New Piggybacking Algorithm on G.722.2 VoIP Codec with Multiple Frame Sizes

By

Professor Benjamin W. Wah, PhD. Batu Sat and Wee Hong Yeo

ECE 497/499

SPRING 2009

ABSTRACT

In this paper, we present the design of a new piggybacking algorithm implemented on the G.722.2 VoIP codec. In a piggybacking algorithm, multiple speech frames that include those transmitted in the past are encapsulated in a single packet. Because redundant copies of each frame are transmitted to the receiver, the receiver can recover frames lost when one or more packets are lost or arrive late in transmission. In this paper, we have developed a codec that removes further redundancies in the multiple frames encapsulated in piggybacking by sending only one set of Linear Prediction (LP) coefficients for all subframes encapsulated. We create multiple versions of the codec, each working with a different frame size. Our new codec can encode the multiple frames with little degradation in Perceptual Evaluation Speech Quality (PESQ), while having substantially more bit savings. The performance of the new algorithm is evaluated against the original method of piggybacking over random losses, as well as that using packet traces collected over PlanetLab.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 General Description of Speech Codecs

A speech codec essentially consists of a coder and a decoder. Its objective is to convert analog speech signals into compressed digital form for efficient transmission over the data network, without an excessive loss in quality. With the widespread usage of Voice-over-IP (VoIP) applications such as Skype, GoogleTalk, and phones equipped with VoIP capabilities, there is a greater need for a detailed study of existing standards in speech codecs used for this purpose. A more efficient and robust codec will achieve a higher quality of service under unreliable network conditions and limitations in bandwidth.

1.2 G.722.2 AMR-WB codec

The G.722.2 is a recommendation for wideband coding of speech at around 16 kbits/s using Adaptive Multi-Rate Wideband (AMR-WB), approved in January 2002 by the Telecommunication Standardization Sector of International Telecommunication Union (ITU-T). AMR-WB/G.722.2 was adopted at ITU for teleconferencing and voice-over-packet applications. It is the mandatory standard codec for wideband speech in Global System for Mobile communications (GSM) and Wideband Code Division Multiple Access (WCDMA) networks. The codec supports a wide range of bit rates from 6.6 to 23.85 kbps, using lower bit rates during network congestion or degradation while preserving audio quality.

The codec is based on the code-excited linear-prediction (CELP) coding model. It maps input blocks of 320 speech samples for a 20-ms frame in 16-bit uniform PCM format to encoded blocks of 132, 177, 253, 285, 317, 365, 397, 461 and 477 bits, giving us possible bit rates of 6.60, 8.85, 12,65, 14.25, 15.85, 18.25, 19.85, 23.05 and 23.85 kbps [1].

For each frame of 20 ms, the coder first performs down-sampling, bringing the sampling rate from 16 kHz to 12.8 kHz, as the encoder works at that sampling rate. The samples are then high-pass filtered with a cut-off frequency of 50 Hz. Linear-prediction (LP) analysis is then performed once per frame to find the set of LP parameters, which are coefficients to the impulse response of the filter. The set of LP parameters is converted into immittance-spectrum pairs (ISP) and vector quantized using split-multistage vector quantization (S-MSVQ). The frame is divided into 4 subframes of 5 ms each with 64 samples per subframe. An open-loop pitch lag is estimated every other subframe or once per frame based on the perceptually weighted speech signal.

For each subframe, the LP residual signal is then encoded via an algebraic codebook and an adaptive codebook. The algebraic codebook is based on interleaved single-pulsed permutation (ISPP) design, while the adaptive codebook essentially involves finding the closed-loop pitch period of the signal. The gains of each of these codebooks are found by determining how much each contributes to the overall signal and vector quantified with 6 or 7 bits.

For decoding, the excitation signal is reconstructed from the two codebooks, based on their respective gains. It is then placed through the inverse LP filter and up-sampled to provide the output speech. Figure 1.1 shows the structure of the G.722.2 encoder.

In this paper, we evaluate the performance of the decoded speech using PESQ [2] which compares the original speech to the output speech from the decoder using two sample speech files, one with male voice and the other with female voice. PESQ evaluates the output speech on a scale of 0 to 4.5, with 4.5 being the closest to the original speech.

1.3 Piggybacking

Even though the codec has shown itself to provide a high-quality encoding of speech and has some robustness to losses at the bit level, it does not incorporate features to deal with losses at the packet level in the communication channel when frames are encapsulated in packets. At the packet level, one simple method is to piggyback multiple consecutive frames in one packet [3]. These consist of the current frame to be transmitted and a copy of those frames that have been transmitted in the past, up to some fixed limit. When a packet is lost or arrives late at the receiver, the receiver can recover the lost or late frame from another packet that will arrive later. The idea works in the current Internet because each frame only takes a small part of the bits in a packet, and the packet size is relatively large to accommodate multiple frames, without affecting the loss or delay of the packet.

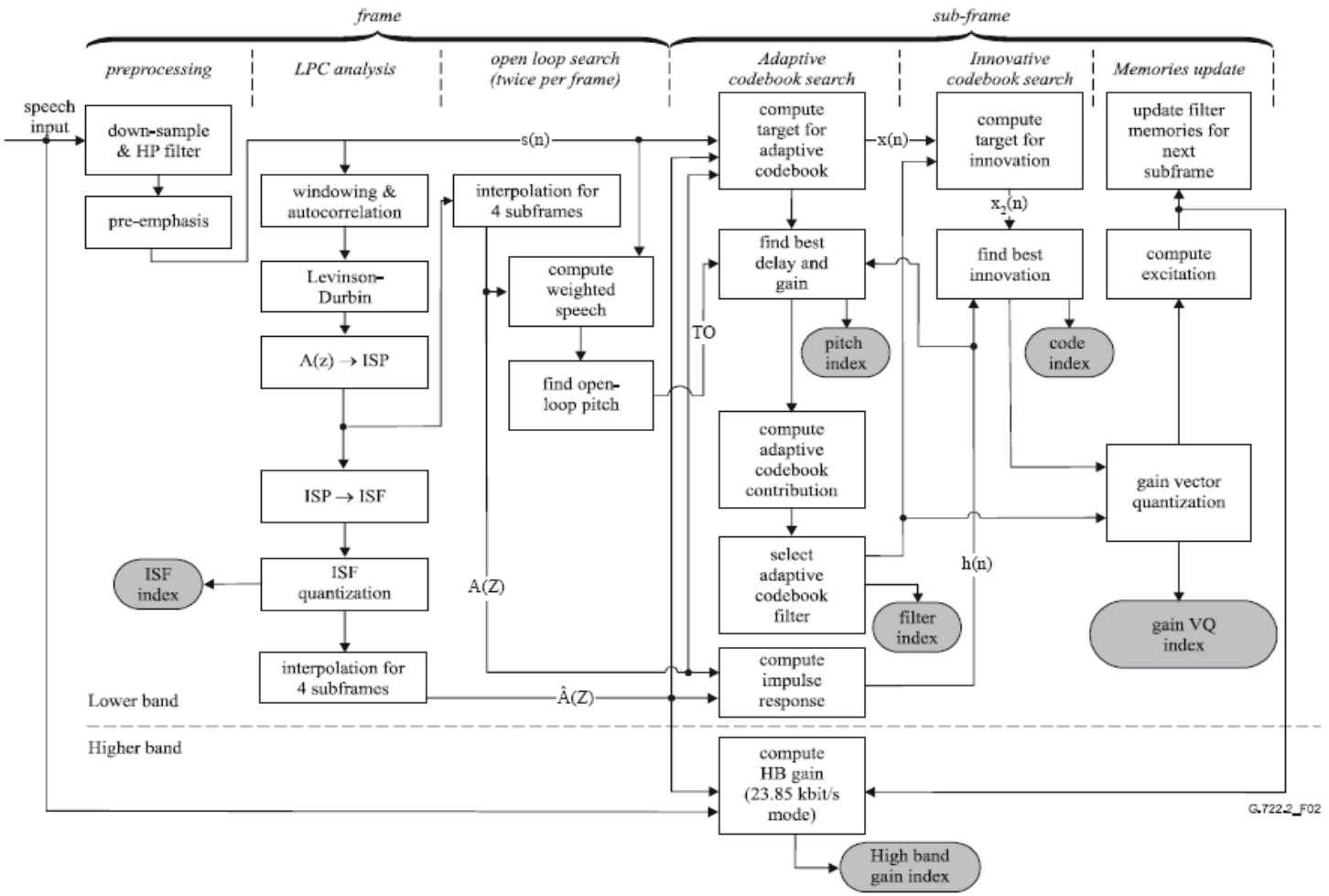

Figure 1.1 Detailed Block Diagram of the ACELP encoder

*diagram from ITU-T Recommendation G.722.2 (07/2003)

1.4 Problem Statement

Because multiple frames are now placed in the same packet, the original coding algorithm that removes the redundancy from each frame is no longer optimal. Further bit savings can be achieved by analyzing the multiple frames encapsulated in one packet and by removing their redundancies. In this paper, we present the modifications to the G.722.2 codec by using a common set of LP coefficients for all the frames encapsulated in a packet. We design the codec to allow it to work with different frame sizes for speech samples encapsulated, while allowing many new possible configurations of piggybacking. With the new configurations, our piggybacking algorithm can merge multiple frames in a packet into one larger frame by achieving high savings in bit rate while incurring little loss in speech quality.

2. COMBINING 20-MILLISECOND FRAMES INTO LARGER FRAMES

2.1 Motivations

The G.722.2 codec currently only works with a frame size of 20 ms. It provides 9 possible modes of operation and a multitude of bit rates, ranging from 6.6 kbps to 23.85 kbps that may be changed at frame boundaries.

The codec was designed with the goal that each frame is transported as a unit in the network. This constant frame rate was designed for time-division multiplexed networks which provide time slots of constant size at periodic intervals for transmitting frames.

In contrast to time-division multiplexed networks, the end-to-end delay of an IP network connection is not constant. To guarantee that frames are played at a constant rate every 20 ms at the receiver, jitter buffers are used at receivers to smooth out irregular arrivals. The constant delay between the mouth of the speaker to the ear of the receiver is called the mouth-to-ear delay (MED) [4] defined as follows:

$$\begin{aligned} \text{MED} = {} & \text{frame size} * \text{frames per packet} \\ & + \text{processing time} + \text{transmission time} \\ & + \text{jitter buffer} + \text{playout delay} \end{aligned} \tag{2.1}$$

Another variation from time-division multiplexed networks is that a packet rate of 50 packets per second (20 ms between packets) may be too high for some Internet connections, especially when heavy traffic and lossy conditions. As a result, multiple

frames may be combined and placed into one packet. Another reason for placing multiple frames into the same packet is to provide redundancy in piggybacking. This can be done as long as the end-to-end delay for each frame does not exceed the MED allowed.

When multiple frames are placed in the same packet, it may be beneficial to code them in such a way that eliminates further redundancies among the frames and to achieve better bit savings because there are no fractional packet losses. This can be done by increasing the frame size by having more subframes in one frame while keeping the subframe size constant. All the subframes share a common set of LP parameters that are applicable to the larger frame. This leads to fewer bits than the original configuration with multiple 20-ms frames, as more subframes use the same LP parameters. For example, for a packet with 40-ms speech, we need only one set of LP parameters for the 40 ms frame, instead of two sets for the two 20-ms frames.

2.2 Code Modifications

Below is the list of files where code modifications are needed to be carried out in the G.722.2 Codec and the various variables in each file that needs to be modified.

- Header Files
  - cnst.h
    - L_FRAME16k
    - L_FRAME
    - NB_SUBFR
    - L_WINDOW
    - L_TOTAL

    - bits.h
        - NBBITS_7k
        - NBBITS_9k
        - NBBITS_12k
        - NBBITS_14k
        - NBBITS_16k
        - NBBITS_18k
        - NBBITS_20k
        - NBBITS_23k
        - NBBITS_24k
    - wb_vad_c.h
        - FRAME_LEN
- Source files
    - cod_main.c
        - interpol_frac[NB_SUBFR]
        - if statement to match number of subframes
    - dec_main.c
        - interpol_frac[NB_SUBFR]
        - if statement to match number of subframes
    - int_lpc.c
        - match number of iteration to number of frames
- Table files
    - homing.tab
        - DHF_PARMS_MAX
    - ham_wind.tab
        - generate the table to match new window size

2.3 Results

Figure 2.1 shows the points representing the original possible configurations in shaded circles. It also shows all the points that represent the new configurations, with frame sizes ranging from 30 ms to 100 ms, together with an envelope (solid-line), which represents configurations that dominate all other configuration with either lower PESQ or lower bit rate. It shows new configurations introduced in the envelope, especially those with lower bit rates that were previously not possible. Note that although many of the newly created configurations lie below the envelope, they are essential in the new piggybacking algorithm described in the next section.

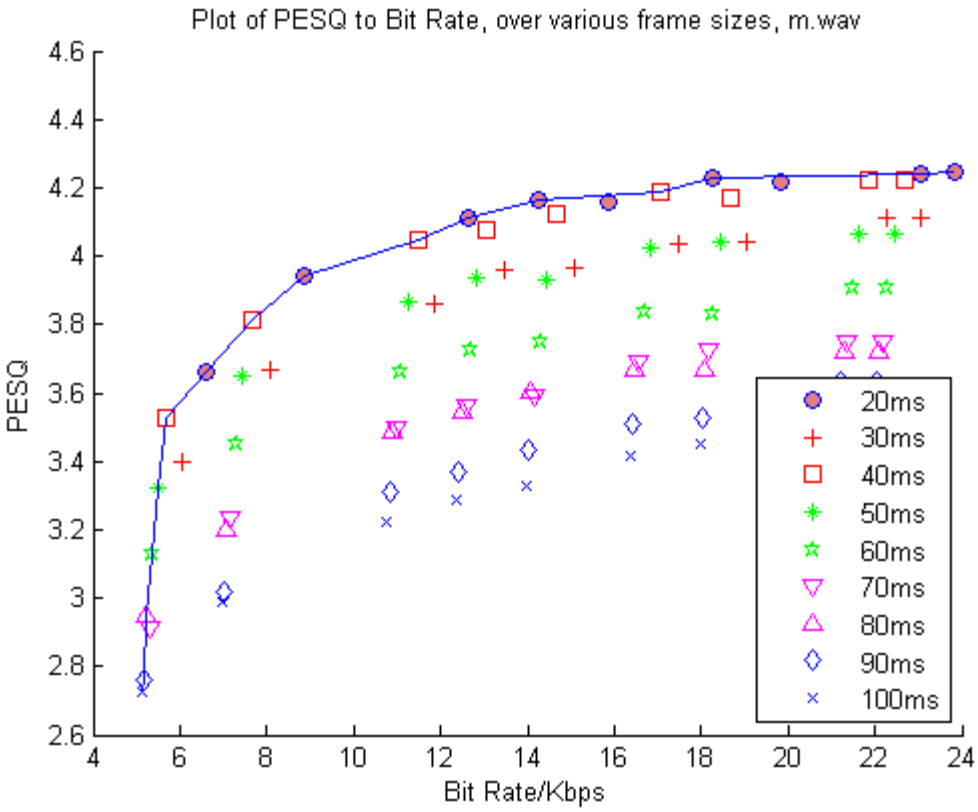

Figure 2.1 New Configurations

3. PROPOSED PIGGYBACKING ALGORITHM

Piggybacking packs copies of previously generated speech frames, together with the current speech frame, into the current packet. This is possible because each frame only occupies a fraction of the bit budget of a packet. By providing redundant copies of a frame in multiple packets, we can overcome some packet losses. For example, with piggybacking degree of two, a frame will be found in packets x and $x+1$. If packet x is lost, then the frame is still recoverable from packet $x+1$ provided that it is not lost.

By increasing the piggybacking degree, we can increase the amount of losses that the connection can tolerate. However, the amount of piggybacking cannot be indefinitely increased, due to constraints on the number of bits available and the MED. A frame is considered lost if it is not available from any of the packets that are received before its scheduled playout time determined by the MED. Hence, it is clear that there is no reason to increase the number of frames in a packet past the MED, as those frames will be considered lost anyway.

The current piggybacking algorithm simply packs multiple consecutive frames into a packet. However since these frames are close in time to each other, it may not be necessary to have separate LP parameters for each. Our new algorithm strives to exploit the temporal locality of these adjacent frames by merging them into a longer frame and by having more subframes using the same LP parameters. Our approach can achieve better bit savings with little degradations in PESQ.

3.1 Algorithm at the Encoder

Assuming 20-ms frame size with piggybacking degree 3, consider three coder streams, where dashes represent empty frames and a number represents the frame ID.

1) − − 1 2 3 4 5 6 7 8 9 A B C D . . .

2) − 1 2 3 4 5 6 7 8 9 A B C D E . . .

3) 1 2 3 4 5 6 7 8 9 A B C D E F . . .

Assume that three 20-ms frames in each packet are now encoded into one 60-ms frame. These packets are sent in the order of Streams 1, 2 and 3. The receiver will receive the following stream of packets, each containing one 60-ms frame:

(− − 1), (− 1 2), (1 2 3), (2 3 4), (3 4 5), (4 5 6), (5 6 7), (6 7 8), (7 8 9), (8 9 A), (9 A B), (A B C), (B C D), (C D E), (D E F), . . .

Because the encoding of each frame depends on the previous frames, it is obvious that the above sequence of frames has to be encoded by three coders. For example, one coder will be responsible for coding the frames (− − 1), (2 3 4), (5 6 7), (8 9 A), (B C D), . . . The number of coders is equal to the piggybacking degree in order to maintain the proper internal state in each stream.

3.2 Algorithm at the Decoder

At the receiver, the decoder will split the input stream back into three separate streams and decode them separately. The outputs from the various streams will be merged in order to create a single stream of speech output. The decoder tries to predict what are in the lost frames. Hence, if three or more consecutive losses occur, it will use the predicted frame generated by the decoder, as it will not be able to recover that frame from the other streams. Note that what is written to the output is after the decoding process. The decoder will process all the frames in the packet but only outputs the relevant frame.

The following shows the pseudocode of the decoder algorithm.

Algorithm 1 Decoder algorithm for piggybacked packets

```
1: if packet is lost then
2: try to recover the current frame from later packets
3: if unrecoverable then
4: output estimated speech frame
5: end if
6: else
7: output current speech frame
8: plus any other frames that need to be recovered
9: end if
```

3.3 Quality vs. Bit-Rate Trade-offs under Random Losses

We tested the following configurations: 20-ms frames with piggybacking degrees 2, 3, 4 and 5, respectively; 30-ms frames with piggybacking degrees 2 and 3, respectively; 40-ms frames with piggybacking degree 2; and 50-ms frames with piggybacking degree 2. We tested the two methods of piggybacking using two benchmarks: one with male voice and the other with female voice. The tests were done over all the configurations. Random losses of 5% to 30% were injected into the packet stream to see how the new algorithm fares against the old one under various loss rates.

Figure 3.1 summarizes our results. Each graph compares the PESQs (and bit rates) between the original and the new configurations computed as follows:

For each of the configurations, we calculate:

$$PESQ\_ratio = \frac{PESQ\_new}{PESQ\_old}$$
$$Bitrate\_ratio = \frac{Bitrate\_new}{Bitrate\_old} \quad (3.1)$$

PESQ_new refers to the PESQ achieved using the new algorithm and PESQ_old refers to the PESQ achieved using the old algorithm. The same applies for bitrate ratio.

To achieve good trade-offs, we would like to have the PESQ_ratio close to 1 and the Bitrate_ratio as small as possible. Each plot also shows a line of unit gradient. Points above the line have a larger PESQ_ratio than Bitrate_ratio and represent those configurations where good trade-offs are achieved.

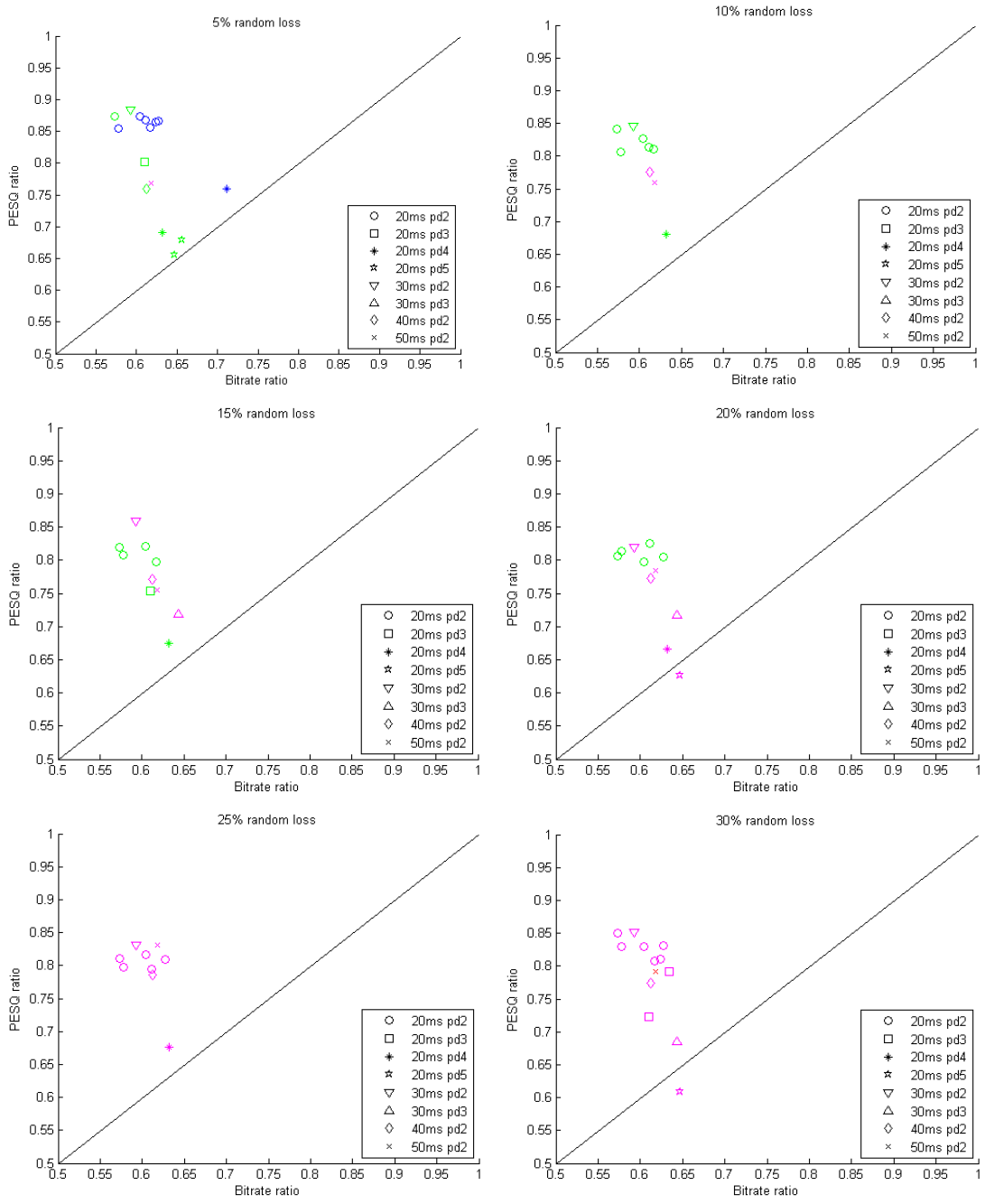

Figure 3.1 Random Loss Test Results, m.wav

The plots clearly show that over the various error rates, we are able to achieve good trade-offs between bit savings and loss in PESQ by using the new piggybacking algorithm. To show the perceptual quality of the various configurations, color codes explained in Table 3.1 are used to classify the perceptual difference of all points with respect to the PESQ of the original configurations.

Table 3.1 Difference in Perceptual Quality with Respect to the Original Configuration

Label	**PESQ Range**	**Color**
No Difference	> 3.8	Not Applicable
Just Noticeable Diff.	3.8 – 3.2	Blue
Acceptable	3.2 – 2.5	Green
Tolerable	2.5 – 1.7	Magenta
Intolerable	< 1.7	Red

The results for female voice are similar but with slightly lower PESQ. This difference is likely because the female voice tends to have more high-frequency components and the input speech is high-pass filtered in the pre-processing stage.

4. ESTIMATING MED FOR PIGGYBACKING

The success of piggybacking to help conceal losses or late arrivals depends on a good estimate of MED defined in Equation (2.1) and the appropriate piggybacking degree. For a given piggybacking degree, if the estimated MED is accurate, then PESQ will not appreciably improve as the actual MED is increased beyond the estimated MED. In contrast, if the estimated MED is too short, then PESQ will improve as the actual MED is increased.

Referring to Equation (2.1), the only variable not under the control of the VoIP system is the transmission delay from the sender to the receiver. With a good estimate of this delay, we can calculate the MED and thus determine the piggybacking degree. In practice, it is hard to accurately estimate this delay because the clocks at the sender and the receiver are not synchronized. However, assuming that the transmission delay is stationary over time, we can estimate the average variation of the transmission delays of the first few packets with respect to the arrival time of the first packet and incorporate this delay into the jitter-buffer delay. Our algorithm for estimating the jitter-buffer delay in Equation (2.1) is:

$$\text{Jitter-buffer delay} = \text{average variation of arrival times of the first } x \text{ packets with respect to the first packet} + \text{jitter tolerance} \quad (4.1)$$

To empirically determine the jitter tolerance in Equation (4.1), we collect UDP packet traces from PlanetLab from various sources to various destinations. The details of the traces used can be found elsewhere [4], [5]. We varied the jitter tolerance in Equation (4.1) from 25 ms to 275 ms in 50-ms increments, while ignoring processing delay, and set x to 10.

4.1 Trace Test Results

Figure 4.1 shows the quality of the received speech in one of the tests. We evaluated all the possible configurations discussed in the last section with respect to a trace originating from California with destination Germany. The trace is 5 min long with a 60-ms period and a payload of 100 bytes per packet [5]. The configurations vary in frame sizes, modes, piggybacking degree, and piggybacking method. The plot depicts the envelope of the configurations at that estimated MED, where each point represents a possible configuration.

As in Figure 2.1, these points show the maximum achievable PESQ at the various bit rates over all possible configurations. At an initial MED estimate of 135 ms, the line does not appear in the graph because there are no points present in the envelope. Because this MED is too short, most of the packets are dropped. The first line in the plot appears after we added 50 ms to the initial estimate. Subsequent lines represent MEDs at 50 ms intervals. It can be observed that the envelope does not increase significantly further after adding 100 ms and saturates shortly after that.

In total, we perform simulations on over 100 traces with varying sources and destinations, such as China, Taiwan, the United States and the United Kingdom. These traces vary in duration between 5 min and 10 min, with packet period of 30 ms or 60 ms. They are also collected over various times of the day to provide us with a good sample of Internet traffic throughout the day [4],[5].

Similar results in other traces show that a 100-ms jitter tolerance is adequate in all the cases tested.

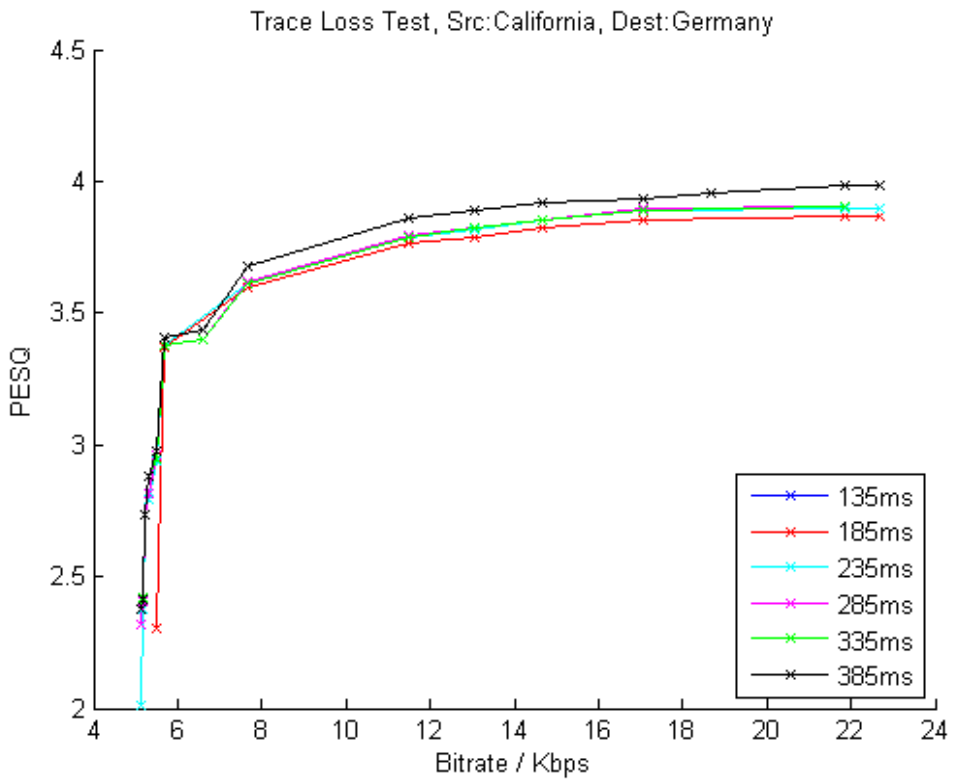

Figure 4.1 Trace Test Plot, m.wav

Through various simulations with random losses and PlanetLab packet traces, Table 4.1 shows some configurations that consistently offer good performance. These configurations allow one to be assured of high performance, without the need of dynamically changing their settings, over varying network conditions.

Table 4.1 Recommended Configurations

Frame Size/ms	**Piggybacking Degree**	**Bitrate/kbps**
20	2	11.35, 15.35, 22.95
30	2	10.733
40	2	10.425
50	2	10.240

5. CONCLUSIONS

In this paper, we have presented modifications to the speech codec as well as algorithms for estimating the mouth-to-ear delay when multiple speech frames are piggybacked in a packet before they are sent to the receiver in a VoIP connection. We have modified the G.722.2 codec in order to allow it to work with various new frame sizes from 30 ms to 100 ms. By utilizing these new frame sizes, we have created a new piggybacking algorithm that offers savings in bit rate with little degradation in speech quality. Our evaluations using random losses as well as packet traces from PlanetLab show that our proposed piggybacking algorithm offers good trade-offs over various error rates.

REFERENCES

[1] *Wideband coding of speech at around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)*, ITU-T Std. G.722.2, 2003.

[2] Perceptual evaluation of speech quality (PESQ): *An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, ITU-T Std. P.862, 2001.

[3] A. M. Kondiz, *Digital Speech, coding for low bit rate communication systems*, 2nd ed. West Sussex, England: Wiley, 2004.

[4] B. Sat, Z. X. Zhuang, and B. W. Wah, "The design of a multi-party VoIP conferencing system over the internet," in *Proc. IEEE International Symposium on Multimedia*, Dec. 2007, pp. 3–10.

[5] B. Sat and B. W. Wah, "Playout scheduling and loss-concealments in VoIP for optimizing conversational voice communication quality," in *Proc. ACM Multimedia*, 2007.