QUIC FEC

Approaches and APIs

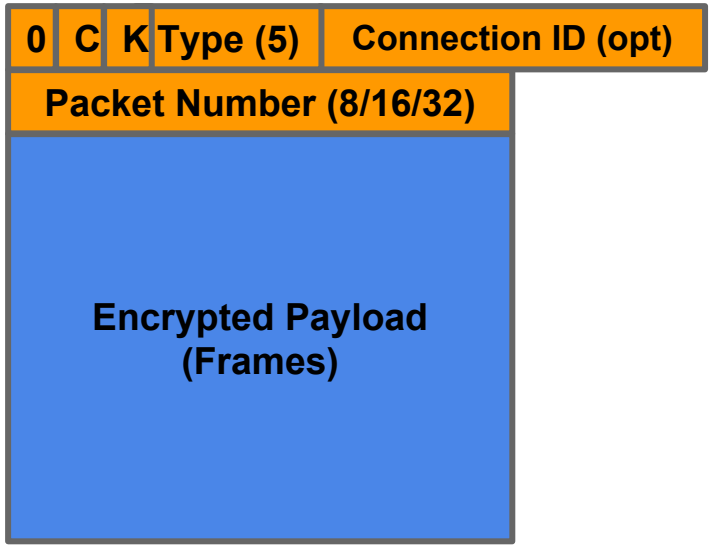
draft-quic-coding-00

Ian Swett, NWCRG@IETF100

QUIC Header

- 1 byte for ‘type’
- 8 byte Connection ID
- 1, 2 or 4 byte packet number

Short Header Packets (optimized for packets encrypted with TLS 1-RTT key)



Why FEC in QUIC?

- We can!
- Realtime Communications
- QUIC Tunnels
- Multicast QUIC(someday)
- Efficient Tail Loss Probe

QUIC features which may help

QUIC has a monotonically increasing packet number

Multiple QUIC flows can share a 5-tuple by using different connection IDs

QUIC provides multiplexed streams which do not block one another

Option 1: FEC outside the crypto

Use a second QUIC connection ID

Pros:

- Can code within the network without end-to-end coordination
- Can use QUIC packet numbers to indicate what is protected
- Does not change the core sequence of QUIC packets

Cons:

- Network visibility into the FEC
- May need FEC ‘termination’ separate from end-to-end
- More difficult to integrate with QUIC’s congestion control

Option 2: FEC within the encrypted payload

Define a new frame type for FEC data and how it protects adjacent packets

Pros:

- No network visibility into the FEC (end-to-end encryption)

Cons:

- CPU cost of coding and encryption
- Consumes a byte to indicate the rest of the packet is coding

Option 3: FEC in the crypto as a stream

Use one or more QUIC streams to send FEC

Pros:

- May work well with existing applications using FEC(ie: RTP)
- Can implement without transport changes

Cons:

- Application specific
- May increase overhead vs packet based coding
- Can’t leverage QUIC packet numbers

Opinion: Option 2 seems promising

Relatively easy to negotiate a new frame in QUIC in the handshake

Should work well with a variety of codes

Crypto is fairly cheap, so coding and crypto is likely a non-problem

Not Observable*

API thoughts

Configuration

1. Know how much overhead FEC needs
2. Is coding rate is dynamic?
3. Can I send data and then coded bytes or only coded bytes?
4. Set coding rate
5. Set coding length

Runtime

1. Add data to be protected
2. Request coded bytes to be sent

Challenges and Questions

FEC adds a small amount of overhead

- Solution: Don’t use the full packet size for non-FEC packets?

  => Need to know the max overhead FEC can add

Ensure the generated code will protect missing packets

- ie: I lost 2 packets one RTT ago and only sent one coded packet, can I generate coded content to recover them now?

Does the API need to understand packet numbers or is there a QUIC specific ‘shim’ to interface with a standard API and provide framing?

QUIC Implementations

Want to experiment?

QUIC implementations:

https://github.com/quicwg/base-drafts/wiki/Implementations