Limited Multi-path Routing on Extended Generalized Fat-trees

Santosh Mahapatra, Xin Yuan, Wickus Nienaber
Department of Computer Science, Florida State University, Tallahassee, FL 32306
{mahapatr,xyuan, nienaber}@cs.fsu.edu

Abstract

We consider a general form of routing, called limited multi-path routing, on extended generalized fat-trees where the number of paths between each pair of processing nodes is a parameter. Existing single-path routing and multi-path routing, where all available paths between a pair of processing nodes can be used for the traffic, are special cases of limited multi-path routing. We propose path calculation heuristics, including *shift-1*, *disjoint*, and *random* for limited multi-path routing on extended generalized fat-trees. All of these heuristics are based on existing single-path routing schemes, work for limited multi-path routing with any given limit of paths between processing nodes, gracefully increase routing performance as the limit increases, and reach optimal when all paths between processing nodes are allowed. Extensive flow-level and flit-level simulation experiments are carried out to study the performance. The results show that the *disjoint* heuristic significantly out-performs the other methods.

1 Introduction

The fat-tree topology has been widely adopted in high performance computing (HPC) clusters and data centers due to its properties such as logarithmic diameter, scalability of bisection-bandwidth, and multiple shortest paths between any pair of processing nodes. To fully exploit the capabilities of fat-trees and maximize user-perceived performance, effective routing schemes must be designed [6].

We consider traffic oblivious routing schemes where the routes are independent of the traffic condition in the network. Existing traffic oblivious routing schemes for fat-trees include single-path routing (e.g. [4, 15]) and multi-path routing [16] where all shortest routes between each pair of processing nodes can be used to route traffic between the pairs. We will call such multi-path routing *unlimited* multi-path routing and use the term path for shortest path. Empirical and theoretical study have shown that single-path routing cannot fully exploit fat-tree capabilities [6, 16] while unlimited multi-path routing can achieve optimal routing for an arbitrary traffic pattern on a special fat-tree [16]. However, for reasonably large networks, the number of paths between a pair of processing nodes can be very large and unlimited multi-path routing will require significant resources, which may not be available. For example, unlimited multi-path routing cannot be supported on many reasonably sized InfiniBand networks due to resource constraints.

Since multi-path routing can achieve better load balancing than single-path routing, it would be desirable to support multi-path routing under various constraints. We propose *limited multi-path routing* for fat-trees where the number of paths between each pair of processing nodes is a parameter. Limited multi-path routing is general in that single-path routing and unlimited multi-path routing are special forms of limited multi-path routing. With limited multi-path routing, different multi-path routing schemes can be deployed for networks with different resource

availability. Although single-path routing and unlimited multi-path routing for fat-trees have been extensively studied, studies pertaining to limited multi-path routing are limited: it is unclear how to select the limited number of routes for each pair to maximize performance. This is the problem we consider in this work.

We propose path calculation heuristics for limited multi-path routing on extended generalized fat-trees, including *shift-1*, *disjoint*, and *random*. The *shift-1* and *disjoint* heuristics are based on the well-known *destination-mod-k* single-path routing scheme [15] while the *random* heuristic utilizes randomization technique. All of these heuristics work for limited multi-path routing with any given limit of paths between processing nodes, gracefully increase routing performance as the limit increases, and reach optimal when all paths between processing nodes are allowed. We perform extensive flow-level and flit-level simulation experiments to study the performance of the proposed schemes on various topologies. Our results indicate that limited multi-path routing performs significantly better than single-path routing even when the number of paths allowed is small (e.g. 2 or 3 paths). In addition, the *disjoint* heuristic significantly out-performs the other methods, which indicates that path selection for limited multi-path routing is important and the *disjoint* heuristic is effective in finding the routes for limited multi-path routing.

The rest of the paper is organized as follows. Section 2 describes the related work. The background of this research is presented in Section 3. Section 4 studies the routing on extended generalized fat-trees and introduces the proposed routing schemes. The results of our performance study are discussed in Section 5. Finally, we conclude the paper in Section 6.

2 Related work

Since the inception of the fat-trees as general purpose interconnection networks [8], many variants of fat-trees have been proposed and studied including the m-port n-trees [10], k-ary n-trees [13], generalized fat-trees (GFT) and extended generalized fat-trees (XGFT) [12]. The XGFT is the most generic: almost all other existing variants of fat-trees can be described as XGFTs. Existing load-balanced traffic oblivious routing schemes for fat-tree networks are limited. Existing single-path routing schemes mainly include random routing where a random path is selected through a top level switch for a source-destination (SD) pair [3, 5], the Source-mod-k routing [9, 12, 15], and the Destination-mod-k routing [4, 5, 10, 15, 17]. The Source-mod-k and the Destination-mod-k routing have been shown to have negligible difference in performance [15]. Analyses performed on Destination-mod-k routing have concluded that it is better than random routing [7] and some adaptive routing [4, 15]. The single-path routing scheme in [16] can only apply to a subset of XGFT. The unlimited multi-path routing scheme for m-port n-trees has also been investigated [16]. All existing routing schemes are either single-path or unlimited multi-path. Study related to limited multi-path routing is limited: the methods to compute paths for limited multi-path routing have not been developed; the performance impact of limited multi-path routing is not understood. These are the issues that we investigate in this work.

3 Background

3.1 Extended generalized fat-trees

The extended generalized fat-tree, $XGFT$, was proposed in [12]. We describe this topology for the completeness of this paper. An extended generalized fat-tree $XGFT(h; m_1, m_2, ..., m_h; w_1, w_2, ..., w_h)$

has $h+1$ levels of nodes with nodes at level 0 (lowest level) only having parents, nodes at level h (highest level) only having children, all other nodes having both parents and children. Each level i node, $0 \leq i \leq h-1$, has w_{i+1} parents; and each level i node, $1 \leq i \leq h$, has m_i children. To connect to other nodes (parents and/or children) in the XGFT, each level i node, $1 \leq i \leq h-1$, has $p_i = w_{i+1} + m_i$ ports, each level 0 node has $p_0 = w_1$ ports and each level h node has $p_h = m_h$ ports. The ports of a node at level i are numbered as $0, 1, \ldots, p_i - 1$ starting at the leftmost upper port spanning across to the rightmost upper port and then continuing at the lower ports starting at the leftmost port and spanning across to the rightmost lower port. An example of port numbering is shown in Figure 2(b). Each level i node, $0 \leq i \leq h-1$, is connected to its w_{i+1} parents through ports $0, 1, \ldots, w_{i+1} - 1$; and each level i node, $1 \leq i \leq h$, is connected to its m_i children through ports $w_{i+1}, w_{i+1}+1, \ldots, w_{i+1}+m_i-1$ except the level h nodes, which are connected to their children through ports $0, 1, \ldots, m_h - 1$. An example of port connectivity is shown in Figure 2(a). Level 0 nodes are processing nodes while nodes in other levels are switches or routers. We will call level 0 nodes *processing nodes* and other nodes *switches*. Level h switches will be called top level switches. $XGFT(h; m_1, m_2, ..., m_h; w_1, w_2, ..., w_h)$ has $\prod_{i=1}^{h} m_i$ processing nodes (level 0 nodes); $\prod_{i=1}^{h} w_i$ top level switches (level h nodes).

$XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$ is constructed recursively as follows. When $h = 0$, $XGFT(0;;)$ consists of one single node. For $h > 0$, $XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$ is formed by using m_h copies of $XGFT(h-1; m_1, ..., m_{h-1}; w_1, ..., w_{h-1})$ and $w_1 \times w_2 \times ... \times w_h$ top level switches. Let us number the $w_1 \times w_2 \times ... \times w_h$ top level switches from 0 to $w_1 \times w_2 \times ... \times w_h - 1$. Similarly, the $w_1 \times w_2 \times ... \times w_{h-1}$ top level switches in the sub- $XGFT(h-1; m_1, ..., m_{h-1}; w_1, ..., w_{h-1})$ are numbered from 0 to $w_1 \times w_2 \times ... \times w_{h-1} - 1$. Then, each of the top level switches, x, $0 \leq x \leq w_1 \times w_2 \times ... \times w_{h-1} - 1$, in the m_h copies of $XGFT(h-1; m_1, m_2, ..., m_{h-1}; w_1, w_2, ..., w_{h-1})$ is connected to top level switches $w_h \times x$ to $w_h \times (x+1) - 1$. At any level $l, 0 \leq l \leq h$, there are $(\prod_{i=l+1}^{h} m_i) \times (\prod_{i=1}^{l} w_i)$ nodes.

Alternatively, we can number the nodes by the $h+1$ digit tuple $(l, a_h, a_{h-1}, ..., a_1)$ where l is the node level; and for all $i, l+1 \leq i \leq h$, $0 \leq a_i < m_i$; and for all $i, 1 \leq i \leq l$, $0 \leq a_i < w_i$. Examples for XGFT is shown in Figure 1. A node $A = (l, a_h, a_{h-1}, ..., a_1)$ at level l connects to node $B = (l+1, b_h, b_{h-1}, ..., b_1)$ at level $l+1$ such that for all $i(\neq l), 0 \leq i < h, a_{i+1} = b_{i+1}$, that is the node labels match at all digits except at the l-th digit, where they may or may not match. An example of node labelling is shown in Figure 2(a). For example node (001) at level 1 connects to nodes (001), (011) at level 2. Next, we will describe some properties of XGFT that will be used later in the paper.

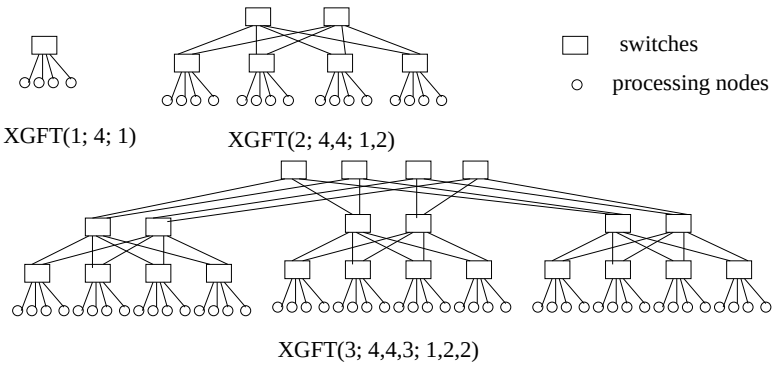
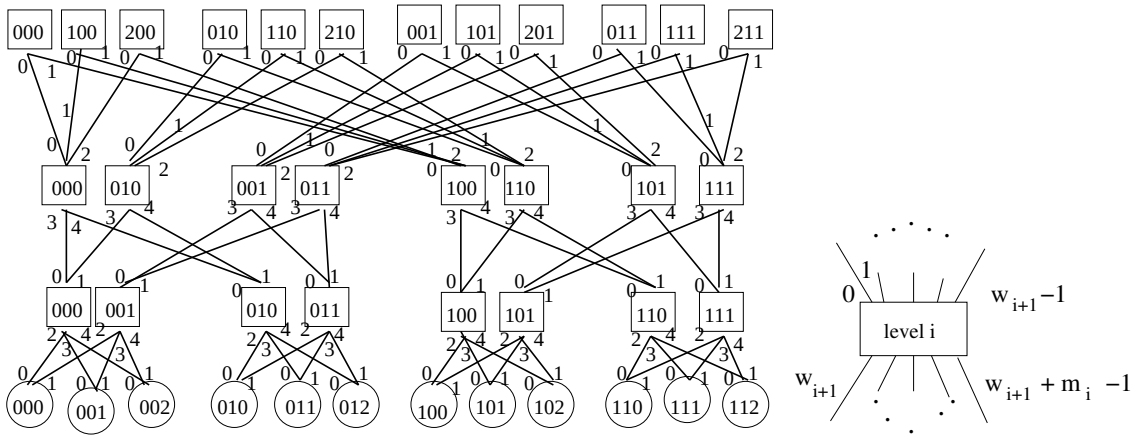


Figure 1: Examples of XGFT



(a) Node labelling xgft(3; 3,2,2; 2,2,3)

(b) Port Numbering

Figure 2: Node labelling on XGFT (3; 3,2,2; 2,2,3) and port numbering

Property 1: Let the Nearest Common Ancesters (NCA) for two processing nodes S and D ($S \neq D$) be at level $k(1 \leq k \leq h)$. There are $\prod_{i=1}^{k} w_i$ different shortest paths between S and D.

If the NCA of processing nodes S and D be at level k, the two nodes are contained in a subtree $XGFT(k; m_1, ..., m_k; w_1, ..., w_k)$. Within this subtree, S can reach D by going up to any of the top level switches and then down to reach D; and there are $\prod_{i=1}^{k} w_i$ top level switches in the sub-XGFT, and thus $\prod_{i=1}^{k} w_i$ shortest paths between the two nodes.

3.2 Routing and its performance metric

Let the system have $N = \prod_{i=1}^{h} m_i$ processing nodes, numbered from 0 to $N-1$. The traffic demand is described by an $N \times N$ Traffic Matrix (TM). Each entry $tm_{i,j}$ in TM, $0 \leq i \leq N-1$, $0 \leq j \leq N-1$ is the amount of traffic from node i to node j.

Routing specifies how the traffic for each source-destination (SD) pair is routed through the network. A multi-path routing scheme is characterized by a set of paths $MP_{i,j} = \{MP_{i,j}^1, ..., MP_{i,j}^{|MP_{i,j}|}\}$ for each SD pair (i, j), and the fraction of the traffic routed through each path $f_{i,j} = \{f_{i,j}^k | k = 1, 2, ..., |MP_{i,j}|\}$ such that $\sum_{k=1}^{|MP_{i,j}|} f_{i,j}^k = 1$. Single-path routing schemes are special multi-path routing with $|MP_{i,j}| = 1$.

For a given traffic matrix, TM, the performance of a routing scheme is measured by the maximum link load. Let $Links$ be the set of all links in the network. For a routing scheme r, the maximum link load for TM is given by

$$MLOAD(r, TM) = \max_{l \in Links} \{ \sum_{i,j,k \ such \ that \ l \in MP_{i,j}^k} tm_{i,j} \times f_{i,j}^k \}.$$

The optimal routing for TM minimizes the maximum link load. Formally, the optimal load for a traffic matrix TM is given by

$$OLOAD(TM) = \min_{\texttt{r is a routing}} \{MLOAD(r, TM)\}.$$

The *performance ratio* of a routing r for TM measures how far r is from being optimal. It is defined as the maximum link load of r divided by the smallest possible maximum link load on TM [1].

$$PERF(r, TM) = \frac{MLOAD(r, TM)}{OLOAD(TM)}.$$

$PERF(r, TM)$ is at least 1. It is exactly 1 if and only if the routing is optimal for TM. The definition of performance ratio of a routing scheme is extended to be with respect to a set of traffic matrices [1]. Let Γ be a set of traffic matrics, the performance ratio of r on Γ is defined as

$$PERF(r, \Gamma) = \max_{TM \in \Gamma} \{PERF(r, TM)\}.$$

When the set Γ includes all possible traffic matrices, the performance ratio is referred to as the *oblivious performance ratio* [1]. The oblivious performance ratio of a routing r is denoted by $PERF(r)$. The oblivious performance ratio is the worst performance ratio that a routing obtains with respect to all traffic matrices. A routing with a minimum oblivious ratio is an optimal oblivious routing scheme and its oblivious ratio is the optimal oblivious ratio of the network.

3.3 Destination-mod-k routing

Existing single-path routing schemes for XGFT in general can be classfied into two classes: source-mod-k (*s-mod-k*) or destination-mod-k (*d-mod-k*). These two routing approaches have similar performance. *d-mod-k* routing is more popular recently since it can be directly realized in InfiniBand [10]. To establish a path from source s to destination d in an $XGFT(h; m_1, m_2, ..., m_h; w_1, w_2, ..., w_h)$ at level k along its upward path to the NCA of s and d, *d-mod-k* chooses a parent node connected to the port identified by $\lfloor \frac{d}{\prod_{i=1}^{k} w_i} \rfloor \bmod w_{k+1}$.

4 Routing on Extended Generalized Fat-trees

Although *d-mod-k* routing and unlimited multi-path routing have been proposed, their performance ratios on XGFT have not been studied. The performance ratios of single-path and unlimited multi-path routing for 2-level and 3-level, m-port n-trees have been established in [16] but the study covers m-port n-trees, a subset of XGFTs, and does not cover XGFT in general. In this section, we will first established the performance ratios of *d-mod-k* routing and unlimited multi-path routing on XGFT. The results show (1) that routing with optimal performance ratio can be achieved with unlimited multi-path routing on XGFT and (2) that the performance ratio for *d-mod-k* is poor for certain type of XGFTs. We will then present several limited multi-path routing schemes of which single-path routing and unlimited multi-path routing are special cases. Limited multi-path routing allows routing performance and resource requirement in between single-path routing and unlimited multi-path routing to be achieved. Three limited multi-path routing heuristics are proposed: *shift-1*, *disjoint*, and *random*. The *shift-1* and *disjoint* heuristics are based on *d-mod-k* routing while the *random* heuristic is based on randomization technique. All schemes gracefully improve the routing performance as the number of paths allowed for each SD pair increases and achieve optimal routing on XGFT when the number of paths allowed is the maximum possible number of paths for the SD pair.

4.1 Performance ratios for single-path and unlimited multi-path routing

$XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$ contains m_h sub-trees $XGFT(h-1; m1, ..., m_{h-1}; w_1, ..., w_{h-1})$, $m_h \times m_{h-1}$ sub-trees $XGFT(h-2; m1, ..., m_{h-2}; w_1, ..., w_{h-2})$, ..., $m_h \times m_{h-1} \times ... \times m_1$ sub-trees $XGFT(0;;)$, which is a single processing node. We will use the notation $ST(k)$ to represent the set of sub-trees of height k $XGFT(k; m1, ..., m_k; w_1, ..., w_k)$'s, st_k to represent one of the sub-trees of height k, and $\overline{st_k}$ to represent the rest of XGFT that are not in st_k.

For a traffic matrix TM, the maximum amount of traffic that goes in or out of a particular sub-tree st_k is given by

$$MT(TM, st_k) = \max\{\sum_{i \in st_k, j \in \overline{st_k}} \{tm_{i,j}\}, \sum_{i \in st_k, j \in \overline{st_k}} \{tm_{j,i}\}\}$$

Let us denote $TL(k) = \prod_{i=1}^{k+1} w_i$ be the number of links connecting st_k to other part of the XGFT. For a TM, let us define

$$ML(TM) = \max_k((\max_{st_k \in ST(k)} MT(TM, st_k))/TL(k)).$$

Lemma 1: For any traffic matrix TM, the optimal load for TM, $OLOAD(TM) \geq ML(TM)$.
Proof: Let st_m be the sub-tree of height m that has the largest $\frac{MT(TM,st_m)}{TL(m)}$ among all sub-trees (of all different heights), that is, $\frac{MT(TM,st_m)}{TL(m)} = ML(TM)$. Since st_m only has $TL(m)$ one-directional links connecting the subtree to the other part of the fat-tree, and the $TL(m)$ links must collectively carry $MT(TM, st_m)$ units of traffic, there exists at least one link whose load must be at least $\frac{MT(TM,st_m)}{TL(m)}$ regardless of the routing algorithm. From the definition of $OLOAD(TM)$, $OLOAD(TM) \geq \frac{MT(TM,st_m)}{TL(m)} = ML(TM)$. □.

Let us consider the unlimited multi-path routing scheme where we can use any number of paths to carry traffic between each SD pair. Consider the following multi-path routing scheme. For each SD pair, let the number of shortest paths in the XGFT between the SD pair be X. The routing evenly distributes the load for the SD pair among the X paths. We will call this routing $UMULTI$.

Theorem 1: For any XGFT, $PERF(UMULTI) = 1$.
Proof: Given any XGFT, let st_k be any sub-tree at level k. Using $UMULTI$, for any SD pair (s, d) such that $s \in st_k$ and $d \in \overline{st_k}$, the traffic will be evenly distributed among the $TL(k)$ links going out of st_k. Similarly, for any SD pair (s, d) such that $s \in \overline{st_k}$ and $d \in st_k$, the traffic will be evenly distributed among the $TL(k)$ incoming links to st_k. Hence, both the incoming and outgoing links will have a load no more than $\frac{MT(TM,st_k)}{TL(k)}$. Hence, $MLOAD(UMULTI, TM) \leq \max_k((\max_{st_k \in ST(k)} MT(TM, st_k))/TL(k)) = ML(TM)$. From Lemma 1, $OLOAD(TM) \geq \frac{MT(TM,st_m)}{TL(m)} = ML(TM)$. Hence,

$$PERF(UMULTI) = max\{\frac{MLOAD(r, TM)}{OLOAD(TM)}\} \leq 1.\square$$

Theorem 2 There exists $XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$ such that $PERF(d-mod-k) \geq \prod_{i=1}^{h} w_i$.
Proof: We will prove this Theorem by constructing such an $XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$ and a traffic pattern TM such that $\frac{MLOAD(d-mod-k,TM)}{OLOAD(TM)} \geq \prod_{i=1}^{h} w_i$. Let us first fix h, m_1, m_2, ..., m_h,

w_1, ..., w_h. Let A be the smallest integer such that $A \times \prod_{i=1}^{h} w_i \geq \prod_{i=1}^{h-1} m_i$. When $\prod_{i=1}^{h} m_i \geq (A+\prod_{i=1}^{h-1} m_i) \times \prod_{i=1}^{h} w_i \geq \prod_{i=1}^{h-1} m_i$, we can construct a traffic pattern that consists of traffic from each processing in the first level $h-1$ subtree (that consists of $\prod_{i=1}^{h-1} m_i$ processing nodes) to one other processing nodes in other level $h-1$ subtrees such that all traffic are concentrated in one outgoing link from the subtree. More specifically, the first level $h-1$ subtree consists of processing nodes number from 0 to $\prod_{i=1}^{h-1} m_i - 1$. The TM consists of traffics in SD pairs $(0, A \times \prod_{i=1}^{h} w_i)$, $(1, (A+1) \times \prod_{i=1}^{h} w_i)$, ..., $(j, (A+j) \times \prod_{i=1}^{h} w_i)$, $(\prod_{i=1}^{h-1} m_i - 1, (A + \prod_{i=1}^{h-1} m_i - 1) \times \prod_{i=1}^{h} w_i), 0 \leq j \leq \prod_{i=1}^{h-1} m_i - 1$. Each SD pair carries 1 unit of traffic. Note that since $A \times \prod_{i=1}^{h} w_i \geq \prod_{i=1}^{h-1} m_h$ and $\prod_{i=1}^{h} m_i \geq (A+\prod_{i=1}^{h-1} m_i) \times \prod_{i=1}^{h} w_i \geq \prod_{i=1}^{h-1} m_i$, all destinations are valid and are in different level $h-1$ subtrees. Since all destination are multiple of $\prod_{i=1}^{h} w_i$, using *d-mod-k* routing, the output port on each switch involved at level k will be port 0 ($\lfloor \frac{d}{\prod_{i=1}^{k} w_i} \rfloor \bmod w_{k+1} = 0$). This leads to the out-going traffic from the subtree to concentrate on the port 0 of the first switch (all traffic goes through one link). Hence, $MLOAD(d-mod-k, TM) = \prod_{i=1}^{h-1} m_i$. From the proof of Theorem 1, we can see that using $UMULTI$, the traffic from the subtree will be evenly distributed to the $\prod_{i=1}^{h} w_i$ outgoing links from the subtree, Hence, $OLOAD(TM) \leq \frac{\prod_{i=1}^{h-1} m_i}{\prod_{i=1}^{h} w_i}$. Thus, $PERF(d-mod-k) = \frac{MLOAD(d-mod-k,TM)}{OLOAD(TM)} \geq \prod_{i=1}^{h} w_i$. □

As can be seen from Theorem 2, *d-mod-k* performs much worse than $UMULTI$. It has been shown that for 2-level and 3-level, m-port n-tree, even the optimal single-path routing schemes have much larger performance ratio than $UMULTI$ [16]. The issue with unlimited multi-path routing is that the number of paths used in the routing algorithm can be too large to be realized for a reasonably large system. For example, for a 3-level fat-tree with 24-port switches (24-port 3-tree) as the one used in the TACC Ranger supercomputer [14]. The topology is equivalent to $XGFT(3; 12, 12, 24; 1, 12, 12)$, the largest number of shortest path between two nodes is $12 \times 12 = 144$. For the InfiniBand network where different paths require different addresses to realize, the address space will run out for the unlimited routing scheme. Hence, it is desirable to develop limited multi-path routing schemes that can achieve better performance than single-path routing while remaining realizable with limited resources.

4.2 Limited multi-path routing on XGFT

Limited multi-path routing uses multiple paths to carry traffic between each SD pair. The number of paths allowed for each SD pair is a parameter and is denoted by K. Our multi-path routing algorithms are built upon a single-path routing scheme. The design objective is that as K increases, the routing performance should gracefully increase and as K reaches $\prod_{i=1}^{h} w_i$, the maximum possible number of paths between each SD pair, the limited multi-path routing achieves optimal performance for all traffic patterns. Since *d-mod-k* routing is a universal single-path routing scheme for XGFT and has been shown to achieve good performance among single-path routing schemes, our limited multi-path routing scheme will build upon the *d-mod-k* routing.

For an XGFT, the set of shortest paths for each SD pair can be easily determined. To compute paths for multi-path routing with K paths, we must select K paths out of the set of all possible shortest paths. For example, *d-mod-k* routing determines one path for each SD pair. In describing the path computation heuristics for multi-path routing, we will use the following simple enumeration of all shortest paths between each SD pair. Let X be the number of shortest paths between two processing nodes: we will number the first path (Path 0) to be the path where the sender goes to the leftmost top level switch in the XGFT and then to the receiver, the second path (Path

1) to be the path where the sender goes second leftmost top level switch and then to the receiver, and the i-th path (Path $i-1$) to be the path where the i-th leftmost top level switch is used. Notice that from each processing node, there is only one path to each of the top level switches. For example, in $XGFT(3;4,4,4;1,4,2)$ in Figure 3, there are $\prod_{i=1}^{3} w_i = 1 \times 4 \times 2 = 8$ paths between SD pair $(0, 63)$ are:

- Path 0: $(0 \rightarrow 64 \rightarrow 80 \rightarrow 96 \rightarrow 92 \rightarrow 79 \rightarrow 63)$
- Path 1: $(0 \rightarrow 64 \rightarrow 80 \rightarrow 97 \rightarrow 92 \rightarrow 79 \rightarrow 63)$
- Path 2: $(0 \rightarrow 64 \rightarrow 81 \rightarrow 98 \rightarrow 93 \rightarrow 79 \rightarrow 63)$
- Path 3: $(0 \rightarrow 64 \rightarrow 81 \rightarrow 99 \rightarrow 93 \rightarrow 79 \rightarrow 63)$
- Path 4: $(0 \rightarrow 64 \rightarrow 82 \rightarrow 100 \rightarrow 94 \rightarrow 79 \rightarrow 63)$
- Path 5: $(0 \rightarrow 64 \rightarrow 82 \rightarrow 101 \rightarrow 94 \rightarrow 79 \rightarrow 63)$
- Path 6: $(0 \rightarrow 64 \rightarrow 83 \rightarrow 102 \rightarrow 95 \rightarrow 79 \rightarrow 63)$
- Path 7: $(0 \rightarrow 64 \rightarrow 83 \rightarrow 103 \rightarrow 95 \rightarrow 79 \rightarrow 63)$

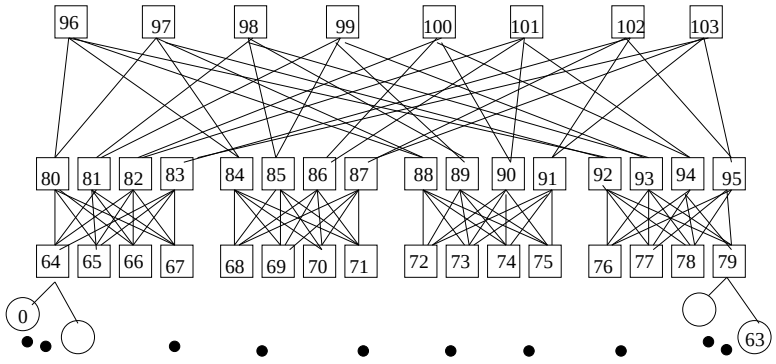


Figure 3: XGFT(3; 4,4,4; 1,4,2)

4.2.1 Random heuristic

Randomization is a powerful technique for routing traffic. For multi-path routing with a maximum of X paths between each SD pair, the random heuristic basically randomly selects a maximum of K paths among all shortest paths for the SD pair. When the total number of different shortest paths is less than K, all of the different paths will be used with the traffic distributed uniformly among the paths. Hence, for $XGFT(h; m_1, ..., m_h; w_1, ..., w_h)$, when $K \geq \prod_{i=1}^{h} w_i$, all shortest paths between each SD pair will be used to carry traffic uniformly: the heuristic becomes $UMULTI$ and is optimal for any traffic matrix. When $K = 1$, it has been shown that random heuristic in general is not as effective as the *d-mod-k* routing [4, 15]. However, this heuristic can serve as the benchmark for other heuristics to compare with.

4.2.2 Shift-1 heuristic

Since the *d-mod-k* routing is known to be effective in routing traffic on XGFT [4, 15], we develop multi-path routing heurstics based on this scheme. The basic idea is to try to create K copies of *d-mod-k* routing with each copy carrying $\frac{1}{K}$ traffic for each SD pair. The first *d-mod-k* based scheme is called the *shift-1* heuristic. The idea is to first enumerate all paths between an SD pair as shown previously and then select K paths built upon *d-mod-k* routing. Let the number of possible paths between an SD pair be X and $ALLPATHS$ be the set of paths for the SD pair. Let us use $ALLPATHS[i]$ to denote Path i for the SD pair. Let *path* be the path computed by the *d-mod-k* routing: *path* is one path in $ALLPATHS$, $path \in ALLPATHS$. Let this path be $ALLPATHS[i]$. The *shift-1* heuristic will use K consecutive paths next to i-th path starting at index i that is $ALLPATHS[i]$, $ALLPATHS[(i+1)\ mod\ X]$, ..., $ALLPATHS[(i+K-1)\ mod\ X]$ to carry traffic for the SD pair.

For illustration, consider the paths between the SD pair $(0, 63)$ in Figure 3. The path *path* computed by *d-mod-k* routing is $(0 \rightarrow 64 \rightarrow 83 \rightarrow 103 \rightarrow 95 \rightarrow 79 \rightarrow 63)$, that is Path 7. Assuming that $K = 3$, the *shift-1* heuristic chooses $K(= 3)$ paths as follows. The first path chosen is *path* at index $i = 7$ (Path 7) in $ALLPATHS$, the second path will be at index $(i + 1)\ mod\ X = (7 + 1)\ mod\ 8 = 0$ and the third path will be at index $(i + 2)\ mod\ X = (7 + 2)\ mod\ 8 = 1$. Therefore we have the $K = 3$ paths as the following: $(0 \rightarrow 64 \rightarrow 83 \rightarrow 103 \rightarrow 95 \rightarrow 79 \rightarrow 63)$ (Path 7), $(0 \rightarrow 64 \rightarrow 80 \rightarrow 96 \rightarrow 92 \rightarrow 79 \rightarrow 63)$ (Path 0), $(0 \rightarrow 64 \rightarrow 80 \rightarrow 97 \rightarrow 92 \rightarrow 79 \rightarrow 63)$ (Path 1).

When $K \geq X$, we will only need to use X paths for the SD pair. For $XGFT(h; m_1, m_2, ..., m_h; w_1, w_2, ..., w_h)$, the largest number of shortest paths for any SD pair is $\prod_{i=1}^{h} w_i$. When $K \geq \prod_{i=1}^{h} w_i$, all of the shortest paths between a SD pair are used and the heuristic becomes $UMULTI$. When K is small, this heuristic basically uses K copies of *d-mod-k* paths (each shift is logically equivalent to one *d-mod-k* routing for all SD pairs).

One issue with this heuristic is that when K is small, the traffic for each SD pair may be spread out only at the top level while there may be contention at the lower levels. For example, for SD pair $(0, 63)$, when $K = 3$, two of the three paths, Path 0 and Path 1, differ only at the top level switch: the links at lower level are shared among the multiple paths. As a result, the lower level links may not be as balanced as the top level links with this heuristic. This limitation motivates another heuristic, which we call the *disjoint* heuristic.

4.2.3 Disjoint heuristic

The key idea of the *disjoint* heuristic is to use a set of size K of *d-mod-k* paths while making the K paths for each SD pair as disjoint as possible. Instead of computing paths by shifting the top level switches in the *shift-1* heuristic, the *disjoint* heuristic shifts the level 1 switches: the links chosen for two different paths for a SD pair at level 1 switches are different leading to different switches at level 2, therefore the two paths for the same SD pair with same lowest level switches are disjoint. For example, for the SD pair $(0, 63)$, Path 7 $(0 \rightarrow 64 \rightarrow 83 \rightarrow 103 \rightarrow 94 \rightarrow 79 \rightarrow 63)$ and Path 1 $(0 \rightarrow 64 \rightarrow 80 \rightarrow 97 \rightarrow 92 \rightarrow 79 \rightarrow 63)$ forks at the level 1 switch and the sub-paths between the level 2 switches $(64 \rightarrow 83 \rightarrow 103 \rightarrow 94 \rightarrow 79)$ and $(64 \rightarrow 80 \rightarrow 97 \rightarrow 92 \rightarrow 79)$ are disjoint.

The *disjoint* heuristic maximizes the use of link-disjoint paths while maintaining the *d-mod-k* routing structure. The heuristic first considers w_1 paths that fork at the lowest level (level 0) processing nodes, then considers the $w_1 \times w_2$ paths that fork at the lowest level switches, then

considers $w_1 \times w_2 \times w_3$ paths that fork at the second lowest level switch. Specifically, let an SD pair with the common ancestor at level k has $X = \prod_{i=1}^{k} w_i$ different shortest paths. Let the *d-mod-k* path be Path i. The heuristic first considers including the w_1 Paths i, $(i + 1 \times \prod_{i=2}^{k} w_i)$ *mod* X, ..., $(i + (w_1 - 1) \times \prod_{i=2}^{k} w_i)$ *mod* X. We will call these w_1 paths *level 1* disjoint paths starting from i. If $K > w_2$, the $w_2 \times w_3$ paths that disjoin at level 2 switches considers the following sequence: *level 1* disjoint paths starting from i, *level 1* disjoint paths starting from $i + 1 \times \prod_{i=3}^{k} w_i$, *level 1* disjoint paths starting from $i + (w_2 - 1) \times \prod_{i=3}^{k} w_i$. These are level 2 disjoint paths starting from i. The higher level disjoint paths can be recursively defined in a similar manner all the way to level k disjoint paths. Notice that the sequence of paths to be included in the multi-path routing is in a close-form and can enumerate easily. Using the disjoint paths not only balances the load at the top level, but also at the lower level and is likely to achieve better load balancing results. Our performance study indicates that the disjoint heuristic significantly out-performs the other two heuristics.

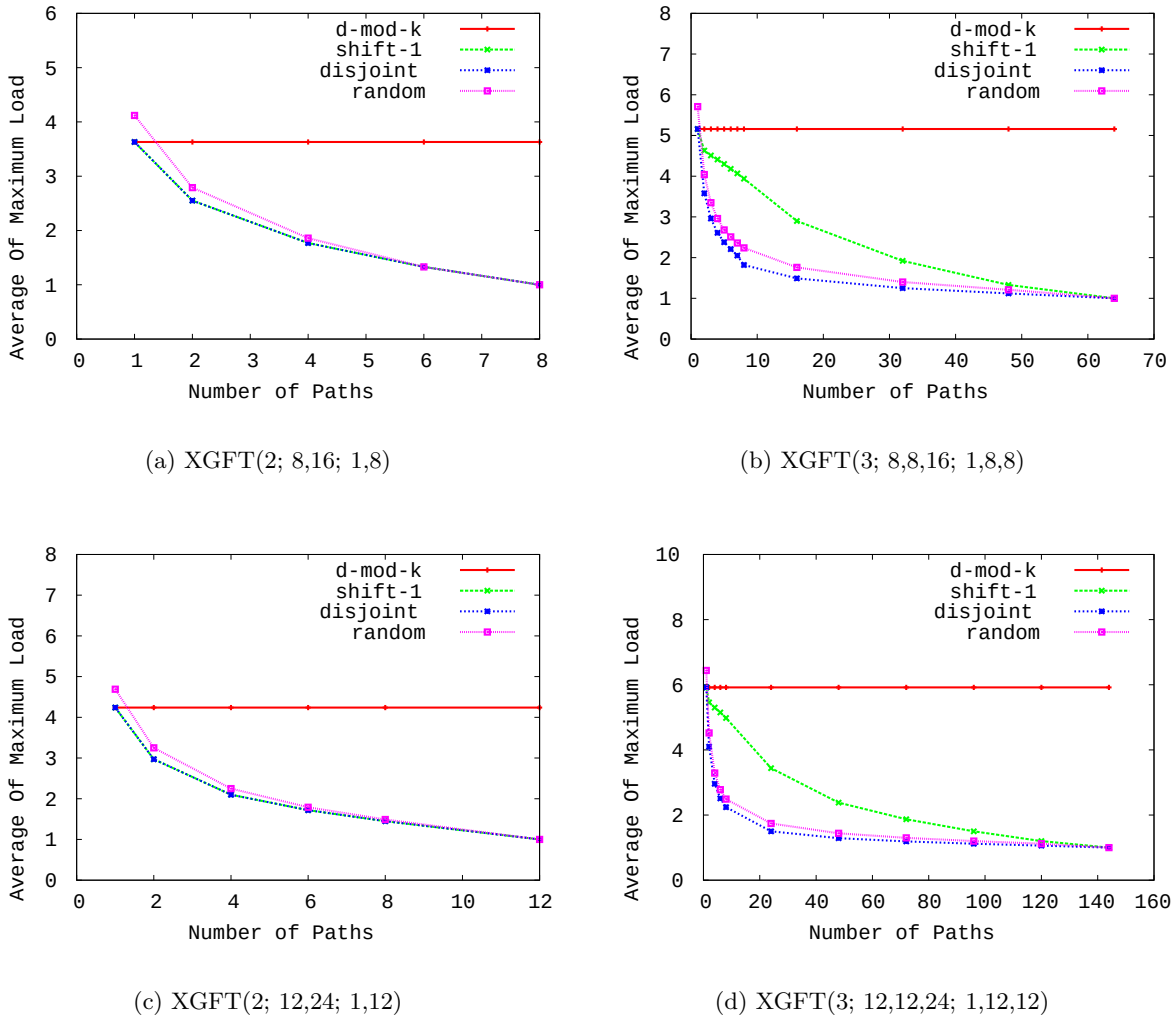
For illustration, consider the paths between SD pair $(0, 63)$ in XGFT(3; 4,4,4; 1,4,2) in Figure 3. In this topology, $w_1 = 1$, $w_2 = 4$, and $w_3 = 2$. There are $\prod_{i=1}^{3} w_i = 1 \times 4 \times 2 = 8$ paths between the two nodes. The exact path is listed previously. The path *path* computed by *d-mod-k* routing is $(0 \rightarrow 64 \rightarrow 83 \rightarrow 103 \rightarrow 95 \rightarrow 79 \rightarrow 63)$ for the SD pair. Clearly $path \in ALLPATHS$ and its index in $ALLPATHS$ is 7. Since $w_1 = 1$, there is no *level 0* disjoint path. The first $w_1 \times w_2 = 1 \times 4 = 4$, *level 2* disjoint paths are Path 7, Path $7 + 1 \times 2$ *mod* $8 = 1$, Path $7 + 2 \times 2$ *mod* $8 = 3$, and Path $7 + 3 \times 2$ *mod* $8 = 5$. By choosing disjoint paths beginning at the lower levels first and gradually moving upwards, the scheme intuitively provides better traffic spreading and overcomes the limitations of the *shift-1* heuristic.

5 Performance study

This section reports our simulation results. We implement a flow-level and flit-level network simulator for the proposed schemes and compare the multi-path routing with single-path routing. In the experiments, we simulate various XGFTs including 3-level trees $XGFT(3; 4, 4, 8; 1, 4, 4)$, $XGFT(3; 8, 8, 16; 1, 8, 8)$ and $XGFT(3; 12, 12, 24; 1, 12, 12)$, which are topologically equivalent to 8-port 3-tree, 16-port 3-tree and 24-port 3-tree respectively [10], and 2-level trees including $XGFT(2; 4, 8; 1, 4)$, $XGFT(2; 8, 16; 1, 8)$ and $XGFT(2; 12, 24; 1, 12)$ that are equivalent to 8-port 2-trees, 16-port 2-trees, and 24-port 2-trees respectively.

Two types of traffic are studied in the experiments: *permutation* traffic where each processing node sends messages to another processing node (possibly itself) and *random uniform* traffic. The performance metrics are maximum link load for flow-level simulation and message delay and maximum aggregate throughput for flit-level simulation. For permutation traffic, we report the results of average permutation performance that is obtained as follows. For each topology and each routing algorithm, we first sample 1000 random permutations and compute the average maximum permutation load among the 1000 random permutations. We then compute the confidence interval with 99% confidence level. If the confidence interval is less than 1% of the average, we stop the simulation and report the average maximum permutation load (the result is deemed to be sufficiently accurate). If the confidence interval is larger than 1% of the average, we double the number of samples and repeat the process until the 1% threshold is reached. For random routing, the results are the average of five random seeds.

Figure 4(a) shows the average maximum link load for permutation traffic using single-path routing and multi-path routing with different K's on 2-level $XGFT(2; 8, 16; 1, 8)$. Results on



(a) XGFT(2; 8,16; 1,8)

(b) XGFT(3; 8,8,16; 1,8,8)

(c) XGFT(2; 12,24; 1,12)

(d) XGFT(3; 12,12,24; 1,12,12)

Figure 4: Average Link Load

other 2-level trees have a similar trend as seen in Figure 4(c). For 2-level trees, the *shift-1* heuristic and the *disjoint* heuristic are identical, as shown in the figure. As can be seen from the figure, multi-path routing with either heuristic *shift-1*, *disjoint*, and *random* gracefully improves the maximum link load as the number of paths per SD pair (K) increases and reaches the optimal when $K = w_1 \times w_2$. Both *d-mod-k* based heuristics are more effective than the *random* scheme when K is small.

Figure 4(d) shows the average maximum link load for permutation traffic using single-path routing and multi-path routing with different K's on 3-level $XGFT(3; 12, 12, 24; 1, 12, 12)$. Results on other 3-level trees have a similar trend as seen in Figure 4(b). For 3-level trees, there is significant difference between *shift-1* heuristic and the *disjoint* heuristic: the *disjoint* heuristic has significant better performance as shown in the figure. In fact, for most K's, *disjoint* is better than *random* which in turn is better than *shift-1*. Basically, link contention at lower level switches are significant for the permutation traffic: *disjoint* and *random* are able to distribute the load more evenly at lower level than *shift-1*. By carefully selecting the paths, *disjoint* is noticeably better

than *random* for all $K < \prod_{i=1}^{h} w_i$. This figure also shows that using multi-path routing, even for a relatively small K, the performance is much better than sinle-path routing. For the large $XGFT(3; 12, 12, 24; 1, 12, 12)$ (24-port 3-tree), with 4 or 8 paths, the maximum link load can be drastically reduced. This shows the effectiveness of limited multi-path routing.

The simulator at the flit-level simulates virtual cut-through (VC) switching to closely resemble Infiniband networks. In the flit-level simulation, we evaluate the performance using *uniform random traffic*, where each source sends traffic to a randomly selected destination node such that each node in the network has an equal probability of being the destination. Since we are evaluating the performance of the routing schemes, we run our simulations using only one virtual channel. The simulator uses credit-base flow-control between switches which implies that a packet is blocked if the destination port does not have available buffer space. We use a packet size of 10 flits and a fixed message size of 10 packets. The message arrival follows *Poisson* distribution with mean value depending on the offered load. The input and output buffers of each of processing nodes and switches have a capacity of 4 packets each. We perform experiments by varying the offered load till the network reaches saturation where the throughput drops sharply and we measure the average throughput and average message delay. We plot the average message delay against the normalized offered load and show the maximum throughput achieved against the normalized offered load in Table 1 under the uniform traffic.

Before we discuss the flit-level simulation results a few finer points about the virtual cut-through switching should provide good insight into the expected results. It is well known that network performance under virtual cut-through dramatically decreases when the network is beyond saturation due to tree saturation [2]. Due to the blocking behavior of such techniques messages get blocked faster thereby augmenting the message delays exponentially when the network is close to saturation. Therefore, the improvement that would be realized due to better routing schemes is in the increased throughput and a higher saturation point in terms of offered load. Under low offered load, the aggregate throughput equals the offered load but beyond saturation, the throughput decreases.

Num-Path	*d-mod-k*	*shift-1*	*random*	*disjoint*
1	49.02	-	38.19	-
2	-	54.88	49.07	61.32
4	-	59.03	59.01	67.65
8	-	67.65	69.75	71.35
16	-	65.30	73.41	76.95

Table 1: Throughput Uniform Traffic XGFT(3; 4,4,8; 1,4,4)

Table 1 shows the maximum throughput achieved with different routing schemes under uniform random traffic. Several observations from the experiments include the following. First, allowing more paths in general increase the maximum throughput for each heuristic. This reaffirms other study that multi-path routing allows resources in the network to be used more effectively. Second, different path selection heuristic can make a significant difference in the performance. For example, with $K = 8$ on topology $XGFT(3; 4, 4, 8; 1, 4, 4)$, *shift-1* achieves a maximum throughput of 67.65%, *random* achieves a maximum throughput of 69.75%, and *disjoint* achieves a maximum throughput of 71.35% under uniform random traffic. When $K = 2$, *disjoint* achieves an improvement of 11.7% over *shift-1*, and 25.0% over *random*. When $K = 4$, *disjoint* achieves an improvement of 14.6% over *shift-1*, and 14.6% over *random*. Third, in terms of message delay, multi-path routing is also significantly better than single-path routing. However, allowing more

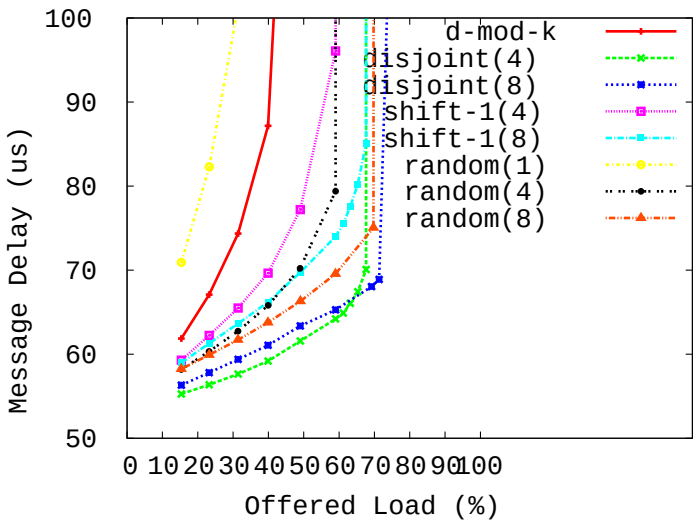


Figure 5: Message Delay Uniform Traffic XGFT(3; 4,4,8; 1,4,4)

paths in general have a better message delay at medium to high load. This is due to the fact that more paths increase maximum throughput. At low load, using more paths does not always imply smaller message delay. With multi-path routing, there are two competing factors that can affect the average message delay. With more paths, the chance of contention increases as traffic is more spread out. However, the penalty for network contention decreases since the traffic is more evenly distributed among the links. Hence, whether increasing the number of paths can improve message delay is not clean. In the experiments, with the *disjoint* heuristic, *disjoint*(4) and *disjoint*(8) both have better delay than *d-mod-k*. But *disjoint*(4) has better delay than *disjoint*(8) at low load. The flit-level simulation results show that for the same K, performance of *disjoint* is better than the other multi-path routing *shift-1* and *random*. This matches the results in the flow-level simulations.

6 Conclusion

We investigate limited multi-path traffic oblivious routing schemes, propose three methods for computing paths for limited multi-path routing, and evaluate the performance through simulation. Our results indicate that limited multi-path routing can significantly reduce link contention and improve message delay and throughput over single-path routing schemes. The results also demonstrate that the path selection method used for limited multi-path routing has significant impact on the effectiveness of limited multi-path routing and that the *disjoint* heuristic is effective in finding paths for limited multi-path routing.

References

[1] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs," *ACM SIGCOMM*, pages 313-324, 2003.

[2] L. Fernández and J. M. Garcia (Spain), "Congestion Control for High Performance Virtual Cut-through Networks", Proceeding (378) Applied Informatics, 2003

[3] J. Flich, M. P. Malumbres, P. Lopez, and J. Duato, Improving Routing Performance in Myrinet Networks, in IEEE IPDPS, pages 27-32, 2000.

[4] C. Gomez, F. Gilabert, M. Gomez, P. Lopez, and J. Duato, Deterministic versus Adaptive Routing in Fat-trees, IEEE IPDPS, 2007

[5] R. I. Greenberg and C. E. Lerserson, Ramdonzied Routing on Fat- trees. In 26th Annual IEEE Symposium on Foundations of Computer Science, pages 241-249, Oct. 1985.

[6] T. Hoefler, T. Schneider, and A. Lumsdain, Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks, IEEE International Conference on Cluster Computing, pages 116-125, 2008.

[7] G. Johnson, D. J. Kerbyson, and M. lang, Optimization of InfiniBand for Scientific Applications, IEEE IPDPS Workshop on Large Scale Parallel Processing (LSPP), 2008.

[8] C.E. Leiserson, "Fat-trees: Universal Networks for hardware efficient supercomputing," IEEE trans. on computers, 34(10):892-901, 1985.

[9] C.E. Leiserson, et. al., The Network Architecture of the Connection Machine CM-5, in Proc. the Fourth Annual ACM Symposium on Parallel Algorithms and ARchitectures, pages 272-285, 1992.

[10] X. Lin, Y. Chung, and T. Huang, A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks. Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS04), p. 11a, Sana Fe, NM, April 2004.

[11] W. Nienaber, S. Mahapatra, and X. Yuan, "Routing Schemes to Optimize Permutation Performance on InfiniBand Interconnects with 2-Level Generalized Fat-tree Topologies," *Technical Report*, Dept. of Computer Science, Florida State University, 2010. Available at http://www.cs.fsu.edu/∼xyuan/paper/103.pdf.

[12] S. R. Ohring, M. Ibel, S. K. Das, and M. Kumar, "On Generalized Fat Trees," *Proceedings of the 9th International Parallel Processing Symposium*, pages 37-44, 1995.

[13] F. Petrini and M. Vanneschi. "K-ary n-trees: High performance networks for massively parallel architectures." Technical report, 1995.

[14] The TACC Ranger supercomputer, http://www.tacc.utexas.edu/resources/hpc.

[15] G. Rodriguez, C. Minkenberg, R. Beivide, R. P. Luijten, J. Labarta, and M. Valero, Oblivious Routing Schemes in Extended Generalized Fat Tree Networks, in Workshop on High Performance Interconnects for Distributed Computing (HPI-DC09) in conjunction with IEEE Cluster 2009, pages 1-8, 2009.

[16] X. Yuan, W. Nienaber, Z. Duan, and R. Melhem, "Oblivious Routing in Fat-Tree Based System Area Networks with Uncertain Traffic Demands," *IEEE/ACM Transactions on Networking*, 17(5):1439-1452, Oct. 2009.

[17] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang, "Optimized InfiniBand Fat-tree Routing for Shift All-to-all Communication Patterns, Concurrency and Computation: Practice and Experience, 22(2):217- 231, Nov. 2009.