ARTICLE IN PRESS

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Computer Communications xx (0000) xxx–xxx
computer communications
www.elsevier.com/locate/comcom

Design and analysis of optimal adaptive de-jitter buffers☆

Gagan L. Choudhury[a,*], Robert G. Cole[b,1]

[a]AT&T Labs, Room D5-3C21, Middletown, NJ 07748, USA
[b]AT&T Labs, 330 St Johns, 2nd Floor, Havre de Grace, MD 21078, USA

Received 8 August 2003; accepted 8 August 2003

Abstract

In order to transfer voice or some other application requiring real-time delivery over a packet network, we need a de-jitter buffer to eliminate delay jitters. An important design parameter is the depth of the de-jitter buffer since it influences two important parameters controlling voice quality, namely voice-path delay and packet loss probability. In this paper, we propose and study several schemes for optimally adjusting the depth of the de-jitter buffer. In addition to de-jitter-buffer depth adjustments within a call, the initial value and rates of changes of the de-jitter buffer depth are allowed to depend on the class of the call and are adaptively adjusted (upwards or downwards) for every new call based on voice-path delay and packet loss probability measurements over one or more previous calls. Parameter adjustments are geared towards either (a) minimizing voice-path delay while maintaining a packet loss probability objective, or (b) maximizing *R*-factor, an objective measure of voice quality that depends both on the voice-path delay and the packet loss probability. Using simulation models and measured packet delay traces, it is shown that adaptive schemes perform better than static ones and adaptive schemes with learning perform better than ones without learning.
© 2003 Published by Elsevier B.V.

Keywords: Adaptive de-jitter buffer algorithm with learning; Voice-call quality; End-to-end delay; Packet loss probability; Call classification

1. Introduction

A major challenge in transporting voice, video or more generally any application requiring real-time delivery over a packet network (using IP, ATM or some other packet-based protocol), is dealing with the delay jitter introduced by the packet network. Since real-time presentations cannot tolerate delay jitter, a de-jitter buffer needs to be used to eliminate it. In this paper, we will mainly consider voice calls although some of the work would also apply to a more general real-time delivery. An important design parameter is the depth of the de-jitter buffer since it influences two important parameters controlling voice quality, namely end-to-end voice-path delay and packet loss probability. The de-jitter-buffer depth is the maximum amount of time a packet spends in the de-jitter buffer before being played out. If it is too small, then many packets would miss the play-out deadline thereby increasing the packet loss probability. On the other hand, if it is too large, then the end-to-end voice-path delay would increase. The key challenge is to choose a de-jitter-buffer depth that is a happy middle ground between too much packet loss and too much voice-path delay. A second aspect is static versus adaptive adjustment of play-out instant. In a static scheme, the play-out instant is set once and for all at the arrival of the first packet of the call. In an adaptive scheme, the play-out instant may be shifted during the call based on the arrival instants of previous packets and thereby can improve the delay or packet loss behavior. However, each time the play-out instant is shifted, it is necessary to either inject silence or drop packets and thereby impact the voice call quality. For this reason, it may be preferable to use a static scheme in some cases since it truly eliminates the delay jitter. A compromise between a static and an adaptive approach is to adaptively compute an ideal play-out instant with the arrival of every packet but use a static play-out instant (thereby avoiding delay jitters) for most of the call. The static play-out instant is synchronized to the adaptive ideal play-out instant at a few selected points in the call thereby limiting the impact on voice-call quality. For calls with voice activity detection and silence suppression, the ideal

☆ Presented in part at SPIE's ITCOM-2002 Conference in July, 2002.
* Corresponding author. Tel.: +1-732-420-3721; fax: +1-732-368-1919.
E-mail addresses: gchoudhury@att.com (G.L. Choudhury), rgcole@att.com (R.G. Cole).
1 Tel./fax: +1-410-939-8732

0140-3664/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/j.comcom.2003.08.018

UNCORRECTED PROOF

ARTICLE IN PRESS

synchronization points are the beginning instants of every talk-spurt since that only involves shrinking or expanding the previous silence period slightly and has practically no impact on voice quality. In addition, we may also synchronize the play-out instant whenever the difference between the currently used play-out instant and the ideal play-out instant exceeds a certain threshold. This is the only type of synchronization possible for calls without silence suppression.

Techniques for delay adaptation for packetized voice have been studied in the literature for over two decades. Refs. [1–4] represent a few papers in this area but many more have been written. One common technique is to predict future delays based on past observations. Ref. [2] stores the actual delay distribution and Ref. [1] stores a statistical approximation to it based on previous packets of the same call. Some type of aging algorithm is used to give less importance to old samples. The de-jitter buffer depth is set such that a packet would be lost with a certain small probability assuming that its delay distribution would follow the same pattern as observed in the past (with aging). Refs. [3,4] set the de-jitter buffer depth to an estimated mean plus a few times the estimated delay variation and the estimates are slowly updated based on the observed delays of each newly arriving packet. Most algorithms allow the de-jitter-buffer depth to be affected only slowly by the actual delay of a newly arriving packet in order to eliminate random fluctuations of individual packet delays. However, there may be occasional delay spikes in the Internet and Ref. [3] allows more rapid change in de-jitter-buffer depths during those events. Ref. [5] is an example of using adaptive de-jitter-buffer mechanisms in an actual product.

Each de-jitter buffer adjustment algorithm is characterized by a set of parameters. Depending on the settings of these parameters and the voice packet stream on which the algorithm is applied, we will get a certain voice-path delay and a certain packet loss probability. In this paper, we propose the adaptive adjustment of the parameters of the de-jitter buffer based on voice-path delay and packet loss probability measurements over one or more previous calls of the same class. Call classification is based on its type (voice, fax, voice-band data, etc.), physical distance between transmitter and receiver, type of access/egress/backbone and terminal capability at each end of the call. All parameter adjustments are geared towards either (a) minimizing voice-path delay while maintaining a packet loss probability objective, or (b) minimizing packet loss probability while maintaining a voice-path delay objective, or (c) maximizing R-factor, an objective measure of voice quality that depends both on the voice-path delay and the packet loss probability [6–8]. To the best of our knowledge, no other previous work adaptively adjusts parameters of an adaptive de-jitter buffer algorithm with the objective of maximizing R-factor. Refs. [9,10] carefully identifies the various components of the R-factor including the de-jitter buffer delay and packet loss probability and shows various case studies. However, they do not provide a dynamic simulation study (similar to what we do) of a specific adaptive de-jitterization algorithm showing how the adaptive parameter adjustments would impact the R-factor. We provide numerical studies using voice packet streams based on actual measurements over the Internet and artificially generated ones assuming different degrees of Quality-of-Service (QoS) mechanisms at the access, egress, and backbone.

2. Objective measure of voice quality: the E-model and extension

The E-model, defined in the ITU-T Rec. G.107 [6] as well as other associated ITU-T recommendations [7], is an analytic model of voice quality used for network planning purposes. It calculates an R-factor which can be related to the Mean Opinion Score (MOS) as follows:

$$\text{For } R < 0: \text{MOS} = 1$$

$$\text{For } R > 100: \text{MOS} = 4.5 \tag{2.1}$$

$$\text{For } 0 < R < 100: \text{MOS} = 1 + 0.035R + 7\times 10^{-6}R(R-60)(100-R)$$

The MOS is a numerical measure of voice quality based on averaging the scores provided by many listeners where the scores 1, 2, 3, 4, and 5 imply 'bad', 'poor', 'fair', 'good', and 'excellent' ratings, respectively. The R-factor depends on several aspects of the voice call. If we choose default values for all parameters other than the ones that depend on the end-to-end voice-path delay and packet loss probability, then we get

$$R = 94.2 - I_d - I_{ef} \tag{2.2}$$

where, I_d and I_{ef} refer to impairment factors associated with delay and packet loss probability, respectively. Ref. [8] gives the following simplified expression for the delay-impairment factor

$$I_d = 0.024d + 0.11(d - 177.3)\ H(d - 177.3) \tag{2.3}$$

where, d is the one-way mouth-to-ear delay in milliseconds (ms). In the standard E-model, the delay d is constant throughout the call. However, in our adaptive de-jitter-buffer algorithms, d changes during the call. In such a situation, we assume the average d value during the call. Also, there is some jitter in the adaptive scheme while impact of jitter is not there in the standard E-model. This is not a significant problem since, as mentioned in the introduction, we basically use a static algorithm throughout the lifetime of the call. However, we keep track of the ideal dynamic algorithm and synchronize the static algorithm with the dynamic one only at the beginning of talk-spurt which does not cause any jitter or very rarely otherwise

UNCORRECTED PROOF

which may cause some jitter but only rarely. A mathematical expression for the factor I_{ef} is not given directly in the E-model but [8] has obtained such expressions. The general form of the expression is

$$I_{ef} = \gamma_1 + \gamma_2 \ln(1 + \gamma_3 e) \tag{2.4}$$

where, e is the total packet loss probability and γ_is are constants that depend on the type of Codec used. Two example cases are given below (see Ref. [8] for more details)

$$I_{ef}(G.729a, \text{random}) = 11 + 40 \ln(1 + 10e) \tag{2.5}$$

$$I_{ef}(G.711\text{concealment}, \text{random}) = 0 + 30 \ln(1 + 15e) \tag{2.6}$$

3. Voice packet stream generation

We observed packet delay jitters by sending a large number of packets periodically over the Internet between locations in New Jersey and Maryland (about 150 miles apart) with Cable Modem access on one side (about 0.5 Mbps upstream and about 2 Mbps downstream) and Fractional DS-3 ATM access on the other side (about 7 Mbps each way). The period chosen was 20 ms, typical packetization interval used in IP telephony, and packets were sent in bursts of 100 (i.e. total burst duration is 2 s) with a gap of 1 min between the start of successive bursts. Since receiver and transmitter clocks were not synchronized (the usual situation), the relative delays within each burst were computed with respect to the minimum-delay packet in the burst. Since we need absolute values of one-way delays for the E-model, a constant value was assumed for the minimum-delay packet within each burst based on the measurements we created several streams. Each stream had between 16,000 and 22,000 packets and for a given simulation we start at a certain offset point of the stream and once we reach the end of the stream we start again at the beginning (different simulations on the same stream differ in the starting offset).

In addition to measured packet delays, we also generated artificial packet delays using analytic models of delays experienced over several access and backbone links where voice packets co-exist with data packets. Specifically, we assumed two access links each at 256 Kbps and six backbone links each at 45 Mbps. Each link was modeled independently. In addition to the voice packets being transferred (once in 20 ms during a talk-spurt), each link also had background traffic which contributed to the majority of the bandwidth usage. The background traffic was assumed to consist of big packets corresponding to file transfers and small packets corresponding to control, interactive, query/response messages and voice packets from other sources. Big packets accounted for 80% of the bandwidth being used and the rest were from small packets. On the backbone links, big packets (including protocol overhead) were assumed to be 1500 bytes and on the access links they were assumed to be 200 bytes (limited by fragmentation). The small packets were assumed to be exponentially distributed with an average of 100 bytes including protocol overhead.

Two types of models were assumed, one with no QoS differentiation among the traffic types and the other with QoS differentiation. In the case with no QoS differentiation, the small packets were assumed to arrive according to a Poisson process, the big packets were assumed to come from five identical On–Off sources and their superposition was modeled as a Markov Modulated Poisson process. The total link bandwidth was assumed to be 60% and the Laplace–Stieltjes Transform (LST) of the delay distribution experienced by a voice packet was obtained using the AT&T Tool Q-SQUARED [11]. In the case with QoS differentiation, it was assumed that the small packets (including voice) have non-preemptive priority over the big packets, the total link utilization was assumed to be 80% and the LST of the delay distribution experienced by a voice packet was obtained using standard results for M/G/1 priority queueing model [12]. The LST of the end-to-end delay distribution was obtained by taking the product of the LSTs over each link and then the delay distribution was obtained through Laplace Transform inversion [13].

In the simulation studies to be presented later, we use three streams. Stream 1 is based on measurements and Streams 2 and 3 are generated based on the analytic model. For both Streams 2 and 3, the access links are assumed to have QoS differentiation. The difference between the two streams is that Stream 2 assumes no QoS differentiation on the backbone links but Stream 3 does assume QoS differentiation on the backbone links.

4. De-jitter-buffer algorithms

The successive voice packets are transmitted strictly periodically with a period equal to the packetization interval. Let $D_{i,\text{net}}$, $D_{i,\text{buf}}$ and D_i represent the network delay, de-jitter-buffer delay and end-to-end delay, respectively, experienced by the ith packet. All delays are one-way delays. Also, throughout this paper we will use ms as the unit of time. Clearly,

$$D_i = D_{i,\text{net}} + D_{i,\text{buf}} \tag{4.1}$$

Usually it is not possible to accurately estimate the one-way network delay without requiring elaborate synchronization procedure between the transmitter and the receiver. We assume that no such procedure is available, but we can accurately obtain the relative network delay among packets as explained below. Specifically, if d_p represents the packetization delay and t_i, t_j represent the arrival instants at the de-jitter buffer of packets i and j, respectively ($j > i$), then due to the strict periodic nature of packet transmission, we get

$$D_{j,\text{net}} - D_{i,\text{net}} = t_j - t_i - (j - i)d_p \tag{4.2}$$

UNCORRECTED PROOF

$t_j - t_i$ may be obtained accurately at the receiver without requiring any synchronization with transmitter. For our algorithms, we only need relative delays among packets except in one case (first paragraph in Section 4.2) where we need an approximate estimate of the absolute one-way delay in order to slowly adapt upwards the arrival instant of the minimum-delay packet following an internet route change that increases the end-to-end propagation delay. If a synchronization procedure exists between transmitter and receiver in order to accurately estimate the absolute one-way delay then that may be used. If not then an estimate may be obtained by taking the minimum of several round-trip delay measurements and dividing it by two. This would be somewhat inaccurate in case of asymmetry of routes in the two directions but that should be OK since we only need an approximate estimate. Furthermore, the algorithms should also work without this adaptation of minimum-delay packet in which case no estimate of absolute one-way delay would be needed.

4.1. Static algorithm

The end-to-end delay D_i is set to a constant D for all packets. The only thing we can choose is the de-jitter-buffer delay, $D_{1,\mathrm{buf}}$, for the first packet. After the play-out of the first packet, each successive packet is played out strictly periodically unless of course the packet is to be dropped for late or early arrivals (explained below). This implies that the end-to-end delay of every packet (that is not dropped) is given by

$$D_i = D = D_1 = D_{1,\mathrm{net}} + D_{1,\mathrm{buf}} \tag{4.1.1}$$

For the ith packet ($i > 1$), if $D_{i,\mathrm{net}} > D$ then it is a late packet and is dropped. The amount of time the ith packet stays in the de-jitter buffer (provided it is not late) is $D_{i,\mathrm{buf}} = D - D_{i,\mathrm{net}}$. If this time is too long, then the physical holding capacity of the buffer may be exceeded requiring the packet to be dropped for being too early. In all simulation results in this paper, we assume that the physical holding capacity of the buffer is large enough so that no packet is dropped for early arrival.

In Figs. 1–4 below, we show the performance of the static algorithm as a function of the only tunable parameter, $D_{1,\mathrm{buf}}$, the de-jitter-buffer depth set for the first packet. The de-jitter-buffer depth is in ms. This is the convention we will use throughout the paper. In all cases, however, the de-jitter-buffer depth in number of packets may be obtained by dividing this quantity by the packetization interval (used as 20 ms in this paper). All simulations are on 5-min voice calls (unless specified otherwise) with average talk-spurt intervals of mean 0.5 s and silence intervals of mean 1 s, each having an exponential distribution (Fig. 9 in Section 4.2 is an exception where no silence suppression is used). All simulations are repeated independently 40 times and usually the average result is shown (in some cases the variation of packet loss probability among the 40 runs is shown). Figs. 1 and 2 show that the same de-jitter buffer depth may give significantly different delay and packet loss probability depending on the type of stream. Fig. 3 shows that even within the same stream (Stream 1), there is a significant difference between the minimum and the maximum of the 40 simulation runs. This is mainly because the performance of the static scheme depends on

UNCORRECTED PROOF

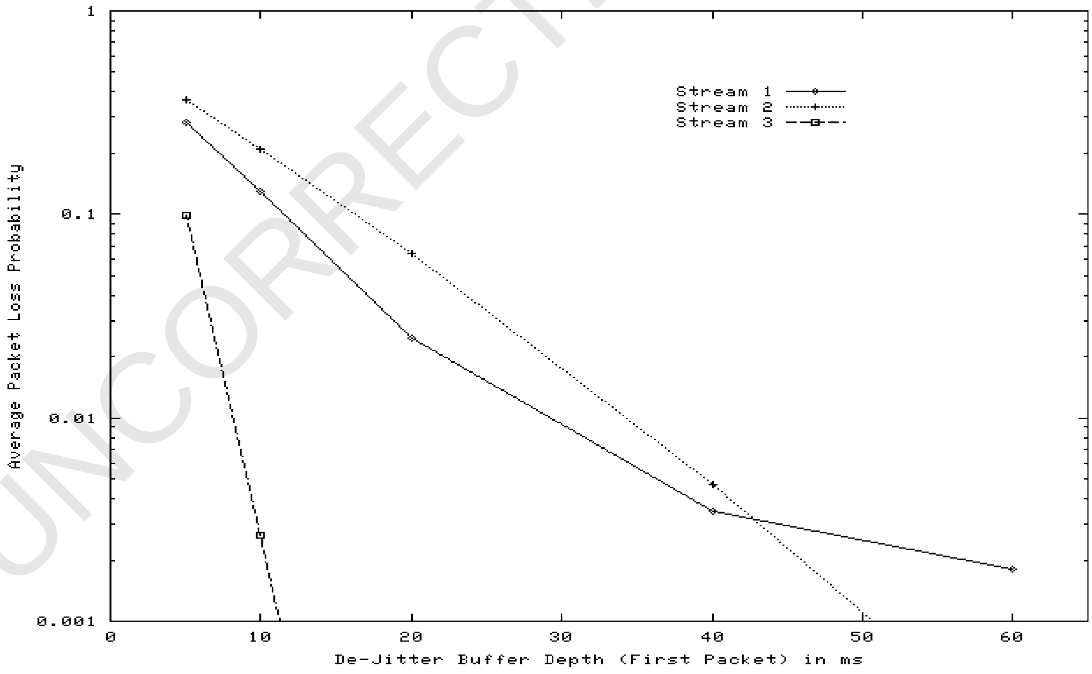

Fig. 1. Packet loss probability, static scheme.

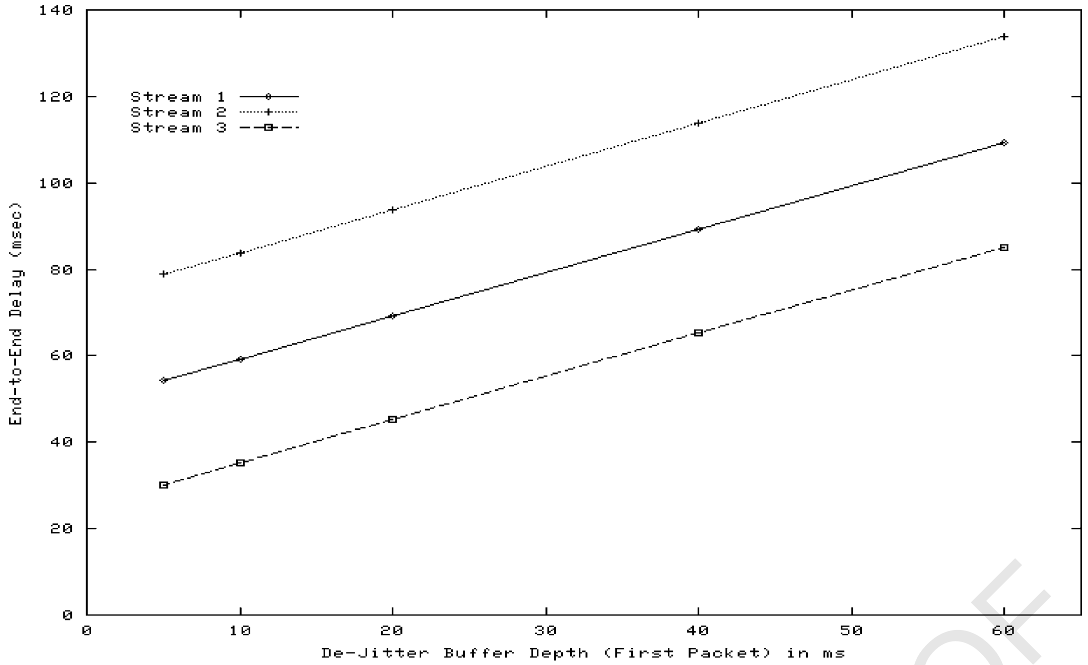

Fig. 2. Delay, static scheme.

the actual delay of the first packet, which may vary significantly. Fig. 4 shows that the packet loss probability is increased significantly if there is a sudden delay change of 20 ms at the halfway point of the voice call (e.g. due to change in propagation delay resulting from failure in the packet path and subsequent reroute over a longer path).

4.2. Adaptive algorithm

In this algorithm, the end-to-end delay D_i (for $i > 1$) is allowed to adapt based on the observed delays of previous packets (unlike the static scheme where $D_i = D_1$ for all i). It is assumed that at the instant, a play-out decision is needed to be made for the play-out of the ith packet,

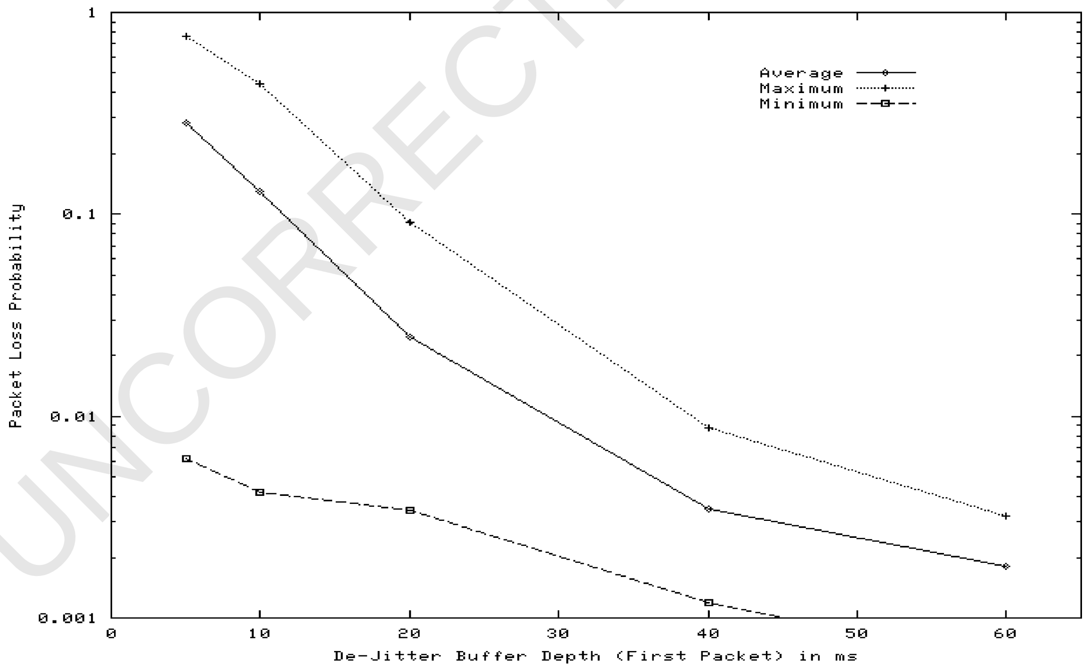

Fig. 3. Stream 1, variation of packet loss probability among 40 simulation runs.

UNCORRECTED PROOF

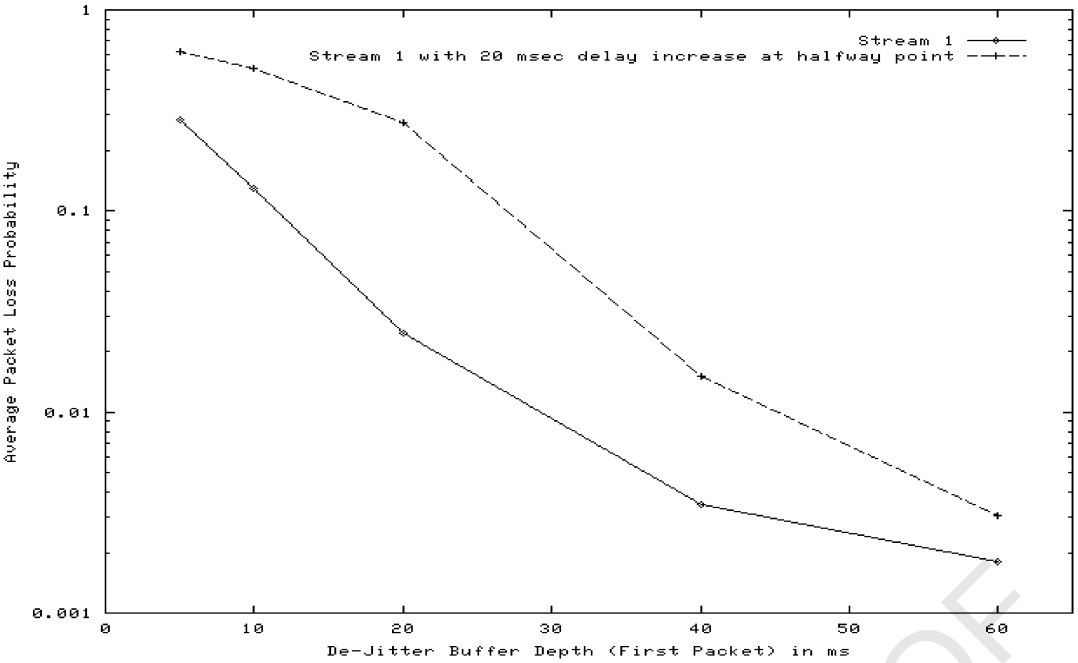

Fig. 4. Static scheme, impact of sudden delay increase on packet loss probability.

the arrival instants of all previous packets are known (if some previous packet has not shown up yet, its delay is assumed to be infinity. Such a packet will be a late packet and will be dropped any way once it shows up). At all times, we estimate a minimum-delay packet and a buffer-depth which is defined as the delay experienced by the minimum-delay packet in the De-jitter buffer. For $i > 1$, let $D_{i,\min}$ represent the network delay of the minimum-delay packet when play-out decision is made for the ith packet. Then

$$D_{i,\min} = \underset{j}{\mathrm{Min}}\,\{D_{j,\mathrm{net}}\}_{j=1,\ldots,i-1} \qquad (4.2.1)$$

In order to slowly adapt the minimum-delay packet upwards in case there is a constant upward shift in network delay (e.g. due to change in network route caused by a failure) we change the minimum-delay calculation slightly compared to Eq. (4.2.1) as given below:

- If $(D_{j,\mathrm{net}} < D_{j,\min})$, then $D_{j+1,\min} \equiv D_{j,\mathrm{net}}$,
- else, $D_{j+1,\min} \equiv \mathrm{Min}(D_{j,\min} + D_\varepsilon, D_{j,\mathrm{net}})$

Where D_ε is a small increment. We set it to $D_\varepsilon = \alpha D_{j,\min}$ and for $D_{j,\min}$ use a rough estimate of one-way delay as explained in Section 4 (note that since D_ε is very small, any error in estimating $D_{j,\min}$ would not have any significant impact on the overall algorithm. We can also set $\alpha = 0$ in which case no upward adjustment of minimum-delay packet would be made and the algorithm would be completely free of absolute one-way delay estimates).

Let B_i represent the buffer depth (in ms) to be used for the ith packet. At the instant play-out decision is to be made for the ith packet, B_{i-1} is available which is given by

$$B_{i-1} = D_{i-1} - D_{i,\min}, \qquad \text{for } i > 1 \qquad (4.2.2)$$

For the ith packet, the relative network delay $D_{i,\mathrm{rel}}$, measured with respect to the minimum-delay packet is given by

$$D_{i,\mathrm{rel}} = D_{i,\mathrm{net}} - D_{i,\min} \qquad (4.2.3)$$

We adjust the buffer depth based on how the relative delay compares to the existing buffer depth. We compute a delay ratio

$$D_{i,\mathrm{ratio}} = \frac{D_{i,\mathrm{rel}}}{B_{i-1}} \qquad (4.2.4)$$

Note that if the $D_{i,\mathrm{rel}}$ exceeds the buffer depth, then the packet is late and dropped and such an event should happen only with low probability. Therefore, if $D_{i,\mathrm{ratio}}$ is near 1 or exceeds it, we should significantly increase the buffer depth. On the other hand, if $D_{i,\mathrm{ratio}}$ is close to zero, we should decrease the buffer depth. However, due to the random nature of delay jitter and our unwillingness to accept high packet loss, the rate of decrease should be slow. In general, we adapt the buffer depth as follows

$$B_i' = B_{i-1}(1 + f) \qquad (4.2.5)$$

UNCORRECTED PROOF

ARTICLE IN PRESS

where B_i' is the initial estimate for B_i and the factor f is chosen based on the value of $D_{i,\text{ratio}}$ as follows:

$$\left\{\begin{array}{l} \text{if, } D_{i,\text{ratio}} > R_1 \text{ then } f = f_1 \\ \text{else if, } R_2 < D_{i,\text{ratio}} \leq R_1 \text{ then } f = f_2 \\ \vdots \\ \text{else if, } R_n < D_{i,\text{ratio}} \leq R_{n-1} \text{ then } f = f_n \\ \text{else if, } D_{i,\text{ratio}} \leq R_n \text{ then } f = f_{n+1} \end{array}\right\} \tag{4.2.6}$$

Note that there are n threshold parameters $\{R_1, R_2, \ldots, R_n\}$ and $n+1$ rate-change parameters $\{f_1, f_2, \ldots, f_{n+1}\}$. The rate-change parameters may be positive or negative (typically, the first few would be positive and the last few would be negative). The end-to-end delay D_i for the ith packet is obtained as

$$D_i = D_{i,\min} + B_i' \tag{4.2.7}$$

The ith packet will be played out only if it is not late, i.e. $D_{i,\text{net}} \leq D_i$. Next we obtain $D_{i+1,\min}$, the network delay for minimum-delay packet to be used for the $(i+1)$th packet. Since the minimum-delay reference is changed, the buffer-depth $B_{d,i}$ has to be re-adjusted as follows:

$$B_i = B_i' + D_{i,\min} - D_{i+1,\min} \tag{4.2.8}$$

As mentioned in the introduction, the adaptive adjustment of play-out instant for every packet as described above is done only for an ideal de-jitter buffer and the real de-jitter buffer is run statically for most of the lifetime of the call except that the real de-jitter buffer is synchronized to the ideal one at the beginning of every talk-spurt and if the ideal de-jitter-buffer depth differs from that of the real one by more than $X\%$ where X is a tuning parameter. We repeated the simulations shown earlier with the adaptive scheme with the following settings of the adaptive de-jitter buffer parameters:

- $n = 3$, $R_1 = 1$, $R_2 = 0.75$, $R_3 = 0.5$, $f_1 = 0.25$, $f_2 = 0.015625$, $f_3 = 0$, $f_4 =$ variable (negative), $\alpha = 0.004$, and $X = 25\%$. The initial de-jitter buffer depth is set at 60 ms and it is never allowed to go over 200 ms or below 4 ms. The upper bound of 200 ms was chosen because above this, conversational voice quality begins to degrade dramatically [6–8].

Figs. 5–9 show the performance as a function of the tuning parameter f_4. Figs. 5 and 6 show (as compared to Figs. 1 and 3) that the variation among streams and variation among the minimum and maximum observed over the 40 runs is less in the adaptive scheme compared to the static scheme (note that log scales were used in Figs. 1 and 3 compared to linear scales in Figs. 5 and 6). Using a large de-jitter-buffer depth (e.g. around 60 ms) in the static scheme may allow it to have a lower packet loss probability compared to the adaptive scheme (with sufficiently high $|f_4|$). However, in such situations, the adaptive scheme produces significantly lower average delay compared to the static scheme. We did verify with several examples that with about the same average delay, the adaptive scheme produces lower packet loss probability compared to the static scheme. Fig. 7 (compared to Fig. 4) shows that performance degradation as a result of delay increase at the halfway point of the call is very small in the adaptive scheme compared to the static scheme. Fig. 8 shows the degradation

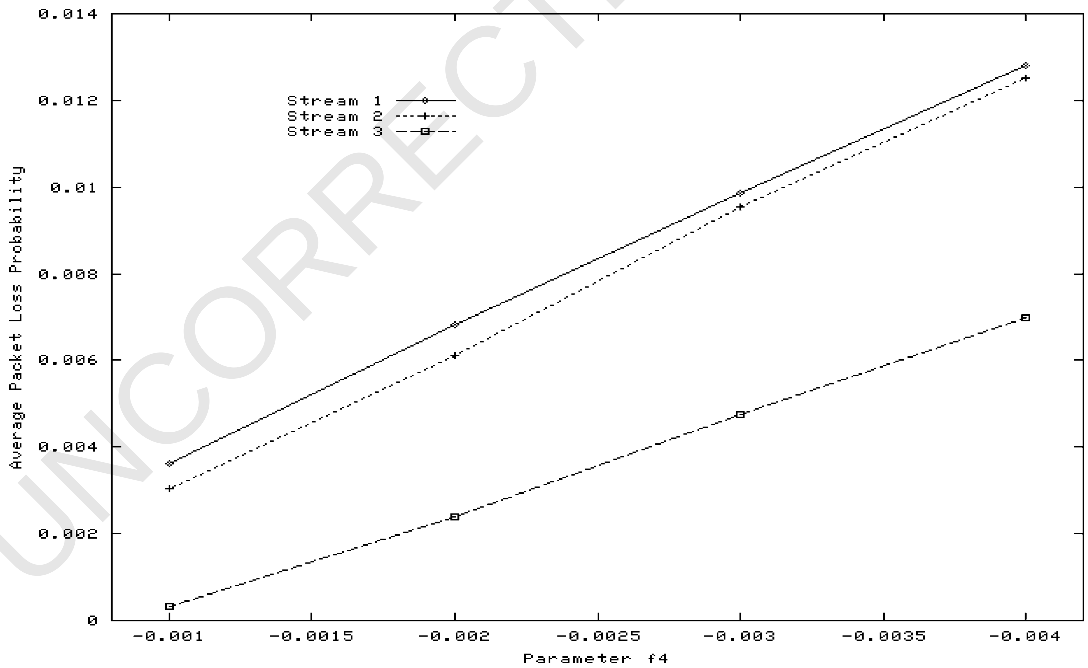

Fig. 5. Packet loss probability, adaptive scheme.

UNCORRECTED PROOF

ARTICLE IN PRESS

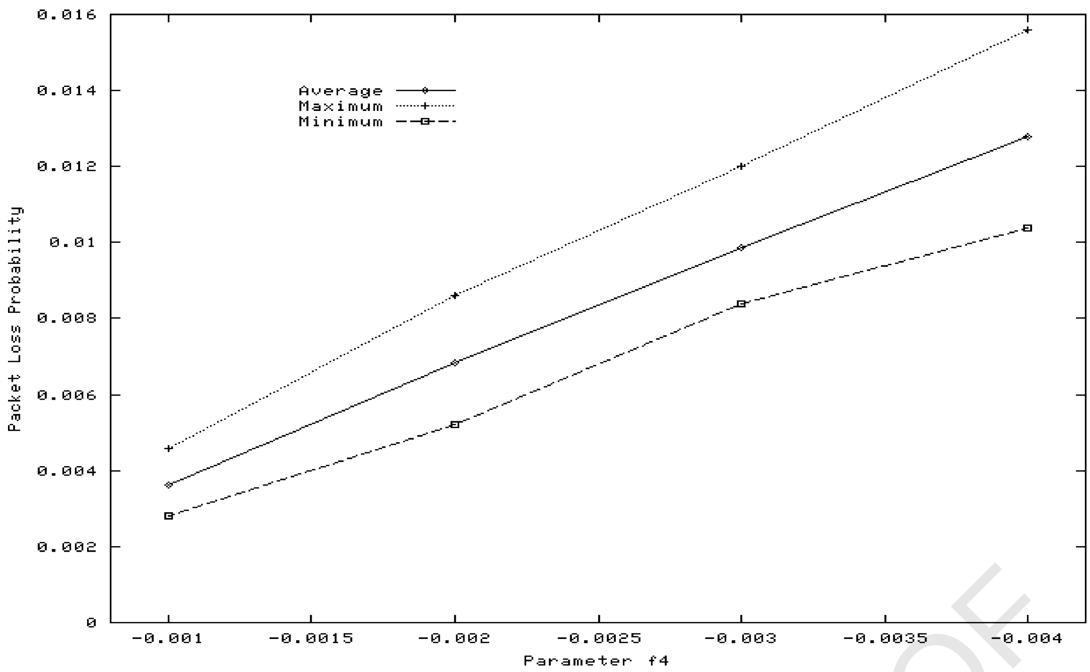

Fig. 6. Stream 1, adaptive scheme, variation among 40 simulation runs.

in packet loss probability if the adjustment of play-out instant is done only at the talk-spurt beginning (i.e. X is set to infinity). Fig. 9 shows the degradation in performance (with the same number of packets and using Stream 1) if no silence suppression is used and synchronization between the static and the ideal adaptive scheme is done only when the ideal de-jitter-buffer depth differs from the real one by more than $X(=25)\%$.

4.3. Adaptive algorithm with learning

The adaptive scheme of Section 4.2 always used the same set of parameters. In this section, we allow the scheme to learn from previous calls of the same type and accordingly adjust its parameters. Fig. 10 shows the mean, 90th, 95th and 99th Percentiles of the end-to-end delay as a function of call duration using Stream 1 and no learning. For

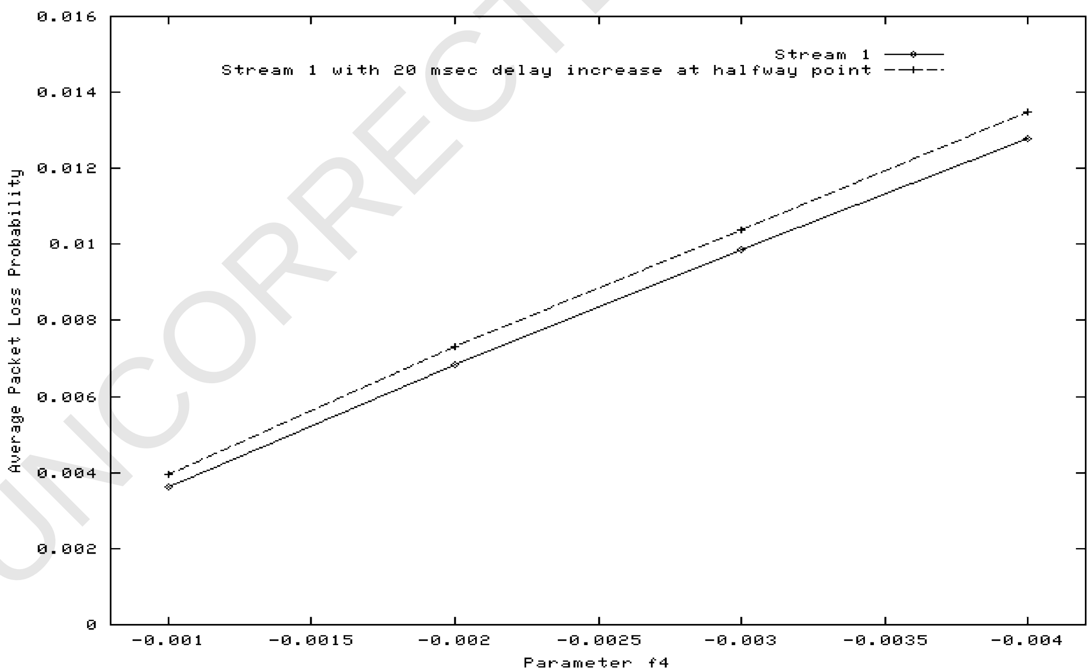

Fig. 7. Adaptive scheme, impact of sudden delay increase.

UNCORRECTED PROOF

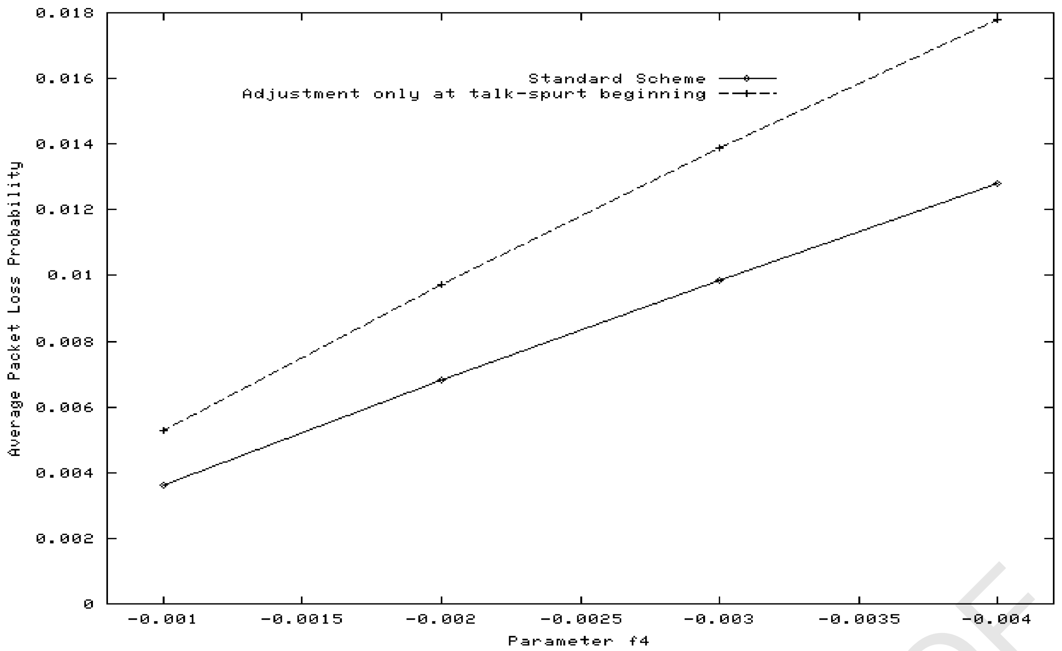

Fig. 8. Adaptive scheme, impact of only adjusting at the beginning of talk-spurts.

good performance, all these parameters should be close to each other. We observe this to be the case for long calls but for calls of short duration, wide variation in end-to-end delay occurs. In Fig. 11, we allow a simple learning in which the initial de-jitter-buffer depth is set to what was observed at the end of the last call and we see that with this change even for 1-min calls, the second and subsequent calls show much better delay performance. In Fig. 12, we allow a different type of learning. Our goal is to adapt to a target packet loss probability irrespective of the type of voice stream. If the packet loss probability on the previous call of the same type differs from the target by $Y\%$ then we adjust the parameter f_4 by $\beta Y\%$ in the proper direction (the direction is known based on study in Section 4.2). Fig. 12 shows the results of this learning-based adjustment with $\beta = 0.33$. Note that for the first call, packet loss probability is quite far from the target and is different for different streams but by the fifth call, both streams give a packet loss probability quite close to the target. Instead of waiting for several calls to approach the target, we can also store the arrival instants of all packets in the first call, do a real-time simulation on this call to get the parameter value that would

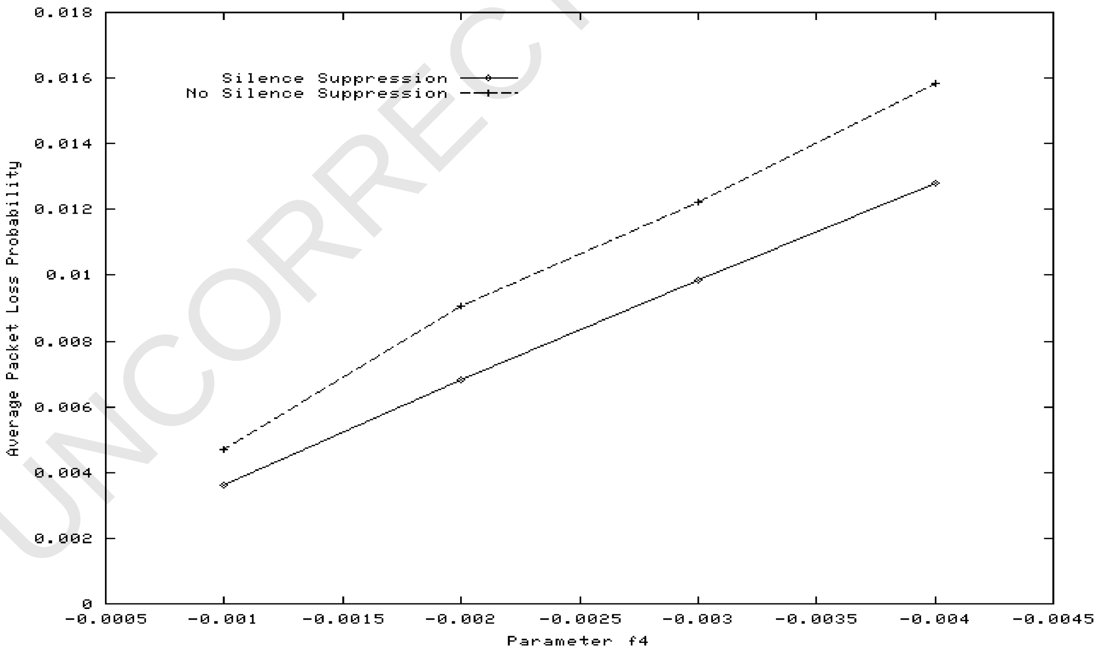

Fig. 9. Impact of not doing silence suppression.

UNCORRECTED PROOF

ARTICLE IN PRESS

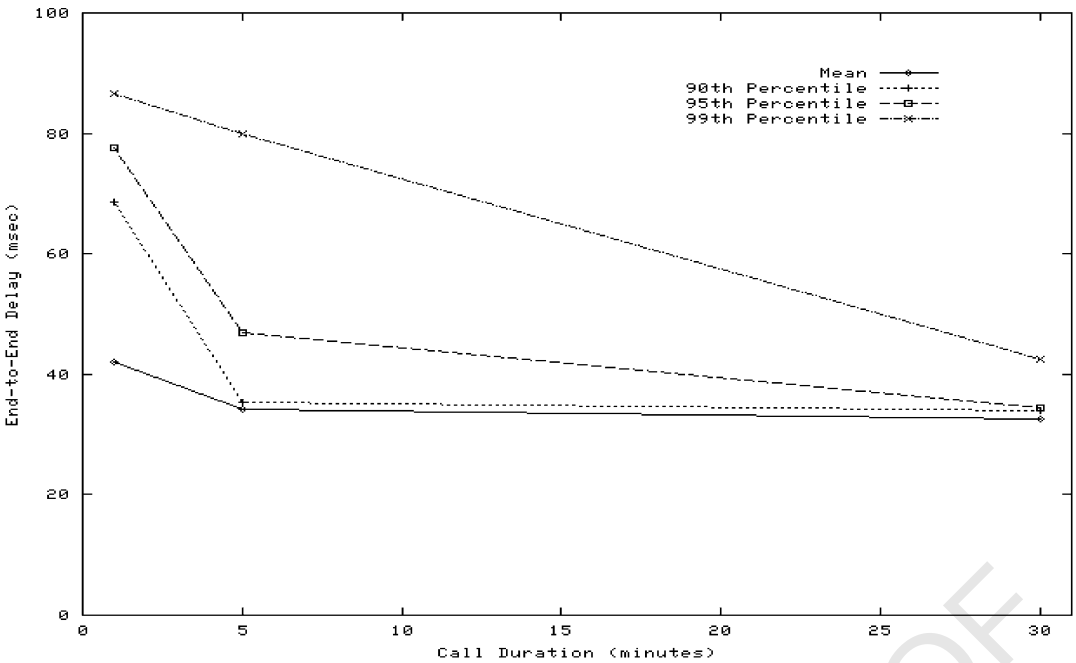

Fig. 10. Delay variation in adaptive scheme with no learning.

force the packet loss probability to be close to the target and then use that parameter value on the next call (note that the actual packet arrival instants on the next call are quite different from that of the previous one but statistically they are similar). Using this approach we observed that we can get quite close to the target even on the second call. Furthermore, each simulation typically takes seconds in a PC and so its results would be available by the time the next call comes in. Instead of storing packet arrival instants, it is also possible to run parallel realizations of the algorithm (requiring no storage and running in parallel in real-time) with different values of the control parameter and choose the best value for the subsequent call.

Fig. 13 shows a learning-based adjustment to improve the *R*-factor (and thereby voice call quality, see Eqs. (2.1)–(2.6)). For the first call, we compute the *R*-factor (as mentioned in Section 2, we use the average delay during the call in order to compute the delay term in

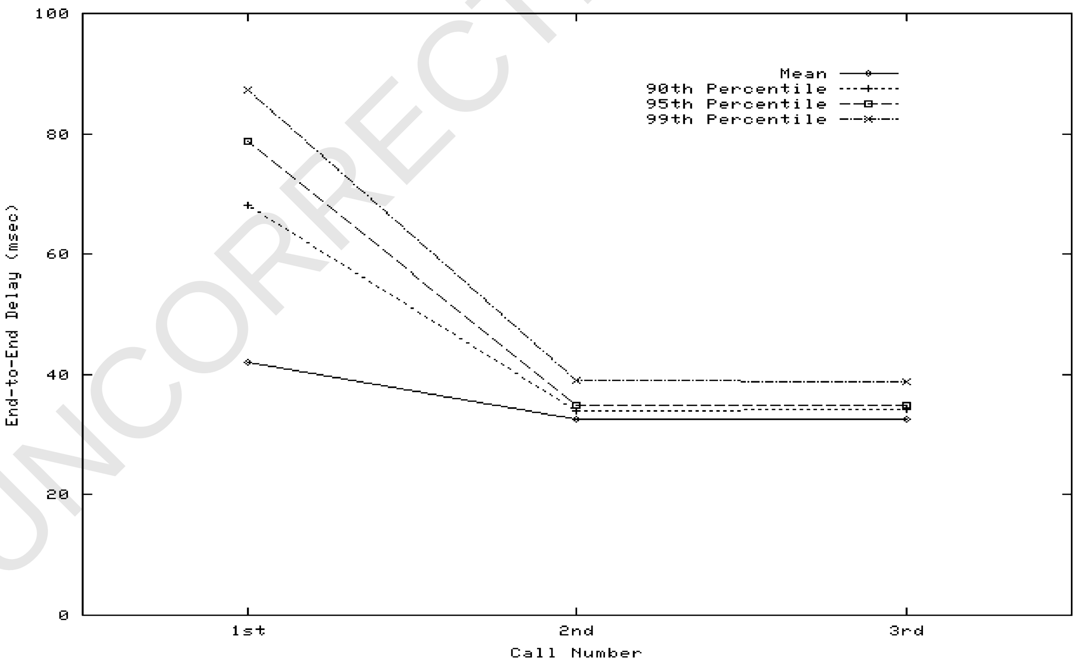

Fig. 11. 1 min Calls, delay variation improvement in adaptive scheme with learning.

UNCORRECTED PROOF

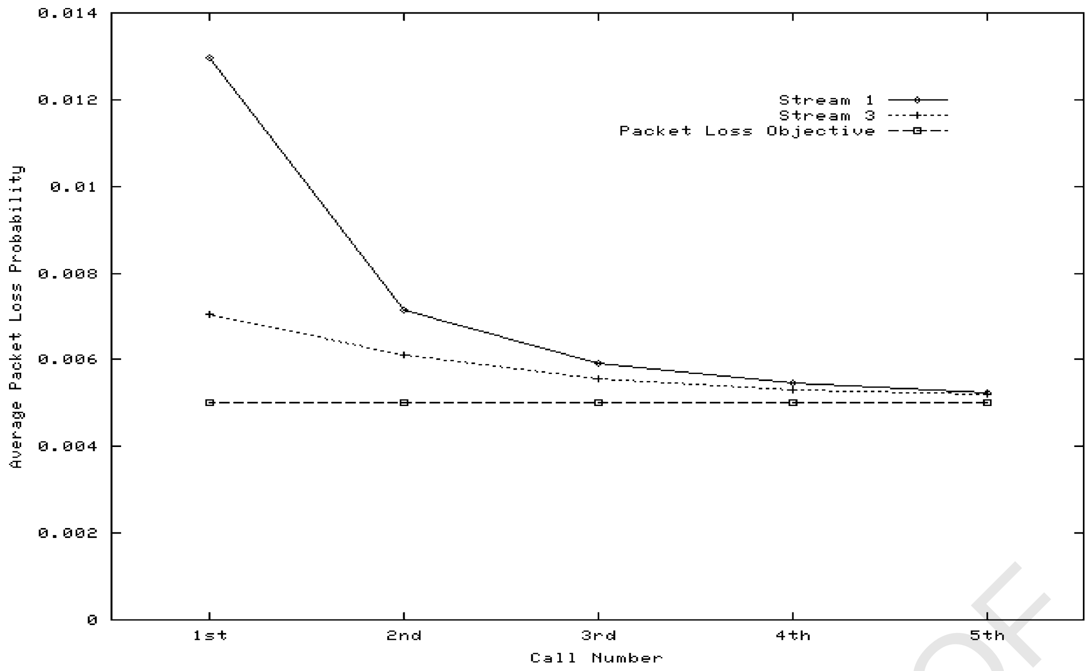

Fig. 12. Adaptive scheme with learning to approach a packet loss target.

the *R*-factor. Also, since we essentially use a static scheme throughout the lifetime of the call with delay adjustments mainly at the beginning of talk-spurts, the jitter introduced is minimal and the fact that it is not explicitly taken into account in the *R*-factor is not a significant problem). We change the parameter f_4 in a certain direction for the next call and note the change in *R*-factor. If it increases, then we keep changing f_4 in the same direction and otherwise change it in the opposite direction. The magnitude of change is dampened by a certain factor. In Fig. 13, the initial change was 0.001 and dampening factor was 0.8. Note from Fig. 13 that the *R*-factor for both streams increase with successive calls and tend to settle towards a maximum value which depends on the voice packet stream. As in the case of Fig. 12, we also found that with repeated real-time simulation on the first call it is possible to get most of the improvement in the *R*-factor by just the second call.

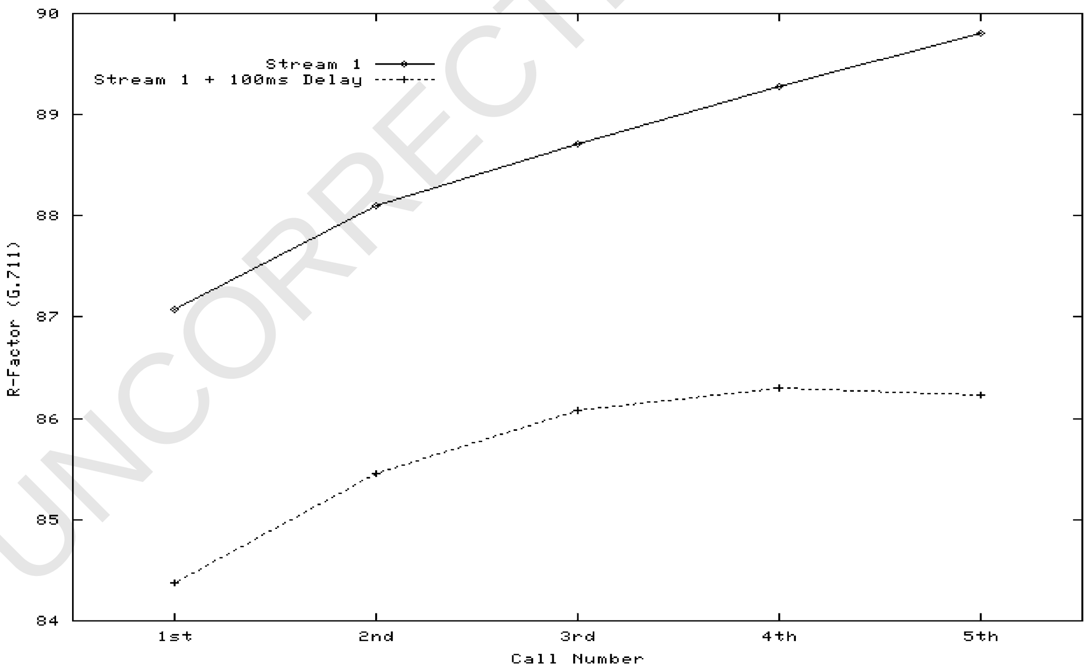

Fig. 13. Adaptive scheme with learning to improve *R*-factor.

UNCORRECTED PROOF

ARTICLE IN PRESS

5. Conclusions

We use a simulation model and measured, as well as artificially generated, packet traces to study the delay and packet loss performance of various de-jitter buffer algorithms used to eliminate delay jitters for voice and real-time presentations transported over a packet network with variable delays. We show that with a static algorithm, there may be significant variation of packet loss probability and end-to-end delay even with the same choice of de-jitter buffer depth for the first packet, the only controllable parameter in static schemes. Furthermore, the packet loss probability is significantly worsened with a delay increase in the middle of the call, potentially caused by a failure in the packet path and subsequent reroute over a longer path. In an adaptive algorithm, the variation of packet loss probability and end-to-end delay is smaller compared to that in the static scheme, and due to its adaptive nature, the packet loss probability is not impacted significantly by a delay increase in the middle of the call. Adaptive schemes with learning from previous calls of the same class (classification based on physical distance between transmitter and receiver, type of access/egress/backbone/encoding, terminal capability, etc.) can perform better compared to adaptive schemes without such learning. Delay variation within a call due to adaptation may be large for short calls but can be significantly reduced by a simple learning whereby the initial de-jitter-buffer depth is set to what was observed at the end of the last call of the same class. It is also possible to approach a packet loss probability target, approach a delay target, or maximize the *R*-factor, an objective measure of voice call quality, through learning from previous calls of the same type and change a parameter value of the adaptive de-jitter-buffer algorithm based on this learning. Even though the numerical studies show the impact of changing one parameter value, it is possible to change more than one parameters in sequence or simultaneously. The change may be a slow adjustment with each new call resulting in a slow march towards the desired target over many calls. Alternatively, it is possible to do multiple real-time simulations with many different sets of parameter values over the same call and thereby make significant changes in parameter values and approach the desired target much faster.

References

[1] C. Sreenan, J. Chen, P. Agrawal, B. Narendran, Delay reduction techniques for playout buffering, IEEE Trans. Multimedia 2 (2) (2000) 88–100.

[2] S.B. Moon, J. Kurose, D. Towsley, Packet audio playout delay adjustment: performance bounds and algorithms, ACM/Springer Multimedia Syst. J. 6 (1998) 17–28.

[3] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, Proceedings of the IEEE Infocom, Totonto, Canada, June 1994, pp. 680–688.

[4] V. Jacobson, Congestion avoidance and control, Proceedings of the ACM SIGGCOMM, August 1988, pp. 314–329.

[5] Playout delay enhancements for voice over IP, Cisco IOS Documentation, http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t5/dt_pod.htm.

[6] ITU-T Recommendation G.107, The E-Model, a Computational Model for Use in Transmission Planning, December 1998.

[7] ITU-T Recommendation G.113, General Characteristics of General Telephone Connections and Telephone Circuits—Transmission Impairments, February 1996.

[8] R.G. Cole, J.H. Rosenbluth, Voice over IP performance monitoring, ACM J. Comput. Commun. Rev. 31 (2) (2001).

[9] A. Van Moffaert, D. De Vleeschauwer, M.J.C. Buchli, J. Janssen, G.H. Petit, P. Coppens, Tuning the VoIP gateways to transport international voice calls over a best effort IP backbone, Proceedings of the Ninth IFIP Conference on Performance Modeling and Evaluation of ATM and IP Networks (IFIP9), Budapest, Hungary, 27–29 June, 2001, pp. 193–205.

[10] J. Janssen, D. De Vleeschauwer, M.J.C. Buchli, G.H. Petit, Assessing voice quality in packet-based telephony, IEEE Internet Comput. Spec. Issue Internet Telephony 6 (3) (2002) 48–57.

[11] G.L. Choudhury, W. Whitt, Q-squared: a new performance analysis tool exploiting numerical transform inversion, Demonstration at the 15th ITC, Washington, DC, USA, June 23–27, 1997.

[12] L. Kleinrock, Queueing Systems, vol. 2, Wiley, New York, 1976.

[13] J. Abate, G.L. Choudhury, W. Whitt, An introduction to numerical transform inversion and its application to probability models, in: W. Grassman (Ed.), Computational Probability, Kluwer, Boston, 1999, pp. 257–323.

UNCORRECTED PROOF