ANALYSIS AND EVALUATION OF THE SKYPE AND GOOGLE-TALK VOIP SYSTEMS

Batu Sat and Benjamin W. Wah

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{batusat, wah}@uiuc.edu

ABSTRACT

In this paper, we study Skype and Google Talk, two widely used VoIP systems, and compare their perceptual speech quality with that of our proposed system using UDP packet traces collected in the PlanetLab. Based on methods for speech coding, packetization, jitter control, estimation and feedback of network conditions, and loss concealments, our results show that Skype has noticeable quality degradations because it only uses a maximum of two-way redundancy for loss concealment, does not handle out-of-order arrivals, and applies a fixed jitter control of 60 ms relative to the expected arrival time. Its slow loss-adaptation time of more than one minute to change from one-way to two-way redundancy makes it susceptible to quality degradation under fast changing loss conditions. In contrast, Google Talk does not employ any loss adaptation and performs similar to Skype under low- to medium-loss scenarios. By addressing the shortcomings of Skype and Google Talk, we demonstrate improvements in speech quality in our proposed prototype.

1. INTRODUCTION

VoIP (Voice-over-IP) has been very popular in the last few years due to its low cost, high quality, and ease of use. This work is motivated by Skype and Google Talk, two widely available but proprietary VoIP systems. The analysis in this paper helps designers understand their limitations, provides the community a base for comparison, and facilitates the design of better VoIP systems in the future.

Figure 1 depicts the general architecture of a general VoIP system. In this paper, we focus on the components for speech coding, packetization, play-out scheduling, and loss concealments. In each section, we overview the designs in Skype and Google Talk,[1] their pros and cons, and our results in Internet experiments. Since the source code of the two systems is unavailable, we have modified the kernel of a Linux router in order to instrument, capture, re-order, drop, and delay UDP speech packets sent by VoIP clients. Our prototype allows us to replay traces collected in the PlanetLab and to test VoIP systems under realistic Internet environments. We do not present the details on echo cancellation and firewall/NAT traversals due to space limitation and a lack of controversy in their designs.

RESEARCH SUPPORTED BY THE MOTOROLA CENTER FOR COMMUNICATIONS, UNIV. OF ILLINOIS, URBANA-CHAMPAIGN.
IEEE INT'L CONFERENCE ON MULTIMEDIA AND EXPO., 2006.

[1] We have used Skype v1.3.0.65 and Google-Talk Beta, both were the latest versions available at the time the research was conducted.

2. SPEECH CODECS

Standard low-bit-rate speech codecs, like ITU G.729, are not explicitly designed for use in IP networks. These codecs achieve high coding efficiency by removing redundancies, making it difficult to reconstruct lost frames. To this end, Skype sends redundant frames to protect against losses and uses the loss-robust *Internet Low Bit-rate Codec* (iLBC) [1].

Using linear PCM speech sampled at 8KHz with 16-bit precision, iLBC has two framing options: 20 ms and 30 ms, with a corresponding bit-rate of 15.2 Kbps and 13.3 Kbps. It is more robust to frame losses because it encodes each frame as self-decodable and eliminates the need for internal states. Like other CELP codecs, it uses linear predictive coding for representing the envelope of a signal. However, it encodes the excitations of each frame differently. It first identifies the portion of a frame with the most energy and encodes it by ADPCM. It then uses an adaptive codebook, generated using shifted and filtered versions of the high-energy portion, to represent the remaining excitations. Although there is some coding inefficiency, a decoder can reconstruct a speech frame perfectly from a received frame, without relying on the internal states of previous frames.

To compare the speech quality of G.729 and iLBC, we evaluate them under a 30-ms packet period and various IID packet loss rates and redundancy degrees. We assume three 10-ms G.729 frames in one voice packet and the 30-ms option in iLBC. We evaluate two-way (*resp*., three-way) *redundant piggybacking* in which frames in one (*resp*., two) previous packet are piggybacked in the current packet in order to allow the receiver to reconstruct the lost data in case the previous packet(s) is lost.

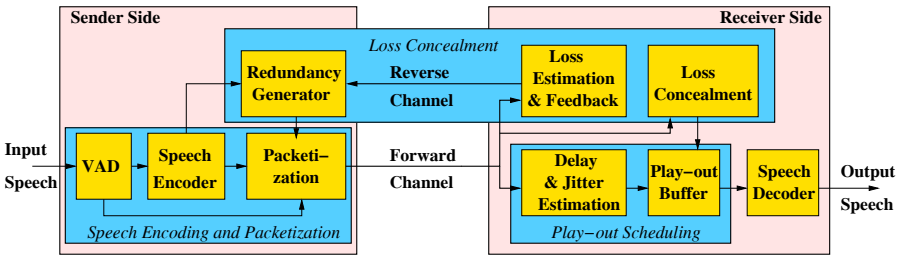

Figure 1: Architecture of a VoIP system with speech encoding, packetization, play-out scheduling, and loss concealment.

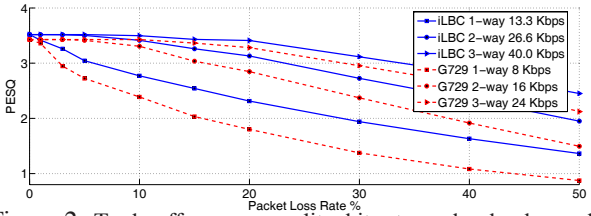

Figure 2: Trade-offs among quality, bit rate, and redundancy degrees for iLBC and G729 under IID packet losses and no jitter.

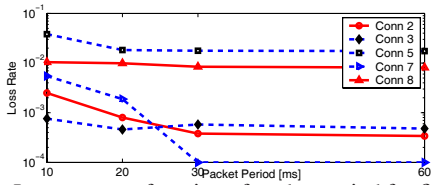

Figure 3: Loss rate as a function of packet period for five connections in Table 1 (collected in January 2006).

Figure 2 depicts the trade-offs between bit rate and quality for G.729 and iLBC. Although more robust to losses with respect to G.729, iLBC has inadequate quality under moderate loss rates and no redundancy. The quality can be improved by redundant piggybacking: G.729 with three-way piggybacking is found to have higher quality and lower bit rate (24 Kbps) than iLBC with two-way piggybacking (26.6 Kbps), and G.729 with two-way piggybacking has higher quality and slightly higher bit rate (16 Kbps) than iLBC with no redundancy (13.3 Kbps). The additional mouth-to-ear delay incurred depends on the packet period and the degree of piggybacking.

Google Talk uses a version of PCM with variable bit rates. In contrast to model-based speech coders, PCM encodes speech sample by sample, while minimizing sample-wise distortions. PCM lacks the efficiency of batch (frame) codecs, leading to a higher bit rate and better speech quality under no loss. When packets are lost, PCM can employ loss concealments but is limited to sample-wise reconstructions that perform worse than model-based reconstructions, especially under long packet periods. (See Figure 7 for the quality of Google Talk under IID losses.)

3. PACKETIZATION

Real-time voice communication requires frames to be encapsulated in IP packets and sent periodically to receivers. A short packet period shortens the mouth-to-ear delay but may incur higher loss rates and consequently lower quality.

Skype sends voice packets by two rates. Using voice-activation detection (VAD), it detects an active period when the caller is speaking and packs two 30-ms iLBC frames of 50 bytes each into an UDP packet and transmits the packets every 60ms (14.4 Kbps average). During silence periods, it transmits 25 bytes in each UDP packet every 100 ms.

Using similar packet periods, Google Talk can code in various standard and non-standard codecs. When communicating with a symmetric client, it uses one of the variable bit-rate PCM codecs. The output payload is speech dependent and is between 120 and 200 bytes (24 Kbps average).

Since the loss rate is not affected when the packet period is 30 ms or longer (Figure 3), the 60-ms packet period used in Skype and Google Talk is too conservative. Skype only uses a maximum of two-way piggybacking because, with a 60-ms period and two-way (*resp.*, three-way) piggybacking, there is an additional 90 ms (*resp.*, 150 ms) mouth-to-ear delay when one (*resp.*, two) packet is lost and is reconstructed from the subsequent packet. In contrast, a 30-ms period incurs an additional mouth-to-ear delay of 30 ms (*resp.*, 60 ms) when one (*resp.*, two) packet is lost. The use of a 60-ms packet period is, however, more network friendly.

4. PLAY-OUT SCHEDULING

Jitter buffers are used to smooth out variations in packet arrivals. For real-time applications, the size of the jitter buffers must be chosen properly in order to balance between the number of arrivals considered late for play-back and the mouth-to-ear delay. This size can be adaptively changed either during silence periods, or by utilizing pitch conserving time expansion and compression to smooth changes in the play-out schedule even during non-silence periods [2].

Skype uses a simple FIFO approach to handle arrivals and play-outs. It smoothly plays packets arriving within 60 ms with respect to the expected arrival time. It also buffers multiple packets arriving ahead of time and stores them for play back later. If two or more packets arrive out of order, it plays them in a FIFO order regardless of the original order sent (Figure 4). This behavior is annoying to listeners, as out-of-order packets should be either dropped or re-ordered. The fixed jitter length in Skype is also inadequate, as there are a moderate amount of jitters over 60 ms with respect to the mean delay for some connections (Table 1).

When no packet arrives for 60ms or more from the ex-

Table 1: Internet traces collected in December 2005: duration 5 min, packet period 60 ms, payload 100 Bytes, $\Delta \doteq |LR_{t+2sec} - LR_t|$.

Path	Source		Destination		One-Way Delay [ms]				Jitter wrt Mean (%)		Loss Rate$-LR$ (%)			
	Location	IP Address	Location	IP Address	Min	Max	Mean	Std.Dev.	$J > 60$ ms	$J > 30$ ms	Max	Mean	Std.Dev.	E[Δ]
1	Netherlands	130.37.198.243	China	219.243.200.53	170.1	222.5	172.4	2.4	0.0	0.1	18.2	5.3	3.9	4.2
2	Brasil	200.19.159.34	Wisconsin	198.133.244.146	120.6	378.7	129.1	17.1	2.0	5.7	12.1	1.7	2.6	2.1
3	Michigan	141.213.4.202	Spain	138.100.12.149	61.7	236.6	62.3	4.3	0.1	0.2	48.5	0.8	4.9	1.3
4	California	169.229.50.12	Germany	132.187.230.2	107.4	423.1	113.8	69.5	0.9	0.9	0.0	0.0	0.0	0.0
5	China	219.243.200.53	Wisconsin	198.133.244.146	120.1	148.0	123.1	3.2	0.0	0.0	63.6	33.5	10.9	15.9
6	Italy	130.136.254.21	Canada	142.103.2.1	99.0	256.5	102.2	40.5	0.3	0.3	15.2	1.2	2.1	1.9
7	Korea	143.248.139.168	Brasil	200.129.0.162	165.7	249.7	169.2	38.7	0.4	0.4	0.0	0.0	0.0	0.0
8	China	219.243.201.17	Ohio	129.22.150.90	110.1	237.3	112.1	3.4	0.1	0.1	30.3	8.0	5.8	5.8
9	Netherlands	130.161.40.154	Hong Kong	137.189.97.17	141.9	927.3	249.4	138.3	25.5	31.4	6.1	0.1	0.8	0.2
10	Wisonsin	198.133.244.146	China	219.243.200.53	121.3	134	122.1	1.8	0.0	7.6	42.4	17.4	7.8	9.8

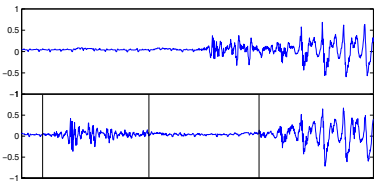

Figure 4: Skype plays two out-of-order packets in the order received: original sent (top) and degraded (bottom) speech.

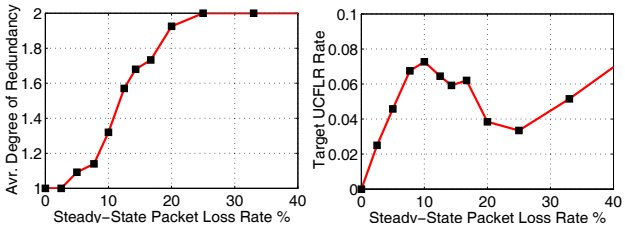

Figure 5: Loss concealment as a function of loss rate in Skype: a) redundancy degree; b) UCFLR (unconcealable frame loss rate).

pected time, Skype first applies iLBC-level loss concealments to the first few missed packets and then substitutes further missed packets by silence. If the original packets arrive later after the loss concealments have been performed, they are still played, and the output waveform is shifted altogether. Because this FIFO jitter control is independent of the one-way delay, all packets will be smoothly played even when consistently delayed by 10 seconds. When no UDP packets are received for 6 seconds, Skype switches to TCP with new sequence numbers. However, it continues to play the UDP packets if they arrive in the mean time, leading to a garbled stream with voice packets from both UDP and TCP.

Google Talk employs a similar play-out algorithm and considers packets with jitter higher than 60 ms from the expected arrival time as late. There is no significant improvement in quality even though Google Talk calculates the average delay more frequently than Skype.

5. LOSS CONCEALMENTS

Since the loss rate and behavior in an Internet path is asymmetric, non-stationary and dynamically changing, loss statistics must be relayed in both directions in order for each side to adapt to changing channel conditions. A trade-off must be made between the network overhead and the amount and accuracy of the feedback information.

By encapsulating 4 or 8 bytes of feedback information in speech packets in the reverse direction, a Skype receiver sends feedbacks to a sender every 2 seconds. Based on a history of lost packets or those received later than 600 ms of their expected time, the feedback contains a decision to make small discrete changes to the redundancy degree.

In response to the feedback, a Skype sender either transmits a packet containing two 30-ms iLBC frames (called 1-way or no redundancy), or piggybacks the payload of the previous packet and doubles the payload (called 2-way redundancy). It transmits different patterns of 1-way and 2-way packets periodically in order to gradually change the redundancy rate. To slowly adapt to changes in the loss rate, it requires 30 "increase-redundancy-rate" messages to switch from no to 2-way redundancy and 80 messages to switch the other way, taking, respectively, about 1 and 3 minutes.

Figure 5a depicts the average redundancy degrees and the corresponding steady-state loss rates. This gradual adjustment is a good alternative to a crude 1-way 2-way switch and results in a smoother curve of target *unconcealable frame loss rate* (UCFLR) as a function of packet loss rate.

Figure 5b shows the UCFLR that would have been observed at a receiver, when the sender has fully adapted to the steady-state condition. We have observed a local minimum of UCFLR at a 25% loss rate, when all packets have the maximum 2-way piggybacking. No further loss-adaptation is done with further increases in the loss rate.

There are three comments on Skype's schemes.

First, each point in Figure 5b depicts the UCFLR for a connection under a given steady-state loss rate. Although the UCFLR is bounded below 7%, Skype has difficulty in achieving this in practice. Our experiments show that there is a significant fraction of the connections with loss rates that change by more than 1% in two seconds (the last column in Table 1). Under such dynamic conditions, the use of small step sizes in Skype only allows connections with loss-rate variations slower than 1% in 2 seconds to be adapted to, and limits the ability to track rapidly changing loss rates.

Second, the frequent occurrence of two or more consecutive packet losses in typical inter-continental connections

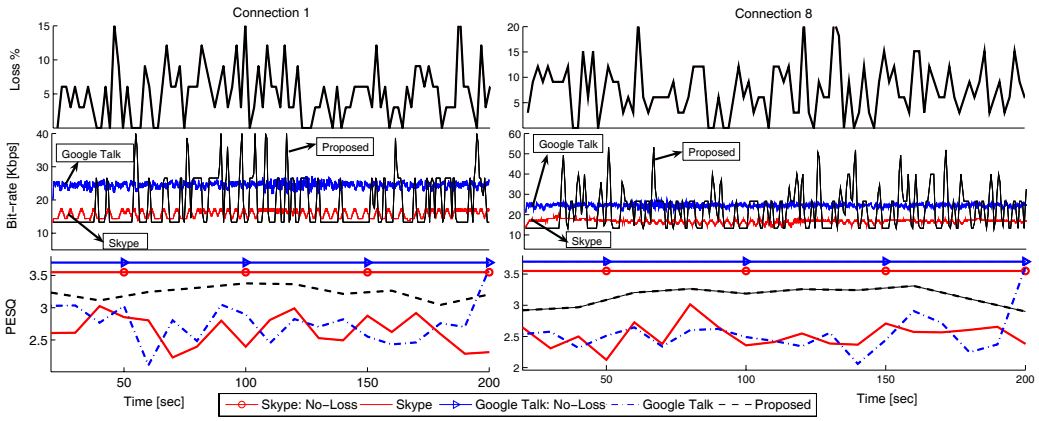

Figure 6: Loss rates (averaged over 2 seconds), short-term average bit rate of the payload, and perceptual speech quality of Skype (in red), Google Talk (in blue), and proposed (in black) for Connections 1 and 8.

means that two-way redundancy is not always adequate [3]. Using redundancies beyond two-way, however, will require a shorter packet period, say 30 ms, in order not to significantly length the mouth-to-ear delay. Further, when Skype uses a redundancy pattern that includes at least one 2-way packet in its data stream (when the perceived loss rate is greater than 2%), it increases the jitter buffer, and thus the mouth-to-ear delay, by 60 ms in order to wait for the redundant packet. This is particularly wasteful when the additional 2-way redundancy is only applied to a few packets.

Last, given the infrequent feedbacks and their importance, they should be transmitted more reliably using TCP.

Our experiments show that there is no change on the period or the average payload size in Google Talk in response to changing conditions. Hence, Google Talk does not adapt to network conditions and does not send feedback packets.

6. EXPERIMENTAL RESULTS

Figure 6 depicts the performance of Skype and Google Talk for Connections 1 and 8 in Table 1. For each connection, we show its loss rates, its short-term average bit rates, and the perceptual quality of each system.

The results show that, because Google Talk does not adapt to changing loss rates, it is susceptible to degradations when the loss rate is higher than 5%. At that point, it performs close to Skype but has 70% more bit rate.

The results also show that Skype cannot track rapid changes in the loss rate because it uses small incremental steps in its loss adaptation. Figure 6 illustrates that Skype operates at a maximum average redundancy degree of 1.2 for a loss rate of 15% in Connection 1, despite its design to operate at 1.7 average degree (Figure 5a). This effectively disables Skype's fine-grain loss adaptation and has it operate close to no redundancy. However, in the rare case of sustained loss rates over 25% (Connection 5), 2-way redundancy, the maximum allowed, is not adequate.

Finally, we propose a system for resolving some of the shortcomings in Skype and Google Talk. Using the iLBC

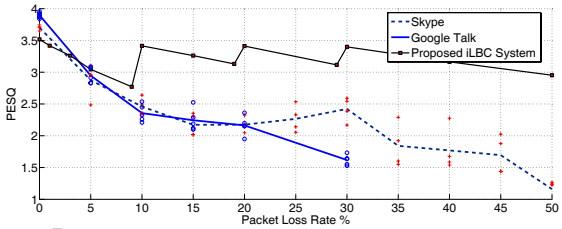

Figure 7: Speech quality of Skype, Google Talk, and our proposed VoIP system under IID packet losses and no jitter.

codec in the 30-ms mode and a 30-ms packet period, our system uses 3-way or 4-way redundant piggybacking when the loss rate is high, while using lower redundancy degrees to limit delays when the loss rate is low. To better track loss rates, our receiver declares a packet as lost when it does not arrive within 100 ms of the expected time (instead of 600ms in Skype), and sends feedback in each UDP speech packet in the reverse direction (instead of every 2 seconds in Skype). The feedback consists of a 2-bit decision that indicates the redundancy degree (1 to 4 ways) to be used at the sender (instead of a small increment in Skype). The receiver also uses a fixed 60-ms jitter buffer and either re-orders packets if they arrive in time or discards them if they are late.

Figures 6 and 7 compare the performance of our system, Skype, and Google Talk. They show that our system is agile in rapidly adapting to changing loss conditions with an adequate redundancy degree, without increasing the mouth-to-ear delay and rarely increasing the payload size.

7. REFERENCES

[1] S. Andersen et. al, "Internet low bit rate codec (iLBC)," http://www.ietf.org/rfc/rfc3951.txt, Dec. 2004.

[2] Y. J. Liang, N. Faber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over IP networks," *IEEE Trans. on Multimedia*, vol. 5, no. 4, pp. 532–543, Dec. 2003.

[3] B. Sat and B. W. Wah, "Speech- and network-adaptive layered G.729 coder for loss concealments of real-time voice over IP," in *IEEE Int'l Workshop on Multimedia Signal Processing*, Oct. 2005.