J Supercomput (2007) 40: 333–352
DOI 10.1007/s11227-006-0019-y

Hardware supported multicast in fat-tree-based InfiniBand networks

Jiazheng Zhou · Xuan-Yi Lin · Yeh-Ching Chung

Published online: 16 March 2007

Abstract The multicast operation is a very commonly used operation in parallel applications. It can be used to implement many collective communication operations as well. Therefore, its performance will affect parallel applications and collective communication operations. With the hardware supported multicast of the InfiniBand Architecture (IBA), in this paper, we propose a *cyclic* multicast scheme for fat-tree-based (m-port n-tree) InfiniBand networks. The basic concept of the proposed cyclic multicast scheme is to find the union sets of the output ports of switches in the paths between the source processing node and each destination processing node in a multicast group. Based on the union sets and the path selection scheme, the forwarding table for a given multicast group can be constructed. We implement the proposed multicast scheme along with the OpenSM multicast scheme and the unicast scheme on an m-port n-tree InfiniBand network simulator. Several one-to-many, many-to-many, many-to-all, and all-to-many multicast cases are simulated. The simulation results show that the proposed multicast scheme outperforms the unicast scheme for all simulated cases. For one-to-many case, the performance of the cyclic multicast scheme is the same as that of the OpenSM multicast scheme. For many-to-many and all-to-many cases, the cyclic multicast scheme outperforms the OpenSM multicast scheme. For many-to-all case, the performance of the cyclic multicast scheme is a little better than that of the OpenSM multicast scheme.

Keywords Multicast · Unicast · InfiniBand · Fat-tree · Union operation · Cyclic · OpenSM

J. Zhou (✉) · X.-Y. Lin · Y.-C. Chung
Department of Computer Science, National Tsing-Hua University, Hsinchu 300, Taiwan, R.O.C.
e-mail: jzzhou@cs.nthu.edu.tw

X.-Y. Lin
e-mail: xylin@cs.nthu.edu.tw

Y.-C. Chung
e-mail: ychung@cs.nthu.edu.tw

1 Introduction

Interconnection networks in cluster systems have great impact on the performance of communication-bounded applications. The InfiniBand Architecture (IBA) [6] is a new industry-standard architecture for server I/O and inter-server communication. The IBA defines a switch-based, point-to-point interconnection network that enables high-speed, low-latency communication between connected devices. Due to the characteristics of the IBA, it is very attractive to use the IBA as the interconnection network of a cluster system.

The multicast operation [1, 4, 5, 9, 13, 16–18, 21–23] is a very common used operation in cluster systems. It can improve the performance of interconnection communication of processors [14] and can be used to implement many efficient collective communication operations. It can also be used in distributed shared memory (DSM) systems [2, 11] to enhance their performance. For data parallel languages, multicast is the fundamental of several operations such as data replication [19] and barrier synchronization [25]. For parallel applications, one can get the benefits from the use of multicast operation [3, 7].

Since the InfiniBand Architecture supports hardware multicast, one can take advantage of this feature to speedup the multicast operation. OpenSM [15] is an implementation of subnet manager. In OpenSM, it implements a hardware supported multicast scheme by using a spanning tree approach to construct the multicast paths for a given multicast group. This scheme can be applied to any network topology. However, its performance may not be satisfied since it does not take the characteristics of a network topology into account.

In this paper, we focus on the fat-tree topology [8, 10, 20, 24] used in most scalable cluster systems nowadays. We propose a *cyclic* multicast scheme for the m-port n-tree (a fat-tree) InfiniBand networks [12] based on the hardware supported multicast feature of the IBA and the characteristics of m-port n-tree fat-trees. The construction of the cyclic multicast scheme consists of three parts: the processing node addressing scheme, the path selection scheme, and the forwarding table assignment scheme. In the processing node addressing scheme, each processing node is assigned a set of LIDs. In the path selection scheme, for a given destination processing node, source processing nodes are divided into cyclic groups based on the cyclic grouping policy. Source processing nodes in the same group choose the same LID of the destination processing node to perform the path selection. In the forwarding table assignment scheme, a one-to-one forwarding table is set first according to the path selection scheme. Then, the multicast forwarding table can be set according to the one-to-one forwarding table and the union operations.

To evaluate the proposed method, we implement the cyclic multicast scheme, the OpenSM multicast scheme, and a unicast scheme on an m-port n-tree InfiniBand network simulator that is written in Java. The combinations of different messages sizes, different numbers of multicast source nodes (traffic load), and different sizes of multicast groups are used as test samples. The simulation results show that the proposed cyclic multicast scheme outperform the unicast schemes for all test cases. The higher the message size, the number of multicast source nodes, and the size of the multicast group, the better speedup can be expected from the proposed multicast schemes. For one-to-many case, the performance of the cyclic multicast scheme is the same as that of the OpenSM multicast method. For many-to-many and all-to-many cases, the cyclic multicast scheme outperforms the OpenSM multicast method. For many-to-all case, the performance of the cyclic multicast scheme is a little better than that of the OpenSM multicast method.

The rest of this paper is organized as follows. Section 2 will introduce the fat-tree-based InfiniBand networks. The proposed multicast schemes will be described in Sect. 3. Section 4 will give the simulation results for the proposed multicast schemes. The conclusions will be given in Sect. 5.

2 Preliminaries

2.1 InfiniBand Architecture (IBA)

The InfiniBand Architecture (IBA) is a new industry-standard architecture for server I/O and inter-server communication. The IBA is designed around a point-to-point, switched I/O fabric, whereby end node devices are interconnected by cascaded switch devices [10]. An InfiniBand network can be divided into subnets. There is one or several subnet manager (SM) in an InfiniBand subnet. The subnet manager is responsible for the configuration and the control of a subnet. A Local Identifier (LID) is an address assigned to an endport by the subnet manager during the subnet initialization process. LID is unique within an InfiniBand subnet.

The InfiniBand network is a packet-switching 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. An example is illustrated in Fig. 1.

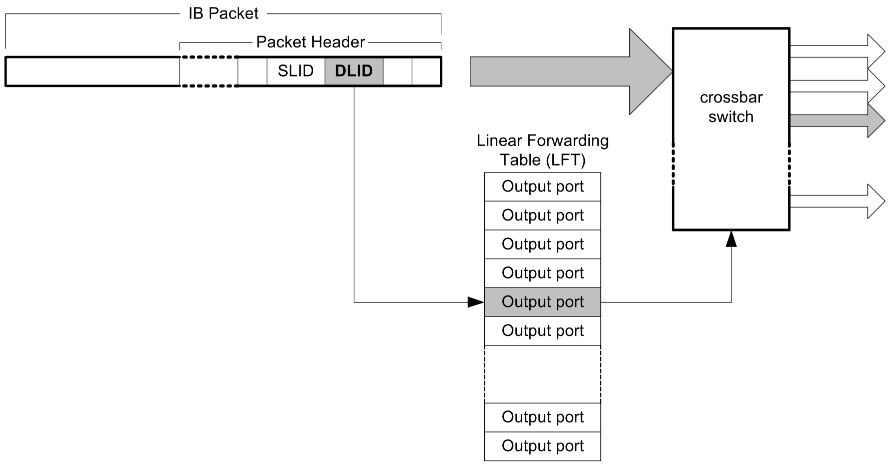


Fig. 1 The switch uses the linear forwarding table (LFT) to determine the output port according to the DLID field in the packet

Since the mapping between DLID and output port is one-to-one, in order to support multiple paths, the 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).

The IBA also supports hardware multicast. In the IBA, each multicast group is assigned a multicast LID and a global identifier (GID) by the subnet manager. The subnet manager will setup the forwarding table of each switch for each multicast group according its LID and GID. The range of the LID is divided into two parts, the unicast LID range and the multicast LID range. Each multicast group is identified by a unique GID. To perform a multicast operation in an InfiniBand network, the source processing node uses the multicast LID and the GID of a multicast group to send packets. When a switch receives a multicast packet, it replicates the packet and forwards the packet to the corresponding output ports according to its forwarding table. When a processing node joins or leaves a multicast group, the subnet manager will send the information to switches and update corresponding forwarding tables.

2.2 The m-port n-tree InfiniBand networks

In [12], we have proposed an m-port n-tree InfiniBand network $\mathit{IBFT}(m,n)$. It has the following characteristics:

1. The height of $\mathit{IBFT}(m,n)$ is $n+1$.
2. $\mathit{IBFT}(m,n)$ consists of $2 \times (m/2)^n$ processing nodes and $(2n-1) \times (m/2)^{n-1}$ InfiniBand switches.
3. Each switch has m ports.

A processing node in $\mathit{IBFT}(m,n)$ is labeled as $P(p = p_0 p_1 \cdots p_{n-1})$, where $p \in \{0, 1, \ldots, m-1\} \times \{0, 1, \ldots, (m/2)-1\}^{n-1}$. An InfiniBand switch in $\mathit{IBFT}(m,n)$ is labeled as $SW\langle w = w_0 w_1 \cdots w_{n-2}, l\rangle$, where $l \in \{0, 1, \ldots, n-1\}$ is the level of the switch and

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

Let $SW\langle w, l\rangle_k$ denote the kth port of $SW\langle w, l\rangle$, where $k = 1, 2, \ldots, m$. 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 $l' = l+1$, $w_0 w_1 \cdots w_{n-3} = w'_0 w'_1 \cdots w'_{l-1} w'_{l+1} \cdots w'_{n-2}$, $k = w'_l + 1$, and $k' = w_{n-2} + (m/2) + 1$. 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 $k = p_{n-1} + 1$. An example is shown in Fig. 2.

3 The proposed multicast scheme

A multicast operation can be either one-to-many or many-to-many. In this paper, we will discuss the one-to-many case. For many-to-many multicast, it can be implemented as many one-to-many multicasts. An example of hardware supported multicast of the IBA is shown in Fig. 3. In Fig. 3a, an 8-port 2-tree InfiniBand network,

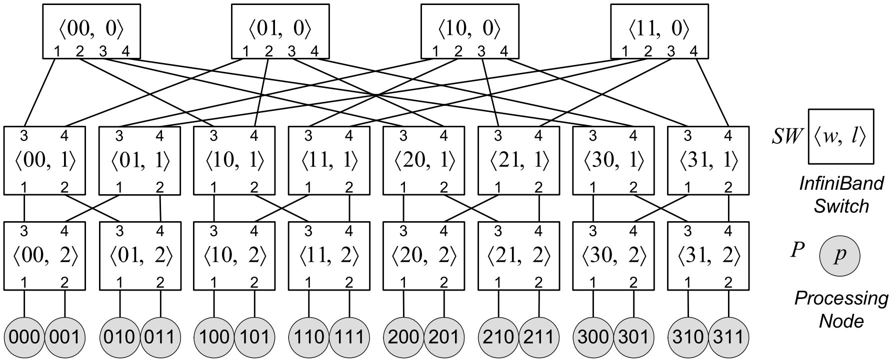


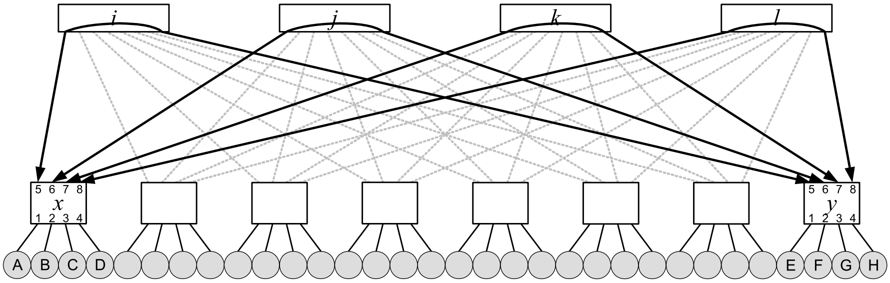
Fig. 2 An example of a 4-port 3-tree InfiniBand network

IBFT(8, 2), is shown. To simplify the presentation, we use the set of DLIDs of a multicast group to indicate the multicast LID of the multicast group. In Fig. 3, if processing node A wants to send a message to processing nodes E, F, and G, it needs to perform three send operations when the unicast operation is used. With hardware supported multicast in the IBA, the forwarding tables of switches can be set as shown in Fig. 3b. Based on the forwarding tables shown in Fig. 3b, processing node A sends only one packet with multicast LID α to processing nodes E, F, and G. If we change the forwarding tables shown in Fig. 3c, we can see that processing nodes E, F, and G will receive the same packet three times.

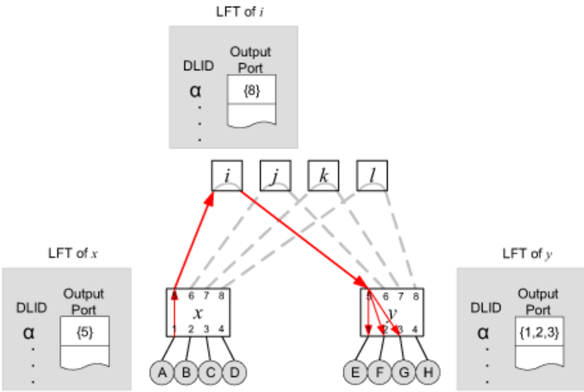
From Fig. 3, we can see that it is important to setup the forwarding tables of switches correctly. In Fig. 3, we observe that if a packet is duplicated in the descending phase, the duplication will result in the case shown in Fig. 3b, that is, the multicast is performed correctly. If a packet is duplicated in the ascending phase, the duplication will result in the case shown in Fig. 3c, that is, the multicast is not performed correctly. From the above observations, we propose a multicast scheme based on the MLID routing scheme [12] to correctly setup the forwarding tables of switches. The proposed multicast scheme consists of three sub-schemes, the processing node addressing scheme, the path selection scheme, and the forwarding table assignment scheme. We need the following definitions when discussing these three schemes.

Definition 1 Given an m-port n-tree InfiniBand network, $IBFT(m,n)$, for processing nodes $P(p = p_0p_1\cdots p_{n-1})$ and $P(p' = p'_0p'_1\cdots p'_{n-1})$, $gcp(P(p), P(p')) = p_0p_1\cdots p_{\alpha-1}$ is the greatest common prefix of $P(p)$ and $P(p')$ if $p_0p_1\cdots p_{\alpha-1} = p'_0p'_1\cdots p'_{\alpha-1}$ 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.

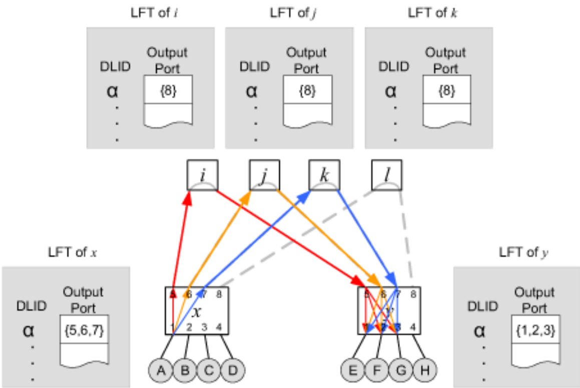
Definition 2 Let $IBFT(m,n)$ be an m-port n-tree InfiniBand network and $p_0p_1\cdots p_{\alpha-1}$ be the greatest common prefix of processing nodes $P(p)$ and $P(p')$, the set of least common ancestors of processing nodes $P(p)$ and $P(p')$, is defined as $lca(P(p), P(p')) = \{SW\langle w, l\rangle \mid w_0w_1\cdots w_{\alpha-1} = p_0p_1\cdots p_{\alpha-1}$ and $l = \alpha\}$.



(a) An example of 8-port 2-tree InfiniBand network



(b) Correct setting



(c) Incorrect setting

Fig. 3 An example of a multicast in an 8-port 2-tree InfiniBand network

Definition 3 Given an m-port n-tree InfiniBand network, $IBFT(m,n)$, a greatest common prefix group, $gcpg(x,\alpha)$, 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 $gcpg(x,\alpha)$. 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 $gcpg(x,\alpha)$ is defined as $\mathrm{rank}(gcpg(x,\alpha), P(p)) = \sum_{i=\alpha}^{n-1} p_i \times (m/2)^{(n-1)-i} = p_\alpha \times (m/2)^{(n-1)-\alpha} + p_{\alpha+1} \times (m/2)^{(n-1)-(\alpha+1)} + \cdots + p_{n-1} \times (m/2)^0$, where $p = p_0p_1\cdots p_{n-1}$. The ranks of processing nodes in $gcpg(x,\alpha)$ 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 Fig. 2, for processing nodes $P(200)$ and $P(211)$, $gcp(P(200), P(211))$ is 2 and $lca(P(200), P(211))$ is $\{SW\langle 20, 1\rangle, SW\langle 21, 1\rangle\}$. Both $P(100)$ and $P(111)$ are members of $gcpg(1,1)$. 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 $PID(P(211)) = 11$.

3.1 The processing node addressing scheme

Given an m-port n-tree InfiniBand network $IBFT(m,n)$, in the multicast scheme, every processing node in $IBFT(m,n)$ is assigned a set of LIDs. 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 $P(p = p_0p_1\cdots p_{n-1})$ in $IBFT(m,n)$, the set of LIDs assigned to $P(p)$, denoted by $LIDset(P(p))$, is $\{BaseLID(P(p)), BaseLID(P(p)) + 1, \ldots, BaseLID(P(p)) + (2^{LMC} - 1)\}$, where $BaseLID(P(p)) = 2^{\mathrm{LMC}} \times (\sum_{i=0}^{n-1} p_i \times (m/2)^{n-(i+1)}) + 1$ 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 4 shows an example of multiple LIDs assignment for each processing node in a 4-port 3-tree

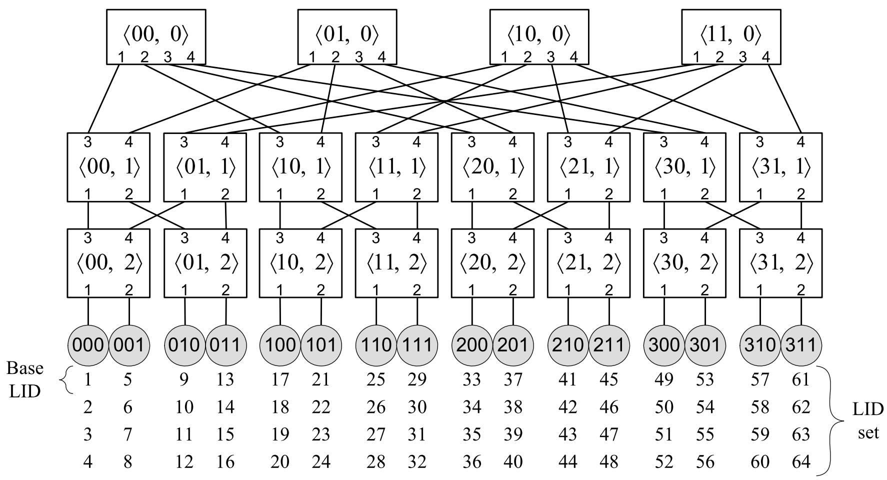


Fig. 4 A multiple LIDs assignment

InfiniBand network. In Fig. 4, for processing node $P(300)$, $BaseLID(P(300)) = 49$. We have $LIDset(P(300)) = \{49, 50, 51, 52\}$.

3.2 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 duplication of a packet will not occur in the ascending phase. We propose a *cyclic* path selection scheme according to the cyclic grouping policy. The grouping policy is to decide what processing nodes are in the same group for a given destination processing node $P(p)$. For the processing nodes in the same group, they will send messages to the destination processing node $P(p)$ by choosing the same LID of $P(p)$.

Given an m-port n-tree InfiniBand network $IBFT(m, n)$, for a destination processing node $P(p = p_0 p_1 \cdots p_{n-1})$, source processing nodes $P(s_1)$ and $P(s_2)$ are in the same cyclic group $CG(P(p), l, y)$ if the following two rules are satisfied.

Rule 1: The level l of the least common ancestors $lca(P(p), P(s_1))$ is the same as that of $lca(P(p), P(s_2))$.

Rule 2: $P(s_1)$ and $P(s_2)$ have the same common suffix y and $|y| = n - l - 1$.

The cyclic path selection scheme is performed as follows. For a destination processing node $P(p = p_0 p_1 \cdots p_{n-1})$ and a source processing node $P(p' = p'_0 p'_1 \cdots p'_{n-1})$ in $CG(P(p), l, y)$, when $P(p')$ wants to send messages to $P(p)$, it will select $BaseLID(P(p)) + \mathrm{rank}(gcpg(p'_0 p'_1 \cdots p'_l, l + 1), P(p'))$ as the LID of $P(p)$.

Figure 5 shows an example of the cyclic path selection scheme for a 4-port 3-tree InfiniBand network $IBFT(4, 3)$. Given a destination processing node $P(200)$, we can divide the source processing nodes into cyclic groups $CG(P(200), 0, 00) = \{P(000), P(100), P(300)\}$, $CG(P(200), 0, 01) = \{P(001), P(101), P(301)\}$, $CG(P(200), 0, 10) = \{P(010), P(110), P(310)\}$, $CG(P(200), 0, 11) = \{P(011), P(111), P(311)\}$, $CG(P(200), 1, 0) = \{P(210)\}$, $CG(P(200), 1, 1) = \{P(211)\}$, and $CG(P(200), 2, \varepsilon) = \{P(201)\}$ based on the cyclic grouping policy, where ε is a null string. Assume that there are four source processing nodes $P(000)$,

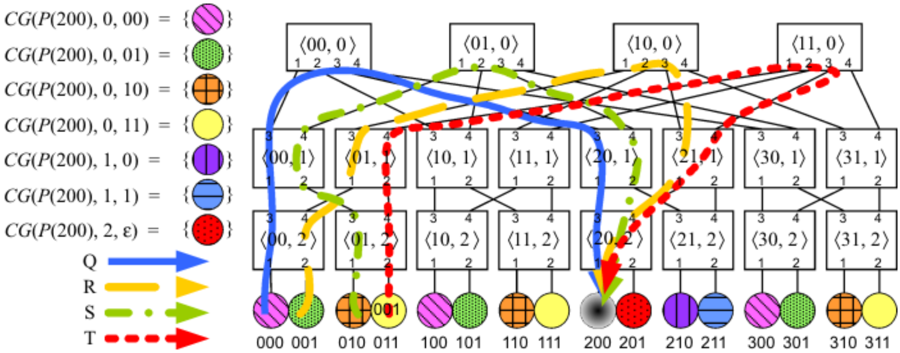


Fig. 5 The cyclic path selection scheme

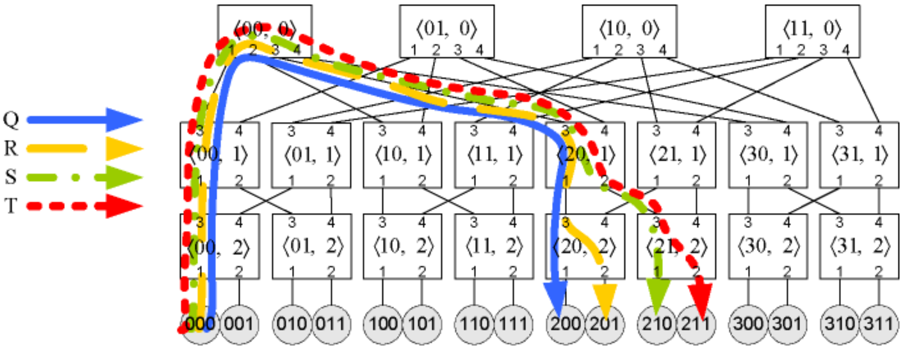


Fig. 6 An example of one-to-many multicast

$P(001)$, $P(010)$, and $P(011)$ want to send messages to the destination processing node $P(200)$. Since the four source processing nodes are in different groups of $CG(P(200), 0)$, they will choose the different LIDs 33, 34, 35, and 36 ($33 + 0$, $33 + 1$, $33 + 2$, and $33 + 3$) of the destination processing node $P(200)$ and send messages through paths Q, R, S, and T, respectively.

According to the path selection scheme, the duplication of packets can be avoided in the ascending phase when a processing node sends packets to different destination processing nodes. An example is shown in Fig. 6. In Fig. 6, the source processing node $P(000)$ sends messages to $P(200)$, $P(201)$, $P(210)$, and $P(211)$ through routes Q, R, S, and T, respectively. From Fig. 6, we can see that all routes take the same path in the ascending phase. Therefore, the duplication of packets will not occur in the ascending phase.

3.3 The forwarding table assignment scheme

The next task is to setup the forwarding table in each InfiniBand switch such that a message sent from one processing node to others will follow the paths we set in the path selection scheme. The forwarding table assignment consists of two phases: the one-to-one forwarding table assignment and the multicast forwarding table assignment based on union operation.

3.3.1 The one-to-one forwarding table assignment

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 $P(p = p_0p_1 \cdots p_{n-1})$ that owns the *lid* can be reached downward from $SW\langle w, l\rangle$, then k can be determined by the following equation

$$k = p_l + 1. \tag{1}$$

For $SW\langle w = w_0w_1\cdots w_{n-2}, l\rangle$, processing node $P(p = p_0p_1\cdots p_{n-1})$ that owns the *lid* can be reached downward from $SW\langle w, l\rangle$ if $PID(P(p)) = \lfloor\frac{(lid-1)}{(\frac{m}{2})^{n-1}}\rfloor$ and $w_0w_1\cdots w_{l-1} = p_0p_1\cdots p_{l-1}$. The conversion between $PID(P(p))$ and $P(p = p_0p_1\cdots p_{n-1})$ 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

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

To verify the correctness of Eqs. (1) and (2), let us take Fig. 6 as an example. In Fig. 6, assume that processing node $P(000)$ wants to send messages to processing node $P(200)$, $P(201)$, $P(210)$, and $P(211)$. According to the path selection scheme, packets sent from $P(000)$ to $P(200)$, $P(201)$, $P(210)$, and $P(211)$ will go through paths Q, R, S, and T, respectively. When a packet is sent from $P(000)$ to $P(200)$ through path Q, the DLID of the packet is 33 and ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$, $SW\langle 00, 0\rangle_3$, $SW\langle 20, 1\rangle_3$, $SW\langle 20, 1\rangle_1$, $SW\langle 20, 2\rangle_3$, and $SW\langle 20, 2\rangle_1$ will be traversed in sequence. When the packet arrives in switch $SW\langle 00, 2\rangle$, $lid = 33$ matches case 2 and the output port of the packet is $k = 3$. When the packet arrives in switch $SW\langle 00, 1\rangle$, $lid = 33$ matches case 2 and the output port of the packet is $k = 3$. When the packet arrives in switch $SW\langle 00, 0\rangle$, $lid = 33$ matches case 1 and the output port of the packet is $k = 3$. When the packet arrives in switch $SW\langle 20, 1\rangle$, $lid = 33$ matches case 1 and the output port of the packet is $k = 1$. When the packet arrives in switch $SW\langle 20, 2\rangle$, $lid = 33$ matches case 1 and the output port of the packet is $k = 1$. From the above analysis, we can see that Eqs. (1) and (2) can correctly setup path Q for the packet sent from $P(000)$ to $P(200)$. For paths R, S, and T, we can obtain similar results.

3.3.2 The multicast forwarding table assignment based on union operations

After the one-to-one forwarding table assignment is performed, we can setup the multicast forwarding table for a given source processing node and a multicast group based on union operations. Let $P(p = p_0p_1\cdots p_{n-1})$ be a source processing node and $lid = \{lid_1, lid_2, \ldots, lid_t \mid t \le 2 \times (m/2)^n\}$ be the DLID of a multicast group, where $\{lid_1, lid_2, \ldots, lid_t | t \le 2 \times (m/2)^n\}$ is the set of LIDs of destination processing nodes in a multicast group. For each switch $SW\langle w, l\rangle$, based on Eqs. (1) and (2), we can determine the output port of a packet whose DLID is $lid_1, lid_2, \ldots,$ and lid_t as $SW\langle w, l\rangle_{k_1}$, $SW\langle w, l\rangle_{k_2}, \ldots,$ and $SW\langle w, l\rangle_{k_t}$, respectively. It means that when a packet whose DLID is $lid_1, lid_2, \ldots,$ and lid_t arrives in switch $SW\langle w, l\rangle$, it will be forwarded to port $SW\langle w, l\rangle_{k_1}$, $SW\langle w, l\rangle_{k_2}, \ldots,$ and $SW\langle w, l\rangle_{k_t}$, respectively. Since an InfiniBand switch can duplicate a packet to different output ports and the path selection schemes given in Sect. 3.2 will prevent the packet from being duplicated in the ascending phase, the output ports of a multicast packet $\{lid_1, lid_2, \ldots, lid_t | t \le 2 \times (m/2)^n\}$ can be set as the union of $SW\langle w, l\rangle_{k_1}$, $SW\langle w, l\rangle_{k_2}, \ldots,$ and $SW\langle w, l\rangle_{k_t}$ in switch $SW\langle w, l\rangle$.

An example is shown in Fig. 7. In Fig. 7, assume that processing node $P(000)$ wants to send multicast packets to processing nodes $P(200)$, $P(201)$, $P(210)$ and

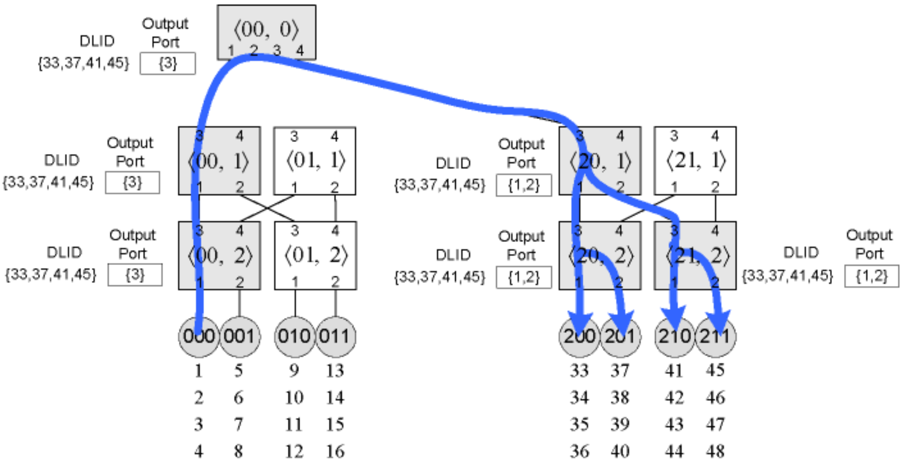


Fig. 7 An example of multicast forwarding table setup

$P(211)$. The *lid* set of the multicast group is $\{33, 37, 41, 45\}$. From Fig. 7, we can see that when a packet is sent from $P(000)$ to $P(200)$, switch ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$, $SW\langle 00, 0\rangle_3$, $SW\langle 20, 1\rangle_3$, $SW\langle 20, 1\rangle_1$, $SW\langle 20, 2\rangle_3$, and $SW\langle 20, 2\rangle_1$ will be traversed. When a packet is sent from $P(000)$ to $P(201)$, switch ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$, $SW\langle 00, 0\rangle_3$, $SW\langle 20, 1\rangle_3$, $SW\langle 20, 1\rangle_1$, $SW\langle 20, 2\rangle_3$, and $SW\langle 20, 2\rangle_2$ will be traversed. When a packet is sent from $P(000)$ to $P(210)$, switch ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$, $SW\langle 00, 0\rangle_3$, $SW\langle 20, 1\rangle_3$, $SW\langle 20, 1\rangle_2$, $SW\langle 21, 2\rangle_3$, and $SW\langle 21, 2\rangle_1$ will be traversed. When a packet is sent from $P(000)$ to $P(211)$, switch ports $SW\langle 00, 2\rangle_1$, $SW\langle 00, 2\rangle_3$, $SW\langle 00, 1\rangle_1$, $SW\langle 00, 1\rangle_3$, $SW\langle 00, 0\rangle_1$, $SW\langle 00, 0\rangle_3$, $SW\langle 20, 1\rangle_3$, $SW\langle 20, 1\rangle_2$, $SW\langle 21, 2\rangle_3$, and $SW\langle 21, 2\rangle_2$ will be traversed. According the above union operations, for a multicast packet whose DLID $= \{33, 37, 41, 45\}$, we can determine that its output ports in switch $SW\langle 00, 2\rangle = \{3\}$, $SW\langle 00, 1\rangle = \{3\}$, $SW\langle 00, 0\rangle = \{3\}$, $SW\langle 20, 1\rangle = \{1, 2\}$, $SW\langle 20, 2\rangle = \{1, 2\}$, and $SW\langle 21, 2\rangle = \{1, 2\}$, respectively. The multicast operation can be performed correctly.

4 Performance evaluation

To evaluate the performance of the proposed multicast scheme, we design an m-port n-tree InfiniBand network simulator by using Java. Three schemes, the proposed cyclic multicast scheme, the OpenSM multicast scheme, and the unicast scheme were simulated for performance evaluation.

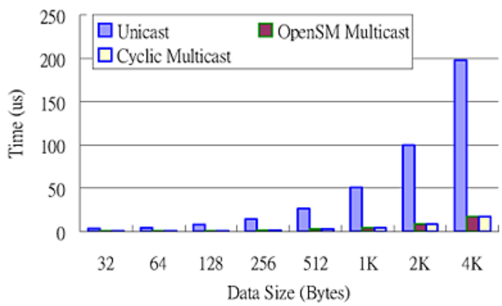
In our simulation, an 8-port 3-tree InfiniBand network is simulated. The network contains 80 switches and 128 processing nodes. The packet size ranges from 32 bytes to 128 Kbytes. The size of source processing nodes is set to 1 processing node, 40% of all processing nodes, 70% of all processing nodes, and all processing nodes. The size of multicast group is set to 10%, 40%, 70%, and 100% of all processing nodes. We assume that the flying time of a packet between devices (endnode-to-switch and switch-to-switch) is 20 ns. The routing time of a packet from one input port to one output port of the crossbar in a switch is 100 ns, including forwarding table lookup, packet replication, arbitration, and message startup time. The byte injection rate is 4 ns assume that a 1X link configuration (2.5 Gbps) is used. Flow control is also taken into account. The packet must wait in the input port buffer until the output port buffer is available.

The simulation results are shown in Fig. 8 to Fig. 16. We have the following cases.

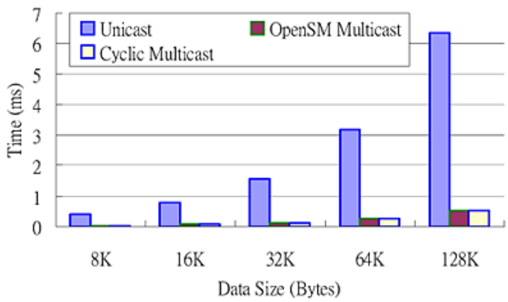
Case 1 (one-to-many multicast): Fig. 8 to Fig. 10 show the results of one-to-many multicast. Since there is only one source processing node, the traffic congestion of two packets using the same buffer is never occurred. From the simulation results, we can see that the multicast schemes outperform the unicast scheme. The time of our cyclic multicast scheme is the same as that of OpenSM multicast scheme given the same destination group size.

Case 2 (many-to-many multicast): Fig. 11 to Fig. 13 show the results of many-to-many multicast. Since there are more than one source processing nodes send messages to the destination processing nodes, the traffic congestion did occur. From the simulation results, we can see that the multicast schemes outperform the unicast scheme. Moreover, our cyclic multicast scheme outperforms the OpenSM multicast scheme when the destination group size is small (10% and 40%). This is because the OpenSM multicast scheme only builds one multicast tree, while our scheme builds more multicast trees to take the advantages of available bandwidth of fat-tree topology. When the destination group size is large (100%), the performance of our scheme

Fig. 8 One-to-10% multicast

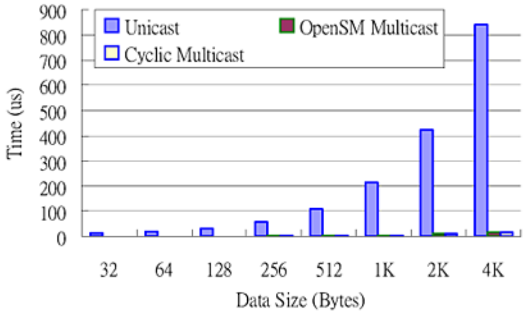


(a) Small size

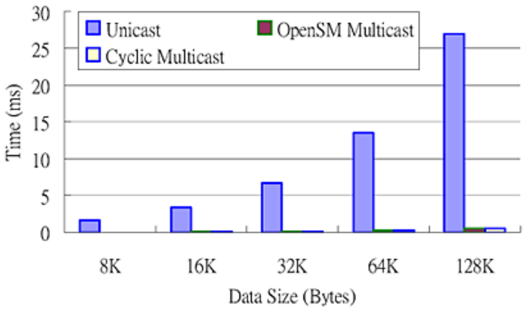


(b) Large size

Fig. 9 One-to-40% multicast

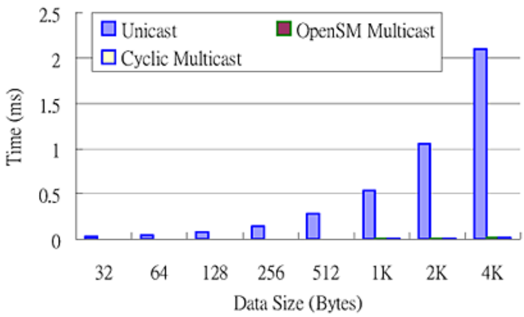


(a) Small size

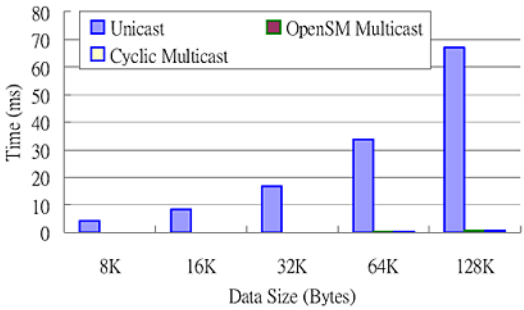


(b) Large size

Fig. 10 One-to-all multicast

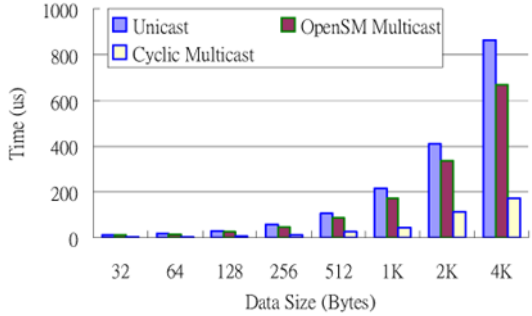


(a) Small size

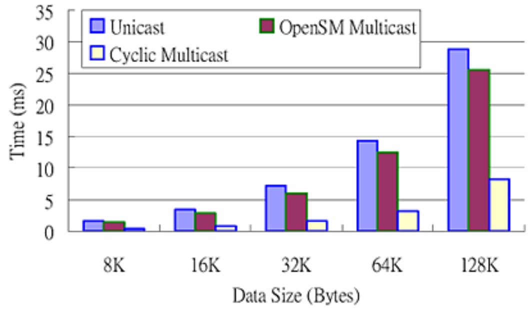


(b) Large size

Fig. 11 40%-to-10% multicast

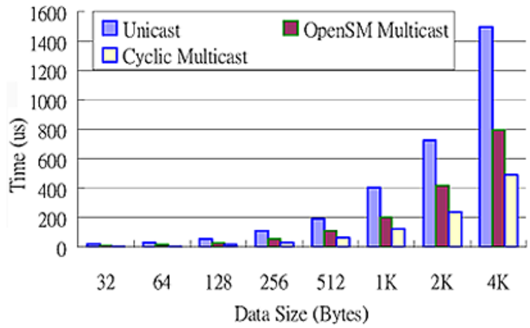


(a) Small size

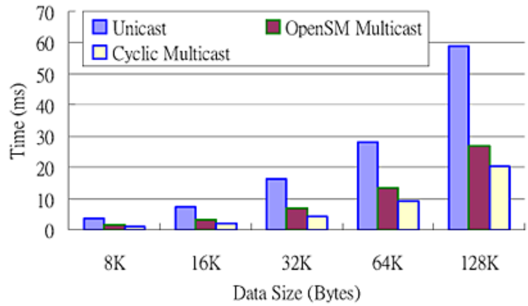


(b) Large size

Fig. 12 40%-to-40% multicast

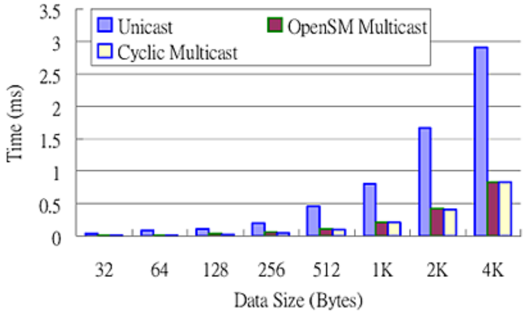


(a) Small size

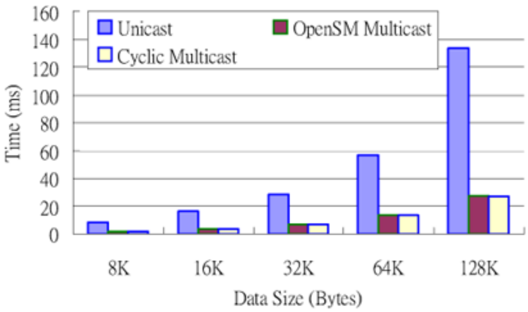


(b) Large size

Fig. 13 40%-to-all multicast



(a) Small size



(b) Large size

is a little better than that of the OpenSM multicast scheme since the serious traffic congestion in the descending phase.

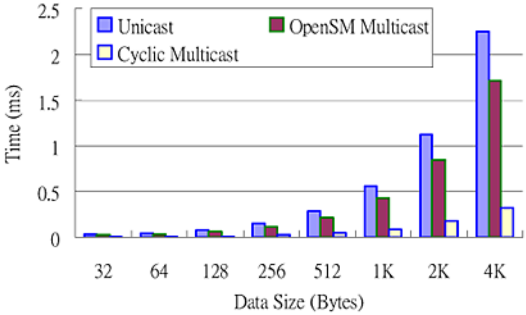
Case 3 (all-to-many multicast): Fig. 14 to Fig. 16 show the results of all-to-many multicast. From Fig. 14 to Fig. 16, we observe that all the simulation results of all-to-many multicast are similar to those of many-to-many multicast. Obviously, the cases of all-to-many multicast spend more time because of more packets need to be transmit and more traffic congestion occurred.

Figure 17 shows the speedups of the proposed multicast scheme over the OpenSM multicast scheme under different multicast group sizes. From Fig. 17, we observe that when the destination group size (multicast group size) is small, we can expect a higher speedup from our method. The reason is that our method can take the advantages of available bandwidth of fat-tree topology. As the destination group size close to 100%, the speedup will close to 1, that is, the performance of our method is a little better than that of the OpenSM method. The reason is that when the multicast group size is getting larger, the multicast packets need to be duplicated and forwarded to most ports in the switches. Serious traffic congestions are raised in the descending phase and dominated the performance.

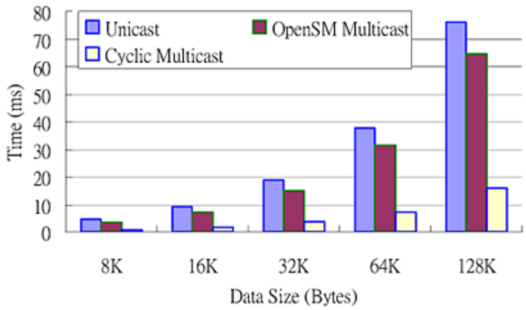
5 Conclusions

In this paper, we propose a hardware supported multicast scheme for the fat-tree-based InfiniBand networks. We describe how to implement the schemes in detail.

Fig. 14 All-to-10% multicast

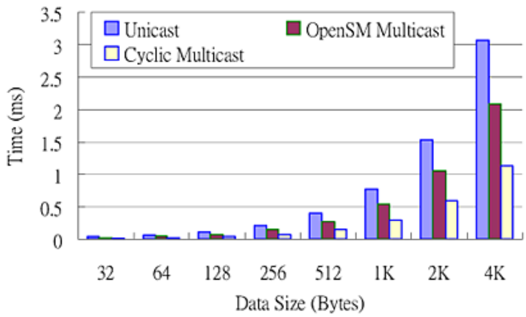


(a) Small size

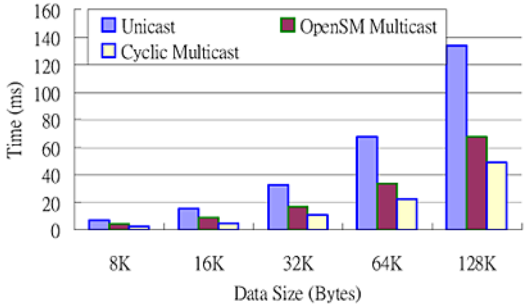


(b) Large size

Fig. 15 All-to-40% multicast

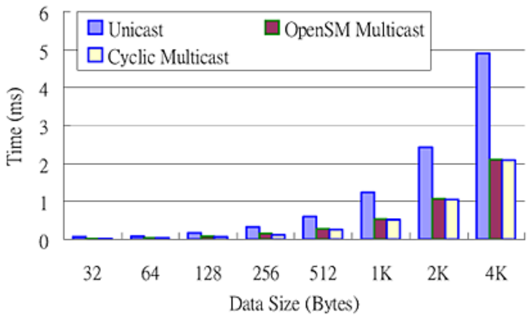


(a) Small size

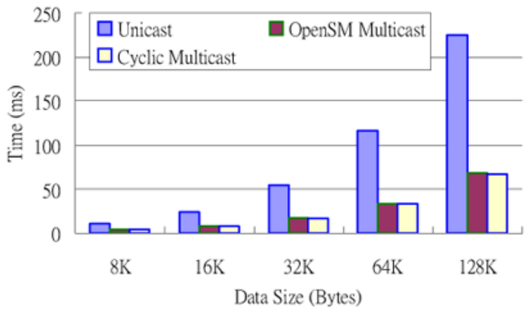


(b) Large size

Fig. 16 All-to-all multicast

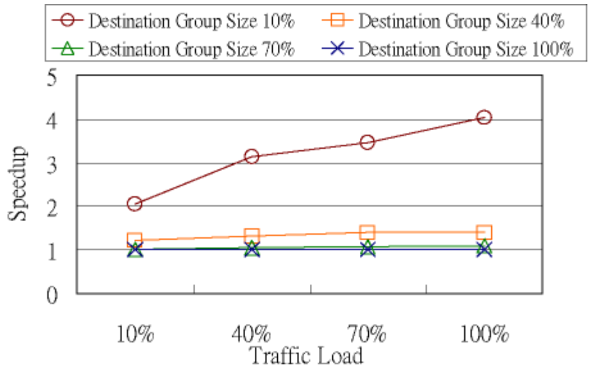


(a) Small size



(b) Large size

Fig. 17 The speedup of cyclic multicast scheme comparing to the OpenSM multicast scheme



We also write a simulator to evaluate the proposed scheme. The simulation results show that the proposed cyclic multicast scheme can speed up the execution of multicast operations. From the simulations results, we have the following remarks:

Remark 1 We observe that the proposed multicast scheme outperforms the unicast scheme for all simulated cases. This result indicates that the hardware supported multicast of the IBA can help to speedup the execution of multicast operations.

Remark 2 Comparing to the OpenSM multicast scheme, for one-to-many case, the performance of the cyclic multicast scheme is the same as that of the OpenSM multicast method. For many-to-many and all-to-many cases, the cyclic multicast scheme outperforms the OpenSM multicast method. For many-to-all case, the performance of the cyclic multicast scheme is a little better than that of the OpenSM multicast method.

References

1. Chiang C-M, Ni LM (1995) Deadlock-free multi-head wormhole routing, In: Proceedings of the first high performance computing-Asia, 1995
2. Dai D, Panda DK (1993) Reducing cache invalidation overheads in wormhole routed DSMs using multidestination message passing. In: Proceedings of the 5th annual ACM symposium on parallel algorithms and architectures, May 1993, pp 2–13
3. DeMara RF, Moldovan DI (1991) Performance indices for parallel marker-propagation. In: Proceedings of the 1991 international conference on parallel processing, St. Charles, Illinois, August 12–17, 1991, pp 658–659
4. Duato J, Yalamanchili S, Ni L (1997) Interconnection networks—an engineering approach. IEEE CS Press
5. Hwang K (1993) Advanced computer architecture—parallelism, scalability, programmability. McGraw-Hill
6. InfiniBand™ trade association (October 2004) InfiniBand™ architecture specification, vol 1. Release 1.2
7. Kumar V, Singh V (1991) Scalability of parallel algorithms for the all-pairs shortest path problem. Tech. Rep. ACT-OODS-058-90, Rev. 1, MCC
8. Kumar S, Kale LV (2004) Scaling all-to-all multicast on fat-tree networks. In: International conference on parallel and distributed systems, July 2004, pp 205–214
9. Leighton FT (1992) Introduction to parallel algorithms and architectures: arrays, trees, hypercubes Morgan Kaufmann Publishers, San Mateo
10. Leiserson CE (1985) Fat-Trees: universal networks for hardware-efficient supercomputing. IEEE Trans Comput 3410:892–901
11. Li K, Schaefer R (1989) A hypercube shared virtual memory. In: Proceedings of the 1989 international conference on parallel processing, vol I, August 1989, pp 125–132
12. Lin XY, Chung YC, Huang TY (2004) A multiple LID routing scheme for fat-tree-based infiniband networks. In: Proceedings of IEEE international parallel and distributed proceeding symposiums, April 2004 (CD-ROM)
13. Lin X, McKinley PK, Ni LM (1991) Performance evaluation of multicast wormhole routing in 2D-mesh multicomputers. In: Proceedings of the 1991 international conference on parallel proceeding, August 1991, vol I, pp 435–442
14. Lin X, Ni LM (1993) Multicast communication in multicomputer networks. IEEE Trans Parallel Distrib Syst 4(10):1104–1117
15. Linux InfiniBand Project. http://infiniband.sourceforge.net
16. Liu J, Mamidala AR, Panda DK (2004) Fast and scalable MPI-level broadcast using infiniband's hardware multicast support. In: Proceedings of IEEE international parallel and distributed proceeding symposiums, April 2004 (CD-ROM)
17. Littlefield RJ (1992) Charaterizing and tuning communications performance for real applications. In: Proceedings of the first intel DELTA applications workshop, February 1992
18. López P, Flich J, Duato J 2001 Deadlock-Free Routing in InfiniBand™ through destination renaming. In: Proceedings of the international conference on parallel processing, ICPP '01, September 2001, pp 427–434
19. McKinley PK, Xu H, Kalns E, Ni LM (1992) ComPaSS: efficient communication services for scalable architectures. In: Proceedings of supercomputing' 92, November 1992, pp. 478–487
20. Petrini F, Vanneschi M (1997) 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
21. Sancho JC, Robles A, Duato J (2001) Effective strategy to compute forwarding tables for InfiniBand networks. In: Proceedings of the international conference on parallel processing, ICPP '01, September 2001, pp 48–57

22. Sancho JC, Robles A, Flich J, López P, Duato J (2002) Effective methodology for deadlock-free minimal routing in infiniband networks. In: Proceedings of the international conference on parallel processing ICPP ’02, August 2002, pp 48-57
23. Sivaram R, Panda DK, Stunkel CB (1996) Efficient broadcast and multicast on multistage interconnection networks using multiport encoding. In: Proceedings of the 8th IEEE symposium on parallel and distributed proceeding, October 1996, pp 36–45
24. Valerio M, Moser L, Melliar-Smith P (1994) 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
25. Xu H, McKinley PK, Ni LM (1992) Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers. J Parallel Distrib Comput 16:172–184

Jiazheng Zhou received a B.S. degree in Computer Science from National Chengchi University in 2002, and the M.S. degree in Computer Science from National Tsing Hua University in 2004. He is currently a Ph.D. student in the Department of Computer Science, National Tsing Hua University. His research interests include cluster computing and interconnection networks. He is a student member of the IEEE computer society.

Xuan-Yi Lin received his B.S. degree in Computer Science and Information Engineering from Da-Yeh University in 2001, and the M.S. degree in Information Engineering and Computer Science from Feng Chia University in 2003. He is currently a Ph.D. student in the Department of Computer Science, National Tsing Hua University, Taiwan. His research interests include cluster systems and grid computing. He is a student member of the IEEE computer society.

Yeh-Ching Chung received a B.S. degree in Information Engineering from Chung Yuan Christian University in 1983, and the M.S. and Ph.D. degrees in Computer and Information Science from Syracuse University in 1988 and 1992, respectively. He joined the Department of Information Engineering at Feng Chia University as an associate professor in 1992 and became a full professor in 1999. From 1998 to 2001, he was the chairman of the department. In 2002, he joined the Department of Computer Science at National Tsing Hua University as a full professor. His research interests include parallel and distributed processing, cluster systems, grid computing, multi-core tool chain design, and multi-core embedded systems. He is a member of the IEEE computer society and ACM.