IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

D-Mod-K Routing Providing Non-Blocking Traffic for Shift Permutations on Real Life Fat Trees

Eitan Zahavi

CCIT Report #776
September 2010

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

D-Mod-K Routing Providing Non-Blocking Traffic for Shift Permutations on Real Life Fat Trees

Eitan Zahavi

***Abstract*—As the size of High Performance Computing clusters grows, the increasing probability of interconnect hot spots degrades the latency and effective bandwidth experienced by MPI collectives. This report presents a proof that using D-Mod-K routing scheme for real life constant bisectional-bandwidth fat-tree topologies solves this problem for Shift Permutations traffic patterns.**

***Index Terms*—Network Topologies, Routing Algorithms and Techniques**

I. Introduction

In recent years, the ever increasing demand for compute power is addressed by building multiple thousand nodes High Performance Computing (HPC) clusters exceeding the peta-flop per second barrier [1]. State of the art parallel programs running on these clusters utilize the standard Message Passing Interface (MPI). Message passing overcomes the need to implement concurrent shared memory among the distributed processes at the cluster nodes. Many MPI applications exhibit a ratio of communication to computation time which is proportional to the number of nodes they use. For these parallel applications, the network latency and bandwidth may inhibit the desired linear performance acceleration with cluster size. There are two main network topologies used by HPC clusters: fat-trees and 3D tori. Fat-trees are commonly used for variable and diverse traffic patterns.

With the rise in HPC cluster size, a recent study [2] has measured degradations of network performance, down to 40% of the nominal bandwidth provided by fat-trees to MPI communication. The source of the degradation is attributed to cases where traffic from multiple sources congests a particular network link, creating a "hot spot". Our simulations reproduced these results, predicting the average degradation is 40% of the nominal network bandwidth for random job assignment. Furthermore, for adversarial job assignment, degradation to 7.1% of the network bandwidth was simulated.

Manuscript received Aug 16, 2010. This work was supported in part by Mellanox Technologies LTD.

Eitan Zahavi is with the Electrical Engineering Department, Technion, Haifa, 32000 Israel and with Mellanox Technologies, Yokneam, 20692 Israel (phone: +972-544-478803; e-mail: ezahavi@tx.technion.ac.il).

Traditionally, it was assumed that the primary bottleneck for performance scalability in HPC is the operating-system latency jitter. However, with the availability of and clock synchronization protocols and Collectives Offloading solutions to the jitter, network hot spots has recently become the new limiting factor for system growth [19].

This report holds a proof that a d-mod-k routing solution for a class of practical fat-tree topologies is able to route shift permutation traffic with no hot-spots.

The report is organized as follows: Background for fat-tree topologies is provided in section II. Section III describes the problem of network scalability. The D-Mod-K routing is presented in section IV and the proof in section V.

II. Real Life Fat-Tree Formulation

This section forms the basis for the discussion of non blocking routing in fat-trees. Although there are several fat-tree representations in the literature, there is no existing formulation that is sufficient to describe real life fat-trees used within today's HPC clusters. The progress from k-ary-n-trees [11][12] through Extended Generalized Fat-Trees [13], to this paper contribution of Parallel Ports Fat-Trees (PGFTs) and their sub-classification into Real Life Fat-Trees (RLFTs) will be discussed in the following. The notations developed in this section are used later in the proof of the proposed routing non-blocking properties (presented in the appendix).

A. Why Parallel Ports Generalized Fat-Trees are required?

The term fat-tree stems from the idea that if a single rooted tree of cross-bar-switches could be built (each node with K connections down and one connection up towards the root of the tree), in order to preserve bandwidth, its links towards the root of the tree should be K times "fatter" then it's down links. Such a tree of height h requires the top link to have a bandwidth which is larger by a factor of K^h than the bandwidth of a leaf link. This requirement renders such trees impractical as different speeds are required from switches and links at different levels in the tree. To overcome this requirement, the concept of a multi-rooted topology was introduced. However, in multi-rooted trees the non-blocking nature of the single rooted tree is replaced by a weaker attribute: they are rearrangeably-non-blocking i.e. given a permutation of source and destination pairs the routing on the tree can be made non-blocking.

Several families of fat-trees are known. The k-ary-n-tree is the basic type of tree built out of switches with an equal number of connections going up or down the tree [11][12]. These were extended by [13] with the introduction of Generalized Fat-Trees (GFT) which allow for a different number of up and down connections. Extended Generalized Fat-Trees (XGFT) further extends the possible topologies allowing for a different number of connections at each level. However, even though XGFTs expand the family of GFTs they still can not capture existing real life topologies. For example an XGFT with a minimal number of 8 port switches for connecting 16 end-ports is represented in Figure 1 (a). Since XGFTs allow only a single connection between switches, the resulting family does not preserve Cross Bisectional Bandwidth (CBB), i.e. the case where bandwidth towards the top of the tree is not equal to that of the leaves. To overcome this limitation we extend the XGFT definition by introducing the concept of Parallel ports Generalized Fat Trees (PGFT). The PGFT in Figure 1 (b) uses two parallel ports to maintain the CBB.

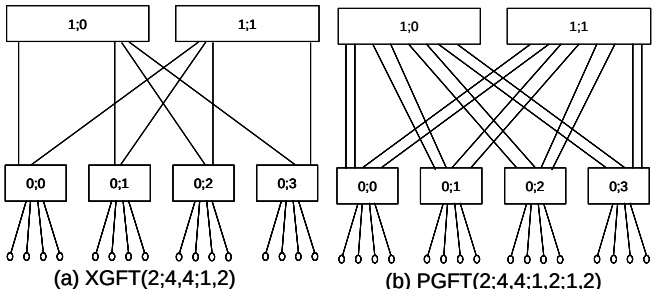
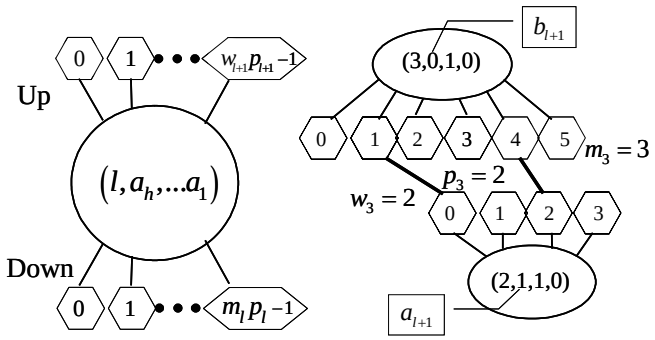


FIGURE 1 AN EXAMPLE OF A NON-MAXIMAL FAT-TREE SHOWING XGFT CAN NOT DESCRIBE FULL CBB WHILE PGFT CAN

B. *PGFTs Formal Definition*

PGFTs are canonically defined as:

$PGFT(h;m_1,...m_h;w_1,...w_h;p_1,...p_h)$ where h is the number of levels in the tree; m_l is the number of different lower level nodes connected to nodes on level l; w_l is the number of different upper level nodes connected to nodes on level $l-1$ and p_l is the number of parallel links connecting between nodes in level l and $l-1$. The XGFT defined by [13] is extended to PGFT by introducing up-going and down-going port objects. Like in XGFT tuples notation, each node is assigned a tuple $(l,a_h,...a_1)$, where l is its level and the vector of digits a_i describe the sub-trees the node is located at. Starting with a_h for the top most sub-tree and recursively a_{h-1} for the index of the sub-tree within that first sub-tree. PGFT adds ports to the nodes of the XGFT. Figure 2 (a) shows a single node (circle) and its ports (hexagons). There are $w_{l+1}p_{l+1}$ up going ports and $w_l p_l$ down going ports. Each port is assigned a tuple of the form: $(l,a_h,...a_1,q)$ which is equal to its node tuple with the addition of a port number - q.



(a) A single switch (b) Two levels with 2 parallel ports

FIGURE 2 INTRODUCING PGFT NODES, PORTS AND THEIR CONNECTIONS

To construct a PGFT one can first draw the nodes and ports on each level and then connect the ports between the levels using the following rules: ports $(l,a_h,...a_{l+1}...a_1,q)$ and $(l+1,b_h,...,b_{l+1}...b_1,r)$ are connected if and only if all the digits $b_i = a_i$ except for $i = l+1$, and the first of the p_{l+1} connections will be between the up-going port $q = b_{l+1}$ and the down-going port $r = a_{l+1}$. The k connection is between the up going port $q = b_{l+1} + kw_{l+1}$ and the down going port $r = a_{l+1} + km_{l+1}$. Figure 2 (b) demonstrates these connections for two nodes at levels 2 and 3: the 2 LSB digits of these two nodes match – so they must be connected. The first connecting link is between port number 0 of the lower node (which equals the 3rd digit of the upper node) and port number 1 of the upper node (which equals the 3rd digit of the lower nodes).

A formal definition of PGFT is given below, including the set of nodes and their up and down going ports for level l:

$$S_{l,h} := \left\{ \begin{matrix} (l,a_h,a_{h-1}...a_l...a_1) \mid \forall j: l < j \le h \to a_j \in [0..m_j) \wedge \\ \forall j: 0 < j \le l \to a_j \in [0..w_j) \end{matrix} \right\}$$

Up-going ports and down-going ports are defined by:

$$P^U_{l,h} := \left\{ \begin{matrix} (l,a_h,a_{h-1}...a_l...a_1,q)_U \mid 0 \le l < h \wedge 0 \le q < w_{l+1}p_{l+1} \wedge \\ \forall j: l < j \le h \to a_j \in [0..m_j) \wedge \forall j: 0 < j \le l \to a_j \in [0..w_j) \end{matrix} \right\}$$

$$P^D_{l,h} := \left\{ \begin{matrix} (l,a_h,a_{h-1}...a_l...a_1,q)_D \mid 0 < l \le h \wedge 0 < q \le m_l p_l \wedge \\ \forall j: l < j \le h \to a_j \in [0..m_j) \wedge \forall j: 0 < j \le l \to a_j \in [0..w_j) \end{matrix} \right\}$$

The graph edges connecting up-going ports or down-going ports to their switches are:

$$E^U_{l,h} := \left\{ \begin{matrix} \{(l,a_h,...a_1),(l,a_h,...a_1,q)_U\} \mid 0 \le l < h \wedge \\ 0 \le q < w_{l+1}p_{l+1} \wedge (l,a_h,a_{h-1}...a_1) \in V_h \end{matrix} \right\}$$

$$E^D_{l,h} := \left\{ \begin{matrix} \{(l,a_h,...a_1),(l,a_h,...a_1,q)_D\} \mid 0 < l \le h \wedge \\ 0 \le q < m_l p_l \wedge (l,a_h,a_{h-1}...a_1) \in V_h \end{matrix} \right\}$$

The set of connections between up and down-going ports (of different nodes) are defined as:

$$E^{p}_{l,h} := \left\{ \begin{array}{l} \{(l, a_h, ...a_{l+1}...a_1, q)_U, (l+1, b_h, ...b_{l+1}...b_1, r)_D\} \mid \\ 0 \le l < h \wedge 0 \le q < w_{l+1}p_{l+1} \wedge 0 \le r < m_{l+1}p_{l+1} \wedge \\ \forall j \in [1..h](j \ne l+1 \rightarrow a_j = b_j) \wedge \lfloor q / w_{l+1} \rfloor = \lfloor r / m_{l+1} \rfloor \wedge \\ b_{l+1} = q \bmod w_{l+1} \wedge a_{l+1} = r \bmod m_{l+1} \end{array} \right\}$$

C. *Descendants Criteria*

Two following two relations are later used by the routing algorithm: $\mathcal{D}_v(S_a, S_b)$ represents the relation between node $S_a = (l, a_h, a_{h-1}...a_l...a_1)$ to be located in the sub-tree of node: $S_b = (m, b_h, b_{h-1}...b_l...b_1)$ and $\mathcal{D}_p(S_a, P_b)$ describes that $S_b = (m, b_h, b_{h-1}...b_l...b_1)$ is min-hop accessible through the down-going port: $P_b^D = (m, b_h, b_{h-1}...b_l...b_1, q)$:

$$\mathcal{D}_v(S_a, S_b) = \left\{ \begin{array}{l} (l, a_h, a_{h-1}...a_l...a_1), (m, b_h, b_{h-1}...b_l...b_1) \mid \\ l < m \wedge \\ \forall j \in [1..h]((j > m) \vee (j \le l)) \rightarrow (a_j = b_j) \end{array} \right\}$$

$$\mathcal{D}_p(S_a, P_b) = \left\{ \begin{array}{l} (l, a_h, a_{h-1}...a_l...a_1), (k, b_h, b_{h-1}...b_l...b_1, q) \mid \\ l < k \wedge \\ \forall j \in [1..h]((j > k) \vee \\ (j \le l)) \rightarrow (a_j = b_j) \wedge a_k = q \bmod m_k = a_k \end{array} \right\}$$

D. *Real Life Fat Trees (RLFT)*

XGFTs and PGFTs support the definition of a large variety of topologies. Not all of them are practical to build. This section describes the characteristics of the Real Life Fat-Trees (RLFT), a sub-class of PGFTs, which are further studied by the rest of this paper. The set of attributes that makes a PGFT into an RLFT are presented below.

The first restriction for a PGFT to be routed in a non blocking manner is that it preserves a constant bisectional bandwidth. If CBB is not constant and is reduced going up the tree, some links must carry more than one flow at a given communication stage of a Shift CPS (since in each stage all of the nodes send data). The constant CBB requirement means that the nodes input BW equals their output BW or: $m_l p_l = w_{l+1} p_{l+1}$.

The second restriction applied to the PGFT is that the end-ports are actually not switches but host network interface cards which connect to the PGFT via a single cable *i.e.* $w_1 = p_1 = 1$.

The third restriction stems from the practical cost aspect of large HPC PGFTs: Real life HPC are always designed using the highest port-count cross-bar switch available, and the same switch is used at all levels in the tree. So PGFTs addressed in the rest of the paper will assume all of the switches have the same number of ports. At each level > 0: $m_l p_l + w_{l+1} p_{l+1}$.

It is common to define switch *arity* $\equiv K$ which is half of the switch ports: $K = (m_l p_l + w_{l+1} p_{l+1})/2$. The top level of the tree has only down-going ports and thus $m_h p_h = 2K$.

Combining the above restrictions for RLFTs:

$$\forall l \in \{1..h-1\} : K = m_l p_l = w_{l+1} p_{l+1} = m_h p_h / 2 \quad (1)$$

$$N = \prod_{i=1}^{h} m_i = \frac{2K^h}{\prod_{k=1}^{h} p_k} \quad (2)$$

III. MPI Collectives Scalability with Fat Tree size

The measurements performed by [2] have shown that the effective bandwidth for various collectives may degrade to ~40% of the network capacity. This bandwidth loss is attributed to hot-spots caused by the communication pattern. These results were reproduced by a simulation of a 1944 nodes Infiniband™ cluster using a OMNet++ [20]. The simulation model is calibrated against Infiniband™ QDR links (4000MBps unidirectional bandwidth) of IS4 switches (36 ports) connected to hosts with PCIe Gen2 8X slots (supporting 3250MBps unidirectional bandwidth).

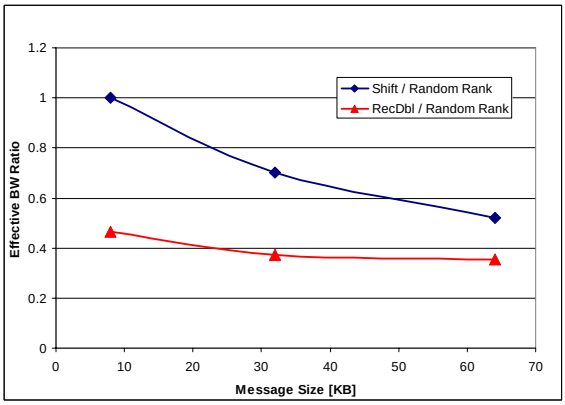


Figure 3 Shift and Recursive Doubling Collectives Normalized BW vs. Message Size

Traffic is injected from each end-port according to a predefined destination sequence. The end-ports progress through this predefined destinations sequence independently when their previous message has been sent to the wire. The Shift and the Recursive-Doubling CSS were simulated and the resulting normalized effective bandwidth for different message sizes is provided in Figure 3 (normalized to the full PCIe bandwidth). A random assignment of the MPI "rank" to cluster nodes is used. It can be seen from the graph that as the message size is increasing the effective bandwidth is decreasing. The reason for that is the buffering available in the fabric is able to compensate for short hot spots. Only when the

buffers fill-up congestion trees are built and reach the source end-ports resulting in actual bandwidth lost.

To further illustrate the importance of correct routing and ordering of end-ports (i.e. MPI "rank" assignment), a simulation is performed for a Ring permutation sequence with adversary routing and node "rank" assignment. The adversary assignment was made such that all of the nodes of each leaf switch send data to nodes of other leaf switches. The selection of the particular destinations for each node is done to maximize the sharing of the leaf up-going port. The results show that the effective bandwidth can drop by a factor of 14 for a Ring permutation sequence. The measured average bandwidth for this traffic pattern is 231.5MBps which is close to the network bandwidth of 4000MBps divided by the worst possible link oversubscription of 18. The normalized effective bandwidth ratio obtained is ~7%.

In all of the above simulations it was assumed that end-ports progress through the stages of the permutation sequence in asynchronous manner. When synchronization is performed the impact of hot-spots can be even worse, as the latency of the worst pair in each stage is accumulated.

IV. Non Blocking Routing for Shift Permutations on RLFTs

This section describes the proposed routing solution for shift permutations on RLFTs and then proves it is non-blocking. Shift permutation with $s \in \{1 \cdots N-1\}$ is defined as the set of source destination pairs such that node n_i sends data to node n_j:

$$\left\{\left(n_i, n_j\right) \mid j = \left(i+S\right) \bmod N\right\}_{i=0}^{N-1}$$

Several previous studies [14][15][16][17][18] have addressed this topic for specific topologies. Most of them describe routing which we refer to as d-mod-k which is defined later in this section. We extend the common d-mod-k to RLFTs making it practical to real-world topologies. The basic property of the d-mod-k routing is that each down-going port is used to pass traffic to a single end-port, so blocking may only happen for traffic going up the tree. This section focuses on providing routing for Shift permutations that route each pair on a different up-going port while maintaining a single destination allocated to each down-going port. These two properties of the routing are then proved by theorems 1 and 2. The section starts with an intuitive description of the routing principles followed by formal definition of the routing and then proves it is non-blocking for all Shift permutations.

A. Routing Description

Consider traffic flowing up the tree from a fully populated leaf switch (switch at level 1). A Shift permutation sequence guarantees that for a contiguous set of traffic sources the set of traffic destinations is also contiguous and in the same order. To avoid congestion on up-going links, the proposed routing scheme spreads the traffic among all up-going ports. For the lowest level leaf-switches, the index of the up-going port route for a given destination is set to be the destination index modulo the total number of up-going ports. The up-going port assignment is cyclical with the destination number such that at any given stage of the Shift, the contiguous range of destinations is evenly distributed through all of the up-going ports in a non-overlapping manner. Destinations routed through a second level switch share the same port index at the first level switches. These destinations form an arithmetic sequence with a difference equal to the number of up-going ports of the first level switches. Such a sequence can be spread without overlaps on the up-going ports of the second level switches by dividing the destination index by the sequence distance modulo the number of up-going ports. Applying this principle recursively on all of the tree levels provides a non blocking routing for the Shift CPS.

Figure 4 holds a fragment of a PGFT to demonstrate the routing described above. Up-going ports are marked with their port number and the level 1 routing is shown to be through port number *q=j mod 4*. The set of destinations through each port is provided on the right most leaf switch. For example, consider traffic flowing from the right most node at level 1 toward destination 9. Traffic traverses the node at level 1 through port 1 and then the node at level 2 through port 2. Note that every 4 contiguous destinations are to be routed through different up-going ports. For the second level switches the routing to destination *j* is through port number *q=(j/4) mod 4*, and the same property holds.

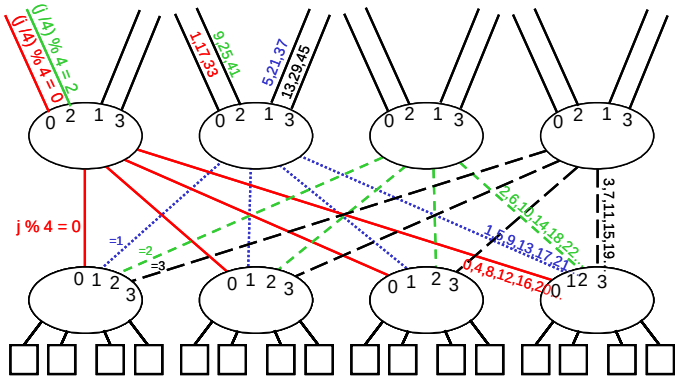


FIGURE 4 EXAMPLE ROUTING ON A PGFT SUB-SECTION

B. RLFT D-Mod-K Routing Formulation

Unlike the traversal based approach presented in [18], it is preferable to have a closed form routing solution following the same principles outlined in [17]. Such formulation for RLFT is described below. Beyond the benefit of being non-blocking for shifts, the formulation allows for a parallel implementation of the routing configuration, solving another important scalability issue. The most known routing algorithm, Dijkstra's SPP [21] runtime is $O(N^2)$ and there is no parallel version for this algorithm that can be easily fit the

cluster topology. However, an algorithm which is expressed in a closed form can be easily executed in parallel on every switch node and thus achieve an O(*N*) run time.

The routing is based on the tuple formulation of the tree. The assignment of tuples to the tree nodes may be performed using the algorithm provided in [18]. Following this step, end-ports are marked with an increasing index. Then the routing tables in the switches are programmed such that traffic to an end-port is either forwarded through an up-going or a down-going port. For destinations which are descendants of the node, the routing is through the descendant port (satisfying the $\mathcal{D}_p(S_a, P_b)$ criterion). For the routing in the up direction we define the up-going port to be used for routing packets to a destination as:

$$P_l^U(j) \equiv (l, s_h..s_1, q_l^U) \mid q_l^U(j) = \left\lfloor j / \prod_{k=1}^{l} w_k \right\rfloor \bmod \left(w_{l+1} p_{l+1}\right) \quad (3)$$

This routing is not a function of the switch, i.e. all switches at a specific level use the same up-going port index to route to the same destination.

V. D-Mod-K Provides Non Blocking Shift on RLFTs

The following two theorems state that the above routing formulation provides non blocking routing in both down and up directions for all Shift permutations on RLFTs

Theorem 1: The routing according to (3) guarantees that no more than one destination is routed through any of the up-going ports in the network, for all stages of the Shift CPS on a complete RLFT .

The following lemmas are used to prove theorem 1:

Lemma 1: The set of destinations routed through up-going ports of a node $(l, b_h, ...b_1)$, not including the top nodes, is a sub-set of the algebraic sequence:

$$J_{(l,b_h,...b_1)} = \left\{ \sum_{t=1}^{l} b_t \prod_{k=1}^{t-1} w_k + i \prod_{k=1}^{l} w_k \mid 0 \le i < N \Big/ \prod_{k=1}^{l} w_l \right\} \quad (11)$$

The motivation for using the super-set described by (11) rather than the actual sequence is that the expression for the accurate set is much more complex than (11) and is not required for proving theorem 1. Note that in the accurate sequence, destinations that are descendants of a node pass through it but are not routed through the up-going ports (but the down-going ports).

The proof of lemma 1 is based on a recursion starting with the destinations of end-ports: Each end-point may send data to all of the other end-points: $J_{(0,b_h,...b_1)} = \{i \mid 0 \le i < N\}$

Based on the routing (3) for level 0: $q_0^U(j) = j \bmod \left(w_1 p_1\right)$ and the PGFT connection rule $b_{l+1} = q \bmod w_{l+1}$ the sequence of destinations passing through the parent with $b_1 = q$ start with destination b_1 and a step of w_1 so $J_{(1,b_h,...b_1)} = \{b_1 + i w_1 \mid 0 \le i < N/w_1\}$. Since all of the children nodes of $(1, b_h, ...b_1)$ connect to it through $b_1 = q$ they all pass the same sequence of destinations to that parent.

Similarly the sequence of destinations passing through second level nodes, is obtained using the Routing (3) on each element of the set $J_{(1,b_h,...b_1)}$:

$$J_{(2,b_h,...b_1)} = \{b_1 + b_2 w_1 + i w_1 w_2 \mid 0 \le i < N/w_1 w_2\}.$$

Finally for a node at arbitrary level the sequence of destinations passing through it is:

$$J_{(l,b_h,...b_1)} = \left\{ \sum_{t=1}^{l} b_t \prod_{k=1}^{t-1} w_k + i \prod_{k=1}^{l} w_k \mid 0 \le i < N \Big/ \prod_{k=1}^{l} w_l \right\} \square$$

Lemma 2: Routing (3) is non-blocking for any continuous sub-sequence of destinations passing through a node (not including the top level nodes) of size equal to the number of up-going ports.

Proof of lemma 2: Given a destination of index i, from the above sequence, the up-going port obtained by (1) is:

$$q_l^U(j) = \left\lfloor \frac{\sum_{t=1}^{l} b_t \prod_{k=1}^{t-1} w_k + i \prod_{k=1}^{l} w_k}{\prod_{k=1}^{l} w_k} \right\rfloor \bmod \left(w_{l+1} p_{l+1}\right) \quad (12)$$

$$q_l^U(j) = \left(C + i\right) \bmod \left(w_{l+1} p_{l+1}\right) \quad (13)$$

So a contiguous sub-sequence of $w_{l+1} p_{l+1}$ destinations which makes a contiguous range of index values maps to a cyclic set of $w_{l+1} p_{l+1}$ up-going ports. So for such sub-sequence all up-going ports are used, each for exactly one destination $\square$

Lemma 3: For RLFTs routing (3) is non-blocking for any contiguous sub-sequence of destinations passing through a node (not including the top level nodes) of size equal to the number of up-going ports which may wrap around from the last possible index to the first element of the destination sequence

Proof of lemma 3: To prove lemma 3 we show that the up-going port used for routing the next destination past the last one is the same as the one used for routing to the first destination. This condition is met if the expression for the number of destinations in the sequence is a multiple of

$w_{l+1}p_{l+1}$.

For RLFTs the number of nodes was given in (2). The index after the last index is:

$$\frac{N}{\prod_{k=1}^{l} w_k} = \frac{2K^h}{\prod_{i=1}^{l} w_k \prod_{k=1}^{h} p_k} = \frac{2K^{h-l}}{\prod_{k=l+1}^{h} p_k} \quad (14)$$

Expression (14) is a product of $w_{l+1}p_{l+1} = K$ if $\frac{2K^{h-l}}{\prod_{k=l+1}^{h} p_k}$ is an integer. On RLFTs with constant radix for all levels:

$$\frac{2K^{h-l-1}}{\prod_{k=l+1}^{h} p_k} = 2\prod_{k=l+1}^{h} \frac{K}{p_k} = 2\prod_{k=l+1}^{h} m_k \quad (15)$$

which is an integer □

Lemma 4: For each switch at levels $l = 1..h-1$ the sub-sequence of destinations routed through it, in every stage of a Shift CPS is contiguous or wraps around the last destination and do not exceed K elements.

Proof of lemma 4: A switch at level l may receive up-going traffic from its descendent leafs. Based on the recursive nature of the tree there are $\prod_{k=1}^{l} m_k$ such descendant leafs for switch at level l. If these descendants end-ports send to a contiguous set of destinations (Shift CPS) the switch will only route the subset which fulfils (2) i.e. are $\prod_{k=1}^{l} w_k$ apart.

The number of destinations passing through a switch in a single Shift CPS stage is:

$$\frac{\prod_{k=1}^{l} m_k}{\prod_{k=1}^{l} w_k} = \frac{\prod_{k=1}^{l} \frac{K}{p_k}}{\prod_{k=2}^{l} \frac{K}{p_k}} = K \quad (16)$$ □

Proof of theorem 1: Since lemma 4 show that there will be no more than K destinations routed up through a switch in an RLFT Shift permutation sequence stage and lemma 3 show that routing of these destinations is through different up-going ports the result is that the routing through that switch is non-blocking □

Theorem 2: The routing according to (3) guarantees that no more than one destination is routed through all the network down-going ports of a complete RLFT

The following lemmas are used to prove theorem 2:

Lemma 5: For routing (3) a single top level switch is passing all the flows to a specific destination

We prove by induction on the tree levels and show that routing towards a destination j from an arbitrary end-port $(0, s_h..s_1)$ at level l the traffic will pass trough switch whose l first tuple digits are independent of the source end-port. The result is that all the digits of the top level switch that a packet will traverse, are independent of the originating end-port. Which means a single top level switch will pass the traffic to a specific destination.

The base of the induction is for $l = 0$: The up-going port is obtained using the Routing (3) formulation $q_0^U(j) = j \bmod (w_1 p_1)$; then using the PGFT connections rule:

$$\left\{ \begin{array}{l} \{(0, a_h, ...a_{l+1}...a_1, q)_U, (1, b_h, ...b_{l+1}...b_1, r)_D\} \mid \\ 0 \le q < w_1 p_1 \wedge 0 \le r < m_1 p_1 \wedge \\ \forall j \in [2..h](a_j = b_j) \wedge b_1 = q \bmod w_1 \wedge \\ a_1 = r \bmod m_1 \end{array} \right\} \quad (17)$$

Since q is known and constant, b_1 is also known and constant which prove that routing from arbitrary end-port to destination j is going through first level switches sharing the first digit of the tuple.

The induction step assumes at level l the first l digits are known and constant and show that the PGFT connection rule adds the digit at place $l+1$:

$$\left\{ \begin{array}{l} \{(l, a_h, ...a_{l+1}...a_1, q)_U, (l+1, b_h, ...b_{l+1}...b_1, r)_D\} \mid \\ 0 \le q < w_{l+1} p_{l+1} \wedge 0 \le r < m_{l+1} p_{l+1} \wedge \\ \forall j \in [1..h](j \ne l+1 \rightarrow a_j = b_j) \wedge \\ \lfloor q / p_{l+1} \rfloor = \lfloor r / p_{l+1} \rfloor \wedge b_{l+1} = q \bmod w_{l+1} \\ \wedge a_{l+1} = r \bmod m_{l+1} \end{array} \right\} \quad (18)$$

All digits b_j with $j \ne l+1$ are preserved and $b_{l+1} = q \bmod w_{l+1}$ is constant since q is independent of the lower level switch □

Lemma 6: The number of destinations passing through an RLFT top level switch is at most the number of its ports

Proof of lemma 6: Using expression (11)

$$J_{(l, b_h, ...b_1)} = \left\{ \sum_{t=1}^{l} b_t \prod_{k=1}^{t-1} w_k + i \prod_{k=1}^{l} w_k \mid 0 \le i < N \Big/ \prod_{k=1}^{l} w_k \right\}$$

for the top level we get the number of destination is:

$$\frac{N}{\prod_{k=1}^{h} w_k} = \frac{2K^h}{\prod_{k=1}^{h} w_k \prod_{k=1}^{h} p_k} = \frac{2K^h}{\prod_{k=1}^{h} p_k w_k}$$

For RLFT the first level is of single connection and the rest of the levels have K connections so:

$$\frac{N}{\prod_{k=1}^{h} w_k} = \frac{2K^h}{p_1 w_1 \prod_{k=2}^{h} p_k w_k} = \frac{2K^h}{K^{h-1}} = 2K \quad \square$$

Proof of theorem 2: Combining lemmas 5 and 6 different sets of exactly 2K destinations are routed through each top level switch which as 2K ports. So each port may be assigned just one destination. Applying these lemmas to sub-trees in a recursive manner show that theorem 2 holds for every switch in the tree

VI. Conclusion

In this report we present formal definition for Real Life Fat Trees which are used in today's High Performance Clusters. We extend D-Mod-K routing for this class of Fat Trees and prove that this routing will be able to provide non blocking traffic for Shift permutations.

Acknowledgment

Thanks to my advisors A. Kolodny and I. Cidon.

References

[1] http://www.top500.org

[2] Torsten Hoefler, Timo Schneider, Andrew Lumsdaine "Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks," Cluster Computing, *IEEE International Conference*, pp.116-125, 2008

[3] K. Asanovic *et al.* "The landscape of parallel computing research: a view from Berkeley," *Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley*, 2006

[4] Kerbyson D.J., Barker K.J, "Automatic Identification of Application Communication Patterns via Templates". *Int. Conf. on Parallel and Distributed Computing Systems (PDCS)*, Las Vegas, NV, 2005

[5] Alme H.J., Hoisie A., Petrini F., Wasserman, H.J., Gittings M.L., Kerbyson D.J., "Predictive Performance and Scalability Modeling of a Large-scale Application", *SC'01*, Denver, CO. 2001

[6] Wolfgang E. Denzel , Jian Li , Peter Walker , Yuho Jin**,** "A framework for end-to-end simulation of high-performance computing systems," *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, March 03-07, 2008, Marseille, France

[7] Petrini, J. Fernandez, E. Frachtenberg, and S. Coll F., "Scalable collective communication on the asci q machine," *Hot Interconnects 12*, 08 2003

[8] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of Collective communication operations in MPICH," *Int'l Journal of High Performance Computing Applications*, 19(1):49–66, Spring 2005

[9] J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg, E. Gabriel, and J. J. Dongarra, "Performance Analysis of MPI Collective Operations," *Proceedings of the 19th International Parallel and Distributed Processing Symposium, 4th International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems* (PMEO-PDS 05), Denver, CO, April 2005

[10] OpenMPI implementation of tuned collectives layer: http://svn.open-mpi.org/svn/ompi/trunk/ompi/mca/coll/tuned/

[11] C.E. Leiserson, "Fat-trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transactions on Computers*, 34(10):892-901, Oct. 1985

[12] Fabrizio Petrini, Marco Vanneschi, "k -ary n -trees: High Performance Networks for Massively Parallel Architectures," *11th International Parallel Processing Symposium (IPPS '97)*, ipps, pp.87, 1997

[13] Sabine R. Öhring , Maximilian Ibel , Sajal K. Das , Mohan J. Kumar**,** "On generalized fat trees," *Proceedings of the 9th International Symposium on Parallel Processing*, p.37, April 25-28, 1995

[14] S, Heller. [ed.] K. Bolding and L. Synder. "Congestion-Free Routing on the CM-5 Data Router," *First International Workshop PCRCW, Seattle, Washington*, LNCS, Vol. 853, pp. 176-184, May 1994

[15] Sameer Kumar, Laxmikant V. Kale, "Scaling All-to-All Multicast on Fat-tree Networks," *Proceedings of the Parallel and Distributed Systems, Tenth International Conference*, p205 2004 DOI:10.1109/ICPADS.2004.77

[16] Xuan-Yi Lin, Yeh-Chin Chung, and Tai-Yi Huang. "A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks," *IEEE International Parallel and Distributed Processing Symposium (IPDPS'04)*. pp. 1-13, 2004

[17] C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez, J. Duato, "Deterministic versus Adaptive Routing in Fat-Trees," *IEEE International Parallel and Distributed Processing Symposium, (IPDPS*'07), pp.292, 2007

[18] Eitan Zahavi, Gregory Johnson, Darren J. Kerbyson, and Michael Lang. "Optimized InfiniBand Fat-tree Routing for Shift All-To-All Communication Patterns," *Concurrency and Computation: Practice and Experience*. Volume 22 Issue 2, Pages 217 – 231. 2009. DOI: 10.1002/cpe.1527

[19] P. Beckman, K. Iskra, K. Yoshii, and S. Coghlan, "The Influence of Operating Systems on the Performance of Collective Operations at Extreme Scale," in *IEEE International Conference on Cluster Computing*, 2006

[20] OMNeT++: an extensible, modular, component-based C++ simulation library and framework: http://www.omnetpp.org/

[21] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM*, 46(3):362–394, 1999

[22] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, A Message Passing Standard for MPP and Workstations," *Comm. ACM*, Vol. 39, No. 7, July 1996, pp.~84--90