QUIC FEC v1

https://www.chromium.org/quic
Author: ianswett@google.com
Last Updated: 2016-02-19

Background

Forward Error Correction (aka FEC) allows extra bytes to be transmitted to provide redundancy in case not all packets arrive. QUIC implements a form of FEC based on XOR, which is both simple and fast and provides N+1 redundancy. FEC is commonly used to improve link-layer reliability (such as in Wifi), and there has long been interest in adding FEC to the end-to-end transport layer, but this is prohibitively complex in TCP. As such, QUIC provides an ideal environment to experiment with FEC and to determine in what cases it may be able to provide latency or reliability advantages.

Summary

The first version of QUIC's FEC implementation had minimal effects on web latency and degraded YouTube QoE metrics, while consuming more bytes on the wire and increasing code complexity, so it will be deprecated. For applications where FEC demonstrated benefit at the tail, a simpler and more effective option is a more aggressive Tail Loss Probe (TLP IETF draft). Some results from that experiment are included here. Design of a new QUIC FEC approach has begun, with input from others who have had past success with FEC.

QUIC’s XOR FEC

QUIC’s current FEC has packet granularity. The FEC scheme covers entire packets in a QUIC connection and uses XOR to reconstruct a packet when one is lost. This XOR-based system has a concept of a ‘group’ of packets, any one of which could be lost and recovered by a final XOR packet. The group number is indicated on each packet in the group, as well as on the XOR packet sent to protect the group. It is valid to mark packets as belonging to an FEC group, but not send the XOR packet.

Since FEC packets consitute load on the network, QUIC treats FEC packets identically to data packets from a congestion control perspective. When an FEC packet is sent, it consumes CWND that could be used for data packets, and if an FEC packet is lost, it causes a reduction in CWND. We note that while QUIC counts FEC packets towards outstanding bytes, QUIC's congestion window does not block the sending of an FEC packet.

Advantages

XOR-based FEC has low computational overhead and is relatively simple to implement. The original packets don’t change, so XOR FEC maintains incremental data processing, unlike some FEC schemes that require multiple packets to arrive before any can be processed.

Disadvantages

XOR-based FEC can only recover one packet. If two or more packets are lost, it ends up wasting the FEC packet, because QUIC’s FEC is packet based and QUIC’s retransmission is frame based, so resent frames don’t help recover missing packets. Instead, all lost packets must be retransmitted. The need to indicate a group number when the packet is sent makes it more difficult and less efficient as a replacement for a Tail Loss Probe.

Loss Distribution

All FEC approaches are wasteful when there is no packet loss. In addition, QUIC’s is wasteful if more than one packet per FEC group is lost. Therefore, it’s helpful to know how often only a single packet is lost in a group, to predict how likely QUIC’s FEC is to be effective.

Pre-Launch Experiments

Before QUIC was used publicly, experiments were done where Chrome sent 20 packets in a burst or paced at a bottleneck bandwidth rate. These indicated approximately 65% of packet losses could be repaired by a single FEC packet. This data, while initially useful, did not directly measure value of FEC in a congestion-controlled transport flow with other loss recovery schemes in place.

Post-Launch Data

More recently, QUIC has started to record detailed information on the distribution of losses. Current data from Chrome Stable indicates only 28% of loss events(losses that occur within an epoch) are a single packet, and the rest are larger.

Experimental Results

QUIC experimented with a variety of policies for when to send and what to cover with XOR-based FEC. All FEC group sizes were variable and set to ½ the current CWND in packets.

1. Protecting packets containing headers. QUIC headers are head-of-line blocking to other headers, so it was believed FEC could mitigate that.
    a. Results for this experiment are not included, as they were insignificant.
2. Protecting packets containing headers and/or body.
3. Sending an FEC packet after a quarter of an RTT of quiescence to cover the last half an RTT of sent packets. This is aggressive TLP-style FEC.

Critical Metrics

YouTube Join Latency: The time before video playback begins.
YouTube Rebuffer Rate: The percent of playback time the video fails to play.(aka ‘the spinner’).
Search Latency: The amount of time it took to load a Google search result.
Chrome Latency: The amount of time a QUIC request took, as measured by Chrome.

FEC on headers and body

An experiment to enable FEC on both headers and body of a stream was run in Chrome Stable. Due to the FEC flag never being enabled in Chrome, this experiment had no effect on data sent from the client to the server, only from the server to the client.

YouTube

Statistically significant results

- Median and mean join latency increased
- Rebuffer rate increased

Search Latency

- No significant change

Chrome Latency

- FEC improved mean latency slightly and degraded median latency slightly, though neither change is statistically significant.

FEC after ¼ RTT of quiescence

The QUIC team enabled an experiment later in Chrome Dev, and this one had the client as well as the server sending FEC. At the same time, another experiment was enabled to send a TLP after ¼ RTT of quiescence. That allows a comparison of potential improvements that consumes a similar number of bytes on the wire.

YouTube

Statistically significant results
- Median and mean join latency increased
- Rebuffer rate increased

Search Latency

- FEC caused no significant change in latency
    - TLP after ¼ RTT did slightly improve mean latency

Chrome Latency

- FEC is slightly, but not statistically significantly slower at the mean.
- Aggressive TLP is slightly, but not statistically significantly faster at the mean.

Observations

The FEC experiment had a small, and mostly insignificant effect on latency, with the exception of YouTube. YouTube is commonly bandwidth limited, particularly at the beginning of playback, so any extra packets tend to degrade join latency.

As a general note, FEC is a *proactive* loss recovery scheme, which complements the set of *reactive* loss recovery schemes (e.g. Fast retransmit, TLP, RTO) that are currently implemented in QUIC. It is possible that the proactive scheme does not provide measurable benefits in the presence of QUIC's reactive schemes.