Exploring IBA Design Space for Improved Performance

Eun Jung Kim, *Member*, *IEEE*, Ki Hwan Yum, *Member*, *IEEE*, Chita R. Das, *Fellow*, *IEEE*, Mazin Yousif, *Senior Member*, *IEEE*, and José Duato, *Member*, *IEEE*

Abstract—InfiniBand Architecture (IBA) is envisioned to be the default communication fabric for future system area networks (SANs) or clusters. However, IBA design is currently in its infancy since the released specification outlines only higher level functionalities, leaving it open for exploring various design alternatives. In this paper, we investigate four corelated techniques for providing high and predictable performance in IBA. These are: 1) using the Shortest Path First (SPF) algorithm for deterministic packet routing, 2) developing a multipath routing mechanism for minimizing congestion, 3) developing a selective packet dropping scheme to handle deadlock and congestion, and 4) providing multicasting support for customized applications. These designs are implemented in a pipelined, IBA-style switch architecture, and are evaluated using an integrated workload consisting of MPEG-2 video streams, best-effort traffic, and control traffic on a versatile IBA simulation testbed. Simulation results with 15-node and 30-node irregular networks indicate that the SPF routing, multipath routing, packet dropping, and multicasting schemes are quite effective in delivering high and assured performance in clusters.

Index Terms—InfiniBand architecture, multicasting, multipathing, NIC/HCA, packet dropping, quality of service, system area network.

1 Introduction

Enabling technology and high-performance computing market force have pushed the processing power as predicted by Moore's law. It is projected that by 2010-2013, the 0.05 micron CMOS-based technology will push the clock rate, chip complexicity, power requirement, processor speed, and memory capacity almost to their limits [2]. In anticipation that the need for the growth in processing power will continue in our daily lives, researchers have already started exploring other emerging technologies [3], [4] to supplement/replace CMOS-based computing. All these cross-cutting issues coupled with innovations in processor architecture and memory design will increase the basic computing power. However, one of the critical issues missing from this high-performance computing equation is high bandwidth networking and I/O support. The improvement in scalable I/O has not kept pace with that of the processor and memory design, and the traditional PCI bus is still the main interface to the outside world. Although PCI Express has been proposed to mitigate the interface problem, it is clearly not the solution. Similarly, the prevailing communication media (like Ethernet, Fiber channel, and SCSI) are not adequate to suffice future communication demands. It thus seems that the bottleneck in high-performance computing will not be processor nor memory subsystems, but the underlying communication network.

Realizing the seriousness of the interconnect technology, a new communication standard, called InfiniBand Architecture (IBA), was proposed in 1999. IBA has been proposed as a new communication standard to design system area networks (SANs) for scalable, high-performance clusters. IBA has tried to solve bandwidth, scalability, reliability, and standardization issues under one unifying design. The recent IBA specification [5] has augmented the earlier one with enhanced features such as Congestion Management, Quality of Service (QoS), and Router Management. QoS is becoming an essential part of the IBA framework [6] because of the sophistication of services that will be supported by clusters connected through SANs.

The motivation of this research 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 setup in the routing/forwarding table is left open for the designers. Moreover, we believe that the multipathing mechanism cannot only be used for fault tolerance, but also for congestion avoidance to improve performance. Therefore, it is essential to understand the

- *E.J. Kim is with the Department of Computer Science, Texas A&M University, H.R. Bright Building, Room 427C, College Station, TX 77843-3112. E-mail: ejkim@cs.tamu.edu.*
- *K.H. Yum is with the Department of Computer Science, University of Texas at San Antonio, 6900 North Loop 1604 West, San Antonio, TX 78249-0667. E-mail: yum@cs.utsa.edu.*
- *C.R. Das is with the Department of Computer Science and Engineering, Pennsylvania State University, 354F Information Science and Technology Building, University Park, PA 16802. E-mail: das@cse.psu.edu.*
- *M. Yousif is with the Corporate Technology Group, Intel Corporation, Hillsboro, OR 97124. E-mail: mazin.s.yousif@intel.com.*
- *J. Duato is with the Department of Computer Engineering, Universidad Politecnica de Valencia, Camino de Vera, 14, 46071-Valencia, Spain. E-mail: jduato@disca.upv.es.*

Manuscript received 29 Mar. 2005; revised 26 Mar. 2006; accepted 31 May 2006; published online 9 Jan. 2007.
Recommended for acceptance by J. Hou.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0218-0305.
Digital Object Identifier no. 10.1109/TPDS.2007.1010.

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. We suggest that packet dropping can also be used for deadlock avoidance, thus, instead of using a complex deadlock-free algorithm, one 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.

For the first issue, we examine two deterministic routing algorithms, called Shortest Path First (SPF) and Up/Down [7], that are suitable for irregular networks. While IBA specifies use of forwarding tables in switches to route packets, the construction of the forwarding table is left to designers. We show how the forwarding table can be constructed for implementing the two routing algorithms.

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 using the required number of extra bits to specify the alternate paths. Then, we 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 [8]. We show that multipathing can be used for improving the performance by avoiding congestion.

For the third issue, we investigate the implementation and impact of packet dropping in IBA. The aim of implementing packet dropping in an IBA-based SAN is two-fold: 1) we can avoid deadlocks possible with the SPF routing algorithm and 2) we can improve performance by dropping *not-so-critical* packets. We use the Head of Queue Lifetime Limit (HLL)-based packet dropping scheme as per the IBA specification [5]. With this scheme, a packet is dropped from the network/switch if its lifetime exceeds a certain threshold.

Finally, for the fourth issue on multicasting, we examine the implementation complexities of different multicasting schemes and their performance implications. We investigate three related issues: building and maintaining a multicast forwarding table, implementation of the replication mechanism, and bandwidth allocation policy for multicast and unicast traffic. The multicast forwarding table can be obtained from the source-based distribution tree algorithm assuming static membership groups [9], [10]. Then, two replicate mechanisms, called synchronous and asynchronous, are studied. We also implement the asynchronous replication scheme with central buffers [11] to provide a separate FIFO queue for multicast packets for each output port.

We have developed a detailed simulator for a pipelined IBA switch and a Host Channel Adapter (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 video 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 outperform 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 can lower DMP and DMT for MPEG-2 streams by about 52 percent 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 percent 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 due to less hardware complexity.

The rest of the paper is organized as follows: In Section 2, we discuss the IBA framework, the switch architecture, and the HCA design used in our study. Section 3 presents the proposed performance enhancement techniques. In Section 4, the experimental platform is discussed. 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)

The IBA specification [5] describes a wired protocol for connecting processor nodes and I/O devices through a fabric, creating a SAN. An InfiniBand fabric includes a number of subnets connected through routers. Within a subnet, switches connect processors and I/O nodes. IBA is based on a packet-switched, point-to-point interconnect technology. Processing nodes are attached to an IBA network through Host Channel Adapters (HCAs) and I/O nodes can be attached to a network through Target Channel Adapters (TCAs).

Virtual Lanes (VLs)[1] provide a mechanism to implement multiple logical flows over a single physical link. They are actually buffers holding incoming and outgoing packets. A port must support at least two VLs and at most 16 VLs. All ports must use VL_{15} for subnet management traffic. A port must also implement at least one, and as many as 15, Data VLs. VL arbitration means selection of a VL to push data to an outgoing link in a switch, router, or Channel Adapter (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

1. VLs are also known as Virtual Channels (VCs) [12].

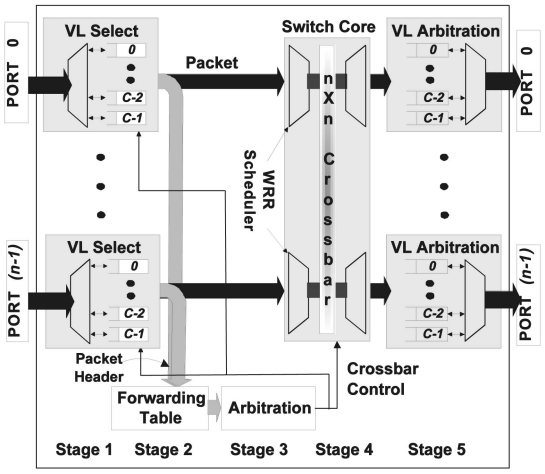


Fig. 1. A 5-stage pipelined switch model.

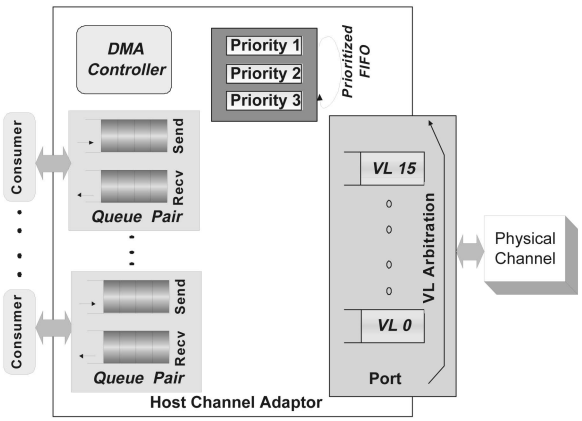


Fig. 2. An InfiniBand HCA with QoS support.

scheduling priority is adopted for different SLs. Next, a Weighted Round Robin (WRR) scheduling is used to schedule packets of the same SL. Additionally, this scheme provides a method to ensure forward progress of the low-priority VLs. Weight calculation, traffic prioritization, and minimum bandwidth guarantee to ensure forward progress are programmable.

2.2 Switch Architecture

The switch used in this paper adopts a five-stage pipelined packet-switched model, as shown in Fig. 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, the packet is forwarded from the input port VL to the crossbar. WRR scheduling is used to service packets arriving at Stage 4. In the final stage, VL arbitration using WRR, as specified in the IBA specification, chooses one packet at a time among multiple VLs to transfer to the next switch or the attached HCA.

There are two kinds of Cyclic Redundancy Code (CRC) checking in IBA. The invariant CRC field (ICRC, 4 bytes) in all IBA packets is constant from end-to-end through all switches and routers in the network. On the other hand, the variant CRC (VCRC, 2 bytes) in all data packets may be regenerated at each link through the subnet. CRC checking in the link layer aims to detect erroneous packets, and when an erroneous packet is found, it is to be discarded from the network. For CRC checking, the whole packet should be stored in the buffer.

Generally, CRC checking should be done at an early processing stage since a corrupt packet does not need to be serviced further. However, under the assumption that packet corruption rarely occurs when a fabric is small, we have decided to defer CRC checking to the last pipeline stage without any performance degradation. With this assumption, it is easier to take advantage of our pipelined model. When a packet header arrives at an input VL, it can be relayed to the forwarding table unit without waiting for the whole packet to reach the input VL. As a result, header processing time in the forwarding table and arbitration units could be overlapped with the waiting time of the remaining part of the packet. The gains from this overlapping become more prominent when the forwarding table is larger, and forwarding table lookup and crossbar arbitration takes more time, as well as when the packet size is larger.

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 [13], [14]. Therefore, a QoS-capable HCA is a natural choice.

Fig. 2 shows our proposed HCA architecture for QoS support [14]. To send packets, a consumer creates one or more Queue Pairs (QPs) in an HCA. A QP actually consists of two work queues: one for send operations and the other for receive operations. The consumer submits a work request (WR), and this is converted to an instruction called a work queue element (WQE) which will be stored in the appropriate work queue. The selected WQE by the HCA causes the packet in the consumer's memory to be DMA'd to the HCA's port.

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 SL so that packets in same SL will be transferred in FCFS order. The HCA firmware decides which queue to service based on its priority, and programs the host DMA engine to transfer the packet to the appropriate VL in the HCA port, where there are also $2 \sim 16$ 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.

In summary, for QoS provisioning, two schedulers are used in the HCA: A prioritized FIFO scheduling selects a WQE from the work queues and a WRR scheduling selects a packet from multiple VLs in the HCA's port.

3 Performance Enhancement Techniques

First, we discuss two deterministic routing algorithms, SPF and Up/Down [7], as the basic packet routing algorthm in IBA. Then, three different approaches to improving the

performance of IB-based SANs (multipathing, packet dropping, and multicasting) and related issues are presented.

3.1 Deterministic Routing Algorithm

IBA specifies 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. 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 [15], 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, utilizing a minimum of routing protocol traffic. SPF routing algorithm can be used to find a minimal cost path, where the cost factor could be user specifiable. Here, we use the number of hops between a source and a destination as the cost factor.

The second routing algorithm considered here is Up/Down routing [7]. Up/Down routing is known as a simple routing algorithm that 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 the 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, the other channels in the network are forbidden to route packets. Several studies [16], [17] have attempted to solve this problem.

López et al. [18] 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 suboptimal in finding a path between a source and a destination.

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 *predetermined* 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 is supported at the verb layer[2] in HCAs and TCAs for Reliable Datagram and Reliable/Unreliable Connection services [5]. 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 DLID bits used as Path Bits is variable, allowing it to be chosen based on the topology. However, 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 [19] 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.[3] Note that $|E| \geq |V| - 1$ for any connected and undirected graph.

We start with a well-known theorem [19] in graph theory to determine if a graph has multiple paths between some source and destination 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 tell 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 number of 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 cycles in a graph and the maximum number of multiple paths:

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

Proof. We prove this by induction on the number of cycles and edges. When $C = 0$, there is no cycle in a graph since $|E_0| = |V| - 1$, and each pair of vertices in $G_0 = (V, E_0)$ is connected by a unique path by Theorem 1. When $C = 1$, there is one cycle in $G_1 = (V, E_1)$, where $E_1 = E_0 \cup \{e_1\}$; that is, e_1 is the additional edge that forms the cycle in G_1. It is clear that any two vertices of a cycle are connected by two paths.

Now, assume that there exists at least one pair of vertices which has 2^k paths in a connected and undirected graph $G_k = (V, E_k)$ having k cycles. We must prove that if there are $(k+1)$ cycles in a connected and undirected graph $G_{k+1} = (V, E_k \cup \{e_{k+1}\})$ formed by adding an edge e_{k+1} to G_k, there exists at least one pair of vertices that has 2^{k+1} paths. Assume that there are 2^k paths between vertices v and w in the graph G_k. We need to consider two cases for the new edge e_{k+1}. If e_{k+1} forms a new cycle that includes one of the vertices in the path between v and w excluding v and w, there will be 2^{k+1} paths between v and w (see Fig. 3a). If the new cycle includes none of the vertices in the path, but v, w, or other vertices, there will be 2^{k+1} paths between v (or w) and another vertex, which lies in the newly formed cycle (see Figs. 3b and 3c). □

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

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

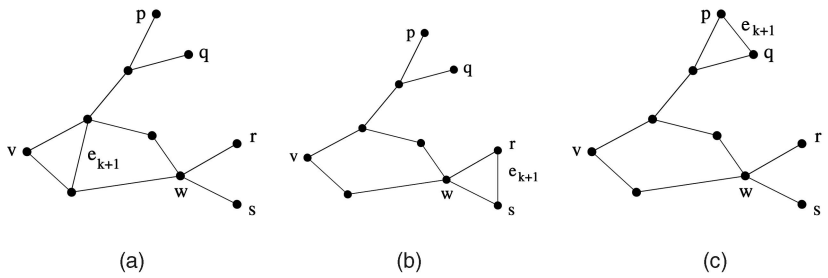


Fig. 3. Examples for counting multipaths.

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

(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[4] $E^c = \{(v_1, w_1), (v_2, w_2), \ldots, (v_m, w_m)\}$, which will be used in the next algorithm that enumerates all cycles in a graph G. Although there is another algorithm to generate all cycles in a graph [20], Algorithm 2 is different from it in that we first construct a path tree so that we can number each cycle.

(1) For each $(v_i, w_i) \in E^c$, $1 \le i \le m$, 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

Fig. 4 shows an example of building the path tree with the back edge (a, c), where the cycle is $acba$.

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 the 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 an edge vw is the set of cycles which contains the edge vw.*

For example, in Fig. 4a, if $C_1 = acba$ and $C_2 = cdbc$, the *cycle list* α of bc is $\{\overline{C_1}, C_2\}$. If an edge does not belong to either C_i or $\overline{C_i}$ for $1 \le i \le C$, its cycle list will be an empty set. Usually the size of a cycle list is one, implying that the edge belongs to only one of the cycles. If we consider two-dimensional irregular networks, the maximum size of a cycle list will be two. When we do not have any topological restriction, the size can be greater than two.

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,*

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

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 Fig. 4a. To decide the length of path bits, we construct a BFS tree with the root b as shown in Fig. 4b. (a, c) and (d, c) are the back

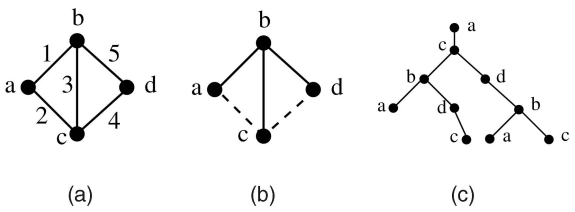


Fig. 4. A path tree construction example. (a) Original graph. (b) After Algorithm 1. (c) After Algorithm 2.

4. A back edge is one that does not belong to the BFS (or DFS) tree.

DLID	Output Port No
b	1
c	2
d	2

(a)

DLID	Output Port No
b	1
c	1
d	1

(b)

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

(c)

Fig. 5. Forwarding table for vertex a. (a) SPF table. (b) Up/Down table. (c) Multipath table.

edges. Algorithm 2 will return the cycles starting with the back edges. Fig. 4c 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 do not use it. The multipath forwarding table for the vertex b can be constructed as follows: There are three paths from b to the destination a; ba, bca, and $bdca$. The following example shows how to assign path bits for a path bca. Since ba lies in C_1, the path bits of the path ba is $1X$. 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. Similarly we can find the path bits of $bdca$ as 00.

3.3 Multipath Forwarding Table Construction

A Multipath Forwarding Table is a forwarding table with path bits in the DLID in order to distinguish it from the one without any multipathing function. It is a linear forwarding table with one entry for each destination. Each path bit setting is counted as a different destination, and the table length is compressed by use of don't-cares in the path bits.

From the previous algorithms, we can build the multipath forwarding table for any network. Let us illustrate it using Fig. 4a. 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 Fig. 5c. 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 a is ($2^{\text{the number of cycles containing } a} \times$ the total number of vertices). Note that the size of a multipath forwarding table is not constant for all vertices since it depends on the number of cycles containing the vertex. In Fig. 5c, the table has 6 ($= 2^1 \times 3$) entries, since vertex a in Fig. 4a 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 may need one additional bit to indicate whether multipathing is enabled or not. Or, we can use the path bits in SLID (Source Local Identifier) for this purpose, since a path is defined by the tuple $\langle \text{SLID}, \text{DLID}, \text{SL} \rangle$ [5]. In Fig. 5c, we have two path bits. To indicate whether multipathing is used for a packet, we can have three 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 Fig. 5c.

In Section 3.1, we examined the SPF and Up/Down routing algorithms. The corresponding forwarding tables for the two algorithms are given in Figs. 5a and 5b 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 Fig. 5c. 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.

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 NACK packet can be decided using the path bits again. (If the path bits from source to the intermediate node is 010, the path bits from the intermediate node to a source will be 101.)

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. [8] 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 [8], we have implemented LFU and Max Credit for the path selection criteria in our study.

Another reason we exclude the LRU scheme is that it requires complex implementation overhead [8]. To implement LFU, we should have only one counter for each VL. As for Max Credit, a switch already have the credits for flow control, so that we can use them without any extra overhead for path selection.

3.5 Packet Dropping in a Switch

The main objectives of implementing the packet dropping scheme in a switch are twofold. First, by removing packets that are residing in a queue for more than a specified time, the average packet latency can be greatly reduced. In addition, since we adopt multipath routing, deadlocks can occur in a network. In this case, the packet dropping scheme can effectively break the cyclic dependencies that cause deadlocks.

The IBA specification [5] 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 refers to the maximum amount of time a packet lives in a switch and HLL refers to the maximum amount of time a packet stays at the head of a queue. SLL is defined as $4.096\ \mu\text{sec} \times 2^{LV}$, where $0 \leq \text{LV} \leq 19$. If $\text{LV} > 19$, then SLL is to be interpreted as infinite. HLL is defined as $4.096\ \mu\text{sec} \times 2^{HV}$, where $0 \leq \text{HV} \leq 19$. If $\text{HV} > 19$, 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 [21], deadlocks in a network rarely occur if the network 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 [21]. 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.

Here, 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*. If a packet arrives at an empty VL, only the arriving time of the packet header 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 in this paper. Dropped packets can be detected at the transport layer and can be recovered if necessary.

3.6 Multicasting Support in IBA

Multicasting is a desired feature in the current IBA specification. Compared to unicast-based schemes, multicasting saves link bandwidth because each cascaded switch replicates the packet such that no more than one copy of the packet appears at each end node. 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 [9], [10]. Thus, the multicast forwarding table is obtained from the source-based distribution tree algorithm assuming static membership groups [9]. Each switch should contain the routing information and manage the multicast membership group. Hence, we only focus on the replication mechanisms and the bandwidth allocation policies in this paper.

3.6.1 Replication Mechanisms

There are two ways to replicate packets in a switch: synchronous and asynchronous. 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 [22]. This scheme requires extra buffers large enough to store the largest packet in the switch [11], and complex buffer control mechanisms. Detailed implementations of both the schemes are given below.

Synchronous Replication. Synchronous replication begins when a multicast packet arrives at Stage 4 in Fig. 1. In our scheme, a multicast packet has higher priority than unicast packets for reserving the crossbar output ports. To prevent deadlocks among multicast packets, a multicast packet can take crossbar output ports already reserved by other multicast packets unless packet transfer has started. The priorities amongst competing multicast packets can be determined using their arrival time (FCFS) or through a RR (Round Robin) polling. Although we implemented both FCFS and RR priority schemes, we only show the results with FCFS in Section 5.2 since FCFS provides a slight better performance. In FCFS, a multicast packet with the lowest timestamp can reserve an output port when the output port is being used by another packet for transmission. Once the transmission is completed, the lowest-timestamped multicast packet can reserve it. Even after reserving all the output ports, if any of outports is not empty, synchronous replication cannot start.

Asynchronous Replication. In this scheme, we need to keep a multicast packet until it is forwarded to all the destinations. This requires buffers large enough to store multicast packets in a switch. We can implement asynchronous replication in two different ways. First, instead of using extra buffers, we can add a counter for each input VL. The counter indicates the remaining number of destinations for which the multicast packet should be forwarded. However, if the input VL is a FIFO queue, we need one-packet-size extra buffer to hold the packet, since the first copy will dequeue the packet from the input VL. Note that since only one multicast transfer can progress at a time, one packet size buffer is sufficient. The other asynchronous replication scheme can be implemented using central buffers as proposed in [11] to provide a separate FIFO queue for multicast packets for each output port. In this scheme, the multicast packets in the central buffers have higher priority than the unicast packets in input VLs. We have implemented both these asynchronous schemes.

TABLE 1
IBA Simulation Testbed Parameters

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

3.6.2 Bandwidth Allocation Policy

Efficient bandwidth allocation to multicast and unicast traffic is another research issue that needs investigation to improve performance. One approach is to allocate more bandwidth to multicast flows in order to give higher priority to multicast traffic [23]. [23] considers three different bandwidth allocation schemes: 1) allocate the same bandwidth to unicast and multicast flows, 2) allocate multicast bandwidth linearly proportional to the number of destinations, and 3) 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

Our IBA simulation 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 workload includes packets from real-time VBR traffic, best-effort traffic, and control traffic. The VBR traffic is generated from seven MPEG-2 traces [24], 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 Maximum Transfer Unit (MTU) and the header.

The best-effort traffic 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

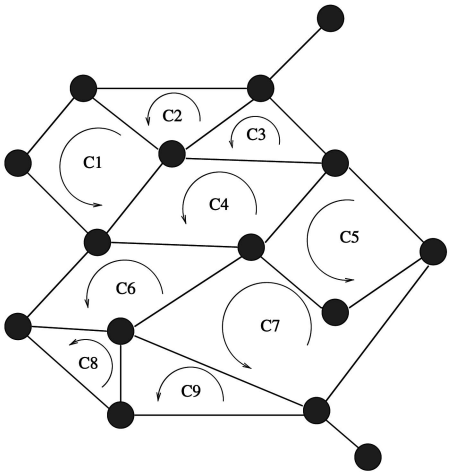


Fig. 6. A 15-node irregular network.

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. Whenever a frame misses its deadline, we measure the deadline missing time and then calculate the average DMT.

5 Performance Results

We have conducted extensive evaluation of the proposed designs for different networks and workload conditions. First, we analyze all the routing schemes in a 15-node and a 30-node irregular networks. Next, we discuss the multicasting results in a 15-node irregular network.

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 in Fig. 6 that has nine cycles, we use nine path bits. The number of entries in the forwarding tables is either $30\ (= 15 \times 2^1)$ or $60\ (= 15 \times 2^2)$.

Fig. 7 shows the results from seven different routing schemes. These are:

1. Up/Down,
2. SPF,
3. SPF with packet dropping (SPF+Drop),
4. multipath routing using LFU as the path selection criteria (Multipath+LFU),
5. multipath routing using LFU and packet dropping (Multipath+LFU+Drop),
6. multipath routing using Max-Credit as the path selection criteria (Multipath+Credit), and
7. 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

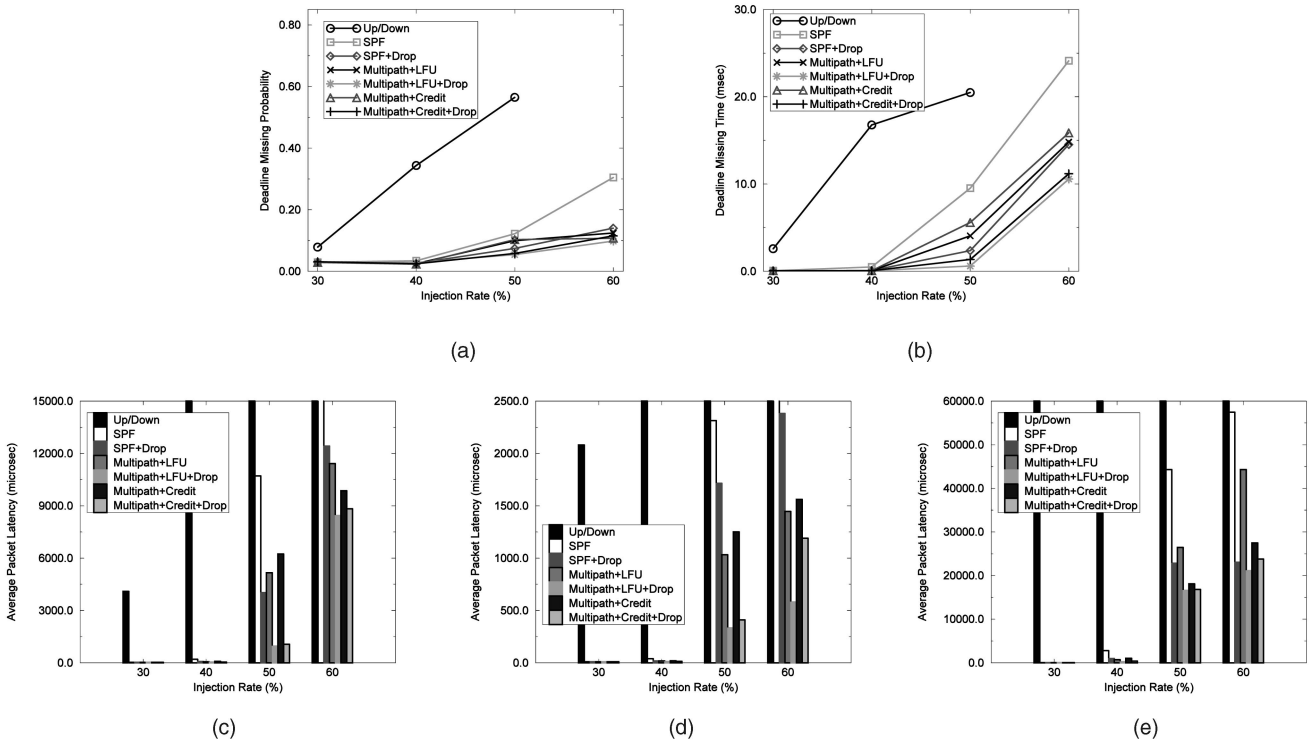


Fig. 7. Comparison of routing algorithms in a 15-node irregular network (Real-time: Best-effort = 70:30). (a) Deadline missing probability. (b) Deadline missing time. (c) Real-time traffic. (d) Control traffic. (e) Best-effort traffic.

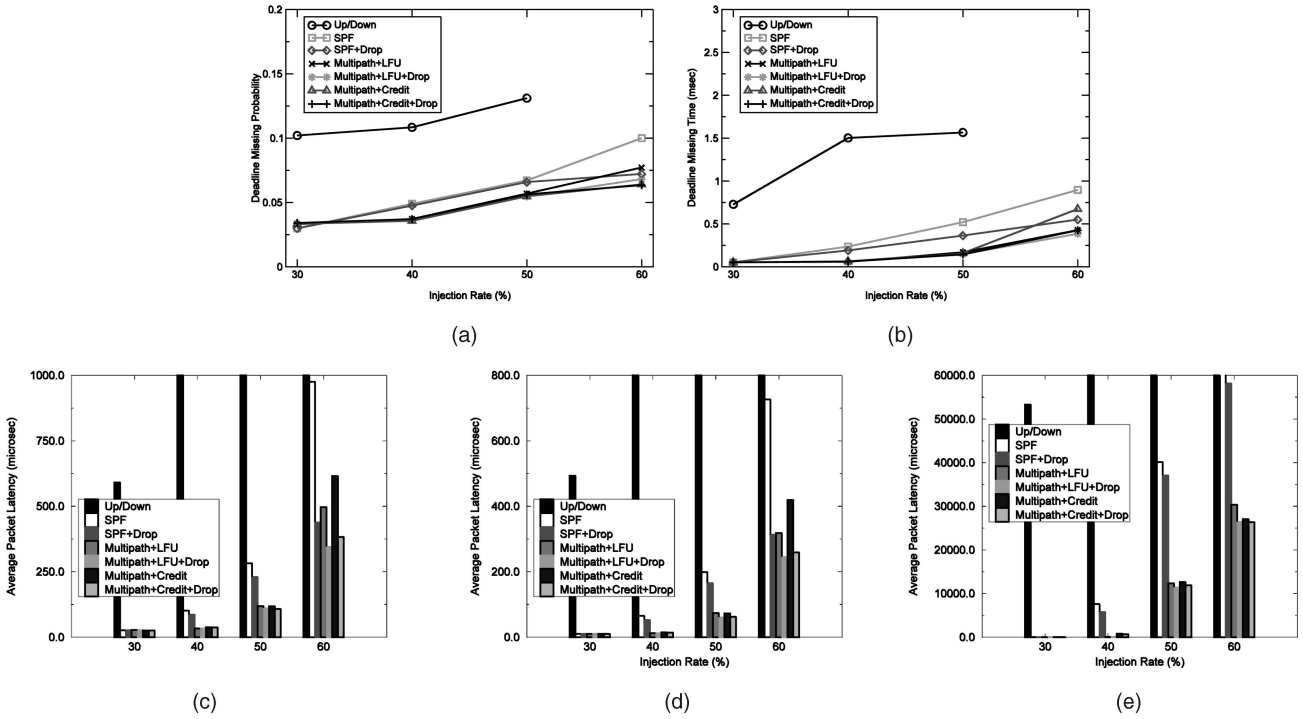


Fig. 8. Comparison of routing algorithms in a 15-node irregular network (Real-time: Best-effort = 30:70). (a) Deadline missing probability. (b) Deadline missing time. (c) Real-time traffic. (d) Control traffic. (e) Best-effort traffic.

adaptively using LFU or Max-credit. The control traffic always uses the shortest path via VL_{15}.

Figs. 7a and 7b show the DMP and the DMT of real-time traffic, while Figs. 7c, 7d, and 7e 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 affected 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 percent). However, as the load increases, these techniques become very effective in reducing those metrics (50 percent $\sim$ 65 percent for the DMP and 30 percent $\sim$ 55 percent for the DMT compared to SPF at the injection rate of 60 percent). The results indicate that multipathing and packet dropping help in jitter-free delivery of media streams at relatively high load.

We can observe the genuine effects of multipath routing by comparing the results of SPF, (SPF+LFU), and (SPF+Credit). The multipath routing (LFU and Max-Credit) schemes reduce the DMP about 50 percent of that with SPF routing at 60 percent input load, and the DMT about 30 percent of that with SPF routing at 50 percent or higher input load. The average packet latency with multipath routing is about half of that with SPF routing with more than 50 percent input 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 percent $\sim$ 55 percent for control traffic, by 40 percent $\sim$ 65 percent for real-time, and by 30 percent $\sim$ 55 percent for best-effort traffic, respectively, compared to that of SPF at an injection rate of 50 percent or higher. With packet dropping, these reductions increase up to 80 percent for control, 90 percent for real-time, and 60 percent for best-effort traffic, respectively.

Fig. 8 shows the results from seven different routing schemes for real-time to best-effort ratio of 30:70. In this experiment, the dominant number of packets are best-effort traffic, which has the lowest priority. Compared to Fig. 7, both the DMP and DMT values of real-time traffic are much smaller, but still show the same tendency. The routing schemes with multipathing and packet dropping provide better performance in terms of the DMP and the DMT at a higher injection rate. Also, (Multipath+LFU) helps to reduce the average packet latency of real-time and control traffic, while (Multipath+Credit) is more effective for that of best-effort traffic.

The effect of the routing schemes in a 30-node irregular network is shown in Fig. 9. Here, we plot the average packet latency of real-time, control, and best-effort traffic. Since the LFU scheme provides better performance for real-time and control traffic, we use the LFU path selection criteria for the 30-node experiments. The performance difference between SPF and (Multipath+LFU) is distinct for all three kinds of traffic in this figure (70 percent for real-time, 60 percent for control, and 90 percent for best-effort traffic at an injection rate of 28 percent). The results of best-effort traffic in Fig. 9c show that multipath routing

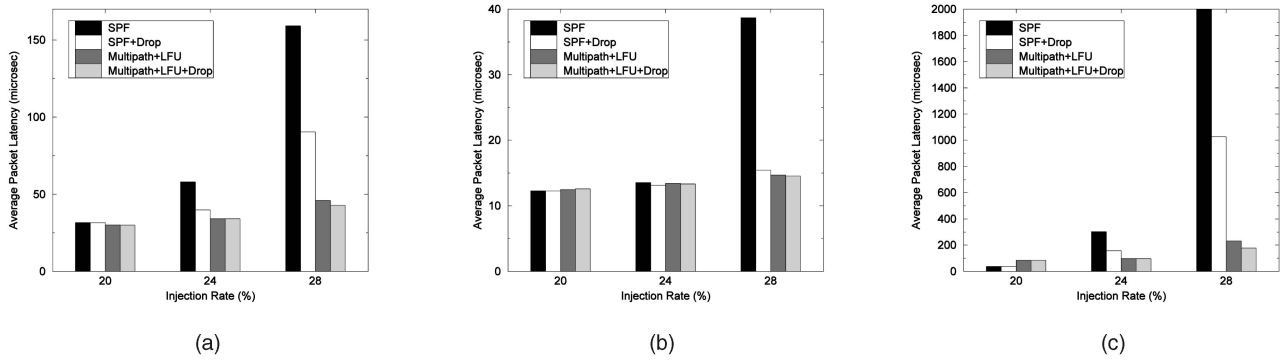


Fig. 9. Comparison of routing algorithms in a 30-node irregular network (Real-time: Best-effort = 70:30). (a) Real-time. (b) Control. (c) Best-effort.

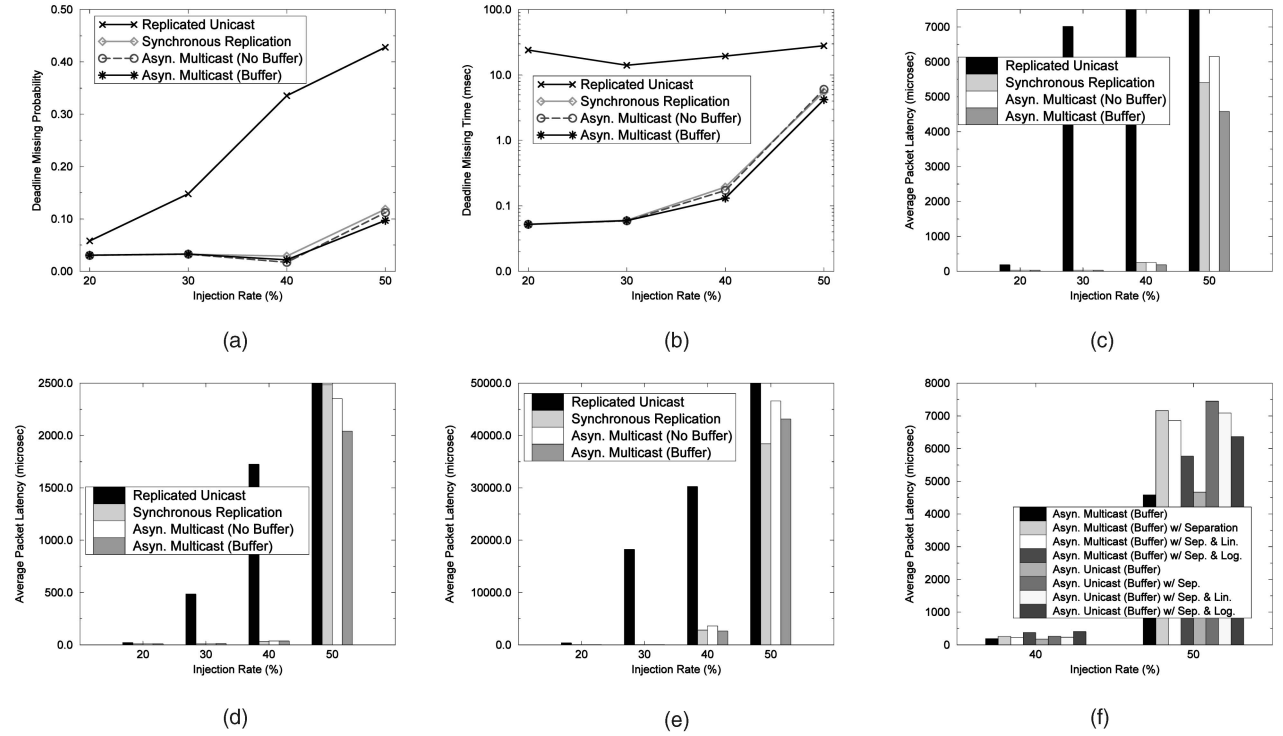


Fig. 10. Evaluation of multicasting in a 15-node irregular network (Real-time: Best-effort = 70:30). (a) Deadline missing probability. (b) Deadline missing time. (c) Multicast traffic. (d) Control traffic. (e) Best-effort traffic. (f) 6:5 Multi versus uni VL allot.

increases the latency of best-effort traffic at a low injection rate (20 percent input load), while it still reduces that of real-time and control traffic. This phenomenon also has been observed in the results of the 15-node network. This is primarily because the best-effort traffic can use a non-minimal path, which hurts performance at a low load. The impact of packet dropping was not as significant as that in the 15-node case. This is attributed to the network topology and relatively low load, which keep the probability of deadlock and congestion low.

5.2 Multicast Results

Fig. 10 shows the simulation results of four multicasting schemes in a 15-node irregular network. These are:

1. Replicated Unicast without any hardware support for multicasting,
2. Synchronous Replication,
3. Asynchronous Replication without Central Buffer, and
4. Asynchronous Replication with Central Buffer.

In this experiment, 20 percent 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 three groups each with two or four multicast members.)

Figs. 10a and 10b 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 in Figs. 10c, 10d, and 10e. 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.[5] By reducing the real-time traffic load, the replication schemes also help control and best-effort traffic latencies as shown in Figs. 10d and 10e.

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 Fig. 10f:

1. the same bandwidth to unicast and multicast without VL separation,
2. the same bandwidth to unicast and multicast with separate VLs,
3. bandwidth linearly proportional to the number of destinations, and
4. bandwidth proportional to the logarithm of the number of destinations.

For 3 and 4, which allocate more bandwidth to multicast traffic, we also allocate separate VLs for multicast traffic. The first four bars are the average packet latencies of multicast traffic with different bandwidth allocation policies, while the other four are those for unicast traffic.

With a low load, the three bandwidth allocation schemes (2, 3, and 4) 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 4 provides the best performance among the three schemes. For the replication scheme 1, 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 results in better performance. The results bring in an interesting observation that giving separate VLs to multicast traffic may not be a good idea in an IBA-style switch.

With these multicast results, we can conclude that hardware support for multicasting is necessary to assure QoS delivery of real-time traffic. Synchronous replication can be a feasible solution, since it does not require large buffer, and it provides comparable performance with the asynchronous scheme. Under the IBA environment, we may not need special treatment for multicast traffic in the sense of bandwidth allocation.

6 Concluding Remarks

We have presented in this paper four 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.

5. Although we have simulated both FCFS and RR scheduling for multicast packets in arbitrating for the output ports, only the results of FCFS, which provides slight better performance, are shown in Fig. 10.

The important conclusions of this work are the following: First, the simulation results show that the SPF routing algorithm is 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 does not require large buffers.

The study presented in this paper can be extended in the following directions. First, the proposed designs need in-depth evaluation with several real workloads. Next, the impact of multipathing on Automatic Path Migration (APM) and overall fault tolerance of SANs will be studied. Also, we plan to implement the SLL-based packet dropping scheme in our simulation testbed by timestamping each packet.

Acknowledgments

The research was supported in part by US National Science Foundation grants CCR-0208734, CNS-0202007, CCF-0429631, CNS-0509251, CCF-0541384, and CCF-0541360. A preliminary version of this paper [1] was presented at the Ninth International Symposium on High-Performance Computer Architecture (HPCA-9), February 2003.

References

[1] E.J. Kim, K.H. Yum, C.R. Das, M. Yousif, and J. Duato, "Performance Enhancement Techniques for InfiniBand Architecture," *Proc. IEEE Int'l Symp. High-Performance Computer Architecture (HPCA),* pp. 253-262, Feb. 2003.

[2] The Semiconductor Research Assoc., "Research Needs for Extending CMOS to Its Ultimate Limit," Nov. 2002, http://www.src.org/fr/nis_cmos_needs.pdf.

[3] M.T. Niemier and P.M. Kogge, "Exploring and Exploiting Wire-Level Pipelining in Emerging Technologies," *Proc. Int'l Symp. Computer Architecture,* pp. 166-177, June 2001.

[4] S.C. Goldstein and M. Budiu, "NanoFabrics: Spatial Computing Using Molecular Electronics," *Proc. Int'l Symp. Computer Architecture,* pp. 178-189, June 2001.

[5] InfiniBand Trade Assoc., "InfiniBand Architecture Specification, Volume 1, Release 1.2," Oct. 2004, http://www.infinibandta.org.

[6] J. Pelissier, "Providing Quality of Service over InfiniBand Architecture Fabric," *Proc. Symp. High Performance Interconnects (Hot Interconnects 8),* Aug. 2000.

[7] M.D. Schroeder, A. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, and C.P. Thacker, "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," *IEEE J. Selected Areas in Comm.,* vol. 9, no. 8, pp. 1318-1335, 1991.

[8] A.S. Vaidya, A. Sivasubramaniam, and C.R. Das, "LAPSES: A Recipe for High Performance Adaptive Router Design," *Proc. IEEE Int'l Symp. High-Performance Computer Architecture,* pp. 236-243, Jan. 1999.

[9] J.E. Klinker, "Multicast Tree Construction in Directed Networks," *Proc. IEEE Military Comm. Conf.,* pp. 496-500, Oct. 1996.

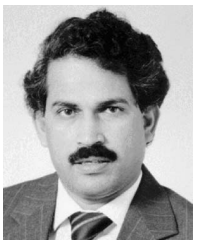
[10] T. Billhartz, J.B. Cain, E. Farrey-Goudreau, D. Fieg, and S.G. Batsell, "Performance and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols," *IEEE J. Selected Areas in Comm.,* vol. 15, no. 3, pp. 304-315, Apr. 1997.

[11] C.B. Stunkel, D.G. Shea, B. Abali, M.G. Atkins, C.A. Bender, D.G. Grice, P. Hochschild, D.J. Joseph, B.J. Nathanson, R.A. Swetz, R.F. Stucke, M. Tsao, and P.R. Varker, "The SP2 High-Performance Switch," *IBM Systems J.,* vol. 34, no. 2, pp. 185-204, 1995.

[12] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, May 1992.
[13] S.S. Mukherjee and M.D. Hill, "A Survey of User-Level Network Interfaces for Systems Area Networks," Technical Report 1340, Computer Science Dept., Univ. of Wisconsin-Madison, Feb. 1997.
[14] K.H. Yum, E.J. Kim, and C.R. Das, "QoS Provisioning in Clusters: An Investigation of Router and NIC Design," *Proc. Int'l Symp. Computer Architecture*, pp. 120-129, June 2001.
[15] J. Moy, "OSPF Version 2," *The Internet Soc.*, 1998, FC 2328.
[16] J. Wu and L. Sheng, "Deadlock-Free Routing in Irregular Networks Using Prefix Routing," Technical Report 99-19, *DIMACS*, Apr. 1999.
[17] F. Silla and J. Duato, "Efficient Adaptive Routing in Networks of Workstations with Irregular Topology," *Proc. Workshop Comm. and Architectural Support for Network-Based Parallel Computing (CANPC '97)*, Feb. 1997.
[18] P. López, J. Flich, and J. Duato, "Deadlock-Free Routing in InfiniBand through Destination Renaming," *Proc. Int'l Conf. Parallel Processing*, pp. 427-434, Sept. 2001.
[19] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. MIT Press, 1999.
[20] E.M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms*. Prentice-Hall, 1977.
[21] T.M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks," *Proc. Int'l Symp. Computer Architecture*, pp. 38-49, June 1997.
[22] C.B. Stunkel, R. Sivaram, and D.K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and Their Impact," *Proc. Int'l Symp. Computer Architecture*, pp. 50-61, June 1997.
[23] A. Legout, J. Nonnenmacher, and E.W. Biersack, "Bandwidth-Allocation Policies for Unicast and Multicast Flows," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, pp. 464-478, Aug. 2001.
[24] M.B. Caminero, F.J. Quiles, J. Duato, D.S. Love, and S. Yalamanchili, "Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic," *Proc. Third Int'l Workshop Comm., Architecture, and Applications on Network-Based Parallel Computing (CANPC '99)*, pp. 62-76, Jan. 1999.

Eun Jung Kim received the BS degree in computer science from the Korea Advanced Institute of Science and Technology, Korea, in 1989, the MS degree in computer science from the Pohang University of Science and Technology, Korea, in 1994, and the PhD degree in computer science and engineering from Pennsylvania State University in 2003. From 1994 to 1997, she worked as a member of technical staff with the Korea Telecom Research and Development Group. Dr. Kim is currently an assistant professor in the Department of Computer Science in Texas A&M University. Her research interests include computer architecture, parallel/distributed systems, computer networks, cluster computing, QoS support in cluster networks and internet, performance evaluation, and fault-tolerant computing. She is a member of the IEEE and the ACM.

Ki Hwan Yum received the BS degree in mathematics from Seoul National University, Korea, in 1989, the MS degree in computer science and engineering from Pohang University of Science and Technology, Korea, in 1994, and the PhD degree in computer science and engineering from Pennsylvania State University in 2002. From 1994 to 1997, he was a member of the technical staff with the Korea Telecom Research and Development Group. Dr. Yum is currently an assistant professor in the Department of Computer Science at the University of Texas at San Antonio. His research interests include computer architecture, parallel/distributed systems, cluster computing, and performance evaluation. He is a member of the IEEE and the ACM.



Chita R. Das received the MSc degree in electrical engineering from the Regional Engineering College, Rourkela, India, in 1981, and the PhD degree in computer science from the Center for Advanced Computer Studies, University of Louisiana, Lafayette, in 1986. Since 1986, he has been with Pennsylvania State University, where he is currently a professor in the Department of Computer Science and Engineering. His main areas of interest are parallel and distributed computer architectures, cluster computing, mobile computing, Internet QoS, multimedia systems, performance evaluation, and fault-tolerant computing. He has served on the editorial boards of the *IEEE Transactions on Computers* and *IEEE Transactions on Parallel and Distributed Systems*. Dr. Das is a fellow of the IEEE and a member of the ACM.

Mazin Yousif received the Masters and PhD degrees from the Pennsylvania State University in 1987 and 1992, respectively. He is a principal engineer in the Corporate Technology Group of Intel Corporation in Hillsboro, Oregon. He currently leads a team that focuses on platform provisioning and virtualization to enable platform autonomics. Prior to that, he was in the Enterprise Product Group focusing on InfiniBand, datacenter I/O interconnects, and datacenter applications workload characterization. During his involvement with the InfiniBand Architecture, he was chairing the InfiniBand Trade Association (IBTA) Management Working Group. From 1993 to 1995, he was an assistant professor in the Computer Science Department at Louisiana Tech University. Dr. Mazin worked for IBM's xSeries Server Division in Research Triangle Park (RTP), North Carolina, from 1995-2000. During his tenure in industry, he served as an adjunct professor at Duke, NCSU, and currently OGI. His research interests include computer architecture, clustered architectures, workload characterization, networking, and performance evaluation. He has published more than 50 articles in his areas of research. Dr. Mazin has chaired several conferences and workshops, was in the program committees of many others, and led several panels. He is in the advisory board of the *Journal of Pervasive Computing and Communications* (*JPCC*), editor of *Cluster Computing*, the *Journal of Networks*, *Software Tools and Applications*, and the *Journal of Autonomic and Trusted Computing* (*JoATC*). He is a senior member of the IEEE.

José Duato is a professor in the Department of Computer Engineering (DISCA) at UPV, Spain. His research interests include interconnection networks and multiprocessor architectures. He published more than 350 papers. His research results have been used in the design of the Alpha 21364 microprocessor, and the Cray T3E and IBM BlueGene/L supercomputers. Dr. Duato is the first author of the book *Interconnection Networks: An Engineering Approach*. He served as associate editor of *IEEE Transactions on Parallel and Distributed Systems* and *IEEE Transactions on Computers*. He was general cochair of ICPP 2001, program chair of HPCA-10, and program cochair of ICPP 2005. Also, he served as cochair, steering committee member, vice-chair, or program committee member in more than 55 conferences, including HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, Europar, and HiPC. He is a member of the IEEE and the IEEE Computer Society.