Guaranteed Delay Bounds in Interconnection Networks

Jason Kassoff, Diana Pang, David Wang

***Abstract* — Guaranteeing bounds on end-to-end delay through an interconnection network is a hard problem. The problem can be defined in terms of its constituent sub-problems: traffic characterization, channel establishment, topology and route selection, flow control, and scheduling. This paper looks at some of the research that has sought solutions to bounding delay, mainly in the areas of channel establishment, flow control, and scheduling. Beginning from the optimization of the multiprogram scheduling problem, researchers adapted those results for use in WANs, and devised methods for channel establishment and flow control. Recently, bounded delay in multiprocessor networks has received some attention, and initial research has focused on adapting the results of the WAN related research. The above body of research is described and critiqued.**

1. Introduction

The problem of guaranteeing bounds on delay and jitter in wide area networks has been a topic of research for many years. As digitized voice and video has found its way onto wide area networks (WANs), much effort has gone into designing the networks to provide real-time end-to-end delay bounds. Because the real-time traffic on these networks typically arrives with predictable rates over long periods of time, most solutions tolerate long delays in establishing a channel in order to guarantee a bound on latency after the channel has been established.

Although multiprocessor interconnects have requirements that differ from those of WANs in many ways, they also have a need for guaranteed delay bounds in some real-time applications. Furthermore, even in applications that are not "real-time," the performance benefits of parallel processing are dependent in part on keeping network delay bounded and predictable. Several researchers in recent years have been adapting delay bounding schemes, which were originally developed for WANs, for use in multicomputer and multiprocessor systems.

In this paper we describe the problem of guaranteeing delay bounds in interconnection networks, examine some of the solutions that have been proposed, and suggest areas where we believe future work should be done.

2. Problem Description and Overview

To solve the problem of bounding end-to-end delay in an interconnection network, we must address the following sub-problems:

- *Traffic Characterization*: When and how can we expect traffic to arrive?
- *Channel Establishment*: How are delay bounds calculated, and what conditions must be met to guarantee these bounds?
- *Topology and Route Selection*: How does a message traverse the network from source to destination?
- *Flow Control*: When is it okay to send to the next node?
- *Scheduling*: How does contention over scarce resources get resolved?

In the following sections we discuss some of the work that has been done to address these sub-problems.

2.1 Traffic Characterization

In order to make any hard guarantees on delay, the network needs to know how much traffic will be entering and where it will go. This need is obvious when we consider an extreme case, such as packets arriving at a source node at an average rate that exceeds the node's bandwidth into the network. Clearly no guarantee can be made in this case, because delay will grow steadily (and without bound) as more and more traffic backs up waiting to enter the network.

A more interesting example better demonstrates the need to accurately model the traffic entering the network. Suppose several source nodes can easily handle their incoming traffic, even during bursts, but when these bursts arrive at an internal node simultaneously, they overload its capacity. In this case, a worst case calculation based on an accurate characterization of the traffic could have predicted this problem and prevented such an over-allocation.

Modeling traffic well is one of the hard problems encountered by researchers. In the papers we examined, traffic arrival is assumed to follow fairly simple models, such as a linear bounded process or a probabilistic distribution. The problem of bounding delay becomes much more difficult if arrivals are assumed to be correlated instead of independent processes.

If traffic does not follow the simple models assumed above, then an admission control policy can regulate the traffic that is destined for the network [3]. For example, two goals of an admission control policy could be: 1) reserve a minimum portion of network bandwidth for best-effort (non-real-time) traffic, and 2) prevent real-time traffic from entering if it will cause existing real-time messages to violate their delay bounds.

2.2 Channel Establishment

Once incoming traffic at a given node has been characterized, the network must allocate sufficient resources to guarantee that data will reach the destination node from the source in a fixed amount of time. Resources can include buffers, switches, and channels, depending on the flow control method and router architecture used. The process of finding these resources, determining whether they will be sufficient to meet a given delay bound, and reserving them so that future calculations will reflect this new load, we call channel establishment.

In order to simplify this step, many proposed solutions use static source routing and store-and-forward (SAF) flow control. Innovations in channel establishment usually focus on improving the speed or accuracy of the delay calculations based on available resources. This is important, because while an overly pessimistic calculation might be easy to implement and fast to execute, it can cause costly storage and bandwidth in the network to go unused.

2.3 Topology and Route Selection

Topology and route selection determine where a packet needs to go at any point while it is in the network. While a delay bound can be calculated for an arbitrary topology and route, that bound will be higher for some topologies and routing algorithms than for others. For example, a simple butterfly or a dimension-order routed mesh tend to experience bottlenecks, where only a fraction of a resource can be allocated to each of the channels which are using it at a given time.

Most of the papers below concern themselves with whether or not a bound exists and how to calculate it, rather than with the problem of how to minimize the bound. Most of them assume static source routing to simplify their models, by considering a single eligible path and testing its ability to guarantee delay within a given bound. Dynamic routing could reduce average latency, but it would make guaranteed delay bounds difficult to calculate and perhaps overly pessimistic.

The topologies chosen for study are often very simple, such as 1D or 2D meshes, because interaction between traffic streams in more general topologies can be quite complex. Some researchers base their work on an arbitrary topology, where the goal is to find the best bound given a set of abstract resource constraints, regardless of factors such as topology that led to the constraints.

2.4 Flow Control

Once resources have been reserved along a route, each packet knows where it needs to go, and it has a guarantee that it will reach its destination in a bounded amount of time. This does not necessarily mean that a source can send its packets into the network at any time. Flow control is needed to tell nodes cycle-by-cycle if resources (usually buffers) are available in the downstream node. For example, two bursty streams of data that pass through the same node could overrun its input buffer if they both experience bursts simultaneously and send all of their packets on immediately.

Most of the schemes that we discuss below use SAF flow control for their real-time traffic, because it simplifies the delay bound calculations. In addition, most of them assume rate-controlled traffic, meaning that the incoming packets will not violate the maximum rates and burst lengths which were assumed during channel establishment. With these two conditions, flow control becomes trivial, because the resources reserved for a real-time channel are guaranteed to be available by the time the next packet arrives.

Note that this simplicity comes at a cost. First, SAF flow control requires packet-sized buffers and latency that scales linearly in both packet length and number of network hops. Second, when traffic is not as dense or as bursty as it could be on the real-time channels, then resources are tied up unnecessarily. To mitigate the effects of these problems, some schemes use virtual channels with wormhole flow control for their best-effort traffic. This class of traffic can use the left-over bandwidth and does not have to suffer the packetization delays or increased buffer sizes imposed by SAF.

2.5 Scheduling

While flow control deals with the availability of resources for packets entering a node, scheduling handles the allocation of resources (e.g. switches) to packets as they leave a node. The problem of scheduling, also called arbitration, is to determine which of many contenders should leave the node next. This problem is closely tied to that of channel establishment, because delay calculations during the channel establishment phase are based in part on assumptions of how long a packet could be delayed while waiting to leave a node.

Scheduling has been the focus of many researchers' efforts, in attempting to provide tighter delay bounds with higher resource utilization. The fundamental tradeoff here is between effective scheduling (i.e. tight delay bounds, high resource utilization) and implementable schedulers (i.e. low latency, small storage requirements). In high-speed networks, the need for simple scheduling algorithms in routers is especially key, since router delays become a larger part of end-to-end delay as transmission speeds increase. In our survey of proposed solutions below, we will look at what has been done in the area of scheduling WAN traffic and how some of these results have been adapted for use in multicomputer and interprocessor networks, where latency is much more crucial.

3. Early Work with WAN Networks

In this section, we examine some of the research that laid the groundwork for guaranteed delay bounds. Most of this work focuses on WANs, but the earliest paper we consider actually approaches the scheduling issue as a multiprogram scheduling problem [13]. Next we look at a second mechanism for scheduling with guaranteed delay bounds [4], [14]. Then we consider an early approach to real-time channel establishment. Finally, we present a survey of scheduling techniques used in bounded-delay networks.

3.1 Foundations of Guaranteed Delay Scheduling: EDD

Liu and Layland [13] build the theoretical foundation for realizing bounded delay in real-time applications with different scheduling algorithms. Even though their theorems are based on a multiprogramming environment, the fundamental problem solved is applicable to any real-time resource sharing environment when prioritization among contentious sources is required. Utilization of buffers and channels in interconnection networks fit into this paradigm. Two scheduling algorithms are proposed and proved to satisfy hard limits on delay: fixed priority assignment and dynamic deadline driven. Both algorithms are priority driven and preemptive. That is, the processing of any task can be interrupted by a request from a higher priority task.

The fixed priority scheduling algorithm assigns priorities to tasks according to their request rates, independent of their run times. Tasks with higher request rates are given higher priorities, a scheme called "rate-monotonic priority assignment." By defining the critical instant and deadline of a task, the paper was able to find and prove the least upper bound of processor utilization while meeting hard task deadlines.

In contrast to the fixed priority scheme, the dynamic deadline driven algorithm assigns priorities to tasks based on the deadlines of their current requests. Liu and Layland show that if a set of tasks can be scheduled by any algorithm, then it can be scheduled by the deadline driven scheduling algorithm. Although the do not provide a tight bound for delay, this result guarantees that there are solutions to the practical bounded delay problem.

Out of the papers we surveyed, this paper contributes the first discussion of a "deadline driven" scheduling algorithm and first proofs that such an algorithm can be used to achieve fixed bounds on scheduling delay. Many more recent papers on network scheduling [5], [12] still rely on this foundation, referring to deadline driven scheduling as "earliest due date" (EDD) or "earliest deadline first" (EDF).

3.2 WFQ: Alternative Bounded Delay Scheduling

Demers, Keshav, and Shenker [4] describe a scheduling algorithm called weighted fair queuing (WFQ). This algorithm first assigns a weight to each incoming channel. Then, when multiple channels are contending for a resource, each one is granted access in proportion to its weight divided by the sum of the weights of all current contenders. The goal in presenting this scheme was to control congestion effectively, not to guarantee bounded delays.

Parekh and Gallager [14] were the first to show that WFQ could be used to guarantee scheduling delay bounds. They prove that if incoming traffic can be modeled as leaky bucket sources[1], then weights can be chosen such that the WFQ algorithm provides guaranteed bounds on scheduling delays. This proof is a key contribution to the study of bounded delay networks, but in the same paper, they acknowledge its limitations. They require that incoming traffic be characterizable by leaky bucket parameters and not change during a session. They also note that the worst-case nature of their analysis result in overly conservative delay bounds, and that much tighter probabilistic bounds could be made.

3.3 Channel Establishment in WANs

Ferrari and Verma [6] modify the EDD algorithm and integrate it into a complete scheme for real-time channel establishment in WANs. This is the key contribution of this paper: a scheme is described, complete with scheduling and flow control mechanisms, for establishing channels with delay bounds. A single round-trip procedure is proposed, which tests and tentatively reserves resources at each node along the channel's path. If resulting reserved resources can guarantee the delay bound requested by the client, then the channel is established, and the guarantee remains in effect until it is disconnected.

As one of the earlier and often cited works in the area of channel establishment, this paper lays out a set of assumptions that guides many other researchers in modeling real-time network traffic. First, they assume that real-time channels will be statically source-routed, which makes resource reservation feasible. Second, they describe a model for the offered load, based on interarrival times and packet lengths. Next, they define two channel types: deterministic channels with hard delay bounds, and statistical channels with delay bounds that must be met within a certain probability. They argue for connection-oriented switching, noting that providing guaranteed delay bounds for a connectionless service would be difficult or impossible. Finally, they establish two performance

[1] *Leaky bucket sources*: sources characterized by parameters (b, r), where b is the bucket size and r is the rate at which packets drain into the network. Packet arrivals must not cause the bucket to overflow.

metrics: the source-to-destination delay bound for a channel's packets, and the maximum packet loss rate.

Ferrari and Verma also make several key observations about the tradeoffs associated with establishing channels with bounded delays. First, they note that pessimistic assumptions reduce the maximum number of bounded-delay channels that a given network can support. Second, they recognize and measure the delay overhead associated with channel establishment, scheduling and flow control.

3.4 Survey of Scheduling Algorithms [3]

The following scheduling algorithms have been proposed to support real-time traffic in packet networks. We have divided them into two groups, work-conserving and non-work-conserving. Work-conserving algorithms will not leave the link idle as long as any packet is contending for it. Non-work-conserving algorithms sometimes prevent a packet from using an idle link in order to reduce congestion.

3.4.1 Work-Conserving

- *WFQ* [8]: This algorithm is described in section 3.2. Its main benefit is that it allocate resources fairly, preventing ill-behaved sources from shutting out well-behaved ones.
- *Delay EDD* [6]: An extension of EDD where the scheduler negotiates a service contract with each source. If the source obeys a peak and average sending rate, the server will provide a delay bound. The server sets a packet's deadline to the time at which it should be sent had it been received according to the contract.

3.4.2 Non-Work-Conserving

- *Virtual Clock* [17]: Here each packet is allocated a virtual transmission time, which is the time at which that packet would have been transmitted were the scheduler doing time division multiplexing. Incoming packets from a given source are given virtual service timestamps and get scheduled according to that time.
- *Stop-and-go* [7]: The aims is to preserve the "smoothness" property of traffic as it traverses the network. Time is divided into frames; during each frame, only packets that arrived at the server in the previous frame time are eligible to be sent.
- *Hierarchical Round Robin* [9]: HRR provides several service levels, where each level provides round-robin service to a fixed number of slots. Higher levels get more bandwidth than lower level slots. Since a server completes one round through its slots in each time frame, it can provide a maximum delay bound to the channels allocated to that level.
- *Jitter EDD* [16]: An extension of Delay EDD with bounds on the minimum as well as on the maximum delay. After a packet has been served at each server, it is stamped with the difference between its deadline and actual finishing time. A regulator at the entrance of the next switch holds the packet for this period before it is made eligible to be scheduled.

4. LANs and Interprocessor Networks

The previous section discusses some key papers that deal with proving and implementing hard delay bounds for WAN traffic. More recently, parallel computing applications are beginning to demand the same real-time services in multicomputer and multiprocessor networks. These smaller networks operate on vastly different time scales, use smaller and more regular topologies, and carry much more latency-sensitive traffic. However, in this section we will see that the theoretical framework developed for WANs is applied to LANs and interprocessor networks with some success.

4.1 A Proof of Concept

Kandlur, Shin and Ferrari [10] present methods for computing delay bounds for delivery of messages on real-time channels in multi-hop networks. As an early work in this area, they set many the conventions which later researchers have followed. First, they model traffic as a linear bounded arrival process[2], not unlike the model used by Ferrari and Verma [6] but with different parameters. Next, they assumed a fixed source-routing algorithm. For a topology, they define a system of processor nodes connected by point-to-point dual simplex links, with multiple links per node that can operate in parallel.

After making the above assumptions about traffic, topology and routing, the researchers address the remaining challenges of channel establishment, flow control, and scheduling. In order to simplify the delay computation during channel establishment, they use a worst-case fixed priority scheduling algorithm. During run time (i.e. when the message is passing through the network nodes) they switch to a multi-class version of Ferrari and Verma's [6] EDD scheduling algorithm. To simplify flow control and buffer allocation, they treat packets atomically, employing SAF flow control. In order to avoid overflowing SAF buffers when new channels are added, they adjust the link horizon[3], not allowing packets to advance if they arrive

[2] *Linear bounded arrival process*: arrivals are assumed to be periodic with a minimum logical inter-arrival time between messages, and maximum burst size, message rate and message size.

[3] *Link Horizon* = (logical arrival time) - (earliest packet departure). This represents a limit on burstiness because it prevents packets from being sent on until they are close enough (within this horizon) to the logical arrival time.

earlier than a given amount (i.e. the horizon) before their logical arrival times. This has the effect of reducing the "burstiness" of the traffic seen in the network, which reduces the storage requirements at each node.

The key contribution of this paper is a set of algorithms, described in brief above, for computing guaranteed delay bounds and for dynamically allocating buffer resources. Although no experimental results or objective performance metrics are presented, at the time of publication the authors were building prototype hardware based on the ideas presented in this paper.

4.2 Hardware Implementation

In 1996, Rexford, Hall, and Shin [15] describe an implementation of a router architecture based on the ideas presented in [10]. Their stated goal was to provide performance guarantees for communication in parallel machines, without sacrificing overall network performance for non-time-constrained (best-effort) communication. To simplify the delay bound calculations, they used SAF flow control for time-constrained packets. To minimize the delay and storage overheads associated with SAF, they only supported small, fixed-size packets. Best-effort traffic, on the other hand, has no incentive to use SAF flow control — indeed, it would suffer a longer latency with no benefit — so the router supports wormhole flow control for this class of messages.

To avoid the situation where best-effort traffic could block higher priority time-constrained traffic (i.e. priority inversion), the two types of traffic are assigned to separate virtual channels (VCs). Thus, a time-constrained packet can always preempt a best-effort flit in mid-packet. Best-effort traffic uses round robin scheduling, while time-constrained traffic adheres to a scheduling scheme based on logical arrival times and deadlines.

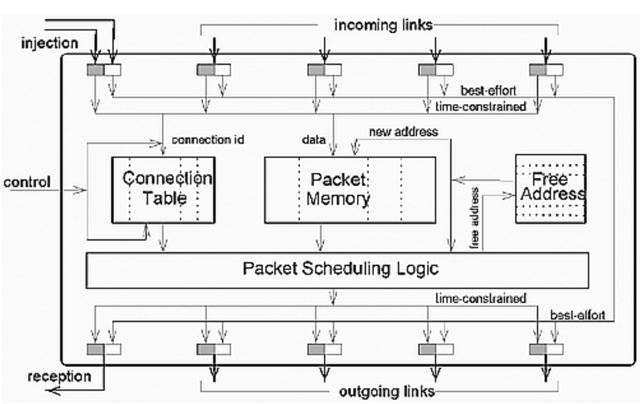

Figure 1

The main contribution of this paper is to show that the algorithms described in [10] can be practically implemented in hardware. The implementation process illustrates the tradeoffs that are needed between design complexity and performance. For example, SAF switching is used for simplicity, even though virtual cut through could reduce average delays. Also, because of its complexity, the scheduling logic for sorting real-time packets is shared between all of the router's outgoing ports, as illustrated in Figure 1.

4.3 An Alternative Approach

Balakrishnan and Ozguner [1], [2] devised a method for providing guaranteed delay bounds which is fundamentally different from the preceding works. They propose using a modified wormhole flow control for all packets. First, a head flit is sent out to establish a channel. If it is blocked by a lower priority packet, it tries to catch the head of this blocking "worm" and kill it, freeing the resources for its own use. If the high priority head cannot catch the head of the low priority packet, then it must wait for the lower priority packet to free the channel before it can continue establishing its route. Thus, the delay of channel establishment is bounded in the worst case by waiting for an entire low priority packet to pass at each node along the route.

There are several drawbacks to this approach. First, the maximum delay described above only applies to the packet with the highest priority. Only one packet can be assigned each priority, so every other packet necessarily has a higher delay bound. Second, a dedicated non-blocking channel is set aside for high priority heads trying to catch and kill lower priority heads. This results in a costly channel that is only used a fraction of the time. Next, this scheme only utilizes a single VC per physical channel. To accommodate multiple VCs per physical link, the network is simply divided into disjoint virtual networks, where the virtual channels do not interact, but instead share the physical link via time division multiplexing. This limitation can lead to significant load imbalance between the disjoint virtual networks.

5. Critique

5.1 Importance of Solving the Problem

Taking a step back, we ask whether guaranteeing end-to-end delay bounds in an interconnection network is an important problem to solve. Certainly some classes of applications, such as live video transmission across a WAN, cannot operate properly without such guarantees. For some other applications which run on parallel computing systems, real-time communication is required to enhance performance or produce correct behavior. In this case, however, the network is a multiprocessor interconnect instead of a WAN.

Thus, an absolute bound on delay is certainly desirable, but it comes at some cost. Specifically, guaranteeing such a bound places stringent constraints on what can be done to reduce the *average* delay, increase throughput, and provide a high level of service to non-guaranteed messages. In certain cases, we cannot tolerate anything less than a guaranteed delay bound, for example in embedded systems controlling airplanes or nuclear power plants. However, there is a large class of applications that would benefit from *minimal*, *probabilistic* delay bounds. In addition, many applications described as "real-time" can in fact tolerate infrequent deadline violations [3].

5.2 Recasting the Problem

If we recast the problem of guaranteeing delay bounds into one of making applications more tolerant of probabilistic delay bounds, then we could focus on reducing overall delays in the network, rather than on computing hard bounds at the expense of increased latency and decreased throughput.

The remaining class of applications, which truly require guaranteed delay bounds, could also be adapted to better fit within network constraints. For example, real-time messages could have fixed lengths and interarrival times as described in [13]. Furthermore, in many cases these applications are critical enough to justify the tradeoff of higher cost for a given level of performance.

5.3 Analyzing the Proposed Solutions

Returning to the original problem of guaranteeing delay bounds, how well do the papers we considered address the key issues? Our survey shows that the issue of resource scheduling was the first to be addressed, beginning with a proof that a guaranteed schedule could be computed under certain traffic conditions [13]. Variations and improvements on scheduling algorithms were presented in [5], [6], [7], [8], [9], [16], and [17].

Channel establishment, in particular delay calculations and buffer reservation, is the other main topic addressed by these researchers. In particular, [6] describes a specific scheme for establishing real time channels in WANs, while [10] outlines a similar method for interprocessor networks.

Several critical issues seem to be missing from this body of research. First, all of the solutions rely on a channel establishment procedure that is itself unbounded. If the network cannot guarantee the delay bound requested by the incoming message, then that message waits until the required resources become available. This amounts to an assumption that real-time traffic is of a streaming nature such that it can wait a long time to establish a channel which it will use for a long time. A good direction for future research would be to find ways to reduce the average delay associated with channel establishment.

Next, we notice that of the five key sub-problems outlined in section 2, only channel establishment, flow control, and scheduling have been experimented with in an attempt to find lower or tighter delay bounds. This leaves novel topologies and routing algorithms as open areas for future research. For example, someone might ask the question of whether certain topologies are fundamentally better suited to guaranteeing delay bounds. At one extreme, non-blocking networks offer completely deterministic delays. Might there be less expensive topologies that approach this ideal?

Finally, some of the more difficult problems associated with bounding delay, described in section 2, merit future work because of their potential to increase overall network performance (or reduce cost). These problems include handling correlated traffic, performing adaptive routing, and utilizing flow control techniques with lower latency and storage requirements than SAF (e.g. virtual cut through or wormhole).

5.3 Conclusions

We have seen that providing guaranteed delay bounds in an interconnection network is a hard problem, and one of critical importance for a certain class of applications. In scheduling, the fundamental tradeoff is between low average delay (work-conserving schemes) and low jitter (non-work-conserving). In channel establishment, the choice is between a simple algorithm with loose bounds on delay and a complex algorithm with tighter delay bounds and thus higher resource utilization.

In many cases, keeping the network cost down is key, in which case high resource utilization would be desirable. In addition, many real-time applications can tolerate jitter as long as a hard delay bounds are met. If these two assumptions hold, then we would select a work-conserving scheduling algorithm, such as delay-EDD, and a channel establishment scheme with bounds as tight as possible to minimize the cost of over-provisioned resources.

Of course, we would prefer to recast the problem as described in section 5.2 above, which would allow us to use a network with probabilistic delay bounds. This would free us from having to provision resources for the worst case traffic and would allow us to reduce average delay by eliminating the channel establishment process.

6. References

[1] S. Balakrishnan and F. Ozguner, "Providing Message Delivery Guarantees in Pipelined Flit-buffered Multiprocessor Networks," in *Proceedings of the Real-Time Technology and Applications Symposium*, June 1996, pp.120-129.

[2] S. Balakrishnan and F. Ozguner, "A Priority-Driven Flow Control Mechanism for Real-Time Traffic in Multiprocessor Networks," in *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, July 1998, pp. 664-678.

[3] D. Clark, S. Shenker, L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network:

Architecture and Mechanism, in *Proc. of the ACM SIGCOMM '92*, pp 106-114, Aug 1992.

[4] A. Demers, S. Keshav and S. Shenkar, "Analysis and simulation of a fair queuing algorithm" in *Proc. ACM SIGCOMM '89*, pp. 1-12.

[5] N. Duffield, T.V. Lakshman, D. Stiliadis "On Adaptive Bandwidth Sharing with Rate Guarantees," *Bell Laboratories Technical Memorandum* 113470-97121507TM.

[6] D. Ferrari, D.C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", in *IEEE Journal On Selected Areas In Communications*, Vol.8, No. 4, 1990, pp. 368-379.

[7] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management, Proc" in *ACM SigComm '90*, September 1990, 8-18.

[8] R. Guerin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions," in *Computer Networks Journal (Elsevier Science)*, vol. 31, no. 3, pp. 169-189, Feb. 1999.

[9] C. R. Kalmanek, H. Kanakia, and S. Keshav, "Rate control servers for very high-speed networks", in *Proc. GLOBECOM '90*, 1990, pp. 12-20.

[10] D.D. Kandlur, K.G. Shin and D. Ferrari, "Real-time Communication in Multi-hop Networks," in *Proc. IEEE Int'l Conf. on Distributed Computing*, 1991, pp.300-307.

[11] K. Knauberand B. Chen "Supporting preemption in wormhole networks" in *Computer Software and Applications Conference, 1999. COMPSAC '99*. Proceedings. 1999 pp. 232 –238.

[12] J. Liebeherr, D.E. Wrege, and D. Ferrari. "Exact admission control in networks with bounded delay services." to appear in *IEEE/ACM Trans. on Networking.*

[13] C.Liu and J. Layland. "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment." In *Journal of the ACM*, 20 pp.46--61, Jan. 1973.

[14] Abhay K. Parekh and Robert G. Gallager. "A generalized processor sharing approach to flow control in integrated services networks: The single node case." in I*EEE/ACM Trans. on Networking*, June 1993 pp. 344-357.

[15] J. L. Rexford, J. Hall, and K. G. Shin, "A Router Architecture for Real-Time Point-to-Point Networks," in *Proc. of the ACM Int'l Symposium on Computer Architecture*, Philadelphia, Pennsylvania, May 22-24, 1996, pp. 237-246.

[16] D. Verma, H. Zhang, and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," in *Proceedings of IEEE TriCom 1991*, pp. 35-43.

[17] L. Zhang, A New Architecture for Packet Switching Network Protocols, PhD thesis, Massachusetts Institute of Technology, July 1989.