

Quality of Service Support in High-Speed, Wormhole Routing Networks*

Mario Gerla, B. Kannan, Bruce Kwan,
Prasasth Palnati, Simon Walton
UCLA,
405 Hilgard Ave,
Los Angeles, CA 90095-1596

Emilio Leonardi, Fabio Neri
Dipartimento di Elettronica
Politecnico di Torino,
Torino, Italy 10129

Abstract

Wormhole routing networks have become increasingly popular for low latency, high-speed interconnection of supercomputer and workstation clusters. An example is the Supercomputer SuperNet (SSN) at UCLA, which interconnects supercomputers across campus and metropolitan area distances. SSN employs a two-level network architecture in which an optical backbone network interconnects several high-speed, wormhole-routing local area networks (Myrinets). SSN applications such as scientific visualization and rendering require that the network support reliable delivery of traffic characterized by Quality of Service (QoS) parameters. Motivated by this requirement, we investigate QoS support in Myrinet-like high-speed, wormhole routing networks. Since native Myrinet protocols do not provide QoS support, we explore several novel strategies including (a) the use of a separate subnet for carrying such traffic (along with source pacing), (b) the overlay of a virtual synchronous system on the asynchronous network, and (c) the introduction of virtual channels. We discuss the tradeoffs among the different options and evaluate them via selected simulation experiments.

1 Introduction

As part of the Supercomputer SuperNet (SSN) project at UCLA, Jet Propulsion Laboratory and the Aerospace Corporation, a network prototype to interconnect supercomputers situated across campus and metropolitan areas is being designed and developed [2, 9, 8]. SSN has a two-level architecture (see Fig. 1) in which several high-speed, wormhole routing local area networks are interconnected via an optical backbone network. To accommodate connection oriented applications (such as scientific visualization, terrain rendering, etc.) in such a network, we evaluate alternative methods to support Quality of Service (QoS) traffic.

Wormhole routing [14] is a modification of the virtual cut-through method [7] for communication in computer networks. In **virtual cut-through**, a packet is forwarded onto the required output port at an intermediate node (switch) as soon as the head of the packet is received, if the required output channel is not busy. Upon blocking, the blocked

*This work was supported by the USDOD ARPA/CSTO under Contract DABT63-93-C-0055. The Distributed Supercomputer Supernet - A Multi Service Optical Intelligent Network.

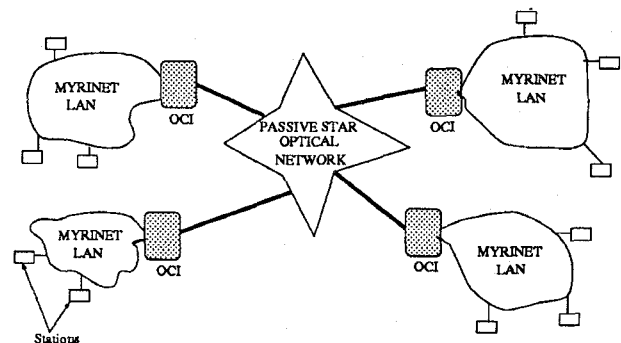


Figure 1. The two-level architecture of SSN. The optical backbone network is based on a physical passive optical star topology. The high-speed wormhole routing LANs are Myrinets. The Optical Channel Interface (OCI) connects the electronic LANs to the optical backbone network.

packet is stored in its entirety at the input port buffer. Instead of storing the blocked packet entirely at the input port buffer, in **wormhole routing**, the packet (or worm) composed of a variable number of flow control digits or flits¹ is stored across several intermediate nodes as a result of backpressure flow control. The advantage of virtual cut-through (over the traditional store-and-forward approach) is latency reduction. Consequently, wormhole routing has been used extensively for internal communication in multiprocessor computers where low latency is of prime importance [14]. This paradigm is now being extended to high-speed local area networks. Myrinet [4] is an example asynchronous, high-speed wormhole routing LAN. Myrinet uses wormhole routing, source routing and backpressure flow control to achieve low latency while providing high bandwidth (640 Mbps). While Myrinet links span only 25 meters or so, SSN is extending the reach of such networks to the campus and metropolitan areas by employing an optical backbone network.

The main goal of the SSN project is to provide support for distributed supercomputing. Target applications include

¹Generally, a flit is a byte.

scientific visualization, distributed memory parallel supercomputing, video display walls [9] etc. These applications generate many different types of traffic with different Quality of Service (QoS) requirements, from low latency datagram traffic (to support fine grain distributed supercomputing) to high-bandwidth connection oriented traffic (uni- and multicasting) with bandwidth guarantee. Thus, QoS support protocols are required. Further, the transport service must be reliable (no worm loss inside the network), scalable and deadlock-free². This is a challenging task since basic wormhole routing supports only low-latency datagram service.

In this paper, we focus on the issue of providing reliable support for connection oriented traffic with QoS parameters. We explore several options that may be employed for QoS support. Some previous work has been done in this area [12, 13], however, the resulting protocols are not reliable (worms may be dropped inside the network to meet delay constraints). In addition, a segmentation-based scheme has been developed [11] that provides best effort bandwidth reservation service for high priority traffic. This scheme does not provide guaranteed throughput and delay performance but rather adds the functionality of preferential bandwidth allocation. We discuss how segmentation may be integrated with our schemes below to aid in traffic shaping for our statistical multiplexing methods.

We assume in this paper that traffic with guaranteed QoS requirements is connection oriented. The desired grade of QoS is specified at connection establishment time, and connections may be refused if the required QoS parameters cannot be guaranteed. The definition of QoS parameters is application dependent. Some connection oriented applications require guaranteed (average or peak) bandwidth. Some others require that the end-to-end delays be guaranteed. Some may just require some bounds on the end-to-end delay jitter (i.e., the time difference between when the destination should receive the message and the actual time at which the destination receives the message). We examine several options to provide QoS support.

A simplistic approach for providing QoS support in high-speed wormhole routing networks is to create on demand a dedicated physical path (consisting of links from the network topology) for each connection. This would guarantee bandwidth and provide desirable delay and delay jitter bounds. However, it requires that the network topology be very dense (in order not to disconnect the network when links are dedicated to guaranteed connections). It also would lead to link capacity under-utilization. Thus, this approach is considered impractical and is not pursued further. More viable approaches, namely, use of a separate QoS subnet, a virtual synchronous overlay and virtual channels are discussed in sections 2,3 and 4 respectively.

2 QoS support via a Separate Subnet

The high-speed wormhole routing network can have an arbitrary topology consisting of several asynchronous, non-blocking crossbar switches. Hosts connect to the network via a host-interface card. If the links and switch ports in the network are inexpensive relative to host interfaces (as in Myrinet), it becomes cost effective to add more links and switches to the topology to create **two separate subnets** -

²Wormhole routing networks are prone to deadlocks [6], thus, some deadlock avoidance routing scheme must be used [15].

one to carry QoS traffic exclusively, the other dedicated to carrying datagram (non-QoS) traffic (see Fig. 2). The links of the QoS subnet would multiplex QoS traffic originating at several different hosts. For such a scheme to effectively support QoS requirements, three basic issues must be considered.

The first issue is the acceptance of connection-oriented QoS traffic into the network. Call admission and control is required since this approach provides support to QoS traffic via statistical multiplexing. To this end, a centralized (or distributed) call admission agent on the network (with complete information of the state of links on the QoS subnet) receives a request for setting up a QoS traffic connection with some specified parameters from the source host. Upon receiving the request, the agent determines a suitable route on the subnet that would satisfy the QoS traffic parameters. If the subnet cannot provide support for a new QoS connection request, the source host is informed and then the source could either wait until a later time and retry or use the datagram subnet (if appropriate). If the QoS connection request is accepted, the source host is informed and it starts transmission. Upon completion of the data transfer, the source host would inform the call admission agent and the agent would update its view of the QoS subnet link status.

The second issue is the behavior of the source host after it has been granted a QoS connection on the subnet. The source host must comply with the parameters that were approved by the call admission agent in order not to interfere with other QoS connections using the same links. One way to do this is to use pacing wherein the source has a mechanism that would only allow some predetermined number of flits to be transmitted per time period from the host. Segmentation of worms into predetermined sizes could also be done in conjunction with pacing to ensure that the host injects only as much traffic as the call admission agent approved [11].

The third issue is the number of interfaces a host would require when the separate subnet approach is employed. Suppose that the host has only one interface connecting it to both subnets. On the transmit side, the host can schedule its own transmissions into the network (and possible, exercise flow control on non-QoS traffic) so that QoS is maintained for the connection oriented traffic. On the receive side, however, it is possible that a non-QoS worm being received by the host blocks a QoS worm destined to the same host. This could introduce delays for the QoS worm. If the host had two interfaces (one per traffic class), this problem would not arise. However, this would practically double the cost of the network since host interfaces dominate overall cost. Another approach is to account for the worst case non-QoS traffic interference on the single host interface at call setup time. For instance, referring to host *G* in Fig. 2, an incoming QoS stream into *G* must share the interface in a round robin fashion with two non-QoS links at the attached crossbar switch. Assuming the same worm size for all links, this leads to a fair 3-way sharing of the host interface of *G*. Bandwidth allocation and delay performance can thus be precomputed by the call admission agent at connection setup time. If the interfering non-QoS traffic is controlled by TCP at the transport level (e.g., a file transfer application), the receiving host can limit the interference by reducing the TCP window for non-QoS connections. It may also request a reduced TCP message size in order to bound delay jitter. Finally, the interference problem would be greatly simplified if the crossbar switches supported privatized round robin service

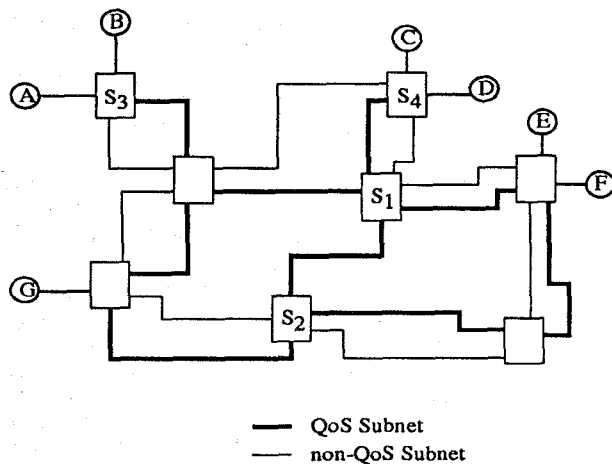


Figure 2. A sample network topology with two separate subnets - one for QoS traffic and one for non-QoS (datagram) traffic. The links between the hosts and the switches do not belong exclusively to either subnet.

of the input ports (possibly with low priority preemption). Current Myrinet switches, however, do not offer this feature.

3 Support of QoS using a Synchronous Framework

Another alternative for supporting QoS is to impose a synchronous structure on top of the asynchronous network. Constructing this synchronous framework enables control over the blocking at the various crosspoints. Thus, delay bounds and message priorities may be implemented. The tradeoff is that the network is no longer asynchronous and under low traffic load conditions, messages suffer delay due to the overhead of supporting the synchronous protocol.

The framework for integrating both QoS and non-QoS traffic is similar to that of the dedicated channels idea as described in section 2. The difference is that the dedicated channel scheme employs traffic shaping (pacing and segmentation) to provide statistical guarantees for ensuring quality of service requirements. In the synchronous structure, control over the traffic streams is obtained using a timed token control mechanism. Consequently, the synchronous structure can provide tighter delay bounds and bandwidth guarantees. Examples of synchronous protocols that may be built on top of a wormhole routing network include polling systems such as token ring, FDDI [17], and Hyper Token Ring (HTR) [3, 10]. FDDI and HTR support Constant Bit Rate (CBR) and Variable Bit Rate (VBR) by controlling the amount of traffic flow at each node via a timer controlled token scheme. To support QoS requirements, a set of timers at each station is used to limit the amount of information transmitted. Details on the specifics of the timer scheme may be found in [19] for FDDI and [10] for HTR. The delay bounds provided by this mechanism are related to one of the parameters used in the protocols, the Target Token Rotation Time ($TTRT$). This value is used to help limit the transmission time. The FDDI protocol guarantees a worst case delay of synchronous traffic of $2 \times TTRT$ and an average delay

of $TTRT$ at each station [18]. This is also the case for the HTR protocol. The timer scheme is capable of providing good jitter control. The only difficulty is in properly setting the parameters to do so.

To support this timed token protocol, a dedicated core unidirectional ring is embedded in the network. As long as all hosts transmitting and receiving QoS traffic have access to the core ring, the QoS scheme will function. For example, in Fig. 2, a ring may be constructed by eliminating one link (the QoS link connecting S_1 and S_2) from the QoS subnet. Although switches with hosts A, B, C, and D do not reside on the embedded ring, the timed token scheme still functions properly on the core ring. The links connecting switches S_3 and S_4 to the embedded ring are used as a part of the unidirectional ring.

This QoS scheme must deal with a similar issue as that of the dedicated subnet idea of section 2. The number of host interfaces affects system operation. The problem occurs due to the interaction of QoS traffic and non-QoS traffic. Typically, QoS traffic travels on the core ring while non-QoS traffic travels on all links not on the core ring. No contention at the switches in the network fabric occurs because the two traffic types are kept on separate subnets. However, delay bounds for QoS traffic may be compromised at the destination host interfaces. If a host is busy receiving a non-QoS message, a QoS message arriving from the ring network would be delayed an unpredictable amount of time if the host only has one network interface. Consequently, QoS traffic must have preemptive priority at the switching nodes when the host only supports a single interface. An alternative would be to provide two host interfaces at the host node, one for the embedded ring and one for the non-QoS traffic.

To help increase the throughput of non-QoS traffic, the embedded ring may also be used to carry this type of data. If bandwidth is not completely taken up by QoS traffic on the ring, non-QoS traffic may be integrated on the ring network using a multi-class implementation of the timer scheme [19].

A problem with this QoS scheme may be its scalability. Although throughput performance for both FDDI and HTR does not degrade as the physical size and propagation length of the system increases, the potential delay bounds and responsiveness of the network do suffer. The throughput performance is maintained by increasing the $TTRT$ parameter and allowing stations to transmit for longer periods of time when they have captured the token. However, increasing the $TTRT$ parameter has the effect of making the network less responsive and less capable of providing tight delay bounds. Consequently, as the network grows, it may be necessary to split the network into multiple smaller rings connected by gateway nodes. The traffic entering the ring via the gateway is treated as a host on the sub-ring. Simulations for this scheme have been run and the results are promising. A drawback of this scheme is that the gateway node has large buffer requirements and must perform complex packet manipulations to support the synchronous QoS scheme. Currently, methods using less complex gateway nodes are being explored.

Other attributes of this QoS support scheme are that the synchronous network is both deadlock free and fair [10]. Since the system has strict control over routing and access to the medium, these attributes are easily provided.

4 Virtual Channel Based QoS Support

In the Virtual Channel approach, each link is split into two different sets of virtual channels, used for datagram and for QoS traffic respectively. Traditionally, the virtual circuit channel approach has been used in wormhole routing networks to prevent deadlocks [6]. Thus, this QoS support alternative would make most sense if combined with deadlock prevention [15].

In the virtual channel approach to deadlock prevention, each buffer associated with the input port of each switch is split into several disjoint buffers. The link between the upstream node and this input port is then treated as a collection of several virtual channels each with separate backpressure flow control.

The use of virtual channels allows worms to be interleaved. As a result, in [5], Dally proposed the use of virtual channels as an efficient way of controlling the critical network resources of link bandwidth and input buffer space. Further, he proposed a priority-based deadline scheduling arbitration scheme (on a per flit basis) to meet end-to-end latency requirements of the application. The problem with an arbitration scheme that works on a per flit basis is that the switch design becomes more complex and the switching latency is increased. In [13], several priority mapping schemes (allocation of virtual channels based upon the (perceived) delay of the incoming worm) along with round-robin and priority arbitration for the physical link were considered. Extensions to this work were considered (along with a worm dropping algorithm) in [12].

Our approach differs from previous approaches in that the allocation of a virtual channel to an incoming worm is fixed as per the deadlock-free routing policy [15]. Given this scenario, we propose to combine deadlock prevention and QoS support in the following way. We use two different sets of virtual channels – one for transporting datagram traffic only and the second for transporting QoS traffic only; and use a priority mechanism to give priority to QoS traffic within the network. One priority mechanism that we propose is the **non-preemptive priority** mechanism in which the arrival of a worm to the QoS set of virtual channels does not immediately cause it to be transmitted on the desired outgoing link. The worm currently being transmitted on the desired outgoing link (either QoS or datagram) is transmitted until either the tail of the worm clears the connection or the worm is blocked at a node further downstream. Then, the QoS set of virtual channels are scanned before the datagram set of virtual channels in order to schedule a worm for transmission on the outgoing link. It is easy to envisage a **preemptive priority** mechanism in which a worm arriving at the QoS set of virtual channels preempts a datagram worm (but not another QoS worm).

Implementation of the non-preemptive priority and preemptive priority mechanisms in the switches is possible only if the switch has intelligence. The node needs to monitor all traffic passing through it and be able to schedule QoS traffic and non-QoS traffic as required by the protocols. The preemptive priority protocol is harder to implement than the non-preemptive protocol as the switch has to check for the arrival of a QoS traffic worm at any of the input ports prior to transmission of the next non-QoS flit from the output port.

The advantage of the virtual channel approach to supporting QoS traffic outlined above is that the network appears the same for both types of traffic (i.e., QoS and non-QoS traffic are integrated). The intelligent switches allocate the

link bandwidth resource as required to support QoS traffic. The virtual channel scheme can be used to provide bounds on delay jitter as shown by the results. Bandwidth guarantees are provided by employing a call admission agent as described earlier.

5 Results

In this section, we present some selected simulation results for the options described earlier. Due to space constraints, we present simulation experiments only for two options – synchronous system and the virtual channel approach. More comprehensive results are reported in [10].

5.1 Synchronous QoS support

In this section, simulation results for a four station unidirectional ring are presented. The HTR simulator is written in C. Each host is capable of transmitting both synchronous data as well as all three classes of asynchronous data. Class 1 is assumed to be the highest priority asynchronous traffic class and Class 3 is assumed to be the lowest. The stations are spaced 15m apart and are capable of transmitting at 640Mbps.

In the following results, only CBR traffic is considered. The synchronous traffic corresponds to a voice stream. The actual bandwidth required for a single voice conversation is very small compared to the total available bandwidth offered in HTR. Consequently, to provide a substantial amount of voice load we allow more than one voice channel per source host. The number of voice channels per station is C_v . Voice packets are of fixed length, L_v and are generated at fixed time intervals of P_v per station. The total voice load of the network G_0 is:

$$G_0 = \left\{ \frac{C_v L_v}{P_v} \right\} N_v \quad (1)$$

N_v is the number of voice sources in the network.

In the four station ring, there are sixteen possible source destination pairs. Each is considered a separate message type. All stations transmit asynchronous traffic. The load of asynchronous traffic at each station is distributed between classes 1, 2 and 3 according to the ratio 3:2:1, respectively. Only 3 message types support both voice and asynchronous traffic, (0,2), (2,0), and (1,3). L_v is set at 3200 bits and each voice packet arrives every 50ms. All other traffic has exponential length with a mean of 3200 bits and arrives according to a Poisson distribution. Offered load includes both synchronous traffic as well as asynchronous traffic. The set of transmission configurations are chosen as if the stations were transmitting a symmetric load across all message types. This provides good throughput performance (see [10] for more details). The $TTRT$ is set such that the average visiting time to each configuration is 25ms and the maximum is 50ms.

The offered load versus throughput for 10% voice traffic is shown in Fig. 3 and for 50% voice traffic in Fig. 4. As the voice traffic load increases, the amount of bandwidth available to the asynchronous traffic drops. When the network is heavily loaded, only class 0 (synchronous traffic) and class 1 are able to use the network. The delay performance for synchronous traffic is acceptable across all loads. Even under high loads such as $\rho = 3.0$ (with 50% of the total load coming from voice traffic), the voice channels receive access to the ring every 18.6ms with a standard deviation of 2.2ms. The results show that HTR with support

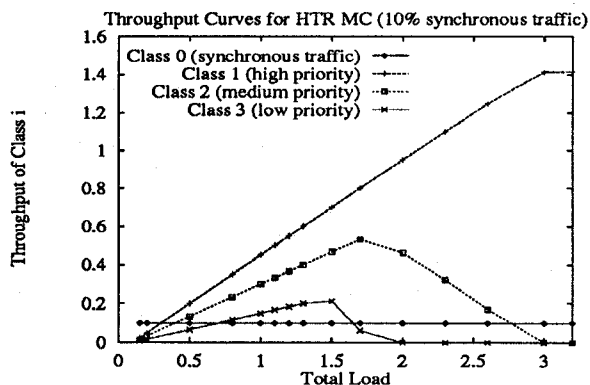


Figure 3. Performance of HTR handling both synchronous traffic and 3 classes of asynchronous traffic. Class 0 is synchronous traffic and Class 1 is the highest priority asynchronous traffic type. 10% of the total load comes from Class 0 traffic which is modeled as a simple voice source.

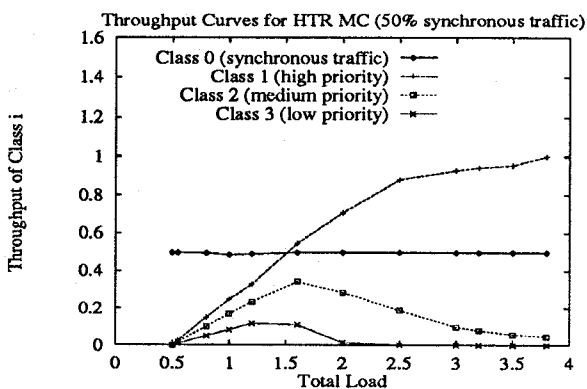


Figure 4. Performance of HTR handling both synchronous traffic and 3 classes of asynchronous traffic. Class 0 is synchronous traffic and Class 1 is the highest priority asynchronous traffic type. 50% of the total load comes from Class 0 traffic which is modeled as a simple voice source.

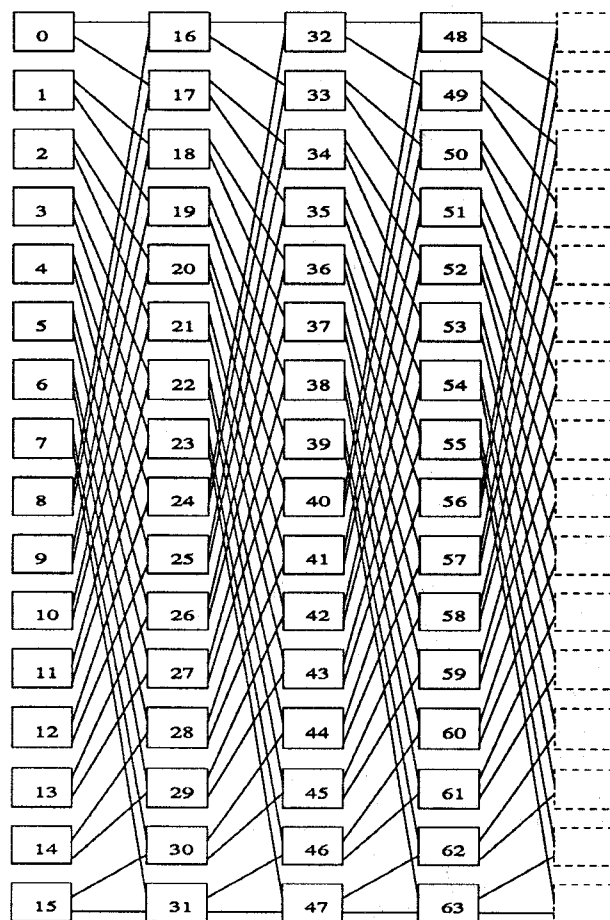


Figure 5. The 64-node bidirectional shuffle net topology. The nodes are numbered from 0 to 63. The last column represents the same nodes as the first column.

for synchronous traffic is capable of providing bandwidth guarantees and strict delay bounds.

5.2 Virtual Channel Approach

The virtual channel simulation model was written in Maisie [1] – a C-based, message-passing, discrete-event simulation language. The virtual channel approach was simulated on a 64-node bidirectional shuffle net topology (see Fig. 5) [16]. Each node is an asynchronous, wormhole-routing switch and has one host connected to it. In the simulation, the link bandwidth was 640 Mbps (the speed at which Myrinet runs) and the links were assumed to be 1 Km long (with a propagation delay of 400 bytes). We simulated both Constant Bit Rate (CBR) and Variable Bit Rate (VBR) traffic types. The CBR traffic source generates worms of length 1000 bytes with a worm interarrival time (from the head of one worm to the head of the next worm) of 10000 bytes. The VBR traffic source simulated transfer of MPEG encoded video frames (from a trace file with frame sizes for an advertisement) at the rate of 30 frames per second. The simulation was run long enough to transmit 1000 frames.

Source routing was employed. Four virtual channels

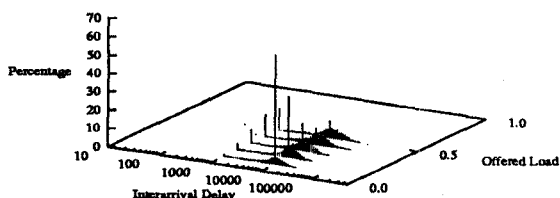


Figure 6. A three-dimensional plot of the Interarrival Delay histogram at the destination for one CBR connection (between node 0 and node 27) for different background non-QoS (datagram) loads. Note the log scale for the interarrival times at the destination.

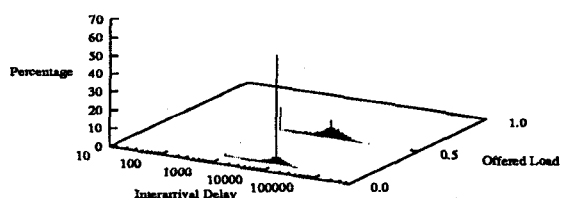


Figure 7. A three-dimensional plot of the Interarrival Delay histogram at the destination for four CBR connections for different background non-QoS (datagram) loads. Note the log scale for the interarrival times at the destination.

were simulated per physical channel (each with its own STOP/GO protocol) in the topology to enable shortest-path deadlock-free routing. This result is true for any bidirectional shufflenet [15]. Four connections were simulated. Connection 1 was between node 0 (source) and node 27 (destination) which has a path length of 6 hops (with a choice of 10 alternative shortest paths at the source). Connection 2 was between node 6 and node 34 which has a path length of 5 hops (with a choice of 2 alternative shortest paths at the source). Connection 3 was between node 11 and node 23 which has one path of length 2 hops. Connection 4 was between node 15 and node 52 which has a path length of 6 (with a choice of 10 alternative shortest paths at the source). Non-QoS traffic was simulated in the background from and to hosts that did not have a QoS connection described above. Non-QoS traffic had a geometrically distributed worm length with an average of 5000 bytes and a maximum of 10000 bytes. In the graphs, we plot the offered load as the fraction of time that each host is busy transmitting non-QoS traffic.

First, we simulated the case where QoS and non-QoS traffic are carried on the same virtual channels (i.e., the network does not distinguish between the two types of traffic). Figs. 6 and 7 show a plot of the histogram of interarrival times at the destination for CBR traffic with only 1 active connection (between nodes 0 and 27) and with all 4 connections active, respectively. The point to note from these graphs is that the tail of the distribution is rather short (note the log scale for delay in byte times). Thus, even when QoS and non-QoS traffic are mixed, the interarrival time between QoS (CBR) worms at the destination are tolerable, though as the background non-QoS traffic load increases, a smaller fraction of worms get to the destination with the same interarrival time as at the source. For example, in Fig. 6, about 50% of the worms get to the destination with the same interarrival time as at the source (i.e., 10000) with a background datagram load of 0.1. This reduces to about 10% with a background datagram load of 0.5.

Next we simulated the case where QoS traffic and non-QoS traffic are carried on different sets of virtual channels

with the non-preemptive priority mechanism between the two sets (as described in section 4). Figs. 8 and 9 plot the interarrival delay histograms at the destination(s) for one CBR connection and for four CBR connections, respectively. By comparing with Figs. 6 and 7, we notice that the non-preemptive priority mechanism improves slightly the delay jitter performance of QoS traffic.

We next show the results for the VBR traffic case. Here we plot the delay jitter histograms at the destinations. The delay jitter is calculated as the difference between the interarrival time of two consecutive worms as noted at the destination and at the source. Thus, if the delay jitter value is 0, then the destination received the worm with the same interarrival time as at the source. In Figs. 10 and 11, we plot the delay jitter histograms for one VBR connection and for four VBR connections, respectively. We see that a high fraction of worms (almost 70% even at high background non-QoS traffic load of 0.5) get to the destination with zero delay jitter. Thus, the mechanism of two separate sets of virtual channels with the non-preemptive priority between the two sets provides adequate support for VBR traffic. Two sets of preemptive priority experiments (not reported here) show 100% delivery of zero delay jitter.

In conclusion, we can say that the use of two different sets of virtual channels for QoS traffic and non-QoS traffic along with the use of preemptive priority provides strong support to QoS traffic. Of course, the switches need to be intelligent for this scheme to be implemented (we note that virtual channel support is not available in Myrinet switches at present).

6 Conclusions

High-speed wormhole routing networks provide natural low-latency, high-bandwidth support for datagram traffic. Providing support for QoS traffic, on the other hand, is a challenging task. In this paper, we have presented and evaluated several different approaches to this problem. Dedicating a subnet to carry QoS traffic exclusively is an approach that when combined with pacing and call admission control can support QoS traffic. Imposing a synchronous frame-

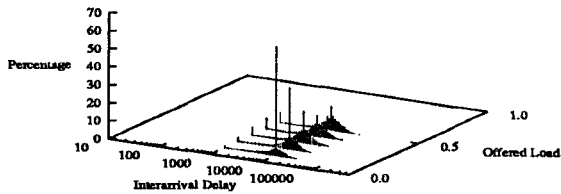


Figure 8. A three-dimensional plot of the Interarrival Delay histogram at the destination for one CBR connection (between node 0 and node 27) for different background non-QoS (datagram) loads when non-preemptive priority is used to give priority to the QoS set of virtual channels. Note the log scale for the delay jitter in byte times at the destination.

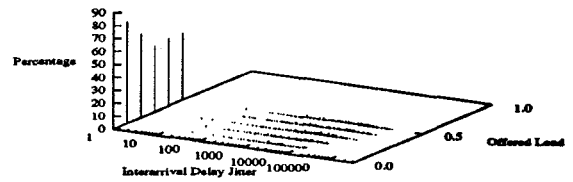


Figure 10. A three-dimensional plot of the Delay Jitter histogram at the destination for one VBR connection (between node 0 and node 27) for different background non-QoS (datagram) loads with non-preemptive priority to QoS Virtual Channels. Note the log scale for the delay jitter in byte times at the destination.

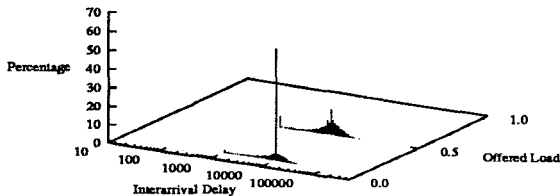


Figure 9. A three-dimensional plot of the Interarrival Delay histogram at the destination for four CBR connections for different background non-QoS (datagram) loads when non-preemptive priority is used to give priority to the QoS set of virtual channels. Note the log scale for the delay jitter in byte times at the destination.

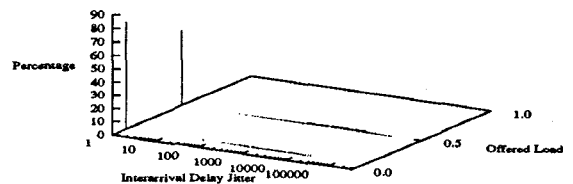


Figure 11. A three-dimensional plot of the Delay Jitter histogram at the destination for four VBR connections for different background non-QoS (datagram) loads with non-preemptive priority to QoS Virtual Channels. Note the log scale for the delay jitter in byte times at the destination.

work on top of the asynchronous high-speed network (via a HTR, FDDI etc.) provides support for guaranteed bandwidth and delay applications. Both approaches can benefit from switch priorities in order to reduce interference of QoS and non-QoS traffic at the host interface.

The use of virtual channels (one set dedicated to QoS traffic and one set to non-QoS traffic) along with a priority mechanism (especially the preemptive priority mechanism) was also shown to be an effective approach to supporting QoS traffic. It becomes even more attractive when combined with deadlock-free shortest path routing (as considered in this paper).

References

- [1] R. Bagrodia, K. Chandy, and J. Misra. A Message-Based Approach to Discrete-Event Simulation. *IEEE Transactions on Software Engineering*, 13(6), June 1987.
- [2] N. Bambos et al. The Supercomputer Supernet (SSN): A High-Speed Electro-Optic Campus and Metropolitan Network. In *Proceedings of SPIE 1996 Conference on Optical Interconnects in Broadband Switching Architectures*, January 1996.
- [3] N. Bambos and A. Nguyen. Queueing Dynamics and Throughput of Ring Networks with Spatial Reuse. In *Proceedings of the 31st Allerton Conference on Communications, Control and Computing*, pages 576–587, October 1993.
- [4] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su. Myrinet: A Gigabit-Per-Second Local-Area Network. *IEEE Micro*, 15(1), February 1995.
- [5] W. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.
- [6] W. Dally and C. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.
- [7] P. Kermani and L. Kleinrock. Virtual Cut-through: A New Computer Communication Switching Technique. *Computer Networks*, 3(3):267–286, September 1979.
- [8] L. Kleinrock, N. Bambos, J. Cong, E. Gafni, M. Gerla, and L. Bergman. OPTIMIC: A Scalable Distributed, All-Optical Terabit Network. *Journal of High Speed Networks*, 4(4):407–424, 1995.
- [9] L. Kleinrock et al. The Supercomputer Supernet Testbed: A WDM Based Supercomputer Interconnect. *Joint issue of IEEE Journal on Selected Areas in Communications and IEEE/OSA Journal of Lightwave Technology*, 14(5), June 1996.
- [10] B. Kwan, N. Bambos, and A. Nguyen. Hyper Token Ring: A high speed ring network with spatial reuse. UCLA Technical Report ENG-96-156, Electrical Engineering Dept., UCLA, 1996.
- [11] B. Kwan, P. Hu, N. Bambos, L. Kleinrock, J. Touch, and H. Xu. Segmentation-based Bandwidth Reservation in Wormhole Routing Networks: A Performance Study. In *Proceedings of International Conference on Computer Communications and Networks*, October 1996.
- [12] J-P. Li and M. Mutka. Priority Based Real-Time Communication for Large Scale Wormhole Networks. In *Proceedings of International Parallel Processing Symposium*, pages 433–438, May 1994.
- [13] J-P. Li and M. Mutka. Real-Time Virtual Channel Flow Control. In *Proceedings of IEEE 13th Annual International Phoenix Conference on Computers and Communications*, pages 97–103, April 1994.
- [14] L. Ni and P. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, February 1993.
- [15] P. Palnati, M. Gerla, and E. Leonardi. Deadlock-Free Routing in an Optical Interconnect for High-Speed Wormhole Routing Networks. In *Proceedings of 1996 International Conference on Parallel and Distributed Systems*, pages 256–264, June 1996.
- [16] P. Palnati, E. Leonardi, and M. Gerla. Bidirectional Shufflenet: A Multihop Topology for Backpressure Flow Control. In *Proceedings of 4th International Conference on Computer Communications and Networks*, pages 74–81, September 1995.
- [17] F. Ross. An Overview of FDDI: The Fiber Distributed Data Interface. *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, September 1989.
- [18] K. C. Sevcik and M. J. Johnson. Cycle Time Properties of the FDDI Token Ring Protocol. *IEEE Transactions on Software Engineering*, SE-13(3), March 1987.
- [19] P. Werahera and P. Jayasumana. Fiber Distributed Data Interface: Throughput Evaluation with Multiple Classes of Traffic. *IEEE Transactions on Communications*, COM-42(2/3/4):499–510, February 1994.