NIC-Based Rate Control for Proportional Bandwidth Allocation in Myrinet Clusters*

Abhishek Gulati Dhabaleswar K. Panda P. Sadayappan Pete Wyckoff‡

Network-Based Computing Laboratory
Dept. of Computer and Information Science
The Ohio State University
Columbus, OH 43201
{gulati, panda, saday}@cis.ohio-state.edu

‡*Ohio Supercomputer Center*
Columbus, OH 43212
pw@osc.edu

Abstract

Simultaneous advances in processor, network and protocol technologies have made clusters of workstations attractive vehicles for high performance computing. However, clusters are now being increasingly used in environments characterized by non-cooperating communication flows with a range of service requirements. This necessitates Quality of service (QoS) mechanisms in clusters. The approaches to QoS in the wide-area networking context are not suitable for clusters because of the high overheads. Also, contention between flows at the end-nodes has not been addressed earlier. In this paper, we explore the use of "rate control" as a means for proportional bandwidth allocation in clusters. A NIC-based solution is presented, with details on implementation in Myrinet/GM. Experimental results show that rate control can handle both end-node and network contention, without adding significant overhead. Our approach is particularly attractive since it does not require hardware modifications, and can hence work with commodity systems with programmable NICs.

1. Introduction

As a result of high-performance interconnection networks such as Myrinet [1], and ever-increasing processor speeds, workstation clusters have emerged as powerful computing vehicles. Past research has largely focused on minimizing the software overhead associated with traditional network protocols like TCP/IP. The so-called "user-level network protocols" (VIA [13], GM [7] etc.) do so by leveraging "smart" Network Interface Cards (NICs), taking operating system involvement out the critical path of communication and eliminating extra message copies.

These technological advancements have made low latency, high bandwidth communication possible, enabling the operation of clusters in the traditional single-application mode. However, efficient resource utilization stipulates simultaneous use by multiple applications. This is particularly true for clusters of symmetric multiprocessors (SMPs), which are gaining popularity because of the growing cost-effectiveness of 2-way and 4-way SMPs. Further, in addition to conventional scientific computing applications, clusters are increasingly being used for latency and bandwidth sensitive applications like interactive simulations, scientific visualization and video-conferencing. As a result of these two trends, network traffic is clusters is now characterized by the coexistence of non-cooperating communication flows with a variety of service requirements.

As shown in Figure 1, coexisting communication flows contend for shared resources (such as buffers, DMA engines) at end-nodes as well as in the network. Disorderly contention may affect performance in ways that may not be tolerated by latency or bandwidth-sensitive applications. In addition, predicable performance also benefits best-effort parallel applications. This motivates the need for Quality of Service (QoS) features, which are not provided by any of the above-mentioned networks and protocols. QoS metrics may be bandwidth, delay or delay jitter. In this work, we focus on the former.

QoS issues have been dealt with in other contexts such as ATM networks. However, the solutions rely on switches to provide sophisticated packet scheduling dis-

* This research is supported in part by an NSF Career Award MIP-95092294, NSF Grants CCR-970452 and EIA-9986052, an Ameritech Faculty Fellow Award, and grants from the Ohio Board of Regents.

ciplines [15]. The complexity and high latency of these schemes makes them unacceptable for high-performance clusters. For this reason, wormhole-routed networks like Myrinet, popularly used in clusters, use simple round-robin packet scheduling. Another issue not raised in prior work is that of contention between multiple flows at an end host. With the widespread use of SMP nodes in clusters, handling of contention between multiple concurrent flows from different processes on the same SMP node is important to address.

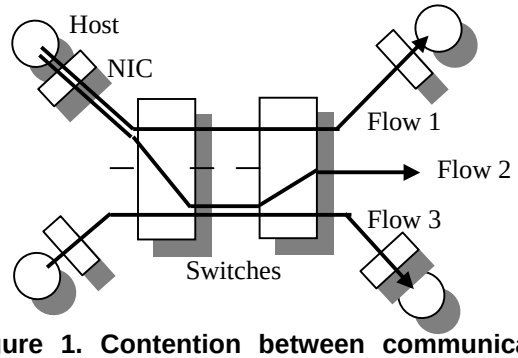

Figure 1. Contention between communication flows in a workstation cluster. Flow 2 contends with Flow 1 at the end-node and with Flow 3 in the network.

In this paper, we propose a scheme wherein the responsibility of imposing the packet scheduling discipline is delegated to the source node instead of the switch. Specifically, we explore "Rate control" at source, which refers to the act of regulating the rate at which data is fed to the network interface and injected into the network. This is used in conjunction with the global knowledge of traffic patterns in the cluster to provide proportional bandwidth allocation to contending flows. While a host-based implementation is possible, a NIC-based implementation is preferred because of the finer granularity of rate control allowed by the latter. Further, since rate control is implemented by the trusted components on the NIC, there is no need for policing in the switch. Hence no modifications are needed to commodity hardware, making our scheme particularly attractive.

We have incorporated the rate control mechanism in the GM messaging layer over Myrinet. Experimental results demonstrate the efficacy of our scheme in allowing proportional bandwidth allocation to flows contending for end-node and network resources, without adding much overhead.

The rest of this paper is organized in the following manner. Section 2 introduces certain basic concepts required for understanding the following treatment. In section 3, we describe our approach in detail. In section 4, we outline the design and implementation of rate control in GM/Myrinet. Experimental results are presented in section 5. Section 6 briefly discusses some related work. Conclusions are drawn and future work stated in section 7.

2. Basic concepts

In this section, we make the notion of communication flow more concrete, and discuss the operation of user-level networking protocols and wormhole-routed networks in general. We also illustrate how various communications flows are serviced in existing systems.

2.1. Communication flows

A "communication flow" is a message stream with a well-defined source and a well-defined sink. The communication end-points (sources and sinks) are logical; there may be many such endpoints on the same network node. Additionally, many communication flows may be multiplexed over the same physical link. Each end-point may be the source or sink of many communication flows. Examples of communications end-points are Virtual Interfaces (VIs) in VIA, Queue Pairs (QPs) in the Infiniband Architecture [8], or TCP/IP sockets. Figure 2 further highlights these concepts.

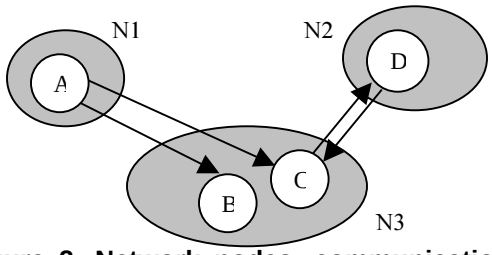

Figure 2. Network nodes, communication end-points and communication flows. This logical diagram represents three network nodes N1, N2 and N3; four communication endpoints A, B, C and D; and four communication flows (A, B), (A, C), (C, D) and (D, C).

For this paper, we assume that all packets belonging to different communication flows have the same size. This is a reasonable assumption because bandwidth sensitive applications use large messages, which are all segmented into MTU-sized packets (where MTU is the maximum transmission unit).

2.2. User-Level Network Protocols

User-level network protocols allow message sends and receives without operating system involvement, while still providing protected access to the network interface (Figure 3). Operating system involvement is required initially for registration of DMAable memory and setting up channels (VIA Virtual interfaces or GM ports) etc. Channels are constructs that, among other things, allow user applications to inform the NIC firmware about new send or receive requests. This is typically done by

posting descriptors describing the required operation. A send descriptor specifies the location and the size of the send buffer, as well as the destination. A receive descriptor describes the location of the buffer where the received message is to be placed.

The NIC firmware obtains descriptors by polling memory-mapped descriptor queues (as in GM), or through a doorbell mechanism (as in VIA). Fetched send descriptors are serviced by DMAing data from registered memory regions into the NIC buffer and then dispatching packets to the network. On receiving packets from the network, the NIC DMAs the data into the registered memory buffers indicated by receive descriptors.

In existing user-level network protocols, send descriptors from various flows are typically serviced in a round robin fashion. As shown in Figure 4a, this is inherently unfair, as it favors communication flows with longer message sizes.

A fairer variation is supported by some user-level network protocols. Here, packet dispatch from various flows is interleaved to allow uniform usage of NIC resources. This is illustrated in Figure 4b.

In either case, the scheduling discipline does not take into account the service requirements of various communication flows.

2.3. Wormhole-routed networks

In wormhole-routed networks, packets are comprised of small units called flits. The key distinction from store-and-forward routed networks is that a flit is forwarded to the output port (if it is free) as soon as possible, without waiting for the entire packet to arrive at the input port. This allows lower latencies and better network utilization.

As mentioned earlier, wormhole-routed networks service packets destined for the same output port in a round-robin fashion. A communication flow desiring high bandwidth may still suffer if its packets have to traverse longer paths to the destination than packets from other flows. This is so because the former may have to wait on more intermediate switches than the latter (Figures 5a-5b).

3. NIC-based Rate control

In this section, we describe our approach in detail. The basic idea is discussed first, followed by the rate control and call admission criteria in greater detail.

3.1. Basic idea

"Rate control" refers to the act of imposing a specific discipline on the rate at which data is fed to the network interface and injected into the network. In this work, we propose the use of rate control to proportionally allocate bandwidth between various flows that contend for NIC and network resources. Our scheme requires agents that regulate communication flows at the source. The rate at which packets are injected to the network interface affects how shared resources are used by the corresponding flow. For example, a severely regulated flow uses the NIC-to-host DMA engine, the NIC buffer, and the host-to-network DMA engine less often than a slightly regulated one. Likewise, a flow can achieve higher bandwidth if the rates of the interfering flows are suitably manipulated (Figure 6). In [2], the authors show how rate control considerably improves network fairness and increases the probability of meeting message deadlines for real-time communication.

3.2. NIC-based vs. Host-based Rate-control

There are two clear alternatives as to where rate control can be implemented: at the host or at the NIC. While a host-based implementation is simpler, it is not preferable due to two reasons. Firstly, in most user-level protocols, the host deals with messages, not packets. Messages can be very large, which makes host-level rate control much coarser. More importantly, since message sends bypass the operating system, there is no way to ensure that communication flows are actually regulated. On the other hand, with a NIC-level implementation, all sends will have to go through the NIC firmware with rate-control features. NIC firmware is downloaded to the NIC by OS components and can hence be trusted.

3.3. Rate Control Algorithm

Our rate control algorithm associates each flow with a parameter called the Inter-packet Dispatch Time (IDT). It refers to the minimum interval between consecutive injections of packets from a communication flow to the network interface.

At any given time, let $f_1, f_2 \ldots f_m$ be m communication flows sourced at a particular node. Let $(idt_1, idt_2, \ldots idt_m)$ be the corresponding IDT values. We also define a set of Next Dispatch Time (NDT) values $(ndt_1, ndt_2, \ldots ndt_m)$, which specify the absolute time before which a packet from a given flow should not be dispatched. NDT values are initialized to the current time when a message send is requested. The rate control algorithm is as follows:

At any time *t*,

Let *j* be the communication flow such that

$ndt_j = min\ (ndt_1, ndt_2 \ldots ndt_m)$

If $ndt_j <= t$

Dispatch packet from flow *j*.

$ndt_j = ndt_j + idt_j$

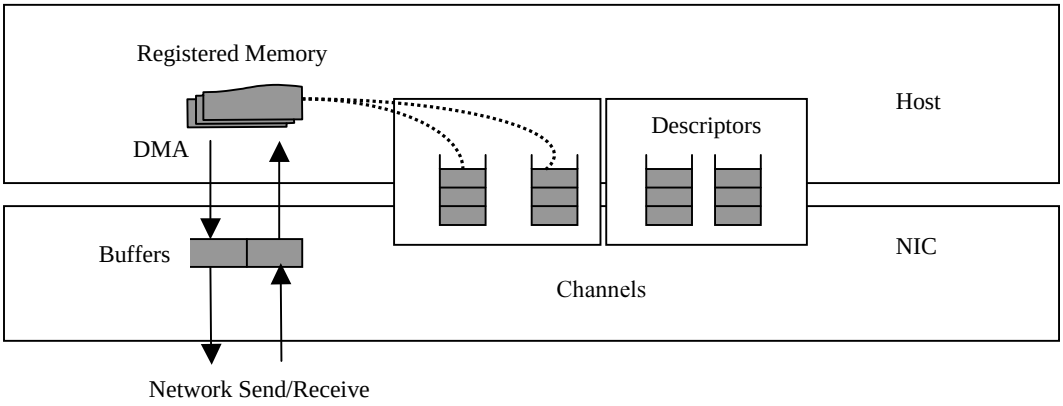

Figure 3. Basic operation of user-level network protocols.

Figure 4. Packet scheduling disciplines. Two communication flows A and B are shown. A's message size is thrice the supported MTU, whereas B has only single-packet messages. a) Round-robin servicing of send descriptors: A uses NIC resources thrice as often as B. b) Fair interleaved Round Robin servicing of send descriptors. NIC usage is uniform, independent of message sizes.

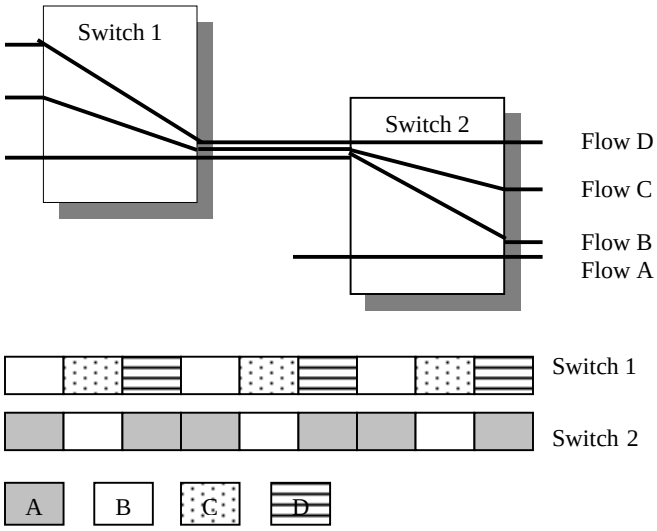

Figure 5: Contention in wormhole-routed Networks. a) Flows B, C, D contend for the same port at switch 1. Flow B contends with Flow A at switch 2. b) Flow B suffers at switch 2.

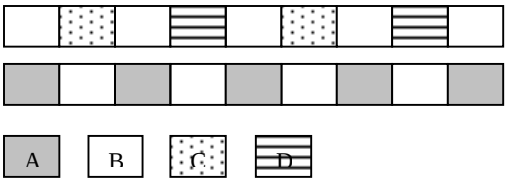

Figure 6. Easing network contention by rate control. Same flows as in Figure 5a, but flows C and D are regulated, hence flow B does not suffer at switch 2.

Table 1. Working of the rate control algorithm.

t	0	1	2	3	4	5
NDT (A)	0	2	2	4	4	6
NDT (B)	0	0	3	3	6	6
Packet dispatched from flow:	A	B	A	B	A	-

The algorithm works as follows. Assuming that there are always outstanding send requests, the communication system can service these requests at most once in every T time units. This T corresponds to the peak achievable bandwidth, B_{peak}. Under the constraint that all IDT values are larger than T, the rate at which the packets from these flows is dispatched will be in the ratio of $1/idt_1 : 1/idt_2 :\ldots 1/idt_m$.

Table 1 illustrates the algorithm with a concrete example. Here, flows A and B have IDTs of 2 and 3 (normalized with respect to T) respectively. Note that the ratio in which packets are dispatched is 3:2 or 1/2: 1/3 as expected.

3.4. Call Admission Criterion

We propose to use rate control in conjunction with a call admission agent. The criteria for admitting a new communication flow with a given bandwidth requirement is as follows:

a) If the flow is admitted, then the cumulative bandwidth requirement for all flows at the source network node should not exceed its peak capacity.

b) If the flow is admitted, the cumulative bandwidth requirement for all flows using the switch output ports traversed by the new flow should not exceed the corresponding outgoing link capacity.

c) Finally, if the flow is admitted, then the cumulative bandwidth requirement for all flows at the destination should not exceed its peak capacity.

If the new flow can be admitted according to the above criteria, then its IDT value is chosen as follows:

$$idt = T * (B_{peak} / B_{req}),$$

where B_{req} is the bandwidth requirement of the admitted flow.

Our scheme relies on certain properties of clusters. Firstly, all traffic in clusters is internal, with well-defined sources and sinks. Further, most clusters use deterministic source routing, which means that the path taken by packets from a given flow is fixed. This simplifies the call admission criteria. Lastly, unlike in wide-area networks, nodes in clusters belong to the same administrative domain and can be trusted. Since rate control at the source node is done by trusted software components, such as NIC firmware, there is no need for policing (enforcement of packet rates) at the switches. Hence no modifications are required in the hardware.

In this paper, we do not address the issue of CPU contention. In environments characterized by more that one compute-intensive job per processor, rate control needs to be used in conjunction with a CPU reservation scheme in ensuring end-to-end bandwidth guarantees.

4. Implementation in Myrinet/GM

Here, we describe the implementation of rate control in Myrinet/GM. Overviews of Myrinet and GM are provided first.

4.1. Overviews of Myrinet and GM

Myrinet is a switched Gigabit-per-second system area networking technology. The switches are crossbar and use wormhole switching. A Myrinet NIC connects to the I/O bus and features a programmable processor, LANai and three DMA engines (one for host-NIC transfer, the other two for sending and receiving data to/from the network respectively). The NIC also contains a small, but fast SRAM, which holds the firmware and is also used to stage data that goes in and out of the network.

GM is a message-based communication system for Myrinet featuring low CPU overhead, low latency and high bandwidth. Reliable delivery is provided between end-points called ports. Two levels of priority are supported, and message order is preserved for messages of the same priority between the same sending and receiving ports. Messages must be sent from and received into DMAable memory.

The software components of GM include a library, a driver and the NIC firmware called the Myrinet Control Program (MCP). The library links with the client and provides the Application Programming Interface (API). It includes functions for opening ports, allocating DMAable memory, sending and receiving messages etc. The former operations are carried out with the help of the driver, but no driver involvement is required for message sends and receives. These are carried out through direct interaction between the client and the MCP as described below.

The MCP consists of four state machines: SDMA, RDMA, SEND and RECV. When a client wishes to send a message, a send descriptor describing the message is written to a memory-mapped location in the NIC. The SDMA state machine detects the descriptor, builds a corresponding send token and inserts the token into an appropriate send queue. Further, it polls send queues as described below, prepares packets and DMAs data into the transmit buffer. The SEND state machine transfers packets from the transmit buffer to the network. This is illustrated in Figure 7.

In GM, queues are hierarchically arranged (Figure 8). At the highest level is a circular connection queue, which has an entry for each remote node with which communi-

cation is active. Each connection entry points to a circular queue of sub-ports (a combination of a port and a priority level) that are communicating over that connection. Each sub-port entry points to a queue of send tokens. Some tokens have been serviced completely and are awaiting acknowledgements, others that have not been completely serviced are called "sendable" tokens. At any time, one connection entry and a corresponding sub-port entry are designated as the "head" entries. After one packet dispatch corresponding to a "sendable" token from the head sub-port, the sub-port queue for the head connection is rotated and so is the connection queue. This allows fair network access as demonstrated in Figure 4b.

The RECV and RDMA state machines similarly cooperate for message receives.

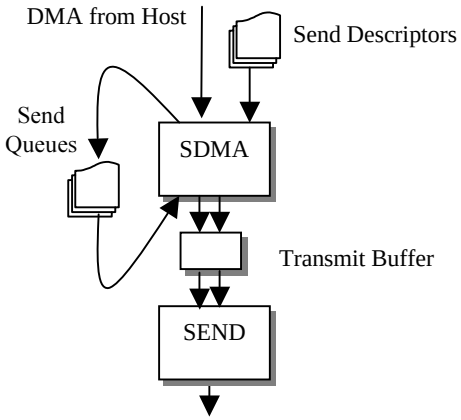

Figure 7. MCP operation for message sends.

4.2. GM Modifications

The GM library has been extended to allow the NIC to be informed of the IDT values for various communication flows. A flow is defined as the ordered pair *(source port, target port, priority).* This ensures that GM message order semantics are not violated, as rate control does not require out-of-order sends on packets belonging to the same flow. The unit in which IDTs are specified is the least count of the Real Time Clock (RTC) available on the LANai processor.

The modified SDMA state machine is responsible for maintaining information about the validity of a flow, and its NDT, IDT values. A flow is defined to be valid if it has send tokens at that moment. Whenever a send token is inserted as the first "sendable" token for a flow, its NDT value is updated to the larger of RTC and the existing NDT value.

Packet scheduling in the NIC is based on NDT values. The SDMA state machine finds the flow with the minimum NDT among all valid flows. If RTC is larger than the NDT value for this flow, its first "sendable" token (if any) is serviced, and its NDT is incremented by its IDT.

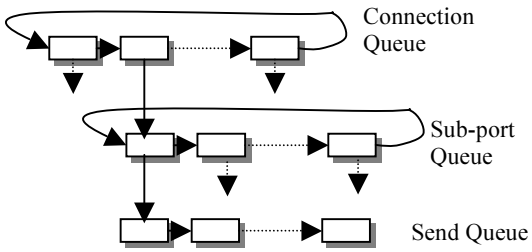

Figure 8. Queue organization in GM.

5. Experimental Results

In this section, we present our experimental results. We show the effectiveness of the rate control mechanism in allowing proportional bandwidth allocation without significant overhead.

5.1. Experimental Setup

We evaluated our implementation on a cluster of machines with 300 MHz Pentium II processors, 128 MB RAM, running RedHat Linux 6.0 with kernel version 2.2.5. These machines were connected by a 16-port Myrinet switch and had LANai 4.3 NICs with 33 MHz processors.

5.2. Overhead of the rate control mechanism

We determined the overhead of the rate control mechanism by comparing the performance of unmodified GM with the modified version. Round-trip latency was measured using the standard ping-pong test. Bandwidth measurements involved pumping N messages, getting a small-sized acknowledgement from the receiver, and dividing the total volume of data transferred by the total elapsed time.

Figure 9 demonstrates that the rate control feature incurs an overhead of less than 1% in terms of latency. Similarly, Figure 10 shows that the bandwidth loss is less than 1% averaged over a range of message sizes, and 4.5% in the worst-case. Note that, for some message sizes, the modified GM seems to perform better than the unmodified version. This is the result of the difference in the message segmentation schemes. In order to keep the DMA cycles balanced, GM tries to segment packets as equally as possible. We have disabled this feature such that all packets except the last one for a message are 4KB-sized.

5.3. Single Flow

Figure 11 shows the effect of varying the IDT on the bandwidth achieved by a single flow. Notice that for all IDT values less than 96, the bandwidth is equal to the peak achievable value. This implies that packets are sent

at the rate of no more than once very 96 units. We use this value of *T* for the remainder of the experiments.

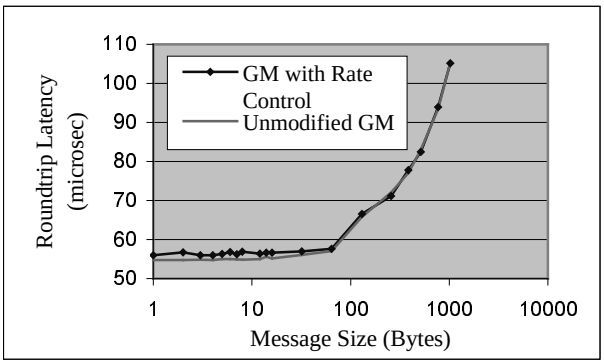

Figure 9. Effect of rate control feature on the round-trip latency.

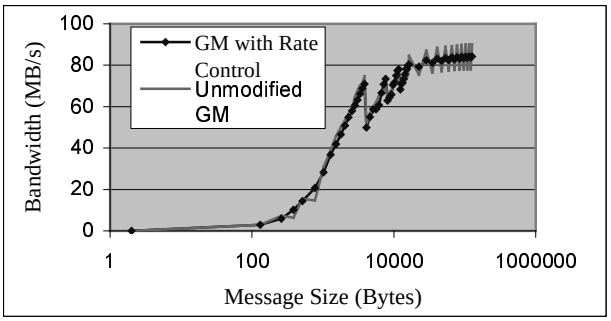

Figure 10. Effect of rate control feature on the achieved bandwidth.

5.4. Multiple Flows

As shown in Figure 12a, bandwidth between two flows sourced at the same node, but with different destinations is shared in the inverse ratio of IDT values. Similar results are observed for three flows with the same source but different destinations (Figure 12b), and for two flows with different sources but the same destination (Figure 12c). In the latter case, the two flows contend for network resources, whereas in previous cases, they contend at the end-node. In Figure 13, we show how, for different rates of an incoming flow, the peak achievable bandwidth (obtained from Figure 14) is shared by two outgoing flows. Interested readers are referred to [6] for detailed explanations of these results, which are omitted here due to space constraints.

6. Related Work

In recent years, a few other studies have examined service differentiation and the related issue of QoS in clusters. In [14] and [4], the authors propose new hardware or hardware/software infrastructure to support this efficiently. Low-cost packet scheduling and queuing algorithms for high performance networks are described in [9]. In [5], Gerla et al describe three schemes for providing QoS in wormhole-routed networks. However, these require modifications to switches or extra NICs on hosts. FM-QoS [3] provides predictable performance by devising conflict-free communication schedules. Lastly [2], [12] and [10] suggest two admission control schemes for meeting delay requirements in wormhole-routed networks. Our work differs from most prior work on QoS in cluster environments in addressing the issues of network and end-node contention without modifications to existing hardware. Although it focuses on Myrinet/GM, the principles can be applied to other systems with programmable NICs.

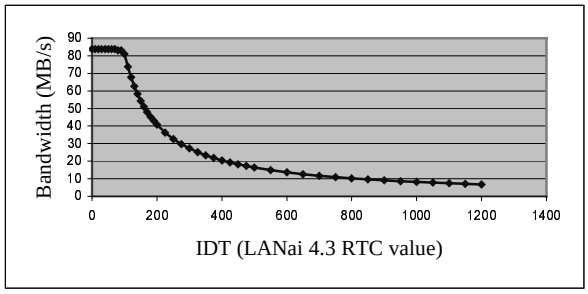

Figure 11. Effect of varying the IDT of a single flow (message size = 64 KB) on the achieved bandwidth.

7. Conclusions and future work

In this paper, we proposed the use of rate control for proportional bandwidth allocation in clusters. A NIC-based solution was presented, with details on implementation in GM/Myrinet. Experimental results showed that rate control could handle both end-node and network contention, without adding significant overhead. Further, no hardware modifications were necessary.

With the rate control implementation in place, we intend to study how its advantages can be carried on to the application level. This would entail integrating our work with higher layers such as MPI. We also intend to look at how we could use our work in conjunction with schemes for QoS in grid computing, such as MPICH-GQ [11] to allow geographically distributed applications.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kualwik, C. L. Seitz, J. N. Seizovic, Wen-King Su. Myrinet-a gigabit-per-second local-area network. IEEE Micro, 15(1):29-36, February 1995.

[2] B. Chen, H. Li, W. Zhao. Meeting Delay Requirements in Computer Networks with Wormhole Routing. In Proc. of the IEEE International Conference on Distributed Computing, Hong Kong, May 1996.

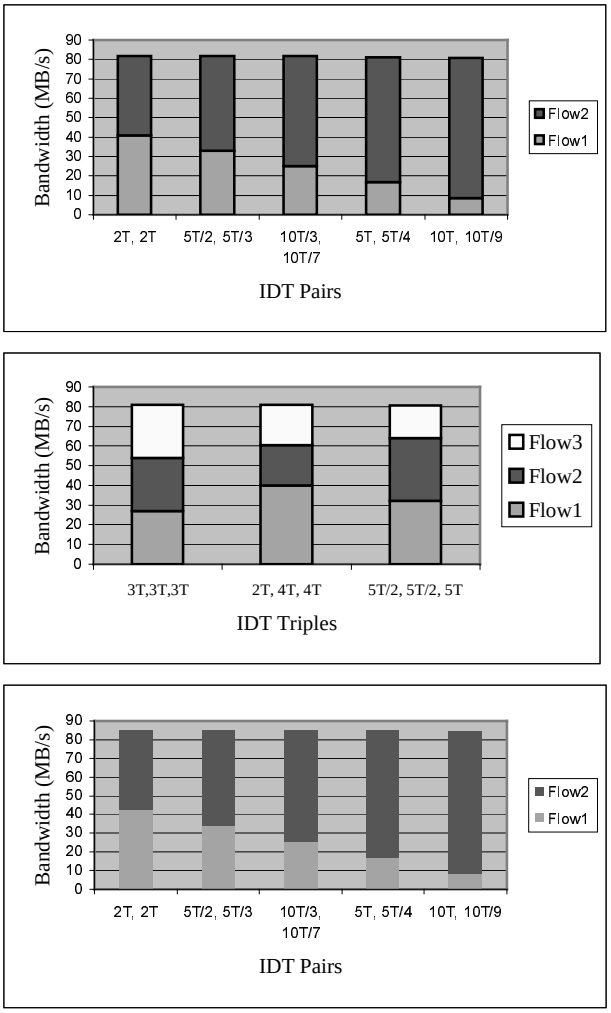

Figure 12. Proportional bandwidth sharing between multiple flows (with 64KB messages). a) Two flows sourced at the same node, but destined for different nodes. b) Three flows sourced at the same node, but destined for different nodes. c) Two flows sourced at different nodes, but destined for at the same node.

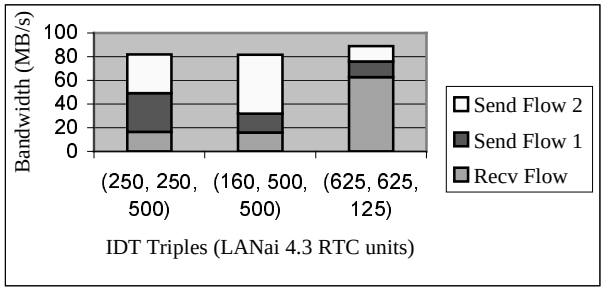

Figure 13. Proportional bandwidth sharing between two outgoing flows (with 64KB messages) for different incoming packet rates.

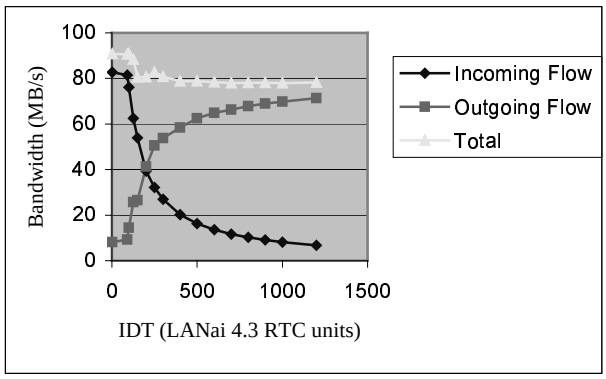

Figure 14. Peak achievable bandwidth of an outgoing flow for different incoming packet rates.

[3] K. Connelly, A. Chien. FM-QoS: Real-time communication using self-synchronizing schedules. In Proc. of Supercomputing Conference, San Jose, CA, November 1997.

[4] H. Eberle, E. Oertli. Switcherland. A QoS communication architecture for workstation clusters. In Proceedings of ACM ISCA '98, Barcelona, Spain, June 1998.

[5] M. Gerla, B. Kannan, B. Kwan, P. Palnati, S. Walton, E. Leonardi, F. Neri. Quality of service support in high-speed, wormhole-routing networks. In Int'l Conf. on Network Protocols, Oct. 1996.

[6] A. Gulati, A proportional bandwidth allocation scheme for Myrinet clusters. M.S. Thesis, Dept. of Comp. & Info. Science, The Ohio-State University, June 2001.

[7] The GM Message Passing System. Documentation available at http://www.myri.com/scs/index.html.

[8] The Infiniband Architecture Specification, Volumes 1 and 2, Release 1.0, available at http://www.infinibandta.org.

[9] J. H. Kim, A. Chien. Rotating combined queuing (RCQ): Bandwidth and latency guarantees in low-cost, high-performance networks. In Proceedings of the International. Symposium on Computer Architecture: 226-236, May 1996.

[10] M. W. Mutka. Using Rate Monotonic Scheduling Technology to Support Real-Time Communications in Wormhole Networks. In the Proceedings of the Second Workshop on Distributed and Parallel Real-Time Systems: 194-199, April, 1994.

[11] A. J. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, B. Toonen, MPICH-GQ: Quality-of-Service for Message Passing Programs. In Proc. of Supercomputing '00, November 2000.

[12] S. Sundaresan, R. Bettati. Distributed Connection Management for Real-Time Communication over Wormhole-Routed Networks. Technical Report 96021, Department of Computer Science, Texas A&M University, December 1996.

[13] Virtual Interface Architecture Specification, Version 1.0. Available at http://www.viarch.org.

[14] R. West, R. Krishnamurthy, W. Norton, K. Schwan, S. Yalamanchili, M. Rosu, S. Chandra. Quic: A quality of service network interface layer for communication in NOWs. In Proc. of the Heterogeneous Computing Workshop, in conjunction with IPPS/SPDP, San Juan, Puerto Rico, April 1999.

[15] L. Zhang, Virtual Clock: A new traffic control algorithm for packet switching networks. In Proceedings of ACM SIGCOMM'90:pages 19-20, Philadelphia, PA, September 1990.