Performance Enhancement Techniques for InfiniBand™ Architecture *

Eun Jung Kim* Ki Hwan Yum† Chita R. Das* Mazin Yousif‡ José Duato§

*Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
{ejkim,das}@cse.psu.edu

†Department of Computer Science
University of Texas at San Antonio
San Antnoio, TX 78249
yum@cs.utsa.edu

‡Advanced Component Division
Intel Corporation
Hillsboro, OR 97124
mazin.s.yousif@intel.com

§Department de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
46071- Valencia, Spain
jduato@gap.upv.es

Abstract

InfiniBand™ Architecture (IBA) is envisioned to be the default communication fabric for future system area networks (SANs). However, the released IBA specification outlines only higher level functionalities, leaving it open for exploring various design alternatives. In this paper, we investigate four co-related techniques to provide high and predictable performance in IBA. These are: (i) using the Shortest Path First (SPF) algorithm for deterministic packet routing; (ii) developing a multipath routing mechanism for minimizing congestion; (iii) developing a selective packet dropping scheme to handle deadlock and congestion; and (iv) providing multicasting support for customized applications. These designs are evaluated using an integrated workload on a versatile IBA simulation testbed.

Simulation results indicate that the SPF routing, multipath routing, packet dropping, and multicasting schemes are quite effective in delivering high and assured performance in clusters. One of the major contributions of this research is the IBA simulation testbed, which is an essential tool to evaluate various design tradeoffs.

1 Introduction

InfiniBand Architecture (IBA) has been proposed as a new communication standard to design SANs for scalable, high performance clusters. IBA is expected to revolutionize the future communication paradigm by solving the bandwidth, scalability, reliability, and standardization issues under one unifying design. The IBA Trade Association (IBTA) consisting of more than 220 industry leaders has released the first IBA specification [1] and is currently augmenting it with enhanced features such as Congestion Management, Quality of Service (QoS) provisioning, and Router Management. QoS is becoming an essential part of the IBA framework [2] because of the sophistication of services that will be supported by clusters connected through SANs.

IBA could use either packet switching or virtual cut-through (VCT) switching technology to connect processors and I/O devices. The specification supports any topology to facilitate ease of expansion and to build large networks consisting of smaller subnets. It outlines only the functionalities without any constraints on the actual design. Therefore, it is conceivable to have multiple design alternatives for the same set of high-level requirements, thus making the design space very complex.

An IBA testbed is, therefore, essential to investigate various design options for satisfying the performance and QoS requirements. However, there is no such simulation platform available now, and as we understand, the IBTA is planning to develop such a platform with help from academia.

The main motivation of this paper is to investigate the following design issues for providing improved and predictable performance in IBA. First, it is not clear what is a good routing algorithm for IBA considering the fact that the interconnect could be an irregular topology. Second, the IBA specification supports multipathing to facilitate Automatic Path Migration (APM) between a source and destination pair to provide fault-tolerance. However, the actual path set up in the routing/forwarding table is left open for the designers. Moreover, we believe that the multipathing mechanism can be used not only for fault-tolerance, but for

*This research was supported in part by NSF grants CCR-9900701, CCR-0098149, CCR-0208734 and EIA-0202007, and Spanish CICYT grant TIC2000-1151.

congestion avoidance to improve performance. Therefore, it is essential to understand the design and performance implications of multipathing. Third, packet dropping is allowed under the IBA framework to limit the life time of a packet in a network. Packet dropping can also be used for deadlock avoidance. Thus, instead of using a complex deadlock-free algorithm, we can use a simple routing scheme with packet dropping to provide competitive performance. Finally, multicasting is a desirable feature in IBA to efficiently support various applications in clusters. Multicast design comes with various flavors and needs a comparative evaluation for selecting the most effective design.

We address these design issues by developing a comprehensive simulation testbed for IBA. For the first issue, we examine two deterministic routing algorithms, called Shortest Path First (SPF) and Up/Down [3], that are suitable for irregular networks. Since the SPF scheme is used for Internet routing, for interoperability, it would be nice if we could adopt this protocol for IBA as well. For the second issue on multipathing, we first present a graph theoretic analysis to determine the number of alternate paths between two nodes in an irregular network, and formulate the forwarding table construction to specify the alternate paths. We also demonstrate how the forwarding table can be populated to implement a multipath routing algorithm using two different path selection heuristics; Least Frequently Used (LFU) and Max-Credit [4]. We then extend these algorithms to include a selective packet dropping scheme to handle deadlock and congestion. Finally, for the fourth issue on multicasting, we examine the implementation complexities of synchronous and asynchronous multicasts, and three bandwidth allocation schemes.

We have developed a detailed simulator for a pipelined IBA switch and a Host Channel Adaptor (HCA)/Network Interface Card (NIC) to construct any arbitrary size network. We use a mixed workload consisting of three types of traffic — short control messages, best-effort traffic, and MPEG-2 streams to evaluate the effectiveness of various designs. We conduct an in-depth performance analysis using average packet latency, Deadline Missing Probability (DMP) and average Deadline Missing Time (DMT) as the performance metrics. The first parameter quantifies performance implications for all kinds of traffic, while the other two parameters are QoS indicators of real-time traffic.

Simulation results with 15-node and 30-node irregular networks indicate that the SPF routing can provide better performance than the deadlock-free Up/Down routing and thus, is a good candidate for implementation. The multipath routing, packet dropping, and multicasting schemes are quite effective in delivering high and predictable performance in SANs. Specifically, the multipath routing with packet dropping can lower DMP and DMT for MPEG-2 streams by about 50% compared to a deterministic routing scheme. For the best-effort and control traffic, these congestion avoidance schemes minimize the average packet latency up to 90% at higher load. The synchronous and asynchronous replication schemes are equally competitive in delivering better performance compared to a switch without any hardware support for multicasting. However, the synchronous replication is a better choice for less hardware complexity.

The rest of the paper is organized as follows. In Section 2, the switch architecture and the HCA design are discussed. Section 3 presents the proposed performance enhancement techniques. In Section 4, we discuss the experimental platform. The performance results are presented in Section 5, followed by the concluding remarks in Section 6.

2 System Architecture

In this section, we summarize the IBA framework, followed by our design details for the switch and the HCA architectures.

2.1 InfiniBand Architecture (IBA)

An InfiniBand fabric includes a number of subnets connected through routers. Within a subnet, switches connect processors and I/O nodes. Processing nodes are attached to an IBA network through Host Channel Adaptors (HCAs) and I/O nodes can be attached to a network through Target Channel Adaptors (TCAs).

Virtual Lanes (VLs)/Virtual Channels (VCs) [5] provide a mechanism to implement multiple logical flows over a single physical link. A port must support at least 2 VLs and at most 16 VLs. All ports must use VL_{15} for subnet management traffic. VL arbitration means selection of a VL to push data to an outgoing link in a switch, router, or Channel Adaptor (CA). IBA specifies a two-level scheme for VL arbitration. First, applications are assigned different Service Levels (SLs) — an SL could refer to a different priority level, and a scheduling priority is adopted for different SLs. Next, a Weighted Round Robin (WRR) scheduling is used to schedule packets of the same class.

2.2 Switch Architecture

The switch used in this paper adopts a five-stage pipelined packet-switched model, as shown in Figure 1. At the first stage, the arriving packets are mapped into one of the C VLs using the SL information in the packet header. The header of the queued packet is extracted and sent to Stages 2 and 3, the forwarding table unit and the crossbar arbitration unit, respectively. After reserving the crossbar, in Stage 4, the packet is forwarded from the input port VL to the crossbar. Note that WRR scheduling is used to service packets arriving at Stage 4. In the last stage, VL arbitration using WRR, as specified in the IBA specification, selects one packet from the VLs to transfer to the following switch or CA.

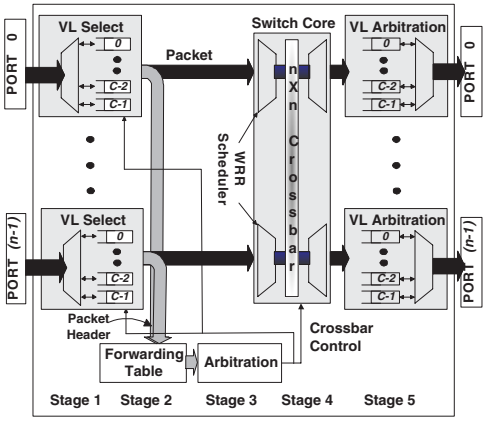


Figure 1. A 5-Stage Pipelined Switch Model

2.3 HCA Architecture

HCAs are used for attaching processing nodes to an IBA SAN. Recent studies have shown that QoS provisioning in the NIC is critical for supporting integrated traffic [6, 7]. A QoS-capable HCA is here a natural choice.

Figure 2 shows a modified HCA architecture for QoS support [7]. To send packets, a consumer creates one or more Queue Pairs (QP) in an HCA. A QP consists of two work queues: one for send operations and the other for receive operations.

We extend the original HCA design to include a prioritized QP scheduling structure to support customized traffic transfer. The structure has a queue for each traffic class. The HCA firmware decides which QP to service in FCFS order based on their priority, and programs the host DMA engine to transfer the packet to the appropriate VL in the HCA port (The HCA also has 16 VLs.). This helps in transferring higher priority packets first to the VLs, where they are scheduled using the WRR scheme to be pushed to the network.

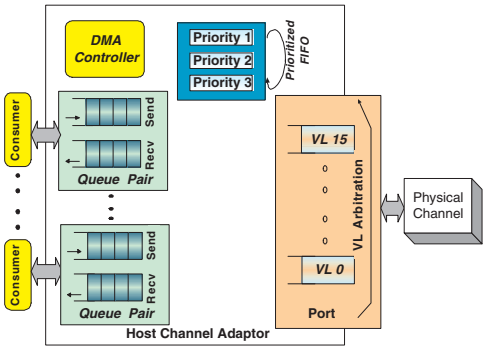


Figure 2. InfiniBand HCA with QoS Support

3 Performance Enhancement Techniques

In this section, we first discuss two deterministic routing algorithms, SPF and Up/Down [3], as the basic packet forwarding scheme in IBA. Then three different approaches to improving the performance of IB-based SANs are presented: multipathing, packet dropping, and multicasting.

3.1 Deterministic Routing Algorithm

IBA specifies the use of forwarding tables in switches to route packets. Each entry in the forwarding table has a destination ID and a corresponding output port number. The construction of the forwarding table is left to designers. We study two routing algorithms for table construction. The first one is called the Shortest Path First (SPF) algorithm; also known as the Dijkstra's algorithm. SPF is used in the Open Shortest Path First (OSPF) protocol for the Internet [8], and is suitable for irregular topologies. In SPF, each switch/router maintains an identical database describing the topology of the network. From this database, a forwarding table is constructed by calculating a shortest path tree. SPF recalculates routes quickly whenever the topology changes.

Next, we consider Up/Down routing [3]. Up/Down routing can be used as a basis of various deadlock-free routing algorithms in irregular networks. This scheme needs construction of a spanning tree with *up* and *down* channels. When the destination node is a descendant node, a packet is forwarded using down channels. Otherwise, a packet always traverses using the up channels first. Once a down channel is selected, an up channel cannot be used for forwarding the packet. Since the original Up/Down routing algorithm allows to use channels in a spanning tree only for routing, the other channels in the network are forbidden to route packets. Several studies [9, 10] have attempted to solve this problem.

Lopez, et al. [11] suggested using a deadlock-free Up/Down routing in irregular networks to build forwarding tables compatible with the IBA specification. Unfortunately, the proposed scheme is sub-optimal in finding a path between a source and a destination. In this paper, we compare both these schemes.

3.2 Multipathing

The IBA specification outlines using alternate paths for a given source-destination pair to improve path availability. For connected transport services, IBA supports Automatic Path Migration (APM), where a channel's traffic may move to a *pre-determined* alternate path in the presence of faults in the original path. The initial alternate path is established at the connection setup, and if a migration occurs to this path due to any fault, an additional alternate path needs to be specified before enabling the migration. This implies

that an alternate path should always be stored for an outgoing connection. APM should be supported by the verb layer[1] in HCAs and TCAs for Reliable Datagram and Reliable/Unreliable Connection services [1]. In this paper, we show that the multipathing concept can be used for performance improvement by reducing network congestion.

Multipathing in IBA is provided by the low-order bits of the DLID (Destination Local ID) field, referred to as Path Bits, which determine the path taken through the fabric. The number of path bits to identify multiple paths is topology dependent. Also, it is not clear how these bits are assigned. In the following, we use a graph theoretic analysis to determine the number of path bits and to assign path bits to alternate paths. This is used to construct the multipath forwarding table.

To describe a network as a graph, the terms and the notations used in [12] are adopted in this paper. Let a network $G = (V, E)$ be a connected and undirected graph, where V is the set of vertices and E is the set of edges consisting of unordered pairs of vertices[2]. Note that $|E| \geq |V| - 1$ for any connected and undirected graph.

We start with a well known theorem [12] in graph theory to determine if a graph has multiple paths between some source and destinations pairs that would need path bits.

Theorem 1 *If a connected, undirected graph G is acyclic, any two vertices in G are connected by a unique simple path.*

If a graph is acyclic, $|E| = |V| - 1$. This implies that if $|E| > |V| - 1$, the graph contains multiple paths for some pair of vertices. So, by simply counting the number of vertices and edges, we can know whether there are multiple paths in a graph. If a graph is not acyclic, the number of bits (p) required to express N alternate paths will be: $p = \lceil \log_2 N \rceil$. Although the number of multiple paths between any pair of vertices can be different, it is complex to implement variable length path bits in the DLID. Therefore, we fix the length of path bits (p) as $p = \lceil \log_2(\max_{\forall i,j} N_{i,j}) \rceil$, where $N_{i,j}$ denotes the number of multiple paths between vertices i and j.

The following theorem shows the relationship between the number of minimal cycles in a graph and the maximum number of multiple paths. The proof for the theorem can be found in [13].

Theorem 2 *If there are C minimal cycles in a connected and undirected graph G, there exists at least one pair of vertices that has 2^C paths.*

Theorem 2 shows that we can compute the length of path bits in the DLID field if we know the number of minimal cycles in a network G. The number of minimal cycles in $G = (V, E)$ can be obtained using algorithm 1.

[1] IBA describes the service interface between an HCA and the operating system by a set of semantics called *verbs*.

[2] In this paper, we assume that all graphs are simple graphs, which means at most one edge connects any two vertices directly.

(1) Construct a BFS (or DFS) tree $T = (V, E_t)$ of G, where E_t is the set of edges in the tree.
(2) Let $E^c = E - E_t$, where E^c is the set of back edges.
(3) Then, $|E^c|$ is the number of cycles in G.

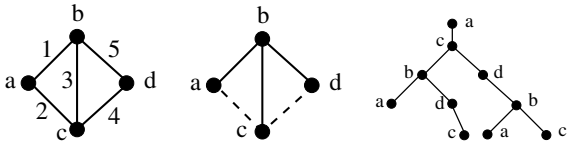
Algorithm 1. Finding the Number of Cycles in a Graph

From Algorithm 1, we also get the set of back edges[3] $E^c = \{(v_1, w_1), (v_2, w_2), \ldots, (v_m, w_m)\}$, which will be used in the next algorithm that enumerates all cycles in a graph G.

(1) For each $(v_i, w_i) \in E^c$, $1 \leq i \leq m$, where $|E^c| = m$, construct a *path tree* T from G as follows.
  (a) The root of T is v_i.
  (b) The child of v_i is w_i only.
  (c) From w_i, find all neighboring vertices except v_i and they become the children of w_i.
  (d) Repeat (c) for each child u of w_i recursively until the same vertex appears twice in the path from the root to itself or the parent of the vertex is the only neighbor.
(2) From T, find the shortest path from the root v_i to the leaf v_i ($v_i w_i \cdots v_i$).
(3) Assign $C_i = v_i w_i \cdots v_i$ as the ith cycle.

Algorithm 2. Finding All Cycles in a Graph

Figure 3(c) shows an example of building the path tree with the back edge (a, c), where the cycle is $acba$.



(a) Original Graph (b) After Algorithm 1 (c) After Algorithm 2

Figure 3. A Path Tree Construction Example

After finding the cycles in a graph, we need to assign path bits to different paths. This is done by considering both the clockwise and counterclockwise directions in a cycle. Let the order in $C_i = v_1 v_2 \cdots v_{k-1} v_k v_1$ be clockwise. Then $\overline{C_i}$ represents a counterclockwise cycle $v_1 v_k v_{k-1} \cdots v_2 v_1$. Note that $\overline{\overline{C_i}} = C_i$.

Before presenting the algorithm to assign the path bits, we need the following two definitions.

Definition 1 *The* **cycle list** α *of a path vw is the set of cycles which contain the path vw.*

For the example in Figure 3 (a), if $C_1 = acba$ and $C_2 = cdbc$, the *cycle list* α of bc is $\{\overline{C_1}, C_2\}$. If a path does not belong to either C_i or $\overline{C_i}$ for $1 \leq i \leq C$, its cycle list will be an empty set. Usually the size of a cycle list is one, implying that the path belongs to only one of the cycles.

[3] A back edge is one that does not belong to the tree.

Definition 2 *The* **cycle list union** *($\sqcup$) of two cycle lists α_1 and α_2 is the set obtained by combining the members, after eliminating the contradicting members such as a and $\overline{a}$. If $a \in \alpha_1$ and $\overline{a} \in \alpha_2$ for some cycle a,*

$$\alpha_1 \sqcup \alpha_2 = \begin{cases} \alpha_1 \cup \alpha_2 - \{a, \overline{a}\}, & |\alpha_1| > 1, |\alpha_2| > 1 \\ \alpha_1 \cup \alpha_2 - \{a\}, & |\alpha_1| > 1, |\alpha_2| = 1 \\ \alpha_1 \cup \alpha_2 - \{\overline{a}\}, & |\alpha_1| = 1, |\alpha_2| > 1 \\ \textit{not defined}, & |\alpha_1| = 1, |\alpha_2| = 1 \end{cases}$$

If there is no such a, $\alpha_1 \sqcup \alpha_2 = \alpha_1 \cup \alpha_2$. The cycle list union of sets $\alpha_1, \alpha_2, \ldots, \alpha_n$ is denoted by $\bigsqcup_{i=1}^{n} \alpha_i$.

In short, the cycle list union gives the set of unique cycles for a given path. The reason for eliminating the contradicting cycles(a,$\overline{a}$) is that a path should not belong to both the clockwise and counterclockwise cycles. The following algorithm uses this to assign the path bits.

For a given path $(v_1 v_2 \ldots v_p)$ where for $1 \le i, j \le p$, $v_i \ne v_j$ and v_1 is the source and v_p is the destination,
(1) For $1 \le i < p$, $\alpha_i =$ a cycle list of $(v_i v_{(i+1)})$.
(2) Make the union of cycle lists $A = \bigsqcup_{i=1}^{p-1} \alpha_i$
(3) Assign path bits $(P_1 P_2 \ldots P_k \ldots P_C)$ as follows:

$$P_k = \begin{cases} 1, & C_k \in A \\ 0, & \overline{C_k} \in A \\ X, & \text{otherwise} \end{cases}$$

Algorithm 3. Constructing Path Bits

We illustrate the path bit assignment using a small irregular network containing two cycles in Figure 3 (a). To decide the length of path bits, we construct a BFS tree with the root b as shown in Figure 3 (b). (a, c) and (d, c) are the back edges. Algorithm 2 will return the cycles starting with the back edges. Figure 3 (c) shows the path tree after executing Algorithm 2 with the back edge (a, c). There are two lists that end with a. Among them, $acba$ is the shortest cycle, and let us assign this as C_1. Repeating the same procedure with the back edge (c, d), we can find the other cycle $C_2 = cdbc$. There is another larger cycle $acdba$, but since it can be obtained from the combination of the two smaller cycles, we don't use it. The following example shows how to assign path bits for a path bca. For path bca, we need to find the cycle lists for bc and ca, which are $\{\overline{C_1}, C_2\}$ and $\{\overline{C_1}\}$, respectively, and the cycle list union is $\{\overline{C_1}, C_2\}$. Using algorithm 3, the path bits of bca are 01.

3.3 Multipath Forwarding Table Construction

From the previous algorithms, we can build the multipath forwarding table for any network. Let us illustrate it using Figure 3 (a). For an entry b in the table for vertex a, we can find all paths from a to b and decide an output port number and path bits for each path using Algorithm 3. There are three paths from a to the destination b: ab, acb and $acdb$. Their path bits are $0X$, 10 and 11, respectively. Therefore, the DLIDs for their paths would simply be $b0X$, $b10$ and $b11$. But since $b10$ and $b11$ have the same output port number 2, after combining them into $b1X$, we can have only two entries for destination b as shown in Figure 4 (c). It is clear that only the first path bit will be examined in vertex a. Thus, the number of entries in the multipath forwarding table for a vertex n is ($2^{\text{the number of cycles containing } n} \times$ the total number of vertices). Note that the size of multipath forwarding table is not constant for all nodes since it depends on the number of cycles containing the vertex. In Figure 4 (c), the table has 6 ($2^1 \times 3$) entries, since node a in Figure 3 (a) belongs to only one cycle.

According to the IBA specification, all zero values in path bits indicate that the local identifier is equal to the base LID. But in our scheme, all zero values may designate one of the paths. For our scheme, we need one additional bit to indicate whether multipathing is enabled. Instead, we can use the path bits in SLID (Source Local Identifier) for this purpose, since a path is defined by the tuple <SLID, DLID, SL> [1]. In Figure 4 (c), we have two path bits. To indicate whether multipathing is used, we need 3 path bits, or we can use the path bits of the corresponding SLID. For example, if the indication bit of a packet is zero, its destination is c and the default routing algorithm is SPF. The output port number of the packet will be '2' as shown in Figure 4 (c).

In Section 3.1, we examined the SPF and Up/Down routing algorithms. The corresponding forwarding tables for the two algorithms are given in Figure 4 (a) and (b) for vertex a. Since our multipath forwarding table contains all possible paths in the network, we can also indicate paths, which are used for each routing algorithm by adding a flag bit in the forwarding table. Note that we need at least one flag bit for the default routing algorithm. With multiple flag bits, we can accommodate different routing algorithms in one forwarding table by simply changing the flag bits, instead of reconstructing the table, as shown in Figure 4 (c). This makes the forwarding table more versatile since we can choose a different routing algorithm depending on the type of traffic class or network dynamics. We call this routing scheme that uses the multipath forwarding table as *multipath routing*. The following section describes multipath routing for a mixed type of traffic.

DLID	Output Port No
b	1
c	2
d	2

(a) SPF Table

DLID	Output Port No
b	1
c	2
d	1

(b) Up/Down Table

DLID	Output Port No	Up/Down	SPF
b0X	1	✓	✓
b1X	2		
c0X	1		
c1X	2	✓	✓
d0X	1	✓	
d1X	2		✓

(c) Multipath Table

Figure 4. Forwarding Table

3.4 Multipath Routing for Mixed Traffic

IBA specifies four kinds of traffic; Reliable/Unreliable Connection and Reliable/Unreliable Datagram. Unreliable connection does not require error-free transmission, while reliable connection does. But both of connection-based traffic requires in-order delivery. Unreliable datagram does not need in-order delivery, while reliable datagram does. The packets of unreliable datagram can be dropped in the networks. We will show multipath routing for each traffic in the following.

There are two ways to facilitate multipathing for Reliable/Unreliable Connections. First, a path/connection is chosen adaptively using a probe packet prior to data transmission, and the remaining data packets use that selected path. The second option is that a packet in a connection can choose its own path adaptively (thus packets can be delivered out-of-order), and the receiver reorders packets using the sequence number in each packet header. The latter approach violates the IBA in-order delivery specification and is not considered here. To use multipathing for connection-based traffic that requires QoS guarantees, the first option is the most viable solution. (The criteria for path selection is a critical issue in QoS routing. Here, we simply use the number of hops to select the best path.)

If the reservation is done successfully with the probing scheme, the destination will send an ACK packet that contains the path bits of the selected path from the source to the destination. If the probe is rejected at an intermediate node, it will send a NACK packet back to the source to release the reserved resources. The back path of a NACK packet can be decided using the path bits again.

For reliable datagrams, which should be delivered in order, we can also send a probe packet and setup the path. However, since datagram traffic usually is short-lived, probe/ACK packets are unnecessary overheads. There are two ways we can deliver reliable datagrams in order without a probe packet. The first one is using a deterministic routing algorithm. But this may cause congestion by overusing some links. The other approach is to use the Verb layer to select one of the alternate paths for a source and a destination, putting the path bits of the selected path in the DLID, and enabling the path bits in SLID to indicate that the packet will traverse through multipath routing. We implement the latter approach here.

Since unreliable datagrams can be delivered out of order, a packet can select a path adaptively in each switch. We need a path selection criteria for both types of datagrams. If we keep the global network information in each HCA and switch, we may use them for path selection. But, keeping the global information is expensive. Vaidya, et al.[4] have presented three traffic-sensitive path selection heuristics (Least Frequently Used (LFU), Least Recently Used (LRU), Max Credit) for improving performance. Path selections in their work are done by investigating the local link status. Since LFU and Max Credit are better performing heuristics over all types of traffic in [4], we have implemented LFU and Max Credit for the path selection criteria in our study.

3.5 Packet Dropping in a Switch

The IBA specification [1] cites several cases such as detection of a corrupt CRC, expiry of the Switch Lifetime Limit (SLL) and expiry of the Head of Queue Lifetime Limit (HLL), when a packet should be dropped in a switch. SLL is defined as 4.096μsec $\times 2^{LV}$, where $0 \leq$ LV ≤ 19. If LV > 19, then SLL is to be interpreted as infinite. HLL is defined as 4.096μsec $\times 2^{HV}$, where $0 \leq$ HV ≤ 19. If HV > 19, then HLL is to be interpreted as infinite. SLL indicates the time limit of a packet after it arrives at a switch, implying that if a packet stays in a switch for longer than SLL, it may be discarded. On the other hand, HLL indicates the time limit of a packet after it arrives at the head of a queue in a switch.

According to [14], deadlocks in a network rarely occur if it has sufficient routing freedom and the freedom is fully exploited by the routing algorithm. Also, deadlock recovery schemes can show better performance than deadlock avoidance schemes. Therefore, instead of using rather a complex deadlock detection scheme, we use the deadlock-prone, but simple SPF algorithm and selectively drop packets that stay in a queue for a sufficiently long time. The motivation here is to examine the impact of packet dropping in avoiding congestion and deadlock.

In this paper, we use the HLL-based packet dropping scheme. When a packet header arrives at the head of a queue (VL in IBA term), its current time is recorded in the storage, called *HQLifetime*, which is attached to each VL. Since a VL can contain multiple packets, *HQLifetime* only indicates the arrival time of the packet header at the head of the queue. Thus, whenever a packet is sent to the neighboring switch (or HCA) or dropped, a new arrival value is stored in *HQLifetime*. When the sum of the value stored in *HQLifetime* and HLL becomes less than the current clock value, the packet is removed from the queue.

We use selective packet dropping in that only real-time and best-effort packets can be dropped from the network. Moreover, I frame packets of MPEG-2 streams are not dropped to maintain correct/uninterrupted media stream transfer. For the simulation purpose, the first and last packets of any MPEG-2 B and P frames are not dropped. For dropped packets, no further recovery procedure is pursued at the link layer.

3.6 Multicasting Support

Multicasting is a desired feature and described in details in the current IBA specification [1]. Three issues need to be considered to support multicasting. These are building

and maintaining a multicast forwarding table, implementation of the replication mechanism, and bandwidth allocation policy for multicast and unicast traffic. In this paper, we examine the source-based multicasting in contrast to the core-based scheme since it is more efficient and simpler for smaller subnets [15, 16]. Thus, the multicast forwarding table is obtained from the source-based distribution tree algorithm assuming static membership groups [15]. Here, we only focus on the replication mechanisms and the bandwidth allocation policies.

We analyze both synchronous and asynchronous replication mechanisms. Synchronous replication begins when a multicast packet arrives at Stage 4 in Figure 1. Synchronous Replication requires that multicast packets proceed in lock-step. If any of the destination ports is not available, the multicast packet will hold all other output ports and will wait for the unavailable port. Thus, this replication is susceptible to deadlocks. We can prevent deadlocks with synchronous replication by prioritizing the output port assignment. Asynchronous Replication, on the other hand, allows multicast packets to be forwarded to a subset of the requested output ports [17]. This scheme requires extra buffers large enough to store the largest packet in the switch [18], and complex buffer control mechanisms.

Efficient bandwidth allocation to multicast and unicast traffic is another research issue that needs investigation to improve performance. Legout, et al. consider three different bandwidth allocation schemes [19]: (i) allocate the same bandwidth to unicast and multicast flows; (ii) allocate multicast bandwidth linearly proportional to the number of destinations; and (iii) allocate multicast bandwidth proportional to the logarithm of the number of destinations. It was shown that the third scheme performs the best with a minimal impact on unicast traffic. In an IBA-style switch, since the bandwidth is allocated to each VL, not to each flow, we need to assign separate VLs to multicast traffic. The number of VLs assigned for multicast traffic will also affect the overall performance. We re-examine the three different bandwidth allocation schemes with respect to the assignment of VLs and study the performance and QoS implications in Section 5.2.

4 Experimental Platform

One of the main motivations of our work is to develop a comprehensive simulation testbed for IBA. Our testbed includes packet-level switches and HCAs conforming to the IBA specification. Table 1 shows the main parameters used for simulation. For our experiments, we simulated 15-node and 30-node irregular networks designed using 5-port switches and HCAs. The results with 30-node networks can be found in [13].

The workload includes packets from real-time VBR traffic, best-effort traffic, and control traffic. The VBR traffic is generated as a stream of packets between a pair of communicating (source-destination) processors. The traffic in each stream is generated from seven MPEG-2 traces [20], where each trace has different bandwidth requirement. Each stream generates 30 frames/sec, and each frame is divided into fixed-size packets, where each packet consists of the MTU and the header. Once the input VL for a connection is determined, the destination processor is picked randomly using a uniform distribution of all nodes, and the destination VL is also drawn randomly from a uniform distribution of the VLs available for the VBR traffic.

Physical Link Bandwidth	2.5 Gbps
Number of Physical Links	5
Number of VLs/Physical Link	16
HCA Buffer Size	8 MB
LRH, BTH, and CRC fields	26 bytes
Real-time, Best-effort Traffic MTU	1024 bytes
Control Traffic MTU	256 bytes
Input VL Buffer Size	4200 bytes
Output VL Buffer Size	4200 bytes

Table 1. IBA Simulation Testbed Parameters

The best-effort traffic (could be Reliable/Unreliable Datagram) is generated with a given injection rate λ, and follows the Poisson distribution. The size of best-effort packets is assumed to be fixed, and a destination is picked using a uniform distribution. Control traffic, called management traffic in IBA, is typically used for network configuration, congestion control, and transfer of other control information. We generate a control packet every 33.3 msec. This traffic has the highest priority in our model, and always uses VL_{15}.

The important output parameters measured in our experiment are Deadline Missing Probability (DMP) of delivered MPEG-2 frames, average Deadline Missing Time (DMT) of deadline missing frames, and average packet latency for all types of traffic. The DMP is the ratio of the number of frames that missed their deadlines to the total number of delivered frames. The deadline for each frame is determined by adding 33.3 msec to the previous deadline. However, if a previous frame misses its deadline, a new deadline is set by adding 33.3 msec to the arrival time of the previous frame.

5 Performance Results

We have conducted extensive evaluation of the proposed designs for different networks and workload conditions. Here we include only a subset of the results and discuss them in two main sections due to space limitation. (Additional results can be found in [13].) Most of the results presented here are for real-time to best-effort ratio of 70:30.

5.1 Comparison of Routing Schemes

In this section, we compare the SPF, Up/Down and the multipath routing algorithms with/without the selective packet dropping scheme. To construct the multipath forwarding table for our 15-node network that has 9 cycles, we use 9 path bits. The number of entries in the forwarding tables is either 30 (= 15 $\times 2^1$) or 60 (= 15 $\times 2^2$) based on (the total number of nodes $\times 2^{(\text{the number of cycles containing the node})}$).

Figure 5 shows the results from seven different routing schemes. These are: (i) Up/Down; (ii) SPF; (iii) SPF with packet dropping (SPF+Drop); (iv) multipath routing using LFU as the path selection criteria (Multipath+LFU); (v) multipath routing using LFU and packet dropping (Multipath+LFU+Drop); (vi) multipath routing using Max-Credit as the path selection criteria (Multipath+Credit); and (vii) multipath routing using Max-Credit and packet dropping (Multipath+Credit+Drop). The first three experiments use deterministic routing algorithms for all three types of traffic. The last four experiments use multipath routing. For real-time traffic, we select the shortest path from among the available paths. For best-effort traffic (Reliable/Unreliable Datagrams), a reliable datagram randomly chooses one of the 2^9 paths in the verb layer, while an unreliable datagram selects a path adaptively using LFU or Max-credit. The control traffic uses the shortest path via VL_{15}, as required.

Figures 5 (a) and (b) show the DMP and the DMT of real-time traffic, while Figures 5 (c), (d), and (e) show the average packet latency for the three types of traffic. All the graphs indicate that the SPF routing outperforms the Up/Down routing, because the latter is a suboptimal solution. In fact, the performance of Up/Down is heavily swayed by the topology of the Up/Down tree. We have tested the network with three different roots and chose the best case results for the Up/Down routing. The proposed congestion avoidance techniques (multipath routing and packet dropping) do not provide noticeably better performance in terms of the DMP and the DMT at a low injection rate (up to 40%). However, as the load increases, these techniques become very effective in reducing those metrics (50% $\sim$ 65% for the DMP and 30% $\sim$ 55% for the DMT compared to SPF at the injection rate of 60%). The results indicate that multipathing and packet dropping help in jitter-free delivery of media streams at relatively high load.

It is interesting to note that (Multipath+LFU) helps to improve the performance of real-time and control traffic, while (Multipath+Credit) aids to reduce the average packet latency of best-effort traffic. This is because LFU tries to select the least frequently used path that other higher priority traffic (real-time and control in our study) does not use. On the other hand, Max-Credit only checks the buffer availability of the neighboring switch, and avoids selecting the path heavily used by best-effort traffic, thereby improving the average packet latency of best-effort traffic. Both (Multipath+LFU) and (Multipath+Credit) reduce the average packet latency by 40% $\sim$ 55% for control traffic, by 40% $\sim$ 65% for real-time, and by 30% $\sim$ 55% for best-effort traffic, respectively, compared to that of SPF at an injection rate of 50% or higher. With packet dropping, these reductions increase up to 80% for control, 90% for real-time, and 60% for best-effort traffic, respectively.

5.2 Multicast Results

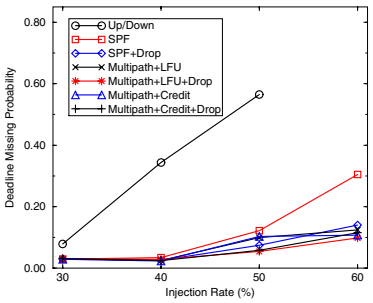
Figure 6 shows the simulation results of four multicasting schemes in a 15-node irregular network. These are: (i) Replicated Unicast without any hardware support for multicasting; (ii) Synchronous Replication; (iii) Asynchronous Replication without Central Buffer; and (iv) Asynchronous Replication with Central Buffer. In this experiment, 20% of the real-time traffic has multi-destinations. We implemented the source-based tree algorithm, which has 3×15 entries in the multicast forwarding table of each node.(We have 3 groups each with two or four multicast members.)

Figures 6 (a) and (b) show the DMP and DMT of real-time traffic. Replicated Unicast has the worst performance as expected. All these replication schemes provide much better performance than the Replicated Unicast in terms of DMP and DMT for real-time traffic. The average packet latencies of all three kinds of traffic also benefit due to replication (Figures 6 (c), (d), and (e)). Asynchronous Replication with a central buffer provides slightly better performance over the other two replication schemes. But this improvement comes with the overhead of a large buffer and complex control circuitry. We did not observe deadlocks with the Synchronous Replication because preemption between multicast packets is resolved in the FCFS order. By reducing the real-time traffic load, the replication schemes also help control and best-effort traffic latencies as shown in Figures 6 (d) and (e)[4].

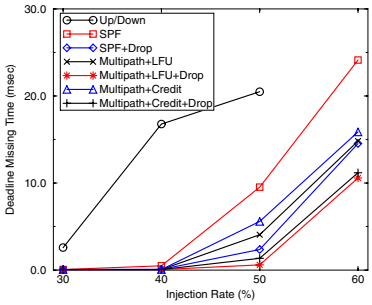
Next, we experiment with the three bandwidth allocation schemes discussed in Section 3.6 with the asynchronous replication. We show results for four different bandwidth allocation schemes in Figure 6 (f): (i) the same bandwidth to unicast and multicast without VL separation; (ii) the same bandwidth to unicast and multicast with separate VLs; (iii) bandwidth linearly proportional to the number of destinations; and (iv) bandwidth proportional to the logarithm of the number of destinations. For (iii) and (iv), which allocate more bandwidth to multicast traffic, we also allocate separate VLs for multicast traffic.

To examine the effect of bandwidth allocation on both of multicast and unicast traffic, we plot the results of multicast and unicast separately in Figure 6 (f). The first four bars are the average packet latencies of multicast traffic with different bandwidth allocation policies, while the other four are

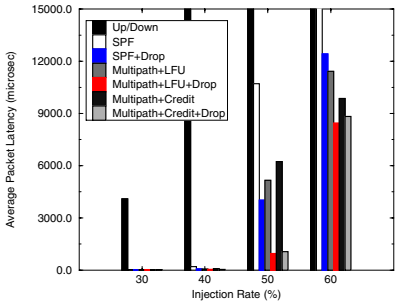
[4]We have used a logarithmic scale in Figure 6 (d) and (e).



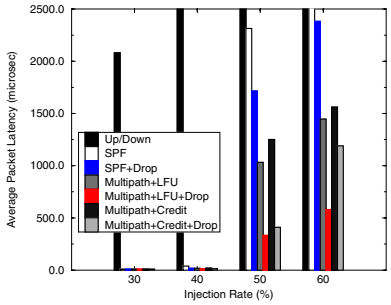
(a) Deadline Missing Probability



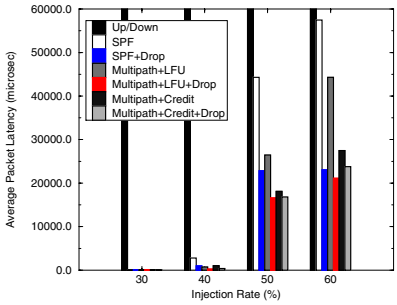
(b) Deadline Missing Time



(c) Real-Time Traffic



(d) Control Traffic



(e) Best-Effort Traffic

Figure 5. Comparison of Various Routing Algorithms in a 15-Node Irregular Network

those for unicast traffic.

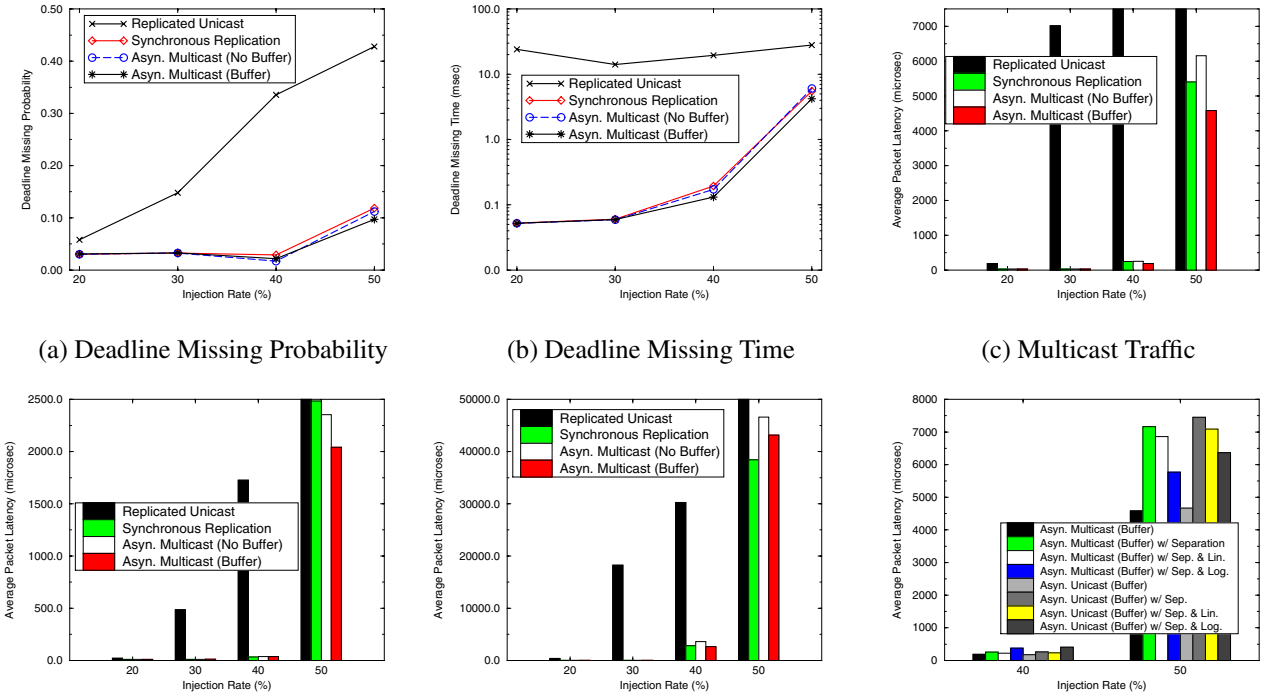
With a low load, the three bandwidth allocation schemes ((ii), (iii) and (iv)) provide almost the same performance. But at higher load, the performance of these schemes are worse compared to the first scheme, although the logarithmic allocation (iv) provides the best performance among the three schemes. For the replication scheme (i), we give multicast traffic higher priority over unicast traffic in reserving the output VLs. Without separation of VLs, multicast traffic can preempt all unicast traffic, while with separate VLs, multicast traffic uses the limited number of VLs. Therefore, the first scheme (i) results in better performance. The results bring in an interesting observation that giving a separate Service Level (separate VLs) to multicast traffic may not be a good idea in an IBA-style switch.

6 Concluding Remarks

We have presented a comprehensive simulation testbed along with several performance enhancement techniques for IB-based SANs. These techniques include exploiting the SPF routing algorithm as the default packet forwarding scheme, adopting a novel and practical multipathing scheme to choose alternate paths, implementing a selective packet dropping mechanism in a switch/router, and implementing several hardware supported multicasting techniques. Although the packet dropping and multicasting concepts are not new, the motivation here is to quantify their contributions to performance enhancement by integrating with suitable routing techniques.

The important conclusions of this work are the following: First, the SPF routing algorithm seems to be a better choice for IBA-style SANs compared to Up/Down routing. SPF routing may lead to deadlocks, but when it is coupled with the packet dropping mechanism, deadlocks can be avoided. Second, the multipath routing using LFU or Max Credit as the path selection criteria along with selective packet dropping provides the best performance for integrated traffic. Finally, the hardware support for multicasting is necessary to assure QoS delivery of real-time traffic and to reduce the network load. While asynchronous and synchronous replication schemes are equally competitive, synchronous replication may be a better solution since it doesn't require large buffers.

One of the major contributions of this research is the IBA simulation testbed, which is an essential tool to evaluate various design tradeoffs. To the best of our knowledge, there is no such platform available at present. We plan to make our simulator publicly available for the research community.



(a) Deadline Missing Probability (b) Deadline Missing Time (c) Multicast Traffic

(d) Control Traffic (e) Best-Effort Traffic (f) Different Bandwidth Allocations

Figure 6. Evaluation of Multicasting in a 15-Node Irregular Network

References

[1] InfiniBand Trade Association, "InfiniBand Architecture Specification, Volume 1, Release 1.0," October 2000. Available from `http://www.infinibandta.org`.

[2] J. Pelissier, "Providing Quality of Service over InfiniBand Architecture Fabric," in *Proc. of Hot Interconnects*, August 2000.

[3] M. D. Schroeder, et al., "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," Tech. Rep. SRC research report 59, DEC, 1990.

[4] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, "LAPSES: A Recipe for High Performance Adaptive Router Design," in *Proc. of HPCA*, pp. 236–243, January 1999.

[5] W. J. Dally, "Virtual-Channel Flow Control," *IEEE TPDS*, vol. 3, pp. 194–205, May 1992.

[6] S. S. Mukherjee and M. D. Hill, "A Survey of User-level Network Interfaces for Systems Area Networks," Tech. Rep. 1340, Computer Science Dept., Univ. of Wisconsin-Madison, February 1997.

[7] K. H. Yum, E. J. Kim, and C. R. Das, "QoS Provisioning in Clusters: An Investigation of Router and NIC Design," in *Proc. of ISCA*, pp. 120–129, June 2001.

[8] J. Moy, *OSPF Version 2*. The Internet Society, 1998. RFC 2328.

[9] J. Wu and L. Sheng, "Deadlock-Free Rrouting in Irregular Networks Using Prefix Routing," Tech. Rep. 99-19, DIMACS, April 1999.

[10] F. Silla and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," in *Proc. of CANPC'97*, February 1997.

[11] P. López, J. Flich, and J. Duato, "Deadlock-Free Routing in InfiniBand through Destination Renaming," in *Proc. of ICPP*, pp. 427–434, September 2001.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1999.

[13] E. J. Kim, et al., "Performance Enhancement Techniques for QoS Provisioning in InfiniBand^{tm} Architecture," Tech. Rep. CSE-02-005, Pennsylvania State University, University Park, PA, February 2002.

[14] T. M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks," in *Proc. of ISCA*, pp. 38–49, June 1997.

[15] J. E. Klinker, "Multicast Tree Construction in Directed Networks," in *Proc. of IEEE Military Comm. Conf.*, pp. 496–500, October 1996.

[16] T. Billhartz, et al., "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols," *IEEE JSAC*, vol. 15, pp. 304–315, April 1997.

[17] C. B. Stunkel, R. Sivaram, and D. K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact," in *Proc. of ISCA*, pp. 50–61, June 1997.

[18] C. B. Stunkel, et al., "The SP2 High-Performance Switch," *IBM Systems Journal*, vol. 34, no. 2, pp. 185–204, 1995.

[19] A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth-Allocation Policies for Unicast and Multicast Flows," *IEEE/ACM Trans. on Networking*, vol. 9, pp. 464–478, August 2001.

[20] M. B. Caminero, et al., "Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic," in *Proc. of CANPC'99*, pp. 62–76, January 1999.