QoS Provisioning in Clusters: An Investigation of Router and NIC Design [*]

Ki Hwan Yum Eun Jung Kim Chita R. Das
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {yum,ejkim,das}@cse.psu.edu

Abstract

Design of high performance cluster networks (routers) with Quality-of-Service (QoS) guarantees is becoming increasingly important to support a variety of multimedia applications, many of which have real-time constraints. Most commercial routers, which are based on the wormhole-switching paradigm, can deliver high performance, but lack QoS provisioning. In this paper, we present a pipelined wormhole router architecture that can provide high and predictable performance for integrated traffic in clusters. We consider two different implementations—a non-preemptive model and a more aggressive preemptive model. We also present the design of a network interface card (NIC) based on the Virtual Interface Architecture (VIA) design paradigm to support QoS in the NIC. The QoS capable router and NIC designs are evaluated with a mixed workload consisting of best-effort traffic, multimedia streams, and control traffic.

Simulation results of an 8-port router and a (2×2) mesh network indicate that the preemptive router can provide better performance than the non-preemptive router for dynamically changing workloads. Co-evaluation of the QoS-aware NIC with the proposed router models shows significant performance improvement compared to that with a traditional NIC without any QoS support.

Index Terms: Cluster Network, Network Interface, Preemption Mechanism, Quality-of-Service, Router Architecture, VirtualClock, Wormhole Router.

1 Introduction

Cluster systems are becoming increasingly more attractive for designing scalable servers with switched network architectures that offer much higher bandwidth than the broadcast-based networks. Quality-of-Service (QoS) provisioning in such clusters is becoming a critical issue with the widespread use of these systems in diverse commercial applications [5]. The traditional best-effort service model is not adequate to support many cluster applications with varying consumer expectations. For example, many web servers and database servers make efficient use of clustering technology from cost, scalability, and availability standpoints. However, the tremendous surge in dynamic web contents, multimedia objects, e-commerce, and other web-enabled applications requires QoS guarantees in different connotations. The guaranteed communication delay and bandwidth requirements of the applications mandate that the cluster interconnect should be able to handle these traffic demands. These demands in turn are passed on to the building blocks of the interconnects, the switching fabrics or routers. Hence, it has become crucial to revisit the design of router architectures to provide high and predictable performance.

Typically two classes of traffic are generated with mixed or integrated workloads. These are best-effort traffic and real-time traffic. While best-effort traffic usually does not have any stringent performance requirements (hence known as available bit rate (ABR)), real-time traffic are further classified as constant bit rate (CBR) and variable bit rate (VBR) workloads. A cluster network should therefore support ABR, CBR and VBR traffic effectively.

A cluster interconnect designed using currently available commercial routers such as Myricom Myrinet [2], SGI SPIDER [13], IBM SP2 [25], and Tandem Servernet [14] can efficiently handle best-effort traffic, but not real-time traffic. On the other hand, a cluster system designed with packet-switched ATM routers can support real-time traffic, but incurs high latency for best-effort traffic. Since none of the existing routers can efficiently support both traffic classes in clusters, the primary motivation of this work is to design such a router and explore various design trade-offs in the context of wormhole switching paradigm.

QoS provisioning in networks can be achieved by prudent management of network resources — mainly the link bandwidth and buffers. In order to share the link bandwidth among different applications, a scheduler should be able to recognize the bandwidth requirements of the competing requests and allocate the bandwidth accordingly. Scheduling techniques such as Fair Queueing [9] and VirtualClock [31] have been proposed for such proportional bandwidth allocation in packet-switched networks. The first attempt to use such a rate-based scheduling mechanism was proposed in

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

the MediaWorm router design [30]. The MediaWorm uses the VirtualClock algorithm for scheduling of virtual channels (VCs)[1] to share the link bandwidth. The VCs are divided statically into two groups, one for best-effort traffic and the other for real-time traffic. It was shown that by using a rate-based scheduling mechanism, it is possible to provide soft guarantee for media streams in wormhole routers.

The MediaWorm router design has several limitations. First, fixed allocation of VCs to best-effort and real-time traffic may not be the best choice for changing workloads. Second, it does not have any preemption capability that is necessary to transfer a higher priority message without being blocked by a lower priority message. Third, the router has not been tested exhaustively with realistic and dynamically changing workloads. Next, the performance evaluation was limited to only the router design. It is known that the network interface (NI) plays a crucial role in reducing the communication overhead. The role of the NI may become even more important to satisfy the QoS requirements. Several user-level communication mechanisms have been proposed recently, where an application can directly communicate with an intelligent NI with minimal kernel support [1]. The virtual interface architecture (VIA) [11, 28] framework is becoming a standard to design user-level communication protocols in NICs. However, it is not clear how QoS provisioning should be provided in the context of a VIA design. In addition, co-evaluation of the cluster router/interconnect with a VIA-style NIC is essential to understand the interplay of different designs on the overall performance of the communication architecture.

To our knowledge, none of the prior work has considered the above research issues in the design and evaluation of QoS capable cluster interconnects. In particular, co-evaluation of the interconnect and the NI to handle QoS sensitive traffic has not been undertaken. This paper presents the design and evaluation of a QoS-aware wormhole router and a compatible VIA-style NIC to handle mixed traffic in clusters. The main contributions of the paper are the following:

- We analyze two design alternatives for a pipelined, wormhole-switched router. These are called the non-preemptive and the preemptive models. Unlike the non-preemptive model, the preemptive model can dynamically allocate any virtual channel to any traffic class. This brings in the necessity that a higher priority message should be able to preempt a lower priority message. Hence, the router core includes a flit-level (or block-level) preemption mechanism. In addition, we propose an acceleration mechanism for faster preemption of lower class traffic to boost performance further.

- We consider two types of rate-based scheduling schemes, known as Fair Queuing [9] and VirtualClock [31], to schedule the VCs for satisfying QoS requirements. Design and performance implications of the two schemes are considered prior to selecting the VirtualClock algorithm for the rest of the design.

- We present a modified VIA design to handle real-time traffic in the NIC. Three design modifications in the VIA implementation are proposed. These include a prioritized doorbell structure in the NIC to support different traffic classes, a VC-aware buffer management in the NIC, and the VirtualClock algorithm to implement rate-based scheduling.

- The QoS-aware NIC and the router designs are integrated to evaluate the entire communication substrate for an end-to-end performance analysis.

[1]The VC concept was proposed by Dally to enhance network performance [8].

We use a mixed workload consisting of three types of traffic — short control messages, best-effort traffic, and MPEG-2 video streams. We conduct an in-depth analysis of the cluster interconnect design using average message latency, deadline missing probability and average deadline missing time of MPEG-2 frames as the performance metrics. The first parameter quantifies performance implications for the best-effort traffic and control traffic, while the other two parameters are indicators of real-time traffic behavior. First, we compare two design alternatives for the router; the non-preemptive model with static allocation of VCs, and the preemptive model with dynamic allocation of VCs. We also examine the traditional router that has no QoS support. Next, two NI designs are considered. One is a VIA-based NI without any QoS provisioning (called the traditional NIC in our study), and the second is a QoS-aware NIC. Then the router is evaluated in conjunction with the two NIC designs to estimate the overall performance.

The simulation results indicate that although the preemptive model increases the design complexity, it can provide better performance than that of the non-preemptive router for dynamic workloads since the number of VCs allocated to different traffic classes can be controlled on the fly. These improvements become more pronounced with higher network load in the case of a single router as well as in a 2-D mesh network. With the suggested modifications to the VIA design, the modified NIC shows significantly better performance compared to the traditional NIC. Co-evaluation of the proposed routers with the QoS-aware NIC reveals the significance of performance predictability in the NIC for transferring the performance benefits of the router to the application level.

The rest of the paper is organized as follows. Section 2 summarizes related work. In Section 3, the router architecture, the rate-based scheduling schemes, and the VIA design are discussed. In Section 4, we discuss the experimental platform. The performance results are presented in Section 5, followed by the concluding remarks in Section 6.

2 Related Work

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 [5]. ServerNet II [14] is the only commercial router that uses a link arbitration policy (called ALU-biasing) for implementing bandwidth and delay control. But this simple mechanism is not sufficient to support media streams. The InfiniBand Architecture (IBA) initiative, aimed at SAN/cluster systems, is currently exploring QoS provisioning issues [21]. Kim and Chien [16] proposed 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 ATM. The router architecture proposed in [22] uses a hybrid approach, wherein wormhole switching is used for best-effort traffic and packet switching is used for time-constrained traffic.

The multimedia router architecture, proposed in [10, 4], also uses a hybrid approach. It uses 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. A connection-oriented scheme like PCS needs one VC per connection, and therefore, may not be practical to provide a large number of VCs per physical channel(PC).

A handful of research efforts have examined the possibility of using wormhole-switched routers/networks for real-time traffic [18, 7, 15]. While [18, 15] investigated hardware support required in a router to facilitate real-time message transfer, the work in [7] considered a software oriented synchronization mechanism in the Myrinet switch. The hardware mechanisms are not sufficient (and may not even be necessary) for providing required performance support for integrated traffic. The software approach in [7] may not be scalable. As stated in the introduction, the MediaWorm router [30] proposed recently uses a rate-based scheduling algorithm, known as VirtualClock [31], to assign proportionate bandwidth based on the application demands.

Message preemption in wormhole routers have been addressed in [23, 17]. In [17], lower priority messages that block higher priority messages are discarded to allow faster delivery of higher priority messages. This approach has the advantage that it does not require extra resources to store routing information of the preempted messages. But preempted messages are lost and thus may not be a viable option for many applications. With additional hardware and flow control, it is possible to recover the lower priority messages. Song *et al.* [23], on the other hand, preempt a lower priority message in favor of a higher priority messages using additional buffers. In their scheme, the router has $(s-1)$ extra input buffers, where s is the number of priority levels it supports. By providing these additional input buffers, the router can always establish a free path for higher priority messages. This scheme requires a history stack for storing the header information of the preempted messages in ascending order of their priorities for each output channel. Unlike our pipelined router model, the authors use a lumped router design. Hence, many architectural details that are required to support flit-level preemption are not addressed in their work. Provisioning for preemption in different stages of the pipeline is much more complex than a single stage (lumped) router model.

Also, to our knowledge, there is no related work on QoS capable NICs and on co-evaluation of QoS-aware routers and NICs. In particular, our design includes a VIA-style NIC that is QoS capable.

3 Router Architecture

Most routers now use a pipelined design to minimize the network cycle time. Accordingly, we use a pipelined, wormhole-switched router similar to the SGI SPIDER and MediaWorm [30] in this paper. Figure 1 shows the five-stage pipelined router with n ports. The first stage of the pipeline represents the functional units which synchronize the incoming flits, demultiplex a flit so that it can go to one of the C virtual channels (VCs) 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), while middle flits and the tail flit of a message bypass stages 2 and 3, and directly move to stage 4. Flits get routed to the correct crossbar output ports in stage 4. As shown in the figure, the router has a scheduler (multiplexer), residing at the input port of the crossbar. In the best-effort router model, the scheduler can select one of the C VCs using the FCFS or round robin (RR) principle. For QoS provisioning, we replace this scheduler with a rate-based scheme as described next. Finally, the last stage of the router performs buffering of flits flowing out of the crossbar, multiplexes the physical channel bandwidth amongst the C VCs, and carries out synchronization with input buffers of other routers or the network interface for the subsequent transfer of flits.

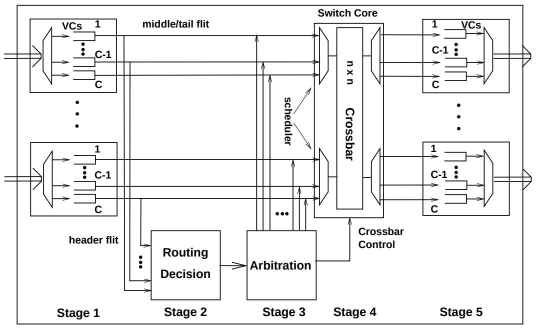


Figure 1. A five-stage pipelined router model

3.1 Rate-based Scheduling for QoS Support

For this study we consider two different work conserving, rate-based schedulers: Fair Queueing [9], and VirtualClock [31]. (Many variations of these two schemes and other rate-based algorithms like the Weighted Round Robin have been proposed for QoS support in packet-switched networks. All these schemes provide almost similar performance. The motivation of this paper is to show that QoS in clusters can be provided by using a simple rate-based scheduling algorithm.) It has been shown that the VirtualClock algorithm cannot handle bursty traffic effectively without any input regulation [24]. In this study, although real-time traffic can exhibit burstiness, the NI regulates it by injecting one frame every 33.3 msec into the router. Therefore, traffic burstiness is avoided to affect the VirtualClock performance.

We have implemented both these schemes at the crossbar input of stage 4. In order to select one scheme for the rest of the design, 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 (SD) of delivery time for the media streams. The results with different input loads are quite similar in both cases as depicted in Table 1. However, since implementation of the Fair Queueing is more complex for maintaining the round robin number, we use the VirtualClock algorithm in the rest of our design.

Load	Inter-frame time(*msec*)/SD	
	VirtualClock	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

Table 1. Comparison of VirtualClock and Fair Queueing algorithms (Ratio of real-time to best-effort traffic is 80:20.)

In the Virtual Clock algorithm, there are two variables, called *auxVC* and *Vtick* for each connection. The values of these two variables are determined when a connection is set up. The *auxVC* indicates the virtual clock value of the connection, while the *Vtick* is the amount of time that should be incremented whenever a flit arrives at that connection. It is important to estimate the counter size required to store the *Vtick* and *auxVC* values. We have developed a VHDL simulator to implement the VirtualClock algorithm in hardware using finite size counters. If the $auxVC_i$ for some VC i overflows during computation, then that $auxVC_i$ is set to 0 and the $auxVC_j$ becomes ($auxVC_j - auxVC_i$) for all $j \neq i$. If the new $auxVC_j$ becomes less than 0, then it is set to 0. The only limitation is that the counter size should be large enough to store the largest possible *Vtick* value. From our simulator and VHDL implementation, we observed that 16-bit counters are adequate to store the *auxVC* and the *Vtick* values.

3.2 Router Design Alternatives

We consider two router models to facilitate QoS support. In the first design, like the MediaWorm Router [30], we statically divide the input and output VCs among the traffic classes. A traffic of class c can only use the VCs assigned to it. We call it as the non-preemptive model since there is no sharing of VCs, and hence, no preemption mechanism is necessary. The VC assignment is done at the system configuration time and cannot be changed during execution. Therefore, the non-preemptive model is not flexible.

A solution to this problem is to develop a preemptive model, where several classes of traffic with different priorities can share the same VC, with the provision that a higher priority message can preempt a lower priority message. The preemptive model can dynamically allocate any VC to any traffic class. Hence, it is more suitable to handle dynamically varying workloads. Preemption occurs when the header flit of a higher priority message arrives at a resource, which is being held by a lower priority message. Specifically, we examine blocking and preemption at the input buffer (VC) of the router.

3.2.1 Preemption in the Input Buffer

The additional hardware required for preemption at any input buffer (VC) include an extra buffer of size $(s-1)$ where s is the total number of priority levels, and a history stack of the same size. The extra input buffer is used for diverting higher priority messages when the regular VC is occupied by a lower priority message. If the input buffer is occupied by a higher priority message, a lower priority message is not allowed to use the extra buffer, and it is blocked behind the higher priority message. On the other hand, if the input buffer is used by a lower priority message, a higher priority message is sent to the extra buffer so that it can subsequently preempt the lower priority message in stage 1 of the router. Similar to [23], the routing information of a lower priority message is stored in the history stack for forwarding it later.

In stage 1, when the extra buffer has a header flit from a higher priority message, the input buffer preemption process begins. The router first checks whether the tail flit of the lower priority message has passed through the stage 1 decoder. If not, a dummy tail flit is created for the preempted message. A dummy tail flit does not carry any payload, but behaves as a regular tail flit to release all the resources held by the message. Otherwise, the resources are reserved and cannot be used by any other message. For example, in Figure 2 (a), when the higher priority message m3 interrupts the lower priority message m1, the dummy tail of m1 is generated. Then the routing information of m1 is stored in the history stack to be used later for making a dummy header for the retransmission of m1. Note that no dummy header is required if no dummy tail was sent. During preemption, the remaining flits of m1 and any other lower priority messages are blocked in the input buffer.

Next, all the flits of m3 in the extra buffer are sent through the router. After that, if the extra buffer is empty, transmission of remaining flits of m1 resumes from the regular input buffer.

3.2.2 A Flit Acceleration Mechanism

When the input buffer preemption starts, there could be remaining flits of m1 between the flit decoder buffer and the input port of the crossbar as shown in Figure 2 (a)[2]. In addition, when the header flit of m3 tries to reserve the output

[2]At best there could be 3 flits. A header flit and a middle flit of m1 at two different stages and a tail flit of another message at the crossbar input.

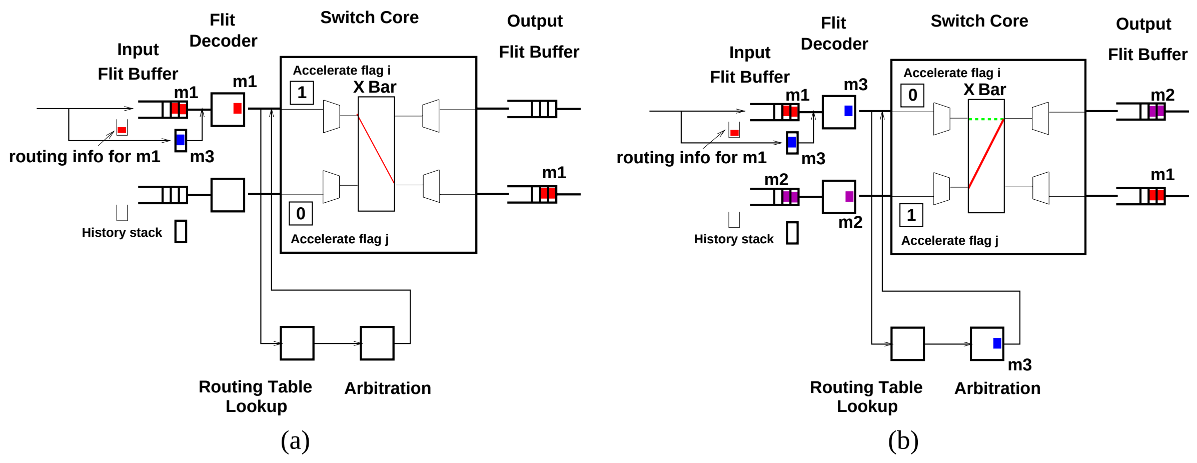


Figure 2. Preemption and acceleration mechanisms in the router. (a) A higher priority message (m3) is blocked and needs to preempt a lower priority message (m1) in the input buffer using extra buffer. (b) When the header flit of m3 tries to reserve the output VC in the arbitration stage, the output VC is already occupied by another lower priority message (m2). Acceleration flag for m2 is set so that the remaining flits of m2 are sent faster to the output VC before m3 can start. This is required to avoid interleaving of m2 and m3 flits in the output buffer.

VC, it could be already occupied by another lower priority message like m2 in Figure 2 (b). In both cases, the flits of lower priority messages (m1, m2) will slow down the progress of m3, until these flits are pushed to the output buffer.

Therefore, we use a flit acceleration mechanism that helps expedite the delivery of flits of such lower priority messages (like m1 and m2) by assigning a specific low virtual clock value to them. This value guarantees that these messages will be selected first at the next cycle of the scheduler unless there are other preempted messages at other VCs. (Then we can select them in a RR fashion.) For this purpose, a flag, called *Accelerate*, is associated with each input VC. The *Accelerate* flag is set until the tail flit of the preempted message (m1) or expedited message (m2) passes through the crossbar.

The other option to handle blocking at other stages is to use the preemption mechanism. The acceleration mechanism is much simpler and easier to control than providing a separate preemptive path at such stages.

3.3 NIC Architecture

The network interface (NI) has a crucial role in the overall communication performance since it is responsible for initiating and responding to communications, for handling data movement, and for providing application isolation. Since improving the performance of the router/interconnect alone will shift the communication bottleneck to the NI, design of faster NIs has become a major research thrust recently. Consequently, a few user-level messaging layers such as Active Messages [27], U-Net [26] and FM [20] have been proposed to minimize the role of the operating system involvement in communication. A generic communication layer, called Virtual Interface Architecture (VIA), has been introduced as a standard communication paradigm for System Area Networks (or SANs) or clusters [3, 6]. The design focus of the VIA is to provide an efficient communication protocol between a user process and the network interface (NI).

VIA is a connection oriented paradigm consisting of Virtual Interfaces (VIs). A VI is the mechanism by which applications talk to the NIC hardware, and establish a connection between the two processes. A VI consists of two queues: a *send queue* and a *receive queue*. For sending a message, an application posts a descriptor in the send queue, and informs the NI of the pending request by ringing a *send doorbell*, which is a memory mapped region on the NI. On receiving the doorbell, the NI transfers the descriptor and the data from the user memory to the NI buffers using two DMAs. The NI transfers the message to the wire using another DMA, and updates the status field of the send descriptor or that of a *completion queue*. The actions on the receive are very similar to that of a send. The application creates an empty buffer, posts a descriptor in the receive queue and rings the *receive doorbell* in the NIC buffer. When a message arrives for a VI, the NI transfers the message to the buffer allocated by the application and updates the status field of the receive descriptor. The message is subsequently consumed by the receiving process.

However, the original VIA framework does not have any QoS design specification. Here, we propose an extension of the VIA design to support different priority classes in the NIC.

3.3.1 A QoS-aware NIC Design

We propose three design modifications in the original VIA framework as described below. These are a prioritized doorbell structure to support different traffic classes, a virtual channel aware buffer management in the NIC, and a hardware supported VirtualClock scheduler to transfer flits to the router. Figure 3 shows the different stages in the flow of data from an applications to the NIC. Each application such as a video source or a a best-effort process has a VI with the corresponding send and receive queues. The send and receive queues reside in the user memory. To support integrated traffic, we implemented prioritized doorbells in the NIC, where there is a doorbell queue for each class. The NIC firmware picks up the doorbells in FCFS order based on their priority and programs the host DMA engine to transfer the descriptor followed by the message. Head of blocking in the doorbell queues can be avoided by several methods. As an example, if the NIC buffer (virtual channel) corresponding to a doorbell is full, the scheduler can pick up the next doorbell in the queue.

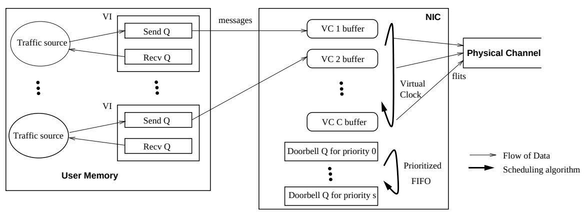


Figure 3. A VIA-style NIC with QoS support

To make the NIC design compatible to the QoS-aware router of the previous section, we implemented an equal number of VCs (C) to enable virtual channel flow control in the NIC. Note that this is a logical separation of the NIC local memory. As messages are transferred into the NIC by the host DMA, they are broken into *flits* by the NIC processor. The NIC buffer behaves as FCFS queues for the different VCs. In the original VIA implementation, the send DMA engine of the NI (for example in the Myrinet network card) is used to transfer a complete message into the network at the rate of one flit per cycle. On the other hand, the router model discussed in this paper (and also in most prior designs) employs a flit-level multiplexing. (We will see in the performance section that flit-level multiplexing is a better choice for QoS support.) Thus, we experimented with two design alternatives for transferring data from the NIC to the wire.

In the first case, in accordance with the router design, we considered a flit-level multiplexing and data transfer to the network. This design will considerably slow down the rate at which the NIC can push data to the network when the DMA overhead is taken into consideration. To avoid the DMA overhead, transfer of the flits to the network needs to be implemented in hardware. We, therefore, considered a hardware-implemented VirtualClock algorithm for transferring flits from different VCs in the NIC buffer. Thus, the flit-level multiplexing has no DMA overhead. In the second case, we adhered to the traditional message-level granularity. However, the messages are selected using the VirtualClock algorithm. This implementation includes the DMA overhead for each message transfer. These two design alternatives were used in analyzing the overall performance implications.

4 Experimental Platform

4.1 Simulation Testbed

For evaluating the architectural concepts, we have developed flit-level simulation models for the traditional router (TR) with FCFS scheduling, the non-preemptive router (NP), the preemptive router (P), and the VIA-style NIC using CSIM. The simulation models are flexible in that one can specify the number of physical channels (PCs), number of VCs per PC, link bandwidth, VBR rates and variation of VBR rate, flit size, message size, and many other architectural and workload parameters. It is also possible to configure any network topology using these routers. The NIC simulator and the router simulator are written with common interface for ease of integration.

For our experiment, we have simulated an 8-port router and a 2-D mesh network using 5-port routers. We used 16 VCs per PC. (It is possible to use less number of VCs per PC, but the performance of real-time traffic degrades if the number of VCs is very small.) The flit size is 128 bits, and each message consists of 40 flits except for the control messages, which are 20-flit long. Physical link bandwidth is 1.6Gbps, and flit buffers are 40-flit deep.

The simulation of the NIC is based on a *Baseline* NI that is similar to Myricom's M3M-PCI64B 64bit, 66MHz SAN/PCI Interface Myrinet network card [19]. The programmable nature of the Lanai-based network card facilitated to emulate the VIA features.

4.2 Workload

Our workload includes messages from real-time VBR traffic, best-effort traffic, and control traffic. The VBR traffic is generated as a stream of messages between a pair of communicating (source-destination) processors. The traffic in each stream is generated from seven real MPEG-2 video traces with different bandwidth requirements [4, 29]. Each stream generates 30 frames/sec, and each frame is fragmented into 40-flit size messages (except possibly the last message of a frame), with each message carrying the bandwidth requirements (*Vtick* information for the VirtualClock algorithm), and the routing information in its header flit.

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 assigned from among the available VCs using a uniform distribution.

The best-effort traffic is generated with a given injection rate λ, and follows the Poisson distribution[3]. Best-effort messages are assumed 40-flit long, and a message destination is assumed to follow a uniform distribution. The input and output VCs for a message are assigning using a uniform distribution of the available VCs for this traffic class.

Control traffic is typically used for network configuration, congestion control, and transfer of other control information. This traffic has the highest priority in our model.

[3]We have also simulated self-similar traffic to capture traffic burstiness [29]. The results are not included here due to lack of space.

Therefore, control traffic can preempt both best-effort traffic and real-time traffic. The generation rate of control traffic is assumed to be low (one message per 33.3 msec), and only one virtual channel is assigned for this kind of traffic. For the non-preemptive router, this VC should be determined when the router is configured, while the preemptive router can use any one of its VC for this traffic, and can even change to another VC later. In our study, we assigned VC 0 for both the models for simplicity. We also assume a uniform destination distribution for the control traffic.

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 input load, we vary the ratio of the two classes ($x : y$, where $x/(x + y)$ is the fraction of the load for the VBR component and $y/(x + y)$ is the fraction of the load for the best-effort component) to generate mixed-mode traffic.

For the traditional (TR) and the non-preemptive (NP) routers, we divide the VCs into two disjoint groups. (The traditional router uses the same number of VCs as the non-preemptive router for each class, but uses the FCFS scheduler.) When the simulation starts, the ratio ($x : y$) determines the number of VCs assigned to each class. We changed the load in 5 different phases during a simulation run to simulate dynamic workload. In the beginning of the simulation, the real-time to best-effort traffic ratio is 20:80. The VCs are assigned using this ratio. Later the ratio changes to 30:70, 50:50, 70:30, and finally 80:20. The VC configuration for the TR and NP routers does not change during all these phases.

For the preemptive router, any type of traffic can use any VC due to the preemption capability. In the beginning of the simulation, the configuration is the same as the NP router. But as the ratio $x : y$ changes, the real-time traffic can use more VCs by preempting the best-effort messages.

It should be noted that we are considering a flit-level simulation in each stage of the router pipeline, and simulating several simultaneous streams per node. We gather simulation results over a few million messages for avoiding transient behavior. In addition, the integration of the NIC makes the entire simulation quite complex and resource intensive. Therefore, we were able to collect results for single routers and a small 2-D mesh network.

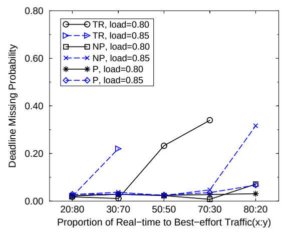
The important output parameters measured in our experiment are deadline missing probability of delivered MPEG-2 frames, average missing time of deadline missing frames, and average network latency for best-effort traffic and control traffic. Deadline missing probability is the ratio of the number of frames that missed their deadlines to the total number of delivered frames. The deadline for each frame is determined by adding 33.3 msec to the previous deadline, since the frame rate is 30 frames/sec for MPEG-2 video streams. However, if a previous frame missed its deadline, a new deadline is set by adding 33.3 msec to the arrival time of the previous frame. Whenever a frame misses its deadline, we measure the deadline missing time and then calculate the average deadline missing time. Average network latency is the averaged time between the injection of the first flit (header flit) of a messages into the network (or a router) and the ejection of the last flit (tail flit) from the network. By including the queueing time with the network latency, we measure the average message latency.

5 Performance Results

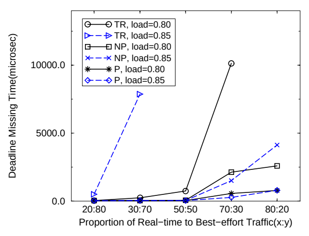
In this section, we analyze the performance results for an 8-port router and a (2×2) mesh network with integrated traffic. Then, we present co-evaluation of a single router and the NIC designs.

Our design spectrum consists of three types of stand alone routers (TR, NP and P models) and two types of NIC architecture (traditional NIC without the VirtualClock algorithm and the QoS-aware NIC with the VirtualClock algorithm). Integration of the NIC and router designs gives 6 combinations. In addition, we also consider the impact of block or message level multiplexing in the router and NIC. Hence, it is impossible to discuss all the results in this paper due to space limitation. We present a subset of the results to highlight the main results. All the results are reported for flit-level multiplexing, unless block/message-level multiplexing is specified.

5.1 Comparison of the Three Router Models

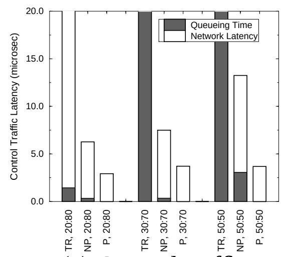


(a)

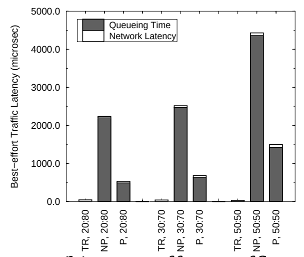


(b)

Figure 4. Deadline missing probability and deadline missing time in a single router under dynamic load variation. The input load is specified in the graphs.



(a) Control traffic



(b) Best-effort traffic

Figure 5. Components of message latency of control traffic and best-effort traffic in a single router under dynamic load variation. The input load is 0.80.

We begin by examining the performance results of a traditional router, a non-preemptive router, and a preemptive

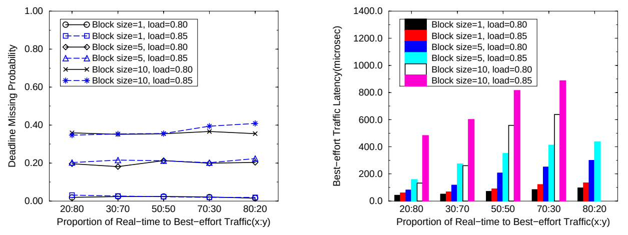


Figure 6. Effect of block-level multiplexing in a single preemptive router under dynamic load variation.

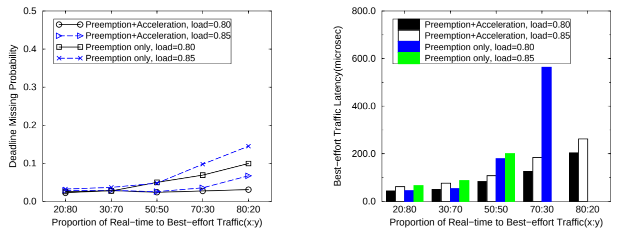


Figure 7. Comparison of (preemption+acceleration) and only preemption in a single router under dynamic load variation. (Some results for the best-effort traffic at higher load are not included due to saturation.)

router under dynamic workloads. Figure 4 shows the deadline missing probability and the average deadline missing time for each model. Some of the data points of the traditional router were dropped due to saturation. It is seen that the preemptive router can service real-time traffic with almost constant deadline missing probability (0.02~0.03), while for the non-preemptive router, the number of frames missing their deadlines increases as the ratio of real-time traffic increases. The deadline missing time in Figure 4 (b) is the minimum for the preemptive router. The traditional router, without a rate-based scheduler, experiences saturation even under light load, and is the worst performer. Since the preemptive router can assign VCs dynamically according to the real-time traffic load, it can provide the best performance among the three architectures. (Moreover, although not shown in the graphs due to space limitations, the advantage of the preemptive model becomes more striking for small number of VCs.)

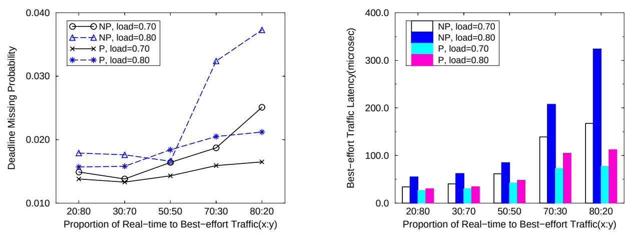
Another important performance parameter is the average latency. Figure 5 (a) shows the control traffic latency in each router. Here, queueing time represents the time spent outside the router before the message is injected into the router. In the traditional router, control traffic is treated as any other types of traffic, and hence its latency is much higher than those of the other two routers. The preemptive router provides the best performance with almost zero queueing time followed by the non-preemptive router. Figure 5 (b) compares the best-effort traffic latency in the three routers. The non-preemptive and the preemptive routers can provide better service for real-time traffic at the expense of best-effort traffic. Therefore, as expected, the TR provides the best performance for best-effort traffic.

Next, we examined the impact of block-level multiplexing in a single preemptive router. Figure 6 depicts the results for block size of 1, 5, and 10 flits, respectively. As the block size increases, the performance degrades significantly. Thus, flit-level multiplexing seems to be the most ideal choice for QoS assurance. However, in an actual implementation, we may have to use block-level multiplexing to amortize the scheduling overhead.

In order to estimate the contribution of the acceleration scheme explained in Section 3 for the preemptive router, we tested the router without the acceleration scheme and with the acceleration scheme. Figure 7 demonstrates the role of the acceleration scheme in the preemptive router. The results indicate that by accelerating the flits of the lower priority messages, performance of both real-time and best-effort traffic improves considerably.

5.2 A (2×2) Mesh Network Results

In this subsection, we examine the performance implications of using the preemptive and non-preemptive routers in a (2×2) mesh network. Figure 8 (a) shows the deadline missing probability for real-time traffic and Figure 8 (b) depicts the average network latency for best-effort traffic. Like the single router results, the preemptive model again exhibits better performance compared to the non-preemptive model. The deadline missing probability increases with an increase of the real-time load. Also as expected, the average network latency of the best-effort traffic in Figure 8 (b) gradually increases with the real-time traffic.



(a) (b)

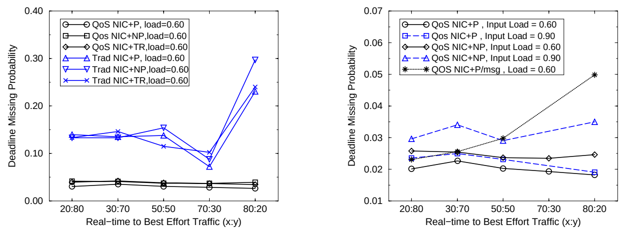
Figure 8. Deadline missing probability and average latency of best-effort traffic in a (2×2) mesh network under dynamic load variation.

5.3 Router and NIC Co-evaluation

Next, we present the co-evaluation of the NIC and a single router. An (8×8) router is integrated with the NICs at both ends. The three different router models (TR, NP, and P) and the two NIC models (traditional and QoS-aware) are combined to examine six possible designs. Figure 9 shows the comparison of the six combinations in terms of the deadline missing probability. Figure 9 (a) is plotted for a fixed input load of 60% since the traditional NIC cannot support higher load. In this figure, all the three routers with the

QoS-aware NIC outperform the traditional NIC combinations (The dip for the 70:30 ratio is a suspect.) suggesting that QoS provisioning in the NIC is rather more important to see any performance benefits in the router design. Not only that a traditional NIC cannot support higher input load, but even with 40-60% input load, the deadline missing probability is quite high. In Figure 9 (b), we compare the preemptive and non-preemptive router models along with the QoS-aware NIC for higher load. The results are again in favor of the preemptive router. In both these graphs, the preemptive router and the QoS-aware NIC combination has the best results.

Finally, we conducted an experiment by using message-level multiplexing in the QoS-aware NIC. In this case, unlike the flit-level multiplexing, the VirtualClock scheduler selects and injects one message at a time to the QoS-aware router. We increased the input buffer of the router to avoid buffer overflow. The simulation model includes the DMA overhead for injecting the messages. The (QoS NIC + P/msg) graph in Figure 9 (b) shows that message-level multiplexing in the NIC results in worse performance compared to the flit-level multiplexing. This trend is similar to the block-level multiplexing results in the router as shown in Figure 6. Moreover, with message-level multiplexing, the system cannot handle higher load (0.6 in the graph). Overall, these results suggest that not only QoS provisioning in the NIC is critical, but also we need hardware mechanism for flit-level multiplexing to maximize the performance benefits.



(a) (b)

Figure 9. Co-evaluation of a single router and NICs. The deadline missing probability is shown for dynamically changing workload.

6 Concluding Remarks

This paper addresses QoS support mechanisms in cluster interconnects. We propose a pipelined wormhole router architecture that can handle multiple traffic types effectively. Two different router implementations, known as non-preemptive and preemptive models, are studied with a mixed workload of best-effort and real-time traffic. Another uniqueness of this work is that it integrates the NIC with the router design to examine the end-to-end QoS issue. We have developed a VIA-style NIC to support integrated traffic in clusters. The important conclusions of this work are the following.

First, it is possible to support integrated traffic in clusters by augmenting the conventional wormhole routers with a rate-based scheduling mechanism. The preemptive and non-preemptive models are feasible choices, and can be implemented in the realm of current VLSI technology. While both the implementations provide similar performance for static workloads, the preemptive model turns out to be a better choice for dynamic workloads, and for routers with small number of VCs. Second, preemption in a pipelined model is more complex than in a non-pipelined (lumped) model since a lower priority message can block a higher priority message at several stages of the pipeline. Instead of providing preemption at these stages, preemption in the input buffers followed by an acceleration mechanism at other stages seems a viable design. Next, it is shown that a QoS capable router alone is not adequate to provide end-to-end QoS support. A NIC with QoS provisioning that uses schemes like prioritized doorbells and a rate-based algorithm is necessary to ensure deadline oriented delivery of real-time traffic. Finally, in all prior studies, the focus was only on the design of routers to provide QoS guarantees. Our study reveals that performance predictability in the NIC is rather more important to translate the performance benefits to the application level. Thus, design of QoS-aware NICs should be undertaken seriously. In this context, design compatibility between the router and the NIC offers several design trade-offs. For example, routers have usually used flit-level multiplexing, while the NIC to network transfer is at a message-level granularity. Therefore, proper hardware support for flit-level multiplexing in the NIC, as used in this paper, is ideal for maximizing the performance. Otherwise, a detailed analysis of block/message-level multiplexing in the router as well as in the NIC is required to arrive at an optimal design.

We are currently examining a number of possible extensions to this work. First, we are exploring other design alternatives in the VIA paradigm to improve the QoS support in the NIC. Second, we were unable to analyze the designs using larger networks primarily due to high time complexity of the simulations. Thus, optimization of the simulation model and possible parallelization should reduce the execution time. Finally, we would like to implement the router as well as the NIC in hardware (prototyping in FPGA followed by the VLSI implementation) to examine the practicality of our models. We have already written a VHDL simulator of the router for this purpose.

7 Acknowledgments

Many thanks to Vithal Shirodkar, Srinivas Hanabe, Giridhar Viswanathan, and Nishant Choudhary for developing and testing the NIC simulators. We also would like to thank the anonymous referees for their comments and suggestions that improved the quality of the paper.

References

[1] S. Araki, A. Bilas, C. Dubnicki, J. Edler, K. Konishi, and J. Philbin. User-Space Communication: A Quantitative

Study. In *Proc. of Supercomputing Conf. (SC'98)*, November 1998.

[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] P. Buonadonna, A. Geweke, and D. E. Culler. An Implementation and Analysis of the Virtual Interface Architecture. In *Proc. of Supercomputing Conf. (SC'98)*, November 1998.

[4] M. B. Caminero, J. J. Quiles, J. Duato, D. S. Love, and S. Yalamanchili. Performance Evaluation of the Multimedia Router with MPEG-2 Video Traffic. In *Proc. of Intl. Workshop on Communication, Architecture and Applications on Network Based Parallel Computing (CANPC'99)*, pages 62–76, January 1999.

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

[6] Compaq Corp., Intel Corp., and Microsoft Corp. *Virtual Interface Architecture Specification, Version 1.0*, 1997. Available at `http://www.viarch.org`.

[7] K. Connelly and A. A. Chien. FM-QoS: Real-Time Communication Using Self-Synchronizing Schedules. In *Proc. of Supercomputing Conf. (SC'97)*, November 1997.

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

[9] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Journal of Internetworking Research and Experience*, pages 3–26, October 1990.

[10] J. Duato, S. Yalamanchili, M. B. Caminero, D. Love, and F. J. Quiles. MMR: A High-Performance Multimedia Router-Architecture and Design-Tradeoffs. In *Proc. of Intl. Symp. High Perf. Comp. Arch. (HPCA-5)*, pages 300–309, January 1999.

[11] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. M. Merritt, E. Gronke, and C. Dodd. The Virtual Interface Architecture. *IEEE Micro*, 18(2):66–76, March/April 1998.

[12] H. Eberle and E. Oertli. Switcherland: A QoS Communication Architecture for Workstation Clusters. In *Proc. of Intl. Symp. Comp. Arch.*, pages 98–108, June 1998.

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

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

[15] 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. of Intl. Conf. Network Protocols*, pages 40–47, October 1996.

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

[17] K. Knauber and B. Chen. Supporting Preemption in Wormhole Networks. In *Proc. of Intl. Comp. Software and Applications Conf. (COMPSAC'99)*, pages 232–238, October 1999.

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

[19] Myricom Inc. M3M-PCI64B Network Interface Card. Available at `http://www.myri.com/myrinet/PCI64/m3m-pci64b.html`.

[20] S. Pakin, M. Lauria, and A. A. Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet. In *Proc. of Supercomputing Conf. (SC'95)*, December 1995.

[21] J. Pelissier. Providing Quality of Service over InfiniBand Architecture Fabric. In *Proc. of Symp. High Perf. Interconnects (Hot Interconnects 8)*, August 2000.

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

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

[24] D. Stiliadis and A. Varma. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. *IEEE/ACM Trans. on Networking*, 6(2):611–623, April 1998.

[25] 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*, 34(2):185–204, 1995.

[26] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. In *Proc. of ACM Symp. Operating Systems Principles*, pages 40–53, December 1995.

[27] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active Messages: A Mechanism for Integrated Commnication and Computation. In *Proc. of Intl. Symp. Comp. Arch.*, pages 256–266, May 1992.

[28] T. von Eicken and W. Vogels. Evolution of the Virtual Interface Architecture. *IEEE Computer*, 31(11):61–68, November 1998.

[29] K. H. Yum, E. J. Kim, V. Shirodkar, S. Hanabe, and C. R. Das. Design and Analysis of a Versatile Router for Supporting Integrated Traffic in Clusters. Technical Report CSE-00-021, Pennsylvania State University, University Park, PA, October 2000.

[30] K. H. Yum, A. S. Vaidya, C. R. Das, and A. Sivasubramaniam. Investigating QoS Support for Traffic Mixes with the MediaWorm Router. In *Proc. of Intl. Symp. High-Perf. Comp. Arch. (HPCA-6)*, pages 97–106, January 2000.

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