Guaranteeing Performance for Real-Time Communication in 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 wide-area network should be investigated, and describes a possible approach. We present a model of the network to be studied, and discuss its generality, as well as the presumable limits to its validity in the future. We also give a careful formulation of the problem, including a precise definition of the guarantees to be provided and a provably correct 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 Corp., 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 in the need for real-time communication, i.e., for computer communication with guaranteed performance, over the next several years. Performance guarantees are required not only in distributed process-control or 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 in a relatively short time after that, 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 talkers 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), a performance index even more important than the actual delay is its variance or jitter, which should be as small as possible. Once a network or internetwork offers 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 guaranteed delay, FDDI using the timed token scheme [Ross86], and HSRB the reservation/priority mechanism [SAE86]. An approach that can be used in wide-area networks as well as in LANs is circuit switching: dedicating a circuit of fixed bandwidth to each connection for the entire duration of the connection provides bounded delay. 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.

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 node function 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). 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 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 virtual circuits 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 reported on here, we have tried to reduce the call setup time by designing as fast an establishment scheme as possible.

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 virtual circuit service (as well as a datagram service for those non-real-time communications where this service is preferable) at the network layer.

To formulate the problem in more detail, we will refer to the *parametrized message channel* (or simply *channel*) abstraction introduced in the design of the communication system of DASH [Ande88b]. DASH is a distributed operating system kernel being developed at the University of California at Berkeley for the very large distributed systems of the future, in which real-time communication requirements are expected to be important and widespread [Ande88a]. A channel (called *real-time message stream* or *RMS* in [Ande88b]) is a simplex connection between a sender and a receiver, which delivers messages (or packets) in sequence and is characterized by a number of 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.

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 (to be denoted in the sequel by s_{max});
- the delay bound or bounds for the channel's packets;
- the maximum packet loss rate.

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 [Ande88b], 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 greater 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.

The channel abstraction will be offered in the DASH network architecture by each layer to the layer above it, starting with the network layer. We are primarily concerned, in this paper, with the feasibility of establishing network-level channels that are "correct", i.e., that provide the required guarantees. Higher-level channels will certainly be implementable if we succeed.

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 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, with respect to the natural frequencies and to the 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, and x_{ave}, the minimum value of the mean packet interarrival time, over an 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.

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, as mentioned above, deadline-based. More precisely, the policy we adopt is a modification of EDD (Earliest Due Date) that gives, in the case of a conflict, priority to deterministic over statistical channels, and to statistical over best-effort channels. All of the other tasks of a host or node, including sending, forwarding, and receiving datagrams, have an even lower priority and are preemptable by real-time packets. Our scheduling policy is summarized in Figure 1. A detailed study of its properties and performance is outside the scope of this paper, and will be the subject of a future publication.

4. An Approach to the Solution

Our channel establishment mechanism for the type of network described in Section 2 may be built on top of any procedure that can be used to set up virtual circuits. 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 virtual circuit, the mechanism will perform several tests and tentatively reserve resources in each node visited by the establishment message. In order to make channel establishment fast, we impose on our procedure the restriction that it require only one round trip. This means that the last point along the path where the acceptance/rejection decision for a channel request can be made is the destination host. When a node is visited by an establishment message during this message's return trip, the resources previously reserved there must be either committed or released, and a final, irreversible decision must therefore have already been made.

The tests to be done in each node are concerned with the availability of sufficient bandwidth in the links, and processing power as well as 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 not all of the tests are successful in 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, it frees all the resources that were tentatively reserved there during its forward trip.

If all the tests are successful in all nodes as well as in the destination host, this host properly divides the delay bound (and the probability of not exceeding the bound if the channel being established is statistical) among the nodes on the virtual circuit, after subtracting the total delays in the links to be traversed by the packets. Satisfying in each node the delay bound d (or d, z) assigned to the channel for that node is a sufficient but not necessary condition for satisfying the overall delay bound. For simplicity, we assume this as the goal to be met by the establishment scheme in each node.

A reply message is then sent back to the source host along the same 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, delay bounds are assigned to nodes by the destination host 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 the loading situation of each node.

Subdivision of the total delay bound by the destination host raises a problem, however: during the establishment message's forward trip, the delay bounds to be assigned to each packet of the new channel in each node are not known, and therefore cannot be used in any of the tests to be performed in that node. We get around this difficulty by defining in each node a parameter, the *minimum node delay* d_{min}, which is a lower bound for all the delay bounds to be assigned to channels going through that node. The introduction of d_{min} allows our distributed establishment algorithm to be executed in only one round trip, and has many important ramifications, as will be seen in the sequel. Selecting an appropriate value for d_{min} in each node is important; we plan to study

this problem in detail by simulation.

To simplify our discussion, we assume in the rest of this paper that buffer space available for real-time connections in the network's hosts and nodes is unlimited, and that the network's error rate is always lower than the acceptable loss rates. 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.

Besides those having to do with the availability of sufficient storage space for buffering packets, with which we are not concerned here, the following tests are to be performed in each node by the local establishment algorithm:

(a) the sum of the minimum node delays for the nodes already traversed (including the current one) is less than the channel's delay bound;

(b) the processing power still available in the node to handle channels is greater than the one required by the channel to be established;

(c) the actual delay of a packet traveling on each channel will satisfy the performance requirements for that channel in that node.

Test (b) could be performed taking into account the burstiness of the packet streams on the various channels intersecting each other in the node. To be sure that the node will not under any circumstances run out of processing power, we plan to adopt (here as at several other points in our establishment procedure) the worst-case approach, and test whether

$$B - \sum_i \frac{s_{max,i}}{x_{min,i}} \geq \frac{s_{max}}{x_{min}}, \qquad (1)$$

where

- B is the *node's bandwidth*, defined as the maximum number of real-time bytes/s the node can switch (note that B may be limited by input or output link bandwidth, or by the processing power of the node's CPUs or network interfaces);
- $s_{max,i}$ and $x_{min,i}$ are parameters of the i-th channel passing through the node; and
- s_{max} and x_{min} are parameters of the channel to be established.

If (1) is not satisfied, then the channel establishment request is rejected by the node.

The bandwidth of a node may be expected to be, in future networks, either the bottleneck or likely to become one. Simulation studies now being carried out will reveal whether the worst-case approach just described makes test (b) by far the predominant cause of establishment request rejection, under reasonable assumptions about future network technology and applications. If so, test (b) would indeed be likely to become the establishment bottleneck, i.e., the limiting factor for the number of established channels, and investigating the adoption of an average-case alternative to (1) would become highly desirable.

Test (c), which is not performed for best effort channels (while (a) and (b) are), is by far the most difficult one to translate into an algorithm. We have to verify that *none* of the packet to be switched by the node will fail to satisfy the delay bound for its channel in that node even after the addition of the new deterministic or statistical channel. This problem becomes much simpler if we assume that the satisfaction by the channel to be established of *its* delay bound (which, because of our ignorance of its value, we assume equal to d_{min}) implies the satisfaction by all other channels of *their* respective bounds. This assumption seems reasonable, since by definition d_{min} is lower than any delay bound in the node, but is not necessarily always valid. We discuss its validity, and ways in which we can correct for the consequences of its lack of validity, in Appendix 1.

Verification of compliance with delay bounds for a statistical channel in a node would require knowledge not only of the channel's d, but also of the value of z in the node. Rather than select a node-wide value of z, our approach will consist of

(i) computing an upper bound for the probability P_{do} (the *probability of deadline overflow*) that delays on a channel will go beyond d_{min},

(ii) checking that $\bar{P}_{do}$ (the upper bound of P_{do}) is lower than or equal to the z of each established channel in that node (i.e., $\bar{P}_{do} \leq \min(z)$), and

(iii) transmitting to the destination host that upper bound along with those of all the other nodes traversed by the channel.

Tests (i) and (iii) are to be performed whenever the channel to be established is statistical, test (ii) whenever at least one statistical channel is already traversing the node.

If test (ii) is not satisfied by all channels passing through the node, the node rejects the request. 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_j (j = 1, 2, ...n), each one of which is greater than or equal to $1 - \bar{P}_{do,j}$:

$$Z = \prod_j z_j, \tag{2}$$

$$z_j \geq 1 - \bar{P}_{do,j}, (j=1,2,...n), \tag{3}$$

where n is the number of machines (nodes plus hosts) along the channel's path. The question about whether Z can be factored as just described can be simply answered by the destination host by checking whether

$$\prod_j (1 - \bar{P}_{do,j}) \leq Z. \tag{4}$$

If (4) does not hold, then the inequalities (3) cannot all be satisfied, and the request is rejected. Note that the assumptions we have implicitly made by writing (2), i.e., that, once a packet is delayed beyond its deadline in a node, it will not satisfy the channel's overall delay bound, and that each node will delay, if any, previously undelayed packets, are really pessimistic. Thus, rejecting a request if (4) 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 will tend to reduce the maximum number of channels of given characteristics 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 (1) how large this reduction is in practice going to be, and (2) whether a less conservative approach could offer similar performance guarantees. Question (1) will be answered by the ongoing simulation effort; question (2) will be the subject of a future investigation.

Before describing a way to compute $\bar{P}_{do}$, we have to choose a suitable flow control mechanism. One possible approach, to be carefully investigated also from the viewpoint of its execution costs, is to increase the deadlines of the "offending" packets, 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, in our calculation of $\bar{P}_{do}$, that each channel going through the node satisfies

those constraints.

We can now proceed to calculate $\bar{P}_{do}$. As already mentioned above, we say that a packet traveling over the channel to be established will suffer from *deadline overflow* in a node if it will spend in that node a time longer than d_{min}. If the channel in question is a deterministic one, an overflow may occur if the total node service time of deterministic packets whose deadlines fall within the interval of duration d_{min} following the arrival of the packet is greater than $d_{min}-t_s$, where t_s is the maximum service time required in the node by the packet under consideration. The probability of deadline overflow for a deterministic channel must be zero even in the worst case, i.e., in the unlikely case in which *all* deterministic channels have packets with deadlines in the d_{min} interval. Thus, test (c) in the case of the establishment of a deterministic channel includes adding the maximum service times of all the deterministic channels going through the node (including the one to be established), and seeing whether the total is lower than d_{min}: if it is, then the establishment request can (as far as this test and this node are concerned) be accepted; if not, it must be rejected.

If the channel is statistical (or if the channel is deterministic but there are statistical channels going through the node), the deadline-overflow probability (or an upper bound for it) is to be calculated. This probability can be computed as the sum of the probabilities of all the possible situations in which the packet's deadline might not be met. These are the situations in which the total service time of the deterministic *and* statistical packets (including the one being considered) that have their deadlines in the d_{min} interval is greater than d_{min}.

Once we have identified all of these possible situations (one way to do this, though not the most efficient, is by enumeration), their respective probabilities must be evaluated. Assuming that the deadlines of packets on different channels at the node are statistically independent (an assumption whose validity is discussed in Appendix 2), the probability of a situation in which packets from channels, say, A, D, and E are *present*, i.e., have their deadlines within d_{min}, and channels B and C are *absent* is

$$Prob(A, \neg B, \neg C, D, E) = Prob(A)\,(1-Prob(B))\,(1-Prob(C))\,Prob(D)\,Prob(E). \tag{5}$$

$Prob(Y)$ is the probability that channel Y will have at least one packet whose deadline falls within the interval of duration d_{min} being considered. Note that, if $x_{min}<d_{min}$ for a given channel, that channel may have more than one packet within the interval; in general, the maximum number of such packets will be $\left\lceil d_{min}/x_{min} \right\rceil$. Note also that, to take these cases into account, the general expression of the total service time for a given situation is

$$\sum_{k=1}^{n} a_k t_{s,k} \left\lceil \frac{d_{min}}{x_{min}} \right\rceil, \tag{6}$$

where a_k=1 if one or more packets from channel k are present, and a_k=0 otherwise.

How can we compute $Prob(Y)$? If the interarrival time distribution of the packets on each channel were known, and had distribution function $P(x)$ and density function $p(x)$, we would have

$$Prob(Y) = \frac{1}{x_{ave}} \left[\int_0^{d_{min}} x\,p(x)dx + (1-P(d_{min}))d_{min} \right]. \tag{7}$$

Figure 3(a) presents a possible probability density function for x. Unfortunately, the distribution of x is not known; the only two descriptors of it that are given by the client are x_{min} and x_{ave} (in general, not even x_{max} is known, and the given x_{ave} is not the mean interarrival time but a lower bound for it). If we plot, as in Figure 3(b), $Prob(Y)$ vs. d_{min}, we see that, between 0 and x_{min}, $Prob(Y)$ grows linearly with d_{min} following the equation

$$Prob(Y) = \frac{d_{min}}{x_{ave}}. \tag{8}$$

For $d_{min} > x_{max}$, $Prob(Y)=1$. If $x_{max}=\infty$, then $Prob(Y)$ approaches 1 asymptotically as d_{min} grows.

An interesting property of (7) is that the curves in the $Prob(Y)$ vs. d_{min} plane corresponding to the different distributions of x are never above d_{min}/x_{ave} for $d_{min} \leq x_{ave}$, and never above 1 for $d_{min} > x_{ave}$ (see Appendix 3). Thus, an upper bound for $Prob(Y)$ is

$$\overline{Prob(Y)} = min(\frac{d_{min}}{x_{ave}}, 1). \tag{9}$$

If packets from channel Y were absent in the situation whose probability is to be assessed, an upper bound of $1-Prob(Y)$ could be obtained from knowledge of a lower bound of $Prob(Y)$. If x_{ave} were the true average interarrival time, such a lower bound could be calculated as

$$\underline{Prob(Y)} = min(\frac{d_{min}}{x_{ave}}, \frac{x_{min}}{x_{ave}}). \tag{10}$$

However, the given x_{ave} is itself a lower bound for the true average. If this true average (or an upper bound for it) is unknown, we can set $\underline{Prob(Y)}=0$ in the calculation of the probability of any situation in which packets from channel Y are absent. The crudeness of these lower bounds is expected not to have a major negative influence on the algorithm's performance, i.e., it should not appreciably reduce the number of channels a given network can support. In Figure 4, we show an example of the application of our establishment scheme in a node. Also shown are the ranges of values $\bar{P}_{do}$ can take in that node for this example. These ranges are relatively small when the total number of absent channels in the overflow situations being considered is small, and when the probabilities of packets being present on the other channels are low. However, there is uncertainty about what to do only if the threshold probability ($\min(z)$) falls within the range of values for $1 - \bar{P}_{do}$; in this case, the conservative approach calls for rejection, while a less conservative course of action might be justified if, for instance, the distance between $\min(z)$ and the upper bound of the range is small. The simulation study we are carrying out is expected to provide useful suggestions about this issue.

This concludes the description of our basic establishment algorithm. The algorithm can be improved in several ways: for example, a method, even an approximate one, which made the calculation of $\bar{P}_{do}$ much faster while provably retaining the correctness of the whole approach would be extremely useful, since channel establishment should be as fast as possible. Some improvements (including such an approximation) have already been devised. Together with a simulation-based evaluation of the establishment procedure described above, they will be discussed in a forthcoming paper.

5. Conclusion

The main goal of this study was to determine the feasibility of offering real-time services in a packet-switching wide-area network. We have precisely defined the problem and the type of network to be studied, adopted the parametrized message channel as the basic abstraction, chosen the types of performance guarantees such a service could provide, carefully specified the meaning and extent of these guarantees, and selected a simple characterization of a channel's input packet stream. We have argued for a virtual-circuit approach to packet switching as the basis of a real-time service, as it seems very difficult or impossible to build channels on top of a datagram service. We have also claimed that virtual circuits (and performance guarantees) must be provided at the lowest level in the protocol hierarchy (i.e., at the network layer) in order for those guarantees to be 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. These tests are such that a channel that passes them 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 of the worst-case bounding arguments our establishment scheme is based on. Thus, our procedure is provably correct; in other words, it allows us to offer true real-time performance guarantees.

Many problems remain to be explored in the area of wide-area real-time communication. Among them are:

- the buffer allocation and management policies suitable for a real-time service in the type of network dealt with in this paper;
- the best ways to guarantee a given bound on delay variance or jitter;
- the possibility of devising provably correct algorithms for the establishment of channels whose traffic is described by parameters different from x_{min} and x_{ave};
- 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;
- the alternative ways of providing performance guarantees in a wide-area network, e.g., using TDM, and their advantages and disadvantages with respect to the approach discussed in this paper;
- the feasibility of implementing real-time services 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.

Most of these questions are now being explored, and will be discussed in a number of forthcoming papers.

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 proposed the concept of minimum delay bound (d_{min}), and 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. The members of the DASH Project contributed feedback and support throughout the effort. 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.

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

References

[Ande88a] D. P. Anderson and D. Ferrari, "An Overview of the DASH Project, Rept. No. UCB/CSD 88/406, University of California, Berkeley , February 1988.

[Ande88b] D. P. Anderson, "A Software Architecture for Network Communication", *Proc. 8th International Conf. on 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.

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

[Harr80] E. Harrington, "Voice/Data Integration Using Circuit-Switched Networks", *IEEE Trans. on Comm. 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.

[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. on Software Engineering SE-13*, 3 (March 1987), 376-385.

[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, December 7-9, 1988, 45-56.

Appendix 1

The deadline-overflow probability P_{do} and its upper bound $\bar{P}_{do}$ computed in Section 4 generally decrease as d_{min} grows beyond a certain value, in spite of the monotonically non-decreasing nature of $Prob(Y)$. This is because the number of overflow situations decreases when d_{min} gets larger, and in the limit reduces to zero. However, the decrease of $\bar{P}_{do}$ is generally non-monotonic, but follows a discontinuous and sometimes complicated pattern (see for example Figure A1-1, which shows how $\bar{P}_{do}$ varies with d_{min} when the node referred to in Figure 4 is traversed by channels D1, D2, D3, S1, and S2). The danger of comparing the value of $\bar{P}_{do}$ for d_{min} = 10 ms with min(z) is apparent from Figure A1-1: if we had chosen d_{min} = 11.4 ms, a perfectly legitimate value, we would have rejected the request for the establishment of D3 instead of letting it go through, as we did in Figure 4(b) (however, since the node delay of S1 and S2 is 12 ms, and $\bar{P}_{do}(12)$ is much lower than 0.02, the guarantees we gave will be satisfied even after the establishment of channel D3, as long as D3 is not assigned a maximum delay in the node between 11.2 and 11.4 ms). To avoid this danger, if the value of $\bar{P}_{do}$ in d_{min} is acceptable, the computation of $\bar{P}_{do}$ should be repeated in the highest peak of the $\bar{P}_{do}$ curve to the right of d_{min}. The values of d corresponding to the peaks of $\bar{P}_{do}$ can be found by solving the equation

$$\sum_k \left\lceil \frac{d}{x_{min,k}} \right\rceil t_{s,k} = d, \tag{A1.1}$$

where k is an index enumerating all the possible situations that may occur in the node after the addition of the channel to be established. An effective algorithm that can identify the highest peak to the right of d_{min} is needed to make this verification practically feasible.

An alternative, also to be investigated, would be to determine a value greater than d_{min} beyond which the peaks are guaranteed to be low enough, and use this value instead of d_{min} as the lower bound for d.

Yet another solution would be making $d = d_{min}$ for the channel to be established, though this would not automatically guarantee that the other d's in the node will not correspond to a dangerous peak after the addition of the new channel. Also, the node may be prevented from supporting a larger number of channels if a maximum delay substantially smaller than necessary is assigned to one or more of the channels traversing it.

Appendix 2

The statistical independence assumption for packet arrival processes at a node is much more reasonable for channels coming into the node from different input links than for those sharing the same input link. When we write an expression like (5), we assume that no correlation exists between, say, packets on channel A and packets on channel D, even if A and D share the same input link into that node. More precisely, if channels A and D are independent (e.g., do not transmit correlated packets, such as the frames of a motion picture and the audio packets corresponding to its sound track), the presence of A can only decrease or leave unchanged the probability that D is also present, since arrivals of packets from A and D are serialized by their common input link. Thus,

$$Prob(D \mid A) \le Prob(D), \tag{A2.1}$$

$$Prob(A \mid D) \le Prob(A),$$

where $Prob(D \mid A)$ is the conditional probability that D is present given that A is present.

Hence,

$$Prob(A)\, Prob(D) \ge Prob(D \mid A)\, Prob(A), \tag{A2.2}$$

$$Prob(A)\,Prob(D) \geq Prob(A \mid D)\,Prob(D).$$

The probability of a situation containing A but not D cannot be computed by replacing $Prob(\neg D \mid A)$ with $Prob(\neg D)$, since in general

$$Prob(\neg D \mid A) \geq Prob(\neg D). \tag{A2.3}$$

However, if we adopt the crude approximation $\underline{Prob(D)}=0$, hence $\overline{Prob(\neg D)}=1-\underline{Prob(D)}=1$, then only the equal sign will be satisfiable in (A2.3). Thus, the independence assumption tends to increase the value of $\bar{P}_{do}$ or leave it unchanged, thereby not endangering the guaranteed delay bounds.

Channels that are naturally correlated with each other should either be dealt with as if they were a single channel for the purposes of calculating $\bar{P}_{do}$, or their conditional probabilities will have to be estimated.

Appendix 3

To prove (9), we rewrite (7) as

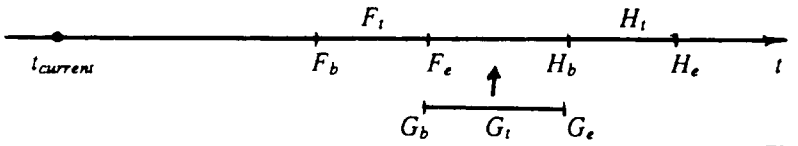
$$Prob(Y) = \frac{d_{min}}{x_{ave}}\left\{\left[\int_0^{d_{min}} \frac{x}{d_{min}} p(x)dx\right] + \int_{d_{min}}^{\infty} p(x)dx\right\}. \tag{A3.1}$$

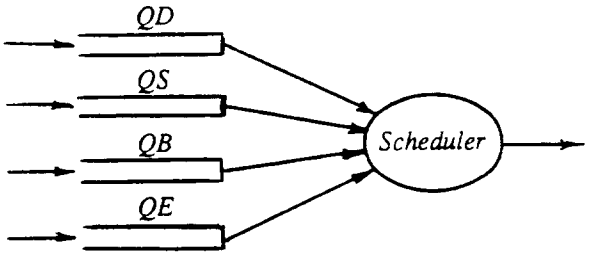
Since

$$\int_0^{d_{min}} \frac{x}{d_{min}} p(x)dx \leq \int_0^{d_{min}} p(x)dx, \tag{A3.2}$$

we have

$$Prob(Y) \leq \frac{d_{min}}{x_{ave}} \int_0^{\infty} p(x)dx = \frac{d_{min}}{x_{ave}}. \quad \text{Q.E.D.} \tag{A3.3}$$





at arrival of packet A:
select appropriate Q
insert (A, Q)
return

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

insert (G, Q)
find correct place of G in Q
$F \leftarrow$ 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

Figure 1. Scheduling policy for hosts and nodes. The four queues are for deterministic (QD), statistical (QS), and best-effort (QB) packets, and for everything else (QE). The notation **statement** $(D \mid S \mid B \mid E)$ is a shorthand for: **statement** (D); **statement** (S); **statement** (B); **statement** (E). F, G, H are deterministic packets; their end times F_e, G_e, H_e coincide with those called *dl* in Figure 2, and their node service times are denoted by F_t, G_t, H_t. Boolean variable F_m, if 1, indicates that F has been moved leftward on the time axis.

```
/*packet arrives on channel at node*/
t2 ← clocktime
/*control average interarrival time*/
if t2−t0≥I then
        if g=1 then c←0 else g←1
        t0←t2
c←c+1
if c>I/xave then
        dl1←t0+I
        dd←dl1−t2+d
        c←0
        g←0
/*control minimum interarrival time*/
if t2−t1<xmin then t1←t1+xmin else t1←t2
/*set packet deadline*/
dl←t1+d
if dl≤dl1 then dl←t1+dd
dl1←dl
return
```

---

t_0= starting time of channel's current I interval
t_1= arrival time of previous packet on channel
c = packet counter for channel (initially 0)
d = maximum delay 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, $dl1$, dd = auxiliary variables

Figure 2. Distributed flow control algorithm. It is executed before the *at arrival of packet A* algorithm in Figure 1.

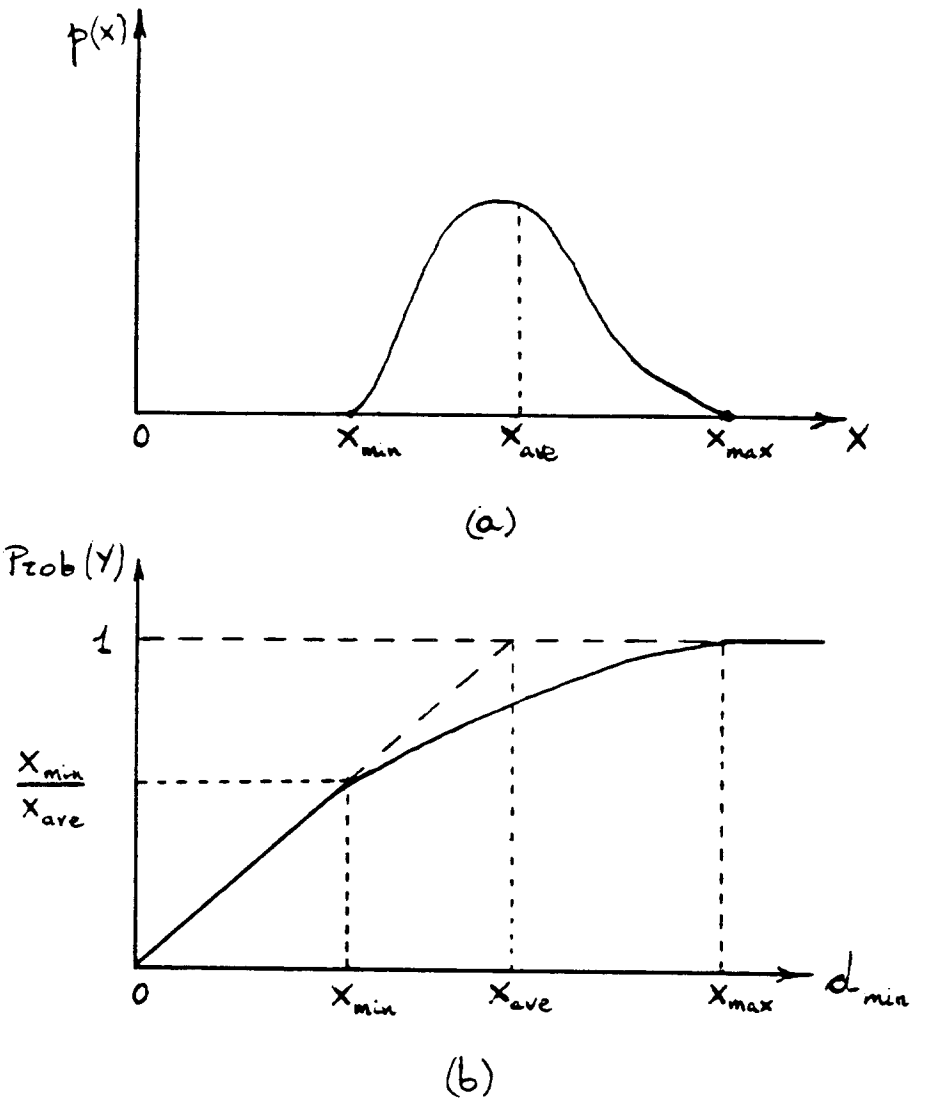


Figure 3. The density function of an interarrival time distribution on channel Y (diagram (a)), and the probability that the deadline of at least one channel Y packet falls within an interval of duration d_{min} (diagram (b)). Note that the given x_{ave} is assumed here to coincide with the mean interarrival time, whereas it usually is a lower bound for the mean.

Parameter	*Unit*	*Interactive voice*	*File transfer*
type	-	D	S
x_{min}	ms	20	1.6
x_{ave}	ms	75	18
I	ms	10K	25K
s_{max}	by	8	8K
t_s	ms	0.5	0.65
d	ms	25	12
z	-	-	0.98
$ceil\,(d_{min}/x_{min})$	-	1	7
d_{min}/x_{ave}	-	0.1333	0.5556

(a)

Establishment request	$\bar{P}_{bo}$	*Node decision*	
S1	-	accept	
D1	-	accept	
S2	-	accept	
D2	0.0055	accept	(i)
S3	0.1865-0.1880	reject	(ii)
D3	0.0150-0.0172	accept	(iii)
D4	0.0273-0.0359	reject	

(b)

	S1	D1	S2	D2	S3	D3	Prob
(i)	+	+	+	+			0.0055
(ii)	+	∅	+	∅	+		0.1715
	+	+	+	+	−		0.0050-0.0055
	−	+	+	+	+		0.0050-0.0055
	+	+	−	+	+		0.0050-0.0055
(iii)	+	+	+	+		+	0.0007
	+	+	+	+		−	0.0048-0.0055
	+	−	+	+		+	0.0048-0.0055
	+	+	+	−		+	0.0048-0.0055

(c)

Figure 4. Application of the channel establishment algorithm to a node with d_{min} = 10 ms. The two types of channels have the characteristics shown in (a). The requests received and the decisions made by the node are reported in (b). An "accept" decision by the node is assumed to be confirmed by the rest of the nodes on the path and by the destination host. The calculation of an upper bound for the deadline-overflow probability at the arrival of a request for the establishment of channel D2 (case (i)), S3 (case (ii)), and D3 (case (iii)) is shown in (c). Each row corresponds to an overflow situation or group of situations. A + indicates the presence of a packet on the corresponding channel, a – indicates absence, and a ∅ is a *don't care* condition. The lower value in a range in the *Prob* column has been obtained by using approximation (10), the higher by setting $\underline{Prob\,(Y)}$=0. The values of $\bar{P}_{bo}$ in (b) can be obtained by adding the values in the *Prob* column and in the appropriate rows.

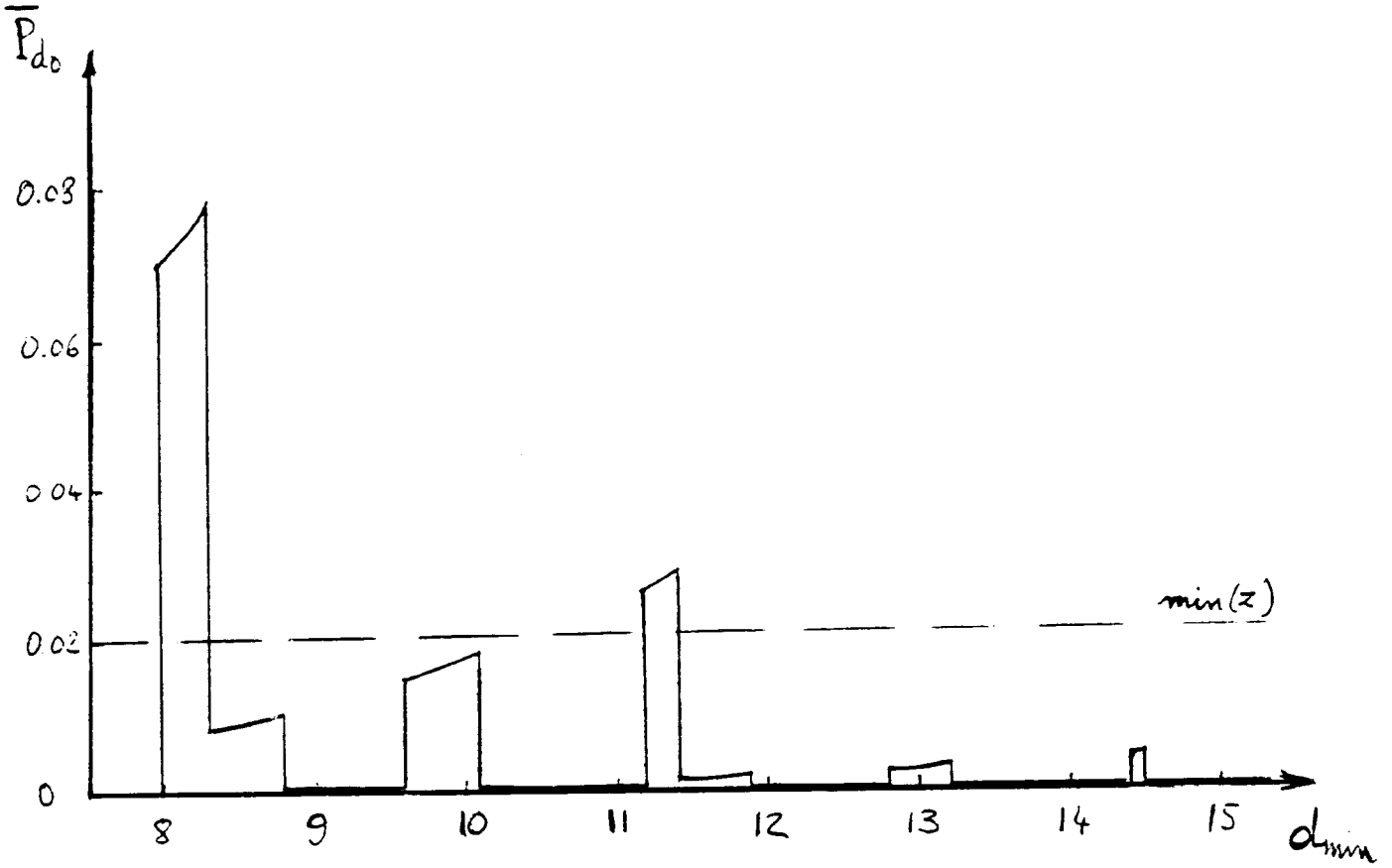


Figure A1-1. $\bar{P}_{d_0}$ vs. d for the example in Figure 4. The values of $\bar{P}_{d_0}$ reported here are based on the approximation $\underline{Prob\,(Y)}$=0.