A Self-tuned Quality-Aware De-jittering Buffer Scheme of Packetized Voice Conversations

Sofiene JELASSI, Habib YOUSSEF
Research Unit PRINCE, ISITCom
Hammam Sousse, University of Sousse, Tunisia
Sofiene.Jelassi@infcom.rnu.tn, habib.youssef@fsm.rnu.tn

Hugh MELVIN
Department of Information Technology
National University of Ireland, Galway, Ireland
hugh.melvin@nuigalway.ie

Abstract

Transport IP-based networks convey data packets toward their destination with variable one-way network delays due to their best-effort feature. Network delay variation of media packets, known also as *network delay jitter*, crucially disturbs the perceptual quality of packetized delay-sensitive services such as VoIP, IPTV, and video conferencing. The media-playing entity should seamlessly hide the perceptual effects due to network delay jitter. Such a mechanism is often referred to as *de-jittering scheme*, which should optimize the lateness-loss / total delay trade-off.

In this paper, we describe a novel de-jittering algorithm for VoIP packet streams. The algorithm accommodates a wide range of network delay jitters. This is achieved with the help of *three self-tuned first-order filters* of experienced one-way network delays and delay variance. Following a new end-to-end delay adjustment event, the de-jittering algorithm tunes for the *safety factor*, which is used to calibrate the variance around the estimated average network delay, in order to optimize the perceptual quality for a given network condition. The calibration process is performed using the recent history of estimated one-way network delays. This enables to predict with high accuracy the optimal end-to-end delay which will be used till the next adjustment event. Simulation results of VoIPoW (VoIP over wireless) show that our de-jittering algorithm significantly outperforms the baseline de-jittering algorithm [5] in terms of instantaneous and overall perceptual quality.

Keywords: VoIP, Network delay jitter, de-jittering buffer schemes, perceptual quality

1. Introduction

The integration of *delay-sensitive* services such as VoIP and IPTV over transport IP-based networks requires the development of suitable protocols, architectures, and QoS control algorithms. This class of services needs the reception of each sent *media unit* before its expected playback instant, but may tolerate some packet losses. However, *unmanaged* data IP networks have been designed to deliver reliably in-sequence delay-insensitive *elastic* traffic such as file, web, and mail transfer. A significant effort has been made within standardization bodies, academic institutions, and industry to integrate *packetized* delay-sensitive services over data networks. For instance, the IETF (Internet Engineering Task Force) has defined a set of protocols to accommodate multimedia services over Internet such as RTP, RTCP, SIP, and SDP [1]. Other important effort has been made by the ITU (International Telecommunication Union) and the ETSI (European Telecommunication Standardization Institute) organization bodies [2, 3].

Packetized Multimedia (including Voice) over IP service increasingly replaces and extends circuit-switched telephone service offered by Telecom providers in homes and enterprises, a move often referred to as *service network convergence* [2, 3]. However delivery of media packet streams over IP transport systems introduces new sources of impairments, which are not found over ordinary telephone network, such as low bit-rate CODECs, transcoding, packet loss, delay non determinism and jitter. The low bit-rate coding schemes (video and audio), which are used to minimize network congestion, significantly influences the intelligibility of original content. Moreover, VoIP service introduces new sources of delay such as framing, queuing, processing, and buffering delays.

There are several remedies to prevent/reduce the perceptual quality degradation of voice conversations carried over IP networks. Basically, perceptual quality can be improved either using network- and/or terminal- centric strategies. A network-centric approach such as IntServ and DiffServ consists of deploying adequate QoS mechanism at intermediate node to satisfy delay-sensitive service class needs [4]. A terminal-centric approach consists of well-engineering QoS control algorithms at sender and receiver sides to properly deal with incurred network impairments. For instance, the sender can automatically switch its transmission rate and protection mechanism according to the prevailing bandwidth and packet loss behavior [5]. On the other hand, the receiver can passively recover individual lost packets using packet loss concealment technique and absorb delay jitter introduced by IP networks.

In this work, we focus on the de-jittering buffer schemes used to remove jitter at the receiver side. Basically, a receiver-based network delay jitter absorption scheme delays arrivals in a buffer, referred hereafter to as *de-jittering buffer*. This allows producing *uniform spaced* media packets, as generated at sender side, which results in a faithful media reconstruction. The buffered media units are sent to the decoder according to their playback instants calculated by the de-jittering buffer management algorithm, referred hereafter to as *play-out algorithm*. The effective removal of network delay jitter generally requires a set of statistical network measurements such as one-way network delays (usually estimated) and delay variance. In accordance, the end-to-end delay is properly adjusted to optimize the perceptual quality for a given network condition. The methodologies adopted by the play-out algorithm to measure and process network measurements significantly influence the performance of any jitter removal scheme.

This paper describes a novel perceptual-based self-tuned playback algorithm of VoIP packet stream. The proposed play-out algorithm updates, upon the reception of a new packet, three statistical measurements of the average network delay and variance based on three self-calibrated first-order filters. Following a new end-to-end delay adjustment event, the proposed algorithm looks for the best safety factor, used to calibrate the delay variance, which optimizes the perceptual quality over the last active period. This predicts with high accuracy the optimal end-to-end delay, for a given network condition, which will be used till next adjustment event. Notice that the vocal conversational perceptual quality is estimated based on a single-ended parametric speech quality models. Simulation results of VoIPoW (VoIP over Wireless) clearly show that our proposed algorithm significantly outperforms the baseline de-jittering algorithms in terms of achieved instantaneous and overall perceptual quality.

The remainder of this paper is organized as follows: Section 2 surveys de-jittering algorithms used in the context of packet-based voice conversations and highlights the difficulty associated with calibration aspects. Section 3 presents our play-out algorithm designed specifically to be self-tuning and quality-aware. The evaluation of our play-out algorithm is given in Section 4. We conclude in Section 5.

2. Tuning difficulty of jitter absorption techniques

Typically, a play-out algorithm of a VoIP packet stream adjusts dynamically the end-to-end latency to follow the current trend of network delay and jitter (delay variance). Basically, the adjustment of end-to-end latency is performed at the start of a new talk-spurt[1]. This results in the expansion or compression of the original silence duration [4]. It is well-recognized that such a strategy efficiently hides to certain extent the disturbing perceptual effect caused by de-jitter buffer dynamics. This class of play-out algorithms, often referred to as *per-talk-spurt algorithms*, has been extensively studied in the literature [4, 5, 6, 7]. Per-talk-spurt algorithms can be classified into *predictive* and *reactive* de-jittering schemes [4]:

- *Predictive approaches*: They gather the history of a set of statistical network measurements such as one-way network delay, delay variation, and packet loss. Following a new end-to-end adjustment event, predictive strategies estimate the optimal play-out latency by treating the recorded history. In our opinion, predictive approaches, which are optimized for traces captured over wide area IP networks, are unsuitable for VoIP conversations over dynamic transport systems such as mobile heterogeneous networks.
- *Reactive approaches*: They use TCP-similar formulas to estimate the current trend of network delay and delay variation. The acquired statistical measurements are used at a given adjustment instant to estimate the optimal end-to-end delay for next active period. The *baseline* reactive strategy described by R. Ramjee et al in [6] uses the following expressions to update the average network delay and delay variance:

$$\hat{T}^n_{i,j} = \alpha \times \hat{T}^n_{i,j-1} + (1-\alpha) \times T^n_{i,j} \quad (1)$$

$$\hat{v}^n_{i,j} = \alpha \times \hat{v}^n_{i,j-1} + (1-\alpha) \left| \hat{T}^n_{i,j} - T^n_{i,j} \right| \quad (2)$$

where, $\hat{T}^n_{i,j}$ and $T^n_{i,j}$ are the estimated and measured network delays upon the reception of j^{th} packet of n^{th} talk-spurt triggered upon the reception of the i^{th} packet, $\hat{v}^n_{i,j}$ is the average of network delay variation, and $0 \leq \alpha \leq 1$ is the auto-correlation factor, referred hereafter to as *filter-gain* . The closer the value of α to 1 (resp. 0) is, the larger (resp. smaller) is the influence of previously sampled measurements relative to current measurement. Figure 1 illustrates timing associated with played voice packets.

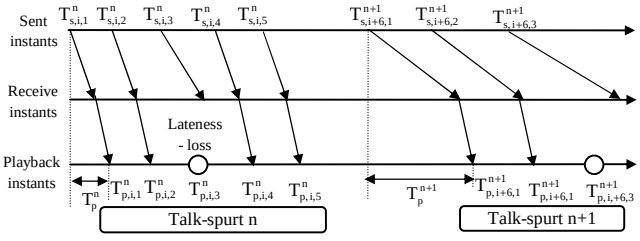


Figure 1: Timing constraints of VoIP playback process.

[1] The speech signal source during an interactive vocal conversation switches between active, known as talk-spurt, and silence periods.

R. Ramjee et al recommended set the filter-gain α to 0.99802 which was obtained empirically [6]. This value was only intended to smooth-out transient one-way network delay variations in the estimation of mean network delay. To improve the responsiveness of the fixed-gain baseline play-out algorithm while filtering transient delay variations, R. Ramjee et al proposed a second reactive strategy which uses two gain factors, α and α' to compute the average network delay (α' < α) [6]. This policy is intended to follow more closely and quickly network delay variations. The performance of this strategy remains close to the fixed-gain baseline strategy since it only uses two fixed gains rather than a dynamic gain factor computed at run-time for each received packet. Moreover, the condition used to select the gain value does not allow filtering adequately transient delay variation resulting in an extremely reactive behavior [4]. This may result in a large end-to-end delay adjustment which likely deteriorates the perceived quality.

To avoid the critical influence of the gain factor on the behavior of reactive playback strategy, R. Ramjee et al. proposed a third policy which does not use a gain factor at all to determine the mean network delay. It is more aggressive to follow network delays by assigning the value of *minimal network delay* observed over the last talk-spurt to the average network delay. A fourth playback algorithm was designed to deal with a pertinent feature of network delays observed over wide area IP networks during packetized voice conversations. Indeed, by probing network delay traces, R. Ramjee et al. noticed the presence of *delay spike* that represents an unpredictable raise followed by a linear decrease of one-way network delays [6]. To properly account for such a pertinent feature, authors equipped this playback policy with two modes: Normal and Impulse. During Normal mode, the algorithm uses the fixed-gain baseline algorithm, whereas during Impulse mode the algorithm updates the average network delay without using any gain factor. Two triggering conditions are defined in order to switch from Normal to Impulse mode, and conversely. Notice that these conditions may differ from one spike-aware playback algorithm to another [7, 8]. The behavior of the spike-aware playback policy remains identical to the fixed-gain baseline algorithm under normal conditions. This algorithm outperforms notably the baseline algorithm only for delay traces characterized by a large number of spikes [6].

The above description proves the difficulty and the relevance of filter-gain factor tuning and calibration process. In reality, a magic static gain factor that performs well under all network circumstances is not possible [9]. This observation has motivated researchers to design new reactive de-jittering algorithms which use a dynamic filter-gain computed at run-time [9, 10, 11, 12]. The discrepancy between different proposals stems from the method used to calculate the value of filter-gain at run-time. As such, the average one-way network delay and delay jitter are given by:

$$\hat{T}_{i,j}^{n} = \alpha_{i,j}^{n} \times \hat{T}_{i,j-1}^{n} + (1-\alpha_{i,j}^{n}) \times T_{i,j}^{n} \quad (3)$$

$$\hat{v}_{i,j}^{n} = \alpha_{i,j}^{n} \times \hat{v}_{i,j-1}^{n} + (1-\alpha_{i,j}^{n}) \left| \hat{T}_{i,j}^{n} - T_{i,j}^{n} \right| \quad (4)$$

where, $\alpha_{i,j}^{n}$ refers to the value of filter-gain at the reception of j^{th} packet of n^{th} talk-spurt started at the i^{th} voice packet. In [9], M. Narbutt et al. proposed a de-jittering policy labeled *α-adaptive*, which computes the filter-gain at run-time for each received packet. To do that, authors build off-line an empirical function which associates with each short-term average network delay variation the suitable value of α. The precise form of the developed function remains unknown since it is a subject to a patent. Basically, a high value is assigned to α when the playback process detects a low delay jitter inside the network. However, a low value is assigned to α when the playback process detects a high delay jitter inside the network. Performance evaluation of α-adaptive policy showed that it achieves better lateness-loss/total delay trade-off than conventional *reactive* and *predictive* de-jittering schemes [9, 10].

Following the reception of the i^{th} packet, which triggers the start of a new talk-spurt, previously described reactive policies calculate the value of next end-to-end delay as follows [6]:

$$T_{p}^{n} = \hat{T}_{i,1}^{n} + \beta \times \hat{v}_{i,1}^{n} \quad (5)$$

where, T_{p}^{n} is the end-to-end delay which will be used for the n^{th} talk-spurt, β, known as the *safety factor*, is used to control lateness-loss/total delay trade-off. The higher (resp. lower) the value of β is, the larger (resp. smaller) is the end-to-end delay and the lower (resp. higher) is the lateness-loss ratio. R. Ramjee et al. recommended setting the value of β to 4. This value was selected following an empirical tuning process using a set of traces captured from a wide area IP network [6]. Hence, this value will surely be unsuitable for other voice transport systems such as last- and multi- hop wireless data networks. This observation has motivated researchers to seek at run-time the safety factor that likely optimizes the reactive de-jittering policy performance for a given network delay behavior [13, 14]. As such, the end-to-end delay is calculated as follows:

$$T_{p}^{n} = \hat{T}_{i,1}^{n} + \beta_{n} \times \hat{v}_{i,1}^{n} \quad (6)$$

where, β_n is the value of safety factor used for the computation of end-to-end delay of the n^{th} talk-spurt. Basically, a small (resp. large) value of β, i.e., below (resp. above) 4, is only required when delay variance is high (resp. low) in order to prevent excessive late arrivals while keeping the delay below an acceptable range [13, 14]. Basically, there are two approaches, which can be adopted to look for the suitable value of β_n at run-time:

(1) *Function-based approach*: The calculation of β_n is based on a pre-defined function which is developed off-line based on a training process [12]. The network condition is characterized using measures such as mean network delay, lateness-loss ratio, or mean network delay jitter, which may be used as input parameters of the developed function. This strategy does not require recording one-way network delay measurements, and hence preserves the intrinsic assumption of reactive de-jittering policy.

(2) *History-based approach*: The calculation of β_n needs the processing of one-way network delay history. β_n may be assigned to the value which maximizes the perceptual quality over the recorded history [13]. As such, the resulting policy can be seen as a combination between intrinsic features of reactive and predictive policies.

The empirical performance study undertaken by R. Ramjee et al. and subsequently by L. Sun et al. of reactive policies showed that each de-jitter algorithm is able to efficiently absorb network delay jitter for a given network delay condition, e.g., low network delay/jitter [6, 14]. Guided by this observation, L. Sun et al. proposed a playback algorithm which automatically adopts the adequate reactive de-jitter policy according to the prevailing one-way network delay condition [14]. H. Melvin proposed a similar *delay-aware* playback strategy which switches between the static and the baseline adaptive de-jittering policies [4]. The selection of a de-jitter scheme is performed according to the prevailing trend of network delay. In our opinion, it is more suitable to concurrently run several filters on incoming one-way network delay measurements. The obtained statistical estimates of mean network delay and delay variation can be subsequently useful to calculate the suitable total delay for the next talk-spurt. In addition, we believe that a de-jittering policy which uses reactive strategy to estimate average network delay and delay variance and predictive strategy to calibrate the safety factor will surely outperform each de-jittering policy run alone, for all network delay conditions. In fact, in such a case, the agility of reactive and stability of predictive policies can be mixed to achieve optimal prediction of network delay behavior, which results in the maximization of users' satisfaction for a given network condition.

3. Self-tuned multi-filter perceptual-based de-jittering buffer algorithm

To efficiently deal with a wide range of network delay jitter conditions, especially observed over *mobile networks*, a flexible de-jittering policy is required. Such a policy should properly trade-off between *agility* and *stability* in the computation of average network delay and delay variance measurements, used to calculate the end-to-end delay. An agile de-jittering policy will result in quick reaction to transient delay changes with consequent significant expansion or compression ratio of the altered silence period duration, and sometimes an overlap between consecutive talk-spurts. This notably impairs the intelligibility of presented voice stream [4]. On the other hand, a stable reactive de-jittering policy will be unable to follow closely and quickly the prevailing network delay condition. This may lead either to a large lateness-loss arrival ratio, when network delay is under-estimated, or needlessly large total latency, when network delay is over-estimated. Notice that the agility/stability of a per-talk-spurt delay jitter removal algorithm is also linked to a set of configuration parameters used by the voice application such as hang-over duration, packet duration, and Voice Activity Detection (VAD) algorithm [4].

To achieve adequate agility over dynamic networks, we develop a new playback policy which updates the average network delay and delay variance based on three *adaptive-gain* first-order filters proposed and studied in [15]. The filter-gain is calculated using glass-box explicit mathematical expressions, unlike black-box empirical functions. These filters have been developed for the provision of accurate prediction of network statistical measures such as network delay and short-term available bandwidth over mobile networks. Upon the reception of a new arrival, each filter separately updates and maintains its estimation of average network delay and delay variance. The first-order and adaptive value of the gain factor constitute the intrinsic common features of all used filters. They differ in the method used to compute the value of filter-gain. The operational mode of the adopted three filters can be summarized as follows:

(1) *Flip-flop Filter (FF)*: It uses *two* static-gain first-order moving average filters to update the average network delay and delay variation estimates. The first (resp. second) filter, that has a static-gain value equal to 0.1 (resp. 0.9), is agile (resp. stable). The FF filter is somehow similar to the second policy described in [6], but differs in the condition used to switch between agile and stable filters. Upon the reception of a new one-way network delay measure, the FF filter selects the agile or stable filter to update average network delay and delay variation using an adapted version of control chart algorithm [15].

(2) *Stability Filter (SF)*: It dynamically adapts the value of filter-gain according to *network instability*. The SF filter was conceived to smooth-out the calculated averages of network delay and delay variance estimates, when the network delay exhibits an unstable behavior. As such, a rise of network instability, detected through consecutive divergent network delay measures, entails an increase of filter-gain. This behavior makes the filter more stable, which lead to efficiently filter-out transit delay variation. The *raw network instability* is measured upon the reception of i^{th} packet, as follows:

$$S_i = \left| T_{net}^{i} - T_{net}^{i-1} \right| \qquad (7)$$

where, T_{net}^{i} refers to the encountered network delay of i^{th} packet. The raw instability measures of S_i are smoothed as follows:

$$\hat{S}_i = \delta \times \hat{S}_{i-1} + (1-\delta) \times S_i \qquad (8)$$

where, $\hat{S}_i$ is the *filtered* measure of network instability calculated upon the reception of i^{th} packet and δ is a static smoothing factor. The greater the value of $\hat{S}_i$ is, the higher is the network instability. M. Kim et al. recommended, following an empirical study, to assign the value 0.6 to δ [15]. The filter-gain of SF filter, α_i, is computed upon the reception of i^{th} packet, as follows:

$$\alpha_i = \frac{\hat{S}_i}{S_{max}} \qquad (9)$$

where, S_{max} is the maximal value of S_i over the ten most recent samples. The closer the value of $\hat{S}_i$ to S_{max} is, the closer the value of α_i to 1. As such, the filter output will become more stable during periods characterized by a high instability.

(3) *Error-based Filter (EF)*: It dynamically updates the filter-gain according to the *prediction error* between estimated and measured one-way network delays. Basically, a higher (resp. lower) filter-gain value is used when the estimates network delays exhibit a good (resp. bad) precision with the measured ones. The prediction error is given by:

$$E_i = \left| \hat{T}_{net}^{i-1} - T_{net}^{i} \right| \qquad (10)$$

where, $\hat{T}_{net}^{i-1}$ is the EF filter output calculated upon the reception of packet number i-1. The raw prediction error measures are smoothed, as follows:

$$\hat{E}_i = \lambda \times \hat{E}_{i-1} + (1-\lambda) \times E_i \qquad (11)$$

where, $\hat{E}_i$ is the filtered prediction error measure upon the reception of i^{th} packet and λ is a smoothing factor set to 0.6. The filter-gain of EF filter is calculated as follows:

$$\alpha_i = 1 - \frac{\hat{E}_i}{E_{max}} \qquad (12)$$

where, E_{max} is computed in a similar way to S_{max}. As we can see, the lower the prediction error value is, the closer is the value of α_i to 1, which assures filter stability. A high prediction error value results in a decrease of the filter-gain, which allows a fast convergence to actual one-way network delay trend.

Upon the detection of a new talk-spurt, three estimates of average network delay and delay variance are available. As such, the estimated end-to-end delay of next talk-spurt can be calculated for each filter as follows:

$$T_{f,p}^{n} = \hat{T}_{f,i,1}^{n} + \beta_n \times \hat{v}_{f,i,1}^{n} \qquad (13)$$

where, $T_{f,p}^{n}$ is the end-to-end delay calculated based on filter f which belongs to the set {FF, SF, EF}. $\hat{T}_{f,i,1}^{n}$ and $\hat{v}_{f,i,1}^{n}$ are the produced estimates by the filter f of average one-way network delay and delay variance. $\beta_{f,n}$ is the value of the safety factor which is dynamically calculated for each filter f at the start of the n^{th} talk-spurt. The selection of suitable value of $\beta_{f,n}$ is based on the maximization of the *conversational perceptual quality* for a given network delay history.

Typically, the conversational quality is quantified using the MOS_c (Mean Opinion Score), which varies between 1 (unacceptable quality) and 5 (excellent quality) [16]. The MOS_c is affected by the network and lateness packet loss ratio and end-to-end delay. The smaller the total packet loss ratio and delay are, the better is the conversational quality. In reality, packet lateness-loss ratio and total delay are tightly linked since an increase of total delay leads to a reduction of packet lateness-loss ratio, and conversely. In our case, a decrease of $\beta_{f,n}$ entails a decay of total delay at the expense of a potential increase of packet lateness-loss ratio, and conversely. Figure 2 illustrates the effect of varying the value of safety factor $\beta_{f,n}$ on MOS_c. The curve trend can be explained as follows, when the value of the safety $\beta_{f,n}$ is too low then roughly all voice packets are ignored. This leads to a very poor *listening perceptual quality*. A gradual increase of $\beta_{f,n}$ results in decrease of lateness-loss packet ratio, which entails an improvement of listening perceptual quality. However, beyond a certain threshold, the total delay becomes excessively large which significantly degrades the quality of interaction without a significant improvement of listening perceptual quality. As such, the goal of our network delay jitter removal policy is to assign for each filter the value that maximizes the MOS_c to $\beta_{f,n}$. Notice that such a strategy requires recording network delay history, which constitutes the intrinsic feature of predictive de-jittering policy.

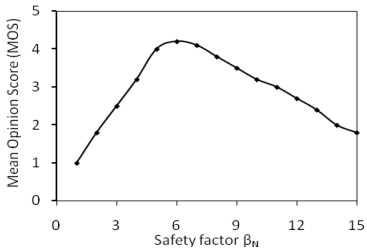


Figure 2: Inherent trend of the conversational perceptual quality scores, MOS_c, as a function of β_n.

As we can see, there is a requirement to quantify *automatically* the MOS_c to determine the suitable $\beta_{f,n}$. Several conversational speech quality estimate models have been reported in the literature [14, 17]. In this work, we use the no-reference parametric speech quality estimate model developed by K. Fujimoto et al. [17]. This model assumes that the perceptual annoying effect due to total packet loss and delay are additive in psychological scale. It has been calibrated for the standard widely-used ITU-T speech CODEC G.711, but can be extended to cover other CODECs. The MOS_c for a given packet loss ratio and end-to-end delay is obtained as follows:

$$MOS_c\,(PLR, d) = 4.10 - 0.195\times plr + 2.64\times10^{-3}d - 1.86\times10^{-5}d^2 + 1.22\times10^{-8}d^3 \qquad (14)$$

where, PLR is the total packet loss ratio and d represents the end-to-end delay. At the start of a new talk-spurt, the developed algorithm seeks, for each filter, the value of $\beta_{f,n}$ that maximizes MOS_c. The input parameters, namely total packet loss ratio and delay, of speech quality estimate model given in (14) are calculated for each value of $\beta_{f,n}$ using the history of recorded network delays. Upon the detection of a reduction of the MOS_c following an increase of $\beta_{f,n}$, our playback policy switches off the tuning process and records the optimal value of the safety factor. The history contains experienced one-way network delays over the last T seconds. The duration of history can be either static or dynamic. In this work, we use the static interval set to 9 s. By the end of the history-based auto-tuning process, three optimal MOS_c scores -- one value per filter -- will be available for the de-jittering algorithm. Obviously, our developed policy selects the configuration, i.e., the average network delay, delay variance, and safety factor, which optimizes the MOS_c. Notice here that the designed algorithm is CODEC aware since the calculation of the end-to-end delay depends on the speech quality estimate model which is specific for each CODEC.

Algorithm 1 summarizes our developed auto-tuned de-jittering policy. Basically, the algorithm records the one-way network delay history using a rolling list which keeps the recent observations. The average network delay and delay variance are separately updated using each filter. At the occurrence of a new talk-spurt, the playback algorithm seeks the safety factor that optimizes the perceived quality for each filter over the recorded history. After obtaining the best safety factor for each filter, our developed algorithm selects the total latency that maximizes the perceptual quality among all available MOS_c scores.

4. Performance evaluation

In order to investigate the performance of our designed playback policy over mobile network, we simulate the configuration illustrated in Figure 3 using NS2 [13]. The simulated test-bed includes two access points used to bridge between wired and wireless worlds. End nodes are attached and served by one access point at a given instant. The access points are linked using an existing infrastructure which includes one core router and two high capacity wired links (see Figure 3). The wireless interface data rate is set to 2 Mbps. Thus, the wireless link constitutes the bottleneck of all established connections. The distributed access protocol IEEE 802.11b is used in order to resolve contentions. All mobile nodes are assumed to remain stationary during a simulation run.

Algorithm 1 : Self-tuned quality-aware de-jittering policy

$T^n_{f,p}$: The n^{th} total latency calculated based on the filter f

$\hat{T}^n_{f,i,j}$: The average network delay calculated using the filter f at the reception of j^{th} packet of n^{th} talk-spurt started at i^{th} packet

$\hat{V}^n_{f,i,j}$: The average delay variance calculated using the filter f at the reception of j^{th} packet of n^{th} talk-spurt started at i^{th} packet

INC: The increment amount of the safety factor β_n

H: A rolling list of one-way delay history

opt[3]: An array containing optimal score achieved by each filter

Beta[3]: An array containing optimal safety factor for each filter

FF: Flip-Flop filter, SF: Stability Filter, EF: Error-based Filter

β_{min} and β_{max} are the lower and upper thresholds of β_n

LossRate ( ): A function which returns the number of ignored packets for a given network delay history and a playback latency

MOS_c( ): A model estimating the conversational perceptual quality

MaxIndex( ): A function which returns the index of the maximal value of a given array of floats

1. **for** each received packet **do**
2. **if** (new talk-spurt == FALSE)
3. update the history of network delay H
4. **for** each filter f ϵ {FF, SF, EF} **do**
5. update the weighted network delay and variance
6. **end for**
7. **else**
8. **for** each filter f ϵ {FF, SF, EF} **do**
9. Beta[f]← β_{min}, score ← 1
10. **do**
11. opt[f] ← score
12. $T^n_{f,p} \leftarrow \hat{T}^n_{f,i,1} + Beta[f]\times \hat{v}^n_{f,i,1}$
13. PLR ←LossRate (H, $T^n_{f,p}$)
14. score ←MOS_c(PLR, $T^n_{f,p}$)
15. Beta[f] ← Beta[f] + INC
16. **until** (score <= opt[f] or $\beta_n \geq \beta_{max}$)
17. **end for**
18. f ← MaxIndex (opt)
19. $T^n_p \leftarrow \hat{T}^n_{f,i,1} + Beta[f] \times \hat{V}^n_{f,i,1}$
20. Initialise H
21. **end if**

// Playback instant of j^{th} packet of n^{th} talk-spurt started at i^{th} packet

22. $T^n_{p,i,j} = T^n_{s,i,j} + T^n_p$
23. **end for**

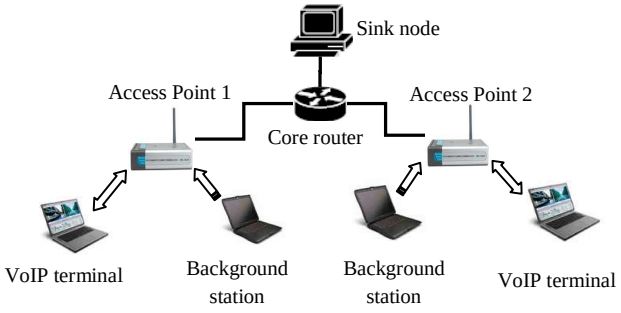
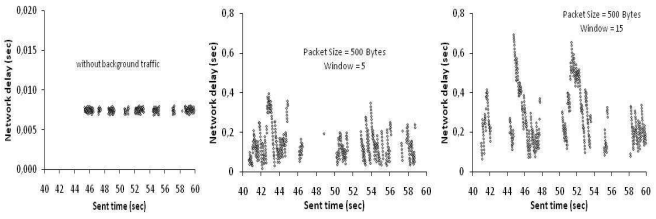


Figure 3: Hybrid wired/wireless simulated test-bed.

A bidirectional voice conversation is established between the two VoIP terminals included in the test-bed (see Figure 3). Voice streams are synthesized using an ON/OFF model which imitates a realistic packet-based voice source when a voice activity detection algorithm is used. The data rate at application level is equal to 64 kbps in order to mimic the output of the CODEC G.711. The packet duration is set to 20 ms which implies a payload size equal to 160 bytes and packet rate equal to 50 packets / second. The voice conversation lasts for 250 sec. Two background stations are used in order to increase the network workload. Background traffic is sent toward the wired sink node from the start to the end of the simulation run. The intensity of background traffic has been varied according to packet size, which was varied from 500 to 1500 Bytes, and maximal TCP tolerable window size, which was varied from 5 to 30 packets. Figure 4a illustrates the incurred one-way network delays when background traffic is disabled. However, Figures 4b and 4b illustrate the effect of different injected background traffic intensities on one-way network delay. From Figure 4a, we can observe that voice packets reached the receiver side roughly instantaneously without notable network delay jitter. Figures 4b and 4b illustrate the incurred one-way network delay when background traffic is enabled. In such conditions, voice packets sustain high one-way delays and delay variations. This is due to the best-effort property of the transport network. In addition, the examined VoIP voice packets travel through two wireless hops which increase dramatically the network latency. Moreover, the packet size of background traffic is relatively long (> 500 bytes) which blackouts the wireless channel for a significant duration.



(a) (b) (c)

Figure 4: The effect of background traffic intensity on network delay variation

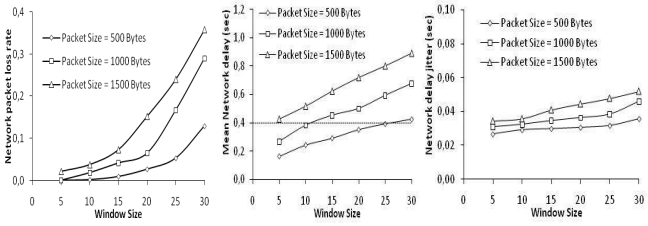
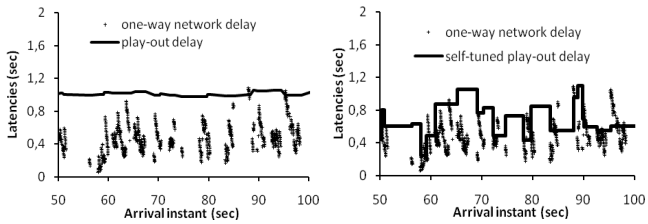


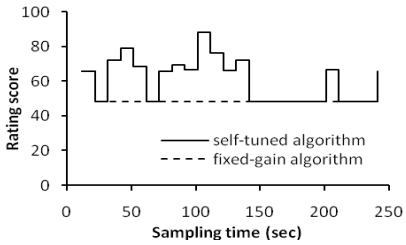
Figure 5: Influence of background traffic on VoIP packets.

As such, voice packets are enforced to wait for a long duration either at the mobile node interface or at the transmission queue of the access point. Moreover, the background traffic attempts to efficiently use the available bandwidth which will quickly saturate the wireless channel. Further, the acknowledgment packets sent by sink node increase the cell load and results in contentions with voice packets as well as background data packets. Figure 5 shows the effect of different background traffic intensities on packet loss ratio and mean one-way network delay and jitter. Figures 6a and 6b illustrate the behavior of the baseline policy, where the gain value is set to 0.99802, and our self-tuned de-jittering policy. Figure 6 clearly shows that our self-tuned play-out algorithm follows more closely network delay variations than the baseline playback algorithm. The self-tuned and baseline playback algorithms entail an overall lateness-loss ratio equal to 32.13% and 10.10%, respectively. On the other hand, the self-tuned and baseline playback algorithms entail a mean total delay equal to 761 ms and 1105 ms, respectively. The reduced packet lateness-loss ratio achieved by the baseline playback algorithm is performed at the expense of a dramatic increase of total delay. This total delay exceeds significantly the tolerable one-way delay in the context of conversational services (<400ms). Figure 7 gives the instantaneous perceptual quality achieved by the self-tuned and the baseline playback algorithms. The perceptual quality is estimated using an adapted version of ITU-T E-Model which produces as output a rating factor varying between 0 (bad quality) and 100 (Excellent quality) [10]. The assessment period duration is set to 10s. These curves prove that the self-tuned play-out algorithm improves considerably the perceived quality over time. Moreover, our self-tuned policy achieves a better overall rating factor at the end of the voice conversation (see Figure 7).



(a) Baseline playback algorithm (b) Self-tuned playback algorithm

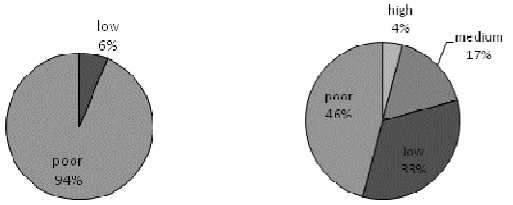
Figure 6 : Behavior of our self-tuned de-jittering policy compared with the baseline policy (background packet size = 1500 bytes).



$$R^{\text{self-tuned}}_{\text{end-of-call}} = 53.59$$

$$R^{\text{fixed-gain}}_{\text{end-of-call}} = 48.24$$

Figure 7 : Instantanoeus and overall perceptual qualiy of self-tuned and fixed play-out algorithms.



(a) Baseline playback algorithm (b) Self-tuned playback algorithm

Figure 8 : Overall user satisfaction throughout the voice conversation.

The pie charts depicated in Figure 8 illustrate the user satisfaction throughout the packet-based voice conversation. These charts are produced based on the perceptual contour concept described in [10]. Plotted pie charts shows that the self-tuned playback algorithm outperforms the baseline playback policy at perceptual level. For instance, the baseline de-jitter algorithm produces a poor perceptual quality during 94% of the voice conversation duration, whereas the self-tuned play-out algorithm reduces this ratio to 46% of the studied voice conversation.

5. Conclusion and future work

In this paper, we presented a new de-jittering algorithm of packet-based voice conversations. The designed play-out algorithm has flexibility to cope with a wide range of delay jitters observed over mobile networks. It concurrently uses three adaptive-gain first-order filters to calculate the optimal end-to-end delay. At the occurrence of an adjustment event, it self-tunes at run-time the safety factor that likely optimizes the perceived quality. Simulation results of VoIPoW show that our de-jittering algorithm outperforms the baseline fixed-gain and safety factor de-jittering policy at perceptual level. The obtained results exhibit that contention delay and jitter constitutes a potential source of quality degradation (through both playout delay and late packet loss) which should be properly reduced using dedicated algorithms and protocols. The delay-sensitive feature of packet-based voice conversations should be considered by access as well as core nodes especially in wireless environments. This will be investigated further in our future work as we examine the contribution that QoS enabled protocols such as 802.11e can make.

Reference

[1] H. Schulzrinne, http://www.cs.columbia.edu/~hgs/, [on-line] Personal Website, visited in March 2009.

[2] European Telecommunications Standards Institute (ETSI), http://www.etsi.org/, [on-line] visited in March 2009.

[3] International Telecommunication Union (ITU-T), http://www.itu.int/, [on-line] visited in March 2009.

[4] H. Melvin, "The use of Synchronized Time in Voice over Internet Protocol (VoIP) Application", PhD dissertation, University College Dublin, October 2004.

[5] C. Hoene, "Internet Telephony over Wireless Links", PhD dissertation, Technical University of Berlin, Germany, December 2005.

[6] R. Ramjee, J. Kurose, and D. Towsley, and H. Schulzrinne, "Adaptive Play-out Mechanisms for Packetized Audio Applications in Wide Area Network", in Proceedings of IEEE INFOCOM, pp. 680-688, Toronto, Canada, 1994.

[7] S. Moon, J. Kurose, and D. Towsley, "Packet Audio Play-out Delay Adjustment: Performances bounds and Algorithms", ACM/Springer Multimedia Systems, Vol. 6, pp. 17-28, January 1998.

[8] P. Hu, "The Impact of Adaptive Play-out Buffer Algorithm on Perceived Speech Quality Transported over IP Networks", Master thesis report, School of Computing, Communication and Electronics, University of Plymouth, September 2003.

[9] M. Narbutt and L. Murphy, "VOIP Play-out Buffer Adjustment using Adaptive Estimation of Network Delays", in Proceedings of 18th International Tele-traffic Congress (ITC-18), pp. 1171-1180, Berlin, Germany, September 2003.

[10] M. Narbutt, A. Kelly, L. Murphy, P. Perry, "Adaptive VoIP Play-out Scheduling: Assessing User Satisfaction", IEEE Internet Computing Magazine, July/August 2005, pp: 18-24.

[11] A. Kansal and A. Karandikar, "Adaptive delay estimation for low jitter audio over Internet", in Proceedings of IEEE Global Telecommunications Conference, (GLOBECOM 01), vol.4, pp: 2591 - 2595, 2001.

[12] Daniel R. Jeske, W. Matragi, B. Samadi, "Adaptive Play-Out Algorithms for Voice Packets", In Proceedings of IEEE International Conference on Communications (ICC 2001), vol. 3, pp:775-779, Helsinki, Finland,11-14 June 2001.

[13] Y. Jung and J. W. Atwood, "β-Adaptive Play-out Scheme for Voice over IP Applications", IEICE Transactions on Communication, vol.E88–B, no.5, May 2005 (LETTER).

[14] L. Sun and E. C. Ifeachor, "Prediction of Perceived Conversational Speech Quality and Effects of Playout Buffer Algorithms", In Proceedings of IEEE International Conference on Communications (ICC 2003), vol. 1, pp:1-6, 11-15 May 2003

[15] Kim M. and Noble B., "Mobile Network Estimation", in Proceedings of the ACM Conference on Mobile Computing and Networking, Rome, Italy, June 2001.

[16] A. Rix, J. Beerends, D. Kim, P. Kroon, and O. Ghitza, "Objective assessment of speech and audio quality: Technology and Applications". IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 6, pp. 1890:1901, November 2006.

[17] K. Fujimoto, S. Ata, and M. Murata, "Adaptive Play-out Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications", in Proceedings of IEEE Globecom 2002, Nov 2002.

[18] K. Fall and K. Varadhan, "The ns Manual", VINT Project, November 2001.