UDP Packet Reordering

MAPRG: IETF 102

Ian Swett

Data from Chrome Stable and Google Servers

Google

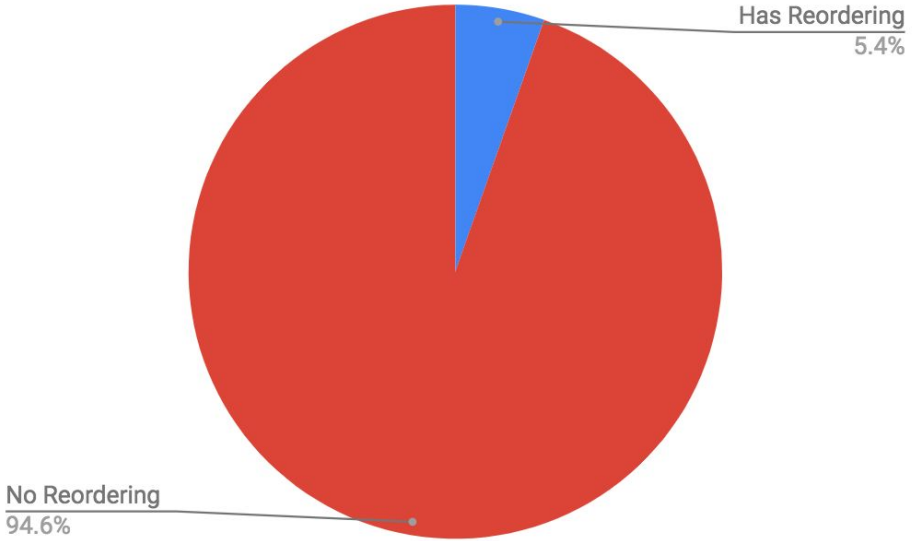
Outline

- Client side reordering data = Client(Chrome) received a packet out of order
  - Direction information, based on received packets
  - Server using BBR congestion control
  - Chrome Stable
  - Representative of bulk flow reordering
- Server side reordering data = Server received a packet out of order
  - Direct information, based on received packets
  - Client using Cubic congestion control
  - Only CDN nodes
  - Represents mostly receipt of handshakes, requests, etc

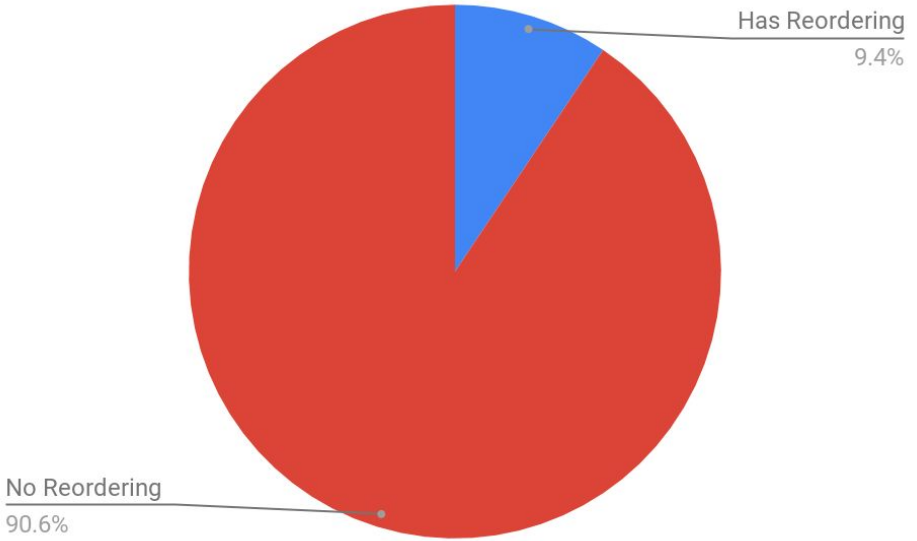
QUIC code here

Percent of Connections with at least one

Client (server sent)



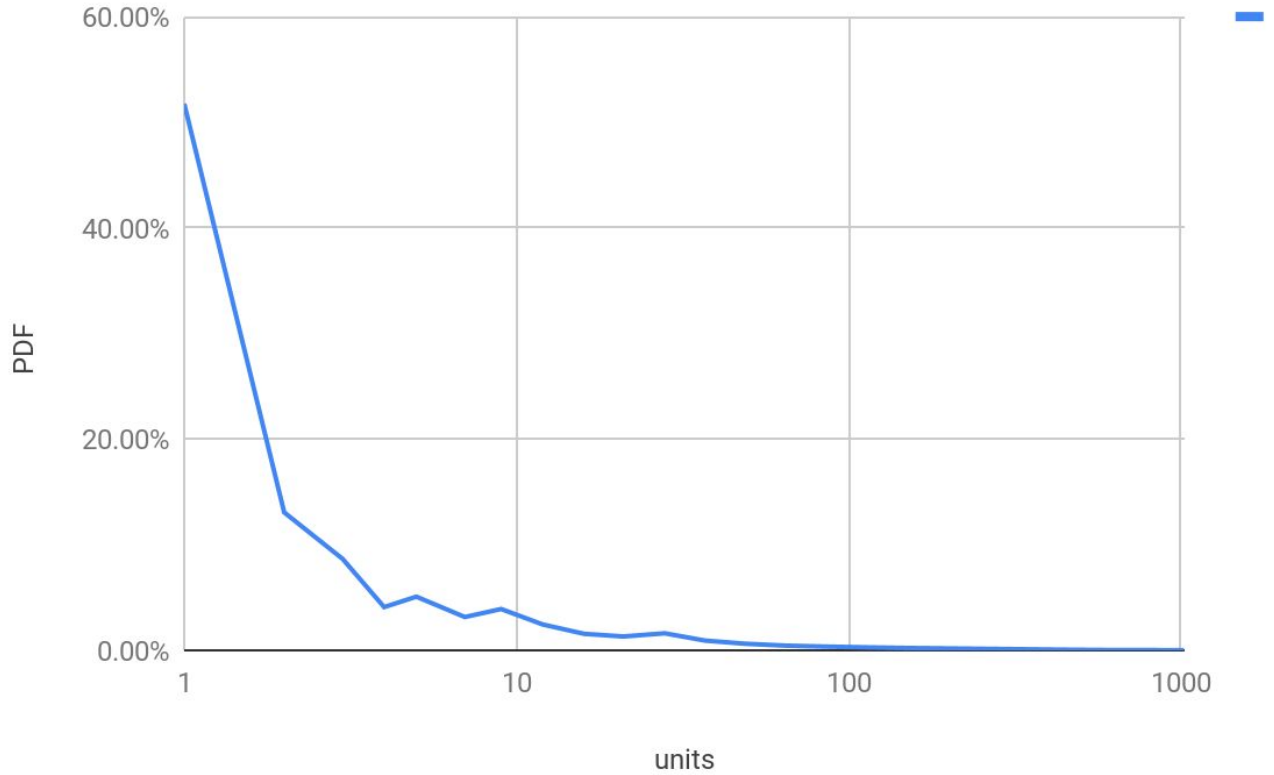
Server (client sent)



Fun data

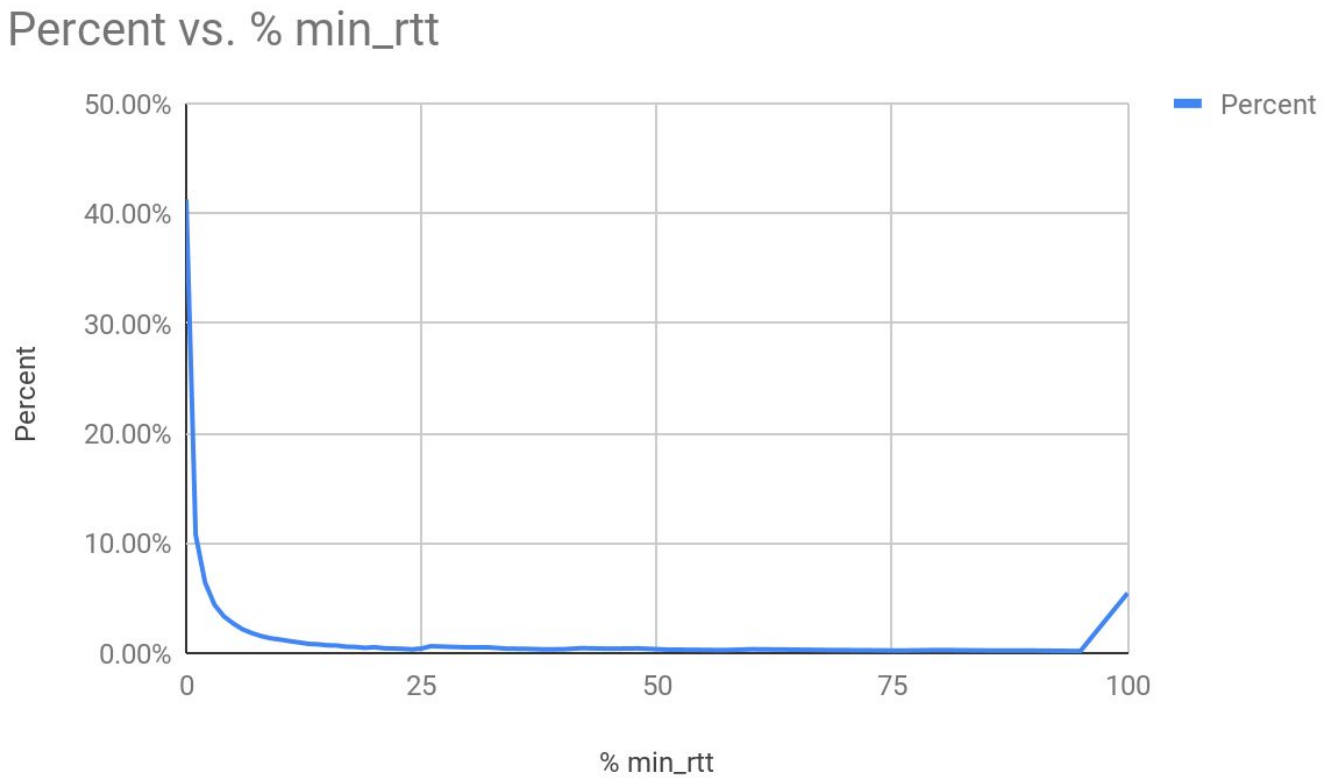
Remaining data **excludes** connections with no reordering

Client: Max gap in QUIC packet number



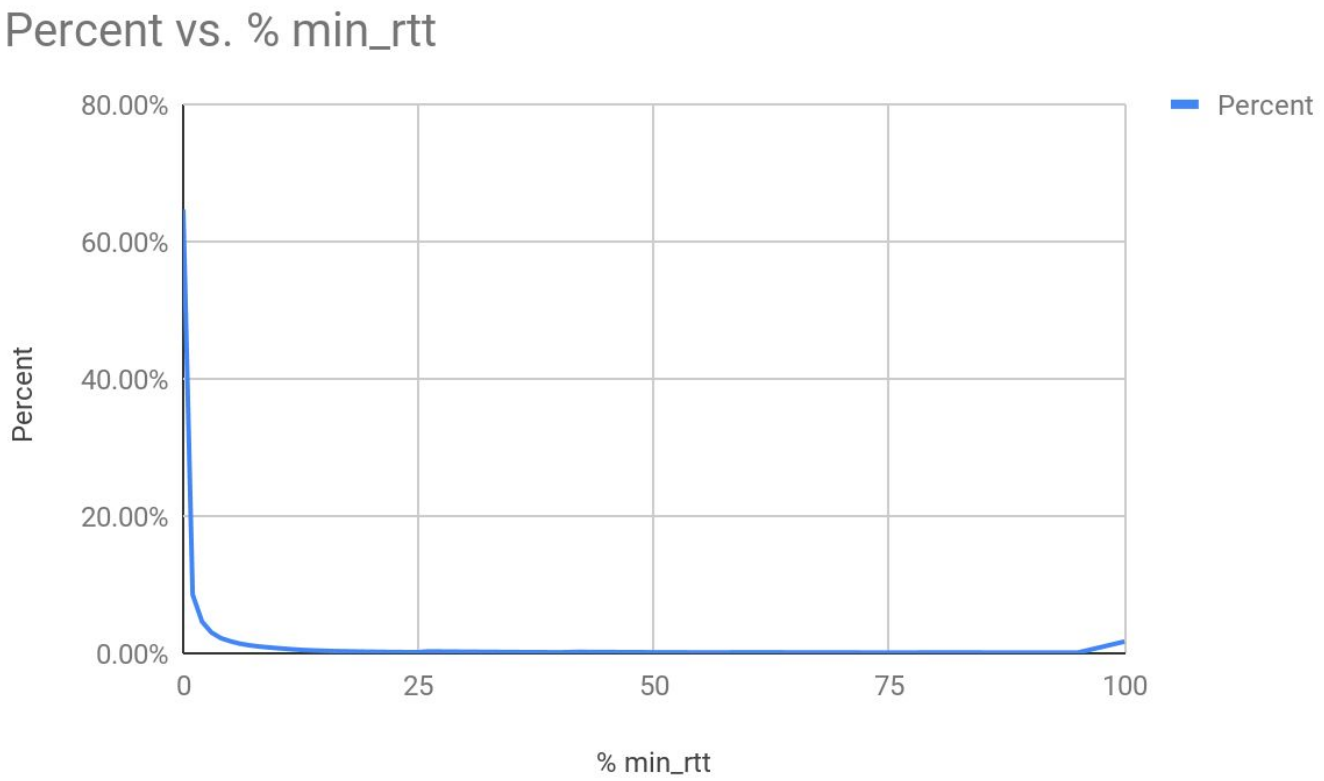
Note: Log X scale for packet numbers

Client: Max time in fraction of min_rtt

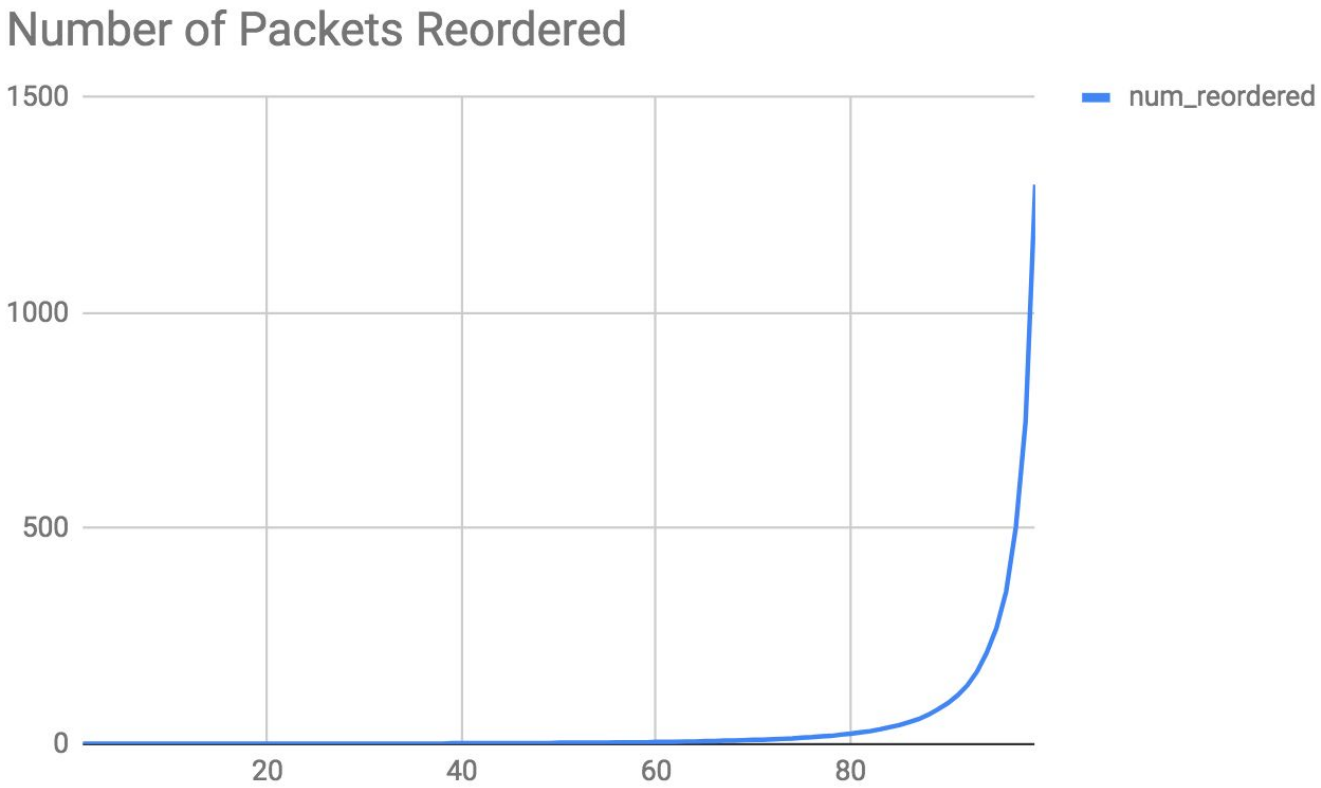


Note: 91.5% are less than 12.5% (recommended QUIC reordering threshold)

Client: Max time in fraction of min_rtt (min_rtt >100ms)

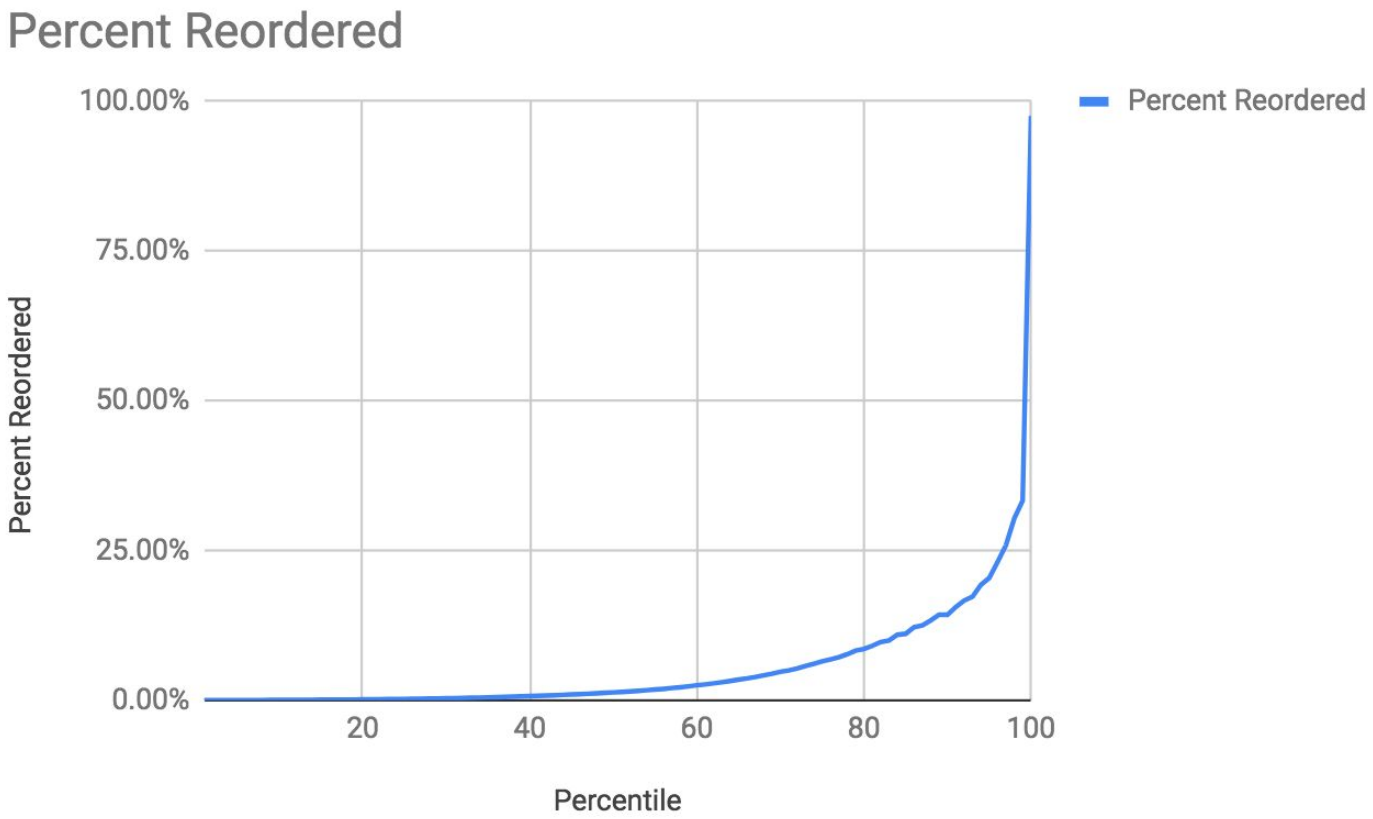


Server: Number Reordered

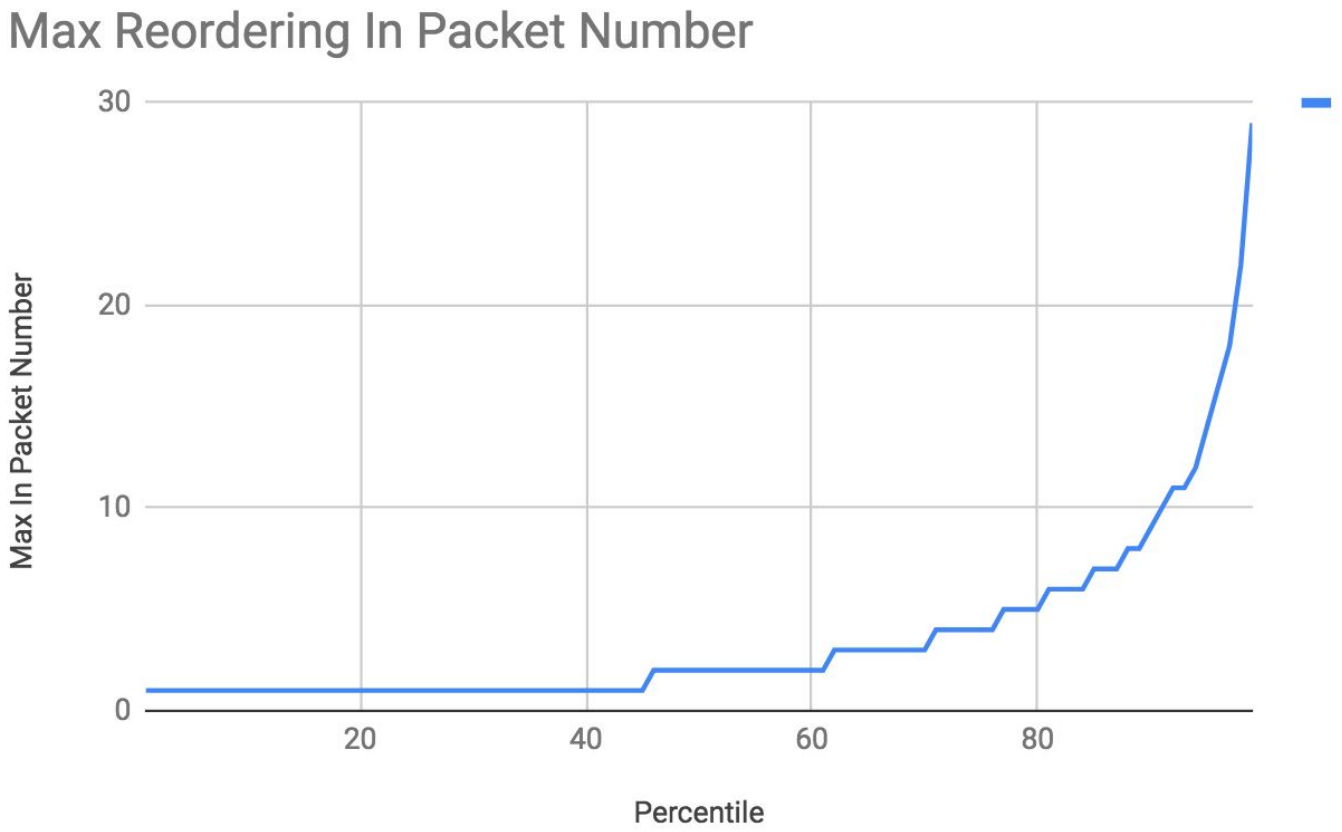


38% had only one packet reordered

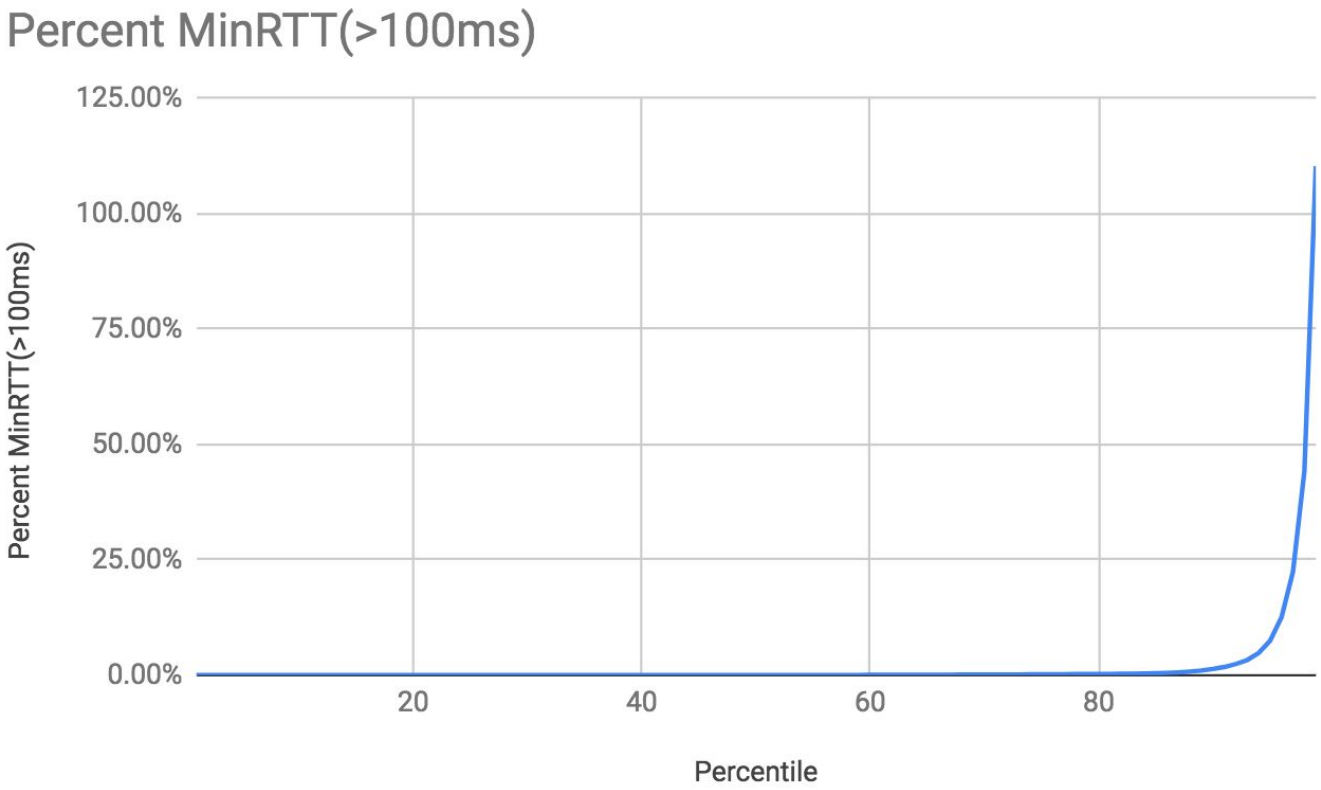
Server: Percent Reordered



Server: Max gap in QUIC packet number



Server: Max time in fraction of min_rtt (min_rtt >100ms)



Note: 96% are less than 12.5% (recommended QUIC reordering threshold)

Conclusion

- The vast majority of connections see no reordering
- The tail is very long
- QUIC runs in userspace, so small networking reordering may translate to a few ms of transport reordering
  - => TCP may see a bit less reordering
- ⅛ RTT reordering threshold in QUIC is large enough for >99% of connections(>100ms)
- Adaptive loss detection should consider starting with a very short threshold to minimize recovery time