A Scheme for Real-Time Channel Establishment in Wide-Area Networks

DOMENICO FERRARI, FELLOW, IEEE, AND DINESH C. VERMA

***Abstract*—Multimedia communication involving digital audio and/or digital video has rather strict delay requirements. A real-time channel is defined in this paper as a simplex connection between a source and a destination characterized by parameters representing the performance requirements of the client. A real-time service is capable of creating real-time channels on demand and guaranteeing their performance. These guarantees often take the form of lower bounds on the bandwidth allocated to a channel and upper bounds on the delays to be experienced by a packet on the channel.**

In this paper, we study the feasibility of providing real-time services on a packet-switched store-and-forward wide-area network with general topology. We describe a scheme for the establishment of channels with deterministic or statistical delay bounds, and present the results of the simulation experiments we ran to evaluate it. The results are encouraging: our approach satisfies the guarantees even in worst case situations, uses the network's resources to a fair extent, and efficiently handles channels with a variety of offered load and burstiness characteristics. Also, the packet transmission overhead is quite low, and the channel establishment overhead is small enough to be acceptable in most practical cases.

I. Introduction

REAL-TIME computer–computer communication, by which we mean computer communication with guaranteed performance, is expected to become a necessary feature of future networks, as the digital audio and digital video components of their multimedia traffic will require much tighter performance control than is exerted in current networks. Once networks start offering real-time services, we believe that the domain of the applications of these services will rapidly grow. Performance guarantees, along with other guarantees, are going to be an important option among those that will define the various grades of service most observers believe will be offered by the networks of the future.

Such guarantees will have to be provided not only by local-area networks, where solutions like the FDDI ring and its protocols are already being developed [9], but also by wide-area networks: the need for multimedia conferencing services is only one of the many examples that can be invoked to demonstrate the validity of this statement. The available options for providing real-time services in a wide-area context are circuit switching, fast packet switching, and combinations of these two techniques. The former is not particularly convenient when the traffic includes (as we expect it will in most cases) a large data/text component [7]. Fast packet switching (for example, ATM) is better suited to the type of traffic we are envisioning; however, to our knowledge and in its formulations to date, it does not guarantee the low delays it can provide. Hybrid schemes are attractive since they use the more appropriate technique for each of the two basic types of traffic, but they require more complex switches and policies, and do not integrate as completely these basic types as the other two schemes [3].

This paper describes a method for guaranteeing delays in a packet-switching wide-area network, and presents an evaluation of some of its most important characteristics. We believe that the method can be used in all connection-oriented packet-switching environments, including ATM-style fast packet switching. However, we present it in the context of a contemporary connection-oriented packet-switching wide-area network, and we evaluate it by simulation in the same context. We state the problem precisely in Section II, sketch our approach to it in Section III, and illustrate the heart of our solution, i.e., the establishment algorithm, in Section IV. Section V describes the simulations we ran to evaluate the scheme, and discusses the results we obtained. Finally, conclusions are drawn in Section VI.

II. Problem Statement

In its full generality, the network we consider in this study has an arbitrary topology, and may consist of several networks of various types (LAN's and WAN's). In it, a message can go from a source host to a destination host passing through a number of intermediate nodes, some of which may be gateways (or routers) connecting one network to another. The functions of a network node may be implemented by switches, by hosts, or by special purpose communication computers. A source-destination path goes through a number of nodes where the transmitted information can be stored and then forwarded to the

Manuscript received June 6, 1989; revised November 10, 1989. This work was supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order 4871, monitored by Naval Electronic Systems Command under Contract N00039-84-C-0089, the University of California with a MICRO Program grant, AT&T Bell Laboratories, Inc., Cray Research, Inc., Hitachi, Ltd., IBM Corporation, Ing. C. Olivetti & C., S.p.A., 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 any of the sponsoring organizations or of the U.S. Government.

The authors are with the Computer Science Division, Department of Electrical Engineering and Computer Science, University of California at Berkeley, and with the International Computer Science Institute, Berkeley, CA 94720.

IEEE Log Number 8934086.

next node. Some of the links between adjacent store-and-forward nodes may actually be nonstore-and-forward networks, provided their total delay can be bounded. They could be, for example, very high-speed circuit-switched trunks, whose transmission speed is so high that there can be no storing or processing in the switches. The traffic on these trunks will generally include packets from different sources to different destinations, but the delay of each packet will be bounded. Thus, the only assumption we make about the links between store-and-forward nodes is that there be a known and finite bound for the link delay of each packet. Without it, we would not be able to offer hard real-time guarantees, i.e., to guarantee that no packet is delayed more than a given bound. It should be noted that this assumption is not exactly satisfied by links governed by contention-based protocols (e.g., Ethernets, see [8]), but it is by other types of LAN's, such as FDDI rings (see for example [14], [5]).

In the type of network we have just described, packet delays are much harder to control, and resources much more difficult to reserve for real-time traffic, if each packet going from a given source to a given destination is in principle allowed to follow any route. For these reasons, we assume that real-time communication will be based on simplex fixed-route connections to be called *real-time channels* or simply *channels*, and that resources will be reserved for them during the connection establishment operation, as well as freed during connection disestablishment. The route of a channel will be chosen at the time of its establishment. A channel then is essentially a virtual circuit with performance guarantees.[1]

The connection-oriented approach unfortunately requires explicit establishment, which may be slow. Our solution tries to reduce the establishment time by limiting the connection procedure to a single source-destination roundtrip and speeding up as much as possible the establishment algorithms. An additional speedup can be obtained by allowing the source to start sending packets immediately following the request message, but will not be described in this paper.

Besides offering real-time service, the network will usually provide other types of service (e.g., datagram, reliable datagram, nonreal-time connection-oriented service, and so on). All of these different paradigms will have to coexist within the same network if heterogeneous traffic is to be handled effectively and efficiently.

In order to provide real-time service, we require that the clients declare their traffic characteristics and performance requirements at the time of channel establishment. There are various meaningful ways in which the offered load and the performance bounds of a channel can be specified. For our first study in this area, we have chosen the following parameters.

- For the offered load,
  - the minimum packet interarrival time on the channel $x_{\min}$,
  - the minimum value x_{ave} of the average packet interarrival time over an interval of duration I,
  - the maximum packet size $s_{\max}$, and
  - the maximum service time t in the node for the channel's packets;
- for the performance bounds,
  - the source-to-destination delay bound (or bounds) for the channel's packets, and
  - the maximum packet loss rate.

Note that x_{ave} is the average interarrival time during the channel's busiest interval of duration I. Note also that specifying $x_{\min}$ and x_{ave} together with $s_{\max}$ corresponds to requesting that the network provide a certain peak bandwidth and a certain long-term average bandwidth, respectively. If the channel request is accepted, i.e., if the desired bandwidths are allocated to the channel, the client is expected to satisfy the interarrival time bounds specified by the offered load parameters, whereas the delay bounds are to be guaranteed by the provider, i.e., by the network.

Various types of channels can be defined, corresponding to the different types of delay bounds. The algorithm presented in this paper considers the following two types:[2]

- *deterministic:* the bound D is an absolute one; this is necessary in hard real-time applications;
- *statistical:* the bound is expressed in statistical terms; for instance, the probability that the delay of a packet is smaller than the given bound D must be greater than a given value Z.

Channel i will be characterized by its own values of the offered load and performance bounds parameters; we will add the subscript i to the symbols of those parameters (e.g., we will write $x_{\min,i}$, D_i, and so on) whenever necessary to avoid ambiguities. Delay bounds are valid only for packets that reach the destination, and are not lost in the network due to buffer overrun or network errors. However, the maximum number of packets lost must be within the loss rate bounds specified at channel establishment time. We also assume that the channels to be established are statistically independent of all the channels sharing totally or partially their route. If there are dependencies between a new channel and some of the existing ones, the client requesting the creation of the new channel should inform the provider. In this paper, we will not deal with the case in which there are dependencies among channels.

III. A Solution

Our approach consists of a scheme (to be described in Sections III-C and IV) for real-time channel establish-

[1]Real-time channels can be regarded as "subcases" of the *parameterized message channels* on which the communication system of DASH [1], [2] is based: the only parameters characterizing a real-time channel are the performance-oriented ones. Real-time channels have similarities with the two types of *flows* recently proposed in the literature [4], [15], and can be easily incorporated into a recently proposed framework for a high-speed internetworking environment [12].

[2]We are not dealing with *best-effort* channels [2] explicitly here, since these channels do not require any reservation of resources, and since requests for their establishment are always accepted; however, our approach can very easily accommodate them.

ment. The scheme works only in conjunction with specific scheduling and flow control policies. These policies are briefly described in Sections III-A and B, respectively.

A. Scheduling

Scheduling in the hosts and in the nodes will be deadline-based. More precisely, we adopt a modification of the EDD (earliest due date) [10] policy in which, in the case of a conflict, priority is given to deterministic over statistical channels. All of the other tasks of a host or node, including sending, forwarding, and receiving datagrams, have a lower priority; local tasks are preemptable by real-time packets.

In each node, the scheduler maintains at least three queues: one for deterministic packets, one for statistical packets, and the third for all other types of packets and all local tasks. The third queue can in turn be replaced by multiple queues, managed by a variety of policies, but we are not concerned with these aspects of node scheduling here.

A packet arriving at the deterministic queue is to be "aligned," that is, its deadline must be appropriately reduced if its service time would overlap with that of another deterministic packet in case they both happened to be shipped at their latest possible times. The ordering of the first two queues is by increasing deadlines.

When the node must choose the next action to be undertaken, the scheduler compares the end time (i.e., the deadline) of the packet at the head of the statistical queue with the beginning time (i.e., the deadline minus the service time) of the head packet in the deterministic queue: if the latter is lower than the former, the deterministic packet is scheduled to be shipped immediately; otherwise, the comparison is repeated between the deadline of the head task in the third queue and the beginning time of the head packet in the statistical queue. If there are more than three queues, the policy, which is a general multiclass version of EDD, can be easily extended.

B. Flow Control

We argue that flow control in a high-speed network offering real-time service should be rate-based rather than window-based. Rate-based flow control (in which the sender controls its packet sending rate on the basis of its knowledge of the characteristics of the receiver and of those of the channel's path) does not require flow control acknowledgments; indeed, acknowledgments would either have to use another real-time channel, and would therefore be quite expensive, or could be sent as datagrams, in which case they might incur a rather large and perhaps highly variable delay. This delay might be too large, especially in long-distance transmissions, to be compatible with the frequencies and regularity of packet generation in most video or audio communication.

Rate-based flow control is feasible because, at channel establishment time, the receiver can check (using the method described in Section IV) whether it will be able to accept packets at the rate declared by the sender.

However, a malicious user could circumvent any rate-based flow control mechanism and from a single-user machine send packets into the network at a much higher rate than the declared maximum value $1/x_{\min}$ or maximum average value $1/x_{\text{ave}}$. The same effect might be caused by a failure in the sending host, even when such a host is a multiuser, protected-kernel system. If we do not take appropriate countermeasures, such malicious or faulty behavior can prevent the satisfaction of the delay bounds guaranteed to other clients of the real-time service, thereby damaging the clients and destroying the credibility of the service.

Our solution to the problem consists of providing distributed rate control by increasing the deadlines of the "offending" packets, so that they will be delayed in heavily loaded nodes, where they would otherwise seriously interfere with the operation of other channels. When buffer space is limited, some of them might even be dropped because of buffer overflow. Of course, a sufficient amount of buffer space will have to be statically allocated to each channel to prevent the offending packets from flooding the buffer space of a heavily loaded node and causing packets from other channels to be dropped. A very simple algorithm for distributed rate control is presented in detail in [6].

One important consequence of this scheme for the algorithms in Section IV is that we can assume that each channel satisfies its $x_{\min}$ and x_{ave} constraints at every node it traverses.

C. The Channel Establishment Procedure

The channel establishment mechanism we propose for the type of network described in Section II may be built on top of any procedure that can be used to set up connections. Besides trying to establish a connection, the mechanism will perform several tests and tentatively reserve resources in each node visited by the establishment request message. In order to make channel establishment fast, we impose on our procedure the restriction that it require only one round trip. Thus, the destination host is the last point along the path where the acceptance/rejection decision for a channel request can be made. When a node is revisited by an establishment message during this message's return trip, the resources previously reserved there must be committed or released; hence a final, irreversible decision must have already been made.

The tests to be done in each node are concerned with the availability of sufficient bandwidth in the links, as well as processing power and buffer space in the node; that is, with determining whether the new channel can go through the node without jeopardizing the performance guarantees given to the already established channels passing through the same node.

If any test fails at a node, the channel cannot be established along that route; the message will be sent back, either to the sender (which may then decide to wait or try another output link) or to an intermediate node that can try sending the message towards the destination along another path. When an unsuccessful establishment request message revisits a node on its way back to the source, it frees all the resources that were tentatively reserved there during its forward trip.

If all the tests for channel i succeed at all nodes and at the destination host, this host subdivides the delay bound D_i (and the probability Z_i of not exceeding the bound if the channel being established is statistical) among the nodes traversed by the channel, after subtracting the link delays along the route. Let $d_{i,n}$ be the delay bound assigned to the channel in a node n, and $z_{i,n}$ the probability that $d_{i,n}$ will not be exceeded in that node. A packet traveling on that channel and arriving at that node at time t will usually[3] be assigned a node deadline equal to $t + d_{i,n}$. To satisfy the overall delay bound D_i, it is sufficient (though not necessary) to satisfy the bound $d_{i,n}$ (or $d_{i,n}$, $z_{i,n}$) in each node along the route. For simplicity, we make the very conservative assumption that satisfying this sufficient condition is the goal to be met by the establishment scheme in each node.

If all tests succeed, a reply message is sent back to the source host along the channel's route; this message notifies each node about the delay bound that has been assigned to it for the new channel, and commits the necessary portions of reserved resources. When the reply message reaches the source host, this host learns that the requested channel has been set up, and can start using it.

The next section describes in some detail the algorithms for channel establishment to be executed in each node along the new channel's route and in the destination host.

IV. The Algorithms

To simplify our discussion, we shall assume here that the buffer space needed for real-time connections in the network's hosts and nodes is always available, and that the network's error rate is always lower than the acceptable loss rates. These assumptions allow us to ignore the loss rate parameter, and to postpone the treatment of buffer space allocation and management, which would make this paper appreciably longer, to a subsequent paper.

When a node receives an establishment request message, it performs two or all three of the following tests:

a) the *deterministic test*, primarily required when the channel to be established is deterministic, and involving the deterministic channels already passing through the node;

b) the *statistical test*, to be performed for the establishment of both statistical and deterministic channels if at least one statistical channel is already passing through the node or is to be set up; this test involves all deterministic and statistical channels passing through the node;

c) the *delay bound test*, which is needed in all cases; if successful, it is to be followed by the computation of the minimum feasible delay bound for the new channel in the node.

[3]There may be exceptions due to alignment (see Section III-A) or distributed rate control (see Section III-B).

If the request passes the tests, then the node sends the establishment message on to the next node, after having added to it some of the parameter values resulting from the tests listed above. The destination host performs the final tests, and then, if the request passes these too, subdivides D (and Z) among the nodes on the channel's path. We now briefly describe each one of the tests as well as two simple destination host algorithms.

A. The Deterministic Test

The deterministic test consists of verifying that enough processing power is available in the node to accommodate the additional deterministic channel without impairing the guarantees given to the others. Since we are dealing with deterministic bounds, this condition must be satisfied even in the worst possible case, i.e., even when all deterministic channels are sending packets into the node at their maximum rates. The maximum utilization of a node n by channel j, whose packets have a maximum service time in the node equal to $t_{j,n}$, is $t_{j,n}/x_{\min,j}$. The condition to be tested is

$$\sum_j (t_{j,n}/x_{\min,j}) < 1 \tag{1}$$

where the sum extends to all deterministic channels passing through node n, including the one to be established (channel i).

B. The Statistical Test

The statistical test has two goals:

i) to determine whether for each statistical channel j passing through node n the probability of a delay higher than the bound $d_{j,n}$ is below its maximum tolerable value, $1 - z_{j,n}$;

ii) to provide the destination host with the information necessary to compute $z_{i,n}$ for the new channel if this is statistical.

Both these goals can be attained by computing the *probability of deadline overflow* $P_{do,n}$. A packet may be delayed beyond its delay bound in a node because of two reasons: a temporary saturation of the node's processing or transmission capacity (*node saturation*), and impossible scheduling constraints (*scheduler saturation*).[4] While the latter is avoided, as discussed in Section IV-C, the statistical test deals with the former.

The probability that channel j is active (i.e., is carrying packets) at some instant of time during an interval I is

$$p_j = x_{\min,j}/x_{\text{ave},j}. \tag{2}$$

Given K independent channels passing through a node, the probability that the members of a subset C of them are simultaneously active is given by

$$\text{Prob}(C) = \prod_{i \in C} p_i \prod_{j \notin C} (1 - p_j). \tag{3}$$

[4]A simple example of scheduler saturation is that of two channels with respective node service times 3 and 4 units, and respective node delay bounds 5 and 6 units; if two packets from the two channels arrive simultaneously at the node, there is no way to schedule them so that both meet their deadlines.

Let at least one of the channels be statistical. To compute $P_{do,n}$, we start by listing all the *overflow combinations*. An overflow combination is a set of channels that, when simultaneously active for a sufficiently long time, may cause packets to miss their deadlines. In other words, an overflow combination is one for which inequality (1) above, with the sum extended to all active channels, is not satisfied. In making this statement, we assume a worst-case situation, i.e., one in which each active channel carries packets at its maximum rate. If the input load were to persist indefinitely, the length of the queue of packets in the node would grow without bounds.

Let H_n be the set of overflow combinations in node n, h be a member of H_n, and $P(h)$ be the probability of occurrence of combination h computed as in (3) above. Then,

$$P_{do,n} = \sum_h P(h). \tag{4}$$

We can then check whether the following inequalities are satisfied:

$$P_{do,n} \leq 1 - z_{j,n}, \tag{5}$$

for all statistical channels j existing in the node. To see whether (5) is satisfied, it is sufficient to verify that

$$P_{do,n} \leq \min\,(1 - z_{j,n}),\ \text{or}\ 1 - P_{do,n} \geq \max z_{j,n}. \tag{6}$$

If test (6) is successful for all statistical channels j in the node, the value of $P_{do,n}$ is sent to the destination host, which will try to assign to channel i a value of $z_{i,n}$ that satisfies inequality (6).

C. The Delay Bound Test

The delay bound test determines the minimum delay bound to be assigned to the channel being established so that scheduler saturation can be avoided in the node. We try to eliminate this type of saturation primarily because of the complexity of dealing with it in the general case.

To determine whether scheduler saturation is possible in a node, we divide the K (deterministic or statistical) channels passing through the node into two sets: we call U the set of those channels whose delay bound in the node is lower than the sum of the service times of the K channels, and V the set of those channels whose delay bound in the node is greater than or equal to that sum. With no loss of generality, we assume that the u channels in U are those numbered 1 through u, and that they are numbered according to the order in which their packets would be scheduled by the algorithm described in Section III-A if they arrived all at the same instant: channel 1 will be the one whose packet would be shipped first, channel u the one whose packet would be shipped last.

$$U = \left\{ i \,\middle|\, i = 1, \cdots u;\ d_{i,n} < \sum_{j=1}^{K} t_{j,n} \right\}, \tag{7}$$

$$V = \left\{ k \,\middle|\, k = u + 1, \cdots K;\ d_{k,n} \geq \sum_{j=1}^{K} t_{j,n} \right\}. \tag{8}$$

In the statistical test (see Section IV-B), we took into account node saturation, which delays packets beyond their deadlines whether or not scheduler saturation is present. Thus, in those situations in which the node is saturated, whether we have scheduler saturation or not is immaterial. However, scheduler saturation could occur also in cases in which the node is not saturated. To determine whether this will indeed happen, and to prevent it from happening, we must consider the worst possible combination, i.e., one that maximizes the likelihood of scheduler saturation without saturating the node. If the combination corresponding to all channels through the node being active saturates the node, we will have to consider for the delay bound test a combination in which some of the channels are not active. Details about how such a combination can be arrived at are given in [6].

For the combination we have selected, let us denote by T the largest of the service times of packets that may traverse the node but do not travel on channels in set U. These include not only those on channels in set V, but also those on other real-time channels (e.g., established channels not included in the combination to avoid node saturation) as well as the nonreal-time packets that may be passing through the node. Let us also assume for simplicity of proof that

$$x_{\min,i} \geq \sum_{j=1}^{K} t_{j,n}, \quad (i = 1, \cdots K). \tag{9}$$

The following theorem is the basis for the delay bound test.

Theorem: Scheduler saturation is impossible if and only if

$$d_{i,n} \geq \sum_{j=1}^{i} t_{j,n} + T, \quad (i = 1, \cdots u). \tag{10}$$

A proof of this theorem can be found in the Appendix, where we briefly discuss also the cases in which inequalities (9) do not hold. [6] contains a few examples of delay bound computations.

The delay bound test consists of verifying that inequality (10) is satisfied for all values of i (i.e., for all channels in U), after the channel to be created has been added to the K already established ones. The arrival of a request for the establishment of a new channel changes Σt_j, hence possibly the memberships of U and V, and may change even the value of T. While testing the delay bounds of established channels, if the delay bounds pass the test it is easy to determine the lower bound $d^l_{i,n}$ of $d_{i,n}$ such that scheduler saturation will be impossible in the node if we choose $d_{i,n} \geq d^l_{i,n}$. The value of $d^l_{i,n}$ will then be sent to the destination host in the establishment message.

D. The Destination Host Tests and Algorithms

The destination host, if and when it receives the establishment request message, has to determine whether the total value of Z_i for the new channel i can be factored into node contributions $z_{i,n}$ ($n = 1, 2, \cdots N$):

$$Z_i = \prod_{n=1}^{N} z_{i,n}. \tag{11}$$

This question can be answered by checking whether

$$Z_i \leq \prod_{n=1}^{N} (1 - P_{do,n}). \tag{12}$$

This may be called the *Z test*. If (12) does not hold, then the request is rejected. Note that (11) implies the assumption that, once a packet is delayed beyond its deadline in a node, it will not be able to satisfy the channel's overall delay bound. This assumption is justified by the behavior of a saturated node: even if the delay bounds are long, but finite, packets will eventually be delayed beyond them. If, however, the saturation condition is short, some or all of the deadlines might still be met. This, and the assumption, also implied by (11), that each node will delay, if any, only previously undelayed packets, are certainly pessimistic assumptions. Rejecting a request if (12) does not hold is a policy based on worst case considerations, which guarantee that the desired results will be achieved in all possible circumstances. Of course, there is a cost associated with worst case design: these and other conservative decisions reduce the maximum number of channels that can be established in a given network with respect to that which the same network could support if the design of the real-time service were less conservative.

The test concerned with the delay bound (the *D test*) consists of verifying that, for channel i with total delay bound D_i to be established, we must have

$$D_i \geq \sum_{n=1}^{N} d^{l}_{i,n}. \tag{13}$$

Two simple expressions for assigning $d_{i,n}$'s and $z_{i,n}$'s to the nodes along the channel's path are

$$d_{i,n} = \frac{1}{N}\left[D_i - \sum_{m=1}^{N} d^{l}_{i,m}\right] + d^{l}_{i,n}, \tag{14}$$

$$z_{i,n} = \left[Z_i \Big/ \prod_{m=1}^{N} (1 - P_{do,m})\right]^{1/N} (1 - P_{do,n}). \tag{15}$$

V. The Simulations

The reader certainly has many questions about the performance of the scheme described in the previous sections. In this section, we provide simulation-based answers to the following ones.

- Is the scheme correct? Can we discover cases when the guarantees of the channels are not met using this scheme?
- How much time is required to establish a connection?
- How much time is required by the scheduler to select the packet to be transmitted next?
- What is the overhead of distributed rate control?
- Providing performance guarantees involves a number of pessimistic assumptions. Is our scheme too conservative? For example, how close are our guaranteed bounds on delays to the actual delays?
- What is the "cost" of a statistical channel as compared to that of a deterministic channel?

We provide answers to these questions by means of a network simulator written in CSIM [13]. We have simulated several different networks but will only present and discuss the results obtained for the 14-node network shown in Fig. 1(a) and for a simple network consisting of only two nodes.

In Fig. 1(a), nodes 0-8 serve as sources and destinations for the real-time channels. These nine nodes were divided into three groups of three nodes each. Each source generated channel requests at random intervals. The destination group of such a channel request was chosen by the probability matrix in Table I. The actual destination was picked randomly from the group. One of the four shortest paths from a channel's source to its destination was picked at random as the route of its establishment request.

A. Workload

Future multimedia networks will have a wide variety of traffic types. To capture some of this diversity, we considered four types of channels. Table II shows the types of channels used in the simulation experiments. Type I channels tax the network's resources heavily, while type IV are very light.

The simulations were run in two phases. In Phase 1, we tried to establish as many channels as we could. The next channel to be established was of any type and either deterministic or statistical with equal probabilities. The exact number and types of channels set up in such a context depended on the order of arrival of channel requests. In some case, a request for a type I channel arrived first, and many subsequent requests for type IV channels were rejected. On the other hand, if the first few channels were of type IV, a subsequent request for a type I channel might be refused, but many more channels of type III could be established.

Let the *load* of a channel i at node n be defined as $t_{i,n}/x_{\min,i}$ (see Section IV-A for a motivation). The total load of all the channels accepted at a node reflects the maximum instantaneous utilization of that node allowed by our scheme. In that sense, the load serves as a better measure of saturation at a node than the number of accepted channels.

The results of one Phase 1 simulation of the network in Fig. 1(a) are shown in Fig. 1(b). The two numbers associated with each node are the load of all the channels and of the deterministic channels only, respectively, at that node. The number [1.55, 0.5] with node 1 implies that when all the statistical channels are sending packets at the maximum possible rate, the arrival rate of packets exceeds the service rate of the node by a factor of 1.55, while if only all the deterministic channels were sending at peak rate, the arrival rate would be 0.5 times the ser-

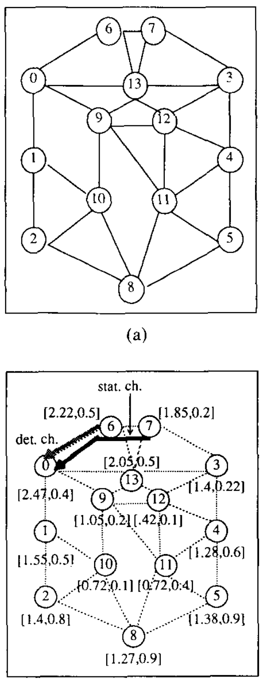


Fig. 1. (a) The simulated network. (b) Results of a typical simulation, with the maximum loads permitted by our scheme [all channels, deterministic channels only].

TABLE I
SOURCE DESTINATION GROUP PROBABILITIES

From nodes	To nodes		
	0-2	3-5	6-8
0-2	0.10	0.72	0.18
3-5	0.72	0.10	0.18
6-8	0.45	0.39	0.16

TABLE II
CHANNEL TYPES

channel type	I (time units)	$x_{\min}$ (time units)	x_{ave} (time units)
I	4	10	60
II	4	40	120
III	1	10	60
IV	1	40	120

vice rate. The maximum possible utilization of node 1 by the deterministic channels alone is 50%.

We selected this configuration containing about 38 successfully established channels for Phase 2 simulations. Note that routing constraints prevent nodes 10, 11, and 12 from being saturated in this network. In Fig. 1(b), we have also highlighted one deterministic and one statistical channel. We shall examine the delays experienced by the packets on these two channels in Section V-C.

In Phase 2, a channel's packets arrived either at intervals equal to $x_{\min}$ or at intervals x_l that brought the average in a period of duration I to x_{ave}. If q is the probability of the interarrival time being $x_{\min}$, the longer interarrival time is given by

$$x_l = \frac{x_{\text{ave}} - qx_{\min}}{1 - q}. \qquad (16)$$

Thus, the durations of the peak activity periods, when the channel was sending packets at the peak rate, were geometrically distributed multiples of $x_{\min}$, while those of the idle periods were geometrically distributed multiples of x_l. Such an arrival pattern, characterized by short bursts of data separated by relatively long periods of silence, resembles those observed in current networks and is likely to be common in future ones as well. Random shifts in packet generation times were introduced with a small probability to avoid synchrony. The value of q was obtained from the channel parameter I by the relationship

$$q = 1 - \frac{x_{\text{ave}}}{I}. \qquad (17)$$

We also experimented with other arrival patterns, but this pattern had the worst possible delay characteristics among them. We will present only the results of this arrival pattern since the other arrival patterns produce qualitatively similar results. However, we will mention briefly the effect of different arrival patterns when we present our Phase 2 results.

B. Overhead

To estimate the overhead of channel establishment, scheduling and rate control, we measured the time taken by various routines in the simulator. The simulator was run on a VAX 8600.

1) Establishment Overhead: Calculating the overflow probability P_{do} for the statistical test could be the most expensive part of the simulation, if it were evaluated by brute-force enumeration of overflow combinations (see Section IV-B). An efficient method to obtain close upper bounds on the overflow probability involves a bookkeeping overhead of about 1 ms every time a channel is tentatively accepted or deleted from a node. We omit the details of this approximation method because of space restrictions. The other portions of the algorithm to be executed in each node during the channel establishment procedure are equally efficient. The deterministic test took about 66 μs on the average, and the delay bound computation took 3.02 ms. The time for a delay bound computation can be reduced significantly with respect to this value by using appropriate data structures. The destination host spent an additional 0.16 ms dividing the delays among the nodes in the path.

2) Scheduling and Flow Control Overhead: The overhead of scheduling can be broken up into two components: the time required to insert a packet in the schedule and the time to select the next packet to transmit. In our Phase 2 simulations, the mean time required to insert a packet in the schedule was 30 μs for a statistical packet and 40 μs for a deterministic packet. The extra overhead in the case of deterministic packets is due to alignment (see Section III-A). The time required to select the next packet for transmission was 60 μs. Distributed rate control took 40 μs on the average.

We expect a real implementation of scheduling to be much faster even on a VAX 8600. The CPU times consumed by our routines were high due to the floating point representation for delays and deadlines forced by CSIM. To avoid roundoff problems, two deadlines were considered equal if the absolute difference between their values was less than a defined minimum. This required calls to UNIX library routines.

To estimate the speedup that might result from using integers, we counted the number of VAX assembly instructions produced by the CSIM code. The transmission routine executed about 167 instructions in the case of a floating point deadline and only 79 with an integer deadline. Even if we assume that integer operations are as slow as floating point operations, the transmission overhead of an integer solution would be less than 30 μs on a VAX 8600. A similar reduction can be expected for the packet insertion routine and for the distributed rate control algorithm. Table III shows the cost of each stage in the transmission of packets.

In a packet switching network with multimedia traffic, there may be large packets (e.g., video frames with size of the order of 1 Mbit). On a gigabit/second network, it will take about 1 ms to put the bits of such a packet on the wire. If the switching hardware has a speed comparable to that of a VAX 8600, and conditions similar to those observed in simulation prevail, our scheduling overhead will be less than 10% of the transmission time and appears to be acceptable. For very small packets, scheduling may prove to be relatively expensive: for a 10 Kbit packet, overhead can be kept to 10% by using switching hardware that is 100 times faster than the VAX 8600.

In summary, our scheme seems to be feasible in the context of packet-switching wide-area network. Scheduling and rate control seem acceptably inexpensive, even under the worst case situation of Phase 2. They appear to be easily implementable in hardware, which should further reduce the overhead.

C. *Effects of Worst Case Assumptions*

To evaluate how pessimistic our assumptions are, we focus on the two channels shown in Fig. 1(b). In node 0, the service time of the statistical channel is 4 units, and its delay bound is 20 units. The service time of the deterministic channel is 1 unit, and its delay bound is 18.5 units. Fig. 2(a) shows the delay obtained by the deterministic and statistical channels at node 0. The probability of deadline overflow at node 0 for the statistical channel was 0.80. I was taken to be 20 times a channel's x_{ave}.

The actual delays have been normalized so that values greater than 1.0 correspond to missed deadlines. Overflow values for the statistical channel have been clustered immediately beyond 1.0. About 86% of the packets meet the deadline. On the other hand, if I were only 10 times x_{ave}, with a corresponding lower probability of arrivals at the maximum rate, the percentage of such packets would be as high as 91%. A low value of I causes packets to arrive smoothly at regular intervals. At higher values of I, we expect the actual overflow to become closer to the guaranteed value of $1 - z$ as the pessimistic conditions we assume are more likely to be met.

The effect of burstiness can be seen in Fig. 2(b). The value of q in (17) can be taken as a measure of burstiness. To differentiate it from burstiness measured as a ratio between peak and average arrival rates, we refer to it as traffic density. On the other hand, when the arrivals were nonbursty, e.g., with x distributed exponentially, or uniformly between $x_{\min}$ and $(2x_{\text{ave}} - x_{\min})$, almost no overflows were observed.

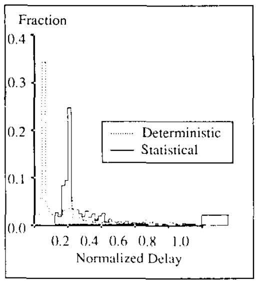
D. *Statistical Channels*

What does a network provider gain when a client requests a statistical channel rather than a deterministic one? Is a low value of Z more advantageous to the provider than a high value of Z? Does a higher delay bound allow

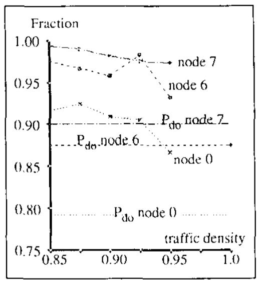
TABLE III
SCHEDULING COSTS

stage	observed mean time μs	instructions with library routines	instructions without library routines	expected mean time μs
flow control	40	112	73	26
insert stat. packet	30	$28\,n+13$	$13n+13$	24
insert det. packet	40	$28n+40m+13$	$13n+18m+13$	31
select packet to xmit	60	167	79	28

n: number of packets examined during scheduling
m: number of packets examined during alignment



(a)



(b)

Fig. 2. (a) The delay distributions of packets on the channels highlighted in Fig. 1(b). (b) The relation between the fraction of packets that meet the delay bound at a node and the traffic density q [see (16)].

the provider to accept more channels? In this section, we present simulation-based answers to these questions.

In the network of Fig. 1(a), we were able to establish 38 channels when they were deterministic or statistical with an equal probability. When all channel requests were deterministic, we could establish only 27 channels. On the other hand, when all channel requests were statistical with a z of 0.85 for each node on the path, we could establish as many as 46 channels.

To avoid the effects of topology, we studied a very simple network, consisting of just two nodes connected by a single bidirectional link. We used the same workload as in previous simulations but varied the Z parameter of the channels, which was taken to be the same for all the channels. The per-node probability bound z for a channel is the square root of the Z parameter in this case [see (11) and (15)]. The delay bound d in each node was equal to 20 units. Five sets of simulation experiments were conducted; in the first four, all channels were of identical types (see Table I), while in the fifth and more realistic case the channels could belong to any type; i.e., the workload was essentially that used for the simulations described in the previous sections. We show the relationship between the overflow probability at a node and the number of channels that can be established in Fig. 3(a).

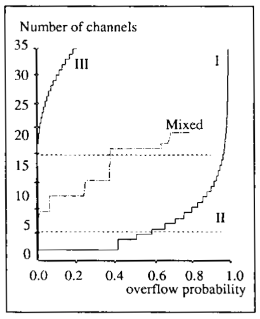
In order to utilize the network efficiently, we would like to accept more channels as channel clients are willing to reduce their probabilistic bounds on delay requirements.

In Fig. 3(a), we see that our algorithm cannot accept more channels of types II or IV even when the overflow probability is allowed to increase. This appears disturbing since the algorithm is unable to take advantage of the fact that the clients' delay requirements become less stringent as Z changes from 1 (deterministic) to 0 (best-effort), and the overflow probability at each node is allowed to increase. This phenomenon is explained in Fig. 3(b), which plots the total load of the channels established at a node versus the possible overflow probabilities. When all requests are of type II or IV, the load at node 0 (and also that at node 1) is less than 1. In this situation, an unsuccessful channel establishment request is rejected by the delay bound test and not by the statistical test. Hence, a change in the value of z does not affect the number of accepted channels unless the total load (i.e., the peak utilization) at a node reaches unity.

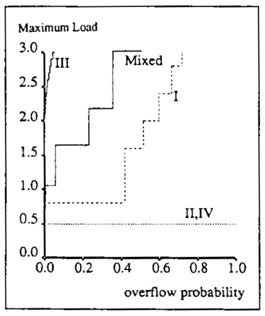
In our very simple network, the number of successful channels is determined by two parameters, $t/x_{\min}$ and $p = x_{\min}/x_{\text{ave}}$. When the overflow probability is 0, it is as if the channels were deterministic, and the number K_d of channels established at a node is given by

$$K_d = \min \left(\lfloor x_{\min}/t \rfloor , \ \lfloor d/t \rfloor \right). \tag{18}$$

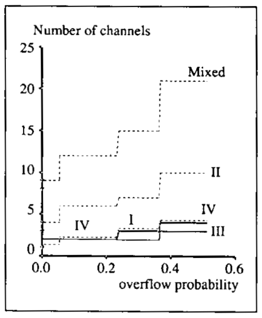
If the delay bound d is large enough so that scheduler saturation is not the bottleneck, we can establish K_s statistical channels with a probabilistic bound with parameter z, where K_s is the largest integer which satisfies the



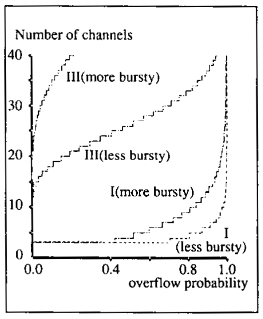
(a)



(b)



(c)



(d)

Fig. 3. (a) The number of statistical channels of each type that can be established at a node versus the permitted overflow probability (see Section IV-B) for the 2-node network. (b) The maximum load of a node permitted by our algorithm versus the overflow probability (see Section IV-B) for the 2-node network. (c) The components of the Mixed curve shown in Fig. 3(a). (d) The behavior of the algorithm for bursty traffic.

inequality

$$\sum_{k=K_d}^{K_s} \binom{K_s}{k} p^k (1-p)^{(K_s-k)} \leq z. \qquad (19)$$

On the other hand, if the delays are shorter, then the number of channels established at a node is K_d, irrespective of the value of z.

From Fig. 3(a) and (b) we can observe the following facts.

- If node saturation occurs before scheduler saturation, lower values of Z will allow more channels to be established; on the other hand, increasing delay bounds will not help the provider in this case;
- if scheduler saturation occurs before node saturation, changing Z is not likely to help the provider, whereas increasing delay bounds will do so.

An unsuccessful statistical channel client should retry with a lower value of Z if the request was turned down due to node saturation, and with a higher value of D if the request was turned down due to scheduler saturation.

When a channel can be of any type (each type chosen with equal probability), results are as shown in the "Mixed" curve in Fig. 3(a) and (b). Fig. 3(c) shows the behavior of the various component types of the "Mixed" curve. It is interesting to note that the number of type II or type IV channels accepted is no longer independent of the overflow probability. The exact number of channels established seems to depend on the sequence of requests.

E. Accommodating Burstiness

In the scheme described in this paper, channels can specify the burstiness of their traffic via the three parameters $x_{\min}$, x_{ave}, and I. Two parameters derived from these and also characterizing burstiness are p, defined in (2), and q, defined in (17). While p, the ratio between the average and peak throughput requirements of a channel, appears to be a simple measure of burstiness, q has similarities with the "traffic density" parameter found in, for example, [11]. Both p and q range between 0 and 1. A value of p close to unity indicates a smooth arrival pattern, while a value close to zero corresponds to burstier traffic. On the other hand, a higher value of q indicates more burstiness.

Our scheme works best when the traffic is bursty (high values of q, low values of p). With a lower value of p, the scheme accepts more channels. This can be seen in Fig. 3(d), which plots the number of channels accepted by the statistical test (see Section IV-B) for channels of different types (see Table I) with different degrees of burstiness. In this figure, the low burstiness versions of type I and III channels have an x_{ave} value equal to 30 instead of 60 time units, which causes p to increase from 1/6 to 1/3. As is clear from Fig. 3(d), we can exploit burstiness in traffic to accept more channels.

We have already examined the effect of q in Section V-C. In Fig. 2(b), the actual delays appear to get closer to the guaranteed bounds as q increases.

VI. Conclusions

This study was undertaken to determine the feasibility of offering real-time services in packet-switching wide-area networks. We defined the problem and the type of network to be studied, adopted the real-time channel abstraction, focused our attention on two types of performance guarantees such a service could provide, specified the meaning and extent of these guarantees, and selected a simple characterization of a channel's input packet stream. We argued for connection-oriented packet switching as the basis of a real-time service, as it seems very difficult or impossible to build real-time channels on top of a connectionless service.

A single-roundtrip procedure for establishing channels has been devised. The procedure entails several tests and tentative reservations of resources to be performed in each node along the channel's path. A channel that passes these tests can be guaranteed the type and value of delay bound requested by the client; these guarantees become effective as soon as the channel is established and remain in effect until it is disconnected. There is no need to verify that this is true while real-time services are being provided, nor to make on-the-fly adjustments to policies in the nodes or the hosts. Our simulation experiments have failed to disprove the correctness of the scheme even in worst case situations. Note that performance can be guaranteed because of the scheduling and distributed rate control policies adopted as well as because of the worst-case bounding arguments on which our establishment scheme is based.

Our approach tries to meet the correctness objective by introducing a number of worst case assumptions. One effect of these worst case arguments is the incomplete exploitation of the network's resources by the real-time service. However, the maximum node loads in our simulations reached high values in most nodes [see Fig. 1(b)]. The average utilizations depend on the burstiness of the traffic and the actual channel activity and packet arrival patterns. During Phase 2 simulations (i.e., under maximum offered real-time load), the node utilizations averaged over time were for several nodes in the neighborhood of 50%. An important point in favor of our approach is that, unlike what happens in the case of circuit switching, or STM (synchronous transfer mode), network bandwidth and node processing power not actually used by the real-time service are always available to other communication services or local processing tasks.

The differences in "cost" between deterministic and statistical channels depend on the guarantees that are requested by the clients (especially Z) and on the channel's characteristics (especially the burstiness of its traffic and its delay bound). A higher burstiness and a lower Z yield higher gains in terms of the maximum number of statistical channels the network can support.

The CPU overheads of scheduling [roughly 50–100 μs/packet on a VAX 8600 and for the maximally loaded network in Fig. 1(a)] and distributed rate control (about

26–40 μs/packet on a VAX 8600) seem to be quite reasonable; thus, the approximate amount of additional processing time a packet requires in each node of the simulated network, if that node has the power of a VAX 8600, is no greater than 140 μs. The execution of the scheduling and rate control algorithms will of course be speeded up in the future by the use of inexpensive high-speed machines as nodes, and even more (actually, much more) by the implementation of these algorithms in hardware, which is made easily feasible by their simplicity. The establishment computations are simple to implement and do not take more than a few milliseconds at each node.

Many problems remain to be explored in the field of wide-area real-time communication. Some of the following are being investigated, while the others will be in the near future:

- buffer allocation and management policies suitable for the type of real-time service described in this paper;
- the best way to guarantee a given bound on delay variance or delay jitter;
- the possibility of devising correct algorithms for the establishment of channels whose traffic is described by parameters different from $x_{\min}$ and x_{ave}, or is dependent on the traffic of other channels;
- the introduction of security, fault tolerance, accounting, and charging capabilities into the design of a real-time service;
- a procedure to be used for *fast channel establishment*, i.e., for setting up a channel while delivering the first packet on that (not yet existing) channel to the destination.

Appendix

Theorem: Let the following inequalities be satisfied in a nonsaturated node n traversed by K deterministic or statistical channels:

$$x_{\min,i} \geq \sum_{j=1}^{K} t_{j,n}, \quad (i = 1, \cdots K) \qquad \text{(A.1)}$$

and let T be the largest of the service times of the packets that may pass through the node but are not traveling on deterministic or statistical channels whose local delay bound is smaller than the sum of the service times of all deterministic and statistical channels through the node. Also, let the smaller delay-bound channels be numbered 1 through u in the order in which they would be shipped by the scheduling algorithm described in Section III-A if they all arrived at the same time at node n. Then, scheduler saturation is impossible if and only if

$$d_{i,n} \geq \sum_{j=1}^{i} t_{j,n} + T, \quad (i = 1, \cdots, u). \qquad \text{(A.2)}$$

Proof: Since we have excluded the possibility of node saturation even in the worst case (i.e., when all the channels we are considering are carrying packets at their maximum rates $1/x_{\min,j}$), there cannot be any buildup of queues in time; we can therefore assume that the node is empty when we start examining arrival patterns, and call time 0 the instant at which the first packet arrives at the node. Arrival times can be assumed to be arbitrary, since channels are supposed to be independent of each other; the dependencies that may result from the sharing of an input link by two or more channels can only improve the situation, as this sharing serializes arrivals on those channels at the node. Packets arriving on a given channel after the first we consider are not independent of that first: the second packet on channel j will in the worst case arrive $x_{\min,j}$ time units after the first packet on the same channel. Because of assumptions (A.1), no deadline for a subsequent packet will fall within the interval between time 0 and time $\Sigma t = \Sigma_{j=1}^{K} t_{j,n}$. Subsequent packets can therefore be ignored in this proof.

i) *Only if* part. We prove that, if condition (A.2) is not satisfied for some i, there is at least one arrival pattern that causes scheduler saturation. Let a packet with a service time T arrive at time 0, and packets from all other channels numbered 1 through u arrive at time 0+. Then, the packet on channel i will be completely shipped only at time $\Sigma t + T$, which is greater than its deadline $d_{i,n}$.

ii) *If* part. We prove that, if conditions (A.2) are all satisfied, there cannot be scheduler saturation. Let the packet on channel i arrive at time 0; in this case, its latest possible departure time is $\Sigma_{j=1}^{i} t_{j,n} < d_{i,n}$. If it arrives at 0+, the latest departure time is $\Sigma_{j=1}^{i} t_{j,n} + T \leq d_{i,n}$. With an arrival time between 0+ and T, the maximum time spent by the packet in the node is less than $d_{i,n}$. Arriving at T+, this maximum time is $\Sigma_{j=1}^{i} t_{j,n} < d_{i,n}$, and later arrivals yield even smaller maximum times. Q.E.D.

If condition (A.1) is not satisfied for all channels, we apply the above theorem to a set of channels that includes some subsequent packets. Since considering these packets increases Σt, we calculate the new value of this sum, $(\Sigma t)'$, from the following recursive definition:

$$\left(\sum t\right)' = \sum t$$
$$\left(\sum t\right)' = \left(\sum t\right)' + Y_k t_{k,n},$$
$$Y_k = 1 \text{ if } x_{\min,k} + d_{k,n} < \left(\sum t\right)',$$
$$Y_k = 0 \text{ otherwise.} \qquad \text{(A.3)}$$

Sets U and V (for their definition see Section IV-C) are determined by the value of $(\Sigma t)'$. The arrival times of some of the packets to be considered are not independent of those of the others. However, it is easy to see that the independence assumption is, once again, worst case (as it may declare scheduler saturation to be possible when instead it is not). Applying the theorem to this case while ignoring dependencies is therefore safe, since it does not endanger any of the performance guarantees provided to the clients of the service.

Acknowledgment

The authors are grateful to D. Anderson for proposing the parameterized message channel abstraction, which is the basis of the real-time channels discussed in this paper. They are also indebted to the members of the DASH and Tenet groups at Berkeley and to Referee 2 for the many useful comments and suggestions. Special thanks are due to K. Shin and D. Kandlur for reporting to the authors' attention an error in a previous version of the theorem proven in the Appendix.

REFERENCES

[1] D. P. Anderson and D. Ferrari, "An overview of the DASH project," Rep. no. UCB/CSD 88/406, Univ. California, Berkeley, Feb. 1988.
[2] D. P. Anderson, "A software architecture for network communication," in *Proc. 8th Int. Conf. Distrib. Comp. Syst.*, San Jose, CA, June 1988, pp. 376-383.
[3] T. M. Chen and D. G. Messerschmitt, "Integrated voice/data switching," *IEEE Commun. Mag.*, vol. 26, pp. 16-26, June 1988.
[4] D. E. Comer and R. Yavatkar, "FLOWS: Performance guarantees in best effort delivery systems," in *Proc. INFOCOM*, Ottawa, Canada, Apr. 1989, pp. 100-109.
[5] D. Dykeman and W. Bux, "Analysis and tuning of the FDDI media access control protocol," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 997-1010, July 1988.
[6] D. Ferrari, "Real-time communication in packet switching wide-area networks," Tech. Rep. TR-89-022, Int. Comput. Sci. Inst., Berkeley, May 1989.
[7] E. Harrington, "Voice/data integration using circuit-switched networks," *IEEE Trans. Commun.*, vol. COM-28, pp. 781-793, June 1980.
[8] J. F. Kurose, M. Schwartz, and Y. Yemini, "Multiple-access protocols and time-constrained communication," *ACM Comp. Surveys*, vol. 16, no. 1, pp. 43-70, Mar. 1984.
[9] F. E. Ross, "FDDI—A tutorial," *IEEE Commun. Mag.*, vol. 24, pp. 10-17, May 1986.
[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," *J. ACM*, vol. 20, no. 1, pp. 46-61, Jan. 1973.
[11] H. Ohnishi, T. Okada, and K. Noguchi, "Flow control schemes and delay/loss tradeoffs in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 6, Dec. 1988.
[12] G. M. Parulkar and J. S. Turner, "Towards a framework for high speed communications in a heterogeneous networking environment," in *Proc. INFOCOM*, Ottawa, Canada, Apr. 1989, pp. 655-668.
[13] H. Schwetman, "CSIM reference manual (Revision 12)," MCC Tech. Rep. no. ACA-ST-252-87, Nov. 1987.
[14] K. C. Sevcik and M. J. Johnson, "Cycle time properties of the FDDI token ring protocol," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 376-385, Mar. 1987.
[15] L. Zhang, "Designing a new architecture for packet switching communication networks," *IEEE Commun. Mag.*, vol. 25, pp. 5-12, Sept. 1987.

Domenico Ferrari (M'67-SM'83-F'87) received the Dr.Ing. degree in electronics engineering from the Politecnico di Milano, Italy, in 1963.

He became an Assistant Professor at the Politecnico in 1967, and was awarded the *Libera Docenza* in computer science in 1969. In 1970, he joined the faculty of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, where he is now Professor of Computer Science. From 1977 to 1979, he was Vice-Chairman for Graduate Matters of the Department of Electrical Engineering and Computer Sciences at Berkeley. From 1983 to 1987, he served as Chairman of the Computer Science Division and Associate Chairman of the Department. After working in such areas as high-speed computer arithmetic, switching theory, and computer-aided design, he became interested in computer systems performance analysis at the time of his move to California, and has for most of his research and teaching career concentrated on the problems of systems performance evaluation. For the last several years, his research has been primarily concerned with distributed system and network performance issues.

Dr. Ferrari has been, since its inception in 1988, Vice President and Deputy Director of the International Computer Science Institute, an organization located in Berkeley and dedicated to advanced research and international cooperation in computer science. He received the 1987 A. A. Michelson Award of the Computer Measurement Group for "outstanding contributions to computer metrics." He is the author of *Computer Systems Performance Evaluation* (Prentice-Hall, 1978) and a coauthor of *Measurement and Tuning of Computer Systems* (Prentice-Hall, 1983). He has been an Editor of *Performance Evaluation* since 1980 and of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS since 1989, and was on the Editorial Board of the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING from 1984 to 1987.

Dinesh C. Verma was born in Deoria, India. He received the B.Tech. degree from Indian Institute of Technology, Kanpur, India, in 1987. He was awarded the President of India's Gold Medal for scholastic achievement that year. Since then, he has been with the University of California at Berkeley as a Ph.D. student.

During the summer of 1989, he worked as a guest research scientist for GMD, Berlin, West Germany, on BERCIM, a networking project for computer integrated manufacturing. His interests include computer networks, real-time communication, performance evaluation and operations research.