Appeared in: Proceedings of the Joint Workshop on Parallel and Distributed Real-Time Systems,
Santa Barbara, California, April 24-26, 1995.

Real-Time Networking over HIPPI

Riccardo Bettati *   *Anisoara Nica*

The Tenet Group
University of California at Berkeley
and
International Computer Science Institute

Abstract

HIPPI provides a very-high-speed communication medium, which is very well suited for a large number of bandwidth-demanding distributed applications. Unfortunately, its circuit-switched nature makes it very difficult to provide real-time guarantees when connections contend for network resources. We present a time-division-multiplex access scheme designed to give timing guarantees to high-speed connections. We describe the problem of scheduling the access to a HIPPI network, and show that, although the problem is very unlikely to be computationally tractable, very simple heuristics give high network utilizations for moderately-sized networks. We present the RMP/RMCP protocol, our implementation of the scheme described in this paper on the XUNET-West HIPPI testbed.

1 Introduction

A large number of applications in distributed control, distributed virtual reality, and remote laboratoring demand for hard delay guarantees in order to satisfy the timing requirements of their time-critical components. Such applications typically consist of long sequences of computation-intensive operations on large amounts of data, such as video, to be performed by special-purpose, remotely located, machines. The bandwidth requirements are very high because of the large amounts of data that are involved and the limited applicability of data compression because of its high computational cost.

A remote microscopy application, for example, may involve (1) an electrone microscope, which is equipped with a camera, (2) a supercomputer for image enhancement and focus control, and (3) a workstation with the human operator. The camera in the microscope captures the picture of the sample under observation and sends it as a stream of video images to the supercomputer. The latter performs the necessary image enhancements and forward the images to the workstation with the operator. It also computes the necessary control inputs to keep the sample in focus, and returns the results to the microscope. The operator observes the enhanced video sequence forwarded by the supercomputer, and guides the experiment by issuing control requests to the supercomputer (tuning of the image enhancement) and to the microscope (positioning of the sample). The underlying communication media must satisfy two criteria. First, the transmission of the video images requires very-high bandwidth connections. Second, the time-critical components of the system, such as the focus control loop, require guarantees by the network on transmission delay bounds.

The bandwidth requirements of such applications are expected to be in the hundreds of megabits per second, and therefore exceed the bandwidth offered by existing cell-switched technologies (e.g. 155Mb/s ATM). We have investigated the applicability of circuit-switched network technologies, in particular High-Performance Parallel Interface (HIPPI) [1, 2] for this class of applications. HIPPI is a simplex point-to-point interface for transferring data at very high rates (800Mb/s or 1600Mb/s) over limited distances (up to 25 meters). HIPPI physical layer switches can be used to interconnect multiple devices into a local area network [3]. Serial fiber extenders make it possible to connect devices over distances of up to 10 kilometers [4].

Once a HIPPI connection has been established, the full peak rate of 800 Mb/s is available end-to-end. In order to bound connection setup delays due to contention for network resources, however, proper con-

*Address: International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94704, USA. E-Mail: bettati@icsi.berkeley.edu. Tel: (510)643-9153. Fax: (510)643-7684

nection management strategies must be devised. The HIPPI standard does not address the issue of how to manage connection setup. Chlamtac et al. [5] analyze and compare connection management schemes for clusters of crossbar switches with respect to connection delay and throughput. The described approaches fail to guarantee either bounds for connection delay or bandwidth allocation.

In this paper we describe a connection establishment protocol for HIPPI networks that addresses the issues of time-constrained setup of guaranteed-bandwidth connections.

2 Scheduling of Real-Time Connections in Circuit-Switched Networks

A HIPPI network consists of a number of crossbar switches and hosts, connected by unidirectional links. Links connecting two switches are called *trunks*, whereas links connecting hosts to a switch are called *ports*. In order to setup a (simplex) *HIPPI connection*, the client at the source first acquires the port connecting the source to the network (the *source port*). Then it sequentially acquires the trunks along a path from the source port to the port connecting the network to the destination (the *destination port*). In the end the destination port must be acquired, and the connection is set up. A duplex connection consists of a pair of two independent connections in opposite directions. When all resources (i.e. ports and trunks) are available, the connection setup takes one round-trip delay between the source port and the destination port. If, however, a resource is used by another connection, the user can choose to either abort the setup process or to block until the resource becomes available, using the so-called *camp-on* feature of HIPPI.

This scheme can lead to deadlocks and to unbounded connection setup delays. The HIPPI standard does not provide a way to avoid these problems. Nevertheless, some recommendations are made about how to reduce connection setup delays [6]. In particular, these recommendations discourage from keeping connections too long (in order to reduce blocking times) and from camping on during connection setup. Although avoiding camping on eliminates the possibility of deadlocks, an additional protocol is necessary to avoid unbounded connection setup delays.

Channel acceptance protocols for circuit-switched networks must rely on a global scheduling scheme of the network resources (as opposed to local schemes used in packet-switched networks [7]). Whenever a connection is being set up, all necessary resources have to be available at that particular point in time for the setup to get through.

Typically, global resource access schemes rely on collision resolution, tokens, or time division multiplex. Traditional collision resolution schemes, such as tree-based resolution [8], are not applicable for HIPPI-style high-speed networks because of the high amount of bandwidth lost due to the resolution process.

Token-based approaches are not practicable as well because of two reasons: First, it can be shown that having tokens rotating in-band (i.e. competing for the same network resources as the data-transmission) would cause tokens to queue up and thus the scheme to degenerate into a single-token scenario and, hence, to a serialization of the access to the network. This would mean that only one connection could be set up at any particular point in time [1] . Second, token-based approaches do not, *per se*, guarantee timing constraints, even for very simple systems. Additional schemes, such as *timed tokens* in FDDI, must be used to give timing guarantees. Such schemes substantially reduce the maximum utilization of the network resources by real-time traffic, e.g. by two thirds for the case of timed tokens in a ring [9].

Time-division multiplexing (TDM), on the other hand, avoids resource contention by explicitely laying out the transmission schedule. This obviates the need for expensive collision resolution schemes and results in a high utilization of the network because it allows for high concurrency of access to the network resources. We will describe in the following how we apply this scheme for the particular case of real-time communication in a HIPPI network.

3 Time Division Multiplex: Allowing Multiple Real-Time Connections on HIPPI

We define the set of requested workload $\mathcal{C}$ to consist of the *real-time channels* $\{C_1, C_2, \ldots, C_N\}$. Each real-time channel C_i (in the following simply called *channel*) is a simplex connection with guaranteed quality of service and is characterized by the following tuple:

$$C_i = (S_i, D_i, p_i, \tau_i),$$

where the components are defined as follows:

[1] The serialization effect could be avoided by having the tokens rotate out-of-band, either by using a separate network for the token traffic or by allocating a portion of the HIPPI bandwidth for that purpose. The second solution would mean that one would have to multiplex the network between data traffic and token-passing. This would cause many of the same problems as in the TDM approach described below.

S_i: Source host of channel C_i.

D_i: Destination host of channel C_i.

p_i: Worst-case interarrival time of messages on channel C_i.

τ_i: Maximum length of messages on channel C_i.

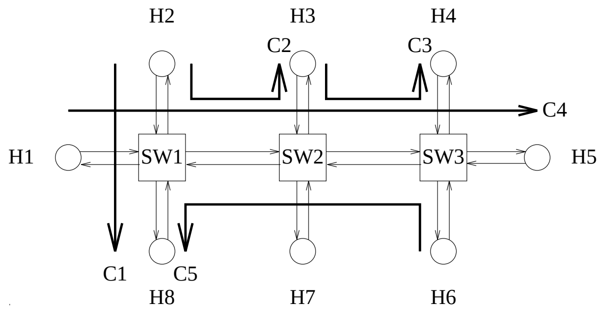
We follow a guaranteed-bandwidth allocation model for quality of service (QOS) and do not take into consideration traffic burstiness because of two reasons: First, letting the application handle traffic bursts (for example with appropriate rate control) simplifies the mechanisms to give guarantees at network level considerably, in particular for circuit-switched architectures. Second, the class of applications considered here generates mainly continuous data streams with little or no compression and thus with small amounts of burstiness. Even if the underlying network would allow for stochastic multiplexing, the gains would be very small.

Underlying our TDM approach is a framing scheme, which partitions the data transmission into a sequence of identical-length *frames* $(F_1, F_2, \ldots, F_j, \ldots)$. In order to schedule the message transmission for a set $\mathcal{C}$ of channels from frame F_j on, a schedule is first generated for the length of F_j, and repeated for each following frame.

Each frame is partitioned into a sequence of *slots* $S_1, S_2, \ldots, S_D$, all of length s, where $D \times s$ is the frame length. (In the following discussion we will make the simplifying assumption that the channel parameters are suitably discretized. In particular, τ_i and p_i are integer multiples of s, and D is an integer multiple of p_i.) The smallest schedulable unit of time is the slot, meaning that HIPPI connections are set up at the beginning of each slot, and last either until the QOS guarantee is satisfied, or until the end of the slot, whichever comes first.

A *valid* schedule is a many-to-many assignment of channels to slots in a frame so that no two channels that share at least a common resource are scheduled during the same slot. A schedule is *feasible* if, in addition to being valid, it satisfies the timing constraints for all channels . More formally, a schedule is feasible if, for every channel C_i, it assigns at least τ_i/s slots during each of its D/p_i periods in the frame.

Figure 1 illustrates the TDM scheme in a simple network with three switches and 8 hosts. The frame length is four time units, which in this example is also the period of channels C_2, C_3, and C_5. The channels C_1 and C_4 have a period of length two. As Figure 1 shows, a number of channels contend for the network resources. C_1 and C_2 contend for their source port

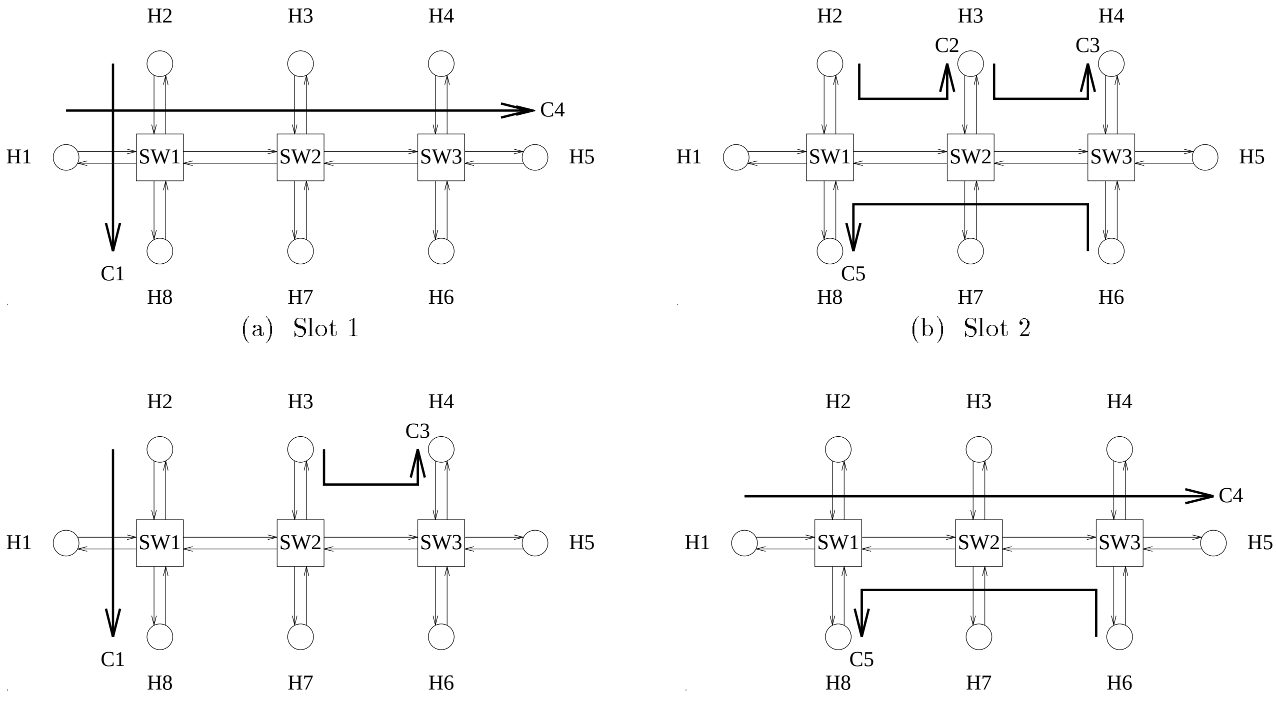


chan.	S_i	D_i	p_i	τ_i
C_1	H_2	H_8	2	1
C_2	H_2	H_3	4	1
C_3	H_3	H_4	4	2
C_4	H_1	H_5	2	1
C_5	H_6	H_7	4	2

Figure 1: Example of Set of Channels to Be Established.

from host H_1 to switch SW_1, C_2 and C_4 for the trunk connecting SW_1 and SW_2, C_3 and C_4 for the trunk connecting switch SW_2 and SW_3. At last, C_1 and C_5 contend for their destination port from SW_1 to host H_8. Figure 2 shows an example of a feasible schedule.

The problem of finding feasible TDM schedules for real-time channels is a generalization of the traditional time slot assignment problem in switched-satellite TDM access (SS/TDMA) systems [10, 11]. In particular, the problem of feasibly scheduling a set of channels with identical periods on a network of arbitrary topology is identical to the time slot assignment problem to minimize completion time in unbuffered SS/TDMA with intersatellite links, which has been proven to be $\mathcal{NP}$-complete even for very simple interconnection topologies [12]. It is therefore very unlikely that the problem of generating feasible TDM schedules for varying periods is tractable for networks with more than one switch. The problem is polynomially solvable (using network flow, for example) for a single switch. A suite of simulation experiments have shown that a simple earliest-deadline-first (EDF) approach performs very well in general. In these experiments, we measured the success rate of EDF, that is, the probability that EDF generates a feasible schedule for schedulable sets of channels. The schedulability of the sets of channels was determined earlier by exhaustive search. In these experiments the source and destination hosts S_i and D_i are randomly chosen, and the *utilization factor* $u_i = \tau_i/p_i$ is uniformly distributed between zero and the *maximum utilization factor* u_{max}, where $0 \leq u_{max} \leq 1$.



(a) Slot 1 (b) Slot 2 (c) Slot 3 (d) Slot 4

Figure 2: Example of a Feasible Schedule.

Figures 6 – 7 shows the result of these measurements for different values for u_{max}. The experiments also compare the results for EDF with the success rates achieved by the Rate-Monotonic algorithm (RM), an example of a simpler, static-priority algorithm. The comparison shows that in all experiments EDF performs substantially better than RM. While the success rate of RM drops to around 20% for larger networks and higher load, the success rate for EDF was always measured above 80%. This encourages us to use a simple EDF-based algorithm to schedule transmission on real-time channels in RMP/RMCP, our real-time channel acceptance protocol for HIPPI.

4 RMP/RMCP: Integrated Connection Establishment and Transmission Scheduling

We designed and implemented RMP/RMCP, a real-time channel acceptance protocol for HIPPI networks. Upon a request for channel establishment from a sending host, the protocol performs the necessary schedulability tests (in form of slot assignments) and initiates the admission control tests at the receiving host. The sender is then notified if, given the current load in the network and the receiver, the channel establishment request can be granted. We have adopted a centralized admission control scheme, in which a single *acceptance control server* (ACS) performs the schedulability tests. The RMP/RMCP protocol is represented on each host by the local *real-time messaging control agent* (RMCA). The RMCA provides (1) an interface of the admission control portion of the RMP/RMCP to the application running on the host, (2) the synchronization between hosts at frame level, and (3) message buffering and forwarding for established channels. Figure 3 illustrates the architecture of RMP/RMCP. When requesting a channel establishment, the sender contacts the control portion of the local RMCA, which forwards the request to the ACS. The latter forwards the request to the receiver through the RMCA on the receiver's host. The receiver performs the local admission control based on its own resource availability and returns the result to the ACS. The latter tentatively constructs a global transmission schedule that includes the connection. If the connection can be feasibly scheduled, the establishment request succeeds. The result of the establishment request is then returned to the requesting RMCA on the sender host. If the establishment request is rejected, the local RMCA simply informs the sender. Otherwise, the local transmission schedule is updated, and the sender is informed that it can start

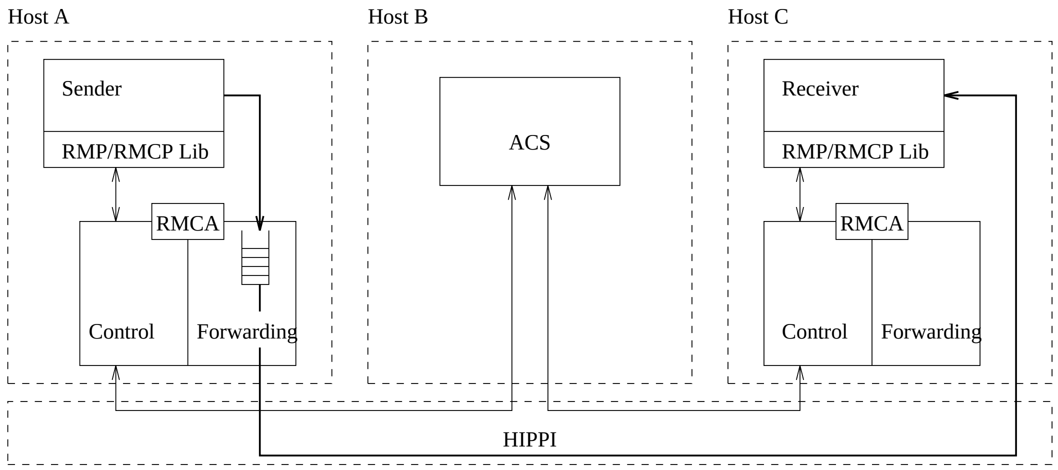


Figure 3: The Architecture of RMP/RMCP.

depositing messages into the appropriate forwarding queue of the RMCA. At the receiving host, the data stream bypasses the RMCA, and the receiver connects directly to the HIPPI drivers.

The tight synchronization among the hosts necessary in this scheme is achieved by a combination of explicit synchronization of the RMCAs at frame level and by taking advantage of the camp-on feature of HIPPI to achieve a partial synchronization at slot level. Figure 4 describes how the admission control messages are embedded in the remaining data transmission to achieve synchronization at frame level. In each frame, two slots are reserved for communication between ACS and the RMCAs. The beginning of the frame is marked by a reserved slot, during which ACS sends out control messages to all the RMCAs. These messages can be (1) forwarded request messages to the receivers, (2) reply messages to inform the senders about requests, or (3) empty messages. In any case, the RMCAs are synchronized on the receipt of this message. (We note that a very tight synchronization can be achieved with this simple scheme because of the quick setup time for HIPPI connections, which is in the order of a few μs.) During the second reserved slot, ACS receives control messages from the RMCAs. Such messages are either requests from senders or replies to requests from receivers. The remaining slots are allocated for data transmission.

Synchronization at slot level is less critical than expected. As long as the synchronization between hosts is within one half of the slot length, the transmission sequence does not get out of order. When the clock of a host falls behind and does not tear down the connection by the end of the allocated slot, the setup of a conflicting connection of another host blocks until the first host tears down its connection. This gives the possibility for hosts with conflicting connections to resynchronize inbetween frame boundaries.

5 Experiments

We are implementing our work on the Xunet3 West testbed, a HIPPI network connecting Lawrence Berkeley Laboratory (LBL) to the UC Berkeley (UCB) campus. (Xunet3 West is part of the Blanca Gigabit testbed research initiative.)

The topology of the testbed is depicted in Figure 5. The network consists of two HIPPI switches, one located at LBL, the other on the UCB campus. The switch at LBL is connected to a SGI Challenge multiprocessor workstation, a Sun Sparc 10 (all Sparcs are running under Solaris), and a PsiTech frame grabber. The campus switch is connected to a Sparc 10 and a Raid II massive storage device [13]. The addition of a third switch is planned, with a Connection Machine CM-5, a Sun Sparc 10 and an interface

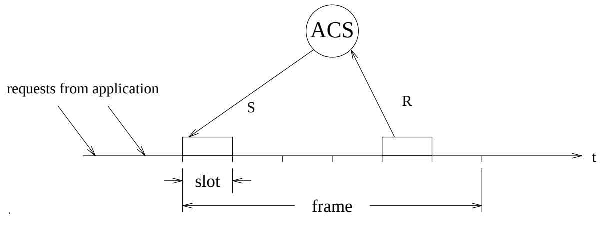


Figure 4: Admission Control Server.

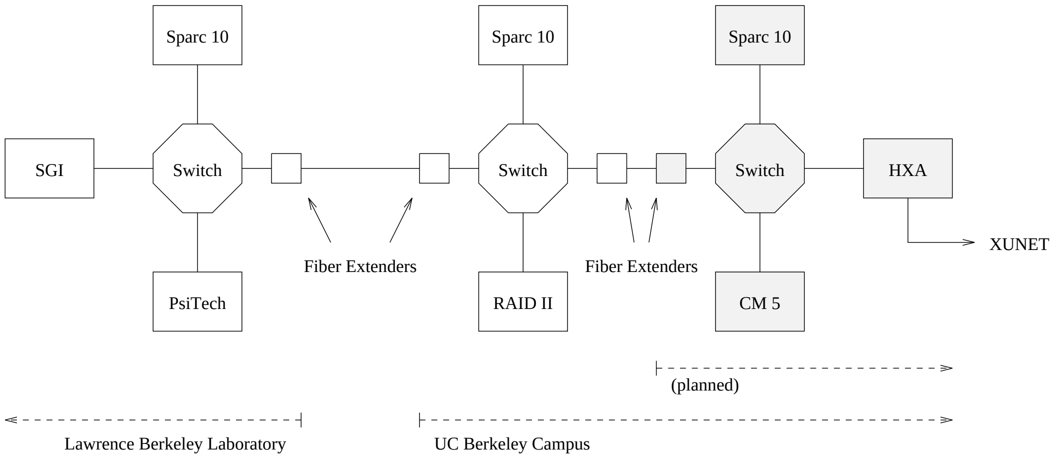


Figure 5: The Xunet3 West Testbed

to XUNET, the transcontinental ATM-style Gigabit testbed of Blanca.

Due to the large variety of platforms, we have opted for a user-level implementation of the RMP/RMCP protocol (although we were well aware of the performance penalties and the reduced control over the timing behavior). Both the ACS and the RMCAs are currently realized as user-level daemons. Our first experiment involved the two Sparcs, with the ACS daemon running at LBL and connections established in both directions. Currently, we can establish connections between all components of the testbed, except for the PsiTech, which can act as receiver only. One important parameter we are currently investigating is the minimum slot length that still allows the hosts to maintain the necessary synchronization. With Solaris bounding the interrupt service time to below 1 ms, a minimum slot length of slightly over 2 ms should be appropriate. However, we still have to verify this in the experimental setting.

6 Summary

Current cell-switched technologies are not adequate in satisfying the high bandwidth requirements in a number of non-traditional distributed multimedia applications. We investigate the applicability of circuit-switched technologies, in particular HIPPI, for this class of applications. In this paper we describe RMP/RMCP, a reservation-based communication protocol that was developed with an eye on distributed laboratoring. We motivate TDM as the underlying scheduling principle and describe a two-level synchronization scheme as the underlying mechanism. We elaborate on the complexity of scheduling real-time connections in a HIPPI-style network and evaluate the applicability of simple EDF-based scheduling.

HIPPI has proven to be a very hostile setting for the development of real-time protocols. First, the lack of prioritization of connection setup and of preemptability of connections makes any real-time protocol vulnerable to the ability to control timing in every hosts's operating systems. Second, the hardware-level flow control of HIPPI allows real-time protocols only guarantees in terms of time to use the connections. In order to guarantee data rates, the bandwidth guarantees must be provided by the admission control at both the sender and the receiver.

The suitability of TDM for constant-rate traffic is well known. We have proposed a TDM scheme for an underlying circuit-switched network technology and traffic that is apt to give very little gain for stochastic multiplexing. As the bandwidth of cell-based technology increases, so does the need for larger buffers in the switches to accommodate long bursts of traffic. In order to reduce these buffer requirements, we will investigate TDM as an underlying principle for burst reservation schemes, such as in Highball [14].

Acknowledgements

The authors would like to thank Andres Albanese and Maurizio Bonuccelli for their helpful input on scheduling algorithms for circuit-switched networks.

Special thanks goes to Prof. Domenico Ferrari for having provided the framework for this research. Jin Guojun from LBL has been very helpful in keeping the testbed in smooth operation.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by the Department of Energy under Grant DE-FG03-92ER25135, by AT&T Bell Laboratories, Digital Equipment Corp., Hitachi, Ltd., Pacific Bell, Sun Microsystems, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

References

[1] ANSI X3T9, "High-performance parallel interface – mechanical, electrical, and signalling protocol specification (HIPPI-PH)", *ANSI X3.183-1991.*

[2] D. Tolmie and J. Renwick, "HIPPI, simplicity yields success", *IEEE Network*, pp. 28–32, Jan. 1993.

[3] ANSI X3T9, "High-performance parallel interface – phyical switch control (HIPPI-SC)", *ANSI X3.222-199x.*

[4] Serial-HIPPI Implementors Group, "Serial-HIPPI specification (serial-HIPPI)".

[5] I. Chlamtac, A. Ganz, and M. G. Kienzle, "An HIPPI interconnection system", *IEEE Trans. Computers*, vol. 42, pp. 138–150, Feb. 1993.

[6] J. Renwick and A. Nicholson, "IP and ARP on HIPPI", *RFC–1374*, Oct. 1992.

[7] H. Zhang and D. Ferrari, "Rate-controlled static priority queueing", *in Proceedings of INFOCOM'93*, San Francisco, 1993.

[8] J. Capetanakis, "Tree algorithms for packet broadcast channels", *IEEE Trans. on Information Theory*, vol. 25, pp. 505–515, Sep. 1979.

[9] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with the timed token protokol", *in Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 468–475, Yokohama, Japan, June 1992.

[10] T. Inukay, "An efficient SS/TDMA time slot assignment algorithm", *IEEE Trans. Commun.*, vol. 27, pp. 1449–1455, Oct. 1979.

[11] J. L Lewandowski, J. W. S. Liu, and Liu C. L., "SS/TDMA time slot assignment with restricted switching modes", *IEEE Trans. Commun.*, vol. 31, pp. 149–154, Jan. 1983.

[12] A. A. Bertossi, G. Bongiovanni, and M. A. Bonucelli, "Time slot assignment in SS/TDMA systems with intersatellite links", *IEEE Trans. Commun.*, vol. 35, pp. 602–608, June 1987.

[13] S. Seshan and R. Katz, "Interfacing a high performance disk array file server to a gigabit lan", *in Proceedings of the 18th Conference on Local Computer Networks*, Minneapolis, Sep. 1993.

[14] D. L Mills, C. G. Boncelet, J. G. Elias, P. A. Schragger, and A. W. Jackson, "Highball, a high speed, reserved-access wide area network", Technical Report 90-9-1, Electrical Engineering Department, University of Delaware, 1990.

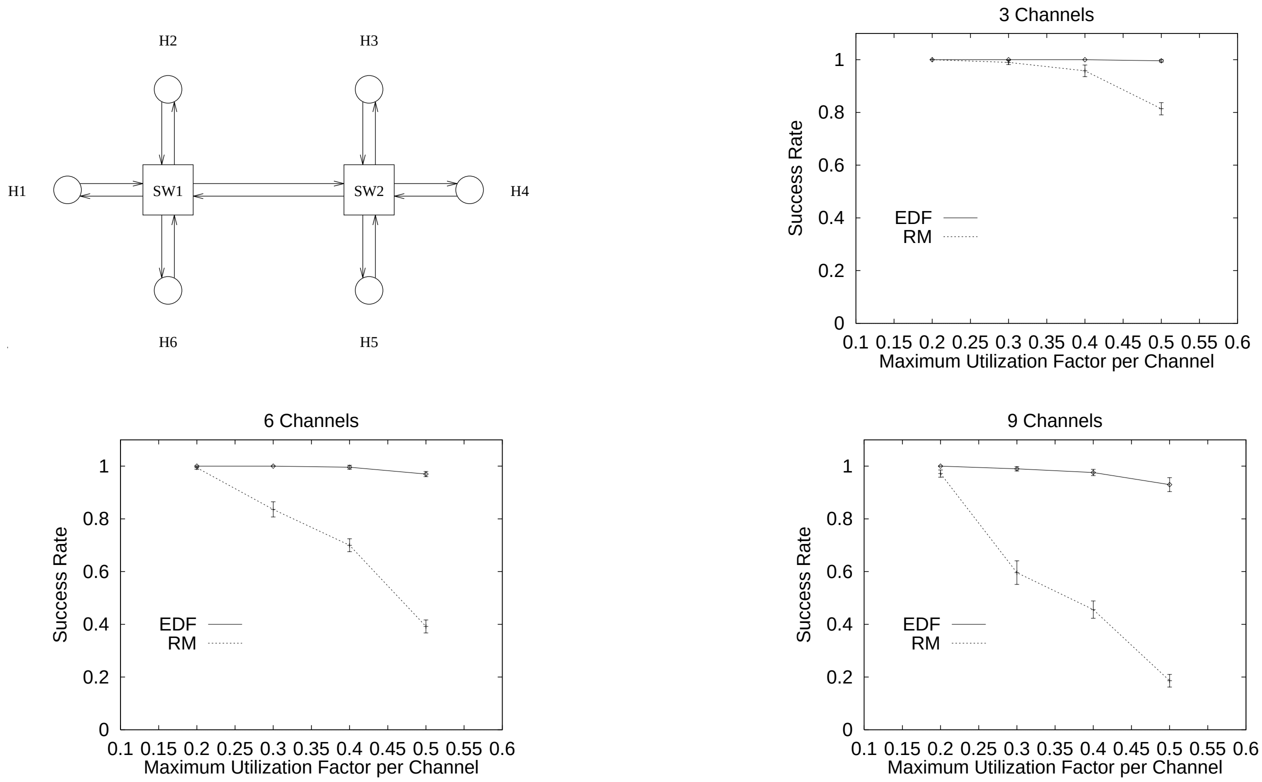


Figure 6: Experiments on Network with Two Switches.

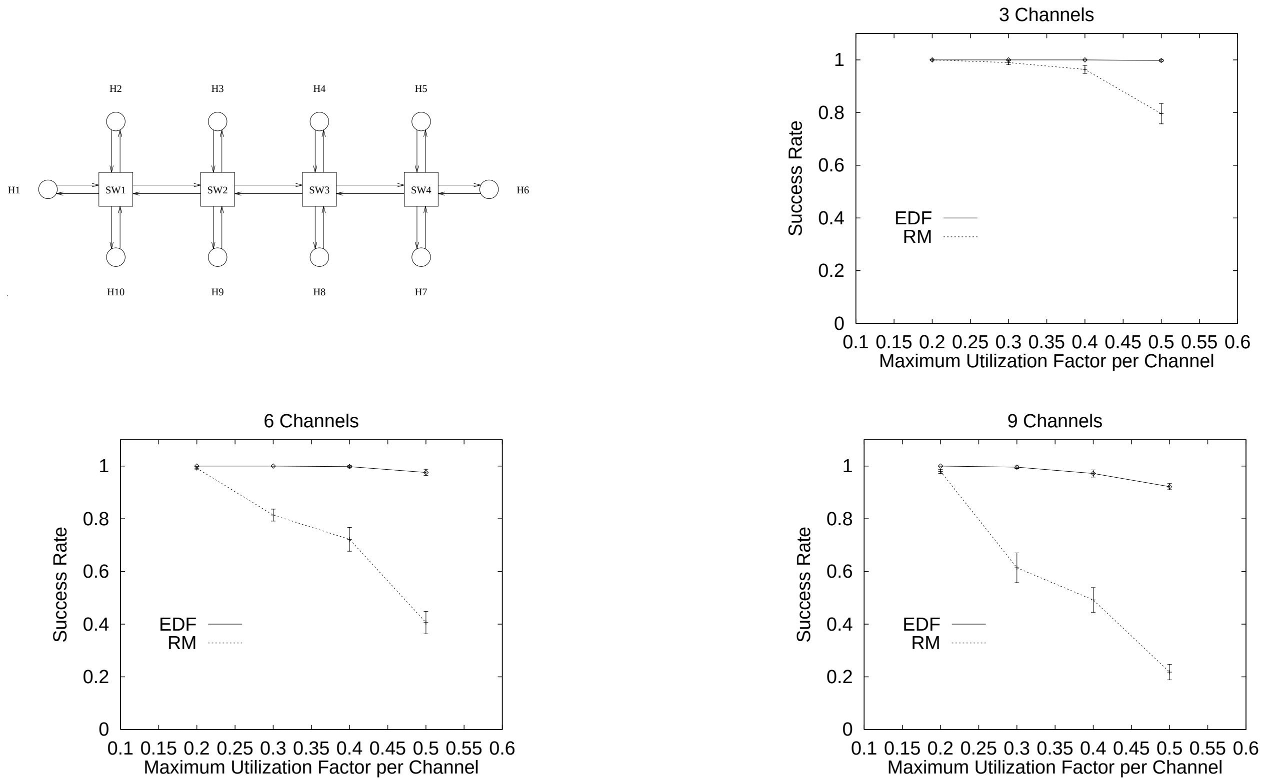


Figure 7: Experiments on Network with Four Switches.