HANDLING PACKET LOSS IN WEBRTC

Stefan Holmer, Mikhal Shemer, Marco Paniconi
Google Inc. 1600 Amphitheatre Parkway, Mountain View, CA, USA

ABSTRACT

WebRTC is an open-source real-time interactive audio and video communication framework. This paper discusses some of the mechanisms utilized in WebRTC to handle packet losses in the video communication path. Various system details are discussed and an adaptive hybrid NACK/FEC method with temporal layers is presented. Results are shown to quantify how the method controls the quality trade-offs for real-time video communication.

Index Terms— WebRTC, real-time communication, error resilience, forward error correction.

1. INTRODUCTION

WebRTC [1] is an open-source project that enables web browsers with real-time audio and video communication. This paper presents some of the underlying video processing aspects of WebRTC that enable reliable transmission of real-time video over lossy networks. It is well known that it is difficult to provide a high user experience for interactive real-time applications such as video conferencing. These applications are limited by the time-varying nature of the network conditions (bandwidth, packet loss, network latency), and requirements of low-latency real-time coding.

Various approaches exist [2] to handle packet losses during a multimedia transmission, such as packet retransmissions based on negative acknowledgment (NACK), forward error correction (FEC) [3][4], and reference picture selection (RPS) [5]. These are often supplemented with codec error-resilience methods [2], such as intra-refresh and error concealment. For real-time applications with strict delay requirements, a hybrid NACK/FEC scheme [6] can be used to achieve some balance of delay cost in the NACK method and redundancy cost in the FEC method.

This paper presents one set of protection tools currently used in WebRTC for handling packet loss. In particular, an adaptive hybrid NACK/FEC method with temporal layers (TL) is proposed as a useful scheme to balance video quality, smoothness of rendering (playout jitter), and end-to-end delay. TL refers to the temporal scalability feature in the VP8 [7] codec used in WebRTC. Various system details and components are also discussed to highlight the adaptive nature of our approach.

The system description of the video processing in WebRTC is discussed in Section 2. Section 3 discusses the simulations and metrics used to quantify the system behavior. Sections 3.1-3.3 contain some results and discussion regarding hybrid NACK/FEC, FEC, and TL. Conclusion follows in Section 4.

2. SYSTEM DESCRIPTION

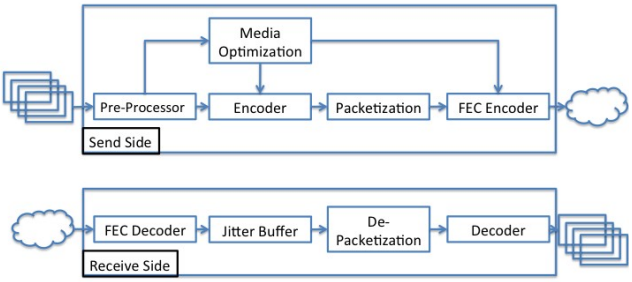

Figure 1 - WebRTC video processing system diagram

Figure 1 shows the WebRTC video processing system diagram. Raw frames entering the send side are first preprocessed, and then encoded at a given target rate. Subsequently, the frames are packetized, and when applicable, an FEC encoder is applied. The FEC is a XOR code based on RFC 5109 [8]. On the receiver side, packetized encoded data is processed by the FEC Decoder, followed by the Jitter Buffer (JB). The latter constructs encoded frames from the received packets and estimates the video jitter. Once a frame is complete, it is sent to the decoder, which outputs raw data (YUV format). The JB is also responsible for building a list of missing packets that are the basis for the retransmission request.

We model the jitter as composed of two components, one random and one relative to the size of the video frames [9]. We then collect the per-frame statistics of the frame's capture time and receive times, and model it as linearly dependent on the frame's size difference. This method of estimating the jitter makes it possible to adapt to changes in frame size and link capacity, which often have an effect on the video jitter. The jitter estimate adapts to frames being late due to FEC decoding, but not due to retransmissions.

The Media Optimization (MO) component on the send side controls the adaptive hybrid NACK/FEC. MO periodically receives network statistics, which are updated with every incoming RTCP receiver report (approx. every second). These network statistics include the available bandwidth, fractions of packets lost and the Round Trip Time (RTT). A receive-side bandwidth estimator computes

the available network bandwidth [9]. The MO also receives encoder statistics such as the incoming frame rate and the actual bitrates sent (video bitrate and FEC/NACK protection overhead rate). The main function of MO with regard to the hybrid NACK/FEC is to set the amount of FEC protection, and update the encoder with the new source rate (available bandwidth minus the estimated protection overhead).

3. SYSTEM BEHAVIOR AND RESULTS

The system was evaluated using an offline simulation tool, which simulates various network conditions in a controlled environment. The simulation tool acts as a transport module between the sending client and the receiving client, and consists of a queue that ads a network transit delay. A packet dropper is placed after the queue, which can inflict packet losses drawn from a bursty loss model using the Gilbert-Elliot model [10]. In the following results, only complete VP8 bitstreams are given to the decoder, so the video is decoded without errors/packet loss, and the receiver is configured to wait for all necessary packets. Video quality is therefore mainly affected by the smoothness of the playback and the available bitrate.

The following performance metrics defined below are measured to characterize the behavior of the system:

- End-to-end delay: the average of the time taken between reading a frame from a file until it is about to be rendered back to a file at the receiver.
- Render standard deviation: the standard deviation of the time delta between two successive frames being rendered. For the best temporal quality the render delta should be close to the captured frame rate with low variance.
- Protection overhead: defined as the average percentage overhead (relative to total bitrate) due to retransmissions and FEC. This is a measure of how much the video protection applied degrades the compressed video signal.

Simulations reported below were conducted on a talking heads scene with a resolution of 640x360 at 30 fps, with low to medium motion. The sequence is about ~30 seconds long, and the results were averaged over multiple runs.

3.1 Hybrid NACK/FEC

WebRTC uses an adaptive hybrid NACK/FEC method to obtain a better trade-off between temporal quality (smoothness of rendering), spatial video quality and end-to-end delay. The adaptive aspects of our method refer to the dynamic setting of the FEC amount at the sender side, and the playout delay at the receiver side.

The cost of the hybrid NACK/FEC method is the overhead penalty of the FEC, as shown in Figure 2a. Apart from that, it has clear benefits over the NACK only method. Figure 2b shows that on average end-to-end delay is reduced when combining NACK and FEC since on average less time is spent waiting for retransmissions, although the wait time for a single retransmission is unchanged. As shown in Figure 2c and d, the standard deviation of the render time delta is significantly reduced as well.

As mentioned in Section 2, the FEC amount (protection level) is determined in the MO based on the received network statistics. In particular, the amount of FEC is conditioned on the RTT. Packets can be retransmitted without substantial freezes when the RTT is low; therefore the amount of FEC can be reduced, resulting in a smaller delay penalty. In large RTTs, the delay has a bigger impact on the temporal quality, and therefore the amount of FEC should not be reduced. This is shown in Figure 2a, where the FEC overhead is reduced for RTT/2 < ~50 ms.

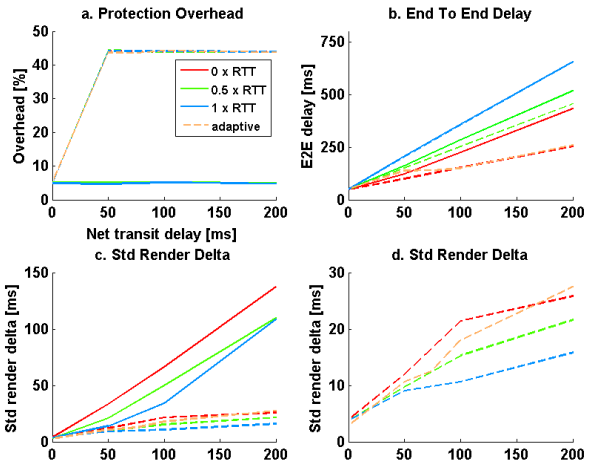

Figure 2 – NACK and Hybrid NACK/FEC: for different values of d_{add}, as a function of network transit delay. Solid lines refer to NACK, dashed lines refer to hybrid NACK/FEC. For a 5% packet loss rate, burst length of 1, at 500kbps. a) Protection overhead. b) End-to-end delay. c) Rendering standard deviation. d) Rendering standard deviation for hybrid NACK/FEC only.

The playout delay is controlled in the JB, and is used to trade-off the temporal quality (smoothness of rendering) with the end-to-end delay. The goal is to delay the playback in order to reduce the duration in which the video is frozen while waiting for a retransmitted packet. However, when the RTT is large, additional playback delay is less suitable, as one-way delays longer than 400 ms severely impair communication [11]. Therefore the additional playback delay should be chosen depending on the fraction of unrecoverable packet losses, u, and the estimated RTT. The additional playback delay can be calculated as

$$d_{add} = min(max(K - RTT/2 - d_{jitter}, 0), RTT), \quad if\ u > U_{min},$$
$$d_{add} = 0, \quad otherwise.$$

K is the maximum acceptable end-to-end delay, *RTT/2* is an estimate of the network transit time, d_{jitter} is the estimated jitter and U_{min} is a threshold on the packet loss. The fraction of unrecoverable losses can be estimated as the number of packets NACKed and received unreasonably late compared to the number of packets received within reasonable time, counted over a fairly large window, e.g., ten seconds. Unreasonably late can be defined as at least *RTT/2* later than when we expected the frame in question to be completed. The cost of increased end-to-end delay can be seen in Figure 2b, while the gain of additional playback delay can be seen in Figure 2c and 2d. The "adaptive" line in Figure 2 uses the equation above for d_{add}, with K=100 ms and U_{min} = 0. The other lines have $d_{add} = k$RTT with k in {0, 0.5, 1}.

3.2 Multi-Frame FEC

The FEC used in WebRTC is a XOR packet level code [8]. We denote the code as *(k, m)*, where k is the number of video packets in the protection group, and m is the number of FEC packets in that protection group. The protection overhead of the FEC is defined as $PL=m/(k+m)$. The maximum number of frames used in the protection group, λ, also characterizes the code. This number is determined dynamically in the MO, based on the received network delay and video frame rate: $\lambda \sim max(1, min(fRTT, \lambda_o))$, where f is the frame rate, λ_o is a fixed upper bound.

Multi-frame FEC can reduce the FEC overhead at low bitrates, where the granularity of the 1-frame FEC becomes very small (i.e., for a small number of packets per frame). Another feature of the multi-frame FEC is that it is generally more effective at recovering losses than the 1-frame FEC, especially for bursty losses. That is, given two codes with similar protection levels, the longer code generally has lower average residual loss than the shorter one. This is due to the possibility of recovering more loss configurations with the generator matrix of the longer FEC code (see Figure 3). This improved recovery comes however at a cost of increased FEC decoding delay.

An excess overhead threshold controls the actual number of frames used for the FEC. Excess overhead is defined as the actual overhead (based on the actual number of packets in the protection group) minus the target overhead (determined by *PL*). Consequently, if the FEC generator receives $\lambda > 1$ from MO, then the actual number of frames used in the FEC protection group is increased (up to potentially the maximum λ) until the excess overhead is below the threshold.

Figure 4 compares the 1-frame FEC with the multi-frame FEC; for this comparison λ was fixed to 1 and 6, respectively. We can see from Figure 4a and 4b that both the end-to-end delay and the render delta variance are reduced with the multi-frame FEC. For this loss simulation, the FEC protection level set in the MO is such that the average protection overhead is ~20/25%. Figure 4c shows how the multi-frame FEC can hit the target protection overhead, whereas the 1-frame FEC overshoots and hence generates excess overhead, particularly for the lower bitrate range (excess overhead decreases at higher bitrates, i.e., more packets/frame). The lower protection overhead for the multi-frame case results in a higher PSNR/quality.

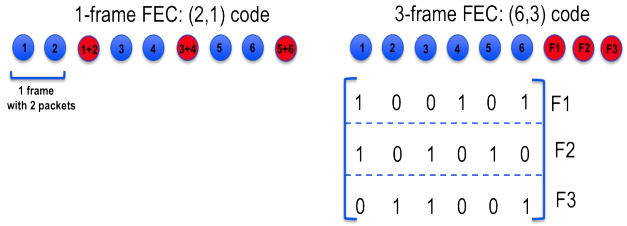

Figure 3 - Multi-Frame FEC - Example of 2 packets per frame at 33% protection overhead (FEC packets follow the source packets they protect). The 1-frame FEC is the XOR of the two source packets in each frame, whereas the 3-frame FEC has the 6x3 generator matrix shown above. The latter can recover more loss configurations, in particular any consecutive loss of size <=3 packets is fully recoverable with the (6,3) code.

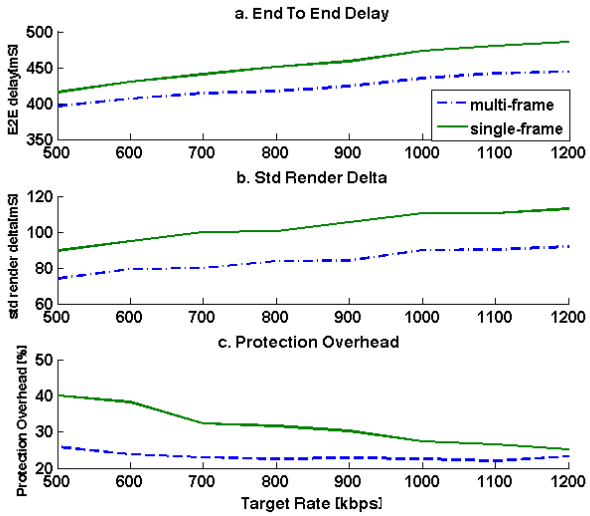

Figure 4 - Hybrid NACK/FEC - Multi-Frame FEC vs. 1-frame FEC. For a packet loss rate of 5%, burst length of 2, and RTT = 300ms; $d_{add} = 0$. a) End-to-end delay. b) Rendering standard deviation. c) Protection overhead.

3.3 Hybrid NACK/FEC With Temporal Layers

The multi-frame FEC discussed above has potential for reducing the protection overhead and improving recovery to packet loss, but the longer FEC decoding delay becomes too costly when the RTT is not significantly larger than the

inverse frame rate. Hybrid NACK/FEC combined with layered coding allows for another mechanism to reduce overhead (e.g., by protecting only the base layer frames), and provides an additional trade-off of lower end-to-end delay with video quality (lower spatio-temporal resolution). In this paper we discuss how temporal layers (TL) are used in conjunction with hybrid NACK/FEC.

Temporal scalability in VP8 [7] enables generating rate-targeted temporal separable streams. For the results reported here, the rate allocation for the base layer is 60% for TL=2 and 40% for TL=3 (2 and 3 temporal layers, respectively). The temporal pattern structure has sync frames (placed every 8 frames) that are used to enable dropping of (incomplete/lost) enhancement frames at the receiver.

The hybrid NACK/FEC + TL system operates as follows:
1) UEP-FEC (unequal error protection) is used where protection is only applied to base layer frames. Sender only re-transmits packets belonging to base layer frames.
2) The temporal layer ID and the sync frame flag are embedded in the codec specific RTP header [12] of each packet, and extracted at the receiver/jitter buffer.
3) A type of selective NACKing is done, where only missing packets that correspond to the base layer frames are NACKed. If a missing packet is detected in an enhancement layer frame, then all enhancement frames are dropped until a complete sync frame is received.

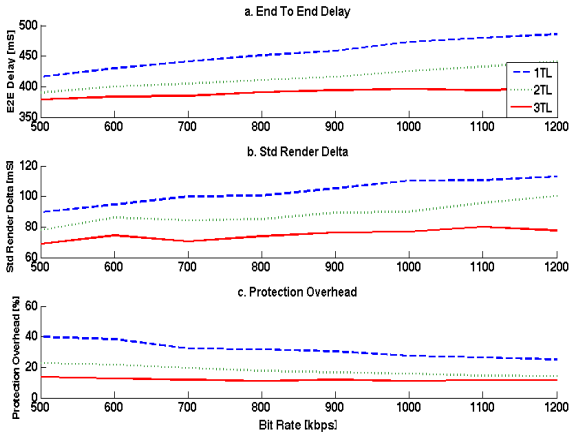

Figure 5 - Hybrid NACK/FEC + TL- For a packet loss rate of 5%, burst length of 2, and RTT = 300ms; $d_{add} = 0$. a) End-to-end delay. b) Rendering standard deviation. c) Protection overhead.

Figure 5 shows the performance metrics for the hybrid NACK/FEC with TL=1, 2, 3. In these comparisons, the 1-frame FEC is used. Figures 5a and 5b show the significant gains in lower end-to-end delay and render variance, along with lower protection overhead in Figure 5c. The overhead reduction is due to applying the FEC only to base layer frames. The overhead reduction corresponds to roughly the base layer bitrates: ~60% for TL=2, and ~40% for TL=3.

The trade-off of using TL has two components: (1) the compression efficiency loss (codec penalty) in the absence of packet loss, and (2) the lower temporal resolution (from dropped enhancement frames at receiver) in the rendered video. The visual quality loss from dropped enhancement frames is difficult to quantify. Regarding the codec penalty for TL > 1, at least in cases with relatively high overhead, we can expect that the gains from the reduced FEC overhead should compensate the compression efficiency loss. This is suggested in Figure 6, which shows codec penalty loss under temporal layers.

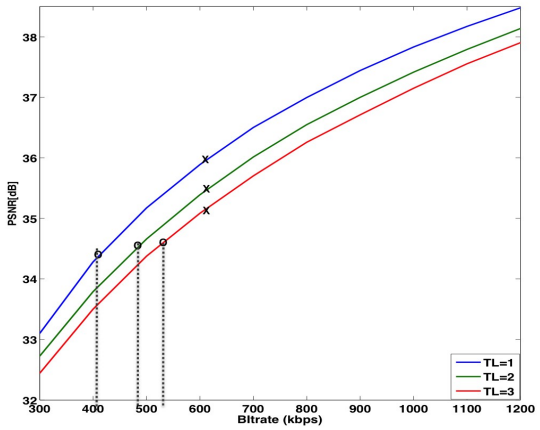

Figure 6 - PSNR for zero packet loss- Indicated in the plot are two sets of points: one set at ~600kbps, the other at reduced bitrates corresponding to protection overheads of ~33%, ~(33*0.6)%, ~(33*0.4)%, for TL =1, 2, 3, respectively. The 33% is an example overhead for TL=1 in Figure 5c at ~600/700kbps.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented some aspects of the video processing tools for handling packet loss in WebRTC. Performance metrics were used to quantify the network effects of packet loss and delay. An adaptive hybrid NACK/FEC method combined with TL was proposed to control the overall video quality, playout jitter, and end-to-end delay. In particular, adaptive playout delay as a mechanism for trading off render jitter with delay was discussed, and two methods (multi-frame FEC and TL) were proposed for reducing the protection overhead cost. When considering bursty loss scenarios and relatively long RTTs, the results indicate the potential for both methods to improve all three system metrics: lower render jitter, lower end-to-end delay, and higher or similar (spatial) video quality/PSNR. Improved performance can be expected from various extensions, such as more optimal use of selective NACKing, multi-frame FEC, and UEP-FEC across temporal layers. Extensions of this work also include improving the metrics to include, e.g., a better measure of jerkiness [13].

5. REFERENCES

[1]http://www.webrtc.org/;
http://code.google.com/p/webrtc/

[2] Y. Wang, S. Wenger, J.T. Wen, A.K. Katsaggelos, "Review of Error Resilient Coding Techniques for Real-Time Video Communications," IEEE Signal Proc. Magazine, vol. 17, no. 4, pp. 61-82, Jul. 2000.

[3] J, Korhonen, P. Frossard, "Flexible forward error correction codes with application to partial media data recovery", Signal Processing: Image Communication 24, (2009), 229-242.

[4] F. Battisti, M. Carli, E. Mammi, and A. Neri, "A study on the impact of AL-FEC techniques on TV over IP Quality of Experience", *EURASIP Journal on Advances in Signal Processing,* 2011.

[5] S. Fukunaga, T. Nakai, and H. Inoue, "Error Resilient Video Coding by Dynamic Replacing of Reference Pictures", Proceedings of IEEE Global Telecommunications Conf. (GLOBECOM), London, vol. 3, Nov. 1996, pp.1503–1508.

[6] F. Zhai, Y. Eisenberg, T. N. Pappas, R. Berry and A. K. Katsaggelos, "Rate distortion optimized hybrid error control for real-time packetized video transmission," IEEE Transactions on Image Processing, pp. 40-53, 2006.

[7] http://www.webmproject.org/; J. Bankoski, P. Wilkins and Y. Xu, "VP8 Data Format and Decoding Guide," RFC6386 (Informational), Nov. 2011.

[8] Li, A., "RTP Payload Format for Generic Forward Error Correction," RFC 5109 (Proposed Standard), Dec. 2007.

[9] H. Lundin, S. Holmer and H. Alvestrand, "A Google Congestion Control Algorithm for RealTime Communication on the World Wide Web," IETF Informational Draft, April 2012.

[10] P. Ferre, D. Agrafiotis, T-K Chiew, A. Doufexi, A.R. Nix, D.R Bull, "Packet Loss Modelling for H.264 Video Transmission over IEEE 802.11g Wireless LANs", in WIAMIS 2005.

[11] ITU-T G.114, February 1996.

[12] P. Westin, H. Lundin, M. Glover, J. Uberti and F. Galligan "RTP Payload Format for VP8 Video," IETF Internet Draft, Jan 2013.

[13] S. Borer, "A model of jerkiness for temporal impairments in video transmission," in Proc. Int. Workshop Quality Multimedia Exper. (QoMEX), Jun. 2010, pp. 218–223.