CLIENT REQUIREMENTS FOR REAL-TIME COMMUNICATION SERVICES

Domenico Ferrari
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
and International Computer Science Institute
Berkeley, California, U.S.A.

Abstract

A real-time communication service provides its clients with the ability to specify their performance requirements and to obtain guarantees about the satisfaction of those requirements. In this paper, we propose a set of performance specifications that seem appropriate for such services; they include various types of delay bounds, throughput bounds, and reliability bounds. We also describe other requirements and desirable properties from a client's viewpoint, and the ways in which each requirement is to be translated to make it suitable for lower levels in the protocol hierarchy. Finally, we present some examples of requirements specification, and discuss some of the possible objections to our approach.

This research has been supported in part by AT&T Bell Laboratories, the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions 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.

1. Introduction

We call *real-time* a computer communication service whose clients are allowed to specify their performance requirements and to obtain guarantees about the fulfillment of those requirements.

Three terms in this definition need further discussion and clarification: *clients, performance*, and *guarantees.*

Network architecture usually consists, at least from a logical viewpoint, of a stack of protocol layers. In the context of such an architecture, the notions of client and server apply to a number of different pairs of entities: every layer (with the support of the underlying layers) provides a service to the layer immediately above it and is a client of its underlying layers. In this paper, our considerations generally apply to any client-server pair. However, most of them particularly refer to human clients (users, programmers) and to the ways in which such clients express their communication and processing needs to the system (e.g., interactive commands, application programs). This type of client is especially important, since client needs at lower layers can be regarded as translations of the needs expressed by human clients at the top of the hierarchy. When the client is human, the server consists of the entire (distributed) system, including the hosts, their operating systems, and the networks interconnecting them.

As for the generic term *performance*, we will give it a fairly broad meaning. It will include not only delay and throughput, the two main network performance indices, but also reliability of message delivery. Real-time communication is concerned with those aspects of *quality of service* that have to do with performance in this broad sense.

The term *guarantee* in this paper has a rather strong legal flavor. When a server guarantees a given level of performance for the communications of a client, it commits itself to providing that performance and to paying appropriate penalties if the actual performance turns out to be insufficient. On the other hand, the client will have to obey certain rules, and will not be entitled to the requested performance guarantees unless those rules are scrupulously obeyed. In other words, client and server have to enter into a *contract* specifying their respective rights and duties, the benefits that will accrue, the conditions under which those benefits will materialize, and the penalties they will incur for not keeping their mutual promises. We believe that a legal viewpoint is to be adopted if serious progress in the delivery of communication services (not only the real-time ones) is desired. Utility services, as well as other kinds of service, are provided under legally binding contracts, and a mature computer communication utility cannot fail to do the same. In the field of real-time communication, such a contract will by definition include performance guarantees.

Real-time services may be offered in any kind of network or internetwork. Some of their predictable applications are:

(a) digital continuous-media (motion video, audio) communication: lower bounds on throughput and upper bounds on delay or delay variability or both are needed to ensure any desired level of output quality; in the interactive case, both the values of delay and delay variabilities have to be bounded; some limited message losses are often tolerable in the cases of video and voice (whenever very high quality is not required), but usually not in the case of sound;

(b) transmission of urgent messages in real-time distributed systems: delay bounds are the important guarantees to be provided in these applications; losses should ideally be impossible;

(c) urgent electronic-mail messages and, more in general, urgent datagrams: again, delay is the obvious index to be bounded in this case, but small probabilities of losses can often be tolerated;

(d) transfers of large files: minimum throughput bounds are usually more important than delay bounds in this application; also, all pieces of a file must be delivered with probability 1;

(e) fast request-reply communication: e.g., data base queries, information retrieval requests, remote procedure calls; this is another case in which delay (more precisely, round-trip delay) is the index of primary interest; reliability requirements are generally not very stringent.

We conjecture that, when networks start offering well-designed and reasonably-priced real-time services, the use of such services will grow beyond the expectations of most observers. This will occur primarily because new performance needs will be induced by the availability of guaranteed-performance options. As the history of transportation and communication has repeatedly shown, faster services bring about major increases of the shipments that are perceived as *urgent*. The phenomenon will be more conspicuous whenever the quality of service provided to non-real-time clients will deteriorate. It is clear from this comment that we assume that real-time services will coexist within the same networks and internetworks with non-real-time communications. Indeed, postulating a world in which the two types of service are segregated rather than integrated would be unrealistic, as it would go against the clear trend towards the eventual integration of all information services. For the same reason, the traffic in the network is assumed to be heterogeneous, i.e., to consist of a variety of types of messages, representing a variety of information media and their combinations, with a wide spectrum of burstiness values (from uncompressed continuous fixed-rate streams to very short and erratic bursts of information).

This paper discusses the client requirements and characteristics of a real-time communication service. Server requirements and design principles will be the subject of a subsequent paper. Section 2 contains some considerations about the ways in which the clients specify their requirements, and those in which a server should reply to requests for real-time services. Performance requirements are presented in Section 3; other properties that clients may need or desire are described in Section 4. Section 5 deals with the problem of translating the requirements of a human client or an application for the equivalent lower-level ones. In Section 6, we briefly present four examples of client requirement specifications, and in Section 7 we discuss some of the objections that can be raised against our approach.

2. Client Requests and Server Replies

No real-time service can be provided if the client does not specify, together with the requirements, the characteristics of the expected input traffic. Describing input traffic and all the various requirements entails much work on the part of a client. Gathering the necessary information and inputting it may be very time-consuming. A well-designed real-time communication service will minimize the effort to be spent by a client.

Sensible default values, the possibility of partial or incremental specifications (e.g., by editing preexisting specifications), and a number of *standard* descriptions should be provided. These descriptions will include characterizations of inputs (e.g., those of a video stream for multimedia conferencing, an HDTV stream, a hi-fi audio stream, a file transfer stream, and so on) and standard sets of requirements. With these aids, it might be possible for a human client to specify his or her request by a short phrase, perhaps followed by a few characters representing options or changes to the standard or default values.

Since requests for real-time services may be denied because of a mismatch between the client's demands and the resources available to the server, the client will appreciate being informed about the reasons for any rejection, so that the request can be modified and resubmitted, or postponed, or cancelled altogether [Herr89]. The information provided by the server to a human client should be meaningful, useful, and non-redundant. The reason for rejection should be understandable by the client (who should be assumed not to know any of the details of the operating system, of the protocols or of the network) and should be accompanied by data that will be useful to the client in deciding what to do as well as how the request ought to be modified to make it successful. If, for example, a bound specified by the client cannot be guaranteed by the server under its current load, the information returned to the client should include the minimum or maximum value of the bound that the server could guarantee; the client will thus be able to decide whether that bound would be acceptable (possibly with some other modifications as well) or not, and act accordingly.

When the client is not a human being but an application or a process, the type of a server's replies should be very different from that just described [Herr89]; another standard interface, the one between an application and a real-time service, must therefore be defined, possibly in multiple, application-specific versions.

Clients will also be interested in the pricing policies implemented by the server: these should be fair (or at least perceived to be fair) and easy to understand. The client should be able easily to estimate charges for given performance guarantees as a function of distance, time of day, and other variables, or to obtain these estimates from the server as a free off-line service.

3. Performance Requirements

A client can specify a service requirement using the general form

$$pred = TRUE,$$

where some of the variables in predicate $pred$ can be controlled or influenced by the server.

A simple and popular form of performance requirement is that involving a *bound*. A *deterministic* bound can be specified as

$$(var \le bound) = TRUE, \text{ or } var \le bound,$$

where variable var is server-controlled, while $bound$ is client-specified. The bounds in these expressions are *upper* bounds; if < is replaced by >, they become *lower* bounds.

When the variable in the latter expression above is a probability, we have a *statistical* bound, and $bound$ in that case is a probability bound; if the predicate is a deterministic bound, we have:

$$Prob(var \le bound) \ge probability\text{–}bound.$$

In this requirement, the variable has an upper bound, and the probability a lower bound. Note that deterministic bounds can be viewed as statistical bounds that are satisfied with probability 1.

A form of bound very similar to the statistical one is the *fractional* bound:

$$C_A(var \le bound) \ge B,$$

where variable var has a value for each message in a stream, and C_A is a function that counts the number of times var satisfies the bound for any A consecutive messages in the stream; this number C_A must satisfy bound B. Obviously, a fractional bound is realizable only if $B \le A$. Fractional bounds will not be explicitly mentioned in the sequel, but they can be used in lieu of statistical bounds, and have over these bounds the avantages of easy verifiability and higher practical interest.

In this section, we restrict our attention to those requirements that are likely to be the most useful to real-time clients.

3.1. Delay requirements

Depending on the application, clients may wish to specify their delay requirements in different ways [Gait90]. The delays involved will usually be those of the application-oriented messages known to the client; for instance, the delay between the beginning of the client-level transmission of a video frame, file, or urgent datagram and the end of the client-level reception of the same frame, file, or urgent datagram [1]. Also, they will be the delays of those messages that are successfully delivered to the destination; the fraction of messages that are not, to which the delay bounds will not apply, will be bounded by reliability specifications. Note that clients will express delay bounds by making implicit reference to their own clocks; the design of a real-time service for a large network will have to consider the impact on bounds enforcement of non-synchronized clocks [Verm90]. Some of the forms in which a delay requirement may be specified are

(i) *deterministic delay bound*:

$$D_i \le D_{\max}, \quad \text{for all } i,$$

where D_i is the delay with which the i-th message sent by the client is delivered to the destination client-level entity,[2] and $D_{\max}$ is the delay upper bound specified by the client;

(ii) *statistical delay bound*:

$$Prob\,(D_i \le D_{\max}) \ge Z_{\min},$$

where D_i and $D_{\max}$ are defined as above, and $Z_{\min}$ is the lower bound of the probability of successful and timely delivery;

(iii) *deterministic delay-jitter bound*:

$$J_i = |\, D_i - D \,| \le J_{\max}, \quad \text{for all } i,$$

where D is the ideal, or target delay, J_i is the delay jitter of the i-th message delivered to the destination, and $J_{\max}$ is the upper jitter bound to be specified by the client together with D; note that an equivalent form of this requirement consists of assigning a deterministic upper bound $D + J_{\max}$ and a deterministic lower bound $D - J_{\max}$ to the delays D_i [Herr90];

(iv) *statistical delay-jitter bound*:

$$Prob\,(J_i \le J_{\max}) \ge U_{\min}, \quad \text{for all } i,$$

---

[1] In those cases, e.g., in some distributed real-time systems, where message deadlines are assigned instead of message delays, we can always compute the latter from knowledge of the former and of the sending times, thereby reducing ourselves again to a delay bound requirement.

[2] In our descriptions we assume, without loss of generality, that the client requesting a real-time service is the sending client, and that the destination (which could be a remote agent of the client or another user) is a third party with respect to the establishment of the particular communication being considered.

where $U_{\min}$ is the lower bound of the probability that J_i be within its limits.

Other forms of delay bound include bounds on average delay, delay variance, and functions of the sequence number of each message, for example, $D_{\max}(i)$ for the deterministic case. There may be applications in which one of these will be the preferred form, but, since we have not found any so far, we believe that the four types of bounds listed as (i)-(iv) above will cover the great majority of the practical cases.

3.2. Throughput requirements

The actual throughput of an information transfer from a source to a destination is bounded above by the rate at which the source sends messages into the system. Throughput may be lower than this rate because of the possibility of unsuccessful delivery or message loss. It is also bounded above by the maximum throughput, which is a function of, among other things, network load. As the source increases its input rate, the actual throughput will grow up to a limit and then stop. Clients concerned with the throughput of their transfers will want to make sure that saturation is never reached, or is reached only with a suitably small probability and for acceptably short intervals. Also, if the bandwidth allocated to a transfer is not constant, but varies dynamically on demand to accommodate, at least to some extent, peak requests, clients will be interested in adding an average throughput requirement, which should include information about the length of the interval over which the average must be computed [Ferr89a].

Thus, reasonable forms for throughput requirements appear to be the following:

(i) *deterministic throughput bound*:

$$\theta_i \geq \theta_{\min}\,, \quad \text{for all } i\ ,$$

where θ_i is the throughput actually provided by the server, and $\theta_{\min}$ is the lower bound of throughput specified by the client, that is, the minimum throughput the server must offer to the client;

(ii) *statistical throughput bound*:

$$Prob\,(\theta_i \geq \theta_{\min}) \geq \zeta_{\min}\ ,$$

where θ_i and $\theta_{\min}$ are defined as above, and $\zeta_{\min}$ is the lower bound of the probability that the server will provide a throughput greater than the lower bound;

(iii) *average throughput bound*:

$$\bar{\theta} \geq \theta_{ave}\ ,$$

where $\bar{\theta}$ is the average throughput provided by the server, θ_{ave} is its lower bound specified by the client, and both variables are averaged over an interval of duration I specified by the client; the above inequality must obviously hold for all intervals of duration I, i.e., even for that over which $\bar{\theta}$ is minimum.

One clear difference between delay bounds and throughput bounds is that, while the server is responsible for delays, the actual throughputs of a non-saturated system are dictated by the input rates, which are determined primarily by the clients (though they may be influenced by the server through flow-control mechanisms).

3.3. Reliability requirements

The usefulness of error control via acknowledgments and retransmission in real-time applications is doubtful, especially in those environments where message losses are usually higher, i.e., in wide-area networks: the additional delays caused by acknowledgment and retransmission, and out-of-sequence delivery are likely to be intolerable in applications with stringent delay bounds, such as those having to do with continuous media. Fortunately, the loss of some of the messages (e.g., video frames, voice packets) is often tolerable in these applications, but that of sound packets is generally intolerable. In other cases, however, completeness of information delivery is essential (e.g., in file transfer applications), and traditional retransmission schemes will probably have to be employed.

A message may be incorrect when delivered or may be lost in the network, i.e., not delivered at all. Network unreliability (due, for example, to noise) is usually the cause of the former problem; buffer overflow (due to congestion) or node or link failure are those of the latter. The client is not interested in this distinction: for the client, the message is lost in both cases. Thus, the simplest form in which a reliability bound may be expressed and also, we believe, the one that will be most popular, is

$$Prob\,(message\ \ is\ \ correctly\ \ delivered\,) \geq W_{\min}\ ,$$

where $W_{\min}$ is the lower bound of the probability of correct delivery, to be specified by the client. The probability of message loss will obviously be bounded above by $1-W_{\min}$. This is a statistical bound, but, as noted in Section 3, a deterministic reliability bound results if we set $W_{\min} = 1$.

In those applications in which any message delivered with a delay greater than $D_{\max}$ must be discarded, the fraction of messages usable by the destination will be bounded below by $W_{\min} Z_{\min}$. The client may actually specify the value of this product, and let the server decide the individual values of the two bounds, possibly subject to a client-assigned constraint, e.g., that the price of the service to the client be minimum.

If the value of $W_{\min}$ is greater than the system's reliability (the probability that a delivered message is correct), then there is no buffer space allocation in the hosts, interfaces, switches and routers or gateways that will allow the client-specified $W_{\min}$ to be guaranteed. In this case, the server uses error correcting codes, or (if the application permits) retransmission, or duplicate messages, or (if the sequencing problem discussed in Section 4.1 can be solved satisfactorily or is not a problem) multiple physical channels for the same logical channel, or has to refuse the request.

4. Other Required or Desirable Properties

In this section, we briefly describe client requirements that cannot be easily expressed as bounds on, but are related to, communication performance. These include sequencing, absence of duplications, failure recovery, and service setup time. We are not concerned here with features that may be very important but have a functionality (e.g., multicast capabilities) or security (e.g., client authentication) rather than a performance flavor. Requirements in these areas will generally have appreciable effects also on performance; we do not discuss them only because of space limitations.

For a given application, some of these properties may be required, some others only desirable. Also, some may be best represented as Boolean variables (present or absent), some others as continuous or multi-valued discrete variables, others yet as partially qualitative specifications.

4.1. Sequencing

For applications involving message streams (rather than single datagrams), it may be necessary or desirable that messages be delivered in sequence, even though the sequence may not be complete. If the lower-level servers are not all capable of delivering messages sequentially, a resequencing operation may have to be performed at some higher level in the hierarchy. In those cases in which reliability requirements make retransmission necessary, resequencing may delay delivery of a large number of messages by relatively long times. An adequate amount of buffer space will have to be provided for this purpose at the level of the resequencer in the protocol hierarchy.

If sequencing is not guaranteed by all servers at all levels, the application may be able to tolerate out-of-sequence messages as long as their number is small, or if the delay bound is so large that very few out-of-sequence messages have to be discarded because they are too late. The client could be allowed to specify a bound on the probability that a message be delivered out of sequence, or to bundle out-of-sequence losses with the other types of message loss described by W_{min}. The client would specify the value of W_{min} (or $W_{min}Z_{min}$), and the server would have to decide how much probability to allow for buffer overflow, how much for network error, and how much for imperfect sequencing, taking into account the stringency of the delay bounds.

On the other hand, with fixed-route connections and appropriate queueing and scheduling in the hosts and in the network, it is often not too hard to ensure sequenced delivery at the various layers, hence also at the top.

4.2. Absence of duplications

Most of the discussion of sequencing applies also to duplication of messages. It is, however, easier and faster to eliminate duplications than to resequence, as long as some layer keeps track of the sequence numbers of the messages already received. The specification of a bound may be needed only if duplications become very frequent, but this would be a symptom of serious network malfunction, and should not be dealt with in the same way as we handle delays or message losses. These observations do not apply, of course, to the case of intentional duplication for higher reliability.

4.3. Failure recovery

The contract between client and server of a real-time service will have to specify what will happen in the event of a server failure. Ideally, from the client's viewpoint, failures should be perfectly masked, and service should be completely fault-tolerant. As we have already mentioned, however, it is usually unrealistic to expect that performance guarantees can be honored even in presence of failures. A little less unrealistic is to assume that service can resume a short time after a failure has disrupted it. In general, clients may not only wish to know what will happen if a failure occurs, but also have a guaranteed upper bound on the likelihood of such an occurrence:

$$Prob\,(failure) \le F_{max}.$$

Different applications have different failure recovery requirements. Urgent datagrams or urgent message streams in most real-time distributed systems will probably not benefit much from recovery, unless it can be made so fast that hard deadlines may still be satisfied, at least in some cases. In the case of video or audio transmission, timely resumption of service will normally be very useful or even necessary; thus, clients may need to be given guarantees about the upper bounds of mean or maximum time to repair; this may also be the case of other applications in which the deadlines are not so stringent, or where the main emphasis is on throughput and/or reliability rather than on delay.

In communications over multi-node routes and/or long distances, the network itself may contain several messages for each source-destination pair at the time a failure occurs. The recovery scheme will have to solve the problems of failure notification (to all the system's components involved, and possibly also to the clients) and disposition of messages in transit. The solutions adopted may make duplicate elimination necessary even in contexts in which no duplicates are ever created in the absence of failures.

4.4. Service setup time

Real-time services must be requested before they can be used to communicate [Ferr89b]. Some clients may be interested in long-term arrangements which are set up soon after the signing of a contract and are kept in existence for long times (days, months, years). Others, typically for economical reasons, may wish to be allowed to request services dynamically and to avoid paying for them even when not in use. The extreme case of short-term service is that in which the client wants to send one urgent datagram, but this is probably best handled by a service broker ("the datagraph office") using a permanent setup shared by many (or all) urgent datagrams. In most other cases, a request for a short-term or medium-term service must be processed by the server before the client is allowed to receive that service (i.e., to send messages). Certain applications will need the setup time to be short or, in any case, bounded: the maximum time the client will have to wait for a (positive or negative) reply to a request may have to be guaranteed by the server in the contract.

5. Translating Requirements

Performance specifications and other requirements are assigned at the top level, that of the human client or application, either explicitly or implicitly (see Section 2). To be satisfied, these specifications need the support of all the underlying layers: we believe that a real-time service cannot be implemented on top of a server at some level that is unable to guarantee performance.[3] Upper-level requirements must be translated into lower-level ones, so that the implementation of the former will be adequately supported. How should this be done?

5.1. Delay requirements

The method for translating delay bounds macroscopically depends on the type of bound to be translated. All methods have to deal with two problems: the effects of delays in the individual layers, and the effects of message fragmentation on the requirements.

(i) *Deterministic delay bound.* A deterministic bound on the delay encountered by a message in each layer (or group of layers) in the hosts will have to be estimated and enforced. The delay bound for a server at a given level will be obtained by subtracting the delay bounds of the layers above it in both the sending and the receiving host from the original global bound:

[3] Some of the other requirements can be satisfied even without this condition: for example, reliable delivery (when retransmission is acceptable) and sequencing.

$$D'_{\max} = D_{\max} - \Sigma_i \, d_{max,i} \, .$$

Message fragmentation can be handled as illustrated in Figure 1, where we assume that delay is defined as the difference between the instant of completion of the reception of a message and the instant when its shipment began.

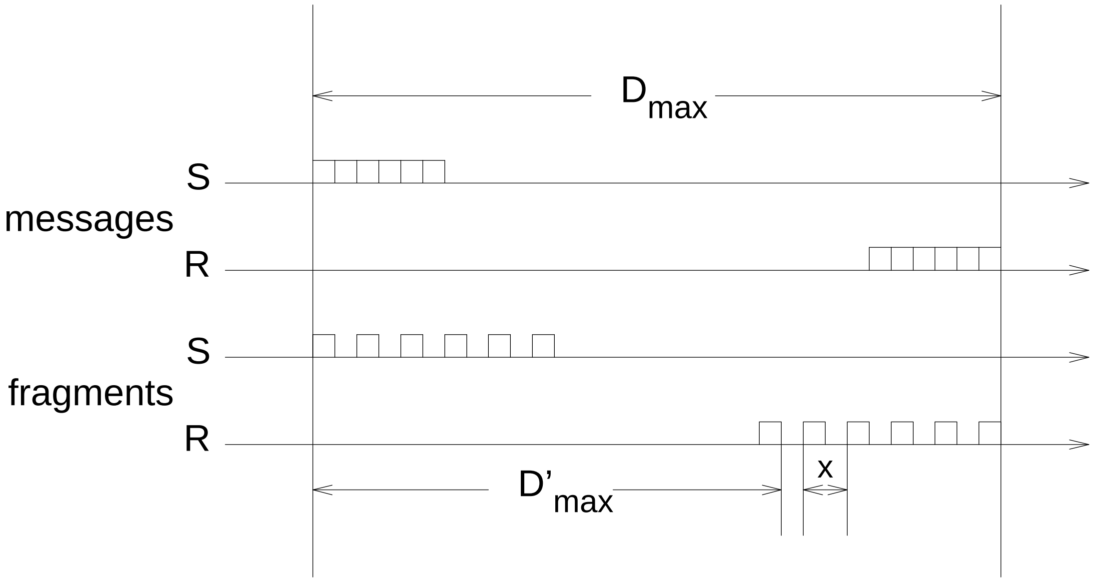

Fig. 1. Relationship between message delay bound $D_{\max}$ and fragment delay bound $D'_{\max}$. S is the time axis of the sender, R that of the receiver.

If x is the interfragment time (assumed constant for simplicity here) and f is the number of fragments in a message, we have

$$D'_{\max} = D_{\max} - x \, (f - 1) \, ,$$

where $D'_{\max}$ is the fragment delay bound corresponding to the message delay bound $D_{\max}$, i.e., the delay of the first fragment.

(ii) *Statistical delay bound*. The statistical case is more complicated. If the bounds on the delay in each layer (or group of layers) are statistical, we may approach the problem of the messages delayed beyond the bound pessimistically, in which case we shall write

$$Z'_{\min} = \frac{Z_{\min}}{\Pi_i \, z_{min,i}} \, ,$$

where the index i spans the layers[4] (or group of layers) above the given lower-level server, $Z'_{\min}$ is the probability bound to be enforced by that lower-level server, and $d_{max,i}$ and $z_{min,i}$ are the bounds for layer i. The expression for $Z'_{\min}$ is pessimistic

[4] A layer has a sender side and a receiver side at the same level in the hierarchy.

because it assumes that a message delayed beyond its bound in a layer will not be able to meet the global bound $D_{\max}$[5].

At the other extreme, we have the optimistic approach, which assumes that a message will not satisfy the global bound only if it is delayed beyond its local bound in each layer:

$$Z'_{\min} = 1 - \frac{1 - Z_{\min}}{\Pi_i \, (1 - z_{min,i})} \, .$$

The correct assumption will be somewhere in between the pessimistic and the optimistic ones. However, in order to be able to guarantee the global bound, the system will have to choose the pessimistic approach, unless a better approximation to reality can be found. An alternative that may turn out to be more convenient is the one of considering the bounds in the layers as deterministic, in which case $Z'_{\min}$ will equal $Z_{\min}$, and the global bound will be statistical only because the network will guarantee a statistical bound.

When estimating the effects of message fragmentation, the new bounds must refer to the fragment stream as though its components were independent of each other. Assuming sequential delivery of fragments, a message is delayed beyond its bound if its last fragment is delayed beyond the fragment bound. Our goal can be achieved by imposing the same probability bound on fragments as on messages [Verm90]. Thus,

$$Z'_{\min} = Z_{\min}.$$

Note that both expressions for $D'_{\max}$ given in (i) above apply to the statistical delay bound case as well.

(iii) *Deterministic delay-jitter bound*. For the case of layer to layer translation, the discussion above yields:

$$J'_{\max} = J_{\max} - \Sigma_i \, j_{max,i} \, ,$$

where $j_{max,i}$ is the deterministic jitter bound of the i-th layer above the given lower-level server. When messages are fragmented, the delay jitter bound can be left unchanged:

$$J'_{\max} = J_{\max} \, .$$

There would be reasons to reduce it in the case of message fragmentation only if the underlying server did not guarantee sequenced delivery, and if no resequencing of fragments were provided by the corresponding reassembly layer on the receiving side.

(iv) *Statistical delay-jitter bound*. The interested reader will be able with little effort to derive the translation formulas for this case from the definition in Section 3.1 (iv) and from the discussion in (ii) and (iii) above.

---

[5] The expression above and the next one assume that the delays of a message in the layers are statistically independent of each other. This assumption is usually not valid, but, in the light of the observations that follow the next expression, the error should be tolerable.

5.2. Throughput requirements

Since all layers are in cascade, the throughput bounds would be the same for all of them if headers and sometimes trailers were not added at each layer for encapsulation or fragmentation. Thus, throughput bounds have to be increased as the request travels downward through the protocol hierarchy, and the server at each layer knows by how much, since it is responsible for these additions.

5.3. Reliability requirements

If we assume, quite realistically, that the probability of message loss in a host is extremely small, then we do not have to change the value of W_{min} when we change layers.

The effects of message fragmentation are similar to those on statistical delay bounds, but in a given application a message may be lost even if only one of its fragments is lost. Thus, we have

$$W'_{min} = 1 - \frac{1 - W_{min}}{f} ,$$

where W'_{min} is the lower bound of the correct delivery probability for the fragment stream, and f is the number of fragments per message. The optimistic viewpoint, which is the one we adopted in Section 5.1 (ii), yields $W'_{min} = W_{min}$, and the observations made in that section about the true bound and about providing guarantees apply.

5.4. Other requirements

Of the requirements and *desiderata* discussed in Section 4, those that are specified as a Boolean value or a qualitative attribute do not have to be modified for lower-level servers unless they are satisfied in some layer above those servers (e.g., no sequencing is to be required below the level where a resequencer operates). When they are represented by a bound (e.g., one on the setup time, as described in Section 4.4), then bounds for the layers above a lower-level server will have to be chosen to calculate the corresponding bound for that server. The above discussions of the translation of performance requirements will, in most cases, provide the necessary techniques for doing these calculations.

The requirement that the server give clear and useful replies to client requests (see Section 2) raises the interesting problem of reverse translation, that from lower-level to upper-level specifications. However, at least in most cases, this does not seem to be a difficult problem: all the translation formulas we have written above are very easily invertible (in other words, it is straightforward to express D_{max} as a function of D'_{max}, Z_{min} as a function of Z'_{min}, and so on).

6. Examples

In this section we describe some examples of client requirements for real-time services. Simplifying assumptions are introduced to decrease the amount of detail and increase clarity. Our intent is to determine the usefulness of the set of requirements proposed above, and to investigate some of the problems that may arise in practical cases. An assumption underlying all examples is that the network's transmission rate is 45 Mbits/s, and that the hosts can keep up with this rate when processing messages.

6.1. Interactive voice

Let us assume that human clients are to specify the requirements for voice that is already digitized (at a 64 kbits/s rate) and packetized (packet size: 48 bytes, coinciding with the size of an ATM cell; packet transmission time: 8.53 μs; packet interarrival time: 6 ms). Since the communication is interactive, deterministic (and statistical) delay bounds play a very important role. Jitter is also important, but does not dominate the other requirements as in non-interactive audio or video communication (see Section 6.2). The minimum throughput offered by the system must correspond to the maximum input rate, i.e., 64 kbits/s; in fact, because of header overhead (5 control bytes for every 48 data bytes), total guaranteed throughput should be greater than 70.66 kbits/s, i.e., 8,834 bytes/s.[6] The minimum average throughput over an interval as long as 100 s is 44% of θ_{min}, due to the silence periods [Brad64].

Voice transmission can tolerate limited packet losses without making the speech unintelligible at the receiving end. We assume that a maximum loss of two packets out of 100 (each packet corresponding to 6 ms of speech) can be tolerated even in the worst case, i.e., when the two packets are consecutive. Since packets arriving after their absolute deadline are discarded if the delay bound is to be statistical, then this maximum loss rate must include losses due to lateness, i.e., 0.98 will have to be the value of $Z_{min} W_{min}$ rather than just that of W_{min}.

This is illustrated in the first column of Table I, which consists of two subcolumns: one is for the choice of a deterministic delay bound, the other one for that of a statistical delay bound and a combined bound on the probability of lateness or loss. If in a row there is a single entry, that entry is the same for both subcolumns. Note that the maximum setup time could be made much longer if connections had to be reserved in advance.

Since voice is packetized at the client's level, we will not have to worry about the effects of fragmentation while translating the requirements into their lower-level correspondents.

6.2. Non-interactive video

At the level of the client, the video message stream consists of 1 Mbit frames, to be transmitted at the rate of 30 frames per second. Thus, the throughput bounds (both deterministic and average) are, taking into account the overhead of ATM cell headers, 4.14 Mbytes/s. As in the case of interactive voice, we have two alternatives for the specification of delay bounds: the first subcolumn is for the deterministic bound case, the second for that of a statistical bound on delays and a combined probability bound on lateness or loss; the latter bound is set to at most 10 frames out of 100, i.e., three out of 30. However, the really important bound in this case is the one on delay jitter, set at 5 ms, which is roughly equal to half of the interval between two successive frames, and between $\frac{1}{4}$ and $\frac{1}{5}$ of the transmission time. This dominance of the jitter bound is the reason why the other delay bounds are in parentheses.

If we assume that video frames will have to be fragmented into cells at some lower level in the protocol hierarchy, then these requirements must be translated at that level into those shown in the first column of Table II. The values of D'_{max} have been calculated with x = 12.8 μs and f = 2605 fragments/frame. The range of W'_{min} and of $(Z_{min} W_{min})'$ is quite wide, and achieving its higher value (a probability of 1) may turn out to be either very expensive or impossible. We

[6] Since the client may not know the overhead introduced by the system, the system may have to compute this value from the one given by the client, which in this case would be 8 kbytes/s.

observe, however, that a frame in which a packet or more are missing or have been incorrectly received does not have to be discarded but can be played with gaps or patched with the old packets in lieu of the missing or corrupted ones. Thus, it may be possible to consider an optimistic approach (e.g., $Z'_{min} = Z_{min}$, $W'_{min} = W_{min}$, $(Z_{min} W_{min})' = Z_{min} W_{min}$) as sufficiently safe.

6.3. Real-time datagram

A real-time datagram is, for instance, an alarm condition to be transmitted in an emergency from one machine to another (or a group of others) in a distributed real-time system. The client requirements in this case are very simple: a deterministic bound is needed (we are assuming that this is a hard-real-time context), the reliability of delivery must be very high, and the service setup time should be very small. The value of 0.98 for W_{min} in Table I tries to account for the inevitable network errors and to suggest that retransmission should not be used as might be necessary if we wanted to have $W_{min} = 1$, because it would be too slow. To increase reliability in this case, error correcting codes or spatial redundancy will have to be resorted to instead.

Note that one method for obtaining a very small setup time consists of shipping such urgent datagrams on long-lasting connections previously created between the hosts involved and with the appropriate characteristics. Note also that throughput requirements cannot be defined, since we are dealing with one small message only, which may not even have to be fragmented. Guarantees on the other bounds will fully satisfy the needs of the client in this case.

6.4. File transfer

Large files are to be copied from a disk to a remote disk. We assume that the receiving disk's speed is greater than or equal to the sending disk's, and that the transfer could therefore proceed, in the absence of congestion, at the speed of the sending disk. The message size equals the size of one track (11 Kbytes, including disk surface overhead such as intersector gaps), and the maximum input rate is 5.28 Mbits/s. Taking into account the ATM cell headers, this rate becomes 728 kbytes/s; this is the minimum peak throughput to be guaranteed by the system. The minimum average throughput to be provided is smaller, due to head switching times and setup delays (seek times are even longer, hence need not be considered here): we set its value at 700 kbytes/s.

Delay bounds are much less important in this example than in the previous ones; in Table I, we show deterministic and statistical bounds in parentheses. Reliability must be eventually 1 to ensure the integrity of the file's copy. This result will have to be obtained by error correction (which will increase the throughput requirements) or retransmission (which would break most delay bounds if they were selected on the basis of the first shipment only instead of the last one).

The second column in Table II shows the results of translating these requirements to account for message fragmentation. The values $x = 78.3$ μs and $f = 230$ have been used to compute those of D'_{max}.

7. Discussion

In this section, we briefly discuss some of the objections that can be raised concerning our approach to real-time service requirements. Some of the objections are fundamental ones: they are at least as related to the basic decisions to be made in the design of the server as they are to client requirements.

Objection 1: *Guarantees are not necessary.*

This is the most radical objection, as it stems from a basic disagreement with our definition of real-time service. The problem, however, is not with definitions or terminologies: the really important question is whether a type of service such as the one we call "real-time" will be necessary or at least useful in future networks. This objection is raised by the optimists, those who believe that network bandwidth will be so abundant that congestion will become a disease of the past, and that delays will therefore be small enough that the enforcement of legalistic guarantees will not be necessary. The history of computers and communications, however, does not unfortunately support these arguments, while it supports those of the pessimists. In a situation of limited resources (limited with respect to the existing demand for them), we believe that there is no serious solution of the real-time communication problem other than one based on a policy for the allocation of resources that rigorously guarantees the satisfaction of performance needs. Even if the approaches to be adopted in practical networks will provide only approximate guarantees, it is important to devise methods that offer without exceptions precisely defined bounds. These methods can at the very least be used as reference approaches for comparison and evaluation.

Objection 2: *Real-time services are too expensive because reservation of resources is very wasteful.*

This may be true if resources are exclusively reserved; for example, physical circuits used for bursty traffic in a circuit-switched network. There are, however, other ways of building real-time services, based on priority mechanisms and preemption rather than exclusive reservation of resources. With these schemes, the real-time traffic always finds the resources it needs by preempting non-real-time traffic, as long as the real-time load is kept below a threshold. The threshold corresponds to the point where the demand by real-time traffic for the bottleneck resource equals the amount of that resource in the system. With this scheme, all resources not used by real-time traffic can be used at any time by local tasks and non-real-time traffic. Congestion may affect the latter, but not real-time traffic. Thus, the only limitation is that a network cannot carry unbounded amounts of real-time traffic, and must refuse any further requests when it has reached the saturation point.

Objection 3: *Real-time services can be built on top of non-real-time servers.*

If one accepts our interpretation of the term "guarantee," one can easily see that performance guarantees cannot be provided by a higher-level server unless it can rely on real-time support by its underlying server. Since this is true at all levels, we conclude that a real-time network service and similar services at all intermediate levels are needed to provide guaranteed performance to human clients and applications.

Objection 4: *Delay bounds are not necessary, throughput requirements suffice.*

Guaranteeing minimum throughput bounds does not automatically and in general result in any stringent upper bound on delay. Delays in the hosts and nodes of a packet-switching network fluctuate because of bursty real-time message streams, starting and ending of traffic on individual connections (even those with continuous, constant-rate traffic), and the behavior of scheduling algorithms. Even if delays did not fluctuate, but had a constant value, it would be possible for a given throughput bound to be satisfied with many different constant values for the delay of each message. If delay bounds are wanted, they must be explicitly guaranteed and enforced.[7]

---

[7] In a circuit-switching network, the circuit assigned to a connection has its own throughput and its own delay.

But are delay bounds wanted? We believe they are in digital video and audio communication, especially in the form of delay jitter bounds, and they will be in other contexts as soon as a service which can bound delays is offered.

Objection 5: *Satisfaction of statistical bounds is impossible to verify.*

Strictly speaking, this objection is valid. No matter how many packets on a connection have been delayed beyond their bound (or lost or delivered with errors), it is always in principle possible for the server to redress the situation in the future and meet the given statistical requirements. A more sensible and verifiable bound would be a fractional one (see Section 3). For instance, such a bound could be specified as follows: out of 100 consecutive packets, no less than 97 shall not be late. In this case, the bound is no longer $Z_{\min}$, a probability of 0.97, but is given by the two values $B = 97$ and $A = 100$; it is not only their ratio that counts but also their individual values.

8. Conclusion

This paper has presented a specification of some of the requirements that human clients and applications may wish to impose on real-time communications. Though those listed seem to be among the most useful and natural ones, no attempt has been made to be exhaustive and comprehensive.

We have investigated delay bounds, throughput bounds, reliability bounds, and other requirements. We have studied how the requirements should be translated from the client’s level into forms suitable (and correct) for lower levels, described some examples of requirement specification, and discussed some of the objections that may be raised.

The material in this paper covers only part of the first phase in the design of a real-time service: that during which the various requirements are assembled and examined to extract useful suggestions for the design of the server. Server needs and design principles will be the subject of the subsequent paper mentioned several times above.

Acknowledgments

Ralf Herrtwich and Dinesh Verma contributed ideas to, and corrected mistakes in, a previous version of the manuscript. The author is deeply indebted to them for their help and for the many discussions he had with them on the topics dealt with in this paper. The comments of Ramesh Govindan and Riccardo Gusella are also gratefully acknowledged.

References

[Brad64] P.T.Brady, "A Technique for Investigating On-Off Patterns of Speech", *Bell Sys. Tech. J.*, vol. 44, pp. 1-22, 1964.

[Ferr89a] D.Ferrari, "Real-Time Communication in Packet-Switching Wide-Area Networks", Tech. Rept. TR-89-022, International Computer Science Institute, Berkeley, May 1989.

[Ferr89b] D.Ferrari and D.C.Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE J. Sel. Areas Comm.* vol. SAC-8, no. 3, pp. 368-379, Apr. 1990.

[Gait90] S.S.Gaitonde, D.W.Jacobson, and A.V.Pohm, "Bounding Delay on a Multifarious

These values may be considered as explicitly guaranteed and enforced.

Token Ring Network", *Comm. ACM*, vol. 33, no. 1, pp. 20-28, Jan. 1990.

[Herr89] R.G.Herrtwich and U.W.Brandenburg, "Accessing and Customizing Services in Distributed Systems", Tech. Rept. TR-89-059, International Computer Science Institute, Berkeley, October 1989.

[Herr90] R.G.Herrtwich, personal communication, February 1990.

[Verm90] D.C.Verma, personal communication, February 1990.

	Interactive Voice *d*	*Interactive Voice* *s*	*Non-Interactive Video* *d*	*Non-Interactive Video* *s*	*Real-Time Datagram*	*File Transfer* *d*	*File Transfer* *s*
Delay bounds:							
deterministic: $D_{\backslash max}$ *[ms]*	200	--	(1000)	--	50	(1500)	--
statistical: $D_{\backslash max}$ *[ms]*	--	200	--	(1000)	--	--	(1000)
$Z_{\backslash min}$	--	(*)	--	(*)	--	--	(0.95)
jitter: $J_{\backslash max}$ *[ms]*	1		5		--	--	
Throughput bounds:							
deterministic: $\theta_{\backslash min}$ *[kby/s]*	8.834		4140		--	728	
average: θ_{ave} *[kby/s]*	3.933		4140		--	700	
I [s]	100		100		--	100	
Reliability bound: $W_{\backslash min}$	0.98	(*)	0.90	(*)	0.98	1	
Delay & Reliability: $Z_{\backslash min} \backslash W_{\backslash min}$	--	0.98	--	0.90	--	--	
Sequencing	yes		yes		--	yes	
Absence of duplications	yes		yes		yes	yes	
Failure recovery (max. repair time [*s*])	10		100		--	100	
Maximum setup time [s]	0.8[(o)]		15[(o)]		0[(‡)]	5[(o)]	

(*)To be chosen by the server
(o)Could be much longer if advance reservations were required
(‡)Could be achieved by using a preexisting connection

	Non-Interactive Video *d*	*Non-Interactive Video* *s*	*File Transfer* *d*	*File Transfer* *s*
Delay bounds				
deterministic: D'_{max} *[ms]*	(966)	--	(1482)	--
statistical: D'_{max} *[ms]*	--	(966)	--	(982)
Z'_{min}	--	(*)	--	(0.95)
jitter: J'_{max} *[ms]*	5		--	
Reliability bound: W'_{min}	0.90-1	(*)	1	
Delay & Reliability: $(Z_{min}\, W_{min})'$	--	0.90-1	--	

(*)To be chosen by the server