Investigating QoS Support for Traffic Mixes with the MediaWorm Router [*]

Ki Hwan Yum   Aniruddha Vaidya   Chita R. Das   Anand Sivasubramaniam
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {yum,vaidya,das,anand}@cse.psu.edu

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 otherwise well known for designing high performance interconnects. In particular, we are interested in supporting multimedia video streams in addition to the conventional best-effort traffic. The proposed MediaWorm router uses a rate-based bandwidth allocation mechanism, called Virtual Clock, 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 the 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 the simpler and cheaper design makes it an attractive alternative. Simulation of a (2×2) *fat-mesh using this router suggests that clusters designed with appropriate bandwidth balance between links can provide good performance for different types of traffic.*

Index Terms: Cluster Network, Pipelined Circuit Switching, Quality of Service, Router Architecture, Virtual Clock, Wormhole Router

1 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 has primarily focussed on designing scalable, high performance networks (low latency and high bandwidth) to accommodate best-effort communication traffic. Over the years, network design philosophy has converged towards direct network topology, wormhole switching, and virtual channel (VC) flow control [7] to meet these design goals. These research ideas have manifested in many commercial switch/router designs [13, 27, 28, 4, 24, 31] and have migrated to, and been successfully assimilated in, cluster interconnects [2, 14]. However, many of the new applications that have real-time constraints dictate 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 a network (router) that can provide high performance and QoS guarantees for integrated or mixed traffic.

The ATM Forum has defined three traffic classes called constant bit rate (CBR), variable bit rate (VBR) and available bit rate (ABR) [21]. CBR is generated during uncompressed video/audio transmission while VBR is exhibited 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 [11]. One is based on the cut-through switching mechanisms (wormhole [9] and virtual-cut-through (VCT) [18]), originally proposed for multiprocessor switches, and the other is based on packet switching. 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 [10, 12]. Therefore, none of the existing network architectures are optimized to handle both best-effort and real-time traffic in clustered environments[1].

In view of this, a few researchers have explored the possibility of providing support for QoS in router architectures [10, 3, 20, 26, 12]. 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

[*]This research is supported in part by NSF grants MIPS-9634197, MIPS-9701475, CCR-9900701, and equipment grants from NSF and IBM.

[1]QoS is a generic term. Since our interest is on real-time multimedia applications, the terms QoS and real-time are used interchangeably.

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 [23, 16, 6, 19, 30]. 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 (VBR, CBR and best-effort) within the framework of wormhole switching, while suggesting minimal modifications to the existing hardware. Real-time support requires providing some mechanism within the router that recognizes the bandwidth requirements of VBR and CBR traffic, and accommodates these requests. One can borrow concepts proposed by the real-time community 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. In particular, the paper attempts to address the following questions:

- Can we provide a mechanism within the wormhole router that provides soft-guarantees to VBR and CBR traffic? Based on this, can we develop an admission control scheme that restricts accesses to the network so that QoS requirements of currently served requests are not compromised?
- 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 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 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*, that is a modification of a conventional pipelined router for meeting bandwidth requirements. A rate-based scheduling algorithm, called Virtual Clock [35], is used to schedule router resources amongst the competing connections. 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 it using a 2×2 fat-mesh topology.

The results indicate that the Virtual Clock algorithm in the MediaWorm does indeed improve the QoS delivered to real-time traffic compared to the FIFO scheduling algorithm that is typically employed to allocate resources within a conventional wormhole router. In the case of a single switch, MediaWorm can provide jitter-free delivery up to an input load of 70–80% of the physical channel bandwidth. For most realistic operating conditions, MediaWorm can deliver as good (jitter-free) performance as PCS for real-time traffic without dropping any connection establishment requests (unlike in PCS). This goal can be attained 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 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 3 presents the architectural details of the MediaWorm router, and the Virtual Clock resource scheduling algorithm. Section 4 gives details on the simulation platform and the workload that is used in the evaluation. The performance results from the simulations are given in Section 5. Finally, Section 6 summarizes the results of this study and identifies directions for future research.

2 Related Work

With the building block of a multiprocessor interconnect being its router or switch fabric, there has been a considerable amount of research that has gone into efficient router designs. Routers from university projects like reliable router [8] and Chaos router [22], and commercial routers such as SGI SPIDER, Cray T3D/E, Tandem Servernet-II, IBM SP2 switch, and Myrinet [13, 27, 28, 14, 31, 2] use wormhole switching, while the HAL mercury [34] and Sun S-Connect [25] 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. A hybrid switch including both wormhole and VCT was designed in [29]. All these routers are primarily designed to minimize 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. Of these routers, ServerNet-II is the only one that provides a link arbitration policy (called ALU-biasing) for implementing some kind of 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 support for QoS 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 [20, 5]. Kim and Chien [21] propose a scheduling discipline, called rotating and combined queue (RCQ), to handle integrated traffic in a packet switched network. The Switcherland router [12], 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 [26] 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, proposed in [10, 3], also adheres to a hybrid approach, 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 should be noted that it needs one VC per connection. If we have a link bandwidth of 1.24 Gbps, with each multimedia stream requiring 4 Mbps, then it 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. Nevertheless, this is, perhaps, the most detailed study where router performance has been analyzed with both multimedia video streams and best-effort control traffic. A preemptive PCS network to support real-time traffic is also proposed in [1].

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 [23, 30, 16, 6, 19]. In many of these studies [23, 30, 19], the focus is on providing some mechanisms within the router to implement priority (for real-time traffic) and pre-emption (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 switching are explored in [16] 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 [6] on Myrinet), is not a scalable option. The third option of using VCs has not been investigated in depth in [16], where it has been cursorily examined in the context of indirect networks.

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 different traffic classes (and pay a high cost) like many 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 in an ad hoc manner as many of the above studies have done, this paper explores how a large number of multimedia connections can be supported (in a jitter-free manner) with wormhole switching in the presence of best-effort traffic.

3 Router Architectures

We assume familiarity with wormhole switched routers. Here we describe the architecture and modifications introduced in a basic pipelined wormhole router to enhance its performance for QoS guarantees. We then briefly describe PCS routers.

3.1 Pipelined Design

We use a pipelined router model, called PROUD [32, 33], and augment it with a bandwidth reservation algorithm. The pipelined model with five stages, as depicted in Figure 1, represents the recent trend in commercial designs such as SGI SPIDER [13] and SGI/Cray T3E [28].

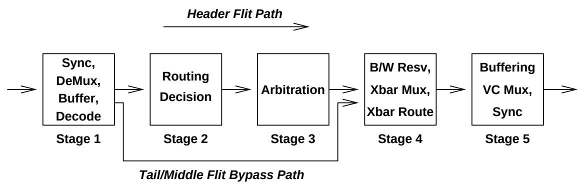

Figure 1. Pipeline model for the router

The first-stage of the pipeline represents the functional units which synchronize the incoming flits, demultiplex a flit so that it can go to the appropriate virtual channel buffer to be subsequently decoded. If the flit is a header flit, routing decision and arbitration for the correct crossbar output is 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 maybe (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 virtual channels 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 physical link (PC) has m VCs, a router could then be modeled as $(n \times m)$ parallel pipes. Resource contention amongst these pipes would occur for the crossbar output ports (and 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).

3.2 Crossbar Design Options

We consider two different crossbar design options — a full crossbar, and a multiplexed crossbar [7]. 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 router performance at a significantly high implementation cost; a multiplexed crossbar is cheaper to implement but requires more complex scheduling. Support for a very large 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.

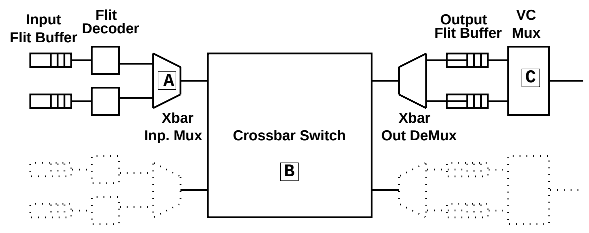

Figure 2. Functional units along a router pipe for a 2 port router with 2 VCs per PC. Additional functional units such as the routing decision block and the arbitration unit are not shown.

Our plan is to use this router architecture to support CBR, VBR and best-effort traffic. In order to allocate bandwidth for different traffic types, the message should carry the necessary information in its header flit to indicate its requirement to the router. This information is essentially the maximum tolerable slack. (i.e. inter-service completion time) for the flits of that message. We use a bandwidth reservation policy, known as the Virtual Clock algorithm [35], to provide soft bandwidth guarantees. This functionality is implemented in Stage 4 of the pipeline shown in Figure 1.

3.3 Virtual Clock

Virtual Clock [35] as originally proposed, is a rate-based scheduling algorithm for resource allocation in a connection-based network. It regulates the bandwidth usage of each connection by assigning a virtual clock value, which ticks at every packet/message arrival on a connection.

In the algorithm, there are two variables called `auxVC` (auxiliary VirtualClock) and `Vtick` for each connection. The value of these two variables are determined when the connection is set up. `auxVC` indicates the current virtual clock value of the connection, while `Vtick` is the amount of time by which `auxVC` should be incremented whenever a packet/message arrives on that connection. In fact, `Vtick` is the expected service interval (slack) of messages/packets on that connection (that is negotiated on startup). A smaller `Vtick` value means higher bandwidth requirement.

Once these two values are set, the Virtual Clock algorithm works as follows. For each connection i, when a message/packet arrives at the resource scheduler, the following two are computed:

$$\texttt{auxVC}_i \leftarrow \max(\texttt{Clock}, \texttt{auxVC}_i)$$

$$\texttt{auxVC}_i \leftarrow \texttt{auxVC}_i + \texttt{Vtick}_i$$

The new arrival is timestamped with $\texttt{auxVC}_i$, and the scheduler services the arrivals in increasing timestamp order.

In a wormhole router, there is no explicit connection setup between the source and the destination. Rather, each message requests its required bandwidth at each router on its way to the destination, and the router implements the virtual clock 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. For instance, if a message requires a bandwidth of 120K flits/sec, then its `Vtick` is set to 1/120K. The message makes its request by carrying its required `Vtick` in its header. When the tail flit leaves the router, its `Vtick` information in the router is discarded. The variable `Clock` in the above algorithm is the actual time (wall-clock time) at the time of flit arrival. For best-effort traffic, `Vtick` is set to infinity since it has the maximum slack.

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 Fig. 2. All these places are potential candidates where a Virtual Clock based bandwidth allocation can be performed. We rule out (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 also not a strong candidate. 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 Virtual Clock algorithm essentially behaves as a FIFO scheduler. Hence, we have chosen to implement the Virtual Clock scheduler at the crossbar input multiplexer (A).

In a router that implements a full crossbar, there is no crossbar input multiplexer (nor a demuliplexer at the crossbar output). Thus, the only contention points are for the crossbar output port (at the time of arbitration) and in the VC multiplexer. In such a router, we have chosen to implement the Virtual Clock algorithm at the VC multiplexer.

The hardware requirements for implementing a Virtual Clock based multiplexing schedule are as follows. For each virtual channel, two registers are required to store the `Vtick` (which is set by the header-flit) and `auxVC` values, a two input comparator for determining $\max(\texttt{auxVC}, \texttt{Clock})$ and an adder to increment `auxVC` by `Vtick`. Further, a multi-input comparator is needed to determine the VC with the lowest timestamp that wins multiplexing. Depending on the delay of this hardware compared to the router clock cycle, a flit-level or block-level multiplexing strategy can be adopted. Flit-level multiplexing is used in this study.

3.4 Interconnection Topologies – Fat Networks

Cluster interconnects are typically built with high degree switches. Myrinet [2] has 8 and 16 port routers, while Servernet-II [14] 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, the fat-tree and the fat-mesh [11]. Other cluster interconnects such as the tetrahedral topologies proposed by Horst [17] can also use “fat” links. Routers such as the Servernet-II [14] include hardware support for using multiple physical links connecting a pair of switches indistinguishably through the notion of “fat-pipes”.

Although most of the studies reported in this paper detail the performance of a single switch, we also analyze the performance of a fat mesh. The fat mesh used here is a 4-switch 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. 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.

3.5 Pipelined Circuit Switching

Pipelined Circuit Switching (PCS) [15] 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 either backtrack and try alternative paths if adaptive routing is permitted. If no path can be established or if adaptive routing is not permitted, a negative-acknowledgment is sent and the attempted connection is dropped. In this paper, we do not assume any adaptive routing capability[2]. Our intention is to compare the PCS scheme with wormhole switching with integrated traffic.

4 Experimental Platform

4.1 Simulation Testbed

The above architectural concepts have been extensively evaluated through simulation. We have developed wormhole and PCS router simulation models 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.

[2]Pipelined Circuit-Switching as originally proposed in [15], assumed non-minimal and adaptive routing capabilities with backtracking and re-routing. This leads to low connection dropping rates. However, if deterministic routing and no backtracking is assumed in a PCS router, this may lead to high connection dropping rates.

It should be noted that we are doing a detailed flit-level simulation with each stage of the router pipeline being modeled, together with several simultaneous streams established from each node in the system. 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 important output parameters 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 the destination. A $\bar{d}$ = 33 msec indicates a frame rate of 30 frames/sec at MPEG-2 rates. Coupled with a σ_d = 0, this implies jitter-free delivery. A higher $\bar{d}$ and/or σ_d implies jitters in transmission.

4.2 Workload

Our workload considers messages generated with VBR, CBR and best-effort requirements, and mixes of these as described below.

4.2.1 VBR and CBR Traffic

The VBR traffic is generated as a stream of messages between pairs of communicating (source-destination pair) processors. The traffic in each stream is based on MPEG-2 characteristics, i.e. frame size selected from a normal distribution with a mean of 16,666 bytes, standard deviation of 3333 bytes, and a inter-frame interval of 33 milliseconds (which gives a mean rate of 500 KBytes/s or 4 Mbps).

In the case of wormhole routing, a frame is broken down into fixed-size messages (except possibly the last message of a frame), with each message of the frame carrying the bandwidth requirements (`Vtick` information for the Virtual Clock algorithm) and the routing information in its header flit. As a result, the network treats/services each message of a stream independently 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. For instance, with a message size of 20 flits and 200 messages to a frame, the interval between successive message injections turns out to be 165 microsecond.

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, simultaneously informing the intermediate switches about its bandwidth requirements (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 wormhole switching, each message carries 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 processor 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 VBR traffic, with the exception that the frame size is kept constant (at 16,666 bytes) instead of using a normal distribution.

4.2.2 Best-Effort Traffic

The best-effort traffic is generated with a constant injection rate that is allocated to this class of traffic (explained in the next subsection). The message length is kept constant at 20 flits, 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.

4.2.3 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 mixed-mode 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 are possible from/to a node is limited by the number of VCs in the case of PCS. In wormhole switching, 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 (since the bandwidth requirement of each stream is 4 Mbps). 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.

5 Performance Results

In this section, we analyze the performance results for the 8-port single router with varying parameters as well as that of the fat mesh. The router parameters used in this performance study are given in Table 1.

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

Table 1. Simulation Parameters

5.1 Virtual Clock vs. FIFO

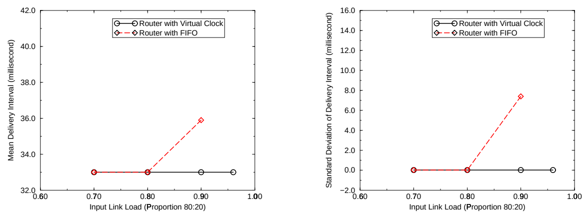

Figure 3. Virtual Clock vs. FIFO (16 VCs)

We first begin by examining how a traditional wormhole router (which uses FIFO scheduling) performs with multimedia/mixed traffic. Fig. 3 shows the mean delivery interval ($\bar{d}$) and its standard deviation (σ_d) for this router with a mixture of 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, if we change the multiplexer scheduling (from FIFO) to a virtual clock algorithm at point A in Fig. 2, the resulting router 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 administer the available bandwidth in the wormhole switched router.

5.2 CBR and VBR Traffic Results

Figure 4 depicts the $\bar{d}$ and σ_d results with only CBR and only VBR traffic (there is no best-effort traffic). It can

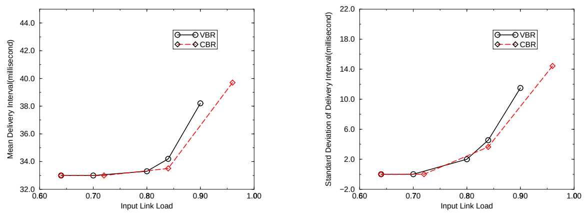

Figure 4. Comparison of CBR and VBR traffic (16 VCs, 400 Mbps links)

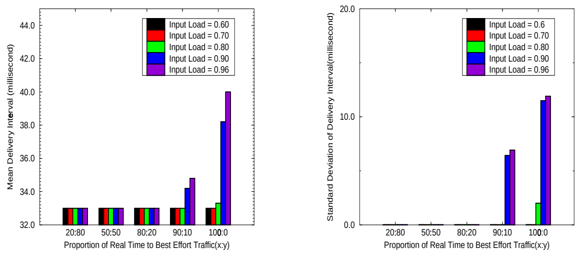

Figure 5. Mixed Traffic (16 VCs)

be gleaned that both exhibit nearly identical performance, with the CBR traffic experiencing jitter-free performance for 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 focus on VBR streams in the rest of the studies in this paper.

5.3 Results with Mixed Traffic

Next, we vary the ratio of real-time (only VBR) and best-effort traffic for different input loads, and study the effect on jitter for VBR and average latency for the best-effort traffic. Fig. 5 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 mix between these two traffic classes. 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 2. 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 Virtual Clock algorithm to the real-time traffic.

5.4 Impact of VCs and Crossbar Capabilities

It should be noted that our workload generates multiple connections on 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 6 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 a large amount of resources in the router. Lower number of VCs, on the other hand, allows us to be able to use a full crossbar (instead of a multiplexed one). 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.

	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.

Table 2. Average Latency for Best Effort Traffic (8×8 switch, 16VCs, 400Mbps links)

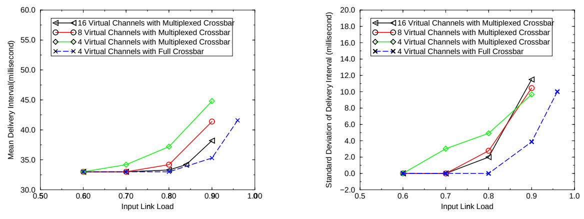

Figure 6. Impact of VCs and Crossbar Capabilities (400 Mbps links, $x : y$ = 100:0)

5.5 Effect of Message Size on Jitter

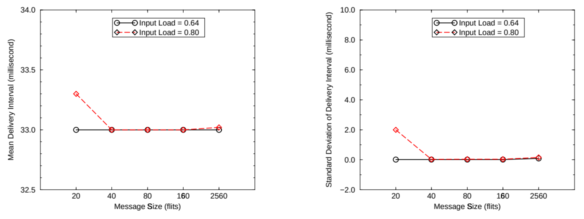

Figure 7. Effect of message size on jitter (16 VCs)

Our next experiment examines the impact of message size on VBR traffic. We vary message size for two different input loads (0.64 and 0.8) that are representative of the behavior observed earlier, and examine changes in $\bar{d}$ and σ_d. The results in Fig. 7 show that except for very small message sizes, there is little impact on QoS for real-time traffic. For very small sizes, the effect of the header flit overhead becomes noticeable. For instance, 1 header flit in a mes-

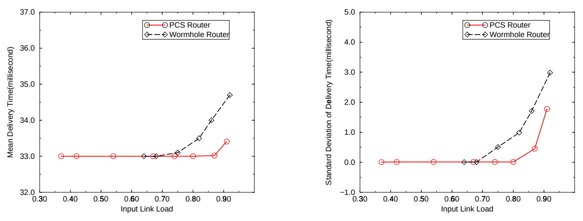

Figure 8. Comparison of MediaWorm with a PCS router (8×8 switch, 100 Mbps, 24 VCs.)

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

Table 3. Number of attempted, established and dropped connections in a PCS router.

sage size of 20 flits consumes 5% of the stream bandwidth. These results show that we do not really need to go for large messages for multimedia traffic. In fact, smaller sizes may help the latency for best-effort traffic. However, this needs further investigation to make conclusive pronouncements.

5.6 Wormhole and PCS Comparison

PCS is expected to provide good performance for VBR traffic. This is because it 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.

In this experiment, we compare the performance of MediaWorm router to that of an (8×8) 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 a PCS router.

As can be expected, wormhole routing 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 a wormhole 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 PCS compared to that for the wormhole router.

While PCS 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 PCS is shown in Table 3.

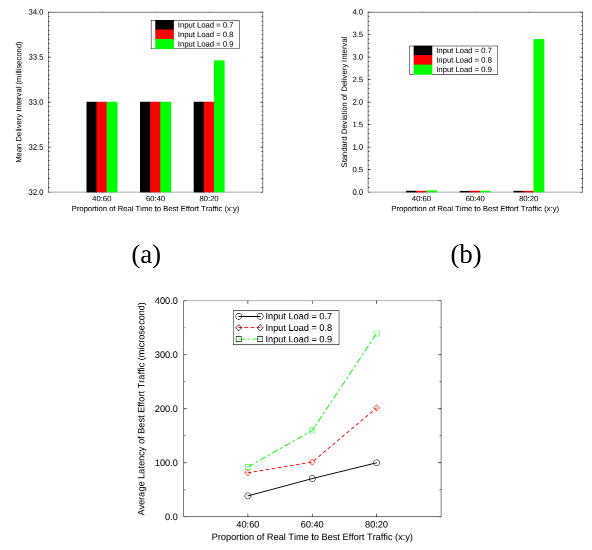

Figure 9. Performance of a (2×2) fat mesh.

These results show that for most realistic operating conditions (an input load of 0.7 is reasonably high), wormhole switching can deliver as good (jitter-free) performance as PCS for real-time traffic, while not turning down connection establishment requests as done in PCS (around 60% of requests are turned down at a load of 0.7).

5.7 Fat-Mesh Results

In this subsection, we examine the performance implications of using MediaWorm routers in a fat-mesh interconnect. 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 the additional points of resource contention in a network. We limit this study to a modest 4 node (2×2) network due to limited simulation resources.

Fig 9 (a) and (b) shows the change in mean delivery interval and the corresponding standard deviation for 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 Fig. 9 (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 Figs. 5 (a) and (b) with Figs. 9 (a) and (b). For example, with a load of 0.9 and a traffic mix of 80:20, we can observe that a single switch is able to 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.

6 Concluding Remarks

Widespread use of clustered 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 quality of service 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 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 Virtual Clock [35], instead of the common *rate agnostic* schedulers such as FIFO or round-robin.

We have studied the capabilities of MediaWorm in supporting real-time and best-effort traffic. Our study makes the following important conclusions.

- We confirm that Virtual Clock can provide considerably improved performance for traffic that require soft real-time guarantees (VBR/CBR).

- Results on the effect of mixed VBR and best-effort traffic on MediaWorm show that there is no adverse effect on performance of VBR traffic in the presence of best-effort traffic. However, as the share of VBR traffic increases at 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 physical channel bandwidth.

- Although the performance of a PCS router is slightly better than 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 cluster sizes directly from our present results, we expect that clusters designed with appropriate bandwidth balance amongst various links by the use of fat-topologies and MediaWorm-like switches may 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. 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.

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 [30].

- A scalability study to large network sizes 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] S. Balakrishnan and F. Ozguner. A Priority-Based Flow Control Mechanism to Support Real-Time Traffic in Pipelined Direct Networks. In *Proc. Intl. Conf. on Parallel Processing*, volume I, pages 120–127. IEEE CS Press, August 1996.

[2] 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*, 15(1):29–36, February 1995.

[3] 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. In *Network Based Parallel Computing : Proc. Third Intl. Workshop on Communication, Architecture and Applications on Network Based Parallel Computing (CANPC99)*, Lecture Notes in Computer Science, pages 62–76. Springer, January 1999.

[4] J. Carbonaro and F. Verhoorn. Cavallino: The Teraflops Router and NIC. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 157–160, August 1996.

[5] A. A. Chien and J. H. Kim. Approaches to Quality of Service in High-Performance Networks. In *Proc. Parallel Computer Routing and Communications Workshop.* Lecture Notes in Computer Science, Springer-Verlag, July 1997.

[6] K. Connelly and A. A. Chien. FM-QoS: Real-time Communication using Self-synchronizing Schedules. In *Proc. of SC97: High Performance Networking and Computing*, November 1997.

[7] W. J. Dally. Virtual-Channel Flow Control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, May 1992.

[8] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. Arhitecture and Implementation of the Reliable Router. In *Proc. of Hot Interconnects II*, Stanford University, Palo Alto, CA, August 1994.

[9] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Trans. on Computers*, C–36(5):547–553, May 1987.

[10] J. Duato, S. Yalamanchili, M. B. Caminero, and F. J. Quiles. MMR: A High Performance Multimedia Router Architecture and Design Tradeoffs. In *Proc. Intl. Symp. on High Performance Computer Architecture (HPCA-5)*, January 1999.

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

[12] H. Eberle and E. Oertli. Switcherland: A QoS Communication Architecture for Workstation Clusters. In *Proceedings of the 25th International Symposium on Computer Architecture*, pages 98–108, July 1998.

[13] M. Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing : The SGI SPIDER Chip. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 141–146, August 1996.

[14] D. Garcia and W. Watson. Servernet II. In *Proc. of the 1997 Parallel Computing, Routing, and Communication Workshop (PCRCW'97)*, June 1997.

[15] P. T. Gaughan and S. Yalamanchili. A Family of Fault Tolerant Routing Protocols for Direct Multiprocessor Networks. *IEEE Trans. on Parallel and Distributed Systems*, pages 482–497, May 1995.

[16] 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 *Proc. Intl. Conference on Network Protocols*, October 1996.

[17] R. W. Horst. TNet: A Reliable System Area Network. *IEEE Micro*, pages 37–45, February 1995.

[18] P. Kermani and L. Kleinrock. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks*, 1979.

[19] B. Kim, J. Kim, S. Hong, and S. Lee. A Real-Time Communication Method for Wormhole Switching Networks. In *Proc. Intl. Conference on Parallel Processing*, pages 527–534, August 1998.

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

[21] J. H. Kim and A. A. Chien. Rotating Combined Queueing (RCQ): Bandwidth and Latency Gurantees in Low-Cost, High-Performance Networks. In *Proc. Intl. Symp. on Computer Architecture*, pages 226–236, May 1996.

[22] S. Konstantinidou and L. Snyder. The Chaos Router. *IEEE Trans. on Computers*, 43(12):1386–1397, December 1994.

[23] J.-P. Li and M. Mutka. Priority Based Real-Time Communication for Large Scale Wormhole Networks. In *Proc. Intl. Parallel Processing Symposium*, pages 433–438, May 1994.

[24] M. D. May. The Next Generation Transputers and Beyond. In *Proc. 2nd European Distrinuted Memory Conference*, pages 7–22, April 1991.

[25] A. G. Nowatzyk, M. C. Browne, E. J. Kelly, and M. Parkin. S-Connect: from Network of Workstations to Supercomputer Performance. In *Proc. of the 22nd Annual International Symposium on Computer Architecture*, pages 71–82, June 1995.

[26] J. Rexford, J. Hall, and K. G. Shin. A Router Architecture for Real-Time Point-toPoint Networks. In *Proc. Intl. Symp. on Computer Architecture*, pages 237–246, May 1996.

[27] S. L. Scott and G. M. Thorson. Optimized Routing in the Cray T3D. In *Proc. Parallel Computer Routing and Communications Workshop (PCRCW)*, pages 281–294. Springer Verlag Lecture Notes in Computer Science, May 1994.

[28] S. L. Scott and G. M. Thorson. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. In *Proc. Symp. High Performance Interconnects (Hot Interconnects 4)*, pages 147–156, August 1996.

[29] K. G. Shin and S. W. Daniel. Analysis and Implementation of Hybrid Switching. In *Proc. Intl. Symp. on Computer Architecture*, pages 211–219, 1995.

[30] H. Song, B. Kwon, and H. Yoon. Throttle and Prempt: A New Flow Control for Real-Time Communications in Wormhole Networks. In *Proc. Intl. Conf. on Parallel Processing*, pages 198–202. IEEE CS Press, August 1997.

[31] 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 SP-2 High-Performance Switch. *IBM Systems Journal*, 34(2):185–204, 1995.

[32] A. S. Vaidya, C. R. Das, and A. Sivasubramaniam. A Testbed for Evaluation of Fault-Tolerant Routing in Multiprocessor Interconnection Networks. *IEEE Trans. on Parallel and Distributed Systems*, 1999. (To appear in the Special Issue on Challenges in Designing Fault-Tolerant routing in Networks.).

[33] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das. LAPSES: A Recipe for Adaptive Router Design. In *Proc. Fifth IEEE Intl. Symp. on High Performance Computer Architecture*, pages 236–243, January 1999.

[34] 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 *Proc. International Symposium on Computer Architecture*, pages 98–107. ACM, 1997.

[35] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks. *ACM Trans. on Computer Systems*, 9(2):101–124, May 1991.