A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks

Xuan-Yi Lin, Yeh-Ching Chung, and Tai-Yi Huang
Department of Computer Science
National Tsing-Hua University, Hsinchu, Taiwan 300, ROC
{xylin, ychung, tyhuang}@cs.nthu.edu.tw

Abstract

In a cluster system, performance of the interconnection network greatly affects the computation power generated together from all interconnected processing nodes. The network architecture, the interconnection topology, and the routing scheme are three key elements dominating the performance of an interconnection network. InfiniBand architecture (IBA) is a new industry standard architecture. It defines a high-bandwidth, high-speed, and low-latency message switching network that is good for constructing high-speed interconnection networks for cluster systems. Fat-trees are well-adopted as the topologies of interconnection networks because of many nice properties they have. In this paper, we proposed an m-port n-tree approach to construct fat-tree-based InfiniBand networks. Based on the constructed fat-tree-based InfiniBand networks, we proposed an efficient Multiple LID (MLID) routing scheme. The proposed routing scheme is composed of processing node addressing scheme, path selection scheme, and forwarding table assignment scheme. To evaluate the performance of the proposed routing scheme, we have developed a software simulator for InfiniBand networks. The simulation results show that the proposed routing scheme runs well on the constructed fat-tree-based InfiniBand networks and is able to efficiently utilize the bandwidth and the multiple paths that fat-tree topology offers under InfiniBand architecture.

1. Introduction

In cluster systems, high bandwidth, low latency interconnection networks are essential for high-throughput computations. The network architecture, the interconnection topology, and the routing scheme are three key elements that dominate the performance of interconnection networks.

The InfiniBand architecture (IBA) is a new industry-standard architecture for server I/O and inter-server communication. IBA defines a switch-based, point-to-point interconnection network that enables high-speed, low-latency communication between connected devices [3]. Due to the characteristics of IBA, it is very suitable to use IBA as the interconnection network of a cluster system. When using IBA as the interconnection network of a cluster system, the interconnection topology can be arbitrarily given by users, based on considerations such as hardware availability, bandwidth requirement, and the utilization of some specific communication patterns between processing nodes.

Many interconnection topologies and related issues have been discussed in the literature [1] [2] [5]. In this paper, we focus on the implementation of the popular fat-tree topology on InfiniBand networks. To implement fat-tree-based InfiniBand networks, the construction of the fat-tree and the routing scheme for the constructed fat-tree are two important issues. There are many ways to construct the fat-trees. In this paper, we proposed an *m*-port *n*-tree approach to construct the fat-tree-based InfiniBand networks.

The InfiniBand network is a packet-switching based network. Routing in an InfiniBand subnet is based on the forwarding table stored in each switch. When a packet arrives in a switch, the packet will be forwarded to the corresponding output port via the forwarding table lookup. The forwarding table stored in each switch maps the local identifier (LID) in the DLID field of a packet to an output port of the switch. Since one LID only maps to one output port, routing in InfiniBand subnet is deterministic. In a fat-tree-based InfiniBand network, there are multiple paths between two processing nodes. If each processing node is associated an LID, when multiple processing nodes want to send packets to the same processing node at the same time, the link congestion may occur and the bandwidth and multiple paths offered by the fat-tree topology may be wasted.

Some routing algorithms and techniques for InfiniBand networks have been proposed in [7] [12] [13]. These routing algorithms, in general, are designed for irregular topologies. When applied to regular topologies like fat-trees, they may not take all the properties of a regular

topology into account and usually cannot deliver satisfactory performance. In order to make good use of the bandwidth and multiple paths offered by the fat-tree topology, we proposed a multiple LID (MLID) routing scheme based on the LID Mask Control (LMC) mechanism provided by IBA. The proposed routing scheme consists of processing node addressing scheme, path selection scheme, and forwarding table assignment scheme. The processing node addressing scheme assigns multiple LIDs to each processing node. The path selection scheme determines an LID of the destination processing node for a packet sent from a source processing node. There exists a one-to-one mapping (unique path) between a source processing node and an LID of a destination processing node. The forwarding table assignment scheme correctly setup the forwarding table of each switch based on the processing node addressing scheme and the path selection scheme. To evaluate the performance of the proposed routing scheme, we have developed a software simulator for InfiniBand networks. The proposed routing scheme along with the Single LID (SLID) routing scheme were simulated under different network sizes and traffic patterns. Simulation results show that performance of the proposed routing scheme outperforms the SLID scheme for all test cases.

This paper is organized as follows. In Section 2, fat-trees and InfiniBand architecture are introduced. Section 3 details the construction of fat-tree-based InfiniBand networks based on fix-arity switch components. Section 4 describes the proposed routing scheme for the constructed fat-tree-based InfiniBand networks. In Section 5, the performance of the proposed routing scheme is evaluated under different network sizes and traffic patterns.

2. Preliminaries

2.1 Fat-Tree Topology

Fat-trees are a family of general-purpose interconnection strategies that effectively utilize any given amount of hardware resource devoted to communication [6]. Kinds of fat-tree variations have been adopted by many research prototypes and commercial machines. The major difference between fat-trees and traditional tree architecture is that fat-trees are more resemble real trees. In a traditional tree, the link bandwidth is fixed no matter how high the tree is. This will cause traffic congestion problems occurring at root switch. In a fat-tree, the link bandwidth increases when upward from leaves to root. The root congestion problem can be relieved. In a fat-tree-based interconnection network, leaf nodes represent processors, internal nodes are switches, and edges correspond to bidirectional links between parents and children. Figure 1 shows the difference between traditional trees and fat-trees. In Figure 1(a), a binary tree is shown. A binary fat-tree is shown in Figure 1(b). From Figure 1(b), we can see that the edge (link bandwidth) gets thicker (higher) when closer to the root. Figure 1(c) shows a more detailed architecture of a fat-tree.

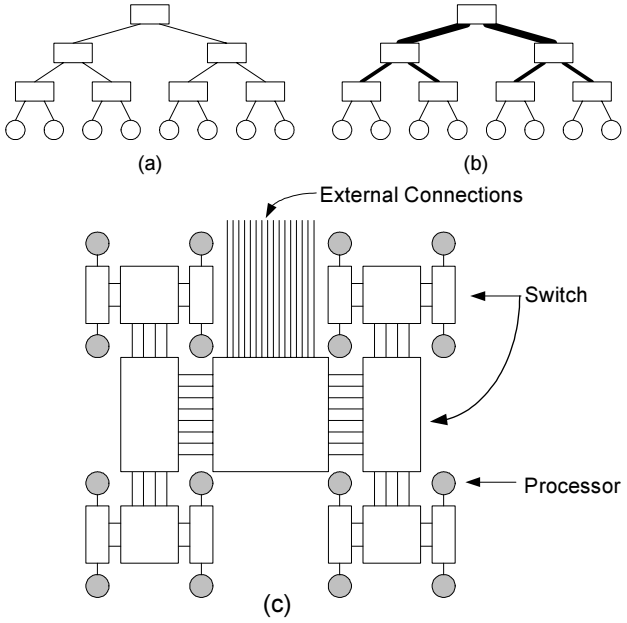


Figure 1: (a) A binary Tree. (b) A binary fat-tree. (c) A more detailed binary fat-tree.

Routing in a fat-tree is relatively easy since there is a unique shortest path between a pair of processing nodes *i* and *j*. The routing consists of two phases, the ascending phase and the descending phase. In the ascending phase, a message from processing node *i* to processing node *j* goes upward through the internal switches of a fat-tree until the least common ancestor *m* of processing nodes *i* and *j* is found. In the descending phase, the message goes downward through the internal switches to processor *j*. Figure 2 shows a routing example in a fat-tree.

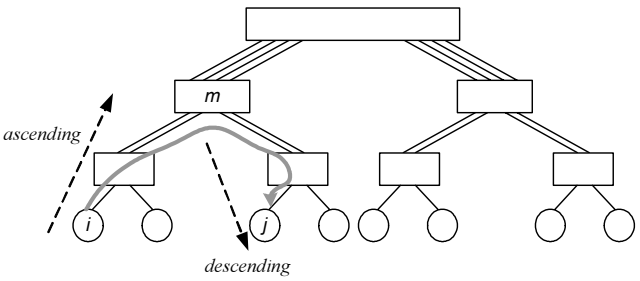


Figure 2: A routing example in a fat-tree.

2.2 InfiniBand Architecture (IBA)

The InfiniBand architecture (IBA) is a new industry-standard architecture for server I/O and inter-server communication. IBA is designed around a point-to-point, switched I/O fabric, whereby end node devices are interconnected by cascaded switch devices [3]. IBA can be used to connect multiple independent processor platforms (i.e., host processor nodes), I/O platforms, and I/O devices. An InfiniBand network can be divided into subnets. Switches are used to perform intra-subnet communication between connected devices and routers are used to perform inter-subnet communication. There is one or several Subnet Manager (SM) in an InfiniBand subnet. The SM is responsible for the configuration and the control of a subnet. In an InfiniBand subnet, packet source/destination is called endport. A Local Identifier (LID) is an address assigned to an endport by the SM during the subnet initialization process. LID is unique within an InfiniBand subnet.

The InfiniBand network is a packet-switching based network. Routing in an InfiniBand subnet is deterministic, based on the forwarding table lookup. For a packet, the LIDs of its source and destination processing nodes are stored in SLID and DLID fields of the Local Route Header (LRH), respectively. A packet within a switch is forwarded to an output port based on the packet's DLID field and the switch's forwarding table. When a switch received a packet, the packet header is parsed and the DLID field is mapped to an entry of the forwarding table stored in the switch. According to the value of the mapped entry, the switch can forward the packet to its next destination. An example is illustrated in Figure 3. This deterministic behavior simplifies the design of switches. The latency of packet relaying can be limited to a very small value as well.

Since there are only one-to-one mapping between DLID and output port, in order to support multiple paths, IBA defines an LID Mask Control (LMC) value that can be assigned to each endport. According to the LMC value, an endport can be associated with more than one LID such that communications between any pair of endports can go through different available paths. The LMC is a 3-bit field that represents 2^{LMC} paths (maximum of 128 paths). During topology discovery process at the subnet initialization, the SM may determine the number of paths for a given endport and will partition the 16-bit LID space to assign a base LID and up to 2^{LMC} sequential LIDs to each endport based on the LMC. An example is given in Figure 4. In Figure 4, if processing node *A* is sending message to processing node *B*, there will be two paths available between processing nodes *A* and *B*. The DLID field in a packet that processor *A* generated will determine the routing path for the packet. For example, if the DLID of a packet is 4, the packet will be sent from processing node *A* through switches *P* and *R* to processing node *B*. If the DLID of a packet is 5, the packet will be sent from processing node *A* through switches *P*, *Q*, and *R* to processing node *B*.

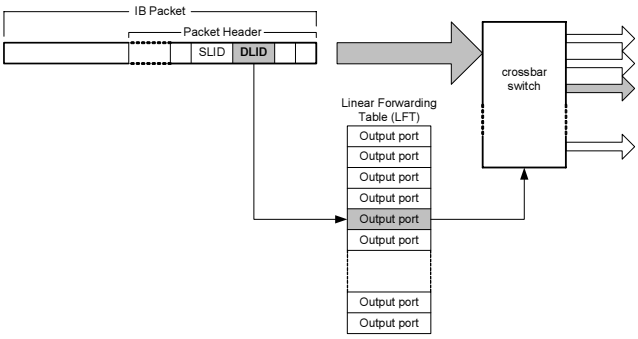


Figure 3: The linear forwarding table (LFT) lookup in a switch.

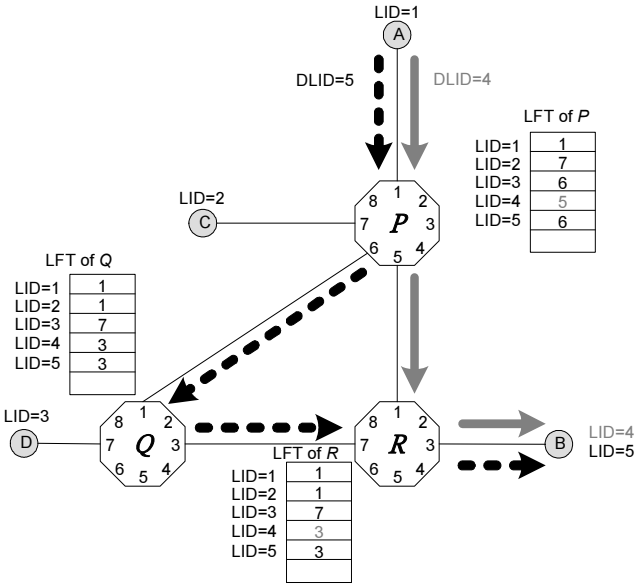


Figure 4: An example of multiple paths to the same processing node based on different DLIDs.

3. The Construction of Fat-Tree-Based InfiniBand Networks

In this section, we will discuss how to use InfiniBand switches to construct fat-tree-based InfiniBand networks. From the fat-tree described in Figure 1(c), we observed that the arity of internal switches of a fat-tree increases as we traverse upward from leaves to root. The requirement of increasing arity switches makes the physical implementation of switch-based fat-tree topology unfeasible. To solve this problem, some alternatives have

been proposed to construct fat-trees using constant size elements [14] or use fixed-arity switches [7][10]. These solutions trade connectivity with simplicity, that is, incoming messages at a given switch in a "full" fat-tree may have more choices in the routing decision than in a corresponding network with fixed-arity switches. Performance and properties of such kind of switch-based fat-trees are studied in [8] [9]. In the following, we will show how to use fixed-arity switches to construct the desired fat-tree-based InfiniBand networks.

Based on the need of fixed-arity communication switches to construct a fat-tree network, we proposed an *m*-port *n*-tree approach to construct the desired fat-trees. The *m*-port *n*-trees are a class of fixed-arity fat-trees. Given an *m*-port *n*-tree *FT*(*m*, *n*), it has the following characteristics:

1 The height of *FT*(*m*, *n*) is $n + 1$.
2 *m* is a power of 2.
3 *FT*(*m*, *n*) consists of $2\times(m/2)^n$ processing nodes and $(2n-1)\times(m/2)^{n-1}$ communication switches.
4 Each communication switch has *m* communication ports.

The processing node in *FT*(*m*,*n*) is labeled as $P(p = p_0 p_1 \cdots p_{n-1})$, where $p \in \{0,1,\cdots,m-1\}$ × {0,1,…, $(m/2) - 1\}^{n-1}$. For example, the set of processing nodes in a 4-port 3-tree is {*P*(000), *P*(001), *P*(010), *P*(011), *P*(100), *P*(101), *P*(110), *P*(111), *P*(200), *P*(201), *P*(210), *P*(211), *P*(300), *P*(301), *P*(310), *P*(311)}. For processing node *P*(310), we have $p_0 = 3$, $p_1 = 1$, and $p_2 = 0$. The communication switch in *FT*(*m*, *n*) is labeled as $SW < w = w_0 w_1 \cdots w_{n-2}, l >$, where $l \in \{0,1,\cdots,n-1\}$ is the level of the switch and

$$w \in \begin{cases} \{0,1,\cdots,(m/2)-1\}^{n-1} & \text{if } l = 0 \\ \{0,1,\cdots,m-1\}\times\{0,1,\cdots,(m/2)-1\}^{n-2} & \text{if } l \in \{1,2,\cdots,n-1\} \end{cases}.$$

Let $SW\langle w, l\rangle_k$ denote the *k*th port of $SW\langle w, l\rangle$, where *k* = 0, 1, 2,…, *m*–1. For switches $SW\langle w, l\rangle$ and $SW\langle w', l'\rangle$, ports $SW\langle w, l\rangle_k$ and $SW\langle w', l'\rangle_{k'}$ are connected by an edge if and only if $l' = l + 1$, $w_0 w_1 \cdots w_{n-3}$ = $w'_0 w'_1 \cdots w'_{l-1} w'_{l+1} \cdots w'_{n-2}$, $k = w'_l$, and $k' = w_{n-2} + (m/2)$. For switch $SW\langle w, (n-1)\rangle$, port $SW\langle w, (n-1)\rangle_k$ is connected to processing node $P(p)$ if and only if $w_0 w_1 \cdots w_{n-2} = p_0 p_1 \cdots p_{n-2}$ and $k = p_{n-1}$.

An example of a 4-port 3-tree is shown in Figure 5. In Figure 5, we can see that the height of the 4-port 3-tree is 4. There are 16 processing nodes and 20 communication switches. Each communication switch has 4 communication ports labeled as 0, 1, 2, and 3. The set of processing nodes is {*P*(000), *P*(001), *P*(010), *P*(011), *P*(100), *P*(101), *P*(110), *P*(111), *P*(200), *P*(201), *P*(210), *P*(211), *P*(300), *P*(301), *P*(310), *P*(311)}. The sets of switches in level 0, 1, and 2 are {*SW*<00, 0>, *SW*<01, 0>, *SW*<10, 0>, *SW*<11, 0>}, {*SW*<00, 1>, *SW*<01, 1>, *SW*<10, 1>, *SW*<11, 1>, *SW*<20, 1>, *SW*<21, 1>, *SW*<30, 1>, *SW*<31, 1>}, and {*SW*<00, 2>, *SW*<01, 2>, *SW*<10, 2> , *SW*<11, 2>, *SW*<20, 2>, *SW*<21, 2>, *SW*<30, 2>, *SW*<31, 2>}, respectively. Ports $SW\langle w, l\rangle_k$ = <01, 0>$_1$ and $SW\langle w', l'\rangle_{k'}$ = <10, 1>$_3$ are connected by an edge since $l' = l + 1$, $w_0 w_1 \cdots w_{n-3}$ = $w'_0 w'_1 \cdots w'_{l-1} w'_{l+1} \cdots w'_{n-2}$ = 0, k = w'_l = 1, and k' = $w_{n-2} + (m/2)$ = 3. Port $SW\langle w, (n-1)\rangle_k$ = <20, 2>$_1$ is connected to processing node *P*(211) since $w_0 w_1 \cdots w_{n-2}$ = $p_0 p_1 \cdots p_{n-2}$ = 21 and k = p_{n-1} = 1.

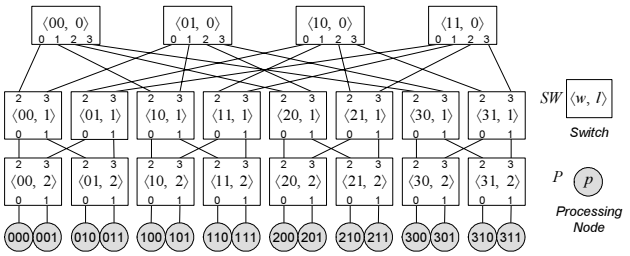


Figure 5: A 4-port 3-tree.

By replacing the communication switches and edges of an *m*-port *n*-tree as InfiniBand switches and InfiniBand links, respectively, we obtain an *m*-port *n*-tree InfiniBand network, denoted by *IBFT*(*m*, *n*). InfiniBand switches have 2 types of ports, internal and external ports. The internal port is a special port for management purpose. Port 0 of an InfiniBand switch is used as the internal port or referred as management port. The external ports numbered from 1 up to 254 are used to connect to other devices. Since port 0 of an InfiniBand switch is reserved for special purpose, when constructing *IBFT*(*m*, *n*), port $SW\langle w, l\rangle_k$ in an *m*-port *n*-tree is mapped to port *k*+1 of an InfiniBand switch. Also, IBA supports multiple links (endports) attached to the same processing node or other I/O devices, for simplicity, we assume that every processing node only contains one endport in this paper. Based on the constructed fat-tree-based InfiniBand network, we have the following definitions.

Definition 1: Given an *m*-port *n*-tree InfiniBand network, *IBFT*(*m*, *n*), for processing nodes $P(p = p_0 p_1 \cdots p_{n-1})$ and $P(p' = p'_0 p'_1 \cdots p'_{n-1})$, $gcp(P(p), P(p'))$ = $p_0 p_1 \cdots p_{\alpha-1}$ is the greatest common prefix of $P(p)$ and $P(p')$ if $p_0 p_1 \cdots p_{\alpha-1}$ = $p'_0 p'_1 \cdots p'_{\alpha-1}$ and $p_\alpha p_{\alpha+1} \cdots p_{n-1} \neq p'_\alpha p'_{\alpha+1} \cdots p'_{\alpha-1}$, where $\alpha \geq 0$ is the length of $gcp(P(p), P(p'))$. If $\alpha = 0$, it denotes that the labels of

two processing nodes have no common prefix.

Definition 2: Let *IBFT*(*m*, *n*) be an *m*-port *n*-tree InfiniBand network and $p_0 p_1 \cdots p_{\alpha-1}$ be the greatest common prefix of processing nodes *P*(*p*) and $P(p')$, the set of least common ancestors of processing nodes *P*(*p*) and $P(p')$, is define as $lca(P(p), P(p')) = \{ SW\langle w, l\rangle \mid w_0 w_1 \cdots w_{\alpha-1} = p_0 p_1 \cdots p_{\alpha-1}$ and $l = \alpha\}$.

Definition 3: Given an *m*-port *n*-tree InfiniBand network, *IBFT*(*m*, *n*), a greatest common prefix group, *gcpg*(*x*, α), is a set of processing nodes that have the same greatest common prefix *x* and $|x| = \alpha$. There are $(m/2)^{n-\alpha}$ processing nodes in an *gcpg*(*x*, α). Set *gcpg*(*x*, 0) is the set of all processing nodes, where *x* is a null string.

Definition 4: Let processing node $P(p) \in gcpg(x, \alpha)$, the *rank* of *P*(*p*) in *gcpg*(*x*, α) is defined as $rank(gcpg(x,\alpha), P(p)) = \sum_{i=\alpha}^{n-1} p_i \times (m/2)^{(n-1)-i} = p_\alpha \times (m/2)^{(n-1)-\alpha} + p_{\alpha+1} \times (m/2)^{(n-1)-(\alpha+1)} + \ldots + p_{n-1} \times (m/2)^0$, where $p = p_0 p_1 \cdots p_{n-1}$. The ranks of processing nodes in *gcpg*(*x*, α) are between 0 and $(m/2)^{n-\alpha} - 1$. Since *gcpg*(*x*, 0) contains all processing nodes in an InfiniBand network, the rank of a processing node *P*(*p*) in *gcpg*(*x*, 0) is also called the PID of *P*(*p*), denoted as PID(*P*(*p*)).

Let us give some examples to explain the above definitions. Given the 4-port 3-tree InfiniBand network shown in Figure 5(b), for processing nodes *P*(200) and *P*(211), *gcp*(*P*(200), *P*(211)) is 2 and *lca*(*P*(200), *P*(211)) is {*SW*<20, 1>, *SW*<21, 1>}. Both *P*(100) and *P*(111) are members of *gcpg*(2, 1). There are 4 processing nodes, *P*(200), *P*(201), *P*(210), and *P*(211), in group *gcpg*(2, 1). The ranks of *P*(200) and *P*(211) in *gcpg*(2, 1) are 0 and 3, respectively. PID(*P*(200)) = 8 and PID(*P*(211)) = 11.

4. The Routing Scheme for *m*-port *n*-tree InfiniBand Networks

Routing in an InfiniBand network is deterministic based on forwarding tables stored in InfiniBand switches. When the deterministic routing scheme is applied to topologies like fat-trees that offer multiple paths between any pair of processing nodes, the construction of forwarding tables is very important. If the forwarding tables are not well designed, the link congestion problem may occur and the advantage of high bandwidth offered by multiple paths of a fat-tree will be wasted. For example, in Figure 6, an 8-port 2-tree InfiniBand network *IBFT*(8, 2) is shown. Switches *i*, *j*, *k*, *l*, are the four least common ancestors of switches *x* and *y*. There are four paths between switches *x* and *y*. Assume that each processing node is associated with one LID. In order to evenly distribute the traffic between switches *x* and *y* on the four paths, *x*-*i*-*y*, *x*-*j*-*y*, *x*-*k*-*y*, *x*-*l*-*y*, we may generate forwarding tables for switches *x* and *y* as shown in Figure 7, where the numbers under processing nodes are their corresponding LIDs. For switch *x*, packets sent to processing nodes *E*, *F*, *G*, *H* will travel through paths *x*-*i*-*y*, *x*-*j*-*y*, *x*-*k*-*y*, *x*-*l*-*y*, respectively. Although the forwarding tables generated in Figure 7 can help dispersing traffic between switches *x* and *y* through four root switches, the link congestion problem may still occur when multiple source processing nodes are sending messages to the same destination processing node at the same time. For example, if processing nodes *A*, *B*, *C* and *D* in Figure 7 all send messages to processing node *E* at the same time, these messages will congest at port 5 of switch *x* as shown in Figure 9(a) since the link associated with port 5 can only send one message out at a given time. One solution to relief the link congestion problem is to assign multiple LIDs to the same processing node as shown in Figure 8. In Figure 8, if processing nodes *A*, *B*, *C* and *D* all send message to processing node *E* at the same time, since processing node *E* have four LIDs, 116, 117, 118 and 119, the DLID field of packets sent from processing node *A*, *B*, *C* and *D* to processing node *E* can be filled with 116, 117, 118, 119, respectively. By doing so, the traffic can be evenly distributed as shown in Figure 9(b).

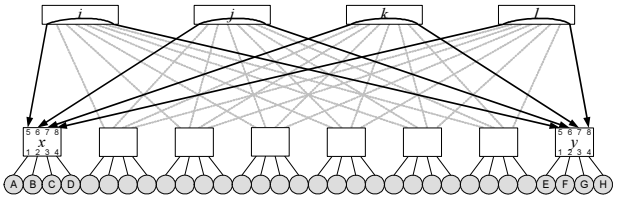


Figure 6: Routing on an 8-port 2-tree IB network.

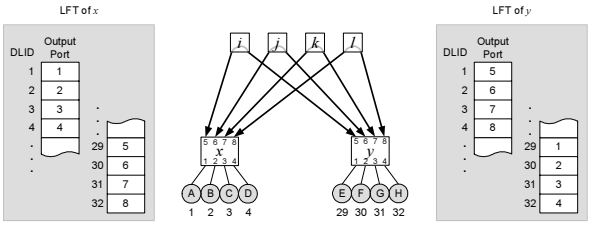


Figure 7: A single LID forwarding table assignment.

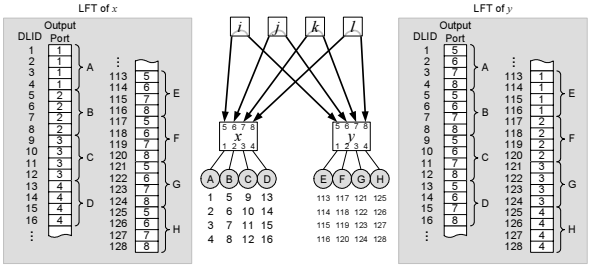


Figure 8: A multiple LIDs forwarding table assignment.

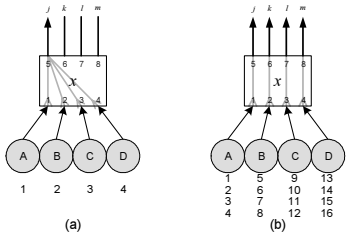


Figure 9: (a) A case of link congestion. (b) A solution of (a).

Based on the above analysis, we proposed a Multiple LID (MLID) routing scheme for the *m*-port *n*-tree InfiniBand networks. The MLID routing scheme consists of the processing node addressing scheme, the path selection scheme, and the forwarding table assignment scheme.

4.1 The Processing Node Addressing Scheme

Given an *m*-port *n*-tree InfiniBand network *IBFT*(*m*, *n*), in order to utilize the multiple paths offered by *IBFT*(*m*, *n*), every processing node in *IBFT*(*m*, *n*) is assigned a set of LIDs instead of an LID in our routing scheme. The set of LIDs assigned to each processing node is formed by the combination of one base LID and a LID Mask Control value *LMC*, where $LMC = \log_2 (m/2)^{n-1}$. For processing node $P(p = p_0 p_1 \cdots p_{n-1})$ in *IBFT*(*m*, *n*), the set of LIDs assigned to *P*(*p*), denoted by *LIDset*(*P*(*p*)), is {*BaseLID*(*P*(*p*)), *BaseLID*(*P*(*p*)) + 1, …, *BaseLID*(*P*(*p*)) + (2^{LMC}−1)}, where $BaseLID(P(p)) = 2^{LMC} \times \left(\sum_{i=0}^{n-1} p_i \times (m/2)^{n-(i+1)} \right) + 1$ is the base LID of *P*(*p*). There are 2^{LMC} LIDs in *LIDset*(*P*(*p*)), which indicates that there are maximal 2^{LMC} paths between any pair of processing nodes. Figure 10 shows an example of multiple LIDs assignment for each processing node in a 4-port 3-tree InfiniBand network. In Figure 10, for processing node *P*(300), *BaseLID*(*P*(300)) = 49. We have *LIDset*(*P*(300)) = {49, 50, 51, 52}.

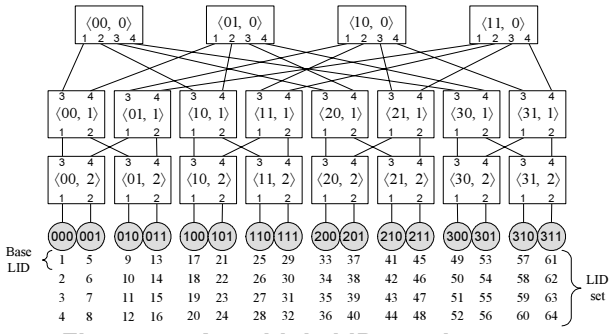


Figure 10: A multiple LIDs assignment.

4.2 The Path Selection Scheme

After each processing node is assigned a set of LIDs, the next problem is how to take the advantage of multiple LIDs of a processing node such that the link congestion can be relieved. Consider the communication between processing nodes *P*(*p*) and *P*(*p*′). Assume that $P(p = p_0 p_1 \cdots p_{n-1})$ and $P(p' = p'_0 p'_1 \cdots p'_{n-1})$ are in *gcpg*(*x*, α), but *P*(*p*) and *P*(*p*′) are in *gcpg*(xp_α, α+1) and *gcpg*(xp'_α, α+1), respectively, where $xp_\alpha \neq xp'_\alpha$. Both *gcpg*($xp_\alpha$, α+1) and *gcpg*($xp'_\alpha$, α+1) groups have $(m/2)^{n-(\alpha+1)}$ processing nodes apiece. For each processing node in *gcpg*(xp_α, α+1), there are $(m/2)^{n-(\alpha+1)}$ paths to each processing node in *gcpg*(xp'_α, α+1). If each processing node in *gcpg*(xp_α, α+1) wants to send messages to the same processing node in *gcpg*(xp'_α, α+1), say *P*(*p*′), the processing nodes with ranks 0, 1, …, $(m/2)^{n-(\alpha+1)} - 1$ can choose *BaseLID*(*P*(*p*′)), *BaseLID*(*P*(*p*′)) …, *BaseLID*(*P*(*p*′)) + $(m/2)^{n-(\alpha+1)} - 1$ as the LID of *P*(*p*′), respectively. In this way, the multiple paths between processing nodes in two groups can be fully utilized.

An example is shown in Figure 11. In Figure 11, we have *gcpg*(0, 1) = {*P*(000), *P*(001), *P*(010), *P*(011)} and *gcpg*(3, 1) = {*P*(300), *P*(301), *P*(310), *P*(311)}. If each processing node in *gcpg*(0, 1) wants to send message to *P*(300) in *gcpg*(3, 1), *P*(000), *P*(001), *P*(010), and *P*(011) will select 49, 50, 51, and 52 as the LID of *P*(300), respectively. Packets send from *P*(000), *P*(001), *P*(010), and *P*(011) to *P*(300) will go through routes *Q*, *R*, *S*, and *T* to *P*(300).

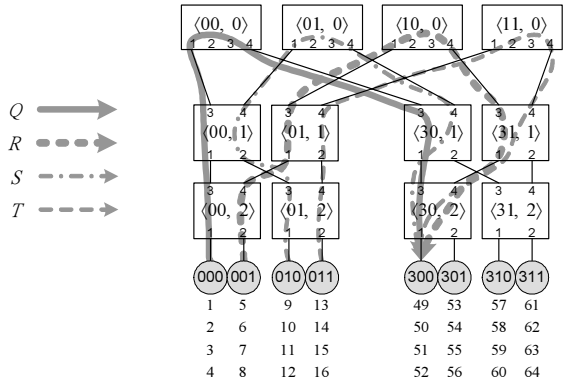


Figure 11: Each processing nodes in *gcpg*(0, 1) selects different path to send messages to *P*(300) according to different LIDs of *P*(300).

4.3 The Forwarding Table Assignment Scheme

After the path selection scheme is performed, the next task is to setup the forwarding table in each InfiniBand switch such that a message sent from one processing node to another will follow the path we set in the path selection scheme. As mentioned in Section 2.2, the InfiniBand network uses the packet-switching routing mechanism. Messages are sent as packets. In each packet, there is a DLID field that specifies the destination LID of the packet. Within a switch, each packet is forwarded to an output port based on the DLID field of the packet and the forwarding table of the switch. The forwarding table of each switch is set at the subnet initialization process. After the subnet initialization process, the packet routing behavior is fixed unless a subnet reconfiguration or for other purpose that the subnet manager re-assigns forwarding table for each switch.

Given an m-port n-tree InfiniBand network $IBFT(m, n)$, a switch $SW\langle w, l\rangle$ of $IBFT(m, n)$, and a packet whose DLID field is lid, when the packet arrives in switch $SW\langle w, l\rangle$, the output port $SW\langle w, l\rangle_k$ of the packet can be determined based on the construction of $IBFT(m, n)$, the processing node assignment scheme, and the path selection scheme. We have the following two cases.

Case 1: If the processing node $P(p = p_0 p_1 \cdots p_{n-1})$ that owns the lid can be reached downward from $SW\langle w, l\rangle$, then k can be determined by the following equation

$$k = p_l + 1 . \qquad (1)$$

For $SW < w = w_0 w_1 \cdots w_{n-2}, l >$, processing node $P(p = p_0 p_1 \cdots p_{n-1})$ that owns the lid can be reached downward from $SW\langle w, l\rangle$ if $\mathrm{PID}(P(p)) = \left\lfloor \frac{(lid-1)}{\left(\frac{m}{2}\right)^{n-1}} \right\rfloor$ and $w_0 w_1 \cdots w_{l-1} = p_0 p_1 \cdots p_{l-1}$. The conversion between $\mathrm{PID}(P(p))$ and $P(p = p_0 p_1 \cdots p_{n-1})$ can be done either by table lookup or by arithmetic operations.

Case 2: If the processing node that owns the lid can not be reached downward from $SW\langle w, l\rangle$, then k can be determined by the following equation

$$k = \left(\left\lfloor \frac{(lid-1)}{\left(\frac{m}{2}\right)^{(n-1)-l}} \right\rfloor \bmod \left(\frac{m}{2}\right) \right) + \left(\frac{m}{2}\right) + 1 \qquad (2)$$

To verify the correctness of Equations (1) and (2), let us take Figure 11 as an example. In Figure 11, assume that processing nodes $P(000)$, $P(001)$, $P(010)$, and $P(011)$ want to send messages to processing node $P(300)$. According to the path selection scheme, packets sent from $P(000)$, $P(001)$, $P(010)$, and $P(011)$ to $P(300)$ will go through paths Q, R, S, and T, respectively. When a packet is sent from $P(000)$ to $P(300)$ through path Q, the DLID of the packet is 49 and ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$ $SW\langle 00, 0\rangle_4$, $SW\langle 30, 1\rangle_3$, $SW\langle 30, 1\rangle_1$, $SW\langle 30, 2\rangle_3$, and $SW\langle 30, 2\rangle_1$ will be traversed in sequence. When the packet arrives in switch SW<00, 2>, lid = 49 matches case 2 and the output port of the packet is k = 3. When the packet arrives in switch SW<00, 1>, lid = 49 matches case 2 and the output port of the packet is k = 3. When the packet arrives in switch SW<00, 0>, lid = 49 matches case 1 and the output port of the packet is k = 4. When the packet arrives in switch SW<30, 1>, lid = 49 matches case 1 and the output port of the packet is k = 1. When the packet arrives in switch SW<30, 2>, lid = 49 matches case 1 and the output port of the packet is k = 1. From the above analysis, we can see that Equations (1) and (2) can correctly setup path Q for the packet sent from $P(000)$ to $P(300)$. For paths R, S, and T, we can obtain similar results.

5. Performance Evaluation

In order to evaluate the performance of the proposed routing scheme, we have developed an InfiniBand network simulator. The simulator was written in JAVA. Two routing schemes, Single LID (SLID) and Multiple LID (MLID) were simulated for performance evaluation. In the SLID scheme, each processing node $P(p)$ is associated with one LID, $\mathrm{PID}(P(p))$. The forwarding table assignment of the SLID scheme is base on the consideration of evenly distributing possible traffic over available paths. The MLID routing scheme is the proposed routing scheme.

5.1 The IBA Network Model

In the simulator, an InfiniBand subnet, which is similar to that of [11], is modeled. There are two basic components in the IBA subnet, endnodes (processing nodes) and switches that are connected by IBA links. The endnodes act as the message (data) producers and consumers while the switches are responsible for correctly forward messages from source endnodes to destination endnodes. InfiniBand packets are routed through switches by forwarding table lookup. Forwarding tables store the output port information for each destination LIDs in the subnet.

IBA specification allows arbitrary topology given by user. In the simulator, an m-port n-tree InfiniBand network is modeled as an IBA subnet in which there are $2\times(m/2)^n$ processing nodes and $(2n-1)\times\left(m/2\right)^{n-1}$ switches. Each switch has m bi-directional communication ports that are attached to either switches or processing nodes. Each switch has a crossbar connecting

all the input ports to the output ports that allowing multiple packets to be transferred without interference. According to IBA, InfiniBand switches can support up to 16 virtual lanes, (one management lane and at most 15 data lanes). In the simulator we have also implemented the virtual lanes mechanism, and each virtual lane of a port uses separate input and output buffers. The size of input buffer and output buffer of a virtual lane are the same for a packet, means that the buffer can only store a packet at a time. Each switch in an *m*-port *n*-tree InfiniBand network is assigned a linear forwarding table according to the routing scheme. Packets arrive in input ports of a switch will be forwarded through the crossbar to their corresponding output ports based on the forwarding table lookup. The packet will be forwarded from an input port buffer through the crossbar to a corresponding output port buffer if the output port buffer is available. Otherwise the packet must wait in the input port buffer until the output buffer is available. The virtual cut-through switching technique [4] is used. To do the flow control, we have implemented the credit based link level flow control mechanism of IBA.

5.2 The Simulation Results

To evaluate the performance of the SLID and MLID routing schemes, we have run these two schemes on different sizes of *m*-port *n*-tree InfiniBand networks as shown in Table 1. We assume that the flying time of a packet between devices (endnode-to-switch and switch-to-switch) is 20ns. The routing time of a packet from one input port to one output port of the crossbar in a switch is 100ns, including forwarding table lookup, arbitration, and message startup time. The byte injection rate is 4ns assume that a 1X link configuration (2.5 Gbps) is used. The packet size is 32 bytes. The number of virtual lanes is set to 1, 2, and 4. For each virtual lane, the input/output buffer size is 32 bytes.

Table 1: The test samples of *m*-port *n*-tree InfiniBand networks.

Switch Ports (*m*)	Level (*n*)	InfiniBand Switches	Processing Nodes
4	4	56	32
8	3	80	128
16	3	320	1024
32	2	48	512

Two traffic patterns were simulated. One is a uniform traffic pattern in which each processing node sends a packet to a randomly selected destination processing node in a given time unit based on a data injection rate. The other is 10% centric traffic pattern in which each processing node has a 10% chance to select one particular destination processing node, that is, 10 out of 100 packets will be sent from all source processing nodes to this particular processing node. In each simulation run, we assume that the packet generation rate is constant and the same for all processing nodes. Different packet generation rates are simulated in order to get the performance data from low network load to saturation. One of the collected performance data is the accepted traffic measured in bytes/nanosecond per processing node (bytes/ns/processing node). The other performance data is the average message latency of all received packets, measured in nanosecond. The message latency is the time elapsed since the packet transmission is initiated until the packet is received at the destination node.

The simulation results are shown in Figures 12 to 15. In Figures 12 to 15, the *x*-axis indicates the accepted traffic and the *y*-axis indicates the average message latency. From Figures 12 to 15, we have the following observations:

Observation 1: For the uniform traffic pattern, if the port number of a switch is not large (4-port or 8-port), the throughput of the MLID scheme is a little higher or equal to that of the SLID scheme for all simulated cases. If the port number of a switch is large (16-port or 32-port), the throughput of the MLID scheme is higher than that of the SLID scheme for all simulated cases.

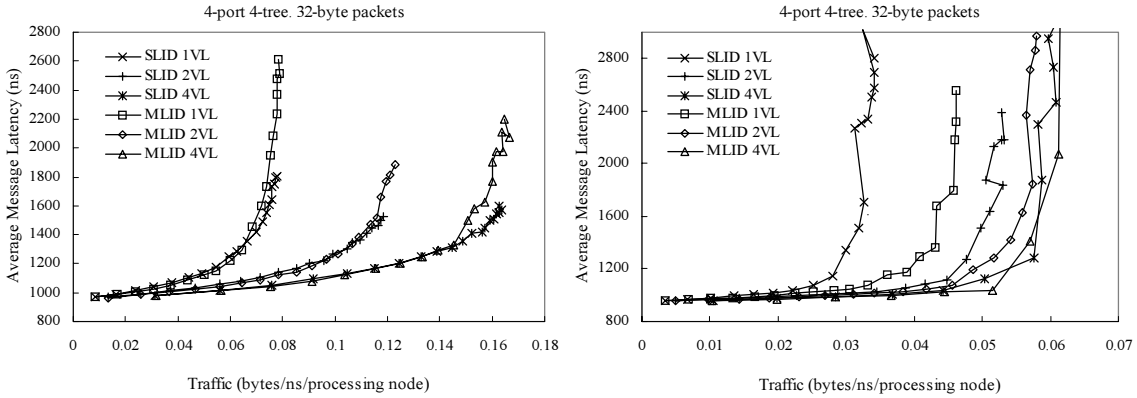
Observation 2: For the uniform traffic pattern, when the network traffic is low, the average message latency of the MLID scheme, in general, is less than or equal to that of the SLID scheme. When the traffic is high, the average message latency of the MLID scheme under the same offered traffic (packet generation rate) is higher than that of the SLID scheme. This is because the bandwidth utilization of the MLID scheme is higher than that of the SLID scheme, that is, in a given time period, the total packets reside in the network under the MLID scheme is more than that under the SLID scheme. These packets will spent more time waiting in the input buffer until output buffer is available in a switch.

Observation 3: For the 10% centric traffic pattern, the throughput of the MLID scheme is much higher than that of the SLID scheme when only one virtual lane is available. When more than one virtual lane is available, the throughput of the MLID scheme is still higher than that pf the SLID scheme for all simulated cases. If the port number of a switch is large (16-port or 32-port), the throughput of the MLID scheme with one virtual lane is higher than that of the SLID scheme with two virtual lanes.

Observation 4: For the 10% centric traffic pattern, if the port number of a switch is not large (4-port or 8-port), the average message latency of the MLID scheme is less than that of the SLID scheme when only one virtual lane is available. This indicates that the MLID scheme is more capable of utilizing the offered bandwidth than the SLID scheme for this case. For the case where more than one virtual lane is available, we have similar observation as

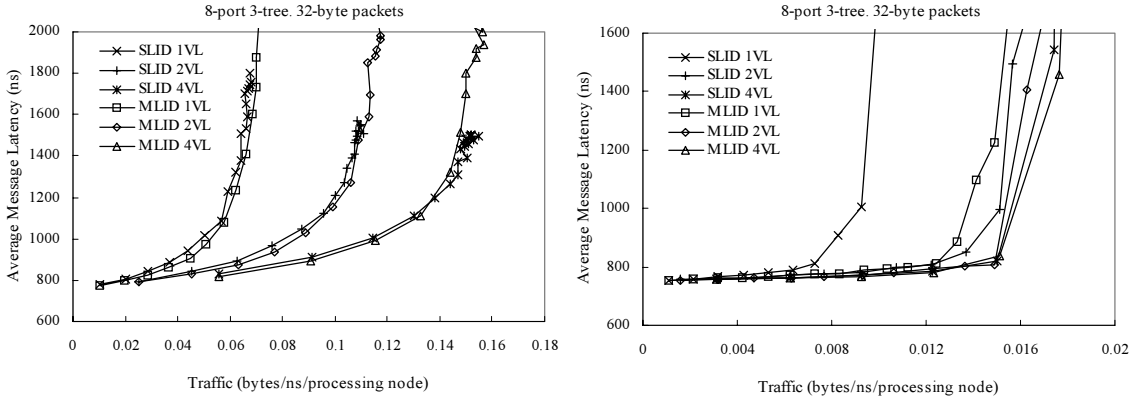
that of Observation 2.

Observation 5: The MLID scheme is more capable of utilizing the offered bandwidth than the SLID scheme for a given *m*-port *n*-tree InfiniBand network. The performance improvement compare to the SLID scheme is more noticeable while a network size is getting larger.



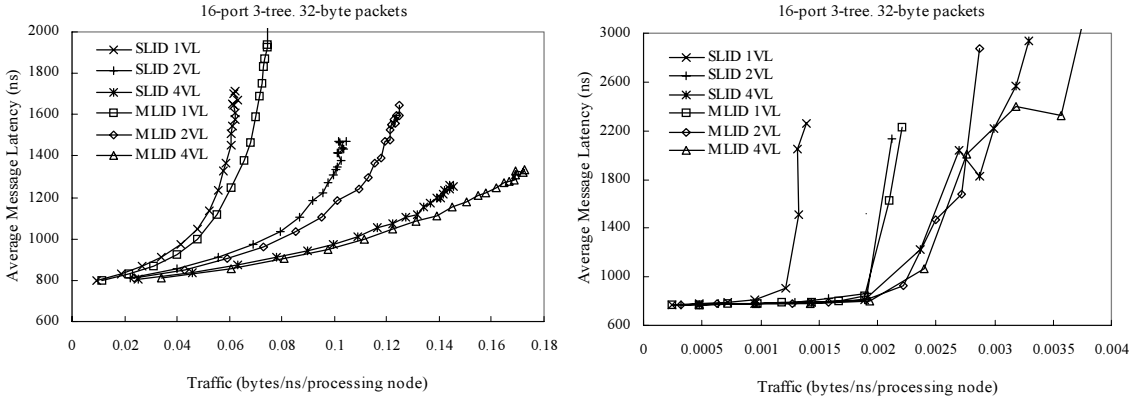
(a) Uniform traffic pattern. (b) 10% centric traffic pattern.

Figure 12: Simulation results of the 4-port 4-tree InfiniBand network.



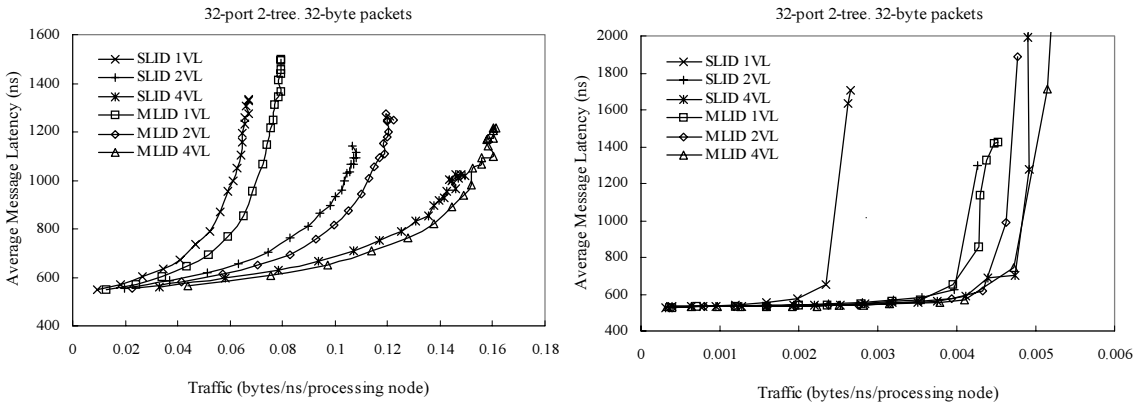
(a) Uniform traffic pattern. (b) 10% centric traffic pattern.

Figure 13: Simulation results of the 8-port 3-tree InfiniBand network.



(a) Uniform traffic pattern. (b) 10% centric traffic pattern.

Figure 14: Simulation results of the 16-port 3-tree InfiniBand network.



(a) Uniform traffic pattern. (b) 10% centric traffic pattern.

Figure 15: Simulation results of the 32-port 2-tree InfiniBand network.

6. Conclusions

In this paper, we have shown how to construct fat-tree-based InfiniBand networks based on *m*-port *n*-trees. We also designed an efficient routing scheme for the constructed fat-tree-based InfiniBand networks. The proposed routing scheme consists of processing node addressing scheme, path selection scheme and forwarding table assignment scheme. From the simulation results, we have the following remarks:

Remark 1: The throughput of the MLID scheme is higher than that of the SLID scheme for all simulated cases.

Remark 2: When the network traffic is low, the average message latency of the MLID scheme, in general, is less than or equal to that of the SLID scheme. When the traffic is high, the average message latency under the same offered traffic of the MLID scheme is higher than that of the SLID scheme.

Remark 3: The MLID scheme is able to utilize the offered bandwidth by *m*-port *n*-tree efficiently when the network scaling up to higher dimensions (higher *n* value) that contains more switches and processing nodes.

References

[1] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks - An Engineering Approach*, IEEE CS Press, 1997.

[2] Kai Hwang, *Advanced Computer Architecture – Parallelism, Scalability, Programmability*, McGraw-Hill, 1993

[3] InfiniBand™ Trade Association, *InfiniBand™ Architecture Specification Volume 1, Release 1.1*, November 2002.

[4] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, Vol. 3, 1979, pp. 267-286.

[5] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.

[6] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transactions on Computers*, vol. 34, no.10, October 1985, pp. 892-901.

[7] P. López, J. Flich, and J. Duato, "Deadlock-Free Routing in InfiniBand™ through Destination Renaming," in *Proceedings of the International Conference on Parallel Processing*, *ICPP '01*, Sept. 2001, pp. 427-434.

[8] L. M. Ni, Y. Gui, and S. Moore, "Performance Evaluation of Switch-Based Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 5, May 1997, pp. 462-474.

[9] F. Petrini and M. Vanneschi, "Network Performance under Physical Constraints," in *Proceedings of the International Conference on Parallel Processing 1997*, *ICPP'97*, August 1997, pp. 34-43.

[10] F. Petrini and M. Vanneschi, "*k*-ary *n*-trees: High Performance Networks for Massively Parallel Architectures," in *Proceedings of the 11th International Parallel Processing Symposium*, *IPPS'97*, April 1997, pp. 87-93.

[11] J. C. Sancho, J.Flich, A. Robles, P. López, and J. Duato, "Analyzing the Influence of Virtual Lanes on the Performance of InfiniBand Networks," in *Proceedings of the International Parallel and Distributed Processing Symposium (CD-ROM)*, *IPDPS'02*, April. 2002.

[12] J. C. Sancho, A. Robles, J.Flich, P. López, and J. Duato, "Effective Methodology for Deadlock-Free Minimal Routing in InfiniBand Networks," in *Proceedings of the International Conference on Parallel Processing*, *ICPP '02*, Aug. 2002, pp. 48-57.

[13] J. C. Sancho, A. Robles, and J. Duato, "Effective Strategy to Compute Forwarding Tables for InfiniBand Networks," in *Proceedings of the International Conference on Parallel Processing*, *ICPP '01*, Sept. 2001, pp. 48-57.

[14] M. Valerio, L. Moser, and P. Melliar-Smith, "Recursively Scalable Fat-Trees as Interconnection Networks," in *Proceedings of the 13th IEEE International Phoenix Conference on Computers and Communications*, April 1994, pp. 40-46.