MediaWorm: A QoS Capable Router Architecture for Clusters

Ki Hwan Yum, Eun Jung Kim, Chita R. Das, and Aniruddha S. Vaidya

Abstract

With the increasing use of clusters in real-time applications, it has become essential to design high performance networks with Quality-of-Service (QoS) guarantees. In this paper, we explore the feasibility of providing QoS in wormhole switched routers, which are widely used in designing scalable, high performance cluster interconnects. In particular, we are interested in supporting multimedia video streams with CBR and VBR traffic, in addition to the conventional best-effort traffic. The proposed MediaWorm router uses a rate-based bandwidth allocation mechanism, called *Fine-Grained VirtualClock* (FGVC), to schedule network resources for different traffic classes.

Our simulation results on an 8-port router indicate that it is possible to provide jitter-free delivery to VBR/CBR traffic up to an input load of 70-80% of link bandwidth, and the presence of best-effort traffic has no adverse effect on real-time traffic. Although the MediaWorm router shows a slightly lower performance than a pipelined circuit switched (PCS) router, commercial success of wormhole switching, coupled with simpler and cheaper design, makes it an attractive alternative. Simulation of a (2×2) fat-mesh using this router shows performance comparable to that of a single switch, and suggests that clusters designed with appropriate bandwidth balance between links can provide required performance for different types of traffic.

Keywords

Cluster Network, Quality-of-Service, Rate-based Scheduling, Router Architecture, VirtualClock, Wormhole Router

I. Introduction

CLUSTER systems are becoming a predominant and cost-effective style for designing high performance computers. Such systems are being used in as diverse applications as scientific computing, web servers, multimedia servers, and collaborative environments. These applications place different demands on the underlying cluster interconnect, making it imperative to reevaluate and possibly redesign the existing communication architecture. Multiprocessor network research [1] has primarily focussed on designing scalable, high performance networks (low latency and high bandwidth) to accommodate traditional *best-effort* (BE) traffic. Over the years, network design philosophy has converged towards direct network topology, wormhole switching, and virtual channel (VC) flow control [2] to meet these design goals. These research ideas have manifested in many commercial switch/router designs [3], [4], [5], [6], [7], [8] and have migrated to, and been successfully assimilated in, cluster interconnects [9],

This research was supported in part by NSF grants MIPS-9634197, CCR-9900701, CCR-0098149, CCR-0208734, and EIA-0202007. A preliminary version of this paper was presented at the Sixth International Symposium on High-Performance Computer Architecture (HPCA-6), January 2000.

K. H. Yum, E. J. Kim, and C. R. Das are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail: {yum,ejkim,das}@cse.psu.edu.

A. S. Vaidya is with Intel Corporation, Mailstop SC12-608, 3600 Juliette Lane, Santa Clara, CA 95054. Email: aniruddha.vaidya@intel.com. Vaidya's contribution to this work was conducted when he was with the Department of Computer Science, The Pennsylvania State University.

[10]. However, many of the new applications require that, in addition to high bandwidth, the network should provide predictable performance or Quality-of-Service (QoS) guarantees. Hence, the new challenge is to design scalable networks (routers) that can provide high performance and QoS guarantees for integrated or mixed traffic.

This challenge requires an understanding of the traffic that the emerging applications are likely to generate on the underlying interconnect, so that the network can be designed accordingly. The ATM forum has defined three traffic classes called constant bit rate (CBR), variable bit rate (VBR) and available bit rate (ABR) [11]. CBR is generated during uncompressed video/audio transmission while VBR is generated due to compression. These two traffic classes need QoS guarantees. Finally, ABR refers to best-effort traffic and subsumes all other applications that do not have real-time requirements. A cluster interconnect, the router in particular, should therefore support CBR, VBR and ABR traffic effectively.

Two switch or router design paradigms have been used to build clusters [1]. One is based on the cut-through switching mechanisms (wormhole [12] and virtual cut-through (VCT) [13]), originally proposed for multiprocessor switches, and the other is based on packet switching (Store and Forward). Current multiprocessor routers, primarily based on the cut-through paradigm, are suitable for handling ABR traffic. However, they may not be able to support the stringent QoS requirements efficiently without possibly modifying the router architecture. On the other hand, packet switching mechanisms like ATM can provide QoS guarantees, but they are not suitable for best-effort traffic primarily due to high message latency compared to cut-through switching [14], [15]. Therefore, none of the existing network architectures are optimized to handle both best-effort and real-time traffic in clusters.

In view of this, a few researchers have explored the possibility of providing QoS support in router architectures [14], [16], [17], [18], [15]. Most of these designs have used a hybrid approach using two different types of switching mechanisms within the same router — one for best-effort and the other for real-time traffic. They have refrained from using wormhole switching because of potential unbounded delay for real-time traffic.

On the other hand, in the commercial world, it appears that wormhole switching has become a *de facto* standard for clusters/multiprocessors. Therefore, it would really be advantageous if we could leverage off of the large amount of effort that has gone into the design and development of these wormhole routers and adapt them to support all traffic classes with minimal design changes. Some recent modifications to wormhole routers have been considered for handling traffic priority [19], [20], [21], [22]. However, to our knowledge, there have been no previous forays into investigating the viability of supporting multimedia traffic with wormhole switching.

The main motivation of this paper is to investigate the feasibility of supporting mixed traffic in wormhole routers with minimal modifications to the existing router architecture. We are specifically interested in transferring multimedia video streams in addition to usual best-effort traffic. This requires providing some mechanism within the router that recognizes the bandwidth requirements of VBR and CBR traffic, and accommodates these requests. One can borrow the concepts from real-time/Internet research to provide hard or soft guarantees. Instead of conservatively reserving resources within the router to achieve these goals with hard guarantees, we are interested in more optimistic solutions that provide soft guarantees to media streams. In particular, the paper attempts to address the following questions:

- Can we provide a mechanism for soft-guarantees to VBR and CBR traffic in a wormhole router?
- How does the existence of best-effort traffic affect real-time traffic, and vice versa?
- How do we configure the wormhole router for best rewards? Should we support more connections within each VC (with fewer VCs per physical channel (PC))? Or should we support more number of VCs with fewer connections per VC? While a larger number of VCs is intuitively expected to perform better, it may not be possible to implement a full-crossbar.
- What is the impact of message size on real-time traffic ?
- How does such a wormhole routing implementation compare with a connection-oriented pipelined circuit switched (PCS) router [23] in terms of number of jitter-free connections, hardware complexity, etc. ?
- Finally, how does a cluster network using such wormhole-switched routers perform with mixed traffic?

We have used a simulation testbed to answer the above questions. We propose a new wormhole router architecture, called *MediaWorm*, using a conventional pipelined wormhole router design for meeting the bandwidth requirements. Two modifications are proposed to a standard wormhole router. First, the VCs are assumed to be partitioned into two classes during the configuration time — one for transferring best-effort traffic and the other for real-time traffic. Dynamic VC assignment is considered in a later version. Second, to satisfy the bandwidth requirements of different applications, the round-robin (RR) or First-In-First-Out (FIFO) scheduler used in a traditional router is replaced by a rate-based scheduling mechanism. We modify the connection-oriented VirtualClock scheduling algorithm [24] to a *Fine-Grained VirtualClock* (FGVC) to provide soft guarantees to media streams without explicit connection setup. Unlike the original VirtualClock algorithm, the FGVC provides bandwidth reservation at a message-level.

An 8-port MediaWorm design has been evaluated using a spectrum of real-time and best-effort traffic mixes, with varying workload and hardware parameters. We have also evaluated a (2×2) fat-mesh topology designed with these routers. The results indicate that the FGVC algorithm in the MediaWorm does indeed improve the QoS delivered to real-time traffic compared to the RR/FIFO scheduling algorithm. In the case of a single switch, the MediaWorm can provide jitter-free delivery up to an input load of 70–80% of the physical channel bandwidth. For most realistic operating conditions, the MediaWorm router can deliver as good (jitter-free) performance as the PCS-based router [23] for real-time traffic without dropping any connection establishment requests (unlike in PCS). This goal can be obtained at a significantly lower cost (requires much lower number of VCs). Increasing the number of VCs per PC improves the QoS of the system, though the crossbar multiplexing overheads can become significant. By allocating the VCs separately to best-effort and real-time traffic, we find that the former does not really interfere with the latter. However, the average latency of the best-effort traffic increases as expected.

The rest of this paper is organized as follows. The next section summarizes the related work. Section III presents the architectural details of the MediaWorm router, and the FGVC scheduling algorithm. Section IV gives details on the simulation platform and the workload used in the evaluation. The performance results from the simulations are given in Section V. Finally, Section VI summarizes the results of this study and identifies directions for future research.

II. Related Work

With the building block of a multiprocessor interconnect being its router or switch fabric, a considerable amount of research effort has gone into the design of efficient routers. Routers from university projects like reliable router [25] and Chaos router [26], and commercial routers such as SGI SPIDER, Cray T3D/E, Tandem Servernet-II, IBM SP2 switch, and Myrinet [3], [4], [5], [10], [8], [9] use wormhole switching, while the HAL Mercury [27] and Sun S-Connect [28] use virtual cut-through (VCT). Most of them support VCs, and at least the Cray T3E, ServerNet-II and S-Connect have adaptive routing capability. Metro [29] and Ariadne [30] employ the pipelined circuit switching (PCS) technique, while the latter is fully adaptive and tolerates link and switch faults. A hybrid switch including both wormhole and VCT was designed in [31]. All these routers are primarily designed to minimize average message latency and improve the network throughput. The SGI SPIDER, Sun S-Connect, and Mercury support message priority. But, none of these routers can guarantee QoS as required for real-time applications like VOD services. ServerNet is the only router that provides a link arbitration policy (called ALU-biasing) for implementing bandwidth and delay control, but it still does not provide any capabilities to support multimedia traffic.

Recently, a few researchers have explored the possibility of providing QoS support in multiprocessor/cluster interconnects. The need for such services, existing methods to support QoS specifically in WAN/long-haul networks, and their limitations are summarized in [17], [32]. Kim and Chien [11] propose a scheduling discipline, called rotating and combined queue (RCQ), to handle integrated traffic in a packet switched network. The Switcherland router [15], designed for multimedia applications on a network of workstations, uses a packet switched mechanism similar to ATM, while avoiding some of the overheads associated with the WAN features of ATM. The router architecture proposed in [18] uses a hybrid approach, wherein wormhole switching is used for best-effort traffic and packet switching is used for time-constrained traffic.

A multimedia router architecture (MMR), proposed in [14], [16], also adheres to a hybrid approach by using pipelined circuit switching (PCS) for multimedia traffic and virtual-cut-through (VCT) for best-effort traffic. The authors have designed a (4×4) router to support both PCS and VCT schemes, and have used MPEG video traces in their evaluations. While a connection-oriented mechanism such as PCS is suitable for multimedia traffic, it needs one VC per connection. For a link bandwidth of 1.24 Gbps, and with each multimedia stream requiring 4 Mbps, the design would require 256 VCs to fully utilize a physical channel. It is not clear whether it is practical to have such a large number of VCs per physical channel and what will be the cost of the corresponding multiplexer and demultiplexer implementations. In addition, the architecture of the router is fairly complex since it has to have facilities for both PCS and VCT transmission. Nevertheless, this is perhaps the most detailed study where router performance has been analyzed with multimedia video streams, best-effort, and control traffic. A preemptive PCS network to support real-time traffic is also proposed in [33].

To our knowledge, there are only a handful of research efforts that have examined the possibility of using wormhole switched networks for real-time traffic [19], [22], [20], [21], [34]. In many of these studies [19], [22], [34], the focus is on providing some mechanisms within the router to implement priority (for real-time traffic) and preemption (when the resources are allocated to a less critical message). However, these mechanisms are not sufficient (and may not even be necessary) for providing soft guarantee for multimedia traffic. Three different techniques for providing QoS in wormhole switched routers are explored in [20] using a simulated multistage network. These include using a separate subnet for real-time traffic, supporting a synchronous virtual network on the underlying asynchronous network, and employing VCs. The first approach may not be cost-effective. The second solution of using a synchronous network (either inherently synchronous or simulated on top of an asynchronous network as is done in [21] on Myrinet), is not a scalable option. The third option of using VCs has not been investigated in depth in [20], where it has been cursorily examined in the context of indirect networks. The software oriented synchronization mechanism in the Myrinet switch proposed in [21] also lacks scalability.

It is still not clear as to what is the best switching mechanism that can support all traffic classes. Should we resort to hybrid routers that differentially service the traffic classes (and pay a high cost) like some of the above studies have done? Or, can we use a single switching mechanism (wormhole switching in particular, since it has been proven to work well for best-effort traffic and we can leverage off of the immense body of knowledge/infrastructure available for this mechanism) with little or no modifications? Instead of discarding the wormhole switching mechanism as an option for multiple traffic classes, this paper explores how a large number of multimedia connections can be supported in the presence of best-effort traffic.

III. MediaWorm Router Architecture

In this section, we focus our discussion on the architectural details of the MediaWorm router. The discussion starts with the architecture of a basic pipelined wormhole router and then outlines the modifications proposed for QoS guarantees. The section concludes with a brief description of PCS routers.

A. A Basic Wormhole Router

In wormhole switched networks, messages are segmented into flow-control units called flits. As a message enters a router, its header flit is used to determine the permitted output port that would route the message to its destination. The message then flows through the router crossbar to the appropriate output port. If resources (such as output buffer space or output ports) are busy, the message blocks until resources become available. Flits of a message flow through the network in a pipelined manner. Performance of wormhole routers can be enhanced through the use of virtual channels (VCs) [2]. VCs are also used for supporting deadlock freedom and providing adaptive routing capabilities. Wormhole routers can be pipelined so that although a flit experiences multi-cycle latency to get from its input port to an output port, the router cycle time can be kept very small (typically a few nanoseconds) depending on the slowest stage of the pipeline.

B. Pipelined Design

We use a pipelined router model, called PROUD [35], [36], to design the MediaWorm router. The pipelined model with five stages, as depicted in Figure 1, represents the recent trend in router designs [37].

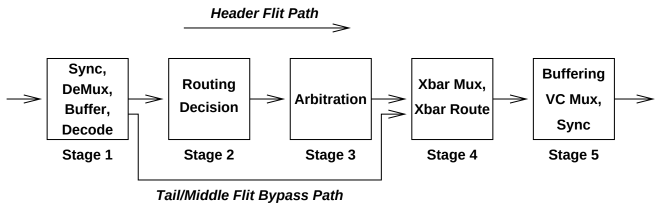


Fig. 1. A basic pipelined wormhole router

Stage 1 of the pipeline represents the functional units which synchronize the incoming flits, demultiplex a flit so that it can go to the appropriate VC buffer to be subsequently decoded. If the flit is a header flit, routing decision and arbitration for the correct crossbar output are performed in the next two stages (stage 2 and stage 3). On the other hand, middle flits and the tail flit of a message bypass stages 2 and 3 to move directly to stage 4. Flits get routed to the correct crossbar output in stage 4. The bandwidth of the crossbar may be (optionally) multiplexed amongst multiple VCs. This is discussed in detail later in this section. Finally, the last stage of the router performs buffering for flits flowing out of the crossbar, multiplexes the physical channel bandwidth amongst multiple VCs and performs hand-shaking and synchronization with input ports of other routers or the network interface for the subsequent transfer of flits.

A pipelined router can thus be modeled as multiple parallel PROUD pipes. In an n-port router, if each PC has m VCs, a router could then be modeled as $(n \times m)$ parallel pipes. Resource contention amongst these pipes could occur for the crossbar output ports (which is managed by the arbitration unit) as well as for the physical channel bandwidth of the output link (which is managed by the virtual channel multiplexer).

C. Crossbar Design Options

We consider two different crossbar design options — a full crossbar and a multiplexed crossbar [2]. A full crossbar has number of input and output ports equal to the total number of VCs supported — $(n \times m)$ for an n-port router with m VCs per PC. On the other hand, a multiplexed crossbar has number of input/output ports equal to the total number of PCs (n). A full crossbar may improve the router performance at a significantly high implementation cost; a multiplexed crossbar is cheaper to implement but requires more complex scheduling. Support for a larger number of VCs may mandate the use of a multiplexed crossbar from a practical viewpoint. For a multiplexed crossbar implementation, a multiplexer has to be used at the crossbar input ports and a demultiplexer at the crossbar output ports. Introduction of the additional multiplexer introduces a new contention point in the router. Figure 2 shows the various functional units along a router pipe when a router implements a multiplexed crossbar.

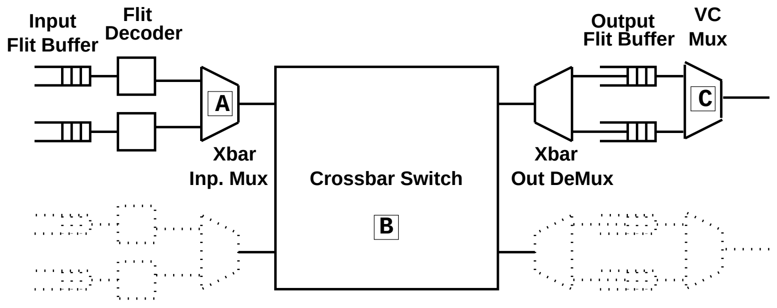


Fig. 2. Functional units along a router pipe for a 2 port router with 2 VCs per PC. Additional functional units such as the routing decision block and the arbitration unit are not shown. With a multiplexed crossbar as is shown in the figure, contention amongst multiple pipes can occur in the crossbar input multiplexer (A) for the crossbar input port, within the crossbar (B) for the crossbar output ports, and in the VC multiplexer (C) for the output PC.

In order to allocate bandwidth for different types of traffic, we plan to use a rate-based scheduling algorithm at one of the contention places as shown in Figure 2. The selection of a rate-based algorithm and its implementation are described next.

D. A Rate-based Scheduling Algorithm for QoS Support

There are primarily two classes of bandwidth scheduling algorithms: flow-based and frame-based. The flow-based algorithms like VirtualClock [24], Fair Queueing [38], General Processor Sharing (GPS) [39], Self-Clocked Fair Queueing (SCFQ) [40], and Frame-based Fair Queueing (FFQ) [41] use time stamps to make scheduling decisions, while the frame-based scheduling algorithms like Round Robin (RR), Weighted RR (WRR) [42], Deficit RR (DRR) [43], and Hierarchical RR (HRR) [44] poll queues sequentially during each round with different priorities. The frame-based algorithms usually assign a known priority to each queue. But, how to assign a priority to each queue with VBR traffic is not obvious. While the precomputed priority to each queue facilitates to reduce computation overhead, the flow-based algorithms require to timestamp and find the minimum amongst arriving packets every cycle. However, since in our router there could be multiple flows in each queue and we want to assign priorities based on flows, not a queue, we focus on flow-based algorithms in this paper.

For this study, we consider two different work conserving, rate-based schedulers; Fair Queueing and VirtualClock. Effectiveness of the two schemes have been analyzed by several researchers for QoS assurance in packet switched networks. In both these algorithms, there is a state variable associated with each channel i to monitor and enforce the rate for that channel. In VirtualClock, the variable is called auxiliary VirtualClock ($auxVC$); in Fair Queueing, it is called Finish Number (F). The computation of $auxVC$ and F for a connection i is shown in Table I. In VirtualClock, AT is the arrival time or wall clock time. In Fair Queueing, R is the number of rounds that has been completed for a hypothetical bit-by-bit round robin server, n is the weight factor, and P is the message length (in bits). $Vtick$ in VirtualClock and $\frac{P}{n}$ in Fair Queueing specify the inter-arrival time of messages. Therefore, a smaller value implies higher bandwidth. For best-effort traffic, the $Vtick$ is assigned the largest possible value. With $Vtick$ and $\frac{P}{n}$ specified, there is no difference between VirtualClock and Fair Queueing except that the Fair Queueing uses the round robin number(R) instead of the actual arrival time(AT) required for the VirtualClock. The computation complexity of R is $O(N)$ where N is total number of connections. Fair Queueing algorithms with less computation complexity can be found in [40], [41]. We can use the system clock for AT in the VirtualClock algorithm, and hence it needs no extra computation. It has been shown that both these schemes have similar performance [45] except that the VirtualClock algorithm cannot handle bursty traffic effectively without any input regulation. Traffic burstiness can be handled by regulating the traffic injection.

TABLE I
VirtualClock and Fair Queueing Algorithms

VirtualClock	Fair Queueing
$auxVC_i \leftarrow \max(AT, auxVC_i)$	$F_i \leftarrow \max(R, F_i)$
$auxVC_i \leftarrow auxVC_i + Vtick_i$	$F_i \leftarrow F_i + \frac{P_i}{n_i}$
timestamp the packets with $auxVC_i$	timestamp the packets with F_i

The above algorithms were developed for connection-oriented networks, where one channel is dedicated for each connection like the PCS, and when a connection is set up, a fixed *Vtick* (or $\frac{P}{n}$) value is assigned for the entire duration of the connection. This results in two problems. First, when dealing with VBR connections, one representative *Vtick* (or $\frac{P}{n}$) value may cause underutilization of the resources or incur higher message delay. Second, since one channel services one connection, a large number of VCs is required to handle multimedia streams. Consequently, it will lead to a complex router design with more hardware circuitry.

In this study, we are interested in a connectionless paradigm without any explicit connection setup since this provides more efficient use of the network resources. To overcome the above two problems, we modified the connection-oriented algorithms as follows: each message requests its required bandwidth at each router on its way to the destination, and the router implements the VirtualClock (or Fair Queueing) algorithm to allocate the requested bandwidth to its flits. So in our router, each message works as if it were a connection, and each flit works as if it were a message of the originally proposed algorithm. In the original algorithms, the fixed *Vtick* (or $\frac{P}{n}$) can be calculated from the average bandwidth requirement or the peak bandwidth requirement of the connection. *Vtick* (or $\frac{P}{n}$) in this study implies the intergeneration time between flits, and is given as

$$Vtick \text{ (or } \tfrac{P}{n}) = \frac{\text{message inter-arrival time}}{\text{message size in flits}}.$$

Thus, *Vticks* (or $\frac{P}{n}$) of two messages in the same connection can be different if they belong to different frames of different sizes. A message makes its request by carrying its *Vtick* (or $\frac{P}{n}$) value in the header.

When the tail flit leaves the router, its *Vtick* (or $\frac{P}{n}$) information in the router is discarded. We name the modified algorithms as Fine-Grained VirtualClock (FGVC) and Fine-Grained Fair Queueing (FGFQ), respectively, since bandwidth reservation is done at the message-level granularity.

In a router implementation with a multiplexed crossbar, contention for link bandwidth can occur at one of 3 places — the crossbar input multiplexer for the crossbar input port, within the crossbar for the crossbar output port and at the virtual channel multiplexer for the output physical channel. These are marked as (A), (B) and (C) respectively in Figure 2. All these places are potential candidates where a rate-based bandwidth allocation can be performed. We rule out contentions at (B) and (C) for the following reasons. In case (B), crossbar output port arbitration is performed at a message level granularity, whereas we are interested in flit-level bandwidth allocation. Case (C) corresponding to the VC multiplexer, is not a strong candidate, either. This is due to the fact that at most one of the VCs of an output PC can receive a flit from the multiplexed crossbar per router cycle. When only one of the VCs has a flit in any given cycle, the scheduling algorithm essentially behaves as a FIFO scheduler. Hence, we have chosen to implement the rate-based scheduler at the crossbar input multiplexer (A), which means that, at any given cycle, if multiple flits from different VCs are competing for the same output port of the crossbar, the one with the smallest auxVC_i will be chosen for transmission.

In a router that implements a full crossbar, there is no crossbar input multiplexer (nor a demultiplexer at the crossbar output). Thus, the only contention points are for the crossbar output ports (at the time of arbitration) and in the VC multiplexer. In such a router, the rate-based algorithm is implemented at the VC multiplexer (case (C)).

In order to select between FGVC and FGFQ schemes for the rest of the design, we conducted a performance analysis. We simulated both these schemes in a router and injected media traffic and best-effort traffic. We measured the inter-frame delivery time and standard deviation of delivery time for the media streams. The results with different input loads to the router are quite similar in both cases as depicted in Table II. However, since implementation of FGFQ is more complex for maintaining the round robin number(R), we use FGVC in the rest of our design. In order to avoid traffic burstiness, we regulate the traffic injection as described in Section IV. Figure 3 shows the final architecture of the MediaWorm router with a multiplexed crossbar and the FGVC scheduling algorithm.

TABLE II

COMPARISON OF FINE-GRAINED VIRTUALCLOCK WITH FINE-GRAINED FAIR QUEUEING WHEN THE RATIO OF REAL-TIME TO BEST-EFFORT TRAFFIC IS 80:20. INTER-FRAME TIME IS THE AVERAGED TIME DIFFERENCE OF FRAMES MEASURED AT THE DESTINATIONS, AND SD IS THE STANDARD DEVIATION OF THE INTER-FRAME TIME.

Load	Inter-frame time(*msec*)/SD	
	Fine-Grained VirtualClock	Fine-Grained Fair Queueing
60%	33.12/0.63	33.14/0.58
70%	32.74/1.25	32.74/1.22
80%	32.28/1.38	32.33/1.31

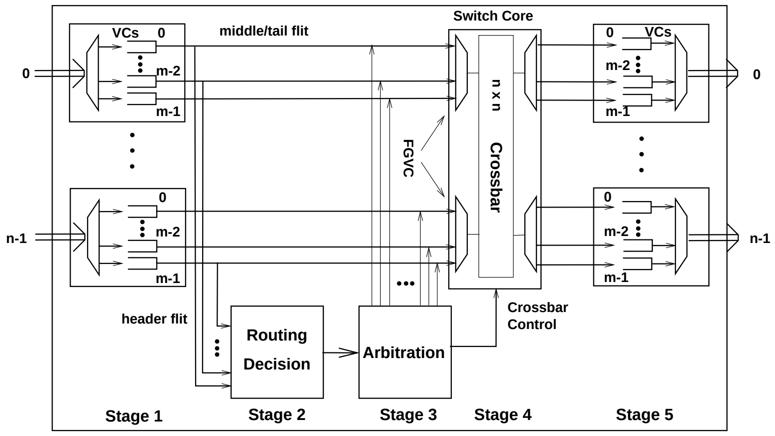


Fig. 3. MediaWorm router architecture

E. Interconnection Topologies – Fat Networks

Cluster interconnects are typically built with high degree switches. Myrinet [9] has 8 and 16 port routers, while Servernet-II [10] routers have 12 ports. These ports may be used to connect to other switches as well as to endpoints. These endpoints may be compute nodes such as clients and servers, as well as I/O devices.

The difference between such cluster networks and those in typical multiprocessors interconnects is that while multiple endpoints per switch may be common in the former, the latter typically has only one endpoint per switch. Depending on the expected traffic pattern, it is likely that multiple endpoints may place higher inter-switch bandwidth requirement on cluster interconnects.

Due to this reason, "fat" topologies have been proposed for clusters. Examples of fat topologies include fat-tree and fat-mesh [1]. Other cluster interconnects such as the tetrahedral topologies proposed by Horst [46] can also use "fat" links. Routers such as the Servernet-II [10] include hardware support for using multiple physical links connecting a pair of switches indistinguishably through the notion of "fat-pipes".

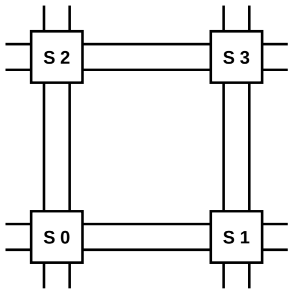


Fig. 4. A 4 switch fat-mesh interconnect. Each switch (S0–S3) is an (8×8) switch. Each *fat* link comprises 2 physical links.

Although most of the studies reported in this paper detail the performance of a single switch, we also experimentally analyze the performance of a fat mesh. The fat mesh used here is a (2×2) topology with 8 port crossbar switches. (We have limited our study for a smaller network due to exceedingly high simulation times. One can design a larger router and a larger network using our model.) Two physical links are used to interconnect each pair of switches in a 4-node mesh. Figure 4 illustrates the studied interconnect. We use deterministic routing and a message can use any one of the two links to traverse to the next node based on the current load.

F. Pipelined Circuit Switching

Pipelined Circuit Switching (PCS) [23] is a variant of wormhole switching in which a message is similarly segmented into flits. However, unlike wormhole routing in which middle and tail flits immediately follow the header as it progresses towards its destination, in PCS, flits of a message wait until the header (or probe) reserves the complete path up to the destination. Once such a path/connection has been established, an acknowledgment is sent from the destination to the source. The rest of the flits then move along this path in a pipelined manner (similar to wormhole switching). During path establishment, if the header cannot progress towards the destination, it can backtrack and try alternative paths if an adaptive routing is used. If no path can be established or if adaptive routing is not permitted, a negative-acknowledgment is sent back and the attempted connection is dropped. In this paper, we do not assume any adaptive routing capability[1]. Due to the requirement of complete path setup before transmission of flits in PCS, it may incur high path setup cost compared to wormhole switching. However, it can potentially provide better bandwidth reservation, which is

[1]PCS as originally proposed in [23], used non-minimal and adaptive routing capabilities with backtracking and re-routing. This leads to low connection dropping rates.

advantageous for real-time traffic. Our intention is to evaluate these trade-offs by comparing PCS with wormhole switching.

IV. Experimental Platform

A. Simulation Testbed

The above architectural concepts have been extensively evaluated through simulation. We have developed the MediaWorm router (MR), a traditional router with FIFO (TR), and a PCS router (PCS) using CSIM. The simulation models are quite flexible in the sense that one can specify the number of physical channels (PCs), number of VCs per PC, link bandwidth, CBR/VBR rates and the variation of VBR rate, flit size, message size (number of flits), and the ratio of real-time traffic (VBR and CBR) to best-effort traffic. In addition, using these routers, one can configure any network topology.

The detailed flit-level simulators capture the router pipeline and can process several simultaneous media streams per node. Typically, we gather simulation results over a few million messages. As a result, these simulations are extremely resource intensive, both in terms of simulation time and memory requirements. Two factors that determine simulation resources are the crossbar size, and physical channel bandwidth. Consequently, even though current technologies permit large crossbar sizes and over 1.28 Gbps link bandwidths, many of our simulations use smaller values for these parameters, without loss of generality, to keep them tractable. We have also conducted some experiments varying these parameters, and the overall trends/results still apply. The sizes of the input and output buffers in the router are one message long, respectively. Also, we have tested with deep buffers, but the results show minimal improvement.

The output parameters analyzed here are mean frame delivery interval ($\bar{d}$) for CBR/VBR messages, standard deviation of frame delivery intervals (σ_d) for CBR/VBR messages, and average latency for best-effort traffic. The delivery interval is measured as the difference between the delivery times of two successive frames at a destination. $\bar{d} = 33$ msec indicates a frame rate of 30 frames/sec at MPEG-2 rates. Coupled with a $\sigma_d = 0$, this implies jitter-free delivery. A higher $\bar{d}$ and/or σ_d implies jitters in transmission.

B. Workload

B.1 VBR and CBR Traffic

Two kinds of VBR traffic are simulated in the experiments. The first one is synthetic video streams with an average bandwidth of 4 Mbps and the other is realistic MPEG-2 video streams. The synthetic traffic consists of streams of messages from video frames, whose size is selected from a normal distribution with a mean of 16,666 bytes and standard deviation of 3333 bytes. (This corresponds to 4 Mbps MPEG-2 streams.) The realistic traffic is generated from MPEG-2 traces [16] shown in Table III, where there are 7 video traces with different bandwidth requirements.

TABLE III
MPEG-2 Video Sequence Statistics

Video Sequences	Average Bandwidth Requirements (Kb/s)	Average Size of I Frame (Kbits)	Average Size of P Frame (Kbits)	Average Size of B Frame (Kbits)
1	7,138.2	430.2	295.6	194.2
2	15,231.4	839.4	680.6	401.0
3	13,526.4	534.7	569.4	387.8
4	8,356.5	393.3	340.3	241.4
5	6,124.2	350.6	242.9	172.9
6	18,406.0	974.7	721.9	529.6
7	13,497.9	637.6	536.0	394.3

Each stream generates 30 frames/sec, and each frame (I, P, or B frame) is fragmented into 20/40-flit size messages (except possibly the last message of a frame), with each message carrying the bandwidth requirements (*Vtick* information for the FGVC algorithm), and the routing information in its header flit. As a result, the network treats each message of a stream independent of the others. The injection rate for the messages of a stream is determined by the message size and the number of messages constituting a frame. Once the injection rate is determined, an input regulator injects messages of a frame at a regular interval of (33msec/number of messages). For instance, with 200 messages in a frame, the interval between successive message injections is 165 microseconds. Such an input regulator provides two advantages. First, in addition to avoiding traffic burstiness, the input regulator allows coexistence of messages from different streams in the queues. Without this ability, the streams are queued only at a frame-level granularity, thereby increasing the delay of certain streams. Second, the input regulator also helps transmission of best-effort traffic in between video frame messages.

In the case of PCS, each stream is transmitted over a distinct connection (as it is connection-oriented). The first flit of the stream establishes the circuit between the source and destination endpoints, while reserving the required bandwidth at the intermediate switches (the required *Vtick* for the entire stream). The frames of the stream are logically grouped into flits, with each group injected into the established circuit at a specified rate (similar to how messages are generated in the wormhole switching case).

In PCS, each connection (and hence a stream) also needs a distinct VC. Therefore, the number of VCs supported by the hardware has to be greater than or equal to the maximum number of concurrent streams in the workload. In the MediaWorm, each message carries the routing and bandwidth information. As a result, it would be possible to support multiple connections on a single VC. This would make sense only when the bandwidth available to a VC is at least as large as the sum of the bandwidths demanded by the streams on that VC. This is however not a problem because each message carries its *Vtick* requirement.

It should be noted that stream establishment does not actually fail in wormhole switching. In PCS, on the other hand, a connection establishment probe may not necessarily succeed. This is termed as *dropping* of a connection. It is assumed that connections may be dropped only at stream set-up.

Once the input VC for a connection is determined, the destination is picked randomly using a uniform distribution of all nodes, and the destination VC is also drawn randomly from a uniform distribution of the VCs available for VBR traffic.

The generation of the CBR traffic is identical to the synthetic VBR traffic, with the exception that the frame size is kept constant (at 16,666 bytes).

B.2 Best-effort Traffic

The best-effort traffic is generated with a given injection rate, λ, that is allocated to this class of traffic (explained in the next subsection), and follows the Poisson distribution. The message length is kept constant at 20/40 flits according to the message length of real-time traffic, and its destination is picked from a uniform distribution of the nodes in the system. The input and output VC for a message are picked from a uniform distribution of the available VCs for this traffic class.

B.3 Traffic Mixes

An important parameter that is varied in our experiments is the input load. This is expressed as a fraction of the physical link bandwidth. For a specified load, we consider different mixes ($x : y$, where $x/(x + y)$ is the fraction of the load for the VBR/CBR component and $y/(x + y)$ is the fraction of the load for the best-effort component) to generate integrated traffic. We divide the VCs into two disjoint groups. $x/(x + y)$ % of the VCs are reserved for the VBR/CBR traffic, and the remaining are allocated to the best-effort traffic. As mentioned earlier, the number of simultaneous VBR/CBR streams that can be supported at a node is limited by the number of VCs in the case of a PCS router. In the MediaWorm, it is limited by the number of VCs and the bandwidth allocated to a VC. For instance, if a physical channel can support 400 Mbps, and the total number of VCs is 16, then we can support at most 6 connections per VC to simulate synthetic VBR. If $x = y = 1$, then the number of VCs dedicated for VBR/CBR traffic is 8, and there can be at most $6 \times 8 = 48$ outstanding/incoming streams at each node in the system. It should be noted that we have an implicit admission control because the traffic injection into each physical link does not exceed the link bandwidth.

V. Performance Results

In this section, we experimentally analyze the performance results for an 8-port MediaWorm router with varying parameters as well as that of a (2×2) fat mesh. The router parameters used in this performance study are given in Table IV.

TABLE IV
Simulation Parameters

Switch Size	8×8
Flit Size	32 / 128 bits
Message Size	20 / 40 flits
Flit Buffers	20 / 40 flits
PC Bandwidth	400 Mbps / 1.6 Gbps
VCs/PC	variable (wormhole), 24 (PCS)
Streams/VC	variable (wormhole), 1 (PCS)

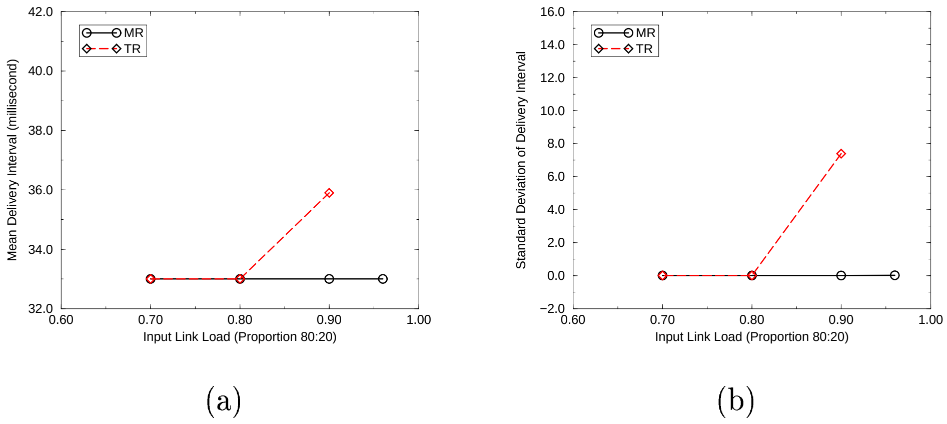


Fig. 5. Comparison of MR and TR ((8 × 8) switch, 16 VCs, 400 Mbps links, $x : y$ = 80:20).

A. *Comparison of MediaWorm and Traditional Routers*

We first begin by examining how a traditional router (TR) and the MediaWorm router (MR) perform with multimedia/mixed traffic. Note that the main difference between the two routers is the scheduling algorithm. The TR uses a FIFO scheduler, whereas the proposed MR uses the FGVC algorithm. Figure 5 shows the mean delivery interval ($\bar{d}$) and its standard deviation (σ_d) for this router with a mixture of synthetic VBR and best-effort traffic (80:20).

We can see that both $\bar{d}$ and σ_d start growing beyond a load of 0.8, showing that there would be significant jitters in delivery of VBR traffic beyond this point. Compared to this, the MR can provide jitter-free delivery even up to a link load of 0.96 (the load of the real-time component is around 0.75). This clearly shows the need for a rate-based scheduling algorithm to effectively administer the available bandwidth for media streams.

B. *Comparison of CBR and VBR Traffic Results*

Figure 6 depicts the $\bar{d}$ and σ_d results with CBR and only synthetic VBR traffic (there is no best-effort traffic). It can be gleaned that both the traffic classes exhibit nearly identical performance, with the CBR traffic experiencing jitter-free performance for a slightly higher load. Although, both CBR and VBR streams have the same mean bandwidth requirement, CBR streams, by their nature, are also intuitively expected to experience better jitter tolerance. Since, VBR streams present a more challenging workload, we use VBR streams in the rest of the experiments in this paper.

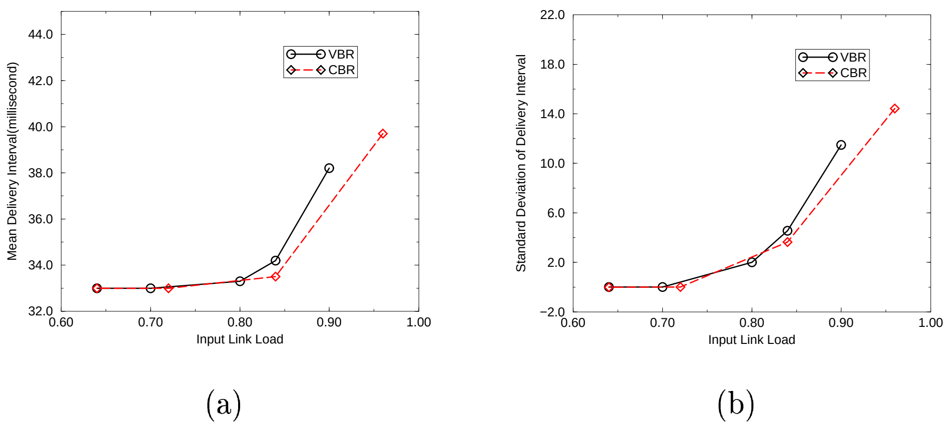


Fig. 6. Comparison of CBR and Synthetic VBR traffic in the MediaWorm router ((8 × 8) switch, 16 VCs, 400 Mbps links, all real-time traffic).

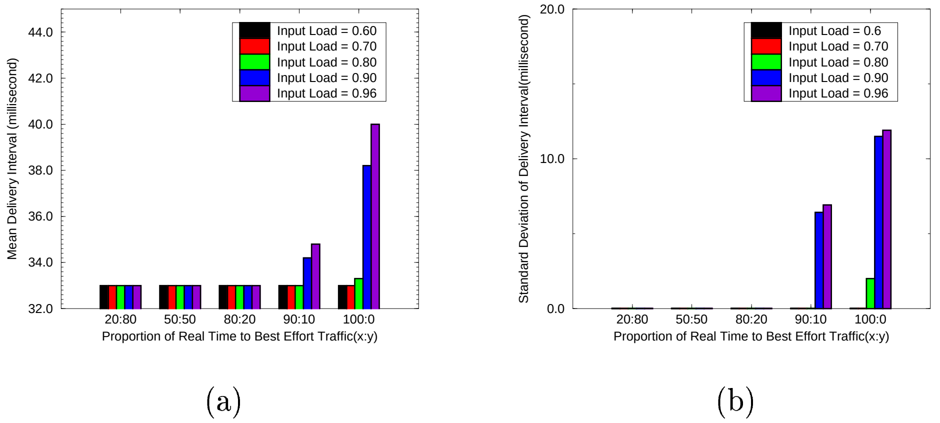


Fig. 7. Mixed Traffic (Synthetic VBR + best-effort traffic) ((8 × 8) switch, 16 VCs, 400 Mbps links).

C. Results with Mixed Traffic

Next, we vary the ratio of real-time (only synthetic VBR) and best-effort traffic for different input loads, and study the effect on jitter for VBR traffic and average latency for the best-effort traffic. Figure 7 shows the variation of $\bar{d}$ and σ_d for these workloads. It can be observed that up to an input load of 0.80, there is no jitter for VBR traffic regardless of the traffic mix. Beyond a load of 0.80, it is only when the real-time traffic becomes a dominant component, does the jitter become significant. The effect of VBR traffic on the average latency of best-effort traffic (in microseconds) is given in Table V. For a given mix, the latency degrades with an increase in the load. The presence of real-time traffic also increases the latency of the best-effort traffic at a given load. This is a consequence of the higher priority given by the FGVC algorithm to the real-time traffic.

TABLE V

Average Latency for Best-effort Traffic (8×8 switch, 16VCs, 400Mbps links)

	Input Load				
x:y	0.60	0.70	0.80	0.90	0.96
20:80	6.3	9.0	16.2	36.9	43.6
50:50	7.7	11.4	25.5	56.1	64.6
80:20	10.3	15.8	39.7	106.9	Sat.
90:10	11.9	19.3	106.2	Sat.	Sat.

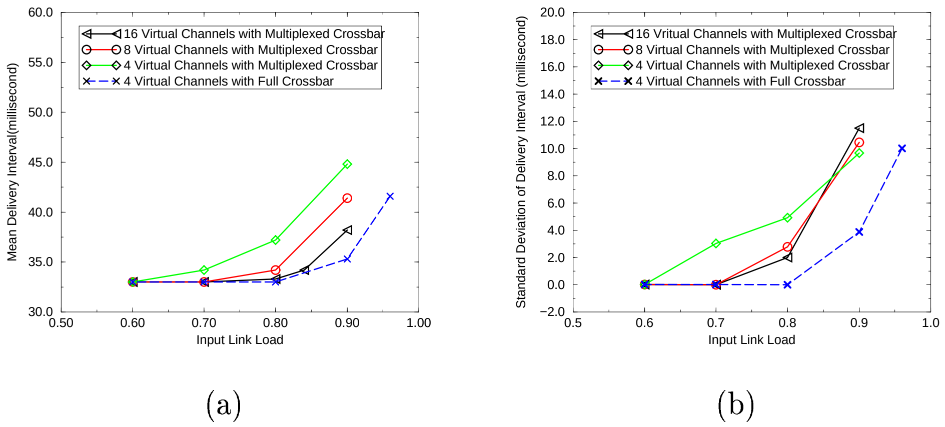


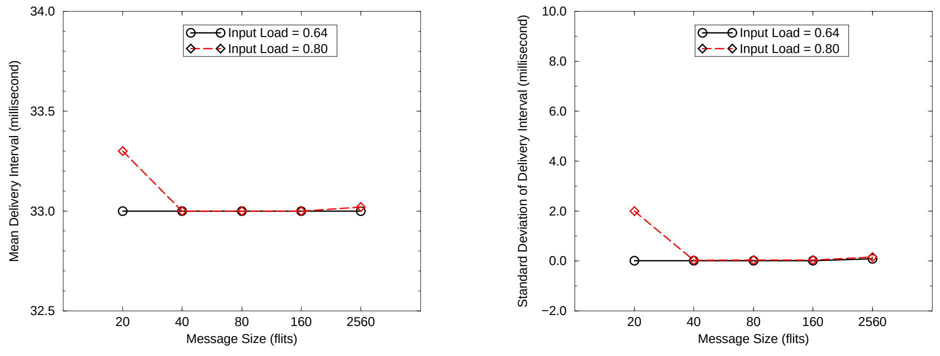
Fig. 8. Impact of VCs and Crossbar Capabilities ((8 × 8) switch, 400 Mbps links, $x : y$ = 100:0).

D. Impact of VCs and Crossbar Capabilities

It should be noted that our workload generates multiple connections for each available VC. An important design consideration is to determine whether one should support more VCs with fewer connections per VC, or vice versa. Intuitively, it may appear that a larger number of VCs would improve performance. The performance results in Figure 8 also confirm this intuition, where the 16 VC case gives jitter-free performance up to a higher load compared to the 4 and 8 VC cases. However, supporting a large number of VCs may require additional resources in the router. On the other hand, it is possible to use a full crossbar (instead of a multiplexed one) with lower number of VCs. This is examined for the 4 VC case (i.e. a 32×32 crossbar), which shows better performance than 8 VCs with the multiplexed crossbar, and competitive performance compared to the 16 VC results.

E. Effect of Message Size on Jitter

Our next experiment examines the impact of message size on synthetic VBR traffic. We vary message size for two different input loads (0.64 and 0.8), and examine changes in $\bar{d}$ and σ_d. The results in Figure 9 show that except for very small message sizes, there is little impact on QoS for real-time



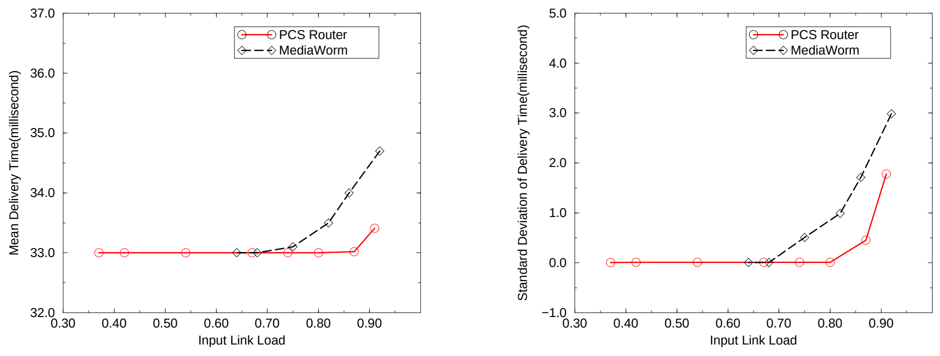
(a) (b)

Fig. 9. Effect of message size on jitter ((8 × 8) switch, 400 Mbps link bandwidth, 16 VCs, all synthetic VBR traffic).

traffic. For very small sizes, the effect of the header flit overhead becomes noticeable. For instance, 1 header flit in a message size of 20 flits consumes 5% of the stream bandwidth. These results show that we do not really need large messages for media traffic. In fact, best-effort traffic latency may benefit from smaller real-time messages.

F. *Comparison of MediaWorm and PCS Routers*

PCS is expected to provide good performance for VBR traffic. This is because PCS is a connection-oriented switching paradigm, and hence can reserve bandwidth at the time of connection establishment. However, it requires a VC per stream, thereby mandating a large number of VCs per PC for high link bandwidth.



(a) (b)

Fig. 10. MediaWorm and PCS comparison ((8 × 8) switch, 100 Mbps link bandwidth, 24 VCs).

In this experiment, we compare the performance of the MediaWorm router to that of the PCS router. Note that this is the only experiment that we perform for a link bandwidth of 100Mbps (24-25

VBR streams can be supported per link, each with 4Mbps bandwidth requirement). This is primarily because of the simulation complexity for supporting the large number of VCs (up to 100 VCs) that would be required for 400 Mbps link bandwidth in the PCS router.

TABLE VI
NUMBER OF ATTEMPTED, ESTABLISHED AND DROPPED CONNECTIONS FOR REACHING A CERTAIN INPUT LOAD IN A PCS ROUTER. THE VALUES PRESENTED ARE FOR AN (8×8) ROUTER WITH 24 VCS, 100MBPS LINKS.

Input Load	**#Conn. Attempts**	**# Established**	**# Dropped**
0.91	718	187	531
0.87	540	175	365
0.80	476	160	316
0.74	372	148	224
0.67	332	134	198
0.64	224	107	117
0.42	172	83	89
0.37	166	73	93

As can be expected, the MediaWorm router can support jitter-free performance only up to a load of about 0.7 compared to over 0.8 in the case of PCS. This is, however, not a fair comparison because all streams started on the MediaWorm router are accepted, whereas the PCS router drops many connections that contend for busy resources. For the same operating load, this in effect unfairly improves the crossbar utilization for accepted connections in the PCS router compared to that for the MediaWorm router.

While the PCS router provides superior performance, this is at the cost of high resource requirements (large number of VCs) as well as a very high number of dropped connections. The number of accepted and dropped connections for various input loads for the PCS router is shown in Table VI.

These results show that for most realistic operating conditions (an input load of 0.7 is reasonably high), the MediaWorm router can deliver as good (jitter-free) performance as a PCS router for real-time traffic, while not turning down connection establishment requests as done in the PCS router. (The connection drop rate can be minimized by using several alternatives as proposed in [23].) Moreover, by increasing the number of VCs in the MediaWorm router to match with the PCS implementation, its performance could be similar to that of the PCS router at higher load.

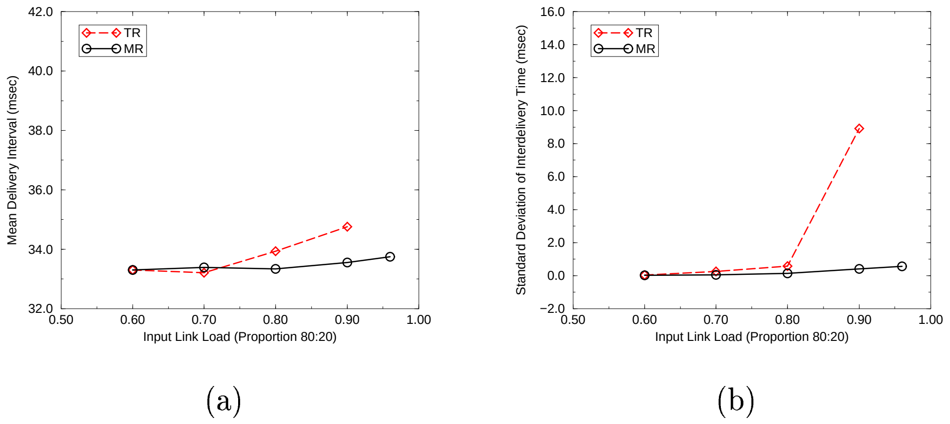


Fig. 11. TR vs. MR with MPEG-2 Video Traffic ((8 × 8) switch, 16 VCs, 1.6 Gbps, $x : y = 80 : 20$).

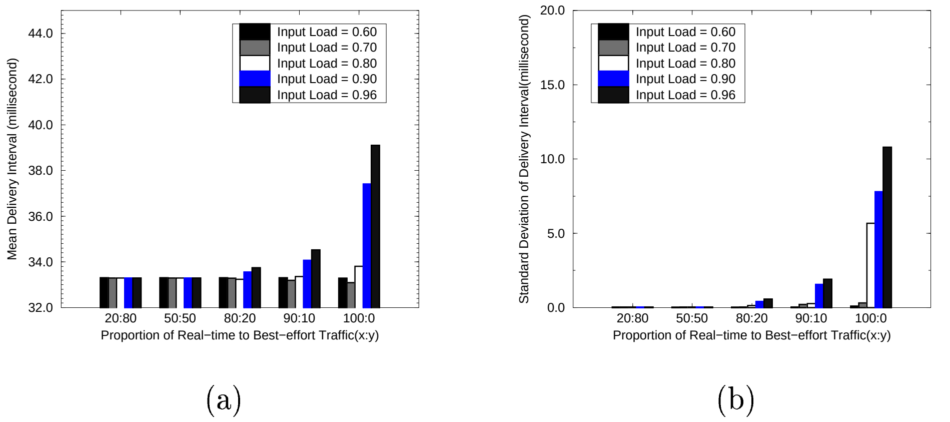


Fig. 12. Mixed Traffic (MPEG-2 Video Trace + best-effort traffic, (8 × 8) switch, 16 VCs, 1.6 Gbps).

G. Results with MPEG-2 Video Traces

Here we examine the performance results of a traditional router and the MediaWorm router with realistic MPEG-2 video traffic shown in Table III. Figure 11 shows the mean delivery interval ($\bar{d}$) and its standard deviation (σ_d) for each router model. Some of the data points of the TR were dropped due to saturation. The results with realistic VBR are almost identical to those with synthetic VBR of Figure 5, although $\bar{d}$ of TR in Figure 11 is slightly better at 90% load. Next, we vary the ratio of real-time (MPEG-2 video) and best-effort traffic for different input loads, and study the effect on jitter for VBR. Figure 12 shows the variation of $\bar{d}$ and σ_d for these workloads. Again we can observe similar results as shown in Figure 7. Although the video traces in Table III have much wider bandwidth variation, the overall results with synthetic and actual traces are almost similar.

H. Fat-Mesh Results

Up to this point, we have focussed on the performance of a single router with CBR/VBR and best-effort traffic. In this subsection, we try to examine the performance implications of using such routers in a fat-mesh interconnect. We limit this study to a modest 4-node network (shown in Figure 4) due to limited simulation resources. In general, it can be expected that an interconnect with multiple routers may have lower performance than that of a single router. This would be due to additional points of resource contention in a network.

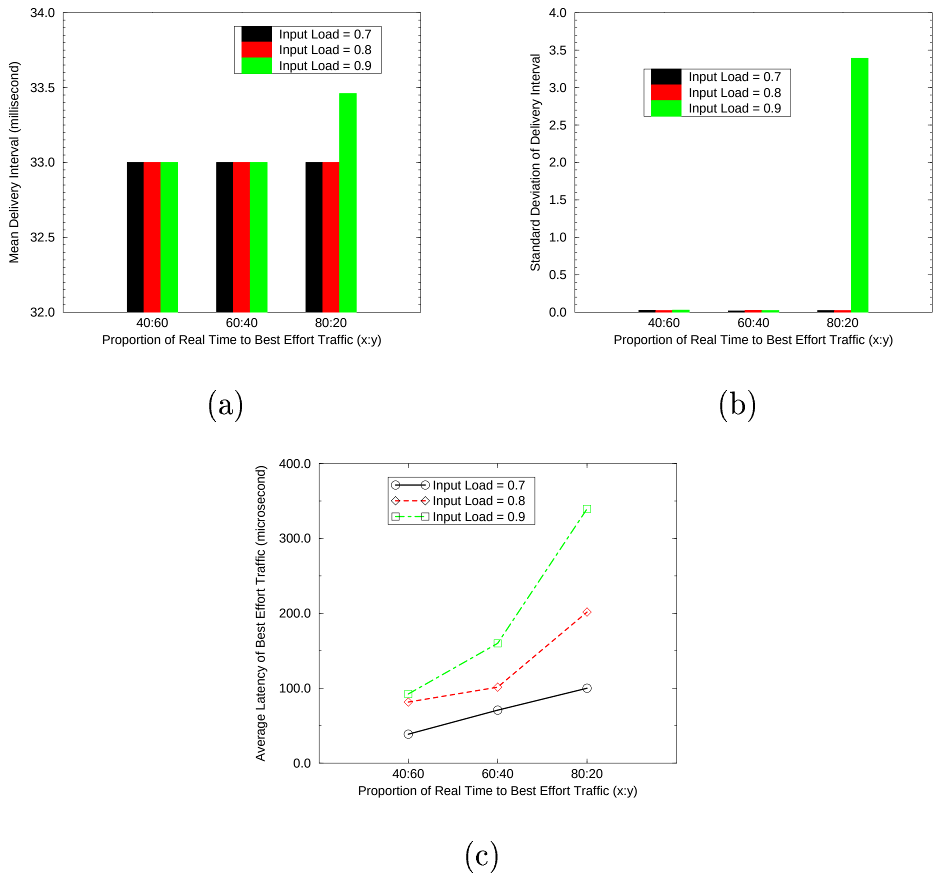


Fig. 13. Performance of a (2×2) fat mesh (8×8) switches, 400 Mbps link bandwidth, 16 VCs).

Figure 13 (a) and (b) shows the change in mean delivery interval and the corresponding standard deviation for synthetic VBR traffic. This is studied with both increasing load and increasing proportion of VBR traffic. The results indicate that VBR performance remains good for smaller proportions of VBR traffic (40% and 60 %) even for a total input load of 0.9 of PC bandwidth capacity. Only at a load of 0.9 with 80% of traffic being VBR, does VBR performance degrade. This good performance for VBR is at the expense of best-effort traffic and is shown in Figure 13 (c). As expected, for any given load, average latency of best-effort traffic increases with increasing proportion of VBR traffic.

It is also illustrative to compare the performance of a (2×2) fat mesh to that of a single switch. As expected, the maximum input load (for a given proportion of VBR to best-effort traffic) that provides jitter-free performance for VBR traffic is lower in the fat-mesh than in the case of a single switch. This can be inferred by comparing Figures 7 (a) and (b) with Figures 13 (a) and (b). For example, with a load of 0.9 and a traffic mix of 80:20, a single switch can provide jitter-free performance, while the fat-mesh cannot.

Admission control criteria, thus, have to consider (for an expected traffic pattern) what is the maximum load and proportion of VBR to best-effort traffic that will provide statistically acceptable QoS to VBR traffic as well as an acceptable latency for best-effort traffic. This load would then determine the number of VBR streams that may be accepted for service.

VI. Concluding Remarks

Widespread use of cluster systems in diverse application environments is placing varied communication demands on their interconnects. Commercial routers for these environments currently support wormhole switching. Although wormhole routers can provide small latencies and good performance for best-effort traffic, these routers are unable to provide QoS guarantees for soft real-time applications such as streaming media.

Our study is motivated by the need to simultaneously handle multiple such traffic types that are becoming important and prevalent in clustered environments. We also feel that it is imperative to leverage off of the existing, mature and commodity technology, i.e. wormhole switching, for providing a cost-effective solution rather than using relatively new or hybrid switching alternatives proposed by other researchers. We have proposed a new router architecture called *MediaWorm* with only one major modification compared to "vanilla" wormhole routers — incorporating a *rate proportional* resource scheduler called FGVC, instead of the common *rate agnostic* schedulers such as FIFO or round-robin.

We have studied the capabilities of the MediaWorm in supporting real-time and best-effort traffic. The main conclusions of our study are the following:

- The FGVC scheduler can provide considerably improved performance for traffic that require soft real-time guarantees (VBR/CBR).

• The MediaWorm router design shows that there is no adverse effect on the performance of VBR traffic in the presence of best-effort traffic. However, as the share of VBR traffic increases for a given load, this adversely effects the latency of best-effort traffic. A wormhole router can provide jitter-free delivery to VBR/CBR traffic up to a load of 70–80% of the physical channel bandwidth.

• Although the performance of a PCS router is slightly better than the MediaWorm, PCS routers are more complex than wormhole routers and they may drop a large number of connections.

• Finally, we find that performance of a small fat-mesh network is comparable to that of a single switch. Although it is difficult to extrapolate performance to much larger clusters directly from our present results, we expect that clusters designed with appropriate bandwidth balance amongst various links by using fat-topologies and MediaWorm-like switches should be able to provide good performance for both real-time and best-effort traffic.

In summary, our results suggest that by augmenting conventional wormhole routers with rate-based resource scheduling techniques, one can provide a viable, cost-effective switch for cluster interconnects to support both real-time and best-effort traffic mixes. Higher level admission control strategies, devised to track network load and proportion of different traffic mixes, would be able to assure good performance for both types of traffic. Moreover, all the design modifications discussed in this paper should be applicable to VCT routers.

The study presented in this paper is our maiden investigation into the design and performance of cluster switches. We plan to expand our investigation in the following directions.

• Characterization of communication demands of popular cluster applications to quantify traffic patterns, requirements, and proportion of various traffic types is important for router design and evaluation.

• In this paper, we have investigated static proportions of traffic mixes with statically partitioned resources (VCs). A more practical scenario would be that of dynamic mixes with dynamically partitioned resources. One way to provide this is to permit message preemption (contrary to the typical *hold-and-wait* resource usage) in wormhole routing [22].

• A scalability study to larger networks and large number of streams is very resource intensive for our current simulation models. We plan to develop more light-weight simulation models for our future studies. We also aim to support more hybrid router models for comparing and evaluating alternative design options.

• Finally, the MediaWorm project also aims to investigate network interface architectures for support of multiple traffic types and appropriate admission control strategies.

References

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

[2] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, May 1992.

[3] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing : The SGI SPIDER Chip," in *Proceedings of Symposium on High Performance Interconnects (Hot Interconnects)*, August 1996, pp. 141–146.

[4] S. L. Scott and G. M. Thorson, "Optimized Routing in the Cray T3D," in *Proceedings of Parallel Computer Routing and Communications Workshop (PCRCW)*. May 1994, pp. 281–294, Springer Verlag Lecture Notes in Computer Science.

[5] S. L. Scott and G. M. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," in *Proceedings of Symposium on High Performance Interconnects (Hot Interconnects)*, August 1996, pp. 147–156.

[6] J. Carbonaro and F. Verhoorn, "Cavallino: The Teraflops Router and NIC," in *Proceedings of Symposium on High Performance Interconnects (Hot Interconnects)*, August 1996, pp. 157–160.

[7] M. D. May, "The Next Generation Transputers and Beyond," in *Proceedings of 2nd European Distrinuted Memory Conference*, April 1991, pp. 7–22.

[8] 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 Journal*, vol. 34, no. 2, pp. 185–204, 1995.

[9] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su, "Myrinet: A Gigabit-per-second Local Area Network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, February 1995.

[10] D. Garcia and W. Watson, "Servernet II," in *Proceedings of the 1997 Parallel Computing, Routing, and Communication Workshop (PCRCW'97)*, June 1997.

[11] J. H. Kim and A. A. Chien, "Rotating Combined Queueing (RCQ): Bandwidth and Latency Gurantees in Low-Cost, High-Performance Networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, May 1996, pp. 226–236.

[12] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, vol. C–36, no. 5, pp. 547–553, May 1987.

[13] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, September 1979.

[14] J. Duato, S. Yalamanchili, M. B. Caminero, D. Love, and F. J. Quiles, "MMR: A High-Performance Multimedia Router-Architecture and Design-Tradeoffs," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, January 1999, pp. 300–309.

[15] H. Eberle and E. Oertli, "Switcherland: A QoS Communication Architecture for Workstation Clusters," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 1998, pp. 98–108.

[16] M. B. Caminero, F. J. Quiles, J. Duato, D. Love, and S. Yalamanchili, "Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic," in *Proceedings of the Third International Workshop on Communication, Architecture and Applications on Network Based Parallel Computing (CANPC'99)*, January 1999, pp. 62–76.

[17] J. H. Kim, *Bandwidth and Latency Guarantees in Low-Cost, High-Performance Networks*, Ph.D. thesis, Department of Electrical Engineering, University of Illinois at Urbana-Champaign, 1997.

[18] J. Rexford, J. Hall, and K. G. Shin, "A Router Architecture for Real-Time Point-to-Point Networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, May 1996, pp. 237–246.

[19] J.-P. Li and M. Mutka, "Priority Based Real-Time Communication for Large Scale Wormhole Networks," in *Proceedings of International Parallel Processing Symposium*, May 1994, pp. 433–438.

[20] M. Gerla, B. Kannan, B. Kwan, P. Palnati, S. Walton, E. Leonardi, and F. Neri, "Quality of Service Support in High-Speed Wormhole Routing Networks," in *Proceedings of 1996 International Conference on Network Protocols*, October 1996, pp. 40–47.

[21] K. Connelly and A. A. Chien, "FM-QoS: Real-Time Communication Using Self-Synchronizing Schedules," in *Proceedings of Supercomputing Conference*, November 1997.

[22] H. Song, B. Kwon, and H. Yoon, "Throttle and Preempt: A New Flow Control for Real-Time Communications in Wormhole Networks," in *Proceedings of International Conference on Paralle Processing (ICPP)*, August 1997, pp. 198–202.

[23] P. T. Gaughan and S. Yalamanchili, "A Family of Fault-tolerant Routing Protocols for Direct Multiprocessor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 5, pp. 482–497, May 1995.

[24] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks," *ACM Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.

[25] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "Arhitecture and Implementation of the Reliable Router," in *Proceedings of Symposium on High Performance Interconnects (Hot Interconnects)*, Stanford University, Palo Alto, CA, August 1994.

[26] S. Konstantinidou and L. Snyder, "The Chaos Router," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1386–1397, December 1994.

[27] W.-D. Weber, S. Gold, P. Helland, T. Shimizu, T. Wicki, and W. Wilcke, "The Mercury Interconnect Architecture: A Cost-effective Infrastructure for High-Performance Servers," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1997, pp. 98–107.

[28] A. G. Nowatzyk, M. C. Browne, E. J. Kelly, and M. Parkin, "S-Connect: from Network of Workstations to Supercomputer Performance," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, June 1995.
[29] F. Chong, H. Minsky, A. DeHon, M. Becker, S. Peretz, and E. Egozy, "METRO: A Router Architecture for High-Performance, Short-Haul Routing Networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1994, pp. 266–277.
[30] J. D. Allen, P. T. Gaughan, D. E. Schimmel, and S. Yalamanchili, "Ariadne — An Adaptive Router for Fault-Tolerant Multicomputers," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1994, pp. 278–288.
[31] K. G. Shin and S. W. Daniel, "Analysis and Implementation of Hybrid Switching," in *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1995, pp. 211–219.
[32] A. A. Chien and J. H. Kim, "Approaches to Quality of Service in High-Performance Networks," in *Proceedings of Parallel Computer Routing and Communications Workshop.* July 1997, Lecture Notes in Computer Science, Springer-Verlag.
[33] S. Balakrishnan and F. Özgüner, "A Priority-Based Flow Control Mechanism to Support Real-Time Traffic in Pipelined Direct Networks," in *Proceedings of International Conference on Paralle Processing (ICPP)*, August 1996, pp. 120–127.
[34] B. Kim, J. Kim, S. Hong, and S. Lee, "A Real-Time Communication Method for Wormhole Switching Networks," in *Proceedings of International Conference on Paralle Processing (ICPP)*, August 1998, pp. 527–534.
[35] A. S. Vaidya, C. R. Das, and A. Sivasubramaniam, "A Testbed for Evaluation of Fault-tolerant Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 10, pp. 1052–1066, October 1999.
[36] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, "LAPSES: A Recipe for High Performance Adaptive Router Design," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, January 1999, pp. 236–243.
[37] L.-S. Peh and W. J. Dally, "A Delay Model for Router Micro-architectures," in *Proceedings of Symposium on High Performance Interconnects (Hot Interconnects)*, August 2000.
[38] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Journal of Internetworking Research and Experience*, pp. 3–26, October 1990.
[39] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 344–357, June 1993.
[40] S. J. Golestani, "A Self-clocked Fair Queueing Scheme for Broadband Applications," in *Proceedings of IEEE INFOCOM*, 1994, pp. 636–646.
[41] D. Stiliadis and A. Varma, "Efficient Fair Queueing Algorithms for Packet-Switched Networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 175–185, April 1998.
[42] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1265–1279, October 1991.
[43] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, September 1995, pp. 231–242.
[44] C. Kalmanek, H. Kanakia, and S. Keshav, "Rate Controlled Servers for Very High-Speed Networks," in *Proceedings of IEEE Global Telecommunications Conference*, December 1990, pp. 300.3.1–300.3.9.
[45] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 611–623, April 1998.
[46] R. W. Horst, "TNet: A Reliable System Area Network," *IEEE Micro*, pp. 37–45, February 1995.

Ki Hwan Yum is a Ph.D. student in the Department of Computer Science and Engineering at the Pennsylvania State University. His 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. He received an MS degree in Computer Science from Pohang University of Science and Technology, Korea and a BS degree in Mathematics from Seoul National University, Korea. He worked as a member of technical staff in Korea Telecom for 3 years. Yum is a student member of the IEEE and the ACM.

Eun Jung Kim is a Ph.D. student in the Department of Computer Science and Engineering at the Pennsylvania State 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 received an MS degree in Computer Science from Pohang University of Science and Technology, Korea and a BS degree in Computer Science from Korea Advanced Institute of Science and Technology, Korea. She worked as a member of technical staff in Korea Telecom for 3 years. Kim is a student memeber of the IEEE and the ACM.

Chita R. Das received the M.Sc. degree in electrical engineering from the Regional Engineering College, Rourkela, India, in 1981, and the Ph.D. degree in computer science from the Center for Advanced Computer Studies, University of Louisiana, Lafayette, in 1986. Since 1986, he has been with the 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 computing, cluster computing, Internet QoS, multimedia systems, performance evaluation, and fault-tolerant computing. He is currently an Editor of the IEEE Transactions on Computers and has served on the Editorial Board of the IEEE Transactions on Parallel and Distributed Systems. Dr. Das is a Fellow of the IEEE and a member of the ACM.

Aniruddha S. Vaidya received his B.Tech in electrical engineering from the Banaras Hindu University, Varanasi (India) in 1991, his M.S. in electrical engineering from the Indian Institute of Science, Bangalore (India) in 1994, and his Ph.D. in Computer Science and Engineering from the Pennsylvania State University, University Park in 1999. Since 1999, he has been working at Intel Corporation in Santa Clara, CA, where is a Senior Platform Architect. His interests are in multiprocessor architecture, interconnection networks, and their performance evaluation and fault-tolerance aspects. Dr. Vaidya is a member of the IEEE and the IEEE Computer Society.