Hardware Support for Controlled Interaction of Guaranteed and Best-Effort Communication *

Jennifer Rexford, James Dolter, and Kang Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

Abstract

Real-time communication typically consists of guaranteed *packets that must satisfy their delivery deadlines and* best-effort *packets that can tolerate occasional deadline misses for improved average latency. This paper presents hardware techniques for supporting the coexistence of these two traffic classes in real-time point-to-point networks. A careful selection of routing and switching techniques, coupled with fine-grain arbitration between traffic classes, can allow network adapters to support the diverse performance requirements of best-effort and guaranteed communication. Cycle-level simulations of SPIDER (Scalable Point-to-point Interface DrivER), a network adapter for point-to-point distributed systems, demonstrate the utility of supporting multiple low-level communication policies for different classes of traffic.*

1 Introduction

Point-to-point networks, with their multiplicity of processors and internode routes, provide a natural platform for real-time applications that require both high performance and dependability. While many parallel machines connect processing elements with a point-to-point network [3,6,13], these processors often reside on a single board or chassis, making them especially vulnerable to single-point failures. A network of physically-distributed computers offers the advantage of independent processor and link failures. However, in many distributed systems, a node interfaces to the communication fabric through only one or two ports, as in a CSMA/CD, token ring, or token bus network. In this configuration, one or two media failures can partition the network.

Combining the high connectivity of point-to-point parallel machines with the communication media and protocols of distributed systems results in a hybrid environment well-suited to real-time applications. Real-time communication is typically categorized into two basic classes of traffic, *guaranteed* messages requiring delivery before their deadlines and *best-effort* messages without delivery-deadline guarantees [2,10,18,19]. Distributed real-time systems aim to satisfy the constraints of guaranteed messages, while providing good performance for the best-effort traffic.

This paper addresses techniques for the effective mixing of guaranteed and best-effort traffic by utilizing flexible hardware support for routing and switching. Unlike protocol software, network hardware can exercise efficient, fine-grain control over the interaction of packets. By closely regulating access to the physical links, real-time systems can control the interaction of guaranteed and best-effort traffic at the byte or word level, providing tight bounds on the intrusion of best-effort traffic on guaranteed packets. In optimizing for the performance needs of each class, this hardware can employ different routing and switching strategies to manage the two classes of traffic.

Section 2 evaluates the suitability of various routing and switching schemes for supporting different higher-level performance requirements. The section compares the schemes running on a cycle-level simulation model of SPIDER (Scalable Point-to-point Interface DrivER), a hardware adapter for real-time multi-hop networks [7,8]. Designed as the front-end interface for HARTS [17], SPIDER supports a variety of routing

*The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895 and an Office of Naval Research graduate fellowship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the NSF or ONR.

and switching schemes through flexible, low-level control over the network links. Section 3 presents mechanisms for utilizing such low-level hardware support to regulate the mixing of guaranteed and best-effort traffic. The paper concludes with Section 4.

2 Routing and Switching

Modern routing and switching techniques significantly reduce packet latency by avoiding unnecessary packet buffering, but they also impinge on predictability and control over packet scheduling. With proper hardware support, real-time systems can capitalize on the various routing and switching schemes to improve the quality of service for both guaranteed and best-effort communication.

2.1 Hardware Support

SPIDER, shown in Figure 1, implements programmable routing and switching schemes for best-effort traffic, while facilitating host control over scheduling and resource allocation for guaranteed communication. Designed to reside on the host processor's private memory bus, SPIDER has direct access to the host memory and provides the host with a memory-mapped control interface. The adapter coordinates bidirectional communication with up to six neighboring nodes, with two virtual channels [5] on each unidirectional link.

The programmable routing controller (PRC), a custom integrated circuit, exploits concurrency amongst the virtual channels and provides fair, fine-grain arbitration at the memory and network interfaces [7]. The twelve PRC TXs provide low-level control of packet transmission, while the twelve microprogrammable PRC RXs coordinate packet reception and implement routing and switching policies for in-transit traffic. The PRC TXs and PRC RXs implement the low-level drivers controlling the actual transmitter and receiver devices on the network interface (NI).

The NI performs the media access and flow control for six pairs of AMD TAXI chips [1], providing a low-cost communication fabric of either fiber-optic or twisted-pair interconnect. Each TAXI TX-RX pair forms a bidirectional link to an adjacent node. The NI TX and NI RX control units perform the necessary interleaving of virtual channels to and from the physical links, on a byte-by-byte basis. The NI serves as a hardware wrapper that generates the abstraction of multiple, bidirectional channels between neighboring nodes.

SPIDER treats the outbound virtual channels (NI TXs) as individually reservable resources, allowing the device to support a variety of routing and switching schemes through flexible control over channel allocation policies. Upon receiving the header bytes of an incoming packet, the PRC RX decides whether to buffer, stall, forward, or drop the packet. The PRC RX bases its routing and switching decision on its microcode, the arriving header, and prevailing network conditions. By downloading different microcode to each PRC RX, SPIDER can tailor the low-level communication policies of each virtual channel.

2.2 Communication Policies

The various routing and switching schemes differ in terms of delivery latency, bandwidth utilization, and predictability. Flexible adapter hardware enables the system to tailor communication policies to application requirements and network conditions. The routing algorithm determines which link and node resources are consumed by an in-transit packet. *Static* routing provides a single, deterministic path between each source and destination node, whereas *adaptive* schemes can incorporate network conditions into the routing decision.

For a given source-destination pair, a packet using static routing consumes fixed bandwidth along a predetermined path, making these schemes appropriate for guaranteed communication. While most static routing algorithms generate only shortest-path routes between the source and destination nodes, some adaptive schemes consider *nonminimal* routes in the hope of circumventing network congestion. Adaptive and nonminimal approaches can improve average latency, but at the expense of predictability, making these schemes better-suited to best-effort traffic.

In defining how packets flow through the network, the various switching schemes exercise different resources at nodes along a packet's route. Traditional packet switching requires an arriving packet to buffer completely before transmission to the subsequent node can begin. Buffering the packet after each hop allows the software protocols to directly schedule traffic to provide guarantees [11]. In contrast, cut-through switching schemes, such as virtual cut-through [12] and wormhole [3], allow an incoming packet to begin transmission to the subsequent node prior to complete reception at the current node if the output link is idle. If the packet encounters a busy outgoing link, virtual cut-through switching buffers the blocked packet at the node, whereas wormhole switching stalls the packet in the network until the link becomes available.

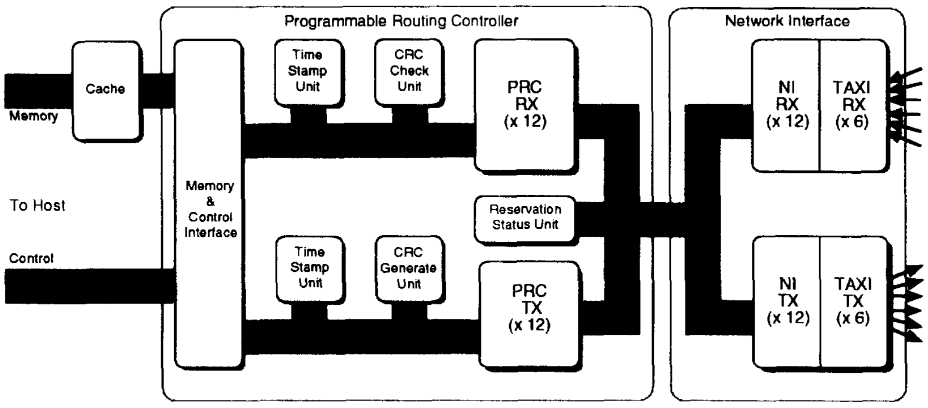


Figure 1: SPIDER architecture

Packet switching consumes predictable network and processing bandwidth, while virtual cut-through *oftentimes* avoids buffering packets at intermediate nodes. Virtual cut-through imparts fewer packets on the protocol software, but limits this software's ability to influence bandwidth scheduling. For example, suppose a guaranteed packet enters an intermediate node well in advance of its deadline. The protocol may wish to detain this packet, even if its outgoing link is free, to avoid overloading the subsequent node unexpectedly.

Wormhole switching, though conceptually similar to virtual cut-through, has quite different characteristics. Stalled wormhole packets form logical queues spanning multiple nodes, complicating packet scheduling. Since a blocked wormhole packet never buffers, it imparts no memory demands on intermediate nodes, but instead consumes unpredictable amounts of channel bandwidth. Figure 2 compares the performance of wormhole and packet switching in simulations of a 6×6 wrapped square mesh (torus).

The simulator [7] includes a cycle-level model of SPIDER that captures the details of flow control, resource arbitration, and microcode execution. Packets use static dimension-ordered routing, with the wormhole packets employing deadlock-free routing on a pair of virtual channels [4, 16]. Each node independently injects 64-byte packets with uniform random selection of destination nodes. As shown in Figure 2(a), wormhole switching results in lower end-to-end latency than packet switching at low loads, but wormhole packets suffer larger latency and delay variance at high loads. Figure 2(b) shows the standard deviation of latency for packets traveling exactly five hops.

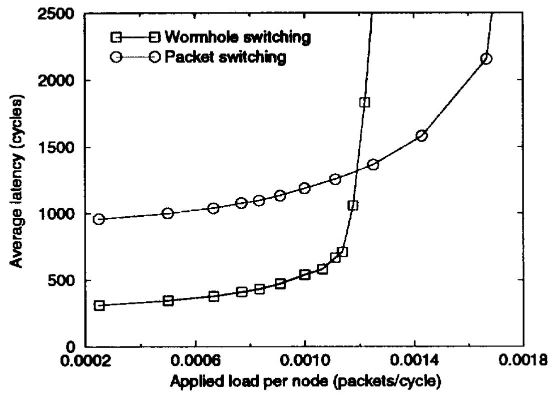
Traditionally, real-time systems have used packet switching and static routing for guaranteed messages, since it is difficult to control the packet delivery time under adaptive routing and cut-through switching schemes. However, best-effort packets could potentially improve their average latency by employing these aggressive schemes. Cut-through switching avoids the delay of buffering the packet, while adaptive routing can increase the possibility of establishing these cut-throughs. Additionally, using cut-through switching for best-effort packets reduces the load these packets impart on the protocol software.

3 Controlled Traffic Mixing

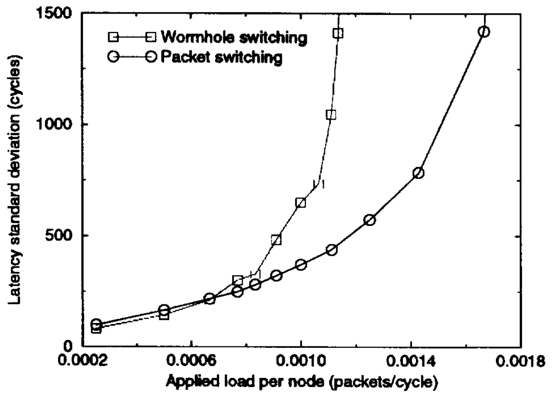
While guaranteed packets can employ static routing and packet switching for predictable performance, best-effort packets can use schemes with less predictability, but better average latency. The effective mixing of guaranteed and best-effort communication hinges on regulating the interaction between these two traffic classes. In particular, the best-effort packets cannot consume arbitrary link, memory, or processing resources while guaranteed packets await service.

3.1 Virtual Networks

Partitioning the best-effort and guaranteed packets onto separate virtual networks can regulate this intrusion. This divides each physical link into multiple virtual channels, where some virtual channels carry best-effort packets and the rest accept only guaranteed packets. Assigning guaranteed and best-effort traffic to different virtual networks, provided by the hardware router, allows best-effort packets to safely



(a) Mean latency (all packets)



(b) Standard deviation (5-hop packets)

Figure 2: Packet switching and wormhole switching

employ adaptive routing and cut-through switching schemes without endangering guaranteed packets.

Packets on separate virtual networks interact only to compete for access to the physical link. Fair, demand-slotted arbitration schemes provide the tight bounds necessary for guaranteed traffic, while allowing best-effort packets to utilize any remaining bandwidth. For instance, SPIDER employs a binary priority tree arbiter [7, 14] to order requests for byte access to the physical links. This provides a tight response time for guaranteed packets, independent of the amount or length of best-effort packets.

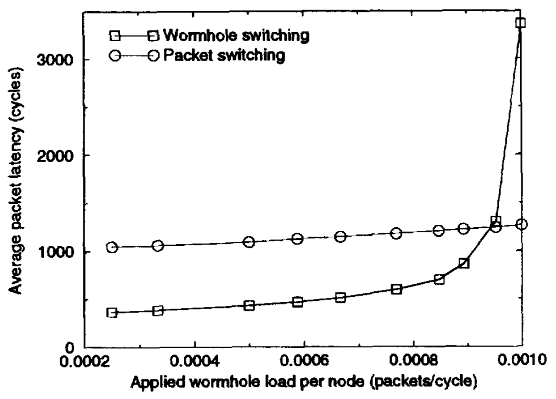
The router can further minimize intrusion on guaranteed packets by imposing priority arbitration between the virtual networks. Priority arbitration enables a guaranteed packet to travel at the same rate through each link in its journey, independent of the number of active best-effort virtual channels. The router can then employ effective scheduling techniques [11, 15] to establish tight delay bounds or bandwidth guarantees. Since priority arbitration varies the service rate for the lower-priority traffic, the best-effort virtual networks could employ adaptive routing to allow these packets to circumvent links and nodes serving a heavy load of guaranteed packets.

Careful selection of routing and switching schemes, coupled with fine-grain arbitration between the virtual networks, allows the traffic classes to share communication bandwidth without sacrificing the performance requirements of either class. Since channel contention in one virtual network does not directly influence the other virtual networks, best-effort traffic can utilize routing and switching schemes with unpredictable consumption of virtual channel resources.

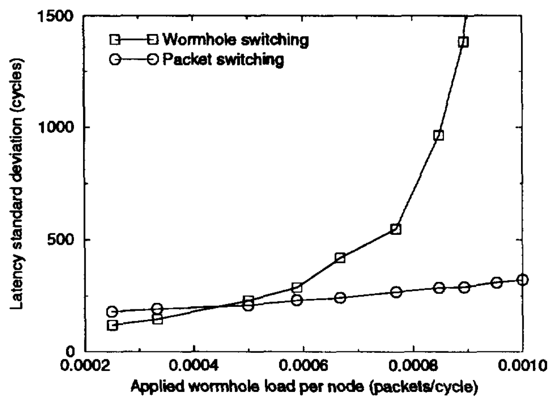
3.2 Wormhole and Packet Switching

The system can limit the resources consumed by best-effort communication to ensure sufficient memory and processing resources for any guaranteed packets. Since wormhole switching does not consume buffer resources at intermediate nodes, the combination of wormhole switching for best-effort traffic and packet switching for guaranteed packets enables effective partitioning of both host and network resources. In this scheme, blocked best-effort packets temporarily stall in their own virtual network instead of consuming resources at intermediate nodes.

Figure 3 shows simulation results for a 6×6 torus carrying a mixture of best-effort and guaranteed traffic on separate virtual networks. The simulations experiment with a SPIDER model that supports three virtual channels on each link, with two allocated to best-effort packets for deadlock-free wormhole routing and one dedicated to guaranteed traffic using packet switching. The experiment varies the injection rate for the wormhole packets, while keeping the packet-switching injection constant at one packet every 1500 cycles. Figure 3(a) shows the average end-to-end packet latency for each class of traffic, while Figure 3(b) shows the standard deviation of latency



(a) Mean latency (all packets)



(b) Standard deviation (5-hop packets)

Figure 3: Mixing packet switching and wormhole switching

for all packets traveling exactly five hops.

As the amount of wormhole traffic increases, the best-effort packets incur increased latency and delay variance because of channel contention within the best-effort virtual network. This contention does not influence the guaranteed packets, since blocked wormhole packets do not consume any physical link or buffer resources. Both the average latency and predictability of the guaranteed packets are largely unaffected by the presence of traffic on the other virtual network, due to fine-grain arbitration amongst the virtual channels.

4 Conclusion

Emerging distributed applications impose a broad range of performance requirements on the communication subsystem, including control over end-to-end latency, delay variance, and bandwidth allocation [9]. For point-to-point networks, these communication requirements affect the suitability of particular routing and switching schemes. While the network hardware alone cannot satisfy application performance requirements, design decisions should not preclude the communication subsystem from establishing guarantees.

The combination of packet switching and static routing simplifies packet scheduling and bandwidth allocation, but impinges on end-to-end packet latency. With flexible control close to the communication links, network hardware can apply routing and switching schemes tailored to the unique performance requirements of guaranteed and best-effort traffic. By employing aggressive routing and switching schemes, best-effort traffic incurs lower latency while reducing intrusion on the software protocols at intermediate nodes.

Adapter hardware can manage these disparate traffic classes by partitioning the physical network into multiple virtual networks, each with its own communication policies. Low-level control over routing and switching, coupled with fine-grain arbitration, enables network hardware to effectively mix guaranteed and best-effort communication. The best-effort traffic can then capitalize on flexible routing and switching schemes that improve average performance, without interfering with the predictable, timely delivery of guaranteed packets.

References

[1] *Am79168/Am79169 TAXI-275 Technical Manual*, Advanced Micro Devices, ban-0.1m-1/93/0 17490a edition.

[2] K. Arvind, K. Ramamritham, and J. A. Stankovic, "A local area network architecture for communication in distributed real-time systems," *Journal of Real-Time Systems*, vol. 3, no. 2, pp. 115–147, May 1991.

[3] W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187–196, 1986.

[4] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

[5] W. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.

[6] W. J. Dally, J. A. S. Fiske, J. S. Keen, R. A. Lethin, M. D. Noakes, P. R. Nuth, R. E. Davison, and G. A. Fyler, "The Message-Driven Processor: A multicomputer processing node with efficient mechanisms," *IEEE Micro*, pp. 23–39, April 1992.

[7] J. Dolter, *A Programmable Routing Controller Supporting Multi-mode Routing and Switching in Distributed Real-Time Systems*, PhD thesis, University of Michigan, September 1993.

[8] J. Dolter, S. Daniel, A. Mehra, J. Rexford, W. Feng, and K. G. Shin, "SPIDER: Flexible and efficient communication support for point-to-point distributed systems," Technical Report CSE-TR-180-93, University of Michigan, October 1993. To appear in *Proc. Int. Conf. on Distributed Computing Systems*, June 1994.

[9] D. Ferrari, "Client requirements for real-time communication services," *IEEE Communication Magzine*, pp. 65–72, November 1990.

[10] D. Ferrari and A. Gupta, "Resource partitioning for real-time communication," in *International Symposium on Global Data Networking*, December 1993.

[11] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," in *Proc. Int. Conf. on Distributed Computer Systems*, pp. 300–307, May 1991.

[12] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, September 1979.

[13] S. Konstantinidou and L. Snyder, "Chaos router: Architecture and performance," in *Proc. Int'l Symposium on Computer Architecture*, pp. 212–221, May 1991.

[14] A. Kovaleski, S. Ratheal, and F. Lombardi, "An architecture and interconnection scheme for time-sliced buses in real-time processing," *Proc. Real-Time Systems Symposium*, pp. 20–27, 1986.

[15] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, January 1973.

[16] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, pp. 62–76, February 1993.

[17] K. G. Shin, "HARTS: A distributed real-time architecture," *IEEE Computer*, vol. 24, no. 5, pp. 25–35, May 1991.

[18] K. G. Shin and Q. Zheng, "Mixed time-constrained and non-time-constrained communications in local area networks," *IEEE Trans. Communications*, pp. 1668–1676, November 1993.

[19] J. A. Stankovic, "Distributed real-time computing: The next generation," Technical Report COINS 92-01, University of Massachusetts, Amherst, January 1992.