Delay Jitter Control for Real-Time Communication in a Packet Switching Network

Dinesh C. Verma
Hui Zhang
Domenico Ferrari
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
& International Computer Science Institute
Berkeley, California

ABSTRACT

A real-time channel is a simplex connection between two nodes characterized by parameters representing the performance requirements of the client. These parameters may include a bound on the minimum connection bandwidth, a bound on the maximum packet delay, and a bound on the maximum packet loss rate. Such a connection may be established in a packet-switching environment by means of the schemes described by some of the authors in previous papers. In this paper, we study the feasibility of bounding the delay jitter for real-time channels in a packet-switched store-and-forward wide-area network with general topology, extending the scheme proposed in the previous papers. We prove the correctness of our solution, and study its effectiveness by means of simulations. The results show that the scheme is capable of providing a significant reduction in delay jitter, that there is no accumulation of jitter along the path of a channel, and that jitter control reduces the buffer space required in the network significantly.

KEYWORDS: Real-time communication, delay jitter, packet-switching network, real-time channel.

INTRODUCTION

Real-time communication services will become a necessity in broadband integrated networks, especially if digital audio and digital video attain the prominence being predicted for them. A real-time communication service allows a client to transport information with performance guarantees. The specific performance guarantees that will be needed will depend on the type of traffic (see [Ferrari 90] for a discussion of these requirements). It is likely that other kinds of traffic (i.e., other than audio or video) will also like to take advantage of guaranteed performance communication. We feel that real-time communication services should be an integral part of future integrated networks, coexisting with the traditional connectionless and connection-oriented services provided by present communication networks [Leiner 89].

The schemes to provide real-time communication can be broadly categorized under the following three switching techniques: circuit switching, packet switching, and hybrid switching. An integrated network based on either circuit switching or hybrid switching typically has very poor resource utilization when bursty traffic needs to be provided with performance guarantees. In addition, hybrid switching requires more complex switches, and does not conform to the goal of fully integrated networks. Full integration is more likely to be achieved by packet switching. However, while packet switching can pro-

---

This research has been supported in part by AT&T Bell Laboratories, Hitachi, Ltd., the University of California under a MICRO grant, the National Science Foundation under Grant No. CDA-8722788, and the International Computer Science Institute. The views and conclusions in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring organizations.

vide performance guarantees regarding delays or loss rates (see [Ferrari 89], [Ferrari 90a] and [Ferrari 90b] for such schemes), it is not very convenient for traffic requiring low delay variation or jitter.

A bound on delay jitter is required by both interactive and non-interactive applications involving digital continuous media to achieve an acceptable quality of sound and animated images. Delay jitter can be eliminated by buffering at the receiver. However, the amount of buffer space required at the receiver can be reduced if the network can provide some guarantees about delay jitter as well. The reduction can be significant for high bandwidth communication. It therefore makes sense to ask whether the schemes providing bounded delays and loss rates can be extended to provide any kind of delay jitter guarantees, and, if so, under what conditions and at what cost. As it turns out, the mechanism to reduce jitter reduces the amount of buffer space required not only in the receiver but also within the network.

From the point of view of a client requiring bounded delay jitter, the ideal network would look like a constant delay line, where packets handed to the network by the sending entity are given to the receiving entity after a fixed amount of time. The jitter of a connection can thus be defined by the maximum absolute difference in the delays experienced by any two packets on that connection. [1] In conjunction with a bound on the maximum delay, a delay jitter guarantee enforces both the maximum and the minimum delay to be experienced by a packet on the channel. The goal of the delay jitter control algorithm, to be described below, is to keep the delay experienced by any packet on a connection within these two bounds, which are specified at connection establishment time.

This paper describes a method for guaranteeing delay jitter in a packet-switching wide-area network, and presents an evaluation of some of its most important characteristics. Since it is an extension to an existing scheme [Ferrari 90a], the method can be used in all environments in which the original scheme can be used. However, like the original scheme, we will present it in the context of a contemporary connection-oriented packet-switching store-and-forward network, and evaluate it by simulation in the same context. Thus, we assume that our network can be modeled as a mesh of nodes connected by links with constant propagation time. Links which do not have a constant propagation time should provide a bound on the maximum delay jitter they can introduce.

The paper is organized as follows: we first revisit the original scheme that provides delay bounds, and then sketch the modifications required to add the delay jitter bounds to the existing scheme. We subsequently describe the simulation experiments we ran to evaluate our scheme, and discuss the results we obtained. Finally, we draw our conclusions.

THE ORIGINAL SCHEME

Real-time communication, as envisioned in [Ferrari 90a], is based on simplex fixed-route connections to be called *real-time channels* or simply *channels*, whose routes will be chosen at establishment time. In order to provide real-time service, clients are required to declare their traffic characteristics and performance requirements [2] at the time of channel establishment according to the following parameters:

- for the offered load
    - the minimum packet interarrival time on the channel, x_{min},
    - the maximum packet size s_{max}, and
    - the maximum service time t in the node for the channel's packets; this includes the time required for transmission, header processing, and any other operations the node may need to perform for the packet;
- for the performance bounds
    - the source-to-destination delay bound D for the channel's packets.

For simplicity, we assume that channels require that there be no packet loss due to buffer overruns in any of the intermediate nodes, and that the client is able to tolerate losses due to the other sources of error in the network.

The original scheme consists of three parts: an establishment procedure, a scheduling mechanism, and a rate control mechanism.

The Establishment Procedure

The channel establishment mechanism may be built on top of any procedure that can be used to set up connections. The goal of the establishment procedure is to break up the end-to-end delay bound D_i required by channel i into the local delay bounds $d_{i,j}$ at each intermediate node j. The local bounds are computed so that, if a node j can assure that no packet

---

[1] Although a bound on the end-to-end delay of a real-time channel is a bound on the delay jitter as well, it is too loose to be acceptable.

[2] We state here only those traffic and performance parameters of the original method that are required to provide a bound on delay jitter. Thus, the parameters mentioned restrict the original scheme to providing deterministic delay bounds for smooth traffic, the kind most likely to require a bound on delay jitter. It is possible to extend the original scheme with its full set of parameters to provide a probabilistic delay jitter guarantee and to incorporate bursty traffic, but we restrict ourselves to the simpler case to keep the length of this paper within reasonable bounds.

on channel i will be delayed locally beyond its local bound $d_{i,j}$, the end-to-end delay bound D_i can be met. As the establishment request moves from the source to the destination, each node on the establishment path verifies that acceptance of the new channel is consistent with the guarantees given by the node to any existing channel. If so, a suggested value of the local delay bound for this channel is included by the node in the establishment request. The destination does the final allocation of the local delay bounds; it may increase the local delay bounds for the intermediate nodes but cannot decrease them. These local delay bounds are assigned to the nodes during the return trip of the establishment message. Each intermediate node also offers an upper bound on the amount of buffer space it can reserve for the new channel. The destination verifies that the amount of buffer space is indeed sufficient for the channel with its final delay bounds, and, if possible, reduces the amount of buffer space required at the intermediate nodes.

Three tests are made at each intermediate node during the forward establishment request.

The deterministic test verifies that there is sufficient bandwidth and processing power at the node. It is done by verifying that

$$\sum_i t_i / x_{min,i} \leq 1 \, , \tag{1}$$

where i ranges over all the channels in the node, including the new one.

The delay bound test verifies that the delay bounds assigned to already existing channels can be satisfied after accepting the new channel. Suppose the scheduling in the node is deadline based (as described later in this section). From the perspective of a channel k, the worst possible arrival pattern on the different channels is one which would cause the deadline of some packet on channel k to be missed. It is possible to determine this worst-case situation for each of the existing channels in the node, and to obtain the value of a lower bound on the new channel's delay bound, so that existing delay bounds are not violated. For further details, we refer the reader to [Ferrari 89].

The *buffer space test* verifies that there is sufficient buffer space in the node for the new channel. In general, the buffer space required for the new channel depends on both the local delay bounds and the traffic characteristics of the channel. Since during the forward trip, the delay bound is not known, the node can use an upper bound (for example, the end-to-end delay) for the purpose of computing the required buffer space. This allocation can be reduced when the final bounds are known. For details, we refer the reader to [Ferrari 90b].

Scheduling

The real-time establishment scheme assumes that scheduling in the hosts and in the nodes will be deadline-based (a variant of the earliest due-date scheduling scheme [Liu 73]). Each real-time packet in the node is given a deadline, which is the time by which it is to be serviced. Let $d_{i,n}$ be the local delay bound assigned to channel i in node n. A packet traveling on that channel and arriving at that node at time t_o will usually [3] be assigned a node deadline equal to $t_o + d_{i,n}$.

The scheduler maintains at least two queues:[4] one for real-time packets and the other for all other types of packets and all local tasks. The first queue has higher priority, is ordered according to packet deadlines, and served in order of increasing deadlines. The second queue can be replaced by multiple queues, managed by a variety of policies.

Rate Control

At channel establishment time, each intermediate node checks whether it will be able to accept packets at the rate declared by the sender. However, malicious users or faulty behavior by system components could cause packets to arrive into the network at a much higher rate than the declared maximum value, $1/x_{min}$. This can prevent the satisfaction of the delay bounds guaranteed to other clients of the real-time service. A solution to this problem consists of providing distributed rate control by extending the deadlines of the ‘‘offending’’ packets. The deadline assigned to an offending packet would equal the deadline that packet would have if it had obeyed the $x_{\min}$ constraints declared at connection establishment time. As a consequence of this rate control scheme, an intermediate node can assume that the clients are obeying the promised traffic specifications even when two packets sent at an interval longer than or equal to $x_{\min}$ by the client come closer together because of network load fluctuations. This rate control scheme requires that the nodes downstream allocate sufficient buffers to provide for any such unintentional violations of the $x_{\min}$ guarantees.

Let us call a node that implements the rate control, scheduling, and admission control mechanisms described above a *bounded-delay server*. A bounded-delay server will ensure that no packet along a channel will spend more than its delay bound in the node, provided that the channel does not send packets faster than its specified rate.

---

[3] There may be some exceptions due to rate control.

[4] There are two types of real-time queues in [Ferrari 89], one for packets with deterministic or absolute delay bounds, and the other for packets with probabilistic delay bounds. Here, as in the rest of the scheme, we have ignored the second type of traffic to simplify our presentation.

THE JITTER SCHEME

The jitter scheme is based on the establishment scheme outlined in the previous section. We utilize the fact that a bound on delay is a bound on delay variation as well. While the global end-to-end delay bound (usually a few tens of milliseconds) is too large to be sufficient as a delay jitter bound, the local delay bound at one single node (a few milliseconds) can serve as a good bound on the jitter of a real-time channel.

In order to provide a delay jitter guarantee, we need sufficiently faithfully to preserve the original arrival pattern of the packets on the channel. If this arrival pattern is faithfully preserved at the last bounded-delay node on the channel's path, the maximum delay variation of packets on the channel will equal the delay bound at the last node.

Each node in our jitter control scheme needs to perform two functions: reconstruct and preserve the original arrival pattern of packets along a channel, and ensuring that this pattern is not distorted too much, so that it is possible for the next node downstream to reconstruct the original pattern. Thus, each node can be looked upon as consisting of a number of logical components: regulators, one for each of the channels passing through the node, each responsible for reconstructing and preserving the arrival pattern of packets along that channel; and a bounded-delay server, shared by all channels and ensuring that the maximum distortion introduced in the arrival pattern at the next node is bounded. This node model is shown in Figure 1.

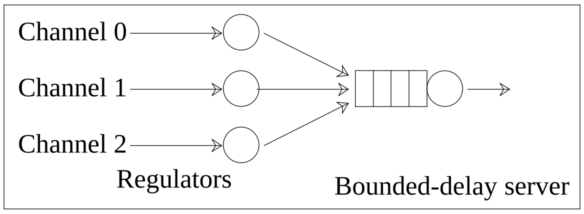

Figure 1: The node model for jitter control

Bounding the delay jitter according to this scheme requires the addition of one performance parameter. Thus, clients must specify the values of the following two parameters for their performance requirements:

- the source-to-destination delay bound D for the channel's packets, and
- the source-to-destination delay jitter bound J for the channel's packets.

The delay of a packet on channel i should not be less than $D_i – J_i$, and not greater than D_i. If no value of J is specified, we set as a default J equal to D.

We call a real time channel with guaranteed delay and jitter bounds (that is, one with different J and D bounds) a deterministic jitter channel, while channels with the same value of J and D are simply called deterministic channels below.

The structure of the regulator module and the channel establishment scheme are described in the next sections.

Establishment Of Bounded Jitter Channels

As in the original scheme, the establishment procedure consists of tests to be performed during the forward trip of the establishment message with each of the intermediate nodes proposing some performance bounds and the destination node relaxing these bounds, if possible. The purpose of the establishment procedure is to determine the local delay bound and local jitter bound at each of the intermediate nodes. For channel i, with an end-to-end delay bound D_i and an end-to-end jitter bound J_i, we need to determine the local delay bound $d_{i,n}$ and the local jitter bound $J_{i,n}$ at each intermediate node n, which has to ensure that every packet on channel i has a local delay greater than $d_{i,n} - J_{i,n}$ but less than $d_{i,n}$.

The paradigm followed is similar to that of the original scheme: each intermediate node offers a suggested value for the performance bounds on the forward trip, and the destination relaxes these bounds, if possible. Thus, three values need to be proposed for a channel being established on the forward trip, a suggested delay bound, a suggested jitter bound, and a suggested bound on the buffer space available at the node. We, however, require that the node always offer the same values for the local delay bound and the local jitter bound. Thus, only two bounds, the bound on buffer space and the bound on delay, need to be suggested by each intermediate node during the forward trip. These are the same bounds as in the original scheme, but the buffer bound computation is done in a different fashion (described below). The delay bound is computed as in the original scheme, and is interpreted as a lower bound on the delay jitter offered by the node as well.

The regulator corresponding to this channel in the node immediately downstream (see Figure 1) is responsible for restoring the original arrival pattern that the channel's traffic had when it entered the network by absorbing the jitter introduced by this node. By reconstructing the original arrival pattern at each intermediate node, the jitter on the channel can be controlled.

Thus, the following two tests will need to be performed at each of the intermediate nodes:

Jitter test: this would comprise the deterministic test and the delay bound computations (except of course that they are now jitter bound computations). The latter would return the value of the least possible jitter that the node can provide to the new channel.

Buffer space test: this determines whether there is sufficient space to accommodate the new channel, and how much of the existing space should be reserved for it.

The buffer space required to prevent any losses consists of two factors: (a) the amount of buffer space required because of the local delay bound at the node, which dictates how long a packet may stay in the node; and (b) the amount of buffer space required to absorb the jitter in the previous node and to reconstruct the original arrival pattern. Assuming the correctness of the algorithms and tests in the original scheme, no packet (after the original arrival pattern is reconstructed) will stay in the node for more than the delay bound in that node. Since the reconstruction is done by delaying each packet so as to absorb the delay jitter from the node immediately upstream, no packet will be held longer than the local delay jitter bound at the previous node in this reconstruction process. It follows that the number of buffers $b_{i,n}$ needed to ensure that no packets from channel i will be lost at node n is

$$b_{i,n} = s_{max,i} \left\lceil \left\lceil d_{i,n} / x_{min,i} \right\rceil + \left\lceil J_{i,n-1} / x_{min,i} \right\rceil \right\rceil \text{ ,(2)}$$

where $d_{i,n}$ is the delay jitter bound assigned to channel i at node n and $J_{i,n-1}$ is the jitter bound assigned to channel i at the previous node.

The buffer space test consists of determining if there is sufficient space (as given by equation 2) available in the node for the new channel. However, at the time of the forward trip, neither the final value of $d_{i,n}$, nor that of $J_{i,n-1}$ are known. Thus the node must assume an upper bound for these values. The simplest way is to bound the sum of both these values by the end-to-end delay requirement D_i of channel i. If the buffer space required by this (admittedly crude) guess is not available, the amount of space available (or a fraction thereof) is temporarily assigned to the channel.

The Destination Host Tests And Algorithms

The destination host, in the modified scheme, has to perform additional tests in order to assure that the end-to-end channel jitter and delay bounds are met. The jitter bound offered by the last node on the path (say the Nth node) must satisfy

$$J_{i,N} \le J_i . \tag{3}$$

The destination must determine whether the delay requirement of the new channel can be met by the nodes along the path. It can do so by verifying that, for channel i with total delay bound D_i,

$$D_i \ \ge \ \sum_{n=1}^{N} J^l_{i,n} . \tag{4}$$

This is a variation of the *D test* of [Ferrari 90a]. The destination now has the responsibility of dividing the delay bound and the jitter bound among the intermediate nodes. A very simple way to do this (distributing delays in the same manner as in the original scheme) would be

$$d_{i,n} = \frac{1}{N} \left[D_i - \sum_{m=1}^{N} J^l_{i,m} \right] + J^l_{i,n} , \tag{5}$$

$$J_{i,N} = J_i , \tag{6}$$

$$J_{i,n} = d_{i,n} , \tag{7}$$

where n ranges over all the intermediate nodes on the channel's path, except the last (Nth) node.

The final test consists of verifying that the buffer space allocated to the channel by the intermediate nodes along the path is sufficient to ensure that packets will not be lost. This consists of recomputing the buffer space requirement at each node according to equation (2), and verifying that the recomputed amount is available in the node.[5]

Rate Control And Scheduling

While the scheduling policy is unchanged in the bounded-delay node(see Figure 1), a new rate control algorithm is performed at each of the newly introduced regulators. The new rate control mechanism is used to restore the arrival pattern of packets that is distorted in the previous node by network load fluctuations. The bounded-delay server needs to perform one more task than in the old scheme: it has to write in each packet's header the difference between the instant the packet is served, and the instant it was supposed to be serviced (its deadline). This information will be used by the regulator immediately downstream.

Let $d_{i,n}$ be the local delay bound assigned to channel i in node n, and $J_{i,n}$ be the local jitter bound assigned to the channel in that node. A packet traveling on channel i that is subjected to the maximum possible delay at node $n-1$ and arriving at node n at time t_o will usually be assigned a node deadline equal to $t_o + d_{i,n}$, and a node *eligibility-time* equal to $t_o + d_{i,n} - J_{i,n}$. The packet is ineligible for transmission until its eligibility-time, which ensures that the minimum delay requirements for channel i are met in node n. Usually the value of $d_{i,n}$ will be the same as $J_{i,n}$; hence, the eligibility-time will be the same as the arrival time at an intermediate node, and a packet will be eligible for transmission as soon as it arrives. Any packet arriving closer than the specified value of $x_{\min}$

---

[5] It is possible to play with the values of delay and buffer space bounds and to devise more sophisticated allocation schemes that would minimize the possibility of rejection by the destination host tests, but we will not discuss them in this paper.

to the previous packet on the same channel is made ineligible for a longer period of time, up to the time it was supposed to arrive obeying the promised minimum inter-arrival time x_{min}. Moreover, the difference between the actual time that the packet was serviced in the previous node and the deadline in the previous node is read from the packet's header (which was stamped there by the previous node), and the packet's eligibility-time (i.e., the time when the packet will be put into the scheduler queue) is increased by this amount. In effect, this extension of the eligibility-time forces each packet on the channel to behave as if it experienced a constant amount of delay, the bound of $d_{i,n-1}$ at the previous node. The difference between a packet's eligibility-time and its deadline always remains the same as the channel's jitter bound at the node.

Let the *holding time* of a packet be defined as the period that the packet is ineligible for service after its arrival. The pseudo-code implementing the above rate control scheme is shown in Figure 2.

Real-time packets which are ineligible for transmission are kept in a queue from which they are transferred to the scheduler as they become eligible. This queue can be maintained as a set of calendar queues [Brown 88] which can be made very fast by hardware implementation; packets are inserted in a queue indexed by their eligibility-time, and all the packets that are in the queue indexed by the current time become eligible.

One important consequence of this rate control scheme is that the arrival pattern of real-time packets entering the scheduler at any intermediate node is identical to the arrival pattern of these packets at the entry point to the network, provided the client obeyed the x_{min} constraint. As a result, the deadline assigned to a packet in a node is given by the time it entered the network plus a constant amount, the constant being the sum of the delay bounds assigned to the nodes along the partial route of the real-time channel covered so far. If $d_{i,k}$ is the delay bound assigned to channel i at the kth node along its path, the deadline dl_n assigned in node n to a channel i packet which entered the network at time t_o is

$$dl_n = t_o + \sum_{k=1}^{n} d_{i,k} + P_n, \qquad (9)$$

where P_n is the propagation delay from the source till node n.

As a result of equation (9), the jitter of packets on a real-time channel at its exit point from the network equals the jitter introduced by the last node in the network, which leads to the justification of the test in equation (3).

```
correction term = deadline in previous node
                  - actual completion time
                  in previous node
holding time    = correction term
                  + delay bound
                  - jitter bound at this node
eligibility-time = holding time
                  + arrival time
deadline        = max[{eligibility-time+jitter bound},
                     {deadline of last packet+$x_{min}$}]
```

Figure 2: The rate control algorithm

Correctness

Before attempting a formal proof of correctness, let us try to give an intuitive explanation why our jitter scheme works. The goal in our scheme is to have a constant delay-line between the sender and the receiver. However, the queueing nodes introduce jitter because the queueing delay in the bounded-delay server is variable. The jitter introduced by each bounded-delay server is absorbed by the regulator at the next node (see Figure 1), and thus, the bounded-delay server and the regulator together provide a constant-delay element. A real-time channel passes through a number of these constant-delay elements, and thus the delay jitter seen by the receiver is merely the delay jitter introduced by the last bounded-delay server along the path.

In order to show that the scheme is correct, and that packet jitter is indeed bounded, we prove two lemmas and then assert a theorem concerning correctness.

Lemma 1: Equation (9) is valid for the first packet of a channel at any node along the path of an established channel. *Proof*: For the first packet, which entered the network at time t_1, the result holds trivially at the first node. As this packet moves on to the second node, it carries with it the correction term, which is the difference between the time it was serviced and its deadline. Suppose the packet was serviced at time instant t'; then the correction term is t_1+d_1-t'. Assuming that the constant propagation delay has been filtered out, the packet reaches the second node at time t' and is assigned a deadline of $t'+d_2+t_1+d_1-t'$ according to Figure 2. Thus, the deadline assigned is $t_1+d_1+d_2$. It is easy to show that the same result holds for other nodes along the path by induction on the path length. The propagation term also builds up as we proceed along the path of the channel.[6]

Lemma 2: If a channel client does not send packets faster than the promised interval $x_{\min}$, equa-

---

[6] If the propagation delay is not a constant on all links, the jitter introduced by the links must also be accounted for. One way to do so would be to add this delay jitter to the right-hand side of equation (6).

tion (9) is valid for all packets on the channel at all nodes along its path. *Proof*: We already know that equation (9) is valid for the first packet on the channel at any node along the path. Let us consider the second packet along the channel, which enters the network at time t_2. Since the client is well-behaved, the packets never enter the network at intervals shorter than $x_{\min}$. Recalling the fact that the rate control algorithm of Figure 2 would execute the same set of instructions as for the first packet, the deadline assigned in the last step would equal that given by equation (9), unless it were modified by the comparison to the previous packet's deadline in the last line of Figure 2. Since the client is well-behaved, the difference between t_2 and t_1 would be at least $x_{\min}$, hence equation (9) applies. Similarly at all the other nodes along the path. Thus, equation (9) is shown to be valid for the second packet and by induction can be extended for all the packets on the channel.

Having proved Lemma 2, we can now state *Theorem 1*: If a packet on an established real-time channel enters the network at time t and the client does not violate his traffic specifications, the packet is delivered by the network to the receiver in between the time instant $t+D-J$ and the time $t+D$.

Proof: From Lemma 2, the deadline assigned at the last node to a packet that