Performance Analysis of AL-FEC for RTP-based Streaming Video Traffic to Residential Users

Martin Ellis
School of Computing Science
University of Glasgow
Email: ellis@dcs.gla.ac.uk

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow
Email: dp@dcs.gla.ac.uk

Colin Perkins
School of Computing Science
University of Glasgow
Email: csp@csperkins.org

***Abstract*—Real-time applications used by residential customers, such as streaming video and IPTV, are sensitive to packet losses, whether due to IP-layer congestion, or link-layer problems such as bit errors induced by impulse noise. To achieve acceptable user experience for these applications, numerous application-layer forward error correction (AL-FEC) schemes have been proposed. We evaluate some of the FEC schemes developed as part of the OpenFEC project, using packet loss traces of IPTV-like traffic measured on ADSL and Cable links. We consider the effectiveness of these schemes in correcting the loss patterns present on residential links, explain why performance is different using measured loss traces compared with previous simulations using uniform random packet loss, and give recommendations for the use of FEC in streaming video applications deployed to residential Internet users.**

I. Introduction

Forward error correction (FEC) is a well-known technique to protect real-time traffic against the effects of packet loss. In IP-based streaming video and IPTV applications, FEC is generally deployed at the application level, adding redundant packets to the media stream to repair loss. To improve transmission efficiency, and to reduce the risk of introducing congestion by increasing the overall data rate, it is desirable to minimise the FEC overhead, while maintaining adequate protection. Finding the correct balance can be difficult, and requires insight into the network conditions.

With ongoing deployment of streaming video and IPTV to residential Internet users, it is becoming more important to understand how to tune FEC to suit such services. In particular, we need to determine how the loss patterns of ADSL and cable access links differ from more widely studied backbone network links, and measure their effect on FEC performance.

In this paper, we compare the performance of three application-layer FEC schemes implemented in the OpenFEC project (http://openfec.org), when applied to RTP-based streaming video. The FEC schemes are 2D parity codes (2D) [1], Reed-Solomon erasure codes (RSE) [2], and LDPC-Staircase codes (LDPC) [3]. We use packet loss measurements from video streams sent from a well-connected server to residential users [4] to inform trace-driven simulations of FEC performance, studying residual packet loss rates and fraction of lost packets successfully recovered. Our contributions are 1) a simulation-based evaluation of AL-FEC performance on real networks, using traces of unmanaged Internet streaming to residential users (i.e., reflecting the conditions that inter-domain IPTV flows could experience if they were deployed in future; 2) an explanation of the differences between our results and those of previous evaluations of these schemes under random packet loss (particularly the effect of bursty packet loss); and 3) guidelines for use of FEC, to recommend which FEC schemes and parameters work well under the loss conditions common on residential networks.

Previous work has examined FEC performance in analytical studies, or using data from well-provisioned networks; to the best of our knowledge, ours is the first to consider FEC performance for streaming traffic sent to residential ADSL and Cable users using real-world network performance data.

The remainder of this paper is structured as follows. We outline related work in §II, describe the FEC schemes in §III, and explain our approach in §IV. §V presents results, and §VI discusses our recommendations. We conclude in §VII.

II. Related Work

The OpenFEC codes we examine were studied under uniform random packet loss by Matsuzono *et al.* [5]. In [5], a lab-scale experiment was set up, with a sender and receiver running a DV (Digital Video) application transmitting data over RTP/UDP/IP [6]. The video streams were protected using 2D parity codes [1], RSE codes [2], and LDPC-Staircase codes [3], and their performance was evaluated under uniform random loss from 0–51%. The study evaluated recovery capabilities, latency introduced, and CPU cost incurred by each of the FEC schemes, concluding that LDPC codes with a source block size of $k = 170$ give the best trade-off between recovery performance, latency, and CPU load. However, this does not tell us about FEC performance under real-world packet loss conditions, which is what we show in the present paper.

A previous study looking at FEC performance on DSL networks was conducted by Begen [7]; this compares the performance of 1D interleaved parity codes against Raptor codes [8]. Begen uses typical DSL noise models to drive simulations of loss patterns, while we evaluate performance using measured loss traces. Similarly, models of packet loss were used in a DVB study [9] to evaluate the performance of 1D parity and Raptor codes, describing the relative strengths and weaknesses of both schemes. Luby *et al.* [10] discuss AL-FEC for IPTV, including a discussion of different FEC

schemes, and the "layered" approach using 1D parity and Raptor codes, discussed in [9]. Mammi *et al.* [11] investigated SMPTE 1D and 2D parity FEC schemes [1], using a testbed setup and models of random i.i.d. and repetitive electrical impulse noise (REIN). Kang & Loguinov [12] analytically studied the effect of packet loss on FEC for video streaming, using models for packet loss, but did not consider any Internet loss measurements. FEC performance for video on optical backbone networks was studied in [13], finding that a suitable trade-off between repair performance and latency can be hard to achieve. Older work examined performance of FEC for packet audio over the academic backbones [14], [15], using measured traces and performance models.

We focus on FEC performance using measured data from residential broadband networks, since streaming video systems are often accessed by home users. The characteristics of the packet loss on these networks has been found to be quite different from uniform random loss [4], so we expect FEC performance to be different too. Since previous work has not looked at FEC performance using data from streaming to residential broadband networks, this is where we focus.

III. FEC for Streaming Video

In this section, we describe the FEC schemes in the OpenFEC framework. Each of these schemes has been considered by the IETF FECFrame working group for use in protecting media streams, and has received interest from both academia and industry. We discuss the operation of the schemes, consider their overhead and latency, and describe how we compare their performance using measured loss traces, in contrast to previous evaluation under random loss conditions [5].

2D parity codes work by arranging the k source packets into a grid of D rows and L columns, and adding repair packets for each row and column. Therefore, a total of $D + L$ repair packets are added. If at most one packet is lost in a row or column, then the remaining packets are used to recover the loss. SMPTE 2022-1 [1] is a widely deployed example. This scheme is simple, but has a limited capacity for recovery, since it is only resilient to up to L consecutive losses (i.e., is able to recover from loss bursts up to length L, provided that no other packets were lost within the grid). As there are constraints on the latency that a real-time system can tolerate, the D and L parameters cannot grow too large. In OpenFEC, the 2D codes use square grids (with $D = L$), and limit k to be ≤ 16.

RFC 5510 [2] defines RSE codes for use in real-time streaming. Reed-Solomon codes are maximum separable distance codes, meaning that of the n encoding symbols sent (including k source symbols and $n - k$ repair symbols), any k can be used for recovery. The computational cost of the mathematical operations (using Galois Fields) increases rapidly with the size of those fields, introducing a practical limitation on the block sizes that can be used. Due to these limitations, we will look only at RSE codes over GF(2^8) (as in [5]).

A third FEC scheme, LDPC-Staircase codes (defined in RFC 5170 [3]), is suitable for use with large block sizes with relatively low computational complexity. Similar to Raptor codes [8], LDPC-Staircase codes require more than k symbols to be received to allow recovery for every k symbols sent (i.e., these are not maximum separable distance codes). However, in practice, the fraction of extra symbols required can be quite low when using an appropriate decoding algorithm ([5] states that $(k \times 1.05)$ is appropriate, based on experimental evidence).

IV. Methodology

The OpenFEC framework provides implementations of the FEC algorithms under study, as well as a performance evaluation tool, eperftool. This tool evaluates the FEC schemes on a single machine, simulating the transmission and reception of packets, and is configurable with a number of transmission schemes (determining the order of transmitted source and repair packets) and loss modes (determining which packets arrive at the receiver).

We use the dataset described in [4], which contains end-to-end measurements of streaming synthetic IPTV-like traffic across the open Internet at a range of bit-rates (1–8.5Mb/s) to residential ADSL and Cable users. The dataset consists of over 3800 traces, varying between one and ten minutes in duration (between 6000 and 120000 packets per trace). For more details of the measurements, and information on where to obtain the trace data, please see [4]. In these traces, sequence numbers indicate the ordering of packets, allowing packet loss to be calculated. We process each trace to generate a binary loss sequence, then use these sequences to evaluate the performance of the FEC schemes.

To evaluate the FEC schemes using loss traces, eperftool was modified, as follows. We added a new loss mode to read a given loss trace and use the loss patterns within the trace to decide whether packets will be received or lost. We also added a new transmission mode (which determines in which order the packets are sent), to support the source and FEC packets for each block being sent together, rather than using the default sending arrangement which sends all source packets (for all blocks) first, then all repair packets. This modification is necessary to prevent large delays when recovering lost packets. Finally, modifications have been made to profile the FEC decoding process, to enable fine-grained reporting of packet loss and repair. To this end, the number of source and repair packets received within each block are recorded, to allow calculation of per-block and overall packet loss statistics. When decoding of a block fails, the number of source packets received (and lost) determine the residual loss rate. A further metric of interest in FEC is the delay incurred by waiting for FEC packets to recover lost packets; to capture this, we added profiling code to record the distance between lost packets and the repair packets which recover them. The modifications to eperftool are available at http://martin-ellis.net/research/fec.

A. FEC Parameters

To begin, we apply the same parameters as [5]; 2D with source block size $k = 16$, RSE with $k = 170$, LDPC with $k = 170$, $k = 500$, and $k = 1000$, and code rate of $\frac{2}{3}$, (i.e., 50% overhead). This allows us to re-run the experiments of

[5] to validate our setup, and to compare FEC performance using real-world loss traces against simulated random loss.

We specify the total number of source (T_k) and repair (T_r) packets that will be used for each simulation. When re-running the experiments of [5], we choose $T_k = 10000$ and $T_r = 5000$ to achieve a code rate of $\frac{2}{3}$, with a reasonable number of blocks for each of the FEC schemes being tested. For the loss traces, the choice of T_k and T_r is determined by the trace length, with $\frac{2}{3}$ of the trace being allocated to the source packets, and the remainder to the repair packets. So, for a trace of length T, $T_k = \lfloor T \times \frac{2}{3} \rfloor$, and $T_r = \lfloor T \times \frac{1}{3} \rfloor$. For 2D FEC, there are further limitations, such that T_k must be a multiple of 16 (the largest 2D block size supported in OpenFEC), and T_r is $T_k/2$.

B. Performance Metrics

In [5], three metrics are used to evaluate performance; residual (post-repair) loss rate, frame delay, and CPU usage. Since we use eperftool rather than a separate sender and receiver, we will look at residual loss rate and delay due to FEC. Measuring CPU usage is not so important, since our goal is not to measure the complexity of the algorithms, but rather to observe how their repair performance is affected by measured loss data from residential networks, and since FEC decoding cost is small relative to the cost of decoding video.

To calculate the delay due to FEC, we count the number of packets received between the time when the lost packet would have been received, and the receipt of the packet that repairs the loss. This is more appropriate than the wall-clock time measured for delay in [5], since it makes no assumptions about sending rates, and is not tied to the particular processor or load of the machine doing the calculations. Although delay in terms of packets does not consider the FEC decoding delay, in practice we expect this to be small compared to the packet arrival delay, especially given the low packet sending rates available on residential networks.

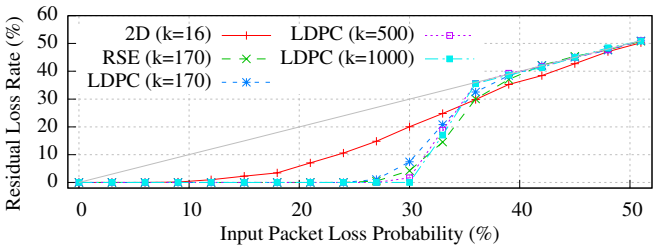
Further work could extend the evaluation to directly measure video quality by decoding a video sequence with the post-repair loss patterns, and report video metrics such as PSNR. In this paper, we focus on the FEC performance, since evaluating video quality depends not only on loss patterns, but also on the encoding method and type of content, whereas FEC performance provides a more general metric.

V. OpenFEC Performance

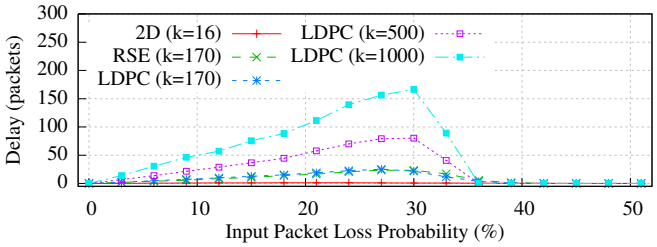
We begin by evaluating the FEC schemes with simulated random loss, as in [5], then present results of applying the same FEC schemes, with the same parameters, to the measured loss traces, and discuss why the performance differs, and the impact this has on real-world streaming video traffic.

A. Applying FEC under random loss

Figure 1a shows the input packet loss probability against residual (post-repair) packet loss rate (the diagonal represents loss rate with no FEC). This shows that for low loss (up to ∼10%), all the schemes perform well, repairing almost all loss. Above 10%, 2D parity FEC starts to show poorer



(a) Loss probability vs. residual packet loss rate



(b) Loss probability vs. mean FEC delay

Fig. 1. FEC performance under simulated random loss

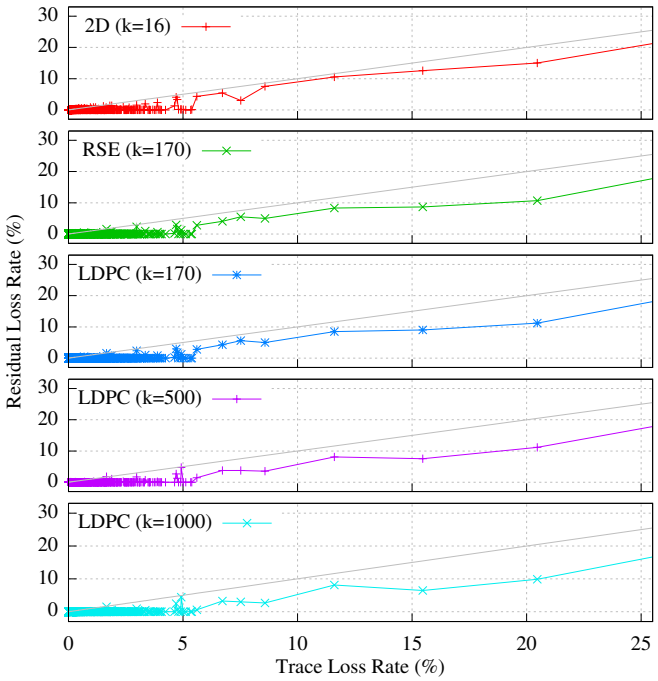
performance, showing steadily higher residual loss rates as the input loss probability increases. Up to ∼30% loss, the other FEC schemes continue to perform well. However, above 30%, they show a sharp degradation in performance, with residual loss rates climbing sharply, eventually matching the input loss probability after 35%. Beyond 35% loss, 2D FEC gives slightly lower residual loss rates (since the small block sizes mean that some blocks will be repaired); however, residual loss rates of above 30% will produce unusable video. These results are comparable with those in [5], which also showed the early decline in 2D parity FEC performance, and the "cliff" in performance for the other schemes.

Figure 1b shows the mean FEC recovery delay (in packets). At low loss rates, delays are low (but related to the block size), since most packets are received normally (i.e., with delay 0). As loss rate increases, the delays increase, with more packets in need of repair, up to the threshold point discussed earlier and seen in Figure 1a. After this point, since fewer packets are actually recovered, the average delay due to FEC decreases.

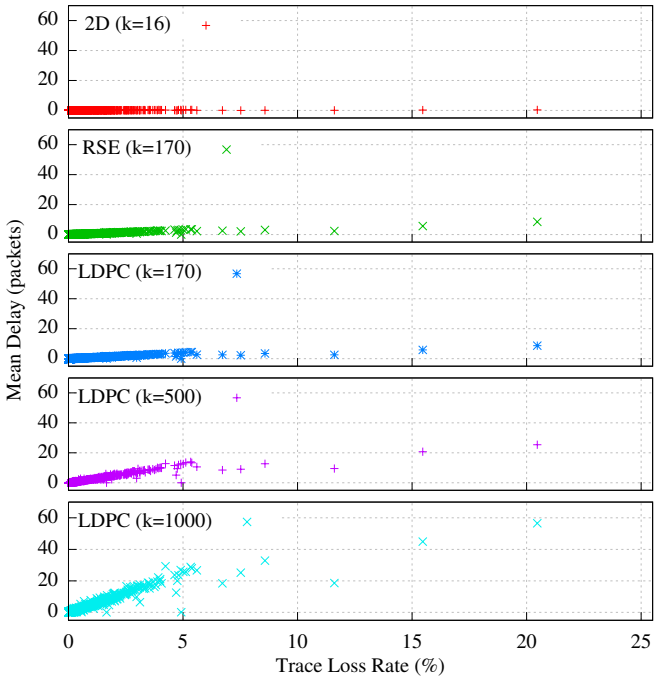
The results presented in Figure 1b are independent of the packet transmission rate. Therefore, they are comparable with, but do not directly replicate, those in [5], which assume a particular transmission rate and include CPU processing time for the FEC. Section VI discusses the impact of sending rate.

B. Applying FEC to loss traces

Figure 2a shows results of applying the FEC schemes to the loss traces from [4]. Since most of the traces show quite low loss rates, we limit the scale to focus on the 0–25% range. Observe that unlike the random loss results (where all the FEC schemes except 2D parity show almost full recovery under 25% loss), there are traces in this range where not all loss is recovered. Table I gives the percentage of traces, for each FEC scheme, which have 0% residual loss. Note that having



(a) Trace loss rate vs. residual packet loss rate



(b) Trace loss rate vs. mean FEC delay

Fig. 2. FEC performance on measured trace data

such a high percentage of low loss traces is not unusual for a well-engineered network. However, those traces which show significant loss are common enough to have a severe effect on user experience. A common target for IPTV services is to have no more than one visible artefact per two hours (or fewer). This corresponds to packet loss of $\sim 10^{-6}$ [7].

FEC Scheme	Num traces	Percentage
2D ($k = 16$)	3264	85.69%
RSE ($k = 170$)	3728	97.87%
LDPC ($k = 170$)	3724	97.77%
LDPC ($k = 500$)	3780	99.24%
LDPC ($k = 1000$)	3789	99.47%

TABLE I
NUMBER OF TRACES WITH 0% RESIDUAL LOSS RATE

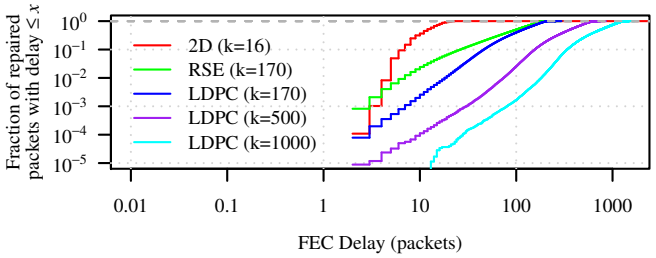


Fig. 3. CDFs of FEC delays from loss traces

It is clear that 2D FEC has the poorest recovery performance, as in [5], since it has the lowest fraction of traces being fully repaired (see Table I). This lower fraction produces the higher points in the top panel of Figure 2a; some of these are clustered near to the diagonal, showing that the performance is little better than it would be without FEC.

The RSE and LDPC schemes perform equally well for $k = 170$. Performance improves using LDPC codes with larger block sizes (larger k). However, unlike the random loss results seen in the previous experiments, there are cases of low loss (less than 10%) where the losses in the trace data cannot be recovered, resulting in residual loss rates above zero. These traces contain loss patterns which overwhelm the capacity of FEC to recover within certain blocks (e.g., large loss bursts). This result reminds us that the relationship between packet loss and FEC performance is not straightforward. It is also worth noting that there are cases where none of the FEC schemes are able to repair the loss (even with a large FEC overhead); in such cases, retransmission-based recovery will be necessary.

Figure 2b shows trace loss vs. per-trace mean FEC delay, as discussed in Section IV-B. We see a similar shape for all schemes, with the overall trend showing higher delays with higher trace loss rates (since more packets need to be repaired). Moreover, mean delay increases with block size; the maximum per-trace mean delay is $\sim$1 packet for 2D ($k = 16$), $\sim$10 packets for RSE and LDPC with $k = 170$, and between 50 and 100 packets for LDPC with $k = 500$ and $k = 1000$.

Mean delay, while useful, does not tell the whole story, since video quality depends not on the average delay, but on how often packets exceed their deadline and cause visible distortions to playback. Figure 3 shows the cumulative distribution of per-packet delays for repaired packets under each FEC scheme. The 2D scheme shows the lowest delays, with no packets delayed more than 24 packets. The RSE and LDPC

schemes with $k = 170$ show higher delays, although less than 2% (RSE) and 4% (LDPC) are more than 200 packets (i.e., correlated with block size). The larger block LDPC schemes show similar worst case delays up to their block sizes (500 and 1000 packets); such large FEC delays make these schemes less attractive for real-time use, despite their better recovery performance, since increased delay results in higher channel change times, degrading user experience. Note however, that these results reflect the worst case, and that over 99.98% of packets are not delayed by FEC (for any scheme). These results highlight the trade-off between optimising recovery performance (larger blocks), and minimising worse-case FEC delay (block size $\approx$ worst-case delay).

C. Understanding performance differences

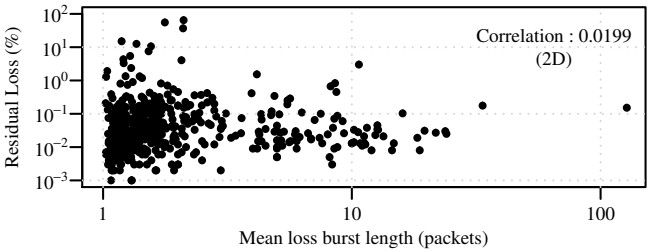
Comparing the results between random loss (Section V-A) and the measured loss traces (Section V-B), it's clear that performance is different. While most of the loss traces show loss rates less than 10%, FEC performance on the traces does not match that of random loss below 10%. The difference is due to the loss patterns in the traces, which exhibit bursty rather than uniform random loss, as described in [4].

However, the length of packet loss bursts is not the only factor affecting FEC performance. Indeed, looking at the mean loss burst lengths (which have been used to analyse FEC performance [16]) does not explain the results in the traces, since some of the traces with poorest performance have short mean burst lengths, as shown in Figure 4a. The reason for this is that there are periods where many packets are lost in short bursts next to short bursts of received packets; while the loss bursts are short, the overall loss patterns are highly bursty.

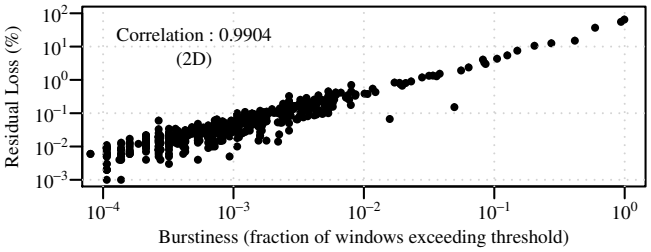
To measure this effect we calculate a different metric, which splits the traces into windows, and counts the number of lost packets and distinct loss bursts in each window. For each trace, we count the number of windows where there are either more than N lost packets or more than M loss bursts (the count for N lost packets is so that windows containing long loss bursts are also counted), and obtain the fraction of the trace that is bursty by dividing this count by the total number of windows.

We choose a window size of 16 packets, since this corresponds to the smallest of the k values we are considering. Also, if the packets are being transmitted at 500 packets per second (as we will discuss in Section VI), the 16 packet window equates to 32ms, which is roughly equal to one frame of video playback at 29.97fps ($1/29.97 \approx 33.33$ms). We choose $N = 4$ packets, since the 99^{th} percentile of all loss burst lengths in the loss traces is 4, suggesting that loss bursts longer than this are unusual. We choose $M = 4$ packets, since 4 loss bursts in a window of 16 suggests loss is reaching the point where repair is ineffective (as seen in Figure 1a).

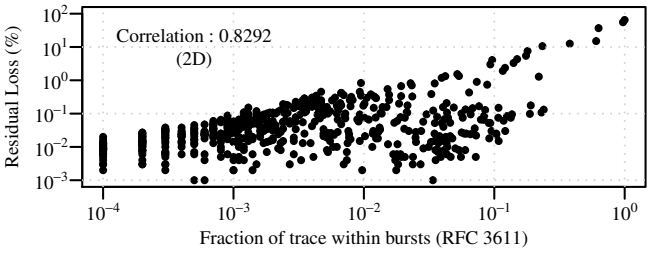
Figure 4b shows the results of plotting this window-based burstiness metric, for 2D FEC. We also compute the correlation between the burstiness and the residual loss rate; the strong correlation (0.9904) indicates this metric is more suitable than the mean burst length (which has a correlation of 0.0199). This window-based burstiness metric performs better



(a) Mean loss burst length vs. residual loss rate (2D FEC)



(b) Trace burstiness (windows) vs. residual loss rate (2D FEC)



(c) Trace burstiness (RFC 3611) vs. residual loss rate (2D FEC)

Fig. 4. Correlating FEC performance with bursts

than the burst loss metric of RFC 3611 [17] (Figure 4c), which identifies bursts as periods starting and ending with a loss and containing no receive runs above a threshold length G_{min} (we use the recommended value of 16). The poorer performance is probably due to the RFC 3611 metric being too sensitive, and describing a whole period as bursty, even if FEC recovery might be possible. Our window-based metric is more accurate since it considers how much loss is tolerable. Adjusting the threshold length G_{min} used in the RFC 3611 metric can achieve better performance, but the correlation with FEC performance is only stronger than our window-based metric when G_{min} is ≤ 3, at which point the RFC 3611 metric does not distinguish between "good and poor quality periods", as was the aim in [17]. The values in Figure 4 are for the residual loss rates after 2D FEC; correlations are similar for the other schemes, but are not shown due to space limitations.

VI. Recommendations

To recommend which of these schemes is most appropriate for use on residential networks, we look at their ability to recover from losses, and the delay they introduce. So far, we have considered the FEC delay in terms of the number of packets sent in the stream between packets being lost and being recovered. However, by considering the sending rate

of these packets, we can calculate the delays introduced by the FEC in terms of time. If we consider rates of 5Mb/s, which can be received by the typical residential broadband user ([18] found that average home downstream bandwidth was 6.2Mb/s), and 1316-byte RTP packets carrying MPEG video, we get a packet rate of 500 packets per second (pps). Using this rate, we can estimate the latency of the FEC. To achieve the overall 430ms latency bound suggested in [19], a constraint of 200ms for the FEC latency seems appropriate, allowing time for other components of channel-change latency.

Looking at the results, we see that both the RSE and LDPC schemes with $k = 170$ show better recovery performance than 2D FEC, although with higher delays. Figure 3 shows the fraction of *repaired* packets exceeding the 200ms limit for FEC delay (100 packets at 500pps) is 50% for RSE, and 60% for LDPC ($k = 170$), although only 0.075% (RSE) and 0.09% (LDPC) of *all* packets exceed the threshold. For a threshold closer to the FEC block size (e.g., 200 packets), only 1.5% (RSE) and 4% (LDPC) of repaired packets would exceed it.

We re-simulated the performance of LDPC with smaller block sizes, to reduce the worst-case delay and measure the effect on recovery performance. The parameters were ($k = 67$, $r = 33$) and ($k = 80$, $r = 20$), with 50% and 25% overhead, respectively (since many traces have low loss rates, we also look at the effect of reducing FEC overhead). Although space limitations prevent a full analysis of these results in this paper, we find that the percentage of traces fully repaired is 95.59% with ($k = 67$, $r = 33$), and 92.02% with ($k = 80$, $r = 20$), compared with $> 97\%$ for the schemes in Table I. Therefore, tuning is necessary to choose the trade-off between desired worst-case delay, overhead, and repair performance.

Considering the higher CPU overhead for the RSE scheme recorded in [5], we agree with the conclusions of [5] that LDPC codes are the most suitable choice among the schemes compared. However, we find that the suggested block size of $k = 170$ is too high for use with a delay bound of 200ms at 5Mb/s, and recommend instead that the LDPC block size is tuned so that the worst-case FEC delay does not exceed the delay bound. It should also be noted that none of the FEC schemes recover *all* lost packets when the loss is too high. In such cases, it is necessary to use retransmission alongside FEC to recover loss, although doing so will add one round-trip time of latency, to account for the retransmission request. Note that infrastructure to support retransmission is likely already present in the system, to support rapid channel-change [20].

VII. Conclusions

In this paper, we have evaluated the performance of three AL-FEC schemes; 2D, RSE, and LDPC-Staircase codes, under loss conditions measured for IPTV-like traffic on residential broadband networks. We have shown that since the measured loss is bursty, the FEC performance doesn't simply follow the input loss rate as suggested in [5], but is also affected by the loss burstiness. Looking at burstiness, we have also demonstrated that the mean burst length is not sufficient to predict FEC performance, since it does not consider the periods of short, clustered loss bursts. We have presented a metric which better correlates with residual loss rate than the burst loss metric presented in [17].

We have found that LDPC-Staircase codes give the best trade-off between recovery, latency and computational cost (a finding consistent with that of [5]), although the worst-case delays incurred when $k = 170$ are too high to be used for Internet streaming. However, we show that smaller block sizes can be used without harming recovery performance too much. Since there are periods of loss that are unrecoverable using FEC (even with high levels of overhead), we recommend that retransmission-based recovery is also used.

Future work will focus on further exploring the trade-off between overhead, latency, and repair performance for LDPC codes, and evaluate the FEC schemes in real-time video streaming to residential users. This work might also evaluate video quality for specific encoding schemes and content, to better understand the effect of ADSL/Cable packet loss on user experience for streaming video.

Acknowledgements

This work was supported by Cisco Research and EPSRC.

References

[1] SMPTE, "FEC for Real-Time Video/Audio Transport Over IP Networks," SMPTE 2022-1, 2007.

[2] J. Lacan *et al.*, "Reed-Solomon Forward Error Correction (FEC) Schemes," IETF, 2009, RFC 5510.

[3] V. Roca *et al.*, "Low Density Parity Check Codes (LDPC) Staircase and Triangle FEC Schemes," IETF, 2008, RFC 5170.

[4] M. Ellis *et al.*, "End-to-End and Network-Internal Measurements of Real-Time Traffic to Residential Users," in *Proc. ACM MMSys*, 2011.

[5] K. Matsuzono *et al.*, "Performance Analysis of a High-Performance Real-Time Application with Several AL-FEC Schemes," in *Proc. IEEE LCN*, 2010.

[6] H. Schulzrinne *et al.*, "RTP: A Transport Protocol for Real-Time Applications," IETF, 2003, RFC 3550.

[7] A. C. Begen, "Error Control for IPTV over xDSL Networks," in *Proc. IEEE CCNC*, 2008.

[8] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, 2006.

[9] "DVB Application Layer FEC Evaluations," DVB Document A115 - TM 3783, 2007.

[10] M. Luby *et al.*, "Application layer FEC in IPTV services," *IEEE Commun. Mag.*, vol. 46, no. 5, 2008.

[11] E. Mammi *et al.*, "Evaluation of AL-FEC performance for IP television services QoS," in *Proc. SPIE*, 2010.

[12] S.-R. Kang and D. Loguinov, "Modeling Best-Effort and FEC Streaming of Scalable Video in Lossy Network Channels," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, 2007.

[13] S. Naegele-Jackson *et al.*, "Multi-layer Performance Measurements over Optical Testbeds and QoS Provisioning for High-Bandwidth Video Applications," in *Proc. BROADNETS*, 2006.

[14] J.-C. Bolot *et al.*, "Analysis of Audio Packet Loss in the Internet," in *Proc. NOSSDAV*, 1995.

[15] C. Perkins *et al.*, "A survey of packet loss recovery techniques for streaming media," *IEEE Network*, vol. 12, no. 5, 1998.

[16] P. Frossard, "FEC Performance in Multimedia Streaming," *IEEE Commun. Lett.*, vol. 5, no. 3, 2001.

[17] T. Friedman *et al.*, "RTP Control Protocol Extended Reports (RTCP XR)," IETF, 2003, RFC 3611.

[18] C. Kreibich *et al.*, "Netalyzr: Illuminating The Edge Network," in *Proc. ACM IMC*, 2010.

[19] R. Kooij *et al.*, "Perceived Quality of Channel Zapping," in *Proc. IASTED CSN*, 2006.

[20] B. Ver Steeg *et al.*, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions," IETF, 2011, RFC 6285.