Real-Time Communication in Packet-Switching Wide-Area Networks

Domenico Ferrari

Computer Science Division
EECS Department
University of California
and International Computer Science Institute
Berkeley, California

Abstract

The increasing importance of distributed multimedia applications and the emergence of user interfaces based on digital audio and digital video will soon require that computer communication networks offer real-time services. This paper argues that the feasibility of providing performance guarantees in a packet-switching wide-area network should be investigated, and describes a possible approach. We present a model of the network to be studied, formulate the problem, give a definition of the guarantees to be provided, and describe a scheme for the establishment of real-time connections with deterministic, statistical, and best-effort delay bounds.

This research has been supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089, the University of California with a MICRO Program grant, 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 author, 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.

1. Introduction

The current evolutionary trends of computing and networking technologies point to a major increase, over the next several years, in the need for real-time communication, i.e., for computer communication with guaranteed performance. Performance guarantees are required not only in distributed process-control and military systems, but also in supercomputer-to-supercomputer communications and even in such business applications as stock trading. Furthermore, guarantees will be needed by all those systems that process and transmit digital audio, digital video, or any combinations of each with other types of information. Multimedia distributed systems are expected to emerge in the near future, and to become predominant shortly thereafter, as the newer audio- and video-based user interfaces are added to the traditional ones (data, text, and images).

Providing performance guarantees typically means guaranteeing that, given certain bounds on the flow of information into the network, the delay with which each information item reaches the destination will satisfy given bounds. For interactive audio and video, the bound will usually be on the value of the delay of each bit, or byte, or message transmitted by a given sender to a given receiver; for instance, it is well-known that, in a phone conversation, delays longer than half a second are sufficient to make the speakers uncomfortable. In the case of non-interactive audio and video communication (e.g., the transmission of music or a movie for immediate or deferred consumption by the receiving user), performance indices more important than the actual delay are its variance and, even more, its jitter. The delay jitter with which a packet arrives at its destination may be defined as the absolute value of the difference between the packet's actual delay and its target (i.e., desired, ideal) delay. Once a network or internetwork starts offering real-time services, other applications of these services are likely to emerge; for example, it may be desirable to have some remote procedure calls and their replies, and some accesses to remote files or databases, performed within certain maximum time intervals. The availability of real-time services, if their prices are not prohibitive, is likely to expand the demand for them well beyond the boundaries of those applications that are motivating their introduction.

Some of the solutions proposed for the problem of designing network protocols and management policies to support real-time communication apply to local-area networks. They include such media access protocols as the Fiber Distributed Data Interface (FDDI) and the High Speed Ring Bus (HSRB); both of them provide a guaranteed delay bound, FDDI using the timed token scheme [Ross86], and HSRB the reservation/priority mechanism [SAE86][1]. Circuit switching can be used for this purpose in wide-area networks as well as in LANs: by dedicating a circuit of fixed bandwidth to each connection for the entire duration of the connection we can bound delays. However, circuit switching is not the most convenient technique for data communication [Harr80]. Hence, schemes that combine packet and circuit switching features in various ways have been proposed for integrated voice/data networks: fast circuit switching, burst switching, hybrid switching, and fast packet switching. An overview of these schemes can be found in [Chen88]. The most promising of them is fast packet switching, which is based on a major simplification of the lower layers in the protocol hierarchy; this simplification yields real integration of services (at the lower layers, all packets are treated in the same way, no matter whether they contain data or voice, or perhaps even video) as well as the possibility of protocol implementation in hardware. It is important to notice, however, that fast packet switching can provide low but not necessarily bounded delays, and therefore does not per se offer true real-time communication.

[1] Many other approaches to providing some form of real-time communication can be described as *best-effort* methods, that is, methods in which the system tries to make sure that most messages will satisfy their deadlines. These approaches include virtual-time [Zhao87] and window-based [Kuro88] [Zhao88] access methods for CSMA/CD networks. [Stro89] provides delay guarantees by controlling access to a token ring network on the basis of prior knowledge about the arrival patterns of messages.

This paper addresses the problem of providing performance guarantees in a wide-area network or internetwork. The solution we present applies to all current and future networks that are adequately represented by the model in Section 2. The formulation of the problem and our approach to its solution are discussed in Section 3. An algorithm that enables a network of the type defined in Section 2 to offer real-time services is described in Section 4. Finally, Section 5 presents our conclusions and plans for future work.

2. The Network

In its full generality, the network we consider in this study has an arbitrary topology, and may actually result from the interconnection of several networks of various types (LANs and WANs). 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 hosts or by special purpose communication computers or by both. A source-destination path goes through a number of nodes where the transmitted information can be stored and then forwarded to the next node. Some of the links between adjacent store-and-forward nodes may actually be non-store-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. It should be noted that this assumption is not exactly satisfied by links governed by contention-based protocols (e.g., Ethernets, see [Kuro84]), but it is by other types of LANs, such as FDDI rings (see for example [Sevc87], [Dyke88]).

We believe that the model presented above, though not absolutely general, is an adequate characterization of many current and future wide-area networks. How long in time will its validity last is an open question: at some point in the future, store-and-forward nodes might disappear altogether, or some of the links might exhibit unboundable delays (even now, the presence of an Ethernet along the path of a real-time packet raises the specter of invalidity when absolute delay bounds are desired). However, it is hard to predict when the number of networks satisfying our model will become negligible.

A network with these properties could be based on circuit switching, or, since circuit switching is a rather wasteful technique in data communications, on hybrid switching. In the latter case, circuits would be allocated to real-time connections, while non-real-time communications could use the remainder of the bandwidth of each link in packet switching mode. If the boundary between the two portions of a link's total bandwidth is made movable as a function of demand, this solution can exploit network resources more efficiently, but requires fairly complex switching in each node. This in turn increases the difficulty of designing the very-high speed switches that the higher network bandwidths of the future will require.

A solution in which the two types of service (real-time and non-real-time) would be more intimately integrated is one based on packet switching, for example, of the fast packet switching variety mentioned above. But can such a network guarantee performance? This is the main question we plan to answer in this paper, by trying to provide a constructive proof of feasibility.

The first question we have to address is that of the model to be assumed for the switches (or nodes) in the network. For the treatment in this paper, we shall choose a simple node model: a single queueing server with multiple queues, as shown in Figure 1(a). The functions of the various queues will be explained in the sequel. We shall assume that the service time of each packet in the node server is known to, or easily predicted by, the node.

Another problem to be considered is whether a real-time service can be built at the transport or higher level in the protocol hierarchy on top of a datagram service. We believe that packet delay is much harder to control if each packet going from a given sender to a given receiver can in principle follow any route, and some of these routes may become congested while one or more real-time packets are in transit. In this paper, we therefore plan to restrict our attention to the case in which all packets belonging to a real-time channel always follow the same route. A possible approach, based on source routing, is one in which the source selects the path to each destination and keeps it fixed for the duration of the communication. However, the same result can be achieved by building real-time services on top of a connection-oriented (virtual-circuit) service offered by the network layer. The latter approach will more easily permit reservation of resources in each node along the route, which we believe is necessary to guarantee delay bounds. Even the *flow* abstraction in [Come88] uses a fixed route scheme with resources reserved in advance, in spite of the authors' declared preference for a connectionless approach.

A major drawback of connections is that they have to be established before any communication can take place; this may, on the one hand, require a time longer than the client is able to wait in some real-time applications, and on the other hand make the shipment of a small amount of urgent information by a real-time service slower than by normal datagrams. In the study described here, we have tried to minimize the call setup time by designing an efficient connection establishment scheme.

3. A Formulation of the Problem

As discussed in the previous section, we study the feasibility of supporting performance guarantees in a wide-area network consisting of store-and-forward nodes interconnected by bounded-delay links. The network is assumed to offer connection-oriented service (as well as a connectionless service for those non-real-time communications where this service is preferable) at the network layer.

To formulate the problem in more detail, we introduce the notion of *real-time channel*: a simplex connection between a sender and a receiver that delivers messages (or packets) in sequence and is characterized by performance and reliability parameters. The client (i.e., the entity that requests the establishment of a channel) specifies the values of the parameters to communicate its needs to the service provider, i.e., ultimately, to the network layer. Note that this notion is very close to that of *real-time message stream* presented in [Ande88], except that the latter includes also parameters that are not of the performance or reliability types.

We are concerned here only with the performance-oriented parameters of a channel:

- the channel's capacity, defined as the maximum amount of information that may be outstanding at any given time in the channel;
- the maximum packet size;
- the maximum packet service time in each node;
- the delay bound or bounds for the channel's packets[2];
- the maximum packet loss rate.

[2] Our approach is similar in several respects to the one based on *virtual clocks* described in [Zhan90]. However, there are also many substantial differences, the most macroscopic of which perhaps is the fact that we allow clients to specify any delay bounds they may need along with their throughput bounds (the latter through capacity and packet size

While capacity and size are to be enforced by the client, delay bound and loss rate, whose values are to some extent interdependent [Sumi88], are to be guaranteed by the provider.

There are various types of channels, corresponding to the different types of delay bounds. In [Ande88], the following bounds are considered:

- *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 Z;
- *best-effort*: the bound D is not guaranteed by the provider, which, on the other hand, promises to do its best to satisfy it.

Note that the delay guarantees are assumed to be valid only for those packets that reach the destination. Delay bounds will not apply to packets that are allowed by the given maximum packet loss rate to go undelivered due to buffer overrun or to failures affecting any part of their channel. 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, for instance because the new channel is expected to carry information supplementing that carried by another channel, which must therefore be somehow synchronized with it, 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.

Two important characteristics of the network are to be discussed before describing our approach: flow control and packet scheduling.

First of all, should flow control be window-based or rate-based? Both approaches to flow control seem feasible, but the rate-based one (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) looks more attractive. Indeed, this solution does not require flow control acknowledgements, which would have to use another simplex channel, and would therefore be quite expensive. Alternatively, they could be sent as datagrams, in which case they may 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 appropriate resources can be committed, and, in particular, the receiver can check whether it will be able to accept packets at the rate declared by the sender. This verification is made possible by the deadline scheduling policy to be used in the receiver as well as in all the nodes (see below); in fact, the ability of the receiver to handle yet another stream of arriving packets can be verified with the same method (described in Section 4) used to test the ability of each intermediate node.

If flow control is rate-based, it is more convenient to replace the capacity parameter with one or more parameters describing the packet arrival (i.e., input) process. Here, we adopt for this purpose the parameters x_{min}, the minimum packet interarrival time on the channel, and x_{ave}, the minimum value of the average packet interarrival time over an interval of duration I. In other terms, x_{ave} is the average interarrival time during the channel's busiest interval of duration I. The ratio between these two parameters is the simplest possible measure of burstiness for the incoming packet stream. The amount of information ''stored'' in a channel is indeed hard to control on the part of the sender without a sliding-window mechanism, especially in a statistical channel, and even more in a best-effort one; the sender, on the other hand, has full control over, or at least full knowledge of, the packet generation rate, and can make sure that $1/x_{min}$ is never exceeded.

---

parameters), whereas the virtual clock method is only concerned with providing average throughput guarantees.

There is, however, the danger that a malicious user will circumvent any flow control mechanism and send into the network packets at a much higher rate than the declared maximum value, $1/x_{min}$, or maximum average value, $1/x_{ave}$. Indeed, current (and future) distributed systems include personal machines, whose operating system can easily be modified or even replaced by their owner or by an occasional user. The same effect might be caused by a failure in the sending host, even when this is a multi-user, 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. Thus, a distributed flow control scheme seems absolutely necessary; with the characterization of the input we have selected, the scheme will have to make sure that neither $1/x_{min}$ nor $1/x_{ave}$ are exceeded. One possible solution to this problem is presented in Section 4.

Scheduling in the hosts and in the nodes will be deadline-based. More precisely, we adopt a modification of the EDD (Earliest Due Date) policy that gives, in the case of a conflict, priority to deterministic over statistical channels, and to statistical over best-effort channels. As illustrated in Figure 1, this is not an absolute priority: the packet at the head of the deterministic queue is shipped before the packet at the head of the statistical queue only when the latest possible time for starting its shipment occurs before the statistical packet's deadline; otherwise, the latest possible time for starting the shipment of the statistical packet is compared with the deadline of the first packet in the best-effort queue, and the latter is shipped if its deadline is the earlier of the two times. Only the packets in the last (non-real-time) queue have to wait to be shipped until the other queues are empty; also, their shipment can be preempted, while the others cannot. Note that, as shown in Figure 1, deterministic packets have to be "aligned", i.e., their deadlines must to be suitably decreased if they fall into the last possible interval during which the node can safely process another deterministic packet. Without alignment, the deadlines of deterministic packets could not always be met.

4. An Approach to the Solution

The channel establishment mechanism we propose for the type of network described in Section 2 may be built on top of any procedure that can be used to set up connections. It is reasonable to design the routing algorithm for channel establishment so that it will exploit any available information about the delays along the various possible paths to the destination and about the "real-time load" of neighboring nodes. 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 either 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. Thus, acceptance of a channel establishment request cannot be guaranteed. The probability that a request is accepted depends on how close to saturation (in terms of real-time peak load) each node along the route is. This probability cannot be easily estimated, unless the source has a complete and up-to-date knowledge of the node loadings and of the probabilities of the various possible routes. An intelligent routing algorithm could increase the probability of success considerably. What can, instead, be guaranteed is a bound on the response time to a request, though the bound would be provided by the server, not chosen by the client.

If all the tests succeed at all nodes and at the destination host, this host subdivides the delay bound D (and the probability Z of not exceeding the bound if the channel being established is statistical) among the nodes traversed by the channel, after subtracting the total delays in the links along the route. Let d be the maximum delay bound assigned to the channel in a node, and z the minimum probability that d will not be exceeded in that node. A packet traveling on that channel and arriving at that node at time t will usually be assigned a node deadline equal to $t+d$. To satisfy the overall delay bound D, it is sufficient (though not necessary) to satisfy the bound d (or d, z) in each node along the route. For simplicity, we assume 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 all the reserved resources. The delay bound assigned to a node is then used to compute the deadline in that node for each packet traveling on that channel (see the calculation of dl in Figure 2). When the reply message reaches the source host, this host learns that the requested channel has been set up, and can start using it.

Note that, in the procedure just described, the destination host assigns delay bounds to nodes since we assume that the source host does not know how long the path to the destination will be. If the number of hops were known at the source beforehand, this assignment could be done by the source host, but it would be done less effectively, due to the source's incomplete knowledge of each node's loading situation.

Before discussing the tests to be performed in each node, we have to choose a suitable flow control mechanism. One possible approach, whose execution costs are to be carefully investigated, consists of increasing the deadlines of the "offending" packets (i.e., packets arriving sooner than expected), so that they will be able to go through lightly loaded nodes fairly rapidly, but will be delayed in heavily loaded nodes, where they would 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, at least some of the buffers will have to be statically allocated to prevent the offending packets from flooding the buffer space of a heavily loaded node and causing packets from other channels to be dropped.

One simple algorithm to increase the deadlines of packets arriving too soon after their predecessors is shown in Figure 2. The algorithm does not need any timers, which may be expensive to maintain. Note that the actual intervals between successive arrivals of a channel's packets at a node may be occasionally shorter than x_{min}, and the averages of the same intervals over I occasionally shorter than x_{ave}, either because of a sudden decrease of the load on the previous node along their path or (if the node in question is the first on the path) because of a higher packet rate illegitimately generated by the sending host. However, the distributed flow control algorithm in Figure 2 acts on the packets' deadlines so as to impose on the node the same switching and transmission load as if the packet stream on the channel obeyed the x_{min} and x_{ave} constraints. We can therefore assume that each channel going through the node satisfies those constraints.

To simplify our discussion, we shall assume here that the buffer space available for real-time connections in the network's hosts and nodes is unlimited. This assumption will allow us to ignore the loss rate parameter, and to postpone the treatment of buffer space allocation and management 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*, required only when the channel to be established is deterministic, and involving only the deterministic channels already existing in the node;

(b) the *statistical test*, required for the establishment of both statistical and deterministic channels if at least one statistical channel is present in the node or is to be set up, and involving all deterministic and statistical channels in the node;

(c) the *delay bound test*, which is needed in all cases, and, if successful, is followed by the computation of the minimum delay bound for the channel in the node.

If the request passes the tests, then the node performs a *delay bound computation*, and then sends the establishment message on to the next node.

Note that best-effort channel requests are not subjected to any tests. Hence, they are always accepted. In the sequel, unless explicitly specified, by the term "channel" we shall refer only to deterministic or statistical channels.

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 must be true 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 by channel i, whose packets have a maximum service time in the node (see Section 2) equal to t_i, is $t_i / x_{min,i}$. Thus, the condition to be tested is:

$$\sum_j t_j / x_{min,j} \le 1, \tag{1}$$

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

The **statistical test** has two purposes:

(i) to determine whether for each statistical channel j in the node the probability of a delay higher than the bound d_j is below its maximum tolerable value, $1 - z_j$;

(ii) to provide the destination host with the information that is necessary to compute z_i for the new channel if this is statistical.

Both these purposes can be obtained by computing the *probability of deadline overflow* P_{do}. A packet may be delayed beyond its delay bound in a node because of two reasons: a temporary saturation of the node’s processing and switching capacity (*node saturation*), and impossible scheduling constraints (*scheduler saturation*). The latter is avoided, as discussed below (in our description of the delay bound test), whereas the statistical test deals with the former.

A measure of the probability that channel j is active[3] during an interval I is

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

Given m independent channels 1, 2, ... m passing through a node, the probability that the members of a subset C of them are simultaneously active is given by

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

[3] The packet arrival pattern on a channel j that minimizes the probability of a deadline overflow is one in which all $I / x_{ave,j}$ packets in I arrive at intervals equal to $x_{min,j}$, while no packets arrive during the remainder of I. In such worst-case situation, channel j is said to be *active* during the periods in which arrivals are occurring. The variable p_j defined in (2) is the fraction of I during which channel j is active.

Let at least one of the channels be statistical. To compute P_{do}, 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, will 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 of both types, is not satisfied.

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

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

We can then check whether the following inequalities are satisfied:

$$P_{do} \le 1 - z_j, \tag{5}$$

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

$$P_{do} \le \min(1 - z_j),\ or\ 1 - P_{do} \ge \max z_j. \tag{6}$$

If test (6) is successful, then the value of P_{do} is sent to the destination host for the purpose referred to in (ii) above. The use of this value by that host is discussed below (see (12)). The destination host will try to assign to channel i in the node a value of z_i that satisfies inequality (6).

Figure 3 shows two examples of statistical test. The node being considered has four channels (numbered 1 through 4) already established when it receives a message requesting the establishment of channel 5. This is a deterministic channel whose addition would cause node saturation whenever all channels have maximum rate traffic on them for a sufficiently long time (see Figure 3(a)). Since, as shown in Figure 3(b), the probability of deadline overflow is 0.000161, and its complement to 1 (0.999839) is greater than the z's of the two statistical channels (channels 2 and 3), the node passes the statistical test. The value of P_{do} is not transmitted to the destination host in this case, as channel 5 is deterministic.

The same figure also shows in (a) the characteristics of another new channel, called channel 6, and in (b) the calculation of P_{do} according to equations (3) and (4), under the assumption that channel 5 has been established. The node passes the statistical test also for channel 6, and transmits the value of P_{do} (0.000943) to the destination host, since channel 6 is statistical.

The **delay bound test** determines whether scheduler saturation can be avoided in the node, and, if so, the minimum delay bound to be assigned there to the channel being established so that this goal will be achieved. We have to eliminate this type of saturation for deterministic channels, since we could not provide absolute guarantees otherwise. For statistical channels, we try to avoid it primarily because of the complexity of dealing with it in the general case. 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, the scheduler will in no way be able to schedule them so that their deadlines are met.

To determine whether scheduler saturation is possible in a node, we divide the m (deterministic or statistical) channels passing through the node into two sets: we call A the set of those channels whose delay bound in the node is lower than the sum of all channel service times, and B the set of those channels whose delay bound in the node is greater than or equal to that sum:

$$\mathrm{A} = \{ i \mid i = 1, \ldots a;\ d_i < \sum_{j=1}^{m} t_j \}, \tag{7}$$

$$\mathrm{B} = \{ l \mid l = a+1, \ldots m;\ d_l \ge \sum_{j=1}^{m} t_j \}, \tag{8}$$

With no loss of generality, we number the a channels in A according to the order in which they would be scheduled by the algorithm in Figure 1 if they arrived all at the same instant: 1 will be the channel that would be shipped first, a the one that would be shipped last.

Let us assume that the node is not saturated: we have already taken into account node saturation, which delays packets beyond their bounds whether or not scheduler saturation is present, in the statistical test; also, let us assume for simplicity of proof that

$$x_{min,l} \geq \sum_{j=1}^{m} t_j \tag{9}$$

for l=1,...m. This assumption is equivalent to postulating that no subsequent arrivals of packets at the node will interfere with the transmission of those (one per active channel) we are considering. A discussion of how to proceed when the assumption is not satisfied can be found in the Appendix. The following theorem, on which we will base the delay bound test, holds.

Theorem 1: Scheduler saturation is impossible if and only if

$$d_i \geq \sum_{j=1}^{i} t_j + \max_{i<k\leq m} t_k, \quad (i=1,...a). \tag{10}$$

Proof: Sketched in the Appendix.

The following corollaries can be easily derived from Theorem 1.

Corollary 1: If B is empty (a=m) and A is non-empty ($a > 0$), there is at least one packet arrival pattern that causes scheduler saturation.

Proof: When i=a=m, we cannot have $i < k \leq m$, hence the second term on the right-hand side of (10) equals 0, and $d_a \geq \sum_{j=1}^{m} t_j$. By (7) and (8), channel a must be a member of B, and this contradicts the assumption that B is empty.

Corollary 2: If B has only one member (a=$m-1$) and A is non-empty ($a > 0$), there is at least one packet arrival pattern that causes scheduler saturation.

Proof: When $i = a = m-1$, (10) yields $d_a \geq \sum_{j=1}^{m-1} t_j + t_m = \sum_{j=1}^{m} t_j$. By (7) and (8), channel a must also be a member of B, and this contradicts the assumption that B has only one member.

Corollary 3: If A is empty (a=0), scheduler saturation is impossible.

Proof: Inequalities (10) are trivially satisfied in all cases for $a<i\leq m$.

The table in Figure 4(a) lists the four channels 1, 2, 3, and 4 in Figure 3(a), ordered by increasing delay bound. The fourth column (r_i) displays the value of the right-hand side of condition (10) for those channels that are members of A. Since $d_i \geq r_i$ for all i, there is no scheduler saturation. The addition of channel 5 causes node saturation whenever packets from all channels are simultaneously present in the node.

Does the addition of channel 5 also cause scheduler saturation? If this type of saturation is either impossible or possible only when there is also node saturation, then the establishment of channel 5 may be allowed to proceed towards the destination. Thus, the delay bound test begins by checking (using Theorem 1) whether scheduler saturation could occur after the addition of the new channel. If the answer is negative, the test is over; the value of the minimum delay bound that can be assigned to the new channel in the node is obtained as a byproduct from the test itself. This is the case of our example: the table in Figure 4(b) shows that channel 5 will not cause any scheduler saturation provided d_5 is greater than or equal to 11 time units; if it were smaller, by Corollary 2 there could be scheduler saturation. Thus, the value of $(min\ d)_5$ to be transmitted to the destination host is 11. Assuming that that host assigns a delay bound of 16 to channel 5 in the node, the addition of channel 6 will not cause any scheduler saturation either, as shown in Figure 4(c). The delay bound computation in this case yields a value of 10 for $(min\ d)_6$.

If, on the other hand, the test reveals the possibility of scheduler saturation, two cases are to be considered. In the case of no node saturation, the new channel's request is rejected. In the other case, i.e., when we are dealing with an overflow combination, we have to apply the test to subsets of the plenary combination (i.e., the combination including all channels in the node as well as the one to be established). It can be shown that, given a combination of channels that does not suffer from scheduler saturation, no subset of it will ever cause scheduler saturation. We can therefore delete one channel in turn and test each of the non-plenary combinations obtained in this way. If one of them causes scheduler saturation but no node saturation, the channel request is rejected. Those combinations that are exempt from scheduler saturation are discarded. Each of the remaining ones is subjected to the same algorithm recursively, until either the channel request is rejected or all are discarded, in which case the new channel passes the test successfully, and the establishment message is allowed to proceed.

Figure 5 illustrates the establishment of channel 6 on a 3-node connection. The characteristics of node 1 (with channels 1 through 5, and $d_5 = 16$) are displayed in Figure 3(a); those of nodes 2 and 3 in Figure 5(a) and (b), respectively. Figures 5(c) and 5(d) show the tables built by the delay bound computations performed in node 2 and 3, respectively. Note that all existing channels in both nodes are members of set B. Channel 6 will become a member of A in node 2 only if it is assigned a delay bound smaller than 12 in that node, and in node 3 only if its delay bound there is smaller than 11.

The destination host, if and when it receives the establishment request message, is to determine whether the total value of Z can be factored into node contributions z_k ($k = 1, 2, ...n$):

$$Z = \prod_k z_k, \tag{11}$$

The destination host can answer the question about whether Z can be factored as in (11) by checking whether

$$\prod_k (1 - P_{do,k}) \geq Z. \tag{12}$$

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, and the delays will grow without bounds while the node is saturated. 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 two important questions this observation raises are (i) how large this reduction is going to be in practice, and (ii) whether a less conservative approach could offer similar performance guarantees. While question (ii) will be the subject of a future investigation, question (i) has been studied by simulation techniques. The results of the study have been reported in [Ferr90]. The extensive simulations we have run have failed to disprove the correctness of our scheme even in the worst cases. The maximum node loads (as given by the sum in (1)) reached reasonably high values even in a non-optimally designed 14-node network with four types of channels characterized by peak-to-average rate ratios between 3 and 6 and node service time ratios between 1 and 4. Maximum loads with a mixture of deterministic and statistical channels ranged between 0.72 and 2.47; maximum utilizations by real-time traffic were roughly between 20% and 60%. With equal probabilities of requesting deterministic and statistical channels, our sample network could offer the required guarantees to 38 channels; with all channels being deterministic, only 27 could be created, but, if they were all statistical, that number climbed to 46. The overhead we measured on a VAX-11/8600 (the code used for scheduling, rate control, and channel establishment in the simulator was non-optimized code that could be used in real switches) were quite modest: in our maximally loaded sample network, scheduling took 50-100 microseconds per packet per node, while rate control took 26-40 microseconds, and the establishment tests required a few milliseconds in each node.

In the example illustrated in Figures 3(a), 4(c), 4(d), and 5, the destination host receives the information summarized in Figure 6. Two very simple algorithms to compute the bounds for channel 6 packets in each node and their results are shown in the same figure. Channel 6's delay bound requirements were $D = 30$ and $Z = 0.98$. (Note that D is assumed to be completely assignable; that is, such fixed and known delays as the propagation time are not included in the value of D; this means that the actual delay bound specified by the client for channel 6 is higher than D, unless propagation delays are negligible.) Since it is possible to satisfy the requirements, the channel establishment request is accepted.

This concludes the description of our establishment algorithm. The algorithm can be improved in several ways: for example, a method, even an approximate one, which made the calculation of P_{do} faster while retaining the correctness of the whole approach would make channel establishment faster even in those cases where performing the statistical test requires the enumeration of overflow combinations.

5. Conclusion

The goal of this study was to determine the feasibility of offering real-time services in packet-switching wide-area networks. We have defined the problem and the type of network to be studied, adopted the parametrized message channel abstraction, selected several 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 have argued for connection-oriented packet switching as the basis of a real-time service, as it seems very difficult or impossible to build channels on top of a connectionless service. We have also claimed that connections (and performance guarantees) must be provided at the network layer in order to make those guarantees available at the higher levels.

A single-round-trip 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. Performance can be guaranteed because of the scheduling and distributed flow control policies adopted as well as because of the worst-case bounding arguments our establishment scheme is based on.

Many problems remain to be explored in the area of real-time communication in packet-switching networks. Among them are the following:

- the invention of channel establishment schemes for networks whose switches are not accurately described by the model presented in Section 2, for example because of their architecture or scheduling policy;
- the extension of establishment schemes to real-time channels spanning several networks within an internetwork;

- 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 not independent of the traffic on other channels;
- the introduction of security, fault tolerance, accounting, and charging capabilities into the design of a real-time service;
- an investigation of the alternative ways of providing performance guarantees in a wide-area network, and their advantages and disadvantages with respect to the approach discussed in this paper;
- the feasibility of implementing real-time services on an Ethernet, or in a virtual-circuit network consisting of Datakit† [Fras83] or Datakit-like nodes;
- the implementability in hardware of the node functions required for real-time services.

Acknowledgements

A large number of individuals have contributed to the ideas introduced in this paper and to their presentation. David Anderson proposed the channel abstraction, which is the basis of the approach to real-time communication described above. Many of the issues and solutions discussed in the paper resulted from conversations with Dinesh Verma and Kang Shin. Dinesh Verma revealed by his simulation experiments a number of interesting and unexpected phenomena that had to be taken into account in the design of the scheme. Parts of the manuscript were read by Ramon Caceres, Caryl Carr, Sandy Fraser, Riccardo Gusella, Sam Morgan, Dave Presotto, Keshav Srinivasan, Kang Shin, and Dinesh Verma, who provided very helpful comments. Peter Danzig's many constructive suggestions improved very substantially the presentation. The errors and omissions that, in spite of all the help he received, can still be found in the paper are an exclusive intellectual property of the author.

---

[3] † Datakit is a trademark of AT&T Bell Laboratories.

References

[Ande88] D. P. Anderson, "A Software Architecture for Network Communication," *Proc. 8th International Conf. Distributed Computing Systems*, San Jose, CA (June 1988), 376-383.

[Chen88] T. M. Chen and D. G. Messerschmitt, "Integrated Voice/Data Switching," *IEEE Communications Magazine 26*, 6 (June 1988), 16-26.

[Come88] D. E. Comer and R. Yavatkar, "FLOWS: Performance Guarantees in Best Effort Delivery Systems," Rept. No. CSD-TR-791, Computer Science Department, Purdue University, July 1988.

[Dyke88] D. Dykeman and W. Bux, "Analysis and Tuning of the FDDI Media Access Control Protocol," *IEEE J. on Selected Areas in Communications SAC-6*, 6 (July 1988), 997-1010.

[Ferr90] D. Ferrari and D. C. Verma, "A Scheme for Real-Time Establishment in Wide-Area Networks," *IEEE J. Selected Areas Communications 8*, 3 (April 1990), 368-379.

[Fras83] A. G. Fraser, "Towards a Universal Data Trasport System," *IEEE J. Selected Areas Communications SAC-1*, 5 (Nov.1983), 803-816.

[Harr80] E. Harrington, "Voice/Data Integration Using Circuit-Switched Networks," *IEEE Trans. Communications COM-28*, 6 (June 1980), 781-793.

[Kuro84] J. F. Kurose, M. Schwartz and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communication," *ACM Comp. Surveys 16*, 1 (March 1984), 43-70.

[Kuro88] J. F. Kurose, M. Schwartz, and Y. Yemini, "Controlling Window Protocols for Time-Constrained Communication in Multiple Access Networks," *IEEE Trans. Communications COM-36*, 1 (Jan. 1988), 41-49.

[Ross86] F. E. Ross, "FDDI – A Tutorial," *IEEE Communications Magazine 24*, 5 (May 1986), 10-17.

[SAE86] "SAE Draft High Speed Ring Bus (HSRB) Standard", October 27, 1986.

[Sevc87] K. C. Sevcik and M. J. Johnson, "Cycle Time Properties of the FDDI Token Ring Protocol," *IEEE Trans. Software Engineering SE-13*, 3 (March 1987), 376-385.

[Stro89] J. K. Strosnider and T. E. Marchok, "Responsive, Deterministic IEEE 802.5 Token Ring Scheduling," *Real-Time Systems 1*, 1 (Sept. 1989), 133-158.

[Sumi88] S. Sumita and T. Ozawa, "Achievability of Performance Objectives in ATM Switching Nodes," *Proc. Int'l Seminar on Performance of Distributed and Parallel Systems*, T. Hasegawa, H. Takagi, and Y. Takahashi, Eds., Kyoto, Japan (Dec. 1988), 45-56.

[Zhan90] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *Proc. SIGCOMM '90 Symp.*, Philadelphia, PA (Sept. 1990), 19-29.

[Zhao87] W. Zhao and K. Ramamritham, "Virtual time CSMA Protocols for Hard Real-Time Communication," *IEEE Trans. Software Engineering SE-13*, 9 (Aug. 1987), 938-952.

[Zhao88] W. Zhao, J. A. Stankovic, and K. Ramamritham, "A Multi-Access Window Protocol for Transmission of Time Constrained Messages," *Proc. 8th International Conf. Distributed Computing Systems*, San Jose, CA (June 1988), 384-392.

Appendix

Theorem 1: Let the following inequalities be satisfied in a non-saturated node traversed by m deterministic or statistical channels:

$$x_{min,l} \geq \sum_{j=1}^{m} t_j, \quad (l=1,...m), \tag{A.1}$$

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 a in the order in which they would be shipped by the scheduling algorithm described in Section 3 and Figure 1 if they all arrived at the same time at the node. Then, scheduler saturation is impossible if and only if

$$d_i \geq \sum_{j=1}^{i} t_j + T, \quad (i=1,...a). \tag{A.2}$$

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 $\sum t = \sum_{j=1}^{m} t_j$. 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 a arrive at time 0+. Then, the packet on channel i will be completely shipped only at time $\sum t + T$, which is greater than its deadline d_i.

(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 $\sum_{j=1}^{i} t_j < d_i$. If it arrives at 0+, the latest departure time is $\sum_{j=1}^{i} t_j + T \leq d_i$. With an arrival time between 0+ and T, the maximum time spent by the packet in the node is less than d_i. Arriving at $T+$, this maximum time is $\sum_{j=1}^{i} t_j < d_i$, 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 $\sum t$, we calculate the new value of this sum, $(\sum t)'$, from the following recursive definition:

$$(\sum t)'_0 = \sum t,$$

$$(\sum t)'_{h+1} = (\sum t)'_h + Y_k t_k, \quad (h=0,1,...), \tag{A.3}$$

with

$$Y_k = 1 \text{ if } x_{min,k} + d_k < (\sum t)', \qquad \text{(A.4)}$$

$$Y_k = 0 \text{ otherwise.}$$

Sets A and B (for their definition see Section 4) are determined by the value of $(\sum t)'$. 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.

AT ARRIVAL OF PACKET *A*:
select appropriate Q
insert (A,Q)
return

insert $(G,\ Q)$
find correct place of G in Q
  according to EDD policy
$F \leftarrow$ left packet
$H \leftarrow$ right packet
place G between F and H
if $Q \neq QD$ **then return**
align $(G,\ H)$
align $(F,\ G)$
if $F_m = 1$ **then**
    $F_m \leftarrow 0$
    insert $(F,\ QD)$
return

align $(B,\ C)$
if $C_b \leq B_e \leq C_e$ **then**
    $B_e \leftarrow C_b$
    $B_b \leftarrow B_e - B_t$
else if $C_b \leq B_b \leq C_e$ **then**
    $C_e \leftarrow B_b$
    $C_b \leftarrow C_e - C_t$
    $C_m \leftarrow 1$
return

(c)

AT COMPLETION OF A PACKET'S PROCESSING:
$D \mid S \mid B \mid E \leftarrow head\,(QD \mid QS \mid QB \mid QE)$
if $D \mid S \mid B = nil$ **then**
    $D_b \mid S_b \mid B_b \leftarrow \infty$
    $D_e \mid S_e \mid B_e \leftarrow \infty$
if $D_b \mid S_b \mid B_b < t_{current}$ **then**
    $D_b \mid S_b \mid B_b \leftarrow t_{current}$,
    $D_e \mid S_e \mid B_e \leftarrow (D_b{+}D_t) \mid (S_b{+}S_t) \mid (B_b{+}B_t)$
if $D_b < S_e$ **then**
    send D
    return
else if $S_b < B_e$ **then**
    send S
    return
else if $B \neq nil$ **then**
    send B
    return
else if $E \neq nil$ **then** start E until arrival
  of next real-time packet
return

(d)

Figure 1. The scheduling policy for hosts and nodes. The four queues in (a) are for deterministic (QD), statistical (QS), and best-effort (QB) packets, and for everything else (QE). F, G, H, in (b) are deterministic packets; their end times F_e, G_e, H_e coincide with those called *dl* in Figure 2; their beginning times are denoted with the subscript *b*, and their service times in the node with the subscript *t*. In (c), a packet's name with the subscript *m* represents a Boolean variable that, if 1, indicates that the packet has been moved leftward on the time axis. The notation **statement**($D \mid S \mid B \mid E$) in (d) is a shorthand for: **statement** (D); **statement** (S); **statement** (B); **statement** (E). Note that (a) and (b) are missing in this version of the paper.

/*packet arrives on channel at node*/
$t_2 \leftarrow$ clocktime
/*control average interarrival time*/
if $t_2 - t_0 \geq I$ **then**
    /*more than I time units since
    start of current interval***/**
    if $g=1$ **then** $c \leftarrow 0$ **else** $g \leftarrow 1$
    $t_0 \leftarrow t_2$
/*count packets***/**
$c \leftarrow c+1$
if $c > I/x_{ave}$ **then**
    /*too many packets arrived
    during last I time units***/**
    $dll \leftarrow t_0 + I$
    $dd \leftarrow dll - t_2 + d$
    $c \leftarrow 1$
    $g \leftarrow 0$
/*control minimum interarrival time***/**
if $t_2 - t_1 < x_{min}$ **then** $t_1 \leftarrow t_1 + x_{min}$ **else** $t_1 \leftarrow t_2$
/*set packet deadline***/**
$dl \leftarrow t_1 + d$
if $dl \leq dll$ **then** $dl \leftarrow t_1 + dd$
$dll \leftarrow dl$
return
==

t_0= starting time of channel's current I interval
t_1= arrival time of previous packet on channel;
  initially $-inf$.
c= packet counter for channel (initially 0)
d= delay bound for channel's packets in node
dl= packet deadline (time it must have left node)
x_{min}, x_{ave}, I= input characteristics of channel
g=termination cause indicator (0: excess packets
  in I; 1: I interval expiration); initially: $g=1$.
dll= auxiliary variable; initially: $dll=0$.
dd= extended deadline when excess packets in I;
  initially: $dd=0$.

(a)

t_2	*0*	*8*	*11*	*15*	*21*	*27*	*38*
t_1	0	8	12	16	21	27	38
t_0	0	0	0	0	0	27	27
c	1	2	3	1	2	3	1
dl	*6*	*14*	*18*	*31*	*36*	*42*	*57*
dd	0	0	0	15	15	15	19
g	1	1	1	0	0	1	0

(b)

Figure 2. The distributed flow control algorithm, to be executed before the *at arrival of packet A* algorithm in Figure 1(c): (a) pseudocode; (b) an example with $x_{\min} = 4$, $x_{ave} = 8$, $I = 24$, $d = 6$, giving the values of the variables in (a) at the time **return** is executed after each packet arrival (note how the packet deadlines *dl* are extended by the algorithm for $t_2 \geq 11$).

Channel #	*Type*	x_{min}	x_{ave}	*t*	*d*	*z*	t/x_{min}	*p*
1	Det	10	35	3	14	-	0.3	0.2857
2	Stat	4	30	1	6	0.99	0.25	0.1333
3	Stat	15	90	2	8	0.982	0.1333	0.1667
4	Det	8	70	2	8	-	0.25	0.1143
5	Det	15	67.5	3	?	-	0.2	0.2222
6	Stat	8	70	2	?	?	0.25	0.1143

(a)

1	2	3	4	5	6	$\sum$	*Prob*	P_{do}
+	+	+	+	+		1.1333	0.000161	0.000161
+	+	+	+	+	+	1.3833	0.000018	
∅	+	+	+	+	+	1.0833	0.000045	
+	∅	+	+	+	+	1.1333	0.000119	
+	+	∅	+	+	+	1.25	0.000091	
+	+	+	∅	+	+	1.1333	0.000142	
+	+	+	+	∅	+	1.1833	0.0064	
+	+	+	+	+	∅	1.1333	0.000142	
+	+	∅	+	∅	+	1.05	0.000322	0.000943

(b)

Figure 3. *Examples of statistical test in a node with channels 1, 2, 3, 4 for the establishment of channel 5, and in the same node with channels 1, 2, 3, 4, 5 for the establishment of channel 6: (a) channel characteristics; (b) overflow combinations and calculation of* P_{do} *(note that each row in the table corresponds to an overflow combination; the channels present in that combination are indicated by +, while ∅ denotes a channel established in the node but not present in the combination).*

Channel #	t_i	d_i	r_i
2	1	6	4
----	----	----	----
3	2	8	
4	2	8	
1	3	14	

(a)

Channel #	t_i	d_i	r_i
2	1	6	4
3	2	8	6
4	2	8	8
----	----	----	----
1	3	14	
----	----	----	----
5	3	?	

(b)

Channel #	t_i	d_i	r_i
2	1	6	4
3	2	8	6
4	2	8	8
----	----	----	----
1	3	14	
5	3	16	
----	----	----	----
6	2	?	

(c)

Figure 4. Tables for the delay bound test and the delay bound computation: (a) all the original channels (1 through 4 in Figure 3(a)); (b) addition of channel 5; (c) addition of channel 6. Note that the channel # in the tables is the same as in Figure 3(a); the channels in set A in each table are ordered as explained in the description of the delay bound test.

Channel #	*Type*	x_{min}	x_{ave}	t	d	z	t/x_{min}	p
7	Det	15	50	4	12	-	0.2667	0.3
8	Stat	18	80	6	15	0.993	0.3333	0.22

(a)

9	Det	15	50	4	16	-	0.2667	0.3
10	Det	10	70	2	14	-	0.2	0.1429
11	Stat	10	60	3	20	0.985	0.3	0.1667

(b)

Channel #	t_i	d_i	r_i
7	4	12	
8	6	15	
----	----	----	----
6	2	?	8

(c)

Channel #	t_i	d_i	r_i
10	2	14	
9	4	16	
11	3	20	
----	----	----	----
6	2	?	6

(d)

Figure 5. Characteristics and delay bound tests for the two additional nodes traversed by channel 6 (see Figure 3(a)) to reach the destination host: (a) node 2; (b) node 3; (c) table for node 2; (d) table for node 3.

Node	min *d*	$1-P_{do}$	*d*	*z*
1	10	0.99905	12	0.9929
2	8	1	10	0.9939
3	6	0.99918	8	0.9931

$$D - \sum \min d = 6;\ d_i = (D - \sum \min d)/n + (min\ d)_i$$

$$g = Z/\prod(1-P_{do,k}) = 0.9817;\ z_i = (1-P_{do,i})\ g{*}exp(1/n)$$

Figure 6. Information and algorithms for the destination host. The results produced by the algorithms for the establishment of channel 6 are displayed in the *d* and *z* columns. n is the number of nodes traversed by the channel being established (in the example, $n = 3$). To compute the delay bound d_i in node i, the algorithm divides the difference between the global delay bound D and the sum of the minimum delay bounds by the number of nodes traversed by the channel, and adds it to each node's minimum delay bound. The probability bound z_i is computed by multiplying the n-th root of the ratio between the global bound Z and the product of the $(1 - P_{do,k})$'s by $1 - P_{do,i}$.