Enhanced Timing Recovery Using Active Jitter Estimation for Voice-Over IP Networks

Hyoung-Gook Kim[1]
[1]Department of Electronics Convergence Engineering,
Kwangwoon University,447-1, South Korea
[e-mail: hkim@kw.ac.kr]
*Corresponding author: Hyoung-Gook Kim

Received October 8, 2011; revised January 2, 2012; revised February 17, 2012; accepted March 19, 2012; published April 25, 2012

Abstract

Improving the quality of service in IP networks is a major challenge for real-time voice communications. In particular, packet arrival-delay variation, so-called "jitter," is one of the main factors that degrade the quality of voice in mobile devices with the voice-over Internet protocol (VoIP). To resolve this issue, a receiver-based enhanced timing recovery algorithm combined with active jitter estimation is proposed. The proposed algorithm copes with the effect of transmission jitter by expanding or compressing each packet according to the predicted network delay and variations. Additionally, the active network jitter estimation incorporates rapid detection of delay spikes and reacts to changes in network conditions. Extensive simulations have shown that the proposed algorithm delivers high voice quality by pursuing an optimal trade-off between average buffering delay and packet loss rate.

Keywords: VoIP, timing recovery, active network jitter estimation, buffering delay and packet loss

This research has been conducted by the Research Grant of Kwangwoon University in 2010. This research was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0004311).

http://dx.doi.org/10.3837/tiis.2012.04.003

1. Introduction

Voice-over Internet protocol (VoIP) is one of the most recently emerging technologies in the area of speech communications. The main advantage of this technology is a utilization of existing infrastructure in the form of Internet connection. In the last few years, VoIP has been very popular because of its low cost, similar quality to traditional telephones, and ease of use.

As a result of the steady growth in VoIP usage, providing reliable services with satisfactory voice quality is now a high priority for Internet and VoIP service providers. A number of factors may affect the service quality of VoIP [1] for example, packet loss, packet delay, network delay variation (also known as "jitter") [2], echo, noise [3], and harmonic and inharmonic distortion [4].

Normally, in packet-switched networks, a voice packet is transmitted from the speaker's node to the listener's node every 20 ms or 30 ms to maintain continuous and smooth speech conversations. However, voice packets can be transferred via a large number of routers, and the amount of routers can be changed during a conversation. Packet queuing delays are variable and hard to predict, so, some packets arrive at the receiver at irregular intervals. Jitter is the irregularity with which packets in a transmission arrive. This irregularity can damage voice quality, as VoIP programs need to receive data at a regular pace.

In this study, we focus on decreasing the influence of delay jitter on voice quality. To reduce the influence of the jitter, a playout buffer can be employed by the receiver to hold a VoIP packet until its scheduled playout time. In adaptive playout, the buffer size can be minimized by using a timing recovery algorithm, which allows each packet to be expanded or compressed [5]. Proper reconstruction of continuous output speech is achieved by scaling individual voice packets using timing recovery. The timing recovery algorithms have a positive effect on the service quality, but they come at the cost of additional algorithmic delay. Several VoIP playout buffer scheduling or timing recovery algorithms have been proposed [6][7][8][9][10][11][12][13][14][15]. For effective expanding or compressing of each packet resulting from changes in network delay, jitter estimation with high accuracy is very important.

From network delay traces, it is common to observe sudden high delays (known as spikes) [16]. A spike begins with the sudden onset of a large increase in network delay and ends when network delays return to a steady-state value. Such delay spikes can be caused by heavy congestion resulting in long queues at routers within the network. The spike period is usually very short and unpredictable, so burst packet loss occurs as a result. In the case of a sudden increase of the transmission delay, an underflow of the playout buffer occurs. A sudden decrease of the transmission delay may cause an overflow of the playout buffer. To reduce the influence of delay spikes, playout buffer algorithms need to incorporate spike detection [6][10], thereby computing their delay estimates differently and rapidly during spikes.

The most challenging issue raised by VoIP playout buffering is how to determine the most appropriate buffer size for current network conditions in order to improve the voice quality transmitted through an IP network.

The contribution of this paper is four-fold: (1) A method for deciding on a timing recovery mode compared to loss concealment mode or merging and smoothing mode for maintaining a balance between conversational interactivity and speech quality is proposed. (2) A subprocess in the timing recovery mode, which is classified into three categories, *time compression*, *time*

expansion, and *normal*, is presented. (3) An enhanced expansion and compression approach in timing recovery algorithms for alleviating the metallic artifacts in a transition region and improving the voice quality is presented. (4) An effective jitter estimation for improving the trade-off between buffering delay and packet loss is proposed. This jitter estimation detects delay spikes rapidly and is applied to adjusting the playout buffering delay according to current network conditions.

The approach used in this paper has three advantages: (1) It improves the trade-off between the buffering delay and packet loss very quickly, (2) it is simpler than existing approaches and delivers high-quality voice as a result, and (3) it is voice codec independent and suitable for any practical mobile VoIP system.

This paper is organized as follows. Section 2 contains a review of related works. Section 3 and Section 4 detail a structure of a receiving part in a VoIP system and the proposed timing recovery method combined with active jitter estimation. Section 5 presents the results of performance comparison of different algorithms, and conclusions are summarized in Section 6.

2. Related Works

Several VoIP playout buffer scheduling or timing recovery algorithms have been proposed to improve the voice quality of VoIP communications.

Ramjee [6] adjusted the buffer size based on the exponential weighted moving average (EWMA) of network delays and their standard deviation, where the weights of the variables are selected empirically and fixed. Subsequently, Narbutt [7] extended the above approach by adaptively adjusting the EWMA weight based on the magnitude of the delay jitter. The weight is set higher when the delay jitter is smaller; conversely, it is set lower when the jitter is larger. The reported simulation results show that this approach improves the trade-off between buffer delay and packet loss significantly. The EWMA algorithm [6] is modified by Kansal [8] to adaptively adjust α to an optimal value. The algorithm calculates the loss rate for previous talkspurts based on the current value of α. The increment value is much smaller than α, as small changes in α lead to large changes in packet loss and total end-to-end delay. Pinto [9] presents a method that adjusts silence periods between talkspurts to improve voice interaction qualities. The approaches proposed in Pinto's work are further extended by adjusting the buffer size within a speech burst [10][11]. The objective is to ensure that the playout buffer adapts to varying network conditions more quickly, and thereby improves the conversation quality of VoIP calls. Chi [12], Li [13], and Aragao [14] suggest statistically-based approaches using the statistics of past network delays to compute the current packet playout delay. Playout scheduling is based on modeling packet arrival times with K-Erlang distribution, Gaussian model, and Pareto distribution, among others. Based on the distribution of past delays, the playout delay is selected such that a tolerable percentage of packets will arrive "late." However, these methods are "packet-based," which means that they decide to stretch or compress a packet when it is received. As such, it does not reflect the changing state of arriving packets until the packet outputs. Florncio [5] uses the "buffer-based" method, which decides to stretch or compress only when the audio playout device needs a frame. An adaptive filter-based algorithm for adaptive playout buffer scheduling is proposed by DeLeon [15]. An accurate prediction of the network delay can rapidly track network changes and thus adjust the delay more effectively. Different ways of detecting sudden high delays such as spikes and adjusting the playout time accordingly are used by Florencio [5] and Ramjee [6]. The adaptive

gap-based algorithm [9] simply incorporates the spike detection mode proposed by Ramjee. However, they are unable to adapt rapidly to delay spikes. The normalize least mean square (NLMS) adaptive filtering algorithm [15] provides a good prediction of the network delay and closely tracks any fluctuations in network delay. One drawback of the NLMS predictor is that it does not explicitly detect delay spikes and therefore does not alter its behavior during a spike.

3. Structure of Receiving Part in a VoIP System

Fig. 1 illustrates a structure of the receiving part including timing recovery in a mobile internet phone.

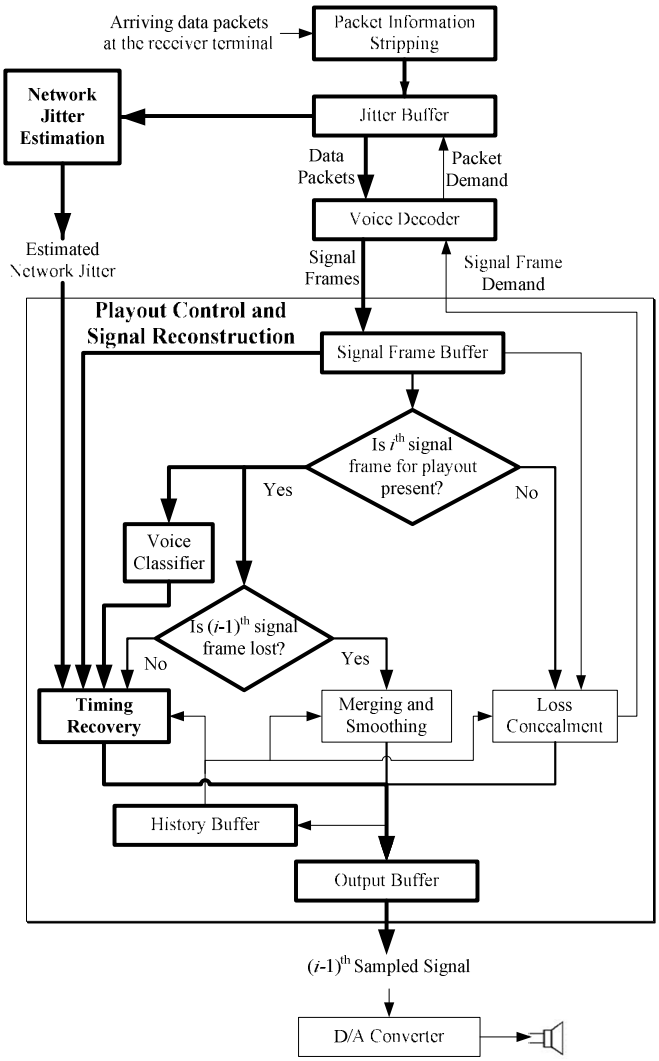


Fig. 1. Structure of the receiver side

The receiving system employs combined playout control and signal reconstruction on decoded signal frames. On the receiver side of mobile VoIP, when a packet arrives at the receiver, the

receiver strips the packet information and places the packet in the jitter buffer. Next, the arriving packet information is passed on to network jitter estimation in order to predict the network condition more quickly.

To play out the arriving packets at a regular interval, the receiving system needs to maintain a jitter buffer, a signal frame buffer, a history buffer, and an output buffer. In this paper, the length of each packet is 20 ms and the sizes of the jitter buffer, the signal frame buffer, the history buffer, and the output buffer as a storage medium are 200 ms, 200 ms, 60 ms, and 20 ms, respectively.

Typically, packets do not arrive at the receiver terminal according to the order of sequence number generated from the sender under the influence of the network jitter. $PA_{i,k}$ denotes the k^{th} arriving packet in the i^{th} talk interval or i^{th} talk-sequence (i is the time index when each voice packet is generated at the sending host, while k is the time index when each packet arrives at the receiving host). The jitter buffer holds incoming packets, rearranges the arriving packets due to the time when the arriving packets are generated at the sending host, and releases them for decoding at a regulated speed (i.e., every 10 ms). The amount of packets in the jitter buffer will be one to ten packets. Each signal frame decoded from the jitter buffer is stored in a signal frame buffer. The currently used voice codecs are G.711, G.722.2, G.726, G.728, G.729AB, and G.729E.

The decoded signal frames are input to the combined playout control and signal reconstruction module (PCSR), which decides on one of three processing modes: loss concealment, merging and smoothing, or timing recovery. The PCSR then performs signal reconstruction based on the decided processing mode. Thereafter, the system plays the current packet at the scheduled time (i.e., every 20 ms) through the speaker.

The proposed decision logic for signal processing is one of the key contributions for maintaining a balance between conversational interactivity and speech quality and performs as follows:

- *Loss concealment mode*: If i^{th} signal frame (subsequent signal frame for playing out) is absent in the signal frame buffer, a packet is declared as lost, and "loss concealment mode" is entered. The objective of packet loss concealment [16][17][18][19][20][21] is to generate a synthetic speech signal to cover missing data. Using waveform similarity overlap-add between the subsequent frame and previous frame, the lost frame is reconstructed.
- *Merging and smoothing mode*: If i^{th} signal frame is present in the signal frame buffer and $(i\text{-}1)^{th}$ signal frame was lost, discontinuity between i^{th} signal frame and $(i\text{-}1)^{th}$ substituted signal frame occurs, and "merging and smoothing mode" is entered. By merging and smoothing, two signal frames in a transition region are smoothly interpolated to alleviate discontinuity of the transitions from a signal frame to the substitute frame or from the substitute frame to the following signal frame [21][22].
- *Timing recovery mode*: If i^{th} signal frame is present in the signal frame buffer and $(i\text{-}1)^{th}$ signal frame was not lost, "timing recovery mode" is entered. Using normalized auto correlation, the i^{th} signal frame is classified into silence, unvoiced, and voiced signal frames. The classified results are then used in the timing recovery process.

For recovering lost packets, the PCSR module often makes a subsequent frame demand from the voice decoder. This causes the voice decoder to make a packet demand from the jitter buffer. The jitter buffer extracts a voice data packet and sends it to the voice decoder, which decodes it as a signal frame. The digital-to-analog (D/A) converter regularly converts the

sampled signal frame from PCSR into an analog signal. Finally, the user hears the analog voice signal through a speaker.

4. Proposed Timing Recovery Combined with Active Jitter Estimation

In the proposed timing recovery, the current packet is expanded or compressed by using the estimated network jitter in order to allow the next packet to be played at the scheduled time. Detailed network jitter estimation and timing recovery are explained in subsections 4.1 and 4.2.

4.1 Active Network Jitter Estimation

The proposed active network jitter estimation incorporates rapid detection of delay spikes to react to changes in network conditions. In the proposed algorithm, the active network jitter is more accurately estimated by using the mean and variation of interarrival jitter than more conventional algorithms.

Fig. 2 depicts an algorithm flow chart of the proposed active jitter estimation.

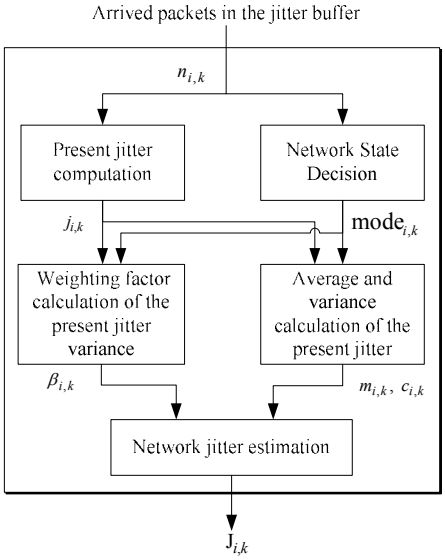


Fig. 2. Block diagram of the proposed network jitter estimation

The active jitter estimation can be divided into five modules: present jitter computation, network state decision, weighting factor calculation of the present jitter variance, average and variance calculation of the present jitter variance, and network jitter estimation.

First, interarrival jitter is calculated, while network state mode is determined using the arriving packet information. The interarrival jitter $j_{i,k}$ is defined as the difference in packet spacing at the receiver compared to the sender for the packet arrival $PA_{i,k}$ and the previous packet arrival $PA_{i,k-1}$. Then:

$$j_{i,k} = n_{i,k} - n_{i,k-1} = \left(a_{i,k} - a_{i,k-1}\right) - \left(g_{i,k} - g_{i,k-1}\right) \quad (1)$$

where $n_{i,k}$, $a_{i,k}$, $g_{i,k}$ denote, respectively, the network delay that the k^{th} transmitted packet experiences, the time when the k^{th} packet arrives at the receiver, and the time when the k^{th} packet is generated at the sending host.

Using the network delay that the k^{th} received packet experiences, network state is classified into one of two zones: *spike* or *normal*. The *normal* state in the yellow part of **Fig. 3** shows the stable network situation with a small amount of network jitter. The variation of the network delay is constant in the *normal* state. In contrast, the *spike* state in the red part of **Fig. 3** is the unstable network situation, with increasing interarrival jitter. At the end of the *spike* state, the interarrival jitter changes rapidly, as shown in the blue part.

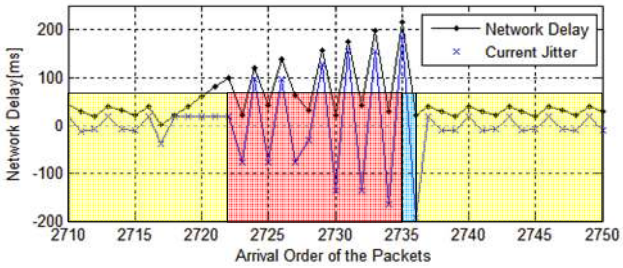


Fig. 3. The types of network situation

The actual formula used for detecting spike is given in **Fig. 4**.

If $(\text{mode}_{i,k-1} = normal)$
  $D_{i,k} = (a_{i,k} - a_{i,k-1}) + (g_{i,k} - g_{i,k-1})$
  If $(D_{i,k} > Th_{Spike})$ /* beginning of a spike */
    $\text{mode}_{i,k} = spike$
    $N_{i,k} = Sn_{i,k} - lSn_{i,k-1} - 1$
  Else
    $\text{mode}_{i,k} = normal$
Else
  If $(lSn_{i,k-1} < Np_{i,k} < Sn_{i,k})$
    $N_{i,k} = N_{i,k-1} - 1$
  Else
    $N_{i,k} = N_{i,k-1}$
  If $(N_{i,k} = 0)$ /* end of a spike */
    $\text{mode}_{i,k} = normal$
  Else
    $\text{mode}_{i,k} = spike$

Fig. 4. Algorithm of network mode decision

In **Fig. 4**, $D_{i,k}$ means the network variation between consecutive arriving packets and is defined as the sum of arrival-time difference and generation-time difference of consecutively arriving packets. $N_{i,k}$ is the number of arriving packets during *spike* state and is used to detect the end of *spike* zone. $lSn_{i,k-1}$ and $Sn_{i,k}$ represent the largest talk-sequence number among the previous arrived packets in *normal* zone and the talk-sequence number of the arriving packet at the

beginning of *spike* zone, respectively. And $Np_{i,k}$ is the sequence number of the current arriving packet during the *spike* state.

At the beginning of the *spike* zone, the difference of consecutive packet arrival may increase, because the packet does not arrive at the receiving terminal for a long time, or the generation-time difference at the sending host may increase because of the reversed arrival of packets, even if the packets arrive at the receiving terminal regularly. A delay spike is detected when $D_{i,k}$ is larger than Th_{Spike} (> 80), such as the case of spike threshold. Upon detecting the delay spike, the algorithm switches to spike mode. In *spike* mode of operation, $N_{i,k}$ is calculated by using $lSn_{i,k-1}$ and $Sn_{i,k}$. $N_{i,k}$ decreases when $Np_{i,k}$ is greater than $lSn_{i,k-1}$ and less than $Sn_{i,k}$. When $N_{i,k}$ drops to 0, the *spike* is judged to have ended, and the network situation returns to *normal*. $N_{i,k}$ equal to 0 indicates that the rest of the packets that did not arrive at the receiving terminal during the *spike* state have finally arrived.

Second, the current weighting factor $\beta_{i,k}$ of the network jitter variance is adjusted only in the *normal* zone and can be obtained as:

$$\begin{aligned} &if\left(mode_{i,k} = normal\right) \\ &\quad \beta_{i,k} = \begin{cases} \beta_{i,k-1} - \alpha_c, & where\ \beta_{i,opt} < \beta_{i,k-1} \\ \beta_{i,k-1} + \alpha_d, & otherwise \end{cases} \\ &else \\ &\quad \beta_{i,k} = \beta_{i,k-1} \end{aligned} \tag{2}$$

using α_c ($0< \alpha_c <1$), α_d ($0.5< \alpha_d <1.5$) and:

$$\beta_{i,opt} = \frac{j_{i,k} - m_{i,k-1}}{c_{i,k-1}} \tag{3}$$

where $\beta_{i,opt}$ is a weighting factor of optimal network jitter variance to minimize the jitter error incurred by varying network conditions. To prevent the increased buffering delay due to $\beta_{i,opt} < \beta_{i,k-1}$ in *normal* state, the current weighting factor $\beta_{i,k}$ is adapted by subtracting α_c from the previous stored weighting factor $\beta_{i,k-1}$. To reduce packet loss rate due to $\beta_{i,opt} \geq \beta_{i,k-1}$ in *normal* state, $\beta_{i,k}$ is adapted by adding α_c to $\beta_{i,k-1}$. Conversely, during the spike state, the previous stored weighting factor $\beta_{i,k-1}$ is reused as the current weighting factor $\beta_{i,k}$.

After the adjustment of $\beta_{i,k}$, average $c_{i,k}$ and variance $m_{i,k}$ of the interarrival jitter are calculated according to the determined network situation, as shown in equation (4):

$$
\begin{aligned}
&if\left(\text{mode}_{i,k} = normal \text{ and mode}_{i,k\text{-}1} = spike\right)\\
&\quad m_{i,k} = \alpha \cdot m_{i,tmp} + (1-\alpha)\cdot j_{i,k},\\
&\quad c_{i,k} = \alpha \cdot c_{i,tmp} + (1-\alpha)\cdot \left|m_{i,k} - j_{i,k}\right|\\
&else\\
&\quad m_{i,k} = \alpha \cdot m_{i,k-1} + (1-\alpha)\cdot j_{i,k},\\
&\quad c_{i,k} = \alpha \cdot c_{i,k-1} + (1-\alpha)\cdot \left|m_{i,k} - j_{i,k}\right|
\end{aligned}
\tag{4}
$$

where α ($0.3< \alpha <0.8$) is a smoothing parameter; *tmp* is the temporal point when the *spike* is detected; $m_{i,tmp}$ and $c_{i,tmp}$ are respectively the mean and the variance of the jitter at the point when the previous *spike* was detected.

For the end of *spike*, the mean and variance of each interarrival jitter are calculated using those of the previous $(k\text{-}1)^{th}$ jitter. Thus, the fast changing network situation can be adjusted more accurately for scheduled playout times.

Finally, the active jitter of the k^{th} arriving packet in the i^{th} talk interval is estimated using the calculated mean and variance of network jitters as:

$$
\mathrm{J}_{i,k} = m_{i,k} + \beta_{i,k} \cdot c_{i,k} \tag{5}
$$

4.2 Timing Recovery

In order to allow the next packet to be played at the scheduled time, the timing recovery algorithm tries to overcome the effect of transmission jitter by way of normal operating or expanding or compressing each packet using the estimated network jitter. By using the proposed timing recovery algorithm, the jitter is smoothed out before packets are played at the receiver. Additionally, the influence of time clock drift in terminal equipment is decreased.

Fig. 5 depicts the algorithm flow chart for timing recovery.

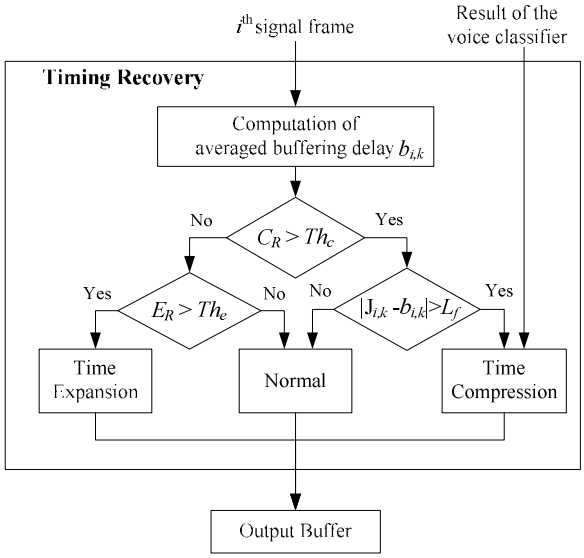


Fig. 5. Diagram for sub-process decision logic of timing recovery

First, current buffering delay $b_{i,k}$ is calculated by a first order recursive averaging, given by:

$$b_{i,k} = \alpha_b \cdot b_{i-1,k} + (1-\alpha_b)\cdot(p_{i,k} - a_{i,k}) \tag{6}$$

where α_b ($0< \alpha_b <1$) is a smoothing parameter, and $p_{i,k}$ and $a_{i,k}$ denote playout time and arrival time of k^{th} packet in the i^{th} talk interval frame, respectively.

Second, subprocess decision logic for timing recovery is performed using the estimated network jitter and the calculated current buffering delay. For this, the following three subprocesses are handled:

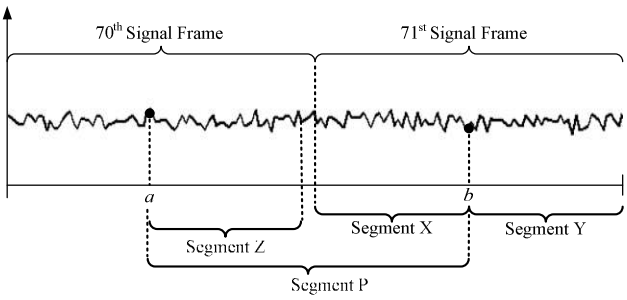
- *Time expansion subprocess*: Let $C_R = L_a / J_{i,k}$ denotes the ratio between total length L_a of the remaining signal frames in the signal frame buffer to the active network jitter $J_{i,k}$ of the k^{th} arriving packet estimated from the jitter buffer. $E_R = T_a / J_{i,k}$ represents the ratio between length of time T_a that the packet did not arrive at the receiver to the active network jitter $J_{i,k}$. If C_R is smaller than the compression threshold Th_c ($0 < Th_c < 3$), and E_R is larger than the expansion threshold Th_e ($0 < Th_e < 1$), the time expansion subprocess is entered.
- *Time compression subprocess*: If C_R is larger than the compression threshold Th_c and $| J_{i,k} - b_{i,k} |$ is larger than length L_f of one frame, the time compression subprocess is entered.
- *Normal subprocess*: If C_R is smaller than the compression threshold Th_c, and E_R is smaller than the expansion threshold Th_e, the normal subprocess is entered. Additionally, If C_R is larger than the compression threshold Th_c and $| J_{i,k} - b_{i,k} |$ is smaller than length L_f of one frame, the normal subprocess is entered.

After the subprocess decision, each subprocess is performed. Enhanced compressing and expansion algorithms are presented as follows:

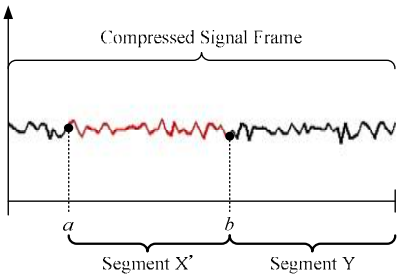
- *Time compression*

A sudden decrease of the transmission delay (i.e., network condition is changed into normal state from spike state) may cause an overflow of the jitter buffer. This situation is an indication that the system introduces excessive delay of the playback. In this case, the receiver performs time compression in order to reduce the buffering delay. The length of signal frames in the signal frame buffer will converge to a length similar to the estimated network jitter. Time compression eliminates unnecessary buffering delay using a compression algorithm.

Fig. 6 illustrates an exemplifying time compression of two signal frames classified as unvoiced.



(a). Original signal frame



(b). Compressed signal frame

Fig. 6. Time compression of two signal frames

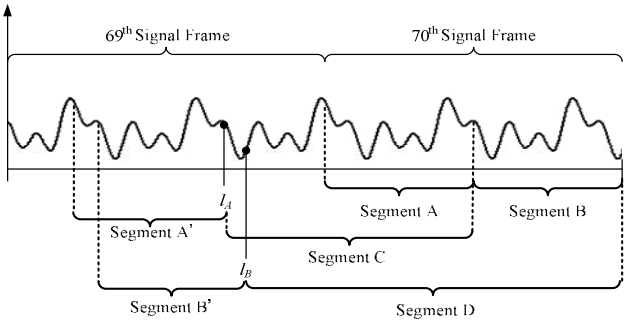
Time compression strategy depends on the signal class of both of the i^{th} signal frame and $(i\text{-}1)^{th}$ signal frame (i.e., using normalized auto correlation, each signal frame is classified into one of three classes: silence, unvoiced, or voiced signal class). The objective voice quality testing results using perceptual evaluation of speech quality (PESQ) indicate that time compression in the voiced region can cause distortion and degrade the voice quality. Therefore, time compression is used only in the region including signal frames classified as unvoiced or silence.

In **Fig. 6-(a)** two consecutive uncompressed frames, the 70^{th} frame and 71^{st} frame, are shown. **Fig. 6-(b)** shows the resulting compressed frame. First, the 70^{th} frame is obtained from the signal frame buffer. *N* samples from the 71^{st} frame are selected as reference segment X. The most similar segment Z to the segment X is detected in the 70^{th} frame by looking for the minimum of the normalized cross-correlation measure. A smooth estimate of segment X' is generated by using synchronized overlap & add (SOLA) between segment Z and segment X. Finally, segment P, encompassing the trailing part of the 70^{th} frame from sample "*a*" and the heading part of the 71^{st} frame up to sample "*b*," has been time compressed to the segment X'.

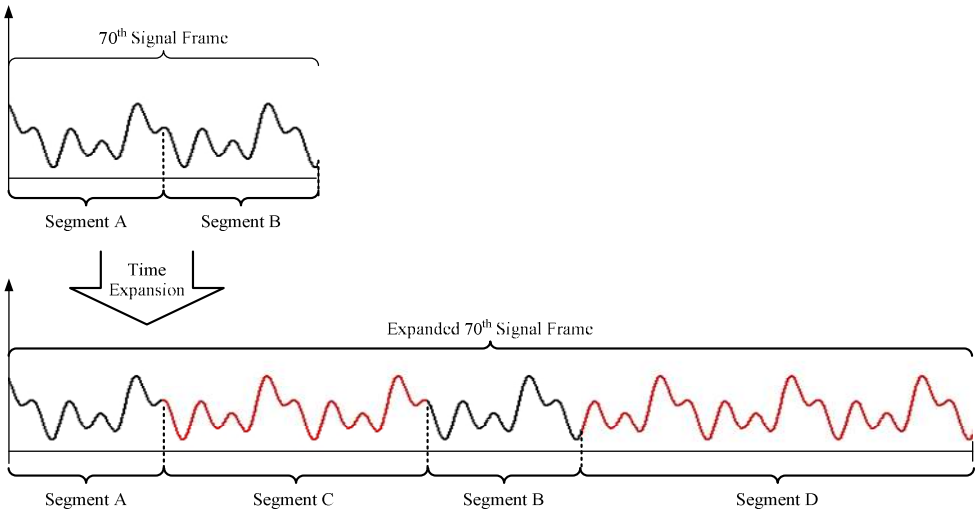
• *Time expansion*

At a sudden increase of the transmission delay, there is the risk that an underflow of the jitter buffer may occur. That is, no data packets are available in the jitter buffer at the required time of decoding to yield the signal frame for continued playback. In this case, the signal frame existing in the signal frame buffer is expanded, and the receiver can effectively eliminate the packet loss by increasing network delay. This paper's time expansion tries to stretch a frame length to triple length.

Fig. 7 shows a diagram illustrating a trailing part of a signal frame to be time expanded.



(a). Original signal frame



(b). Expanded signal frame

Fig. 7. Time expansion of two signal frames

In **Fig. 7-(a)** two consecutive unexpanded frames, the 69th frame and the 70th frame, are shown. **Fig. 7-(b)** shows the resulting expanded frame. First, the 70th frame is divided into two parts, such as segment A and segment B. The most similar segment B' to segment B is searched in the 69th frame while the most similar segment A' to segment A is searched in the 69th frame by looking for the minimum of the normalized cross-correlation measure. Segment C encompasses the trailing part of the 69th frame from sample "l_A" and segment A of the 70th frame, while segment D encompasses the trailing part of the 69th frame from sample "l_B" and the 70th frame. Segment C and segment D are used as expansion segments, as shown in **Fig. 7-(b)**.

4.3 Playout-time Computation

The basic operation of the playout process is as follows. When a new packet $PA_{i,k}$ arrives, its network delay $n_{i,k}$ is obtained from the real-time transport protocol (RTP) header it carries. Actual network delay $n_{i,k}$, together with past delays, is taken into consideration to estimate delay $n_{i,k+1}$ and playout time $p_{i+1,k}$ of the next incoming packet. The current i^{th} signal frame is then scaled by the process of timing recovery immediately. If the delay of the next packet is correctly estimated, the next packet should arrive before the last sample of the current packet is played. Usually, packets are scaled to either retard the speech when the network delay is increasing, or speed up the speech when it is decreasing. It is important to note that this scheme introduces some additional delay due to signal processing.

In the proposed method, the estimate for the network delay $En_{i,k}$ is computed to be:

$$En_{i,k} \cong n_{i,k+1} = \mathbf{w}_{i,k}^{T}\mathbf{n}_{i,k} \tag{7}$$

where $En_{i,k}$ is the predicted network delay value for the i^{th} packet, $w_{i,k}$ is the $T\times 1$ vector of adaptive filter coefficients of the Enhanced normalized least mean squares (NLMS) algorithm [23], $(\,)^T$ is the vector transpose, and $n_{i,k}$ is $T\times 1$ vector containing the past T network delays (up to and including the delay for packet $(i-1)$). The filter tap weights $w_{i,k}$ are then updated after

each packet using the Enhanced NLMS algorithm:

$$\mathbf{w}_{i,k+1} = \mathbf{w}_{i,k} + \frac{\mu}{\mathbf{n}_{i,k}^T \mathbf{n}_{i,k} + a} \mathbf{n}_{i,k} e_{i,k} \tag{8}$$

where μ is the step size, a ($0< a <1$) is a small constant to prevent division by zero, and the estimation error $e_{i,k}$ is given by $e_{i,k} = En_{i,k} - n_{i,k}$.

Table 1 lists the empirical values of the parameters used in the implementation of the Enhanced NLMS algorithm.

Table 1. Enhanced NLMS parameter values

Parameter	Values
$\mathbf{w}_o$	$(10\ldots0)^T$
T	20
μ	0.001

The playout times are then adjusted as:

$$p_{i+1,k} = En_{i,k} + R_{i,k} + \eta \cdot \mathrm{J}_{i,k} \tag{9}$$

where $R_{i,k}$ is the timing recovery delay, and η ($0< \eta <3$) controls the additional buffering delay and lateness loss ratio.

5. Experimental Results

5.1 Testbed Infrastructure and Measurements

In order to evaluate the algorithm, a testbed, as shown in **Fig. 8,** is set up. The testbed consists of three modules, which are session initiation protocol (SIP) signaling, audio data transport, and network traffic emulator.

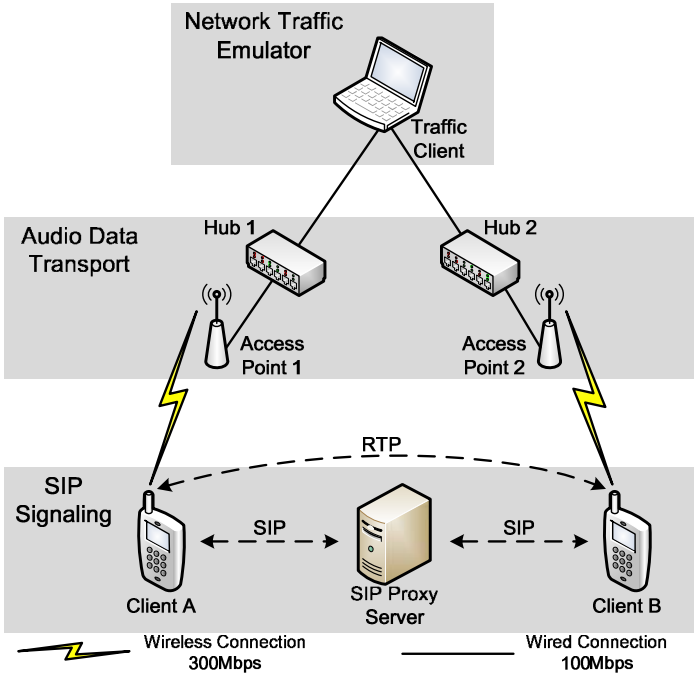


Fig. 8. Test bed infrastructure

These are connected to each other by two types of networks: ethernet (100 Mbps) and wireless local area network (WLAN) (300 Mbps). SIP signaling involves a SIP Proxy Server and two *clients*. VoIP application is developed by using Visual C++ and is installed in *clients*. *Clients* are mobile devices that have the following specifications: 800 MHz CPU, 4 GB memory, and Windows Mobile 6.1 Professional Edition operating system (OS). SIP signaling messages are transferred through the audio data transport module and are sent from clients to the SIP Proxy Server, and then the SIP signaling messages redirected to clients accordingly. Clients send audio data in RTP packets. In the audio data transport module, Clients A and B are each connected to access points (access points 1 and 2) of ipTime N604M (Hubs 1 and 2). In the network traffic emulator module, a traffic generator is used in order to simulate WLAN connections with different traffic loads such as delay, jitter, and packet loss. The traffic client receives the RTP packets via Access Point 1 and sends it to Access Point 2 with the traffic load.

The speech samples used for the experiments are spoken by various male and female speakers and are sampled and digitized at 8 kHz or 16 kHz. Each trace lasts for approximately five minutes, consists of 15,000 packets, and each packet consists of 20 ms of speech content.

To evaluate the quality degradation, objective voice quality testing using PESQ is performed. PESQ is a recognized method of accurately testing the level of quality that will be perceived by the user of a VoIP network, and it is described in the latest ITU-T recommendation P.862, Amendment 2 [24]. PESQ provides a score in the range of 1 to 5, where 1 is *unacceptable* and 5 is *excellent*. A typical range for VoIP is 3.5 to 4.2.

5.2 Performance Comparison

This paper compares the performance of the proposed method with three methods. The three methods used have been modified from the contents of reference papers and implemented.

Method 1 is based on the adaptive gap-based algorithm [9] incorporated with spike detection [6], while Method 2 is based on an adaptive NLMS playout algorithm with delay spike detection [2]. In Method 3, a timing recovery and loss substitution method [21] is combined with modeling the statistics of the interarrival times with the K-Erlang distribution [13]. Four methods commonly incorporate the packet loss concealment [17].

For the performance comparisons, three tests are performed: jitter estimation based on spike detection, total loss rate vs. average buffering delay, and average buffering delay vs. PESQ.

5.2.1 Comparison Results of Jitter Estimation based on Spike Detection

Traces of network delay measurements are characterized by occurrences of large delay spikes (e.g., maximum jitter over 250 ms). The four network conditions are listed in **Table 2**. In **Table 2**, average of the network delay, standard deviation (STD) of the network delay (which reflects the jitter characteristics for each condition), and maximum jitter (which is the difference between the maximum and minimum delay in the short trace) are depicted. Because we want to focus on the effect of jitter estimation based on spike detection in this section, four network delay traces with the extreme maximum jitter over 250 ms are chosen from the Internet links of the testbed infrastructure. And the network traces do not carry any network packet loss.

Table 2. The statistics of network traces

Trace	End-to-end network delay (ms)	STD of network delay (ms)	Maximum jitter (ms)	Network packet loss (%)
A	49.29	26.02	295	0
B	42.17	57.75	392	0
C	48.79	31.97	374	0
D	47.17	34.77	342	0

The jitter estimation performance of the proposed algorithm, Method 1, Method 2, and Method 3 is compared. **Table 3** shows the experimental results on jitter estimation error and late loss rate for each trace. PM denotes the proposed algorithm. As shown in **Table 3**, the proposed jitter estimation based on spike detection achieves smaller jitter-estimation errors and late loss rate overall than Method 1, Method 2, and Method 3.

Table 3. Performance of the jitter estimation

Trace	Method	Jitter estimation error(ms)	Late loss rate (%)
A	PM	34.4	1.06
	Method 1	68.6	2.37
	Method 2	51.8	1.73
	Method 3	60.5	2.02
B	PM	46.1	1.87
	Method 1	86.9	3.22
	Method 2	67.5	2.53
	Method 3	77.3	2.89

C	PM	36.9	1.46
	Method 1	74.6	2.55
	Method 2	55.8	2.11
	Method 3	66.2	2.43
D	PM	34.3	1.30
	Method 1	65.1	2.39
	Method 2	50.3	1.86
	Method 3	58.2	2.13

Fig. 9 shows the results of the jitter estimation by the four methods on Trace A. The black line denotes the network delay of the arriving packets. The blue line marked with the symbol 'x'; the black line marked with the symbol '•'; the black line marked with the symbol '▲'; the red line marked with the symbol '○'represents the estimated network jitter by Method 1, Method 2, Method 3, and the proposed algorithm, respectively.

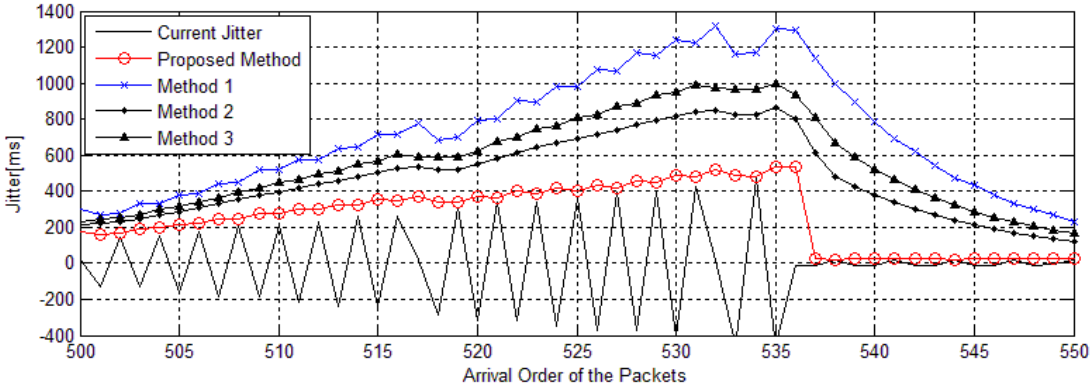


Fig. 9. Jitter estimation results of the proposed algorithm, Method 1, Method 2, and Method 3.

As shown in **Fig. 9**, the proposed algorithm can estimate the network jitter more correctly than Method 1, Method 2, and Method 3 during spike intervals, thereby effectively reducing the jitter estimation error.

5.2.2 Comparison Results of Total Loss Rate vs. Average Buffering Delay, and Average Buffering Delay vs. PESQ

For the performance comparisons of total loss rate vs. average buffering delay, and average buffering delay vs. PESQ, four network delay traces obtained from the Internet links of the testbed are used and are listed in **Table 4**, showing that the maximum jitter increases from 44 ms to 371 ms. Compared with **Table 2**, the network packet loss is included in three traces for proving the efficiency of the timing recovery combined with the jitter estimation. Traces 2 and 3 carry an approximately 2% loss, while Trace 4 includes a 4% loss.

Table 4. Statistics of network traces

Trace	End-to-end network delay (ms)	STD of network delay (ms)	Maximum jitter (ms)	Network packet loss (%)
1	30.74	7.25	44	0

2	66.19	19.29	124	2
3	32.07	11.88	146	2
4	78.22	31.22	371	4

Fig. 10 shows the late loss rate vs. average buffering delay using different algorithms for the four traces. M1, M2, M3, and PM denote Method 1, Method 2, Method 3, and the proposed method, respectively. Although our simulations are based on G.711 voice codec, the algorithms are audio codec independent and can be implemented on the receiver alone.

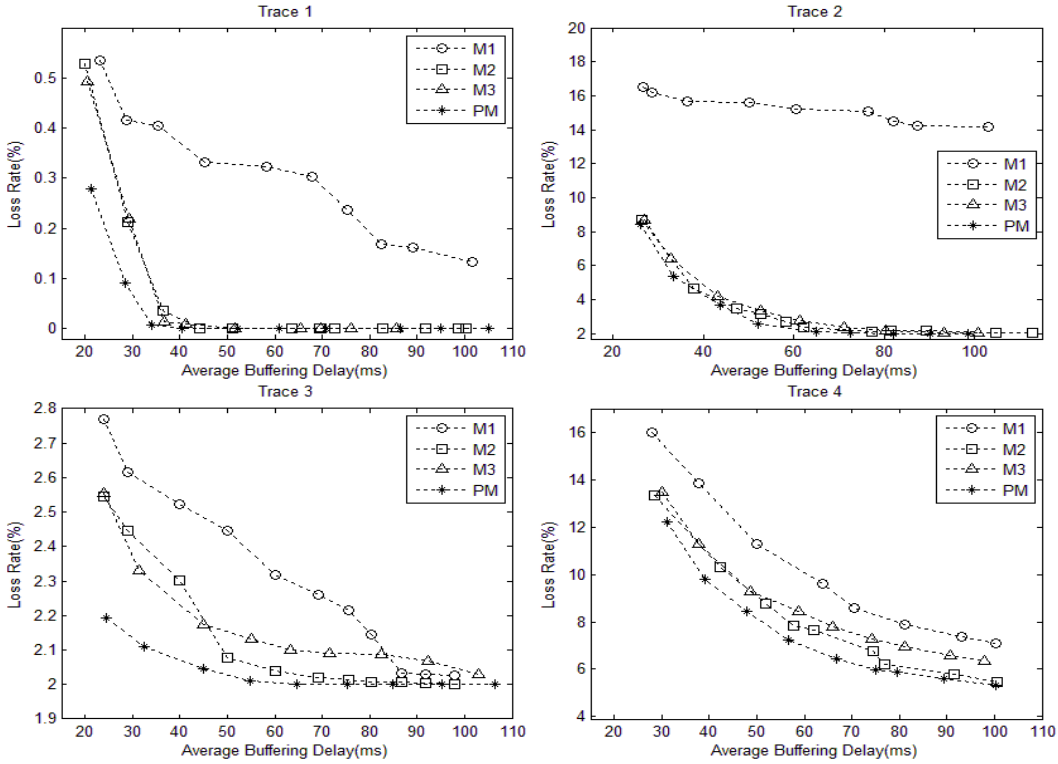


Fig. 10. Performance of timing recovery algorithms

The performance gain of the proposed method over Methods 1, 2, and 3 differs from trace to trace. For Trace 1, the late loss rates of all four methods are very low, under 1%, although the average buffering delay increases. The reason for this is that the STD of network delay is small and the maximum jitter is low. The proposed method (PM) yields the best performance, while the performance of M2 is very similar to M3. On Trace 2, the performance of the PM is slightly better than that of M2 and M3. The worst performance is provided by M1, compared with M2, M3, and PM. For Trace 3, the PM results in the lowest loss rate, but the difference of loss rate between the four methods is small. On Trace 4 with the high maximum jitter, the performance of the PM is significantly better than the other three methods because it results in more accurate jitter estimation.

The performance comparison related to average buffering delay vs. PESQ is illustrated in **Fig. 11**. From **Fig. 11**, one observes that the PM outperforms the reference methods such as M1, M2, and M3, in low jitter, medium jitter, high jitter, 2% packet loss rate, and 4% packet loss rate. The highest PESQ scores above 4.0 are obtained using the PM with above average buffering delay of 30 ms in Trace 1, which does not carry any loss. As the jitter level or packet

loss rate increase, the PESQ scores decrease (the performance difference become more significant, with clear advantage of the PM). PESQ scores of M1 appear to be significantly lower than those of M2 and M3. More importantly, the PM is very suitable for operating at a low buffering delay, handling various loss patterns and maximum jitters more effectively compared to other methods.

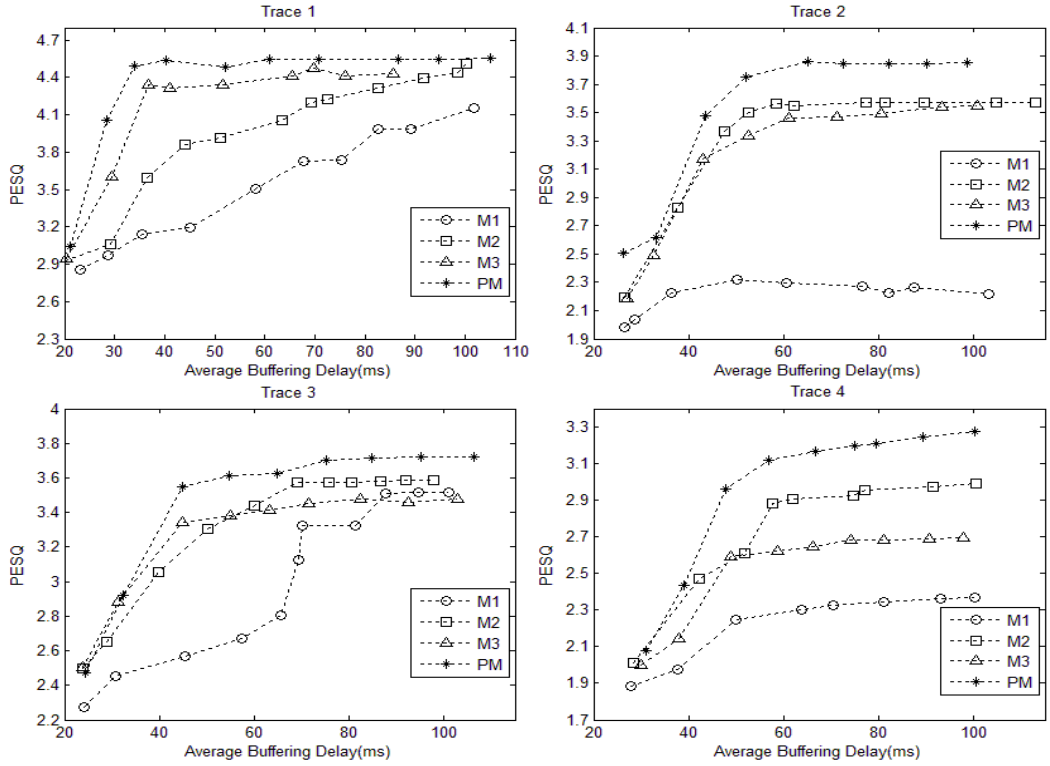


Fig. 11. PESQ score vs. average buffering delay of timing recovery algorithms

6. Conclusion

This paper proposes an enhanced timing recovery algorithm combined with active jitter estimation to improve voice quality. The proposed algorithm overcomes the effect of transmission jitter by way of expanding or compressing each packet according to the estimated predicted network delay and variations more accurately than conventional algorithms. The active network jitter estimation incorporates rapid detection of delay spikes and reacts to changes in network condition, thus the proposed method effectively reduces buffering delay and late loss rate. Additionally, enhanced compressing and expansion algorithms alleviate the metallic artifacts in transition regions and produce high voice quality.

Simulation results based on testbed infrastructure show that the trade-off between buffering delay and late loss can be improved by the proposed method. The proposed method achieves higher PESQ scores than other existing methods and is suitable for any practical mobile VoIP system. In future work, the method proposed here may be applied to advanced teleconferencing applications running on smart-TV.

References

[1] B. Sat and B. W. Wah, "Analyzing voice quality in popular VoIP applications," 2009 IEEE Computer Society, *IEEE Multimedia*, vol.16, pp.46–59, Jan.2009. Article (CrossRef Link)
[2] A. Shallwani, "An adaptive playout algorithm with delay spike detection for real-time VoIP," *Department of Electrical and Computer Engineering McCill University Master Thesis*, vol. 2, pp. 997–1000, Sep.2003. Article (CrossRef Link)
[3] Z. Becvar, L. Novak, J. Zelenka, M. Brada and P. Slepicka, "Impact of additional noise on subjective and objective quality assessment in VoIP," *2007 International Workshop on Multimedia Signal Processing*, pp.39–42, Oct.2007. Article (CrossRef Link)
[4] Z. Becvar, J. Zelenka and M. Brada, "Impact of saturation on speech quality in VoIP," *15th International Conference on Systems*, *Signals and Image Processing*, pp.381–384, Jun.2008. Article (CrossRef Link)
[5] D. Florencio and L.-W. He, "Enhanced adaptive playout scheduling and loss concealment techniques for Voice over IP networks," *2011 IEEE International Symposium on Circuits and Systems*, pp.129–132, May.2011. Article (CrossRef Link)
[6] R. Ramjee, J. Kurose, D. Towsley and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide area networks," *IEEE Infocom Conference on Computer Communication*, vol.2, pp.680–688, Jun.1994. Article (CrossRef Link)
[7] M. Narbutt and L. Murphy, "VoIP playout buffer adjustment using adaptive estimation of network delays," *18th International Teletraffic Congress (ITC-18)*, pp.1171–1180, Sep.2003.
[8] A. Kansal and A. Karandikar, "Adaptive delay estimation for low jitter audio over Internet," *IEEE Global Telecommunication Conference*, vol.4, pp.2591–2595, Nov.2001. Article (CrossRef Link)
[9] J. Pinto and K. J. Christensen, "An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods," *24th Conference on Local computer Networks*, vol.5, pp.224–231, Oct.1999. Article (CrossRef Link)
[10] Y. Liang, N. Farber and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over IP networks," *IEEE Transactions on Multimedia*, vol.5, no.4, pp.532–543, Dec.2003. Article (CrossRef Link)
[11] C. J. Sreenan, J.-C. Chen, P. Agrawal and B. Narendran, "Delay reduction techniques for playout buffering," *IEEE Transactions on Multimedia*, vol.2, pp.88–100, Jun.2000. Article (CrossRef Link)
[12] S. Chi and B. F. Womack, "QoS-based optimal adaptive playout buffer scheduling using the packet arrival distribution," *IEEE MILCOM*, pp.15–19, Oct.2009. Article (CrossRef Link)
[13] H. Li, G. Zhang and W. Kleijn, "Adaptive playout scheduling for VoIP using the K-Erlang distribution," *The 2010 European Signal Processing Conference*, pp.1494–1498, Aug.2010.
[14] J. Aragao Jr. and G. Barreto, "Novel approaches for online playout delay prediction in VoIP applications using time series models," *Computers & Electrical Engineering*, vol.36, pp.536–544, May.2010. Article (CrossRef Link)
[15] P. DeLeon and C. Sreenan, "An adaptive predictor for media playout buffering," *IEEE International Conference on Acoustics, Speech*, *Signal Processing*, vol.6, pp.3097–3100, March, 1999. Article (CrossRef Link)
[16] K. Fujimoto, S. Ata and M. Murata, "Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications," *IEEE Global Telecommunication Conference*, vol.3, pp. 451–2457, Nov.2002. Article (CrossRef Link)
[17] N. Aoki, "A VoIP packet loss concealment technique taking account of pitch variation in pitch waveform replication," *Electronics and Communications in Japan*, vol.89, pp.1–9, Mar.2006. Article (CrossRef Link)
[18] V. P. Bhute and U. N. Sharawankar, "Speech packet concealment techniques based on time-scale modification for VoIP," *ICCSIT 2008 International Conference on Computer Science and Information Technology*, pp.825–828, Aug.2008. Article (CrossRef Link)
[19] J.-H. Chen, "Packet loss concealment for predictive speech coding based on extrapolation of speech waveform," *ACSSC 2007 Conference Record of the Forty-First Asilomar Conference on Signal*, *Systems and Computers*, pp.2088–2092, Nov.2007. Article (CrossRef Link)
[20] K. Kondo and K. Nakagawa, "A speech packet loss concealment method using linear prediction,"

IEICE Transactions on Information and System, vol.E89-D, no.2, pp.806-813, Feb.2006. Article (CrossRef Link)
[21] S. V. Andrsen, W. B. Kleijn, and P. Sorqvist, "Method and arrangement in a communication system," *U.S. Patent 7 321 851*, Sep.2008.
[22] D. Dorran, "Audio time-scale modification," *Dublin Institute of Technology Doctoral Thesis*, May.2005.
[23] S. L. Ng, S. Hoh and D. Singh, "Effectiveness of adaptive codec switching VoIP application over heterogeneous networks," *2nd Conference on Mobile Technology*, *Applications and Systems*, Nov.2005. Article (CrossRef Link)
[24] ITU-T Recommendation P.862(2001) Amendment 2(11/05), http://www.itu.int/rec/T-REC-P.862-200511-I!Amd2

Hyoung-Gook Kim received a Dipl.-Ing. degree in Electrical Engineering and a Dr.-Ing. degree in Computer Science from the Technical University of Berlin, Berlin, Germany. From 1998 to 2005, he worked on mobile service robots at Daimler Benz, speech recognition at Siemens, and speech signal processing at Cortologic, and he served as adjunct professor of the Communication Systems Department, Technical University of Berlin, Berlin, Germany. From 2005 to 2007, he was a project leader at the Samsung Advanced Institute of Technology, Korea. Since 2007, he has been a professor in the Department of Electroncis Convergence Engineering, Kwangwoon University, Korea. His research interests include audio signal processing, audiovisual content indexing and retrieval, speech enhancement, and robust speech recognition.