J Supercomput (2012) 59:901–919
DOI 10.1007/s11227-010-0478-z

A hardware supported multicast scheme based on XY routing for 2-D mesh InfiniBand networks

Jiazheng Zhou · Shen-En Liu · Yeh-Ching Chung

Published online: 16 September 2010

Abstract The multicast operation is a useful operation in parallel applications. It is therefore important to ensure that for a given architecture, the parallel application runs efficiently. With the hardware-supported multicast of the InfiniBand Architecture (IBA), we propose a multicast scheme for $m \times n$ mesh InfiniBand networks based on XY routing. The basic concept of the proposed multicast scheme is to find the union sets of the output ports of switches, which are in the paths between the source node and each destination node in a multicast group. Furthermore, in the proposed scheme, we consider the usage of virtual lanes and evaluate their performance. We implement the proposed multicast scheme on a 2-D mesh InfiniBand network simulator. Several multicast configurations consisting of different message size, different traffic workload and different number of virtual lanes are simulated. The simulation results show that the proposed multicast scheme outperforms its corresponding unicast scheme for all simulation cases. The larger the message size, the larger the number of multicast source nodes, and the larger the size of the multicast group, the better the speedup that can be expected from the proposed multicast scheme. The usage of virtual lanes is also shown to improve the speed of the multicast operations.

Keywords Multicast · Unicast · InfiniBand · Mesh · Union operation · Virtual lane

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

J. Zhou
e-mail: jzzhou@cs.nthu.edu.tw

S.-E. Liu
Department of Flight Simulation, Aerospace Industrial Development Corporation, Taichung, Taiwan, R.O.C.
e-mail: en.arg@msa.hinet.net

1 Introduction

Interconnection networks in cluster systems have a great impact on the performance of communication-bounded applications. Therefore, a high-speed, low-latency, and high-throughput network is essential for a cluster system. The InfiniBand architecture (IBA) [9] 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. Due to the characteristics of the IBA, it is very suitable to use as an interconnection network for a cluster system.

The multicast operation is a very commonly used operation in parallel applications [18]. It can be used to implement many collective communication operations as well. Therefore, its performance will affect applications and collective communication operations greatly. Various researches on multicast operations have been proposed in the literature [2, 5, 6, 14–16]. Since the IBA supports hardware multicast, both applications and collective communication operations can take advantage of this feature to speed up their execution.

In this paper, we propose a hardware-supported multicast scheme for 2-D mesh InfiniBand networks. To perform multicast operations in a 2-D mesh InfiniBand network, our proposed scheme consists of three parts: the node addressing scheme, the path selection scheme, and the forwarding table assignment scheme. First, in the node addressing scheme, each node in the mesh network is assigned a Local Identifier (LID). Second, in the path selection scheme, we develop different methods to associate a switch port with different number of virtual lanes. The XY routing [7] is applied to avoid deadlock. Third, in the forwarding table assignment scheme, a two-phase forwarding table setup is used, one-to-one forwarding table assignment and multicast forwarding table assignment. In one-to-one forwarding table assignment, we set up the forwarding tables according to the node addressing scheme and the path selections scheme. In multicast forwarding table assignment, we set up the forwarding tables based on the union operation.

To evaluate the proposed multicast scheme, we develop a 2-D mesh InfiniBand network simulator. We then implement the proposed multicast scheme and the corresponding unicast scheme. Several multicast cases with different message size and different traffic workload are simulated in a 16×16 mesh InfiniBand network with 1, 2 and 4 virtual lanes. The simulation results show that the proposed multicast scheme outperforms the corresponding unicast scheme for all test cases. The larger the message size, the larger the number of multicast source nodes, and the larger the size of the multicast group, the better the speedup that can be expected from the proposed multicast scheme. The virtual lanes can also be used to help speed operations. The more virtual lanes, the better the performance obtained. For the port association methods, for both unicast scheme and multicast scheme, given the same number of virtual lanes, a better performance can be expected when a switch port is not associated with any dedicated virtual lane.

The rest of this paper is organized as follows. In Sect. 2, we introduce works related to routing on mesh topology. In Sect. 3, we describe our proposed multicast scheme. In Sect. 4, we give the simulation results for the proposed multicast scheme. Finally, Sect. 5 concludes this paper.

2 Related work

In a 2-D mesh network, how to avoid deadlock is an important issue. There are many methods proposed in the literature to offer deadlock-free routing in a 2-D mesh topology. These methods can be divided into deterministic and adaptive routings. Deterministic routing algorithms always use the same path between every pair of nodes. For example, XY routing [7] is for a 2-D mesh and an e-cube is used for hypercubes [23]. Most multicomputer architectures like Cray T3D [12] and Stanford DASH [13] use deterministic routing. Most partially adaptive algorithms are based on the absence of cyclic dependencies between channels to avoid deadlock. Planar-adaptive routing [3] provides adaptivity in only two dimensions at a time and can minimize resources needed. The turn model [8] provides a systematic approach to develop partial adaptive routing algorithms. Fully adaptive routing algorithms use virtual networks to avoid deadlock [10, 17]. A virtual network is a subset of channels used to route packets toward a set of destinations.

Based on the deadlock-free routing algorithms mentioned above, some multicast algorithms have been proposed for 2-D mesh topology. In [15], a tree-based multicast algorithm, known as the double-channel XY multicast routing algorithm, is proposed for 2-D mesh. The method uses double channels to avoid the deadlock. In this method, a 2-D mesh network is divided into four subsets for a given source node. Each subset can be viewed as a virtual network where the XY routing is used. The tree-based multicast with branch pruning [23] allows any deadlock-free routing algorithm of unicast message. It can be applied to any topology. Using this method, the deadlock can be recovered by pruning the branches. This scheme outperforms other mechanisms when the multicast traffic consists of short messages. Tree-based multicast routing can also be applied to multistage interconnection networks (MINs) [2].

Other than the tree-based multicast routing algorithms, path-based multicast routing algorithms can also be applied to a 2-D mesh topology. Based on the Hamiltonian path, the dual-path multicast routing and the multi-path multicast routing are proposed [14]. In [14], the network is separated into two sub-networks, the high-channel sub-network and the low-channel sub-network. The high-channel sub-network contains all the channels from lower label nodes to higher label nodes and the low-channel sub-network contains all the channels from higher label nodes to lower label nodes. In the dual-path routing algorithm, for a given source node, messages are sent to destination nodes with lower labels through the low-channel and higher labels through the high-channel sub-network. The multi-path routing algorithm further partitions the two sub-networks from the dual-path routing into four sub-networks and can utilize the four channels for transmission. In [16], the label-based dual-path (LD) adaptive multicast routing algorithm is proposed. It is similar to the dual-path routing algorithm and allows both minimal and non-minimal paths. In other adaptive routing algorithms [5, 6], the messages can also use the alternative minimal paths. There have been various researches into fault-tolerant algorithms [4, 11, 19, 21, 24, 25]. The reason that fault-tolerant algorithms are required stem from the fact that a regular topology may become an irregular topology if there exist faulty links or nodes. In [20], the authors propose a hardware multicast routing algorithm for 2-D meshes. However, we can see that cyclic waiting can occur when two multicast operations are performed. The authors also mention that in some cases a multicast message has to be temporarily stored and retransmitted in order to eliminate any cyclic dependencies.

Therefore, in the hardware-supported multicast routing algorithm in InfiniBand networks, we have the following considerations. First, since the IBA offers the mechanisms of hardware-supported multicast but not offers any routing algorithm, we need to provide a deadlock-free multicast algorithm in InfiniBand networks. Second, the routing in InfiniBand networks is deterministic, and InfiniBand switch only considers the DLID to forward a packet to the corresponding port. Therefore, the multicast operation may not perform correctly if we do not set up the forwarding table correctly [26]. This is discussed further in the next section. Third, since the IBA offers virtual lanes, we also need to take advantage of virtual lanes to fully utilize this feature of the IBA. Previous works do not address the use of virtual lanes in InfiniBand networks. Our work will take advantage of hardware-supported multicast when performing a multicast operation and will discuss the usage of virtual lanes in InfiniBand networks as well.

3 The proposed multicast scheme

An InfiniBand network can be divided into subnets. In an InfiniBand subnet, a packet source/destination is called an end port. A Local Identifier (LID) [22] is an address assigned to an end port by the subnet manager during the subnet initialization process. An LID is unique within an InfiniBand subnet. Since 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 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.

The IBA supports hardware multicast. In the IBA, each multicast group is assigned a multicast LID and a GID by the subnet manager. The subnet manager will set up the forwarding table of each switch for each multicast group according its LID and GID. To execute a multicast operation in an InfiniBand network, the source 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 packets to the corresponding output ports according to its forwarding table.

In a 2-D mesh topology, a deadlock may occur as shown in Fig. 1. In the figure, each big block presents a node and there is one port (including input buffer and output buffer) on each edge of the block. Source nodes A, B, C, and D want to send messages to their destination nodes A', B', C', and D', respectively. The deadlock may occur if the four nodes A, B, C, and D send messages to their destination nodes simultaneously.

To avoid the deadlock, XY routing can be used. In XY routing, a packet is forwarded first in the X-direction followed by the Y-direction. In this way, the cyclic resource dependency shown in Fig. 1 does not occur, as shown in Fig. 2. The deadlock is avoided since the conditions required to create a deadlock cannot be met. Since the IBA uses the virtual cut-through technique, a packet will stay in a channel

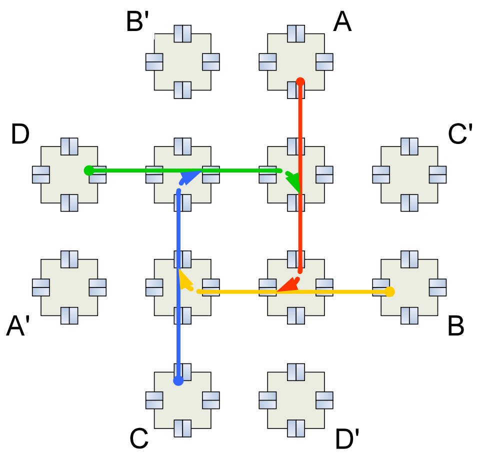


Fig. 1 A deadlock in 2-D mesh

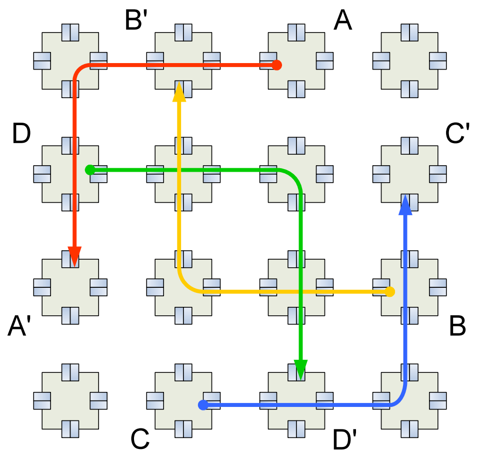


Fig. 2 XY routing can avoid deadlocks in 2-D mesh

(buffer) until the next required channel is free. This deadlock can be avoided when we apply XY routing.

Now we shall discuss the reason that our multicast is based on XY routing. In [8], for unicast, we can see that the performance of negative-first routing is better than that of XY routing. Figure 3(a) and Fig. 4(a) show the examples of XY routing and negative-first routing, respectively. The source node $N(1,1)$ sends messages to nodes $N(2,2)$, $N(3,2)$, and $N(3,3)$ through the paths P, Q, and R, respectively. Other than sending 3 individual unicast packets, the source node $N(1,1)$ can send a multicast packet to these 3 destinations. The corresponding multicast operations of XY routing and negative-first routing are shown in Fig. 3(b) and Fig. 4(b), respectively. The multicast operation can perform correctly in Fig. 3(b) but not in Fig. 4(b). In Fig. 4(b), due to the fact that InfiniBand switch only considers the DLID of the packet when it forwards the packet to the output port, the destination node $N(3,2)$ receives two multicast packets, which is incorrect. Therefore, even though the performance of negative-first routing is better than that of XY routing in unicast cases, we cannot

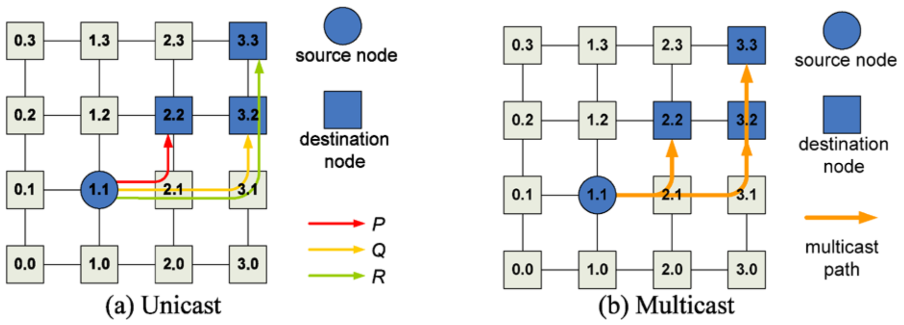


Fig. 3 An example of XY routing

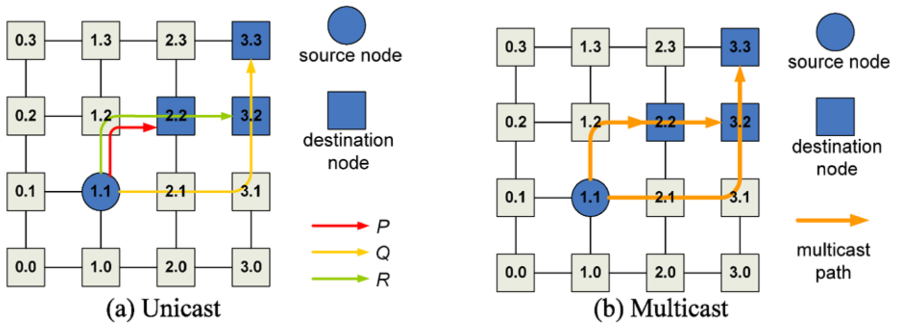


Fig. 4 An example of negative-first routing

choose it as the base routing. As a consequence, our multicast algorithm is based on XY routing, and with hardware-supported multicast in IBA we can get a significant performance improvement in multicast operations.

The proposed multicast can be divided into three schemes: the node addressing scheme, the path selection scheme, and the forwarding table assignment scheme.

3.1 The node addressing scheme

Given an $m \times n$ mesh InfiniBand network $IbaMesh(m, n)$, nodes in $IbaMesh(m, n)$ are labeled as $N(x, y)$, where $x \in \{0, 1, \ldots, m-1\}$ and $y \in \{0, 1, \ldots, n-1\}$. It has the following characteristics:

1. There are $m \times n$ nodes in $IbaMesh(m, n)$.
2. For each node, there are four directions (or ports)—north (N), east (E), south (S), and west (W)—for the message routing.

An LID is assigned to each node $N(x, y)$ in $IbaMesh(m, n)$, using the formula $lid = x \times m + y + 1$. The node addressing scheme is very simple. Each node can calculate its LID by using its coordinate and the dimensions of the mesh. With a simple calculation, a node can easily know what is the neighbor's LID to which it needs to send a packet.

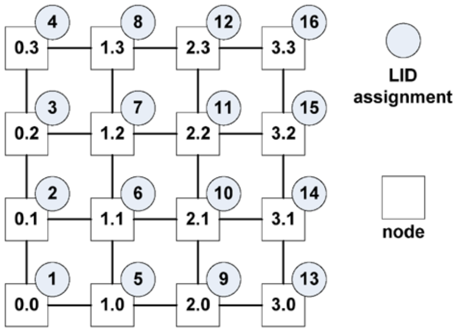


Fig. 5 The node addressing scheme in *IbaMesh*(4, 4)

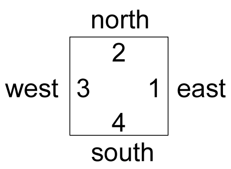


Fig. 6 Direction and port number mapping in a switch

In Fig. 5, there are 16 nodes in a 4 × 4 mesh InfiniBand network *IbaMesh*(4, 4). For node $N(3, 2)$ in *IbaMesh*(4, 4), an *lid* $= 3 \times 4 + 2 + 1 = 15$ is assigned.

3.2 The path selection scheme

Based on the XY routing, the packet will be forwarded first in the X-direction and then in the Y-direction. As shown in Fig. 6, we define the east port as port 1, the north port as port 2, the east port as port 3, and the south port as port 4. Given the source node $N(x_{\text{source}}, y_{\text{source}})$ and the destination node $N(x_{\text{dest}}, y_{\text{dest}})$ in *IbaMesh*(m, n), for each node $N(x_{\text{current}}, y_{\text{current}})$ in the traversed routing path, we can determine the output port of $N(x_{\text{current}}, y_{\text{current}})$. We have the following cases.

Case 1. If $(x_{\text{dest}} - x_{\text{source}} \geq 0)$ and $(y_{\text{dest}} - y_{\text{source}} \geq 0)$, the path of the packet will be $N(x_{\text{source}}, y_{\text{source}})$, $N(x_{\text{source}} + 1, y_{\text{source}})$, $N(x_{\text{source}} + 2, y_{\text{source}}), \ldots, N(x_{\text{dest}}, y_{\text{source}})$, $N(x_{\text{dest}}, y_{\text{source}} + 1)$, $N(x_{\text{dest}}, y_{\text{source}} + 2), \ldots, N(x_{\text{dest}}, y_{\text{dest}})$.

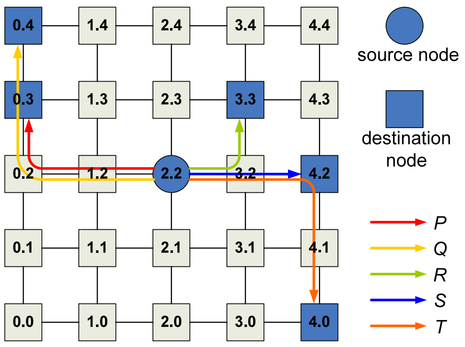
Case 2. If $(x_{\text{dest}} - x_{\text{source}} \geq 0)$ and $(y_{\text{dest}} - y_{\text{source}} < 0)$, the path of the packet will be $N(x_{\text{source}}, y_{\text{source}})$, $N(x_{\text{source}} + 1, y_{\text{source}})$, $N(x_{\text{source}} + 2, y_{\text{source}}), \ldots, N(x_{\text{dest}}, y_{\text{source}})$, $N(x_{\text{dest}}, y_{\text{source}} - 1)$, $N(x_{\text{dest}}, y_{\text{source}} - 2), \ldots, N(x_{\text{dest}}, y_{\text{dest}})$.

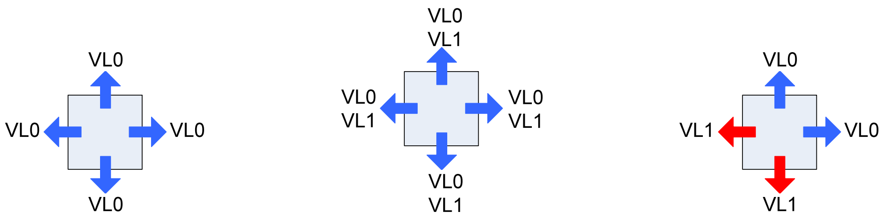
Case 3. If $(x_{\text{dest}} - x_{\text{source}} < 0)$ and $(y_{\text{dest}} - y_{\text{source}} < 0)$, the path of the packet will be $N(x_{\text{source}}, y_{\text{source}})$, $N(x_{\text{source}} - 1, y_{\text{source}})$, $N(x_{\text{source}} - 2, y_{\text{source}}), \ldots, N(x_{\text{dest}}, y_{\text{source}})$, $N(x_{\text{dest}}, y_{\text{source}} + 1)$, $N(x_{\text{dest}}, y_{\text{source}} + 2), \ldots, N(x_{\text{dest}}, y_{\text{dest}})$.

Case 4. If $(x_{\text{dest}} - x_{\text{source}} < 0)$ and $(y_{\text{dest}} - y_{\text{source}} \geq 0)$, the path of the packet will be $N(x_{\text{source}}, y_{\text{source}})$, $N(x_{\text{source}} - 1, y_{\text{source}})$, $N(x_{\text{source}} - 2, y_{\text{source}}), \ldots, N(x_{\text{dest}}, y_{\text{source}})$, $N(x_{\text{dest}}, y_{\text{source}} - 1)$, $N(x_{\text{dest}}, y_{\text{source}} - 2), \ldots, N(x_{\text{dest}}, y_{\text{dest}})$.

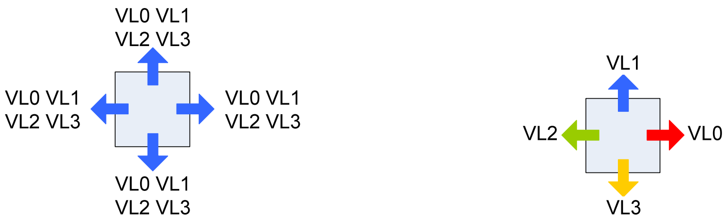
An example is shown in Fig. 7. In the figure, the source node $N(2, 2)$ sends messages to destination nodes $N(0, 3)$, $N(0, 4)$, $N(3, 3)$, $N(4, 2)$, and $N(4, 0)$ through

Fig. 7 An example of XY routing in 5×5 mesh





(a) Using 1 virtual lane (b) Using 2 virtual lanes (c) Using 2 virtual lanes



(d) Using 4 virtual lanes (e) Using 4 virtual lanes

Fig. 8 Different configurations for using different number of virtual lanes

P, Q, R, S, and T, respectively. All the packets will be forwarded first in the X-direction, then in the Y-direction.

The path selection scheme mentioned above can be applied to those networks without virtual lane support. Furthermore, to take advantage of the characteristics of InfiniBand networks, we want to explore the usage of virtual lanes provided by InfiniBand networks. We have different methods to associate a switch port with virtual lane(s) in a switch. Since there are at most four directions in a mesh network, we consider three situations in the system: using 1, 2, or 4 virtual lanes.

With one virtual lane, the packet will always be forwarded to the output port using the only virtual lane available, as shown in Fig. 8(a). With two virtual lanes, the packet can be forwarded to the output port through virtual lane VL0 or VL1. Therefore, we have two configurations for two virtual lanes. One configuration is that when the packet is forwarded to the output port, the packet can use either VL0 or VL1 by the SL-to-VL mapping function, as shown in Fig. 8(b). By using the assigned Service

Level (SL) and SL-to-VL mapping table, the IBA can offer more traffic control and ensure the QoS. The second configuration associates each output port with one dedicated virtual lane by packet relay function in the IBA. It means that when the packet is forwarded to the output port, it will always use the selected virtual lane. In our settings, we associate the north port and east port with VL0, the south port and the east port with VL1, as shown in Fig. 8(c). With four virtual lanes, we also have two configurations, as shown in Fig. 8(d) and (e). In Fig. 8(d), the packet will be forwarded to one of the four virtual lanes according to the assigned SL and SL-to-VL mapping table. In Fig. 8(e), we associate the east port, the north port, the west port, and the south port with VL0, VL1, VL2, and VL3, respectively according to the packet relay function.

3.3 The forwarding table assignment scheme

After the path selection scheme is performed, the next task is to set up the forwarding table assignment. It 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 $m \times n$ mesh InfiniBand network $IbaMesh(m,n)$, a node $N(x,y)$ of $IbaMesh(m,n)$, and a packet whose DLID field is lid, when the packet arrives at node $N(x,y)$, the output port $N(x,y)_k$ of the packet can be determined based on the node assignment scheme, and the path selection scheme. We have the following four cases.

Case 1: If $\lfloor \frac{lid-1}{m} \rfloor > x$, the packet will be forwarded in the X-direction to the east port and $k = 1$.

Case 2: If $\lfloor \frac{lid-1}{m} \rfloor < x$, the packet will be forwarded in the X-direction to the west port and $k = 3$.

Case 3: If $\lfloor \frac{lid-1}{m} \rfloor = x$ and $lid - x \times m - 1 > y$, the packet will be forwarded in the Y-direction to the north port and $k = 2$.

Case 4: If $\lfloor \frac{lid-1}{m} \rfloor = x$ and $lid - x \times m - 1 < y$, the packet will be forwarded in the Y-direction to the south port and $k = 4$.

To verify the correctness of these four cases, let us take Fig. 7 for an example. In the figure, assume that node $N(2,2)$ wants to send a packet to node $N(0,3)$ whose $lid = 4$. According to the path selection scheme, the packet sent from $N(2,2)$ to $N(0,3)$ will go through nodes $N(2,2)$, $N(1,2)$, $N(0,2)$, and $N(0,3)$. When the packet arrives at node $N(2,2)$, its $lid = 4$ matches case 2 and the output port of the packet k is 3. When the packet arrives at node $N(1,2)$, its $lid = 4$ matches case 2 and the output port of the packet k is 3. When the packet arrives at node $N(0,2)$, its $lid = 4$ matches case 3 and the output port of the packet is $k = 2$. From the above analysis, we can correctly set up path P for the packet sent from $N(2,2)$ to $N(0,3)$. For paths Q, R, S, and T, we can obtain similar results.

3.3.2 The multicast forwarding table assignment based on union operation

After the one-to-one forwarding table assignment is performed, we can set up the multicast forwarding table for a given source node and a multicast group based on union operation. Let $N(x, y)$ be a source node and $lid = \{lid_1, lid_2, \ldots, lid_t | t \leq m \times n\}$ be the DLID of a multicast group, where $\{lid_1, lid_2, \ldots, lid_t | t \leq m \times n\}$ is the set of LIDs of destination-processing nodes in a multicast group. For the node $N(x, y)$, based on the one-to-one forwarding table assignment, we can determine the output port of a packet whose DLID is $lid_1, lid_2, \ldots,$ and lid_t as $N(x, y)_{k_1}$, $N(x, y)_{k_2}, \ldots,$ and $N(x, y)_{k_t}$, respectively. It means that when the packet whose DLID is $lid_1, lid_2, \ldots,$ and lid_t arrives at node $N(x, y)$, it will be forwarded to ports $N(x, y)_{k_1}, N(x, y)_{k_2}, \ldots,$ and $N(x, y)_{k_t}$, respectively. Since an InfiniBand switch can duplicate a packet to different output ports, the output ports of a multicast packet $lid = \{lid_1, lid_2, \ldots, lid_t | t \leq 2 \times (m/2)^n\}$ can be set as the union of $N(x, y)_{k_1}$, $N(x, y)_{k_2}, \ldots,$ and $N(x, y)_{k_t}$.

An example to describe union operation is shown in Fig. 9. In the figure, assume that processing node $N(2, 2)$ wants to send multicast packets to processing nodes $N(0, 3)$, $N(0, 4)$, $N(3, 3)$, $N(4, 0)$ and $N(4, 2)$. The *lid* set of the multicast group is $\{4, 5, 19, 21, 23\}$. From Fig. 9 we can see that when a packet is sent from $N(2, 2)$ to $N(0, 3)$, node ports $N(2, 2)_3$, $N(1, 2)_1$, $N(1, 2)_3$, $N(0, 2)_1$, $N(0, 2)_2$, and $N(0, 3)_4$ will be traversed. When a packet is sent from $N(2, 2)$ to $N(0, 4)$, node ports $N(2, 2)_3$, $N(1, 2)_1$, $N(1, 2)_3$, $N(0, 2)_1$, $N(0, 2)_2$, $N(0, 3)_4$, $N(0, 3)_2$, and $N(0, 4)_4$ will be traversed. When a packet is sent from $N(2, 2)$ to $N(3, 3)$, node ports $N(2, 2)_1$, $N(3, 2)_3$, $N(3, 2)_2$ and $N(3, 3)_4$ will be traversed. When a packet is sent from $N(2, 2)$ to $N(4, 0)$, node ports $N(2, 2)_1$, $N(3, 2)_3$, $N(3, 2)_1$, $N(4, 2)_3$, $N(4, 2)_4$, $N(4, 1)_2$, $N(4, 1)_4$, and $N(4, 0)_2$ will be traversed. When a packet is sent from $N(2, 2)$ to $N(4, 2)$, node ports $N(2, 2)_1$, $N(3, 2)_3$, $N(3, 2)_1$, and $N(4, 2)_3$ will be traversed. According to the union operation, for a multicast packet whose DLID =

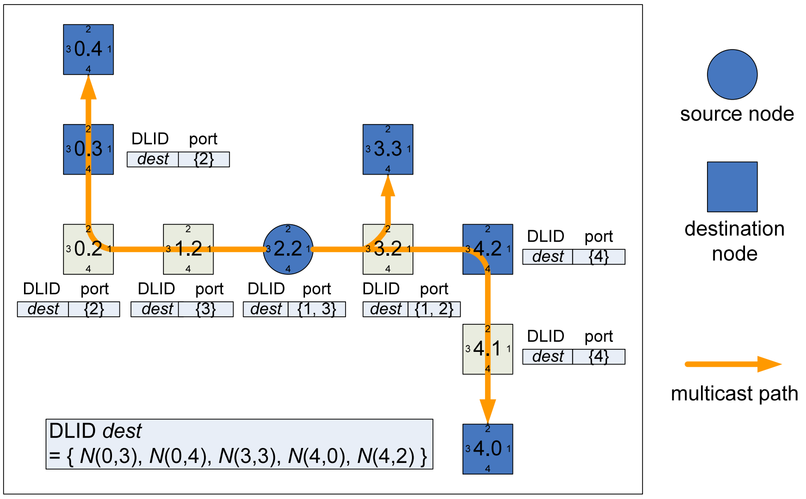


Fig. 9 Multicast forwarding table setup

$\{4, 5, 19, 21, 23\}$, we can determine that its output ports in nodes $N(2,2) = \{1,3\}$, $N(1,2) = \{3\}$, $N(0,2) = \{2\}$, $N(0,3) = \{2\}$, $N(3,2) = \{1,2\}$, $N(4,2) = \{4\}$, and $N(4,1) = \{4\}$, respectively.

4 Performance evaluation

To evaluate the performance of the proposed multicast scheme, we design an $m \times n$ mesh InfiniBand network simulator. The multicast scheme and its corresponding unicast scheme are simulated for performance evaluation. Several cases with different message sizes and traffic workload sizes are simulated in a 16×16 (256 nodes in total) mesh InfiniBand network with 1, 2 and 4 virtual lanes. The packet size ranges from 32 bytes to 8 Kbytes. The link is 1X SDR and the signaling rate is 2.5 Gbps. Since IBA uses 8B/10B encoding—every 10 bits sent carry 8 bits data, the data rate throughput is 2 Gbps. Therefore, the time to transfer one byte (8 bit) from one node to another is 4 ns. According to the size of the source nodes, we have one-source multicast, multi-source multicast, and all-source multicast. That is, there is one node, multiple nodes, and all the nodes to perform multicast in the network. The simulation results are shown in Figs. 10 to 27. We have the following three cases.

Case 1 (One-source multicast): Figs. 10 to 15 show the results of one-source multicast cases. There is only one source node to perform multicast operation. The destination group size is 40% and 100% of all nodes for Figs. 10 to 12 and Figs. 13 to 15, respectively. Figures 10, 11 and 12 show the results with 1, 2, and 4 virtual lanes, respectively. Figures 13, 14 and 15 show the results with 1, 2, and 4 virtual lanes, respectively. Since there is only one source node, the traffic congestion of two packets using the same buffer never occurs. From the simulation results we can see that all the multicast scheme cases outperform their corresponding unicast scheme cases. There are some indications in the figures. For two virtual lanes, "NESW: VL0, VL1" means the north, east, south, and west ports in the switch can use either virtual lane 0 or virtual lane 1. "NE: VL0; SW: VL1" means the north and east ports in the switch can use virtual lane 0 only; the south and west ports in switch can use virtual lane 1 only. For four virtual lanes, "NESW: VL0, VL1, VL2, VL3" means the north, east, south, and west ports in the switch can use either virtual lanes 0, 1, 2, or 3. "N:VL0; E:VL1;

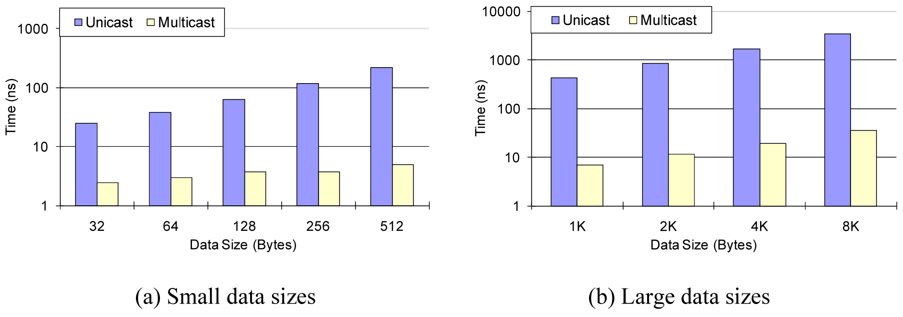
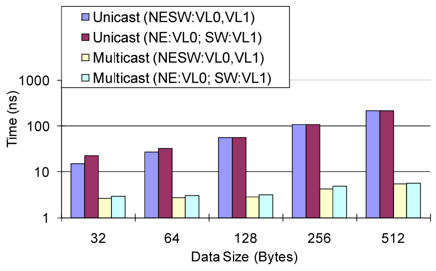
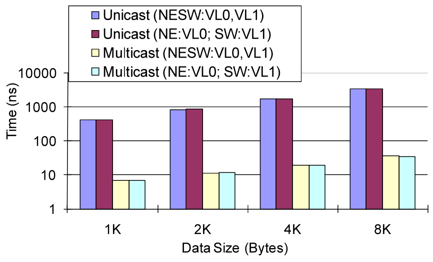


Fig. 10 One-source multicast with 1 virtual lane (destination group size: 40%)

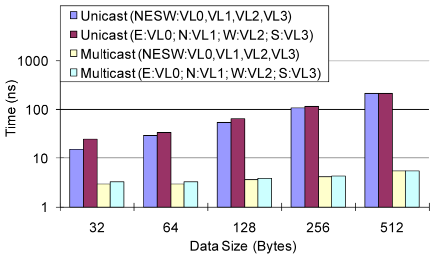


(a) Small data sizes

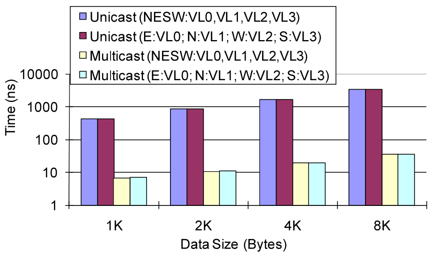


(b) Large data sizes

Fig. 11 One-source multicast with 2 virtual lanes (destination group size: 40%)



(a) Small data sizes

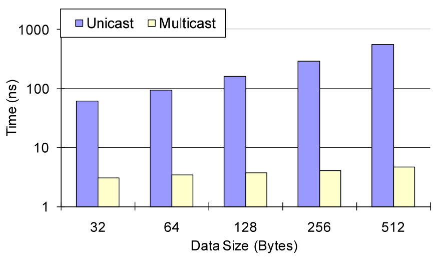


(b) Large data sizes

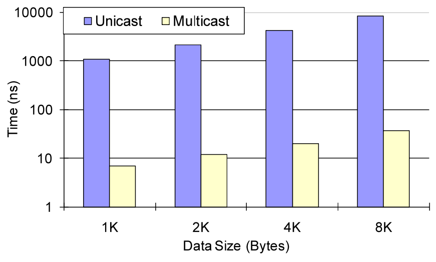
Fig. 12 One-source multicast with 4 virtual lanes (destination group size: 40%)

S:VL2; W:VL3" means the north port in the switch can use virtual lane 0 only; the east port in the switch can use virtual lane 1 only; the south port in the switch can use virtual lane 2 only; the west port in the switch can use virtual lane 3 only. With more virtual lanes, less time is spent to perform multicast operations, especially for unicast cases. For both the unicast and the multicast schemes, given the same number of virtual lanes, we can get better performance if we do not associate the switch port with dedicated virtual lane(s). The observation can be clearly found in Figs. 11(a), 12(a), 14(a) and 15(a). This is because we can utilize the virtual lanes efficiently. However, this observation is not clear when the data size is large.

Case 2 (Multi-source multicast): Figs. 16, 17, 18, 19, 20, 21 show the results of multi-source multicast cases. The size of source nodes is 40% of all nodes. The destination group size is 40% and 100% of all nodes for Figs. 16 to 18 and Figs. 19 to 21, respectively. Since there are multiple source nodes sending messages to the destination nodes, the traffic congestion does occur. From the simulation results, we can see that the multicast scheme cases outperform the corresponding unicast scheme cases. By adding virtual lanes, we can speed up the performance, as it reduces traffic congestion. For both the unicast scheme and the multicast scheme, with the same number of virtual lanes, we can get a better performance if we do not associate the switch port with dedicated virtual lane(s). This observation can be clearly seen in Figs. 17(a),

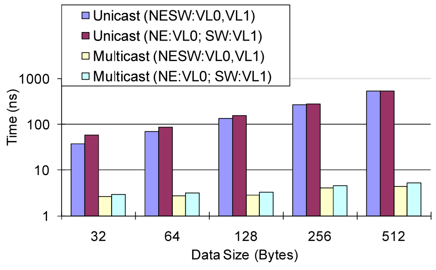


(a) Small data sizes

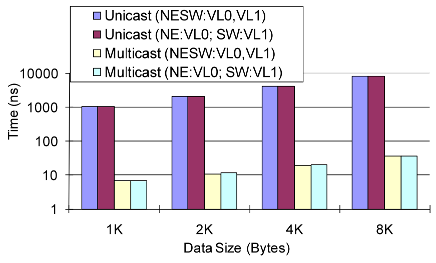


(b) Large data sizes

Fig. 13 One-source multicast with 1 virtual lane (destination group size: 100%)

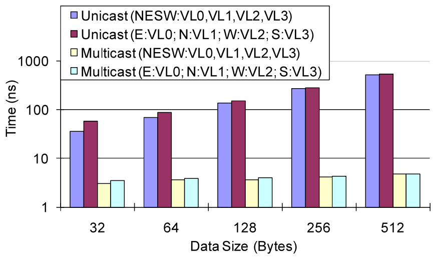


(a) Small data sizes

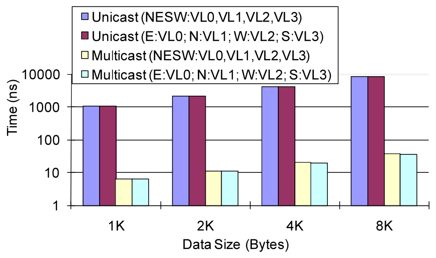


(b) Large data sizes

Fig. 14 One-source multicast with 2 virtual lanes (destination group size: 100%)



(a) Small data sizes

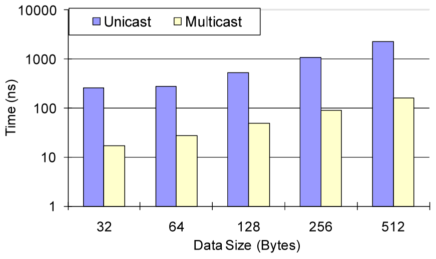


(b) Large data sizes

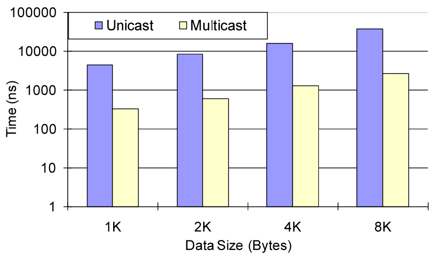
Fig. 15 One-source multicast with 4 virtual lanes (destination group size: 100%)

18(a), 20(a) and 21(a). However, this observation is not obvious when the data size is large.

Case 3 (All-source multicast): Figs. 22, 23, 24, 25, 26, 27 show the results of all-source multicast cases. The destination group size is 100% of all nodes. The destination group size is 40% and 100% of all nodes for Figs. 22 to 24 and Figs. 25 to 27, respectively. Since all the processing nodes in the system perform multicast opera-

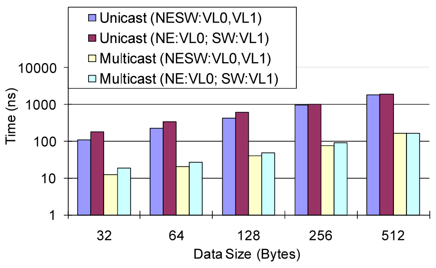


(a) Small data sizes

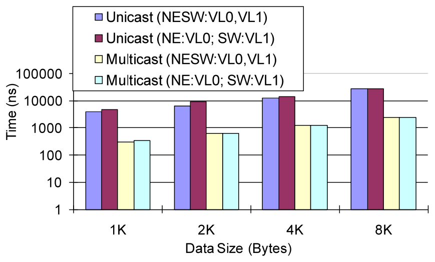


(b) Large data sizes

Fig. 16 Multi-source multicast with 1 virtual lane (source size: 40%, destination group size: 40%)

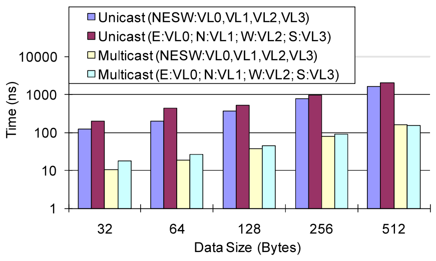


(a) Small data sizes

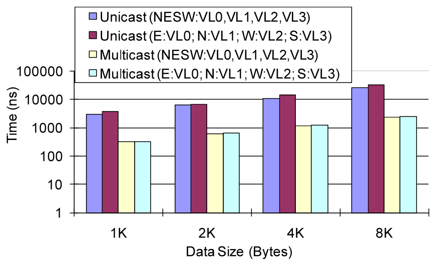


(b) Large data sizes

Fig. 17 Multi-source multicast with 2 virtual lanes (source size: 40%, destination group size: 40%)



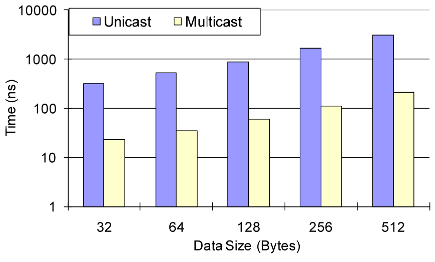
(a) Small data sizes



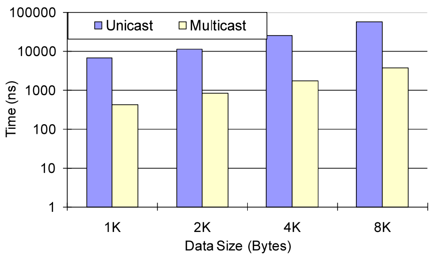
(b) Large data sizes

Fig. 18 Multi-source multicast with 4 virtual lanes (source size: 40%, destination group size: 40%)

tions, the more traffic congestion we can expect. We observe that all the simulation results of all-source multicast are similar to those of multi-source multicast. Obviously, the all-source multicast takes longer to send packets because there are more packets to be transmitted and therefore more traffic congestion occurs.

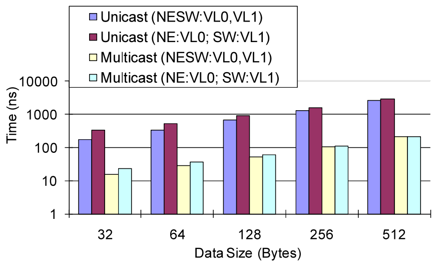


(a) Small data sizes

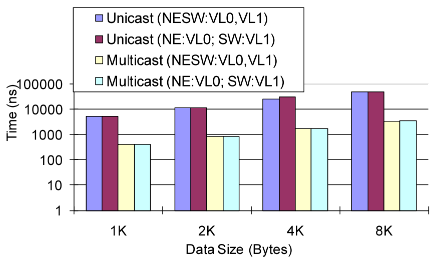


(b) Large data sizes

Fig. 19 Multi-source multicast with 1 virtual lane (source size: 40%, destination group size: 100%)

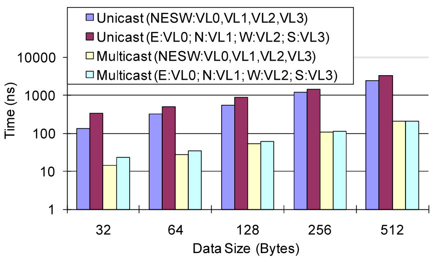


(a) Small data sizes

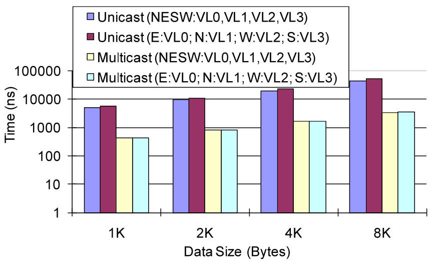


(b) Large data sizes

Fig. 20 Multi-source multicast with 2 virtual lanes (source size: 40%, destination group size: 100%)



(a) Small data sizes

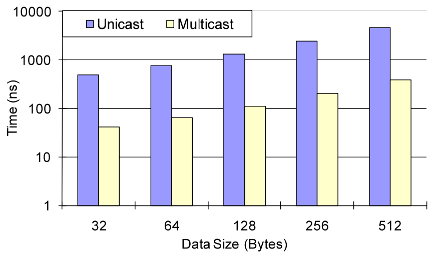


(b) Large data sizes

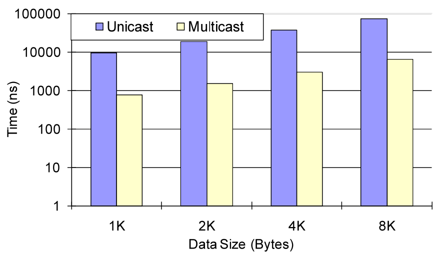
Fig. 21 Multi-source multicast with 4 virtual lanes (source size: 40%, destination group size: 100%)

5 Conclusions and future work

In this paper, we propose a hardware-supported multicast scheme for InfiniBand network in 2-D mesh topology. We describe how to construct the proposed scheme in detail. The simulation results show that the proposed multicast scheme can speed up the execution of multicast operations tremendously. From the simulation results, we have the following remarks.

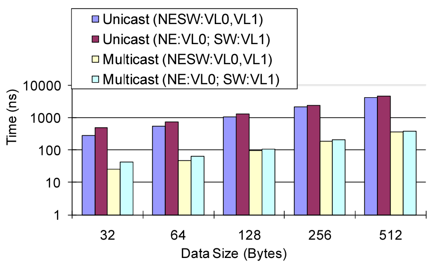


(a) Small data sizes

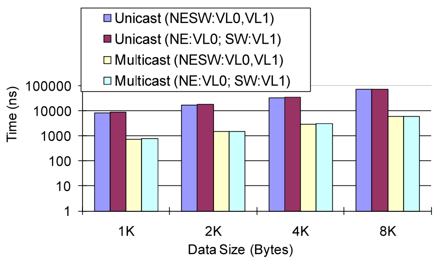


(b) Large data sizes

Fig. 22 All-source multicast with 1 virtual lane (source size: 100%, destination group size: 100%)

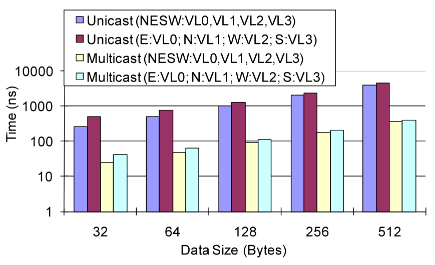


(a) Small data sizes

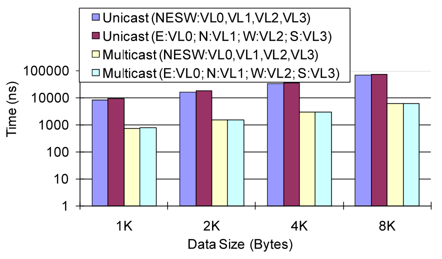


(b) Large data sizes

Fig. 23 All-source multicast with 2 virtual lanes (source size: 100%, destination group size: 100%)



(a) Small data sizes

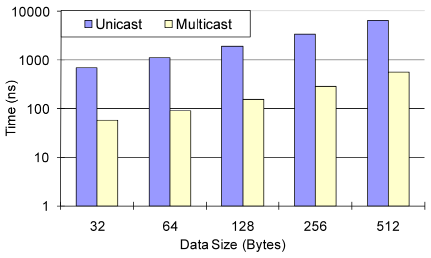


(b) Large data sizes

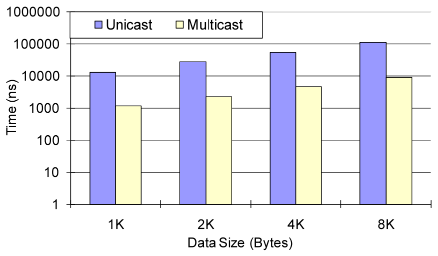
Fig. 24 All-source multicast with 4 virtual lanes (source size: 100%, destination group size: 100%)

Remark 1 The proposed multicast scheme outperforms the unicast scheme for all the simulation cases. The result indicates that the hardware-supported multicast of the IBA can help speed up the execution of multicast operations.

Remark 2 The larger the message size, the larger the number of multicast source nodes, and the larger the size of the multicast group, the better the speedup that can be expected from the proposed multicast scheme.

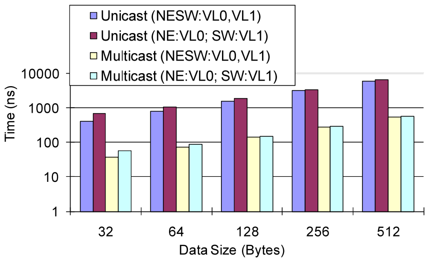


(a) Small data sizes

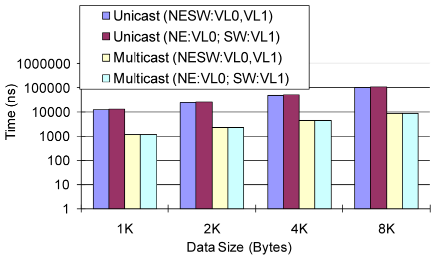


(b) Large data sizes

Fig. 25 All-source multicast with 1 virtual lane (source size: 100%, destination group size: 100%)

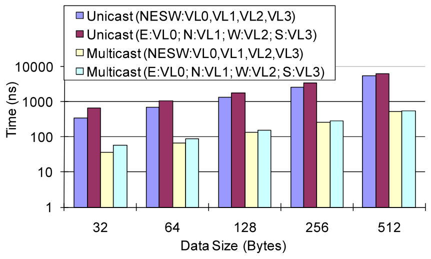


(a) Small data sizes

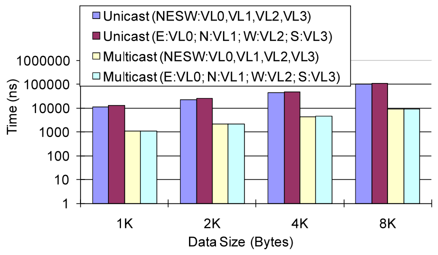


(b) Large data sizes

Fig. 26 All-source multicast with 2 virtual lanes (source size: 100%, destination group size: 100%)



(a) Small data sizes



(b) Large data sizes

Fig. 27 All-source multicast with 4 virtual lanes (source size: 100%, destination group size: 100%)

Remark 3 With more virtual lanes, we can get a better performance, especially for the unicast scheme. For both the unicast scheme and the multicast scheme, with the same number of virtual lanes (2 or 4 in this work), we get a better performance when we do not associate the switch port with dedicated virtual lane(s).

In the future, we will discuss the QoS issue [1]. Since different applications have different traffic requirements, we need to analyze them and use the multicast operations to help speed up communication while still maintaining the QoS requirements.

References

1. Alfaro FJ, Sanchez JL, Duato J (2004) QoS in InfiniBand subnetworks. IEEE Trans Parallel Distrib Syst 15(9):810–823
2. Chiang C-M, Ni LM (1995) Deadlock-free multi-head wormhole routing. In: Proceedings of the first high performance computing—Asia
3. Chien A, Kim JH (1992) Planar-adaptive routing: low-cost adaptive networks for multiprocessors. In: Proceedings of the 19th international symposium on computer architecture, May, pp 268–277
4. Duan X, Zhang D, Sun X (2009) Fault-tolerant routing schemes for wormhole mesh. In: Proceedings of international symposium on parallel and distributed processing with applications, August, pp 298–301
5. Duato J (1993) A new theory of deadlock-free adaptive multicast routing in wormhole networks. In: Proceedings of the 5th IEEE symposium on parallel and distributed processing, December, pp 64–71
6. Duato J (1994) A theory of fault-tolerant routing in wormhole networks. In: Proceedings of the international conference on parallel and distributed systems, December, pp 600–607
7. Duato J, Yalamanchili S, Ni L (1997) Interconnection networks—an engineering approach. IEEE CS Press, Los Alamitos
8. Glass J, Ni LM (1992) The turn model for adaptive routing. In: Proceedings of the 19th international symposium on computer architecture, May, pp 278–287
9. InfiniBand™ (2008) Trade Association, InfiniBand™ architecture specification vol 1, Release 1.2.1, January
10. Jesshope R, Miller PR, Yantchev JT (1989) High performance communications in processor networks. In: Proceedings of the 16th international symposium on computer architecture, May–June, pp 150–157
11. Jiang Z, Wu J, Wang D (2005) A new fault information model for fault-tolerant adaptive and minimal routing in 3-D meshes. In: Proceedings of international conference on parallel processing, June, pp 500–507
12. Kessler RE, Schwarzmeire JL (1993) CRAY T3D: a new dimension for Cray research. In: Proceedings of compcon, pp 176–182
13. Lenoski D et al (1992) The Stanford DASH multiprocessor. IEEE Comput 25(3):63–79
14. Lin X, Ni LM (1991) Deadlock-free multicast wormhole routing in multicomputer networks. In: Proceedings of the 18th international symposium on computer architecture, May, pp 116–125
15. Lin X, McKinley PK, Ni LM (1991) Performance evaluation of multicast wormhole routing in 2-D-mesh multicomputers. In: Proceedings of the international conference on parallel processing, August, vol I, pp 435–442
16. Lin X, Mckinley PK, Esfahanian AH (1993) Adaptive multicast wormhole routing in 2-D mesh multicomputers. In: Proceedings of parallel architectures and languages Europe 93, June, pp 228–241
17. Linder H, Harden JC (1991) An adaptive and fault-tolerant wormhole routing strategy for k-ary n-cubes. IEEE Trans Comput C-40(1):2–22
18. Littlefield RJ (1992) Characterizing and tuning communications performance for real applications. In: Proceedings of the first international DELTA applications workshop, February
19. Mejia A, Flich J, Duato J, Reinemo S-A, Skeie T (2006) Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In: Proceedings of international symposium on parallel and distributed processing, April
20. Mohapatra P, Varavithya V (1996) A hardware multicast routing algorithm for two-dimensional meshes. In: IEEE symposium on parallel and distributed processing, October, pp 198–205
21. Nickmanesh S, Movaghar A, Rookhosh F (2008) Performance modeling of fault-tolerant fully adaptive wormhole switching 2-D meshes in presence of virtual channels. In: Proceedings of international conference on systems and networks communications, October, pp 109–114
22. Nienaber W, Yuan X, Duan Z (2009) LID assignment in InfiniBand networks. IEEE Trans Parallel Distrib Syst 20(4):484–497

23. Sullivan H, Bashhow TR (1977) A large scale, homogeneous, fully, distributed parallel machine. In: Proceedings of the 4th international symposium on computer architecture, March
24. Theiss I, Lysne O (2006) FRoots: a fault-tolerant and topology-flexible routing technique. IEEE Trans Parallel Distrib Syst 17(10):1136–1150
25. Vishnu A, Krishnan M, Panda DK (2009) An efficient hardware-software approach to network fault-tolerance with InfiniBand. In: Proceedings of international conference on cluster computing and workshops, August, pp 1–9
26. Zhou J, Lin X-Y, Chung Y-C (2007) Hardware supported multicast in fat-tree-based InfiniBand networks. J Supercomput 40(3):333–352