{0 \le i < S} (c - P_i) P_i'$$

Replacing the P_i with $P_i' + r$ yields

$$2r \sum_{0 \le i < S} (c - r - P_i') P_i'$$

Since the sum of the entries of the P' matrix is zero, this simplifies to

$$-2r \sum_{0 \le i < S} P_i'^2$$

For the last term, expand $ab = (a^2 + b^2 - (a-b)^2)/2$:

$$\frac{1}{2}\left(\sum_{\substack{0 \le i,j < S \\ D_{ij}=1}} P_i'^2 + \sum_{\substack{0 \le i,j < S \\ D_{ij}=1}} P_j'^2 - \sum_{\substack{0 \le i,j < S \\ D_{ij}=1}} (P_i' - P_j')^2 \right)$$

The first two are symmetrical, and for each there are $c - P_i$ values of the index for which $D_{ij} = 1$:

$$\sum_{0 \le i < S} (c - r - P_i') P_i'^2 - \frac{1}{2} \sum_{\substack{0 \le i,j < S \\ D_{ij}=1}} (P_i' - P_j')^2$$

Adding the terms results in the following expression for $\bar{p}$ in terms of S, L, c, r, D, N, and P':

$$\bar{p} = 2 - \frac{1}{N^2}\Big(2Sr^2 + 2 \sum_{0 \le i < S} P_i'^2 + 2Lr^2 - 2r \sum_{0 \le i < S} P_i'^2 + \sum_{0 \le i < S} (c - r - P_i') P_i'^2 - \frac{1}{2} \sum_{\substack{0 \le i,j < S \\ D_{ij}=1}} (P_i' - P_j')^2\Big)$$

Only the last term mentions D. Since we are working with the optimal topology, we assume we can choose D to fit P in such a way that this term is minimized; we do this by connecting nodes with the same number of terminal nodes, as much as possible, perhaps by switching edges. By doing so we make the last term negligible. In addition, since we are calculating a lower bound for $\bar{p}$, and this term can only increase $\bar{p}$, at most we sacrifice some tightness. Note that for regular topologies this last term is always zero.

Collecting the remaining terms yields

$$\bar{p} \ge 2 - \frac{2}{S} - \frac{2L}{S^2} + \frac{1}{N^2} \sum_{0 \le i < S} (3r + P_i' - c - 2) P_i'^2$$

In this equation, the first (analytic) terms represent the "continuous" portion of $\bar{p}$ and the last term represents the quantization effects. Typically this term is very small. For many topologies of interest, such as those for which $2r > c$ (most ports in the network are connected to terminal nodes rather than trunk links), the term is always positive and thus just tightens our bound on $\bar{p}$. A simple useful lower bound just uses the continuous portion; a more accurate but more complex lower bound includes the quantization effects for a tighter match.

If $L > \binom{S}{2}$, the above bound is not tight because we can build a complete graph (all nodes connected to all other nodes) and still have edges left over. This is not a problem for throughput estimation because the cut bound (presented below) will subsume the $\bar{p}$ limit in this interval.

Given a lower bound on $\bar{p}$, we can convert this to an upper bound on g because

$$g \le \frac{2L}{\bar{p}N}$$

Using the continuous value for a limit on $\bar{p}$, this gives us

$$g \le \frac{(cS - N)S^2}{N(2S^2 - (2+c)S + N)}$$

It turns out that just the continuous portion is a remarkably tight match to the empirical results.

2.6 Cut Size Limits

Another limit on the performance of a particular topology is the size of cuts. A cut is a set of edges whose removal partitions the graph.

If a topology has a cut of size s that partitions the graph into two parts, one with n terminal nodes and the other with $(N - n)$ terminal nodes, this means that all traffic between these two sets of nodes must flow through the edges of the cut. This yields a performance bound on g based on this cut of

$$g \le \frac{sN}{n(N-n)}.$$

One required cut that often leads to tighter bound on the topology throughput is that of the node with the smallest degree. A graph with S switches and L trunk links must have at least one switch of degree $m = \lfloor \frac{2L}{S} \rfloor$. This cut leads to a performance bound of

$$g \le \frac{N\left(c - \lceil \frac{N}{S} \rceil\right)}{\lceil \frac{N}{S} \rceil \left(N - \lceil \frac{N}{S} \rceil\right)}$$

which is tighter than the $\bar{p}$ limit on performance for some values of S, c, and N.

Figure 3 shows the two components, both the path length limit and the cut bound, to the analytical performance bound for $S = 7$ and $S = 15$ and $c = 16$ for different values of N. (Note that the jaggedness in the cut size bound is unrelated to the jaggedness in the empirical symmetrical topology throughput in Figure 2.) The overall analytic bound is the minimum of these two. For $S = 7$, both components are

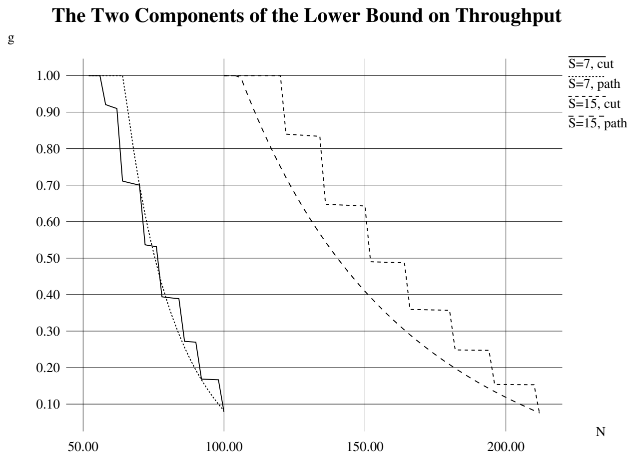

Figure 3: The two components of the analytical performance bound for $S = 7$ and $S = 15$.

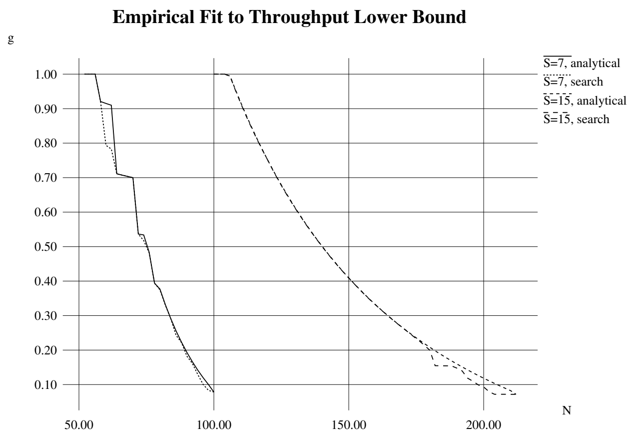

Figure 4: The analytical performance bound versus the best-known irregular topology for $S = 7$ and $S = 15$.

essential; for $S = 15$, the path length bound is usually the tighter one.

Figure 4 shows how close actual irregular topologies approach the bounds. For $S = 7$, there are only a few values of N for which there is any discernable difference. For $S = 15$, the actual irregular topologies are extremely close for $g > 0.2$; topologies with throughput less than 20% are probably of little interest anyway. In addition, since we did not perform exhaustive search for $S = 15$ because of the sheer number of topologies with that many switches, we cannot be sure that irregular topologies with higher throughput do not exist.

3 Deadlock

When choosing a topology for cascading Fibre Channel switches, another critical aspect to consider is deadlock.

To illustrate the deadlock problem and its possible solution, consider the envelope topology shown in Figure 5. By deadlock we mean the situation when some cycle of trunk link buffers are filled with packets that need to be transferred forward in the cycle. Since all buffers are full, no individual packet can be forwarded, causing deadlock. If the paths between nodes 1, 2, 3, and 4 are routed in a clockwise direction, the envelope topology can deadlock. In simulation experiments, we find that even when using completely random traffic streams, deadlock occurs very rapidly.

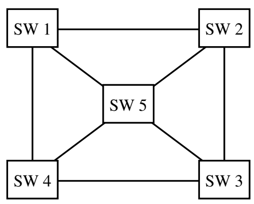

Figure 5: The envelope configuration

This situation can be corrected by using a different routing scheme. If we change the routing so that all the packets with destinations 2 hops away are routed via central switch 5, then the routing scheme is deadlock free. Unfortunately, this routing also tends to concentrate traffic in the links connected to switch 5, so they become the bottleneck prematurely; thus, this is an *unbalanced* routing. It is possible to derive a deadlock-free, balanced routing for this topology; we call such a routing scheme "Smart Routing". It turns out that, for most optimal network topologies, it is possible to find a routing scheme that generates very nearly the performance of the best routing that ignores the possibility of deadlock.

In Fibre Channel switches, buffers are associated with input ports, and thus can be considered to be associated with directed edges between the switches connected switches. It is a cycle of these buffers that cause deadlock in Fibre Channel switches; this is in contrast to other types of networks, where a centralized buffer pool causes a different set of deadlocks to occur.

We do not consider deadlock-avoiding strategies based on buffer classes [Gun81] or virtual channels [DS87] since Fibre Channel supports neither.

A *routing* is a set of paths, such that there is at least one path between each source and destination node. The *complete* routing contains every path that includes a node no more than once. The distance between two nodes in a routing is the length of the shortest path between those nodes that is in the routing. A *minimal* routing has each component path of the minimal length between its particular source and destination; it is *non-minimal* if some path is longer. The complete minimal routing contains every minimal path and no other paths.

A routing imposes *dependencies* between input buffers and thus directed links; there is a dependency between two input buffers if some path contains the two input buffers in succes-

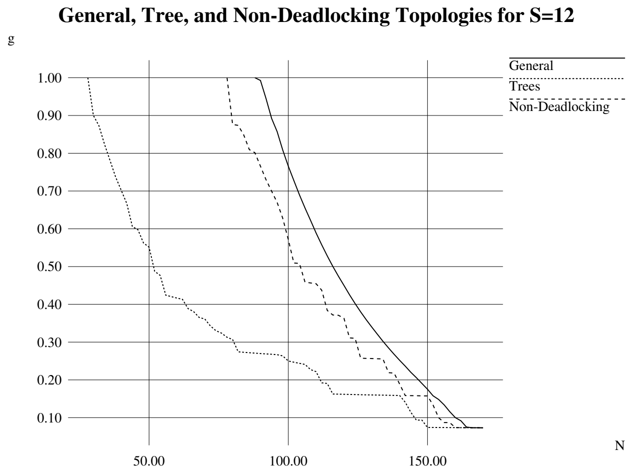

Figure 6: The throughput of the best possible tree, non-deadlocking, and optimal topologies.

sion.

Input buffers connected to terminal ports (injection buffers) can never participate in deadlock because no other Fabric buffer can be waiting on these buffers; thus we can ignore these buffers when considering deadlock.

If the dependency graph on input buffers that is imposed by a particular routing is cycle-free, that routing is said to be *deadlock-free*.

One-hop paths can never contribute to deadlock because they never introduce buffer dependencies; thus, all topologies for which the maximum path length is one (complete topologies) have a deadlock-free minimal routing.

The throughput of a topology under a routing is the maximum throughput supported by that topology when the paths are restricted to the given routing.

3.1 Avoiding Deadlocking Topologies

Many topologies simply do not deadlock under "reasonable" routings. Tree topologies can never deadlock, since paths are restricted to never visit the same node twice, and thus there is no way to construct a cycle of paths. Thus, one possible solution to the deadlock problem is to only consider tree topologies.

Unfortunately, tree topologies have much lower throughput than general topologies. The leftmost line in Figure 6 shows the throughput of the optimal tree topologies as well as the throughput of the optimal general topologies for $S = 12$.

We call all topologies that do not deadlock under the complete minimal routing *non-deadlocking* topologies. Note that non-minimal routings exist for which these topologies deadlock. The topologies for which the complete minimal routing does not deadlock can be characterized by the fact that they do not contain any chord-free cycles with four or more switches. A chord-free cycle is a connected cycle of nodes such that every edge in the original topology that connects two of the nodes in the cycle lies on the cycle. A proof of this theorem is based on the idea that a deadlocking cycle in a minimal routing requires a non-chorded cycle of length four or more in the network topology; we omit the proof for brevity. An example of a non-deadlocking topology is shown in Figure 7.

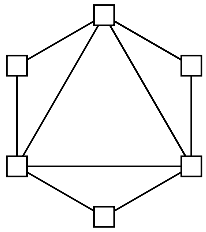

Figure 7: An example of a non-deadlocking cyclic topology.

Non-deadlocking topologies are also restricted in throughput, although not so badly as trees. The middle line in Figure 6 shows the throughput attained by the best possible non-deadlocking topologies; it is clear that they are significantly better than trees, although still much worse than general topologies. As an example, using 12 switches and 100 terminal nodes, the best tree topology supports only 25% throughput; the best non-deadlocking topology supports 57% throughput, and the best general topology supports 77% throughput.

In addition to performance, there are several reasons why we cannot simply restrict the class of topologies. First, the routing algorithm may not be able to specify how a network is wired; rather, it may be required to function correctly and as good as possible given an arbitrary connection between switches. Second, even if the original network topology is restricted, switch and link failures can yield a topology that deadlocks. Thus, an algorithm that takes an arbitrary irregular topology and finds a deadlock-free routing with the best throughput is important.

Two additional classes of topologies are worthy of note. One class is the set of topologies that has some minimal routing that is deadlock free. The ring of four switches (4-cycle) (or any unwrapped n-dimensional mesh) is an example of such a topology; the classic dimension-ordered or e-cube [SB77] routings are minimal and deadlock-free. Other topologies have no minimal routing that is deadlock free; for instance, the 5-cycle (or any wrapped n-dimensional mesh, where the size of some dimension is five or greater) has no deadlock-free minimal routing; for these graphs, every deadlock-free routing has a larger average path length than the original graph.

Every topology has some deadlock-free routing. One way to find such a routing is to remove edges until the resulting graph is a tree or deadlock-free topology; then, construct a routing using only those edges.

Finding a deadlock-free routing with the maximum throughput for an irregular topology is a difficult problem, because the number of deadlock-free routings is astronomical, even for a relatively small graph. In the next two sections we discuss techniques to find a good deadlock-free routing for an arbitrary irregular topology. Our contribution is such a technique that works much better than the best technique known in the literature.

3.2 Up*/down* routing

Up*/down* routing [OK92] is the only general technique in the literature for finding a deadlock-free routing for switches with edge buffers on general irregular topologies. The procedure is straightforward. A root node is chosen at random. Edges that go closer to the root node are marked as "up" edges; edges that go farther away are marked as "down" edges. Edges that remain the same distance from the root note are ordered by a randomly assigned node ID.

The routing consists of all paths that use zero or more "up" edges followed by zero or more "down" edges; that is, all paths that can be described by the regular expression up* down*. The proof that the routing includes a path between all nodes and that the resulting buffer dependency graph is acyclic is straightforward and omitted here.

This technique has some good properties. Every length one path is legal. The maximum distance between any two nodes in the resulting routing is equal to twice the maximum distance from the root node to its farthest node. Finally, calculating an up*/down* routing requires time proportional to the number of edges in the original topology; balancing the resulting routing dominates the solution time.

Up*/down* is a randomized algorithm, both in the initial selection of the root node and the 'tie-breaker' by arbitrarily assigned node IDs. Thus, it can be run multiple times, and the routing with the maximum throughput retained.

Unfortunately, up*/down* routing tends to concentrate traffic in the node chosen as the root. For most topologies, each node must handle traffic it receives and generates, as well as "transit" traffic through the node. In up*/down* routing, a node at distance k from the root can handle no transit traffic that comes from and is destined for nodes at lesser distances. This usually makes it difficult to balance traffic well.

For instance, consider the 4-cycle. The node at distance two from the root cannot carry any traffic between the two nodes at distance one; all such traffic must flow through the root. This restriction means that any up*/down* routing in the 4-cycle has a maximum of 80% of the throughput of the complete routing. Figure 8 shows the results of running up*/down* on our set of optimal 12-node topologies. The value reported is the maximum of ten trials with up*/down*. For instance, for 100 terminal nodes, up*/down* yields a maximum throughput of 67%, compared to the best general topology of 77% and the best non-deadlocking topology throughput of 57%. This was the best of ten trials at finding a routing; the average trial was even worse than this. Generally, up*/down* routing (with multiple trials and selection of the best solution) generates better results than using non-deadlocking topologies, but still significantly worse than if deadlock were not a problem.

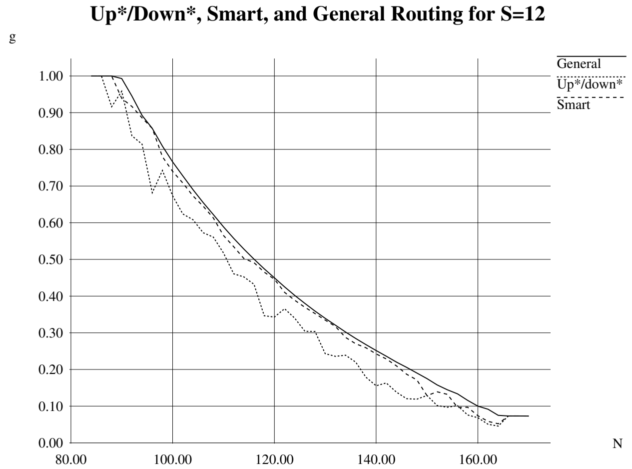

Figure 8: A comparison of up*/down* and Smart Routing.

3.3 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 multicommodity flow problem, it seems reasonable to use a more complex deadlock-free routing algorithm that does not suffer from the root-node congestion problem.

Smart routing is our attempt at this. Rather than break the buffer cycles by arbitrarily picking a root node and performing a search from that node, we instead build an explicit buffer dependency graph and search it for cycles. For each cycle, we 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 multicommodity flow solution, putting this in the inner loop of the routing search would be prohibitively costly. Instead, we use a much simpler heuristic: the average path length. A secondary heuristic attempts to distribute the cuts among the various switches in the topology.

3.4 Implementation

Because each path in our routing connects adjacent edges, and each buffer is associated with an edge, there can only be buffer dependencies of the form (i, j) depends on (j, k), where i, j, and k are switch indices. Since our adjacency matrix is relatively dense (in order to get good throughput), we represent the buffer dependency graph by a three-dimensional array $C(i, j, k)$ representing the buffer dependency between (i, j) and (j, k). If E is the number of edges in the topology, then the number of nodes in the buffer dependency graph is $2E$ and the number of edges is initially

$$\sum_{0 \leq i < S} {P_i}^2$$

which is bounded by S^3 and by Sc^2. Searching the buffer dependency graph for a cycle takes, at most, S^3 time.

We can eliminate many initial cycles in the buffer dependency graph by simply disallowing any path to use the reverse edge immediately after using a particular edge; that is, we initialize all entries of the form $C(i, j, i)$ to CUT. This has no effect on our average path length or the throughput of the topology.

Note that a routing has an path length just as a topology does, and is calculated the same way; the D matrix contains the lengths of the shortest paths between pairs of nodes that is allowed by the routing. This $\bar{p}$ calculation forms the inner loop in the smart routing algorithm, and thus should be relatively fast.

Because of our cuts, we can no longer calculate $\bar{p}$ using Floyd's or Djikstra's algorithm, since the existence of a path from i to j and another from j to k does not imply the existence of a path from i to k through j. Instead, we need to calculate the distance from any given source to all destinations (or vice versa) using a breadth-first search, which potentially takes an amount of time proportional to the number of edges in our buffer dependency graph, which is bounded by S^3, but typically on the order of S. Since we need to take S such path lengths for the entire $\bar{p}$ calculation, the overall $\bar{p}$ calculation takes time bounded by S^4 but typically S^2.

We optimize it by calculating $\bar{p}$ incrementally, given a particular set of changes to the graph. The only incremental change we currently support is the cut of an arc in the buffer dependency graph. As we calculate $\bar{p}$, we keep track of which cuts are used for paths from what sources. When we cut an arc in the dependency graph, we only recalculate the path length for those sources that used the particular arc we cut. Since the number of such necessary cuts is bounded by $S^2(\bar{p} - 1)$, and since $\bar{p}$ is usually less than two for most graphs we are concerned with, most cuts are not on the current $\bar{p}$ tree, so most of the time $\bar{p}$ remains unchanged and need not be recalculated. When it does need to be recalculated, typically only one source is affected, and then only time bounded by S^3 but typically proportional to S is required.

Each time we find a cycle, we consider all possible cuts in order, and determine the one that maximizes our heuristic function. If there are more than one with precisely the same value for the heuristic function, we choose one at random.

The time to find a cycle is bounded by the maximum number of edges in the dependency graph, which is bounded by S^3; the resulting cycle may have as many as S^2 buffers in it, although typically it has fewer than S, and we may have to repeat the search on the order of S^3 times, since often a substantial fraction of the dependencies must be removed to make the dependency graph acyclic. Thus, overall, the run-time is bounded by S^9, although more typically it is S^4. Empirically, the time to solve the multicommodity flow problem to balance the routing is an order of magnitude longer than that to find the deadlock-free routing.

This procedure is not guaranteed to yield a deadlock-free routing. In the course of breaking the cycles in the buffer dependency graph, it is possible to have made a poor selection of cuts such that a cycle is encountered where breaking any link will cause one node to not be reachable from another node. In practice this rarely happens, but when it does, the algorithm restarts itself in tree mode.

In tree mode, which is normally false, a tree subset of the original trunk links in the topology is selected as the "backbone"; all dependencies between connecting trunk links in the backbone are marked as essential and will never be broken. This guarantees some path between each pair of nodes. This backbone is found through breadth-first search from a randomly chosen root node to make it shallow. In addition, since the tree can contain no buffer dependency cycles, every buffer dependency cycle can be broken without breaking any essential dependencies.

This greedy cycle-breaking technique tends to break many cycles multiple times, and thus at the end some of the dependencies that were removed can be added back in without introducing cycles. Adding such cuts back in helps maximize the number of paths that can be taken between nodes, allowing the topology to be balanced more closely.

Thus, as each break is made, its expense is calculated as the change it made in the heuristic value. After all cycles have been eliminated, the breaks are considered most expensive first, and those that do not reintroduce cycles are restored to the buffer dependency graph.

Since this is a randomized algorithm, it is typically run several times and the best solution is taken. Figure 8 shows the results applied to the set of topologies with 12 switches; as can be seen, it generally yields results very close to the general throughput, and significantly better than up*/down*.

We compared up*/down* and Smart Routing for our set of generally optimal topologies with twelve or fewer switch nodes. Figure 9, shows a compilation of the results for the topologies we have tried. We tried 248 different topologies. The results for both deadlock-free routing techniques were compiled and sorted in order of increasing efficiency. (The

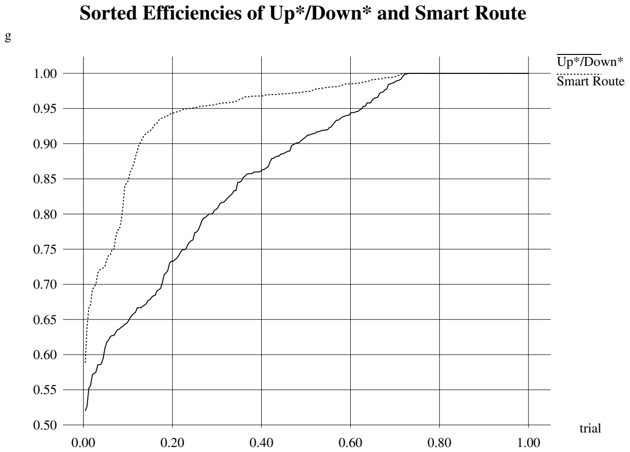

Figure 9: The sorted efficiency ratings of up*/down* and Smart Routing for 248 generally optimal topologies.

efficiency is the fraction of the general, deadlock-ignoring routing that the routing obtained.) From this graph we can easily see that 48% of the topologies had an efficiency less than 90% using up*/down* routing, while only 12% of the topologies did for Smart Routing. Similarly, 63% of the topologies had an efficiency less than 95% with up*/down*, but only 24% did with Smart Routing. Up*/down* averaged 86.8% efficiency, while Smart Routing averaged 95.1% efficiency. When examining the cases for 12 switches, up*/down* averaged 79.7% efficiency, while Smart Routing averaged 93.5% efficiency.

Since Smart Routing attempts to minimize the number of cuts and their impact, it can also be used in those networks that allow table-based adaptive routing.

4 Conclusion

In this paper, we present an analytical model to establish both a theoretical upper bound on fabric performance over the range of topologies and to propose a method for practical evaluation of interconnect topologies. This model allows us to calculate the number of terminal ports, average path length, and maximum traffic supported by a given particular network graph in a simple and straightforward manner. We also presented the results of a computer search for high-performance irregular topologies.

We also presented and analyzed an algorithm for finding deadlock-free routings in arbitrary topologies. We tested the algorithm on the topologies discovered by our computer search and found that, in general, it would find a deadlock-free routing with a throughput very close to that of the unrestricted routing.

There are several ways these results can be extended and applied. One particularly interesting possibility is to take into account non-uniform traffic patterns. Given either predicted or measured traffic distribution information, all of the algorithms in this paper can be easily adapted to generate better topologies and deadlock-free routings by simply replacing the term $P_i P_j$ in the $\bar{p}$ definition with some estimate of the actual traffic. Since Fibre Channel switches generally accumulate statistics as they run, this information could allow periodic network routing reconfiguration that improves throughput and latency. The effect of this on Smart Routing will be that any deadlock-preventing cuts will generally be made to paths that carry little traffic, thus minimizing the effects of deadlock-free routing even further.

References

[AK89] Akers, Sheldon B. and Krishnamurthy, Balakrishnan: A Group-Theoretic Model for Symmetric Interconnection Networks. IEEE Transactions on Computers, Vol. 38, No. 4, April 1989, pp. 555–566.

[AJ75] Anderson, George A. and Jensen, E. Douglas: Computer Interconnection Structions: Taxonomy, Characteristics, and Examples. Computing Serveys, Vol. 7, No. 4, December 1978, pp. 197–213.

[DS87] W.J. Dally and C.L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. IEEE Transactions on Computers, vol. C-36, No. 5, May 1987.

[Gun81] Klaus D. Günther. Prevention of Deadlocks in Packet-Switched Data Transport Systems. IEEE Transactions on Communications, vol. COM-29, No. 4, April 1981.

[ODH94] Öhring, Sabin R., Das, Sajal K., and Hohndel, Dirk H.: Scalable Interconnection Networks Based on the Petersen Graph. Proceedings of the ISCA International Conference on Parallel and Distributed Computing Systems, October 1994, pp. 581–586.

[OK92] S.S. Owicki and A.R. Karlina. Factors in the Performance of the AN1 Computer Network. Performance Evaluation Review, vol. 20, No. 1, June 1992, pp. 167-180.

[SS88] Saad, Youcef and Schultz, Martin H.: Topological Properties of Hypercubes. IEEE Transactions on Computers, Vol. 37, No. 7, July 1988, pp. 867–872.

[Sen89] Sen, Arunabha: Supercube: An Optimally Fault Tolerant Network Architecture. Acta Informatica 26, pp. 741–748, 1989.

[SB77] H. Sullivan and T. R. Brashkow. A large scale homogeneous machine. Proceedings 4th Annual Symposium on Computer Architecture, 1977, pp. 105–124.