

BUFFERING AT INTERNET CACHES FOR IMPROVING QoS OF STREAMING MEDIA

Adam Wierzbicki¹ Jacques Resing² Warner ten Kate³ Jaap Wessels⁴

¹Institute of Telecommunications, Warsaw University of Technology *

ul. Nowowiejska 15/19, 00-665 Warsaw, Poland; e-mail: adamw@tele.pw.edu.pl

²Department of Mathematics and Computing Science, Eindhoven University of Technology

P.O.Box 513, NL-5600 MB Eindhoven; e-mail: resing@win.tue.nl

³Philips Research Laboratories

Prof. Holstlaan 4, 5656 AA Eindhoven; e-mail: warner.ten.kate@philips.com

⁴EURANDOM, Eindhoven University of Technology

P.O.Box 513, NL-5600 MB Eindhoven; e-mail: wessels@win.tue.nl

ABSTRACT

Transmission of streaming media over the Internet requires the use of buffers for removing jitter and recovery from packet loss. The size and playback delay of these buffers depend on the acceptable amount of packets in time for playback. The use of buffers at Internet caches can lead to smaller buffer sizes at receivers at the cost of a small additional playback delay. The paper studies the costs and gains of buffering streams at Internet caches to remove jitter and recover from loss. Trace-driven simulations are used to give a realistic evaluation of buffering at caches.

1. INTRODUCTION

Transmission of streaming media over a best-effort network often degrades quality of playback at the receiver. This degradation is on one hand due to variability in network transmission times, also called network jitter, and on the other hand due to network loss. To remove jitter and recover from loss in time for playback, receivers delay the playback of the stream and store portions of the stream in buffers. If the clocks of the sender and the receiver are synchronized using a mechanism such as NTP, the sender and receiver determine a playback delay which allows a majority of the packets to arrive before they are needed for playback.

The stream transmitted from sender to receiver can pass through an Internet cache. This may be

*this work has been carried out while the author was visiting Philips Research and EURANDOM, Eindhoven University of Technology

because the receiver requested the stream from the cache in the hope of finding a copy there. Our interest focuses on the case where the stream is multicasted from the sender to a large number of receivers, with the cache acting as a proxy. This is illustrated in Figure 1. A receiver can first join the multicast group of the sender, then perform an expanding ring search [7] to discover which server (cache or sender) is closest. A cache would respond by sending the address of its multicast group. If the cache is closer to the receiver than the sender, the receiver would leave the multicast group of the sender and join the group of the cache.

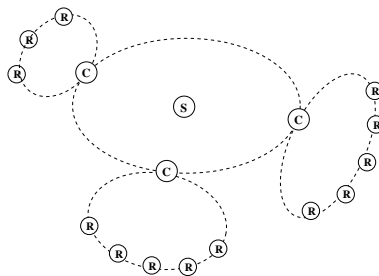


Figure 1: Use of caches for multicast transmission of streaming media. S =Sender, C =Cache, R =Receiver.

Every cache can be a member of two multicast groups. They receive the streamed packets from the sender and forward them to the receivers. The caches also can recover lost packets on behalf of the receivers. Caches buffer the packets, which enables them to smooth the stream between the sender and

the receivers. During the transmission from the cache to the receivers the stream may again be subject to loss and variable delay in the network. However, retransmission requests from the clients can be satisfied by the cache. This leads to savings in bandwidth, decreased latencies, and avoids the feedback implosion problem of multicast transfer.

Our aim is to study the effect that buffering at a cache and receiver can have on the playout quality of the stream at the receiver. The buffer at the cache removes part of the jitter from the stream, and therefore the receiver can have a smaller buffer. Since one cache can serve several receivers, the savings in buffer space for a single receiver are multiplied by the number of receivers. However, use of a buffer at the cache can introduce additional playback delay.

The object of our work is to determine the performance gains and costs of using Internet caches to combat jitter. The approach which uses a buffer at the cache and at the receiver will be compared with the approach which uses a buffer only at the receiver. We first study an analytical model of the two approaches and derive expressions for the buffer sizes and playback delays required to guarantee a desired playout quality at the receiver. We use these expressions in a simulation driven by traces of measurements of network delay and loss. By using traces, we are able to give realistic values of buffer size and playback delay, which allows for a better comparison of the two approaches.

The use of buffers for combating jitter has been studied in [9] for voice transmission. The same approach has been used in ATM networks for circuit emulation facilities [3]. The COST monograph [3] contains an analysis of the performance of buffers at the receiver which is similar to ours. However, the expression given for overflow probability is valid only in a special case. We give a general upper bound and have evaluated the performance of the bound using simulation [11].

The use of distributed buffers has been suggested in [1]. The analysis used by the authors concerned networks with performance guarantees, while our analysis considers best-effort networks. Depending on the protocol design philosophy, the distributed buffering is implemented at the network layer, or uses Internet caches, which operate at the application layer. In [8], the authors suggest the use of caches to combat jitter, but focus on the smoothing of bandwidth requirements for variable bit-rate video. They also suggest the caching of a prefix of the entire stream to hide the playback delay from the receiver. Our analysis can be used to calculate the length of such a prefix. To our knowledge, the

suggested approaches have not before been evaluated on traces of Internet traffic.

The comparison of the two approaches: a single buffer at the receiver and two buffers at a cache and the receiver, begins with an analysis which ties the buffer dimensions to the quality of playback at the receiver. The buffer dimensions meant here are buffer size and playback delay. The quality of playback at the receiver is measured by the number of packets in time for playback. Packets may not be available at playback time either because they were too late or because they have been dropped when the buffer ran out of storage space.

The analysis allows us to give expressions for buffer dimensions in both approaches given a certain required quality of playback at the receiver. The buffer dimensions depend on distributions of transmission delays, which in the approach with a buffer only at the receiver concern the delay from the sender to the receiver, and in the approach with buffers both at a cache and at the receiver concern the delays in the two separate parts of the network path. We use empirical distribution functions derived from trace data to find realistic buffer dimensions. Since many of our measurements have high percentages of lost packets, we simulate protocol mechanisms for recovery of lost packets on our traces. This influences the distribution functions of transmission delay and the buffer dimensions. We compare the two approaches for the same levels of playback quality at the receiver. We find that in most cases the use of a cache reduces the required buffer size at the receiver at the cost of a negligible additional playback delay.

The outline of the paper is as follows. In Section 2 we present the models for the two different approaches. Section 3 is devoted to the analysis of the models and its results. Next, in Section 4 we briefly describe the methodology of measurements and the traces used in our study. In Section 5 we describe the simulation of protocol features for recovery of loss and show the empirical distribution functions of transmission delay derived from the traces. Section 6 gives results and discusses costs and gains of using caches for improving quality of playback at the receiver. Finally, we give some conclusions in Section 7.

2. MODEL DESCRIPTION

In this section we describe two models, for the approach with only a buffer at the receiver R and for the approach with buffers at a cache C and at the receiver R .

In Figure 2a, we show the model for the approach with a buffer only at the receiver. For simplicity we

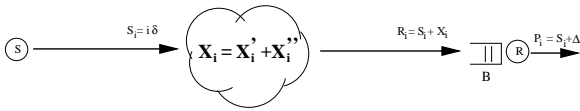


Figure 2a: Single buffer at receiver

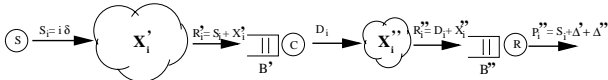


Figure 2b: Buffers at a cache and receiver

Figure 2: Models of the two buffering approaches.

consider the transmission of a Constant-Bit-Rate (CBR) stream of packets, all having the same size¹. Packets are sent by the sender S every δ milliseconds at times $S_i, i = 0, 1, 2, \dots$. The transmission delay of the i -th packet, possibly including the time needed to recover from loss, is denoted by X_i . The only assumption that we make for the stochastic process $X_i, i = 0, 1, 2, \dots$ is that it is a stationary process and that $X_i > 0$. The i -th packet has a schedule for playback, P_i , which is given by shifting the sender's transmission schedule by a fixed playback delay Δ . The moment that the i -th packet arrives at the buffer at the receiver is denoted by R_i . So, clearly the sequences S_i, R_i and P_i are given by

$$S_i = i\delta, \quad R_i = S_i + X_i, \quad P_i = S_i + \Delta \quad (1)$$

If $R_i > P_i$ then packet i is too late for playback. This is called *buffer underflow*. If on the other hand $R_i < P_i$, then packet i has arrived early for playback and must be stored in a buffer of fixed size. We shall denote the buffer size by B . If the buffer is full when a packet that must be stored arrives, one of the packets has to be removed from the buffer. This is called *buffer overflow*. The probabilities of buffer overflow and underflow at the receiver are measures of quality of playback.

We consider a pessimistic buffer management scheme based on *reservations*. The buffer reserves storage space for packet i during the interval $[P_{i-B}, P_i]$. If packet i arrives before time P_{i-B} , it will be dropped, independently of the actual number of packets in the buffer at that moment. If it arrives during $[P_{i-B}, P_i]$, it will be stored in the reserved space. (In this way, out-of-order arrivals are allowed in our model.)

In this paper, we use the fraction of dropped packets in the reservation scheme as a (pessimistic) approximation for the overflow probability in a

¹For Variable-Bit-Rate encodings, if packets have varying sizes but are sent at equal intervals, the expressions for playback delay are valid.

more realistic buffer management scheme. Furthermore, remark that a reservation expires at the time P_i , so that packets which are late for playback are never stored. This means that the use of reservations does not affect the underflow probability. Reservations are introduced solely for the purpose of analysis, as will become clear in the next section. Later on, in the simulations in Section 5, we simulate buffers which use other buffer management schemes.

In Figure 2b, we show the model for the approach with buffers both at a cache and at the receiver. We use a similar notation and terminology as before, and add a single prime to refer to the part of the network from sender to cache and a double prime to refer to the part of the network from cache to receiver. Note that this model does not take into account the fact that there might be many receivers connected to the sender by different paths, which would make it necessary to model the network as a tree. Such a model would be more appropriate for a study of placement of the cache. However, our purpose is to evaluate the buffer savings, and we believe that for this purpose the model is sufficient. If we need many receivers in our model, we can also think that a cluster of receivers is connected by a very fast Local Area Network to the last node in the path, R .

The transmission delays in the two parts of the network are denoted by X_i' and X_i'' , respectively. For a comparison with the previous model, we shall assume $X_i = X_i' + X_i''$. Processing delay at the cache is assumed to occur within the packet's stay in the buffer and otherwise to be negligible.

We assume that the buffer at C uses a similar reservation scheme and packets are dropped upon overflow and underflow. We refer to this variant as a **dropping buffer**. Hence, the departure time D_i of packet i at the cache is either equal to the playback time P_i' or undefined (if the packet is dropped). The stream leaving the cache can be seen as a CBR stream with gaps.

Alternatively, we could have considered the approach where the buffer at C does not drop packets which underflow or overflow, but will instead send a packet to the receiver. Such a buffer can delay a packet even if it is much too early and caused overflow. In case of overflow, the packet closest to playback (that has been delayed most) in the buffer is pushed out (D_i is ahead of P_i'); in case of underflow, the packet is forwarded upon eventual arrival (D_i is lagging P_i'). We refer to this variant as a **smoothing buffer**. The stream leaving the cache can be seen as a jittered CBR stream (with no gaps). In our analysis we use the drop-

ping buffer strategy, since the use of a smoothing buffer would considerably complicate the analysis. Instead, a smoothing buffer is studied by simulation in Section 6.

In the next section, we shall consider the following questions.

- What is the required playback delay Δ and buffer size B in the approach with only a buffer at the receiver, in order to achieve certain acceptable underflow and overflow probabilities ϵ_u and ϵ_o ?
- What are the required playback delays Δ' and Δ'' and the buffer sizes B' and B'' in the approach with both buffers at the cache and at the receiver, in order to achieve certain acceptable underflow probabilities ϵ'_u and ϵ''_u and overflow probabilities ϵ'_o and ϵ''_o ?

Once we have answered these two questions we are able to compare the playback delays and buffer sizes required in the two approaches in order to obtain the same level of playback quality at the receiver.

3. ANALYSIS OF BUFFER DIMENSIONS

We begin with an analysis of the model of a single buffer at R . The results can be used for the model of a buffer at C ; for an analysis of the buffer at R in the approach with a cache we will need to know the departure times from C .

The probability of buffer underflow of packet i is given by the formula:

$$P\{\text{underflow}\} = P\{R_i > P_i\} = P\{X_i > \Delta\} \quad (2)$$

An increase in Δ would therefore result in a decrease of the probability of underflow. Let $F_{X_i}(t) = P\{X_i < t\}$ be the cumulative distribution function (CDF) of transmission delays. Then we can give an expression for Δ in terms of the maximum acceptable probability of underflow, ϵ_u :

$$\Delta = F_{X_i}^{-1}(1 - \epsilon_u) \quad (3)$$

If the buffer uses reservations, it will drop packets which arrive earlier than P_{i-B} ; in other words, overflow occurs when $R_i < P_{i-B}$. If the buffer uses reservations, it will never run out of space. The probability of overflow of packet i in a buffer with reservations is therefore given by:

$$\begin{aligned} P\{\text{overflow}\} &= P\{R_i < P_{i-B}\} \\ &= P\{S_{i-B} + B\delta + X_i < S_{i-B} + \Delta\} \\ &= P\{X_i < \Delta - B\delta\} \end{aligned} \quad (4)$$

Note that when the buffer size $B \geq \Delta/\delta$, overflow probability is zero.

The analysis of overflow is more complex if we do not assume reservations, since the amount of packets in the buffer can depend on the entire history of the arrival process. It also depends on which packet is discarded when overflow occurs.

Note that the pessimism of the use of reservations does not affect the fairness of our comparison of the two approaches of a single buffer and a buffer at C and R , since both apply pessimistic sizes. Also, an application would have to determine buffer size in some way, and a better approximation than the given one is not known to us.

Let ϵ_o be the acceptable probability of overflow. If the buffer at R uses reservations, its size is given by the expression:

$$\begin{aligned} B &= \frac{\Delta - F_{X_i}^{-1}(\epsilon_o)}{\delta} \\ &= \frac{F_{X_i}^{-1}(1 - \epsilon_u) - F_{X_i}^{-1}(\epsilon_o)}{\delta} \end{aligned} \quad (5)$$

We have expressed the dimensions of the buffer in terms of the distribution of delays $\{X_i\}$ and acceptable levels of overflow and underflow. The same can be done for the buffer at C in the approach which uses a cache:

$$\begin{aligned} \Delta' &= F_{X'_i}^{-1}(1 - \epsilon'_u) \\ B'\delta &= \Delta' - F_{X'_i}^{-1}(\epsilon'_o) \end{aligned} \quad (6)$$

For the buffer at R in the approach with the cache, the distribution of delays is influenced by the amount of jitter of the packet stream released by the cache. To dimension the buffer at R , we need to consider distributions of delays $D_i - P'_i + X''_i$.

$$\begin{aligned} \Delta'' &= F_{D_i - P'_i + X''_i}^{-1}(1 - \epsilon''_u) \\ B''\delta &= \Delta'' - F_{D_i - P'_i + X''_i}^{-1}(\epsilon''_o) \end{aligned} \quad (7)$$

If the cache uses a dropping buffer, the departure times are never different from the playback schedule: $D_i = P_i$ (if a packet is dropped, its departure

Table 1: Summary of trace measurements. The source is Eindhoven, The Netherlands.

id	destination	nr. of hops	δ	loss	average RTT	min RTT	max RTT
AMS	Amsterdam	4	20	0.0%	5	4	106
NY1	New York 1	5	35	0.5%	80	79	200
NY2	New York 2	6	76	1.3%	82	81	194
EST	Estonia	12	430	15.8%	566	82	2810
SIN	Singapore	14	16	22.9%	416	374	936
NOV	Novosibirsk	15	210	36.9%	623	278	1090
CAL	California	18	227	0.9%	173	170	195

time is undefined). This means that we are able to give closed expressions for the buffer dimensions at R without knowing the distribution of X'_i :

$$\begin{aligned} \Delta'' &= F_{X''_i}^{-1}(1 - \epsilon''_u) \\ B'' \delta &= \Delta'' - F_{X''_i}^{-1}(\epsilon''_o) \end{aligned} \quad (8)$$

If the buffer at C does not drop packets but sends them to the receiver, the departure times from the cache depend on which packet is chosen to be removed from the buffer when overflow occurs. In some cases, they can be expressed analytically; in other cases, the departure times can be obtained by simulation.

In Section 5, we shall use trace-driven simulations to evaluate the performance effects of the approach which uses a cache. Every simulation will use two traces for the delays X'_i and X''_i . The buffers will be dimensioned under the assumption that all buffers use reservations, and that the cache uses a dropping buffer. The expressions for buffer dimensions derived in this section will be applied to empirical cumulative distribution functions of delays to determine the buffer dimensions. However, we shall simulate a system which violates these assumptions and uses other buffer management schemes to judge the pessimism of the expressions for buffer size. To compare the two approaches for the same playback quality, we will choose the underflow and overflow such that $\epsilon'_u + \epsilon'_o + (1 - \epsilon'_u - \epsilon'_o)(\epsilon''_u + \epsilon''_o) = \epsilon_u + \epsilon_o$, i.e. the drop probability is the same in both approaches (if $\epsilon'_u + \epsilon'_o > \epsilon_u + \epsilon_o$, comparison for the same level of playback quality is impossible.)

In the next section, we will describe the measurements made to gather our trace data.

4. MEASUREMENTS OF ROUND-TRIP

DELAY AND LOSS

To obtain our trace data, we use round-trip measurements obtained from ICMP ping. Although the RTT measured by Ping may include several other (processing) delays, it provides us with sufficient accuracy to capture the range of variability of transport delays. For the simulation of recovery from packet loss, which is described in the next section, round-trip loss and round-trip time (RTT) are more relevant than one-way measurements, since many protocols will use acknowledgments which can be lost as well as data packets, and adapt retransmission timeouts based on the RTT.

We have selected destinations from a group of Web servers which offer access to a reverse traceroute service: clients can run a CGI script on the server which runs traceroute in the reverse direction [2]. This has given us the opportunity to observe both the forward and reverse path. We have preceded our measurements by a period of path observation to determine if routing is stable and if paths are sufficiently symmetric, since that increases the precision of estimates of one-way transfer time and loss [10].

Finally, we have used the treno program and (for loss below 3%) the formula for TCP throughput [4] to determine the throughput available to TCP connections on the measurement paths. The ping packets had a size of 1008 bytes. Their sending frequency was limited so that we would not exceed available TCP throughput. Although the experiment may not exactly reproduce the conditions of a real streaming media application, we believe that it is sufficiently realistic to be used as an evaluation of buffer dimensions.

Table 1 summarizes the traces obtained in our study. The measurements were made from the campus of Eindhoven University of Technology, The Netherlands. All these traces were made to destinations which showed symmetric, stable paths. The

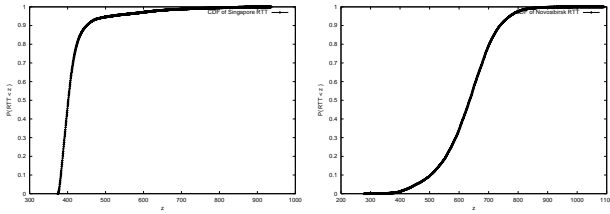


Figure 3: Empirical CDF of delay in traces SIN and NOV assuming no packet loss.

measurements were made on working days in May and June, 2000, starting around noon (MET). Their duration depended on the number of packets and sending frequency, but never exceeded 6 hours. All times in the table are given in milliseconds (rounded up). Loss values are given for the round trip path, transmission times are round trip times, number of hops concerns the one-way path.

We shall always assume that the receiver R is located at Eindhoven. The destinations in the first three rows of the table represent possible locations of the cache C : **AMS**, **NY1**, **NY2**. The destinations in the other four rows represent the possible locations of the sender S . The routes to these four destinations all passed through a router in Amsterdam, and two routers called New York 1 and 2.

If the cache were located at Amsterdam and the sender at Singapore, then the **SIN** trace would model the path from S to C and the **AMS** trace the path from C to R . Since we can only make round-trip measurements between Eindhoven and the destination, the network path between Eindhoven and Amsterdam is part of both paths. Instead of using delays from Amsterdam to Singapore which we cannot estimate, we model the path from S to C by measurements of the path from Eindhoven to Singapore. This should not influence the fairness of our comparison, since we construct delays from S to R by adding delays from the traces **AMS** and **SIN**. However, it makes the simulation of placement more difficult since when we change the position of the cache we also change the path from S to R .

The traces **AMS**, **NY1**, **NY2** have little loss, jitter and delay when compared to the others, with the exception of **CAL**. The finding that jitter and loss observed on a regional network are usually smaller than on international or intercontinental paths is supported by [5], where the authors compared the amount of jitter in various portions of a wide area network.

In Figure 3 two examples are shown of empirical cumulative distribution functions of the transfer delay obtained from the traces **SIN** and **NOV**. The reader can use expressions 3 and 5 to find dimen-

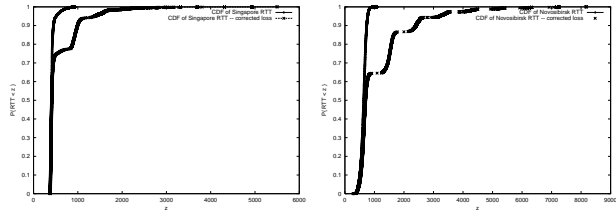


Figure 4: Empirical CDFs of delay with and without loss recovery in traces SIN and NOV.

sions of a buffer at C from the figures. Since both traces have significant loss, the simulation of recovery from loss will change the shape of the distribution functions, as we will show in the next section.

5. RETRANSMISSIONS OF LOST PACKETS

We shall consider recovery from loss based on retransmissions. Another possibility would be the use of Forward Error Correction (FEC). If a simple FEC scheme were used which transmits FEC coding for every packet, then we would expect that the transmission delay will not be affected by loss. On the other hand, the use of more sophisticated block-based FEC schemes may affect transmission delay (since if a packet is lost, it may be recovered only after a sufficient number of packets of a block have been transmitted successfully). This is a subject of future work.

We chose to simulate a retransmission scheme based on positive acknowledgments (ACKs) sent from R to S . This loss recovery scheme is best suited to our experimental data, which contain round-trip delay and loss on a path from R to S . We assume that the sender S maintains a timeout based on measured RTT values. The timeout is calculated based on the moving average of the RTT and the RTT variation [6]. After a packet is sent, a timer is started. If no ACK arrives for the packet before the timeout expires, the sender retransmits the packet. The retransmission may also be lost, leading the procedure to repeat.

While a retransmission scheme based on negative acknowledgments may be more efficient for multicast distribution, its effect on the transfer delay will not be much different from the described scheme. It is also slightly more difficult to simulate from our traces since it requires a timeout calculated from the One-way Transfer Times (OTT) for the first transmission of a packet.

Based on these assumptions, we simulate the recovery of lost packets in the following way: in the trace of ping measurements, we identify the gaps in the sequence numbers. Since we cannot identify

Table 2: Comparison of the two buffer approaches.

S to C	C to R	$B'; \Delta'$	$B''; \Delta''$	$B; \Delta$	$B - B''$	$\Delta' + \Delta'' - \Delta$
SIN	AMS	89; 2153	1; 13	85; 2078	84	88
EST	AMS	252; 5267	1; 13	247; 5168	246	112
NOV	AMS	242; 5218	1; 13	238; 5161	237	70
CAL	AMS	9; 346	1; 13	1; 188	0	171
EST	NY1	252; 5267	5; 166	247; 5243	242	190
EST	NY2	252; 5267	5; 166	247; 5245	242	188

whether a loss has happened in the forward (data) or the return (its ACK) path, we always assume that a loss in the trace means that the packet has not yet arrived at the receiver. (A lost ACK would cause a data retransmission, but would not affect the delay associated with that, doubly received, data packet. In the NAK-based scheme it would, however.)

We calculate the timeout TO_i^{rtt} based on the expressions:

$$\begin{aligned}
 \bar{X}_i &= \alpha \bar{X}_{i-1} + (1 - \alpha) X_i \\
 \sigma_i^X &= \beta \sigma_{i-1}^X + (1 - \beta) |X_i - \bar{X}_i| \\
 TO_i^{rtt} &= \bar{X}_i + 4\sigma_i^X
 \end{aligned} \tag{9}$$

where X_i is the round-trip time, $\alpha = \beta = 0.998$.

In case of a gap, we use the timeout to identify the position in the trace where retransmission would occur (since ping sends packets at a constant rate). If this position is a loss, another retransmission is simulated. Therefore, the final transmission time with loss recovery is given by the equation:

$$X_i = (k - 1)TO_i^{rtt} + ott \tag{10}$$

where k is the number of times packet i is transmitted until it is received successfully, ott is a one way transfer time.

The effect of the simulation of loss recovery on the traces to Singapore and Novosibirsk is shown in Figure 4. The distribution of round trip times without loss recovery is plotted for reference. The new distribution is much more broad and has a characteristic stair-like shape.

6. BUFFER SIMULATIONS

Since we are now able to give realistic estimates of transmission times, we can use them as arrival processes to obtain an evaluation of the approach with a cache in terms of saved buffer space at the

receiver ($B - B''$) and additional delay ($\Delta' + \Delta'' - \Delta$).

For every simulation, we use two traces: one of them simulates the network path from S to C , while the other simulates the path from C to R . First, we dimension the buffers in the two approaches. We apply the expressions from Section 3 to empirical cumulative distribution functions of delay in the two traces with simulated loss recovery.

Since the expressions were derived under the assumption of using reservations, they may be inappropriate for other buffer management schemes. We simulate a buffer which deals with out-of-order arrivals by keeping the packets sorted according to their sequence numbers. Overflow occurs if the buffer size is exceeded.

We compare the frequencies of overflow obtained from the simulated buffer with the values the buffer had been dimensioned for. For most simulations, the buffer size was set close to Δ/δ ; and while the predicted overflow was low, the simulations revealed zero overflow. More detail is described in [11].

The performance of the approach with a buffer at the cache depends on the chosen level of underflow and overflow at C . In our simulations, we have kept underflow and overflow equal to each other. We shall refer to them as $\epsilon = \epsilon_u = \epsilon_o$ for R in the approach without a cache and $\epsilon' = \epsilon'_u = \epsilon'_o$ for C and $\epsilon'' = \frac{\epsilon - \epsilon'}{1 - 2\epsilon}$ for R in the approach which uses a cache. ϵ'' is chosen for comparison with the same playback quality. A buffer releases packets at their playout time with probability $1 - 2\epsilon$ (at R) or $1 - 2\epsilon'$ (at C).

The savings in buffer size at R are given in numbers of packets. away from the receiver, the potential population of receivers which could use it was large. For all simulations, we assumed a packet would be sent every 20 milliseconds. All delay values are rounded up to an integer number of milliseconds.

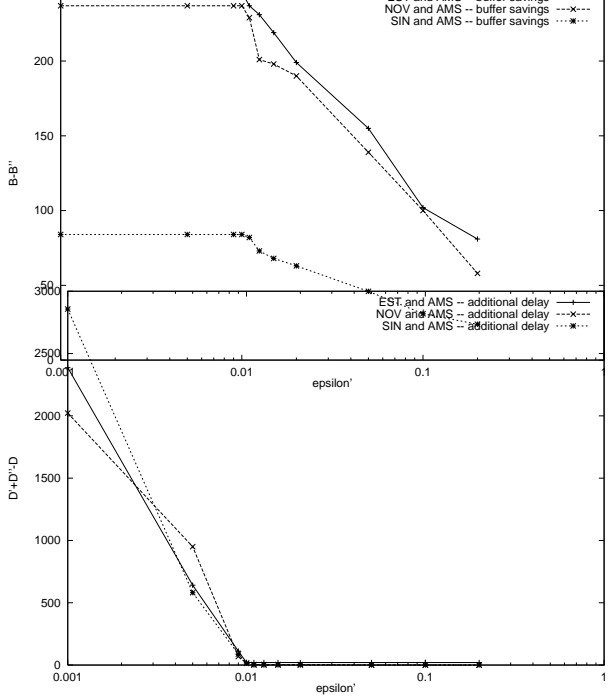


Figure 5: Performance of two buffers for varying ϵ' , constant $\epsilon = 1\%$.

In Table 2 we show the buffer dimensions in the two approaches for various traces. The table shows the two traces chosen for the simulation; buffer dimensions of the first and second buffer in the approach with a cache; buffer dimensions for a single buffer at the receiver. All simulations were made for $\epsilon = 1\%$, $\epsilon' = 0.9\%$, $\epsilon'' \approx 0.1\%$. The table does not show the combinations of all traces. The traces **NY1** and **NY2** are too similar to **AMS** to change the results much. We show simulations of **EST** with **NY1** and **NY2**.

The cost in terms of additional delay compared to the end-to-end delay is on the order of a few percent in all cases but one. That one, the **CAL-AMS** trace, exhibits a low end-to-end delay (and a low jitter, as well) by itself. The buffer savings compared to the single buffer case are on the order of 99%. Note, however, that the buffer introduced at the cache is larger than the buffer in the single buffer case. The observed $B' > B$ is caused by choosing $\epsilon' < \epsilon$ and by the fact that there is little jitter and loss downstream of C . The overall savings in buffer space are $c(B - B'') - (B' - B)$, where c is the number of receivers. If the cache is shared by multiple receivers, this still can lead to a saving of the total buffer space. If a smoothing buffer is used, then we can set $\epsilon' < \epsilon$, which implies $B' < B$. The performance effect of ϵ' is discussed next.

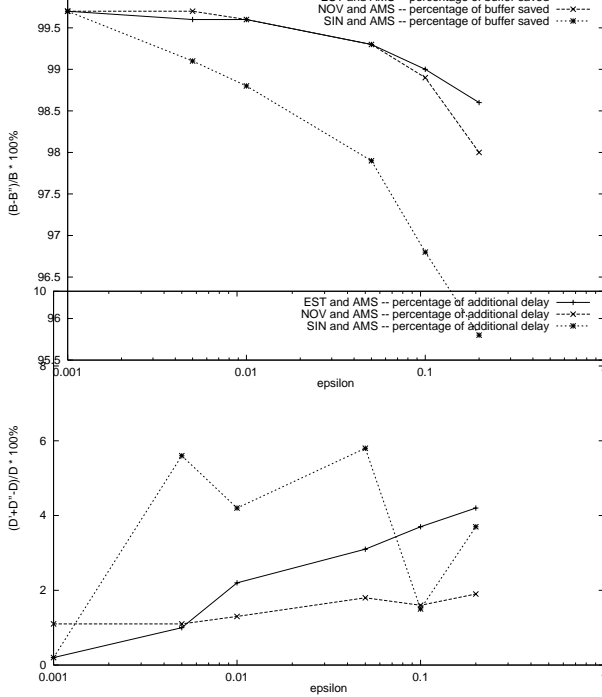


Figure 6: Performance of two buffers for varying ϵ' , $\epsilon' = 0.9\epsilon$.

We now show how the buffer savings at the receiver and additional delay depend on the value of ϵ' when ϵ was constant. For this simulation, we have used the **NOV**, **EST**, **SIN** and **AMS** traces; the buffer at the receiver was dimensioned for $\epsilon = 1\%$. Since ϵ was constant, the buffer dimensions at R in the approach without a cache were constant (they are given in table 2). We have simulated both a dropping buffer and a smoothing buffer at C . However, for $\epsilon' < \epsilon$ the values for the dropping buffer were almost identical to the smoothing buffer, and in the other case we do not have the same levels of playback quality in the two approaches if the dropping buffer is used. Therefore we show results only for the smoothing buffer.

The results are shown in Figure 5. ϵ' is shown on the x axis on a logarithmic scale. When the buffer at C is designed to release much less jitter than the receiver can accept, the comparison results in high additional delay and high buffer savings. When the amount of jitter released by C is slightly smaller than the receiver can accept, the additional delay is small, and the buffer savings are as large as before. When ϵ' increases further, above 1%, the performance deteriorates slowly, but additional delay stays low. These results indicate that the buffer at the cache should reduce the jitter of a stream to a level slightly less than what the receiver is capable of accepting, which depends on the quality of the path from C to R and on the requirements of the

application. This conclusion is also supported by the deviating **CAL-AMS** combination, since the **CAL** trace already has less jitter than the **AMS** trace, and therefore the cache should forward the stream without buffering.

For the same combinations of traces, we show the performance effects when underflow and overflow at R is varied for the two approaches. ϵ' is always equal to 90% of the chosen levels of underflow and overflow at R . Because of that, the performance of a dropping and a smoothing buffer is almost identical. In Figure 6 we show the results of this simulation. ϵ varies from 0.1% to 20% and is shown on the x axis on a logarithmic scale. We show the relative values of buffer savings and additional delay. As expected, the savings decrease when we relax our requirements for the amount of packets which are in time for playback; however, the decrease is small. On the other hand, additional delay is not affected by the change of ϵ .

7. CONCLUSIONS

We discussed the performance of using caches to combat jitter in terms of saved buffer space and additional end-to-end delay at the receivers. We developed an analytical model and performed simulations using measured trace data. We found that the cost of additional delay can be low, while the gain of receiver buffer size can be high. The results depend on the network conditions of the path from sender to receiver. If the portion of the path from cache to receiver had low loss and jitter, the discussed approach performed well. If the conditions downstream of the cache were bad compared to the entire path, the approach did not lead to savings and could introduce high additional delay.

The buffer at the cache could drop packets which were too late or too early, or send these packets to the receiver. In the latter case, it could delay the early packets, thus improving the stream which was sent to the receiver. This approach resulted in savings even if the buffer at the cache was releasing more jitter than the receiver could accept.

In our analysis and simulations, we have not assumed that the buffer size is sufficiently large to have zero overflow. The reason for that is that our buffer dimensioning works well only if the conditions of the path do not change significantly. We have assumed such a static network behavior; in practice, it is dynamic and the buffer would have to adapt its dimensions to the changing conditions. This has been proposed, for example, in [6]. However, the speed with which the buffer adapts its dimensions can have media-specific constraints. If that is the case, the buffer cannot be designed so

that no overflow will occur. We consider these issues a subject of future work.

If a dropping buffer is used, the receiver can dimension its buffer without knowing the amount of jitter released by the buffer at the cache. This may be an advantage from the point of view of protocol design, since the receiver does not need to be aware of the cache's existence: the protocol could be semantically transparent in this respect. On the other hand, if the buffer at the cache releases little jitter, there is little difference in the dimensions of a buffer at the receiver if the smoothing or dropping buffer is used.

A protocol which would be aware of buffering for reducing jitter and recovering lost packets should include features for negotiation of the desired playback quality and playback delay. An application could use the expression described in Section 3 to do this. An application could also choose whether to use a cache or receive the stream from the origin server. To do so, the application could join both the multicast group of a cache and of the origin server. After negotiating the playback delays and determining the buffer sizes required for the two streams the application can leave one of the multicast groups, thereby choosing the sender which best satisfies its requirements as to buffer size and playback delay.

Whether the receiver should use a cache or receive the stream directly from the sender depends on the cache's placement in the path from the sender to the receiver as well as on network conditions. The placement of a cache should take into account the number of receivers which are downstream of a network node and the conditions of the network downstream of the node where the cache is placed. For this purpose it may be better to model the network as a tree. We shall study the subject of placement in future work.

References

- [1] D. Ferrari. Distributed delay jitter control in packet-switching internetworks. Technical report, International Computer Science Institute, Berkeley, CA, 1991.
- [2] Thomas Kernen. www.traceroute.org. World Wide Web page.
- [3] Annie Gravey, Soren Blaabjerg, editor. *COST Monograph*. COST 242, December 1994.
- [4] C. Perkins, O. Hodson. Options for repair of streaming media. Request For Comments: 2354.

- [5] N. F. Maxemchuk, S. Lo. Measurement and interpretation of voice traffic on the internet. In *Proc. ICC*, 1997.
- [6] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proc. Infocom '94*, 1994.
- [7] R. Yavatkar, J. Griffioen, M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proc. ACM Multimedia '95*, 1995.
- [8] S. Sen, J. Rexford, D. Towsley. Proxy prefix caching for multimedia streams. In *Proc. Infocom '99*, pages 1310–1319, 1999.
- [9] W. A. Montgomery. Techniques for packet voice synchronization. *IEEE J.Sel.Areas Comm.*, SAC-1(6), December 1983.
- [10] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, 1997.
- [11] A. Wierzbicki. Applications of replication for transmission of streaming media over the internet. Technical report, Philips Research Laboratories, 2000.