University of Glasgow

School of Computing Science

60 YEARS OF COMPUTING AT GLASGOW

QUIC Multiplexing and Peer-to-Peer

draft-aboba-avtcore-quic-multiplexing-01

Bernard Aboba – Microsoft
Peter Thatcher – Google
Colin Perkins – University of Glasgow

With input from Martin Thomson – Mozilla

Presentation given to IETF AVTCORE working group on 16 November 2017

QUIC?

- QUIC is a new transport protocol – runs over UDP
- Initial focus on transport for HTTP/2, but should become general purpose

- This work has two goals:
  - Enable peer-to-peer use of QUIC
  - Enable coexistence of QUIC and WebRTC on a single UDP port

Goal #1: Enable Peer-to-peer use of QUIC

- Initial use cases for QUIC are client-server: HTTP-over-QUIC
- If QUIC is to be general purpose, it must also support peer-to-peer use → will need to work through NAT

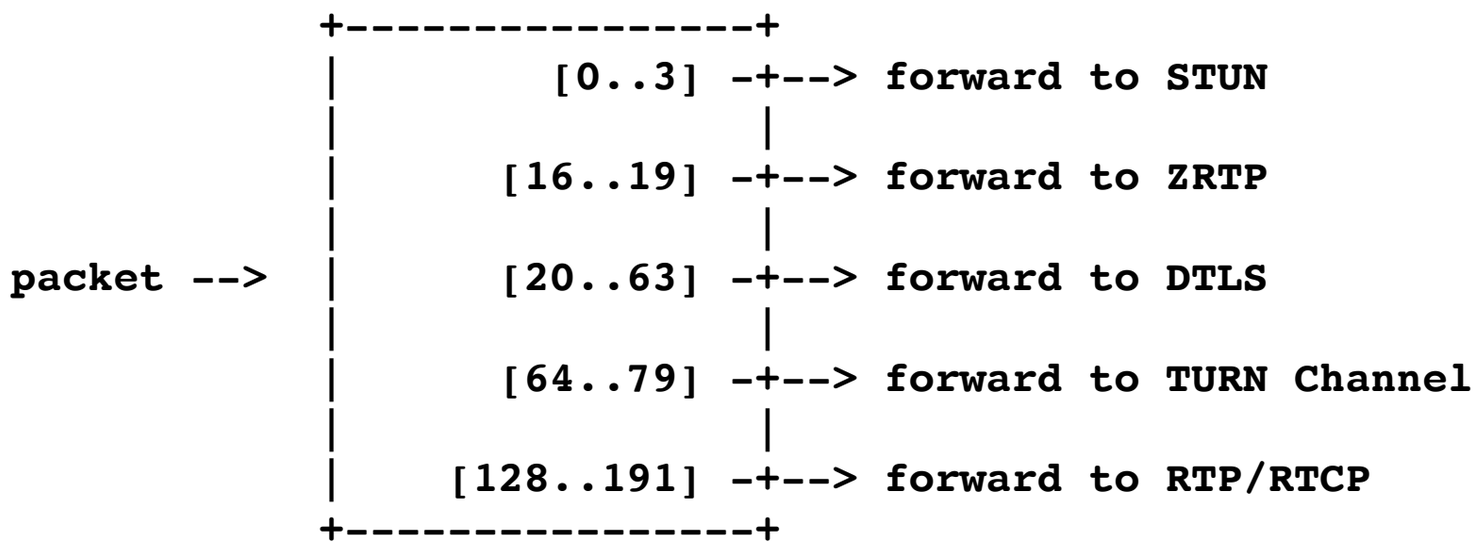
- NAT traversal enabled via ICE → signalling + STUN
  - Signalling we can do later
  - STUN *must* run on the same port as QUIC to work → *must* be able to demux STUN packets and QUIC packets
  - Or re-invent STUN as a QUIC sub-protocol – don't want to go there…

Goal #2: Coexistence with WebRTC

- WebRTC starting to see wide deployment
- Web servers starting to speak HTTP/QUIC rather than HTTP/TCP, might want to run WebRTC from the server to the browser
  - In principle can run media over QUIC, but will take time a long time to specify and deploy – initial ideas in draft-rtpfolks-quic-rtp-over-quic-01
  - SRTP key exchange could leverage QUIC handshake, but currently needs DTLS demux
- Potential use of QUIC as replacement for WebRTC data channel in peer-to-peer use
- Both require us to be able to demux QUIC packets and WebRTC packets (i.e., STUN, TURN, DTLS, SRTP, SRTCP)

How does WebRTC demultiplex?

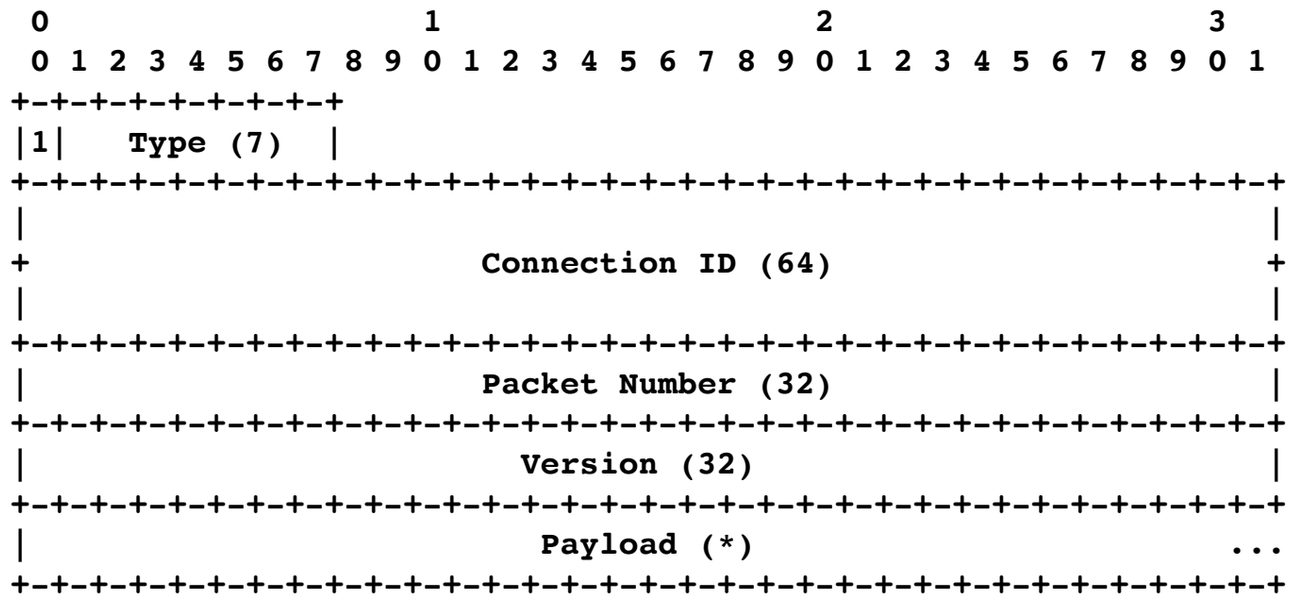
- RFC 7983 defines the demux used by WebRTC
  - On arrival, look at first octet of packet
  - Forward to higher-layer protocol according to the following:



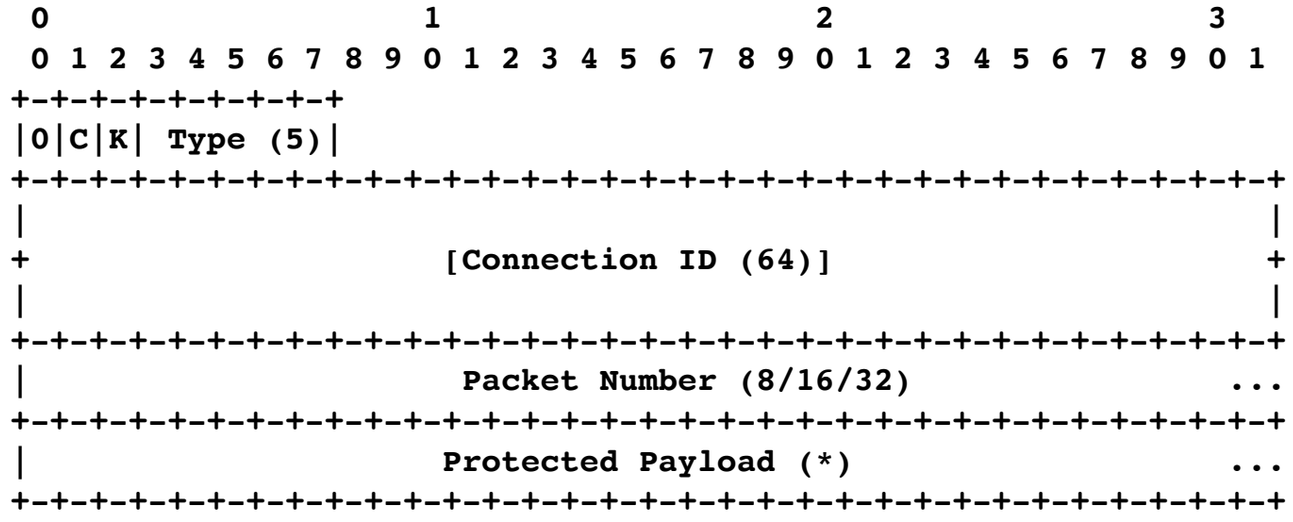
- This is a kludge, but works in practice for WebRTC
  - Demux of STUN+TURN needed for communication with TURN server, not for other uses
  - No clear extensibility strategy/IANA registration policy for new demux

Demuxing QUIC?

- QUIC long header packet
  - Initial bit set to 1
  - Types 0x01 - 0x06 currently defined
  - First octet in range 129-134
    → conflict with RTP/RTCP



- QUIC short header packet
  - Initial bit set to 0
  - C = 1 if Connection ID present
  - K = Key phase
  - Types 0x01 - 0x03 currently defined
  - First octet in range 1-3, 33-35, 65-67 or 97-99 → conflict with STUN, DTLS and TURN



Demuxing QUIC?

- Several solutions considered to avoid this conflict:
  - Option 1: rely on crypto – everything is authenticated so check signatures for each protocol, and forward to the one that succeeds
  - Option 2: rearrange QUIC packet formats, so all packets start with top two bits set to one to avoid collision → fit with RFC 7983 space
  - Option 3: add a single octet shim to the start of QUIC packets when used peer-to-peer (or likely always, to avoid ossification)
  - Option 4: avoid conflicts – we don't care about demux with TURN, DTLS can be replaced by QUIC keying, long packets only used during handshake → mostly fits with RFC 7983, if pay attention to sequencing of packets

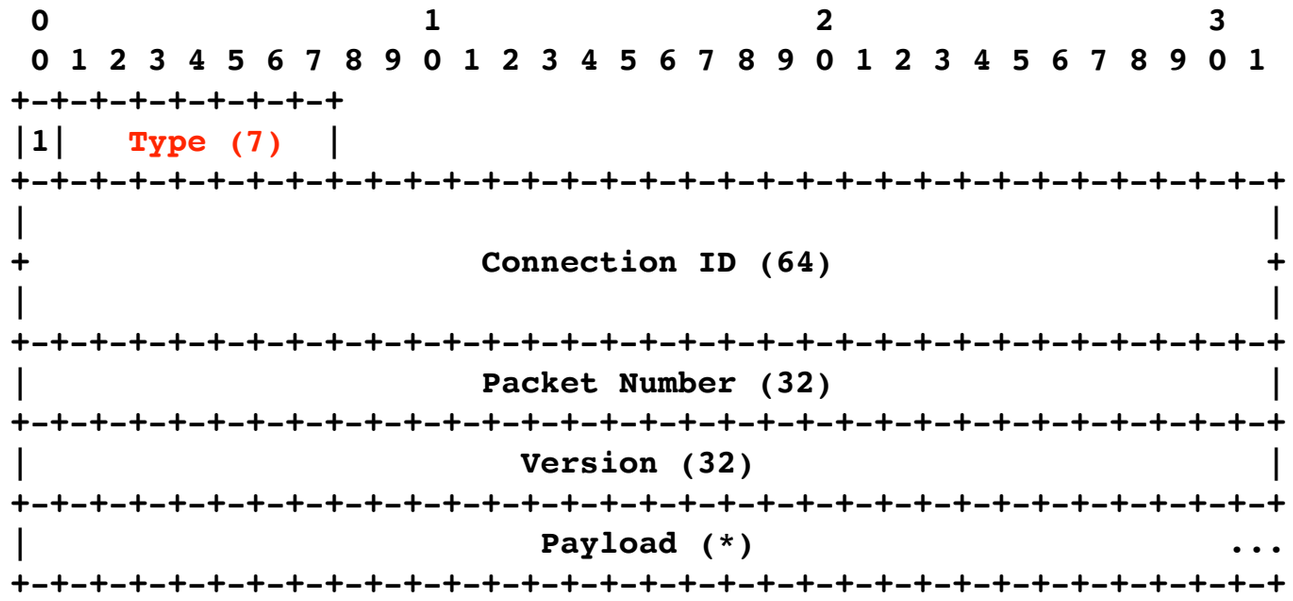
- See https://datatracker.ietf.org/meeting/100/materials/slides-100-quic-sessa-invariants/ for details (presented in QUIC on Tuesday)
- None of these are ideal

Demuxing QUIC?

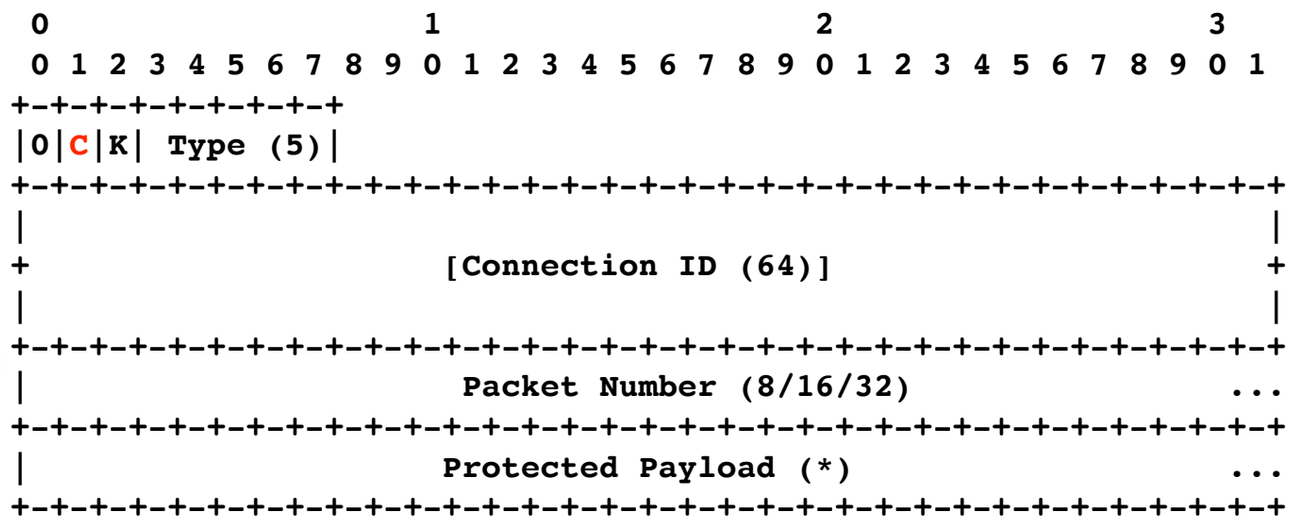
- Several solutions considered to avoid this conflict:
  - Option 1: rely on crypto – everything is authenticated so check signatures for each protocol, and forward to the one that succeeds
  - Option 2: rearrange QUIC packet formats, so all packets start with top two bits set to one to avoid collision → fit with RFC 7983 space
  - Option 3: add a single octet shim to the start of QUIC packets when used peer-to-peer (or likely always, to avoid ossification)
  - Option 4: avoid conflicts – we don't care about demux with TURN, DTLS can be replaced by QUIC keying, long packets only used during handshake → mostly fits with RFC 7983, if pay attention to sequencing of packets
  - Option 5: renumber QUIC packets to avoid collisions

Renumbering QUIC to Avoid Collisions

- QUIC long header packet
  - Initial bit set to 1
  - Renumber packet type field: 0x01-0x06 → 0x7F-0x7A



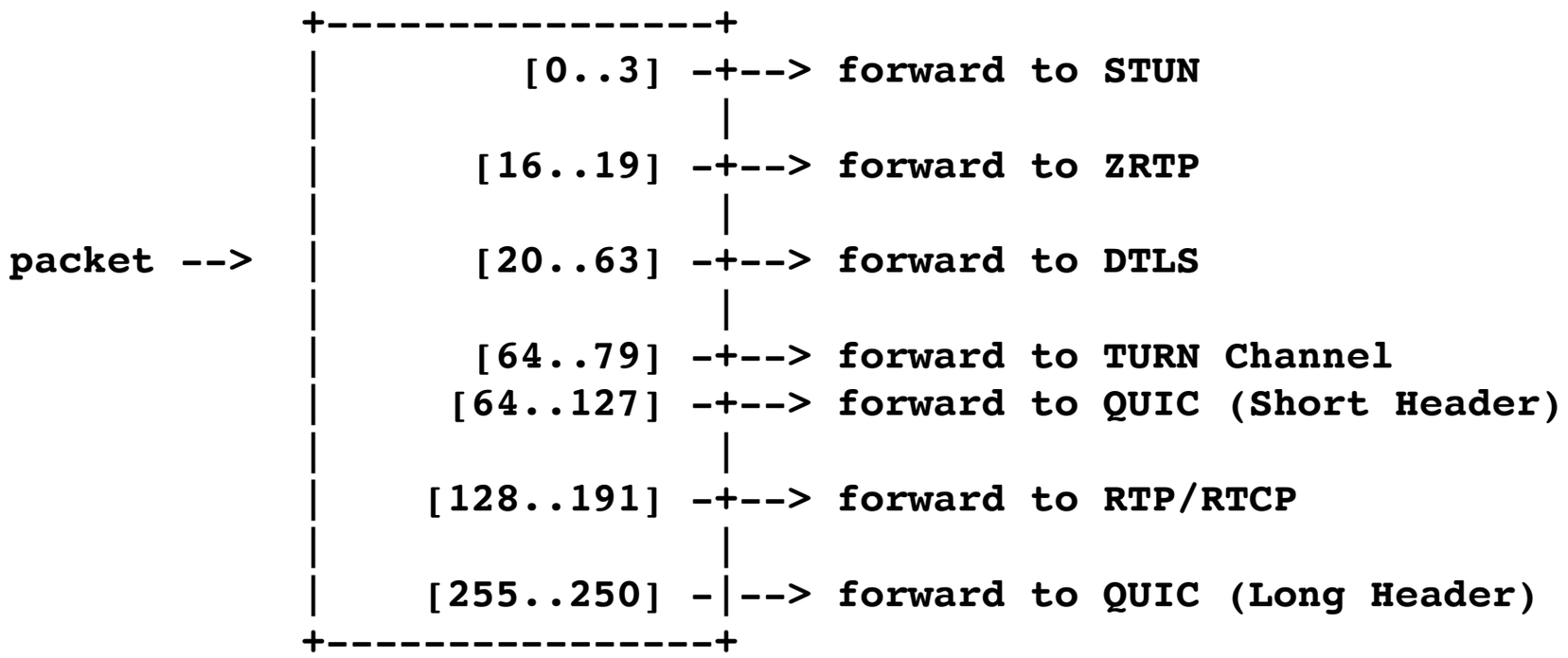
- QUIC short header packet
  - Initial bit set to 0
  - Invert sense of C bit = 0 if Connection ID present, 1 if absent



- Give guidance on greasing and demuxing of QUIC and other packets

Resulting Demultiplex

- QUIC Long Header packets don't conflict with others in the RFC 7983 scheme
- QUIC Short Header packets conflict with TURN → not believed problematic, since not likely to want to co-locate TURN server and QUIC server
- QUIC might use more of the space in future → need to give guidance to avoid conflicts *if* multiplexing in that case
  - Either by intentional use of other packet types, redefinition of QUIC header in future version, or greasing
- Update RFC 7983 or QUIC specific demux?



Conclusions

- We believe we have a workable demultiplexing solution for QUIC packets, STUN, and WebRTC
- Does this group agree the approach is workable?
- Next steps:
  - Submit pull request to draft-ietf-quic-transport to implement this change and discuss in QUIC WG
  - If accepted, consider if we should update RFC 7983
    - What is the extensibility strategy for RFC 7983?
    - Does QUIC conform to this strategy or does QUIC demux work differently?