New Piggybacking Algorithm In VoIP Using Enhanced G.722.2 Codec With Larger Frames

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

Department of Electrical and Computer Engineering and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
{wyeo2, batusat, wah}@illinois.edu

***Abstract*—In this paper, we present the design of a new piggybacking algorithm for VoIP implemented using the G.722.2 codec. In piggybacking, 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 those lost frames when one or more packets are lost or arrive late in their transmission. In this paper, we have enhanced the G.722.2 codec by removing further redundancies in the multiple frames encapsulated in piggybacking and by having only one set of LP coefficients for all the subframes encapsulated. We create multiple versions of the codec, each using a different frame size. Our new codec can encode the multiple frames with little degradation in PESQ, while having substantial bit savings. Its performance is evaluated against the original method of piggybacking over random losses, as well as that using packet traces collected in the PlanetLab.**

I. Introduction

The G.722.2 codec [1] is an International Telecommunication Union (ITU-T) standard for wideband speech coding. Also known as the Adaptive Multi-Rate Wideband (AMR-WB) codec, it has been deployed in various 3G Networks. It is the mandatory standard codec for wideband speech in the widely used GSM and WCDMA networks. It also interoperates with the latest 3GPP2 wideband standard [7].

The codec is based on the algebraic code-excited linear-prediction (ACELP) coding model. It maps input blocks of 320 speech samples for a 20-ms frame in the 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 or 23.85 kbps. Each of these possible bit rates is defined as an *operating mode* of the codec, with Mode 0 having the lowest bit rate for this frame size and Mode 8 having the highest bit rate [2].

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. 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, each of 5 ms and with 64 samples. An open-loop pitch is estimated for 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 and an adaptive codebook. The gains of each are found by determining how much each contributes to the overall signal and vector quantified by 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. In this paper, we evaluate the performance of the decoded speech using PESQ (perceptual evaluation speech quality) [3] 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.

Even though the codec has shown itself to provide 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 [4]. 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 those lost or late frames from another packet that will arrive later. The idea works well in the current Internet because each frame only takes a small part of the payload in a packet, and the packet size is relatively large to accommodate multiple frames, without affecting the loss or delay of the packet.

Because multiple frames are placed in the same packet, the original coding algorithm that removes the redundancy within each frame is no longer optimal with respect to those frames in the packet. Further bit savings can be achieved by removing the redundancies among the multiple frames encapsulated.

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 piggybacked in a packet. We design the codec to allow it to work with different frame sizes for the speech samples encapsulated, and facilitate many new possible configurations of piggybacking. Our design, however, is not limited to G.722.2 and will work with any codec with a similar frame structure, such as G.729.1 and G.723.1.

Research supported by National Science Foundation Grant CNS 08-41336.
978-1-4244-4652-0/09/$25.00

II. Combining 20-ms Frames into Larger Frames

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 is 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 that provide time slots of constant size at periodic intervals.

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) [5] and is defined as follows:

$$\begin{aligned} \text{MED} \;=\; & \text{end-to-end transmission delay of first packet} \\ & + \frac{\text{frame size * frames/packet}}{\text{bit rate}} + \text{proc. time/packet} \\ & + \text{jitter-buffer delay} \;+\; \text{playout delay/packet}, \end{aligned} \tag{1}$$

where the jitter-buffer delay is defined with respect to the arrival time of the first packet.

A second variation from time-division multiplexed networks is that a packet rate of 20 ms may be too high for some Internet connections, especially under heavy-traffic and lossy conditions. As a result, multiple frames are usually combined and transmitted in 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 of 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 because there are no fractional packet losses. This can be done by increasing the number of 5-ms subframes in each frame and by letting them share a common set of LP parameters. This leads to less bits than the original configuration with multiple 20-ms frames, as a larger number of subframes share the same LP parameters. For example, for a 40-ms frame, we need only one set of LP parameters instead of two sets for the two 20-ms frames.

Figure 1 shows the points that represent the original possible configurations in shaded circles. It also shows all the points that represent the new configurations, with frame size ranging from 30 ms to 100 ms. We also show an envelope, which depicts configurations that dominate all other configurations with either lower PESQ or higher bit rate. The envelope presents the new configurations introduced, especially those with lower bit rate 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.

III. 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 packet losses to some degree. For example, with a 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 received within its MED.

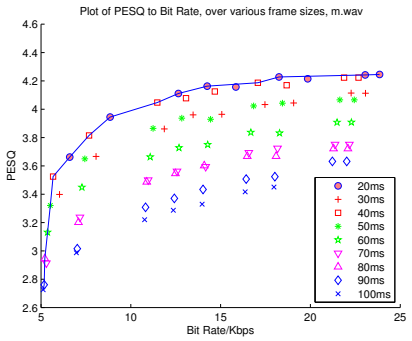

Fig. 1. Quality-bit-rate trade-offs of the new configurations generated. Each configuration is represented by its frame size. The envelope denotes all the dominating configurations.

By increasing the piggybacking degree, we can increase the amount of losses that a connection can tolerate. However, the degree of piggybacking cannot be indefinitely increased, due to constraints on the packet size and the MED. A frame is considered lost if it is not available from any of the packets that were received before its scheduled playout time determined by the MED. It is, therefore, 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 consecutive in time, it may not be necessary to have separate LP parameters for each. Our new algorithm exploits the temporal locality of these adjacent frames by merging them into a longer frame and by having more subframes share the same LP parameters. Our approach can achieve better bit savings with little degradations in PESQ.

A. Algorithm at the Encoder

Assuming a 20-ms frame size with a 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), …

Because the encoding of each frame depends on the previous frames, it is obvious that the above sequence of frames have 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), ... To maintain the proper internal state in each stream, the number of coders is equal to the piggybacking degree.

B. Algorithm at the Decoder

At the receiver, the decoder will split the single input stream into three distinct streams and decode them separately. The decoded frames from the three streams will then be merged in order to create a single stream of speech output.

The decoder tries to predict information in those lost frames, in case it cannot recover the information from the piggybacked frames. Hence, if three or more consecutive packets are lost and the receiver cannot recover the common frame in these packets, the decoder will provide predicted information on the lost frame from the information on those frames received.

Note that the information written to the output is after the decoding process. The decoder will process the multiple frames in each packet but only outputs the relevant frame. The following shows the pseudo code 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
```

C. Quality vs. Bit-Rate Trade-offs Under Packet Losses

We have 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 have tested the two methods of piggybacking using two benchmarks, one with male voice and the other with female voice. Between 5% to 30% random losses were injected into the packet stream to see how the new algorithm fares against the old one under various loss rates.

Figure 2 summarizes our results. Each graph compares the PESQs (*resp.* bit rates) of the original and the new configurations by computing the relative improvements:

$$\text{PESQ\_Ratio} = \frac{\text{PESQ of new piggybacking algorithm}}{\text{PESQ of old piggybacking algorithm}}, \quad (2)$$

$$\text{Bit\_Ratio} = \frac{\text{bit rate of new piggybacking algorithm}}{\text{bit rate of old piggybacking algorithm}}. \quad (3)$$

To achieve good trade-offs, we would like PESQ_Ratio to be close to 1 and Bit_Ratio as small as possible.

Each plot also shows a line of unit gradient. Points above the line have larger PESQ_Ratio than Bit_Ratio and represent those configurations where good trade-offs have been achieved. The plots clearly show that, over the various loss rates, we are able to achieve good trade-offs between bit savings and reductions in PESQ by using the new piggybacking algorithm. To show the perceptual quality of the various configurations, Table I uses color codes to classify the perceptual difference of all the points with respect to the PESQ of the original configurations.

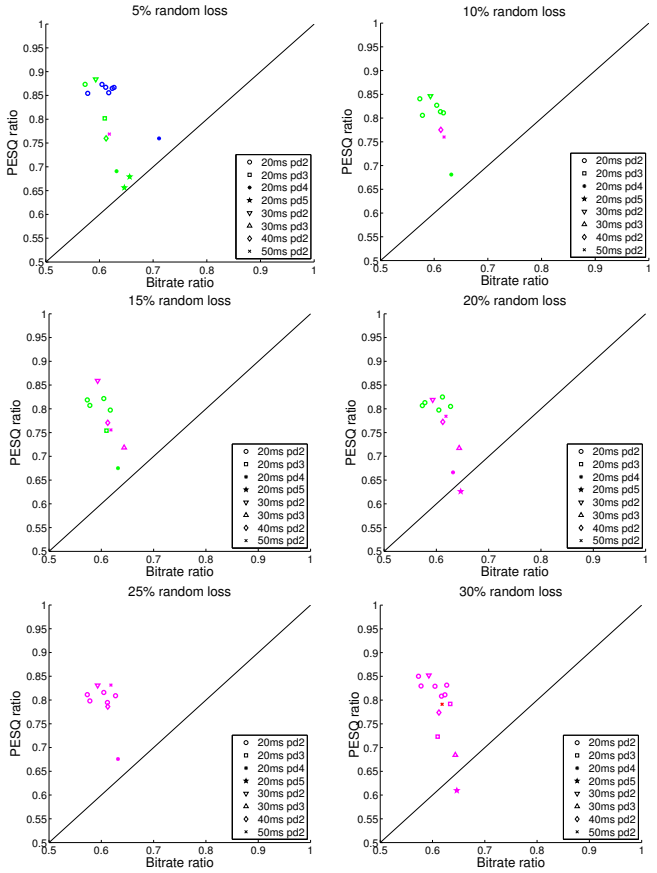

Fig. 2. Trade-offs between speech quality and bit rate under random losses with rates of 5%, 10%, 15%, 20%, 25%, and 30% using the speech file m.wav with male voice. ('pd' stands for piggybacking degree.)

TABLE I
DIFFERENCE IN PERCEPTUAL QUALITY WITH RESPECT TO THE ORIGINAL CONFIGURATION.

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

The results for the wave file with female voice are similar but with slightly lower PESQ. This is likely caused by the facts that the female voice tends to have more higher frequency components and that the input speech is high-pass filtered in the pre-processing stage.

Similar tests have been carried out for losses based on packet traces collected in the PlanetLab [5], [6]. The results obtained are similar to those shown for random losses.

IV. Estimating MED for Piggybacking

The success of piggybacking to help conceal losses or late arrivals depends on a good estimate of MED defined in (1) and

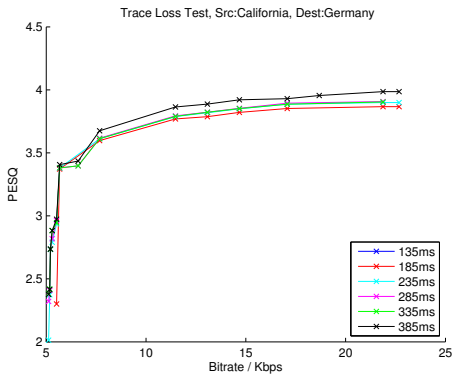

Fig. 3. Marginal improvements in PESQ as jitter tolerance is increased from 25 ms to 275 ms in 50-ms increments for the California-Germany PlanetLab connection.

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 (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 delays are stationary over the recent past, 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 (1) is:

$$\text{jitter-buffer delay} = \begin{array}{l}\text{average variation of arrival times}\\ \text{of the first } x \text{ packets with respect}\\ \text{to the first packet} + \text{ jitter tolerance.}\end{array} \quad (4)$$

To empirically determine the jitter tolerance in (4), we collect UDP packet traces in the PlanetLab from various sources to various destinations. The details of the traces used can be found elsewhere [5], [6]. We vary the jitter tolerance in (4) from 25 ms to 275 ms in 50-ms increments, while ignoring the processing delay, and set x to 10.

Figure 3 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 UDP packet trace from California to Germany in the PlanetLab. The trace is 5-min long with 60-ms period and a payload of 100 bytes [6]. 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. Similar to Figure 1, these points show the maximum achievable PESQ at the various bit rates over all possible configurations.

Starting from 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 have added 50 ms to the initial estimate. Subsequent lines represent MEDs at 50ms intervals. It can be observed that the envelope does not significantly increase further after adding 100 ms and saturates shortly after that.

TABLE II
RECOMMENDED PIGGYBACKING AND FRAME-SIZE CONFIGURATIONS

Frame Size/ms	Piggybacking Degree	Bit Rate in kbps
20	2	11.35, 15.35, 22.95
30	2	10.733
40	2	10.425
50	2	10.240

In total, we have performed simulations on over 100 traces between various sources and destinations, such as China, Taiwan, US and UK. These traces vary in durations between 5 min to 10 min, with packet period of 30ms or 60ms. They were collected over different times of the day to provide a good sample of Internet traffic throughout the day [5], [6]. The results in these experiments show that a 100-ms jitter tolerance is adequate in all the cases tested.

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

V. CONCLUSIONS

In this paper, we have modified the G.722.2 speech codec in order for it to work with frame sizes ranging from 30 ms to 100 ms, when multiple speech frames are piggybacked in a packet before they are sent to the receiver in a VoIP connection. By utilizing these new frame sizes, we have created an effective piggybacking algorithm that offers savings in bit rate with little degradation in speech quality.

Although the audio frames are not compliant with the standard after the modifications, the main structure of the encoding process (ACELP) is unchanged. As long as both the encoder and the decoder are consistent in recognizing the frame size used in transmission, the speech stream can be recovered in a way similar to the original standard.

REFERENCES

[1] International Telecommunication Union, "ITU-T G-Series recommendations," http://www.itu.int/rec/T-REC-G/en.
[2] *Wideband coding of speech at around 16 kbps using Adaptive Multi-Rate Wideband (AMR-WB)*, ITU-T Standard G.722.2, 2003.
[3] *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 Standard P.862, 2001.
[4] A. M. Kondiz, *Digital Speech, coding for low bit rate communication systems*, 2nd ed. West Sussex, England: Wiley, 2004.
[5] 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.
[6] B. Sat and B. W. Wah, "Playout scheduling and loss-concealments in VoIP for optimizing conversational voice communication quality," in *Proc. ACM Multimedia*, Sep. 2007, pp. 137–146.
[7] Voiceage Corporation, G.722.2 (AMR-WB), http://www.voiceage.com/prodg7222.php, June 2009.