Improving Performance of Deterministic Single-path Routing on 2-Level Generalized Fat-trees

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

***Abstract*—This paper focuses on deterministic single-path routing schemes on 2-level generalized fat-trees. We develop a routing algorithm that is optimal in terms of worst-case permutation performance. In comparison to existing routing schemes for such topologies, our algorithm also improves the average performance of common communication patterns including bisect patterns, full permutation patterns, and dissemination (Bruck) patterns on various 2-level generalized fat-trees as demonstrated in our evaluation results.**

***Keywords*-Generalized fat-tree; deterministic routing; single-path routing; permutation pattern;**

I. Introduction

Fat-tree-based interconnects have been widely adopted in commodity high performance computing clusters. Figure 1 (a) illustrates the 2-level generalized fat-tree topology, which belongs to a family of extended generalized fat-trees [8]. This topology consists of two levels of switches and one layer of leaf nodes (processing nodes). There are r bottom level switches and m top-level switches. Each bottom level switch connects to n leaf nodes and each of the m top level switches and is thus an $n + m$-port switch. Each top level switch connects to each of the r bottom level switches and is thus an r-port switch. We will use the notion $T(n + m, r)$ to denote such a 2-level generalized fat-tree. The cross-bisection bandwidth (CBB) ratio [12] of $T(n+m, r)$ is $\frac{m}{n}$: when $m = n$, $T(n+m, r)$ is a *full bisection bandwidth* fat-tree; when $m < n$, $T(n + m, r)$ is a *slimmed* fat-tree; when $m > n$, $T(n + m, r)$ is a *fatted* fat-tree. Examples of full bisection bandwidth, slimmed, and fatted fat-trees are shown in Figures 1 (b), (c), and (d), respectively.

We consider deterministic single-path routing supported by InfiniBand [5]. In this routing scheme, one path is used to carry all traffics from a source to a destination. Existing single-path deterministic routing for generalized fat trees include *Source-mod-k* routing [6], [8], [9] and *Destination-mod-k* routing [3], [9], [12]. As will be shown later, all of these existing routing schemes are not ideal for many common communication patterns. In this paper, we develop a novel single-path routing scheme for 2-level generalized fat-trees that is optimal in terms of worst-case permutation performance. By improving worst-case permutation performance, our algorithm also achieves much higher average-case performance on all of the three types of fat-trees (full bisection bandwidth, slimmed, and fatted) for common communication patterns that are used to evaluate application level performance of interconnection networks, including the bisect patterns [2], the full permutation patterns, and the dissemination (Bruck) patterns [1], [2]. Our evaluation results show that the new routing scheme improves average-case performance of bisection patterns by up to 17.6%, full permutation patterns by up to 57.7%, and dissemination patterns by up to 57.7%.

The rest of the paper is organized as follows. Section 2 introduces the notations. Section 3 analyzes existing routing schemes and presents the proposed algorithm. Section 4 reports the results of our performance evaluation. Finally, Section 5 concludes the paper.

II. Notations

In $T(n + m, r)$, there are m top level switches, r bottom level switches, and $N = r \times n$ processing (leaf) nodes. We will assume that N is an even number. We number the m top level switches from 0 to $m - 1$, the r bottom level switches from 0 to $r - 1$, and the $N = r \times n$ processing (leaf) nodes from 0 to $r \times n - 1$, as shown in Figure 1 (a). We will use the notion TSW_i to denote top level switch i, $0 \leq i \leq m-1$; BSW_i to denote bottom level switch i, $0 \leq i \leq r - 1$; i to denote the processing node i, $0 \leq i \leq r \times n - 1$; and $a \rightarrow b$ to denote the link from a to b, where a and b can be either processing nodes or switches. Clearly, processing node i is directly connected to switch $BSW_{i/n}$.

Let us denote (s, d), $0 \leq s, d \leq N-1$, a source-destination (SD) pair with source node s and destination node d. Let S be a set, $|S|$ is the size of the set. A communication pattern can be represented by a set of SD pairs, $\{(s_1, d_1), (s_2, d_2), ..., \}$.

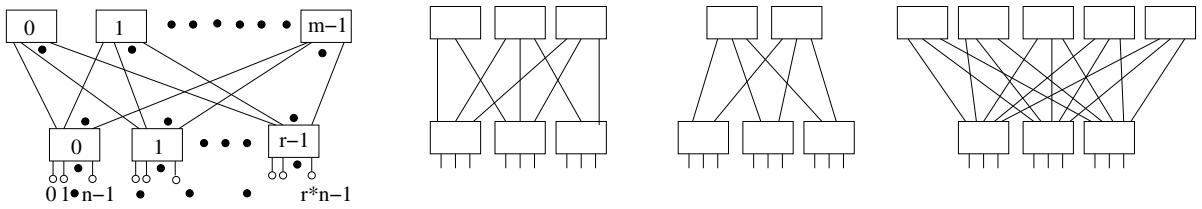
Definition 1: A *permutation* is a communication pattern $P = \{(s_1, d_1), (s_2, d_2), ..., (s_{|P|}, d_{|P|})\}$ such that (1) $s_i \neq s_j$ and $d_i \neq d_j$ for any $1 \leq i \neq j \leq |P|$; and (2) $s_i \neq d_i$ for all $1 \leq i \leq |P|$.

Definition 2: A *bisect* communication pattern [4] is a permutation $P = \{(s_1, d_1), (s_2, d_2), ..., (s_{|P|}, d_{|P|})\}$ such that $|P| = \frac{N}{2}$ and all of the sources and destinations are different.

Definition 3: A *full permutation* communication pattern is a permutation $P = \{(s_1, d_1), (s_2, d_2), ..., (s_{|P|}, d_{|P|})\}$ such that $|P| = N$.

Definition 4: A *dissemination (Bruck)* communication pattern is a permutation $P = \{(s_1, d_1), (s_2, d_2), ..., (s_{|P|}, d_{|P|})\}$ such that (1) $|P| = N$; and (2) if $(s, d) \in P$, then $(d, s) \in P$.

A permutation is a communication pattern where each node can send and receive at most once in the pattern. Not all nodes need to participate in a permutation communication: a



(a) T(n+m, r) (b) full bi. BW T(3+3, 3) (c) slimmed T(3+2, 3) (d) fatted T(3+5, 3)

Fig. 1. 2-level generalized fat-trees

node that is not involved can be considered as communicating with itself, which is not represented in P by Definition 1. A bisect communication pattern is a permutation where half of the nodes are sending to the other half of the nodes; and each node is either a sender or a receiver [2], [4]. Note that we assume N is an even number. The full permutation pattern is an extension of the bisect pattern since each node in modern HPC clusters can send and receive data simultaneously. In a full permutation pattern, each of the nodes is sending to another node and receiving from another node. Finally, the dissemination patterns are used in the Bruck's pair-wise all-to-all algorithm [1]. Definition 4 captures the situation when each logical process can be mapped to any physical node. These communication patterns are used to measure the application level performance for interconnection networks [2], [4], [10].

For an SD pair (s, d), let $p_R^{s,d}$ denote the set of links in the path for (s, d) using a single-path routing scheme R. Let l be a link, and $Links$ be the set of all links in $T(n + m, r)$. For a communication pattern P, $SDSET_R(l) = \{(s, d)|(s, d) \in P \wedge l \in p_R^{s,d}\}$ is the set of SD pairs routed through l using routing algorithm R. The maximum link load for the communication pattern is $ML_R^P = \max_{l \in Links}\{SDSET_R(l)\}$. For a routing algorithm R, we define the worst-case permutation load, $WORST_R$, for all permutation patterns as

$$WORST_R = \max_{P \text{ is a permutation}}\{ML_R^P\}. \quad (1)$$

Let us normalize the link bandwidth to be 1. Since ML_R^P communications in pattern P share a link, the link bandwidth that is available to P is $\frac{1}{ML_R^P}$. We define *average bisect bandwidth* (ABB) to be the average bandwidth for all bisect patterns, that is, $ABB_R = \texttt{average}_{P \text{ is a bisect pattern}}\{\frac{1}{ML_R^P}\}$. Similarly, the *average full permutation bandwidth* ($AFPB$) for routing R is $AFPB_R = \texttt{average}_{P \text{ is a full permutation}}\{\frac{1}{ML_R^P}\}$; and the *average dissemination bandwidth* (ADB) is $ADB_R = \texttt{average}_{P \text{ is a dissemination}}\{\frac{1}{ML_R^P}\}$.

For a system where all nodes are connected by a non-blocking crossbar, $ABB = AFPB = ADB = 1$. For other systems with link contention ($ML_R^P > 1$), ABB, $AFPB$, and ADB are less than 1, representing fractions of the performance of a nonblocking crossbar system. This paper uses the four metrics, $WORST$, ABB, and $AFPB$, and ADB to evaluate routing algorithms on fat-trees. $WORST$ characterizes the worst-case permutation performance while ABB, $AFPB$, and ADB characterize average performance for different communication patterns.

III. Single-path routing in $T(n + m, r)$

In the following, we will first establish the lower bound of $WORST_R$ of any single-path routing scheme R on a 2-level generalized fat-tree $T(n+m, r)$, and show that existing routing algorithms are not ideal for this metric. We will then present our routing scheme, OPT, that achieves optimal $WORST_R$.

Theorem 1 ($WORST_R$ lower bound): Consider $T(n + m, r)$ where $m < n^2$, $r \geq n$, and $(r - 1) \times n > 2m$. Let $\sqrt{m}$ be an integer and n be divisible by $\sqrt{m}$. For any single-path routing algorithm R, $WORST_R \geq \frac{n}{\sqrt{m}}$. □

The proof of this theorem can be found in our technical report (also Theorem 1) [7]. The conditions, $m < n^2$, $r \geq n$, and $(r-1)\times n > 2m$, are quite general since in most practical $T(n + m, r)$, $r \approx n + m$ and $n \approx m$. Almost all common 2-level generalized fat-trees, including slimmed and fatted fat-trees, satisfy the conditions. Examples include $T(8 + 8, 16)$, $T(16+16, 32)$, $T(24+4, 28)$, $T(8+16, 24)$, and all topologies that we use in the evaluation section. When $\sqrt{m}$ is not an integer and/or n is not divisible by $\sqrt{m}$, the lower bound is $WORST_R \geq \lfloor \frac{n}{\lceil \sqrt{m} \rceil} \rfloor$.

A. Worst-case permutation load of existing routing algorithms

Existing deterministic single-path routing algorithms for $T(n + m, r)$ are either the *Source-mod-k* routing (*S-m-k*) [6], [8], [9] or the *Destination-mod-k* routing (*D-m-k*) [3], [9], [12]. A good summary of routing in generalized fat-trees can be found in [9]. Using the switch and node numbering scheme in Section 2, in the *D-m-k* routing, the (s, d) pair where s and d are not in the same switch is routed through top level switch $TSW_{d \bmod m}$. In the *S-m-k* routing, the (s, d) pair is routed through top level switch $TSW_{s \bmod m}$. Since these two routing schemes are fundamentally the same in terms of the routing performance on most $T(n + m, r)$ [9], we will analyze *D-m-k* in this paper. This algorithm tries to balance the link load by spreading the traffic among different links: the traffic from one node to all other nodes are spread out uniformly among all possible links. However, *D-m-k* is not ideal in terms of worst-case permutation performance as shown in the following Theorem.

Theorem 2: Consider using *D-m-k* on $T(n + m, r)$. Let i be the smallest integer such that $i \times m \geq n$. When $\frac{r \times n - i \times m}{m} \geq n$, $WORST_{D-m-k} \geq n$.

Proof: Consider a permutation $P = \{(0, i \times m), (1, (i+1) \times m), ..., (n-1, (i+n-1) \times m)\}$ with n communications. All of the n source nodes are connected to BSW_0. When $\frac{r \times n - i \times m}{m} \geq n$, using ***D-m-k***, all of the n communications in P will use link $BSW_0 \rightarrow TSW_0$. Hence, $ML^P_{D-m-k} \geq n$ and $WORST_{D-m-k} \geq n$. □

Theorem 2 shows that ***D-m-k*** has poor performance in terms of the worst-case permutation load, especially since the optimal $WORST_R$, $\frac{n}{\sqrt{m}}$, can be achieved using our proposed routing algorithm. Notice that the condition $\frac{r \times n - i \times m}{m} \geq n$ in the theorem is quite general. Almost all common $T(n+m, r)$'s where $r \approx n + m$, such as $T(12+12, 24)$, $T(8+8, 16)$, $T(16+16, 16)$, and all of the topologies used in the evaluation section meet the condition.

B. A routing algorithm with the optimal $WORST_R$

Our routing algorithm, called OPT, that achieves optimal $WORST_R$, is depicted in Figure 2. OPT is a generalization of an algorithm designed for m-port n-trees when the traffic pattern is uncertain and changing [11]. To simplified exposition, let us assume that $\sqrt{m}$ is an integer and that $\frac{n}{\sqrt{m}}$ is an integer. The idea is to route SD pairs such that each link carries traffics either from at most $\frac{n}{\sqrt{m}}$ sources or to at most $\frac{n}{\sqrt{m}}$ destinations. When each link carries SD pairs either from at most $\frac{n}{\sqrt{m}}$ sources or to at most $\frac{n}{\sqrt{m}}$ destinations, the maximum link load for any permutation will at most be $\frac{n}{\sqrt{m}}$ since each node can be a source or a destination at most once in a permutation. OPT partitions the n processing nodes in each bottom level switch into $\sqrt{m}$ groups, each group having $\frac{n}{\sqrt{m}}$ nodes. Let the n processing nodes in each switch be numbered from 0 to $n-1$. Nodes 0 to $\frac{n}{\sqrt{m}} - 1$ belong to group 0; ...; Nodes $i \times \frac{n}{\sqrt{m}}$ to $(i+1) \times \frac{n}{\sqrt{m}} - 1$ belong to group i, $0 \leq i \leq \sqrt{m} - 1$. Let us denote $g_i \rightarrow g_j$ as SD pairs from nodes in group i in the one switch to nodes in group j in all other switches. For example, $g_0 \rightarrow g_0$ means the SD pairs from nodes in group 0 in the one switch to all other group 0 nodes in other switches. OPT schedules all SD pairs in $g_i \rightarrow g_j$, $0 \leq i, j \leq \sqrt{m} - 1$, through top level switch $TSW_{i*\sqrt{m}+j}$, that is, $TSW_{i*\sqrt{m}+j}$ carries all SD pairs in $g_i \rightarrow g_j$ for all bottom level switches. Hence, each uplink from a bottom level switch to top level switch $TSW_{i*\sqrt{m}+j}$ carries traffic from one group (group i with $\frac{n}{\sqrt{m}}$ nodes) in the bottom level switch to destinations in group j in all other switches; and each downlink from $TSW_{i*\sqrt{m}+j}$ to a bottom level switch carries traffics to one group (group j) of $\frac{n}{\sqrt{m}}$ nodes in that bottom level switch from sources in group i in all other switches.

Theorem 3: When $\sqrt{m}$ is an integer, and n is a multiple of $\sqrt{m}$. OPT achieves optimal worst-case permutation load for $T(n+m, r)$ with $m < n^2$, $r \geq n$, and $(r-1) \times n > 2m$. $WORST_{OPT} = \frac{n}{\sqrt{m}}$.

The formal proof for this theorem can be found in [7]. Again, the conditions $m < n^2$, $r \geq n$, and $(r-1) \times n > 2m$ are very general: almost all practical $T(n+m, r)$'s with $r \approx n+m$ and $n \approx m$ satisfy the conditions. To achieve the best results,

Algorithm OPT **(route from node** s **to node** d**)**:
Let $k = \lfloor \sqrt{m} \rfloor$;
$src\_switch = s/n$;
$dst\_switch = d/n$;
if ($s == d$) return; /* no need to route */
else if ($src\_switch == dst\_switch$)
  use route: $node\ s \rightarrow BSW_{src\_switch} \rightarrow node\ d$
else { /* must route through a top level switch */
  $grp\_size = \lfloor n/k \rfloor$;
  if ($n/k * k$!= k) grp_size++;
  $src\_grp = (s\%n)/grp\_size$; /* % is the mod op */
  $dst\_grp = (d\%n)/grp\_size$;
  use route: $node\ s \rightarrow BSW_{src\_switch}$
  $\rightarrow TSW_{src\_grp*k+dst\_grp}$
  $\rightarrow BSW_{dst\_switch}$
  $\rightarrow node\ d$;
}

Fig. 2. The OPT algorithm with the optimal worst-case permutation load

OPT requires $\sqrt{m}$ to be an integer and n to be a multiple of $\sqrt{m}$. The OPT algorithm described in Figure 2 deals with the cases when $\sqrt{m}$ is not an integer and/or n is not a multiple of $\sqrt{m}$. In those cases, $WORST_{OPT} = \lceil \frac{n}{\lfloor \sqrt{m} \rfloor} \rceil$. When $\sqrt{m}$ is not an integer, OPT only uses $(\lfloor \sqrt{m} \rfloor)^2$ top level switches to route SD pairs. Similarly, when n is not a multiple of $\sqrt{m}$, some top level switches are not used by OPT to route traffics.

topology type	topology	OPT	***D-m-k***	improvement
full bisection bandwidth fat-trees	$T(9+9, 18)$	3	9	200%
	$T(16+16, 32)$	4	16	300%
	$T(25+25, 50)$	5	25	400%
	$T(12+12, 24)$	4	12	200%
	$T(24+24, 48)$	6	24	300%
slimmed fat-trees	$T(12+4, 16)$	6	12	100%
	$T(24+9, 33)$	8	24	200%
	$T(24+16, 40)$	6	24	300%
	$T(16+8, 24)$	8	16	100%
	$T(24+8, 32)$	12	24	100%
fatted fat-trees	$T(8+16, 24)$	2	8	300%
	$T(12+16, 24)$	3	12	300%
	$T(10+25, 35)$	2	10	400%
	$T(8+24, 32)$	2	8	300%
	$T(16+32, 48)$	4	16	300%

TABLE I
WORST-CASE PERMUTATION LOAD ($WORST_R$) WITH OPT AND ***D-m-k***

Table I compares the worst-case permutation load ($WORST_R$) for OPT and ***D-m-k*** on various fat-trees. The last column is the improvement percentage of $WORST_{OPT}$ over $WORST_{D-m-k}$. From the table, it is clear that $WORST_{OPT}$ is significant better than $WORST_{D-m-k}$ on all the topologies considered: $WORST_{OPT}$ is 100% to 400% better than $WORST_{D-m-k}$ for the topologies. For the topologies in the table, $WORST_{D-m-k} = n$ while $WORST_{OPT} = \frac{n}{\sqrt{m}}$: OPT is better by a factor of $\sqrt{m}$. The larger the value of m,

type	topology	Bisect (ABB)			Full permutation ($AFPB$)			Dissemination (ADB)		
		OPT	*D-m-k*	imp.	OPT	*D-m-k*	imp.	OPT	*D-m-k*	imp.
full	$T(9+9,18)$	0.380	0.362	5.0%	0.333	0.266	25.2%	0.334	0.264	26.5%
	$T(16+16,32)$	0.317	0.296	7.1%	0.253	0.220	15.0%	0.265	0.219	21.0%
	$T(25+25,50)$	0.278	0.262	6.1%	0.224	0.193	16.1%	0.235	0.194	21.1%
	$T(12+12,24)$	0.333	0.323	3.1%	0.265	0.239	10.9%	0.266	0.239	10.9%
	$T(24+24,48)$	0.278	0.264	5.3%	0.215	0.196	9.7%	0.215	0.195	10.3%
slimmed	$T(12+4,16)$	0.234	0.228	2.6%	0.176	0.158	11.4%	0.184	0.157	17.2%
	$T(24+9,33)$	0.198	0.192	3.1%	0.149	0.131	13.7%	0.155	0.132	18.9%
	$T(24+16,40)$	0.244	0.235	3.8%	0.192	0.169	13.6%	0.199	0.169	17.8%
	$T(16+8,24)$	0.248	0.239	3.8%	0.185	0.171	8.2%	0.185	0.168	10.1%
	$T(24+8,32)$	0.189	0.183	3.3%	0.136	0.127	7.1%	0.136	0.126	7.9%
fatted	$T(8+16,24)$	0.500	0.442	13.1%	0.500	0.326	53.4%	0.500	0.326	53.4%
	$T(12+16,28)$	0.369	0.343	7.6%	0.333	0.259	28.6%	0.333	0.259	28.6%
	$T(10+25,35)$	0.500	0.425	17.6%	0.500	0.317	57.7%	0.500	0.317	57.7%
	$T(8+24,32)$	0.487	0.461	5.6%	0.430	0.351	22.5%	0.428	0.350	22.3%
	$T(16+32,48)$	0.374	0.340	10.0%	0.311	0.264	17.8%	0.311	0.264	17.8%

TABLE II
AVERAGE BANDWIDTH FOR DIFFERENT COMMUNICATION PATTERNS

the more improvement. Notice that the table also includes the sub-optimal cases for OPT when $\sqrt{m}$ is not an integer (e.g. $T(12+12,24)$) and/or when n is not divisible by $\sqrt{m}$ (e.g. $T(16+32,48)$). OPT has much better worst-case permutation loads in all cases.

IV. AVERAGE-CASE PERFORMANCE FOR COMMON COMMUNICATION PATTERNS

We use simulation to evaluate average-case performance of OPT on different fat-trees. The metrics used include average bisect bandwidth (ABB), average full permutation bandwidth ($AFPB$), average dissemination bandwidth (ADB). Since the numbers of bisect patterns, full permutations, and dissemination patterns are very large, we resort to a statistical method similar to one used in [2], [4] to compute average bandwidth for each patterns. We will describe how we compute ABB. Methods to compute $AFPB$ and ADB are similar. For each topology and each routing algorithm, we first sample 1000 random bisect patterns, and compute average bandwidth for the 1000 random patterns. We then compute the confidence interval with 99% confidence level for the average bandwidth. If the confidence interval is less than 1% of the average, we stop the simulation and report the computed average bandwidth as ABB (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. We compare our simulation results with those from the ORCS simulator [10] for several topologies and routing schemes, the results match perfectly, which validates our simulator.

As discussed earlier, when $\sqrt{m}$ is not an integer and/or when n is not divisible by $\sqrt{m}$, OPT does not use all top level switches for the traffic. In this evaluation, we enhance OPT with a "balancing" mechanism. In the enhanced OPT algorithm, we first apply OPT to obtain a baseline routing. If there are unused top level switches (e.g. when $\sqrt{m}$ is not an integer), the enhanced algorithm will then move SD pairs to the unloaded switch while maintaining that each link carries traffics from at most $\lceil \frac{n}{\lfloor \sqrt{m} \rfloor} \rceil$ sources or to at most $\lceil \frac{n}{\lfloor \sqrt{m} \rfloor} \rceil$ destinations. The SD pairs in each switch are considered to be moved to the unloaded switches in a round-robin fashion to balance the load on each switch.

We perform experiments on full bisection bandwidth, slimmed, and fatted fat-trees. Table II shows ABB, $AFPB$, and ADB for both OPT and *D-m-k*. Since we normalize the bandwidth (for different patterns) of a system where all nodes are connected by a single nonblocking crossbar switch to 1, an average bandwidth of 0.317 in the table means that the scheme achieves on average 31.7% of the performance of a nonblocking cross-bar switch for that particular type of communication patterns. As can be seen from the table, the average bandwidth with deterministic single-path routing is significantly less than 1, which confirms the findings in [4]. OPT out-performs *D-m-k* on all topologies. Specifically, for bisection patterns (ABB), OPT is between 3.1% to 7.1% better than *D-m-k* with an average of 5.3% on full bisection bandwidth fat trees; OPT is between 5.6% to 17.6% better with an average of 10.8% on fatted fat-trees; OPT is between 2.6% to 3.8% better with an average of 3.3% on slimmed fat-trees. For full permutation patterns, OPT is between 9.7% to 25.2% better with an average of 15.9% on full bisection bandwidth fat trees; OPT is between 17.8% to 57.7% better with an average of 35.3% on fatted fat-trees; OPT is between 7.1% to 13.6% better with an average of 10.8% on slimmed fat-trees. For full dissemination (Bruck) patterns, OPT is between 10.3% to 26.5% better with an average of 18.0% on full bisection bandwidth fat trees; OPT is between 17.8% to 57.7% better with an average of 35.3% on fatted fat-trees; OPT is between 7.9% to 18.9% better with an average of 14.4% on slimmed fat-trees. These results demonstrate that OPT is a more effective routing algorithm than *D-m-k* on 2-level generalized fat-trees. Notice that the table includes cases where $\sqrt{m}$ is an integer and n is divisible by $\sqrt{m}$ as well as

cases where $\sqrt{m}$ is not an integer and n is not divisible by $\sqrt{m}$.

There are several interesting observations in the table. First, OPT improves the $AFPB$ and ADB performance over *D-m-k* much more than the ABB performance. This is because the number of communications in a full permutation and a dissemination pattern is twice that in a bisect pattern. OPT is more effective with more communications to route in a pattern. This also explains the fact that the $AFPB$/ADB value is always less than or equal to its corresponding ABB value: more communications result in more link contention and lower bandwidth. Second, for all three patterns, OPT is more effective on fatted fat-trees and less effective on slimmed fat-trees. This is because fatted fat-trees have more route options while slimmed fat-trees have less route options. As a result, a better routing scheme OPT has more impacts on fatted fat-trees than on slimmed fat-trees. Nonetheless, OPT is consistently better on different types of fat-trees.

V. Conclusion

We investigate single-path routing schemes to improve performance on 2-level generalized fat-tree topologies. We show that existing single-path routing schemes are not ideal in terms of worst-case permutation performance. We design a routing scheme that achieves optimal worst-case permutation performance. Our evaluation demonstrates that our algorithm achieves better average-case performance for common communication patterns on full bisection bandwidth, slimmed, and fatted fat-trees in comparison to existing routing schemes.

References

[1] J. Bruck, C.-T. Ho, E. Upfal, S. Kipnis, and D. Weathersby, "Efficient Algorithms for All-to-all Communications in Multi-port Message-Passing Systems," *IEEE TPDS*, 8(11):1143-1156, 1997.

[2] P. Geoffray and T. Hoefler, "Adaptive Routing Strategies for Modern High Performance Networks," *IEEE HOTI'08*, pages 165-172, Aug. 2008.

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

[4] T. Hoefler, T. Schneider, and A. Lumsdain, "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks," *IEEE Cluster'08*, pages 116-125, 2008.

[5] Infiniband^{TM} Trade Association, *Infiniband ^{TM} Architecture Specification*, Release 1.2.1, January 2008.

[6] 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.

[7] 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.

[8] S.R. Ohring, M. Ibel, S.K. Das, M.J. Kumar, "On Generalized Fat Trees," *International Parallel Processing Symposium*, pages 37-44, April 1995.

[9] 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-DC'09) in conjunction with IEEE Cluster 2009*, pages 1-8, 2009.

[10] T. Schneider, T. Hoefler, and A. Lumsdaine, "ORCS: An Oblivious Routing Congestion Simulator," *Indiana University Techniocal Report, TR-675*, Indiana University Computer Science, Feb. 2009,

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

[12] 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.