QUIC

Internet-Scale Deployment on Linux

TSVArea, IETF 102, Montreal

Ian Swett

Google

A QUIC History - SIGCOMM 2017

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

Improved application performance

YouTube Video Rebuffers: 15 - 18%

Google Search Latency: 3.6 - 8%

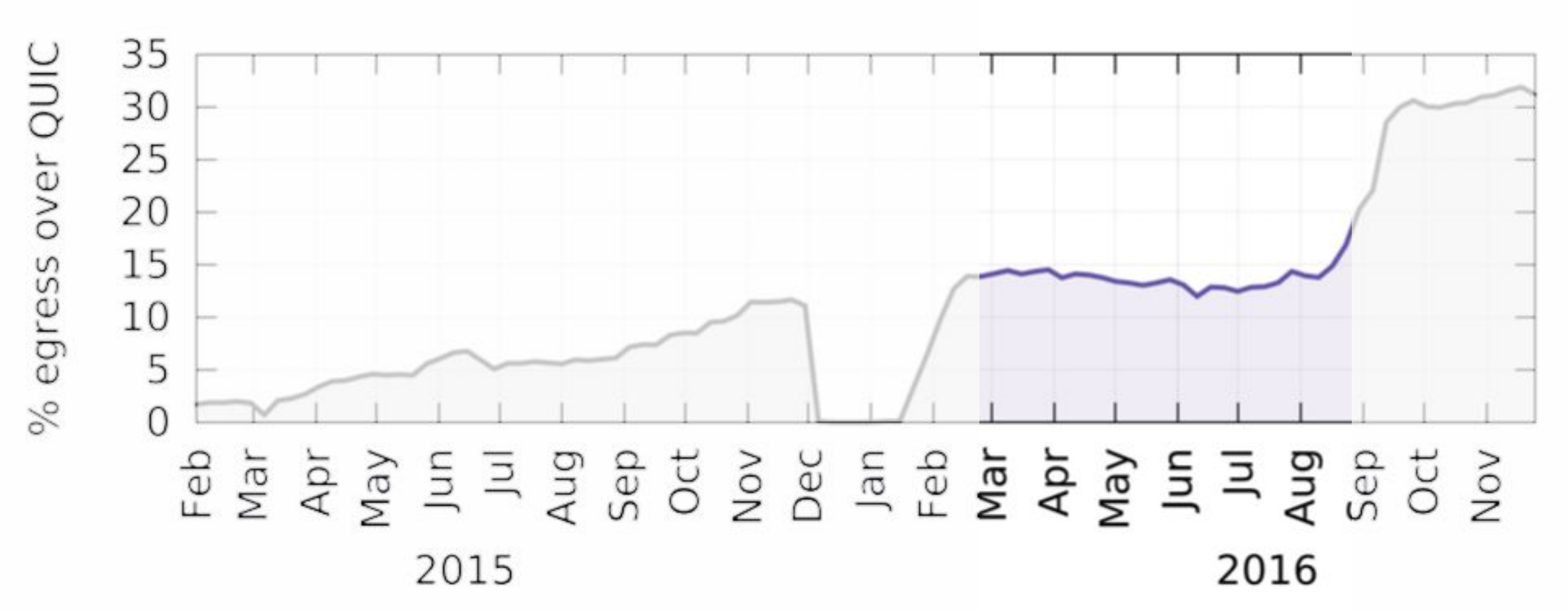
35% of Google's egress traffic (7% of Internet)

IETF QUIC working group formed in Oct 2016

Modularize and standardize QUIC

Google's QUIC deployment

QUIC vs. TLS/TCP CPU
3.5x → 2x



QUIC CPU Utilization: Major Sources

Crypto (esp. ChaCha20)

Sending and receiving UDP (sendmsg, recvmsg)

QUIC-internal state

Processing encrypted acks

QUIC CPU Utilization: Major Sources

Crypto (esp. ChaCha20)
- Used hand-optimized assembly
- Inplace encryption

Sending and receiving UDP (sendmsg, recvmsg)

QUIC-internal state

Processing encrypted acks

QUIC CPU Utilization: Major Sources

Crypto (esp. ChaCha20)

- Used hand-optimized assembly
- Inplace encryption

Sending and receiving UDP (sendmsg, recvmsg)

- PACKET_RX_RING, UDP GSO
  - (kernel bypass is still very lucrative)

QUIC-internal state

Processing encrypted acks

QUIC CPU Utilization: Major Sources

- **Crypto (esp. ChaCha20)**
  - Used hand-optimized assembly
  - Inplace encryption
- **Sending and receiving UDP (sendmsg, recvmsg)**
  - PACKET_RX_RING, UDP GSO
    - (kernel bypass is still very lucrative)
- **QUIC-internal state**
  - Improved cache efficiency, data structures
  - Minimize allocations and memcpy
- **Processing encrypted acks**

QUIC CPU Utilization: Major Sources

- **Crypto (esp. ChaCha20)**
  - Used hand-optimized assembly
  - Inplace encryption
- **Sending and receiving UDP (sendmsg, recvmsg)**
  - PACKET_RX_RING, UDP GSO
    - (kernel bypass is still very lucrative)
- **QUIC-internal state**
  - Improved cache efficiency, data structures,
  - Minimize allocations and memcpy
- **Processing encrypted acks**
  - Ack decimation: Reduce ack rate to ¼ RTT or 10 packets

‘Recommended’ Use of Sockets for QUIC

- Use socket per thread with SO_REUSEPORT for receive
  - Provides stable 4-tuple hashes among flows
  - App dispatches based on QUIC Connection ID

- NAT rebinding, conn migration are << 1% of conns
  - Relying on 4-tuple is mostly adequate
  - Tossing packets between threads for the rest
  - A BPF can provide CID-based steering

‘Recommended’ Use of Sockets for QUIC (cont’d)

- Use socket per thread for sending
  - send-socket per connection mostly impractical
  - also largely not beneficial

Issues

- Can’t use FQ-pacing because many flows share a socket
- Need an extra-large send buffer for so many flows
- FQ qdisc creates unfairness between QUIC and TCP

  => Lots of blocked writes, even with a large buffer

Packet Sockets

Packet sockets with shared memory(RX_RING) are still a substantial improvement over just SO_REUSEPORT.

Packet sockets with TX_RING were not a visible win, though it's not clear why.

Using packet sockets for send is much more complex than receive, so the complexity wasn’t worth it.

UDP GSO

UDP GSO achieves performance similar to TCP! (3x faster)

Releases all datagrams from a send call at once

=> Don’t get full CPU savings until 512Mbps
(64KB sends at 1ms pacing granularity)

Ideally the segment could be split and paced to reduce loss

LWN article

Packet Pacing

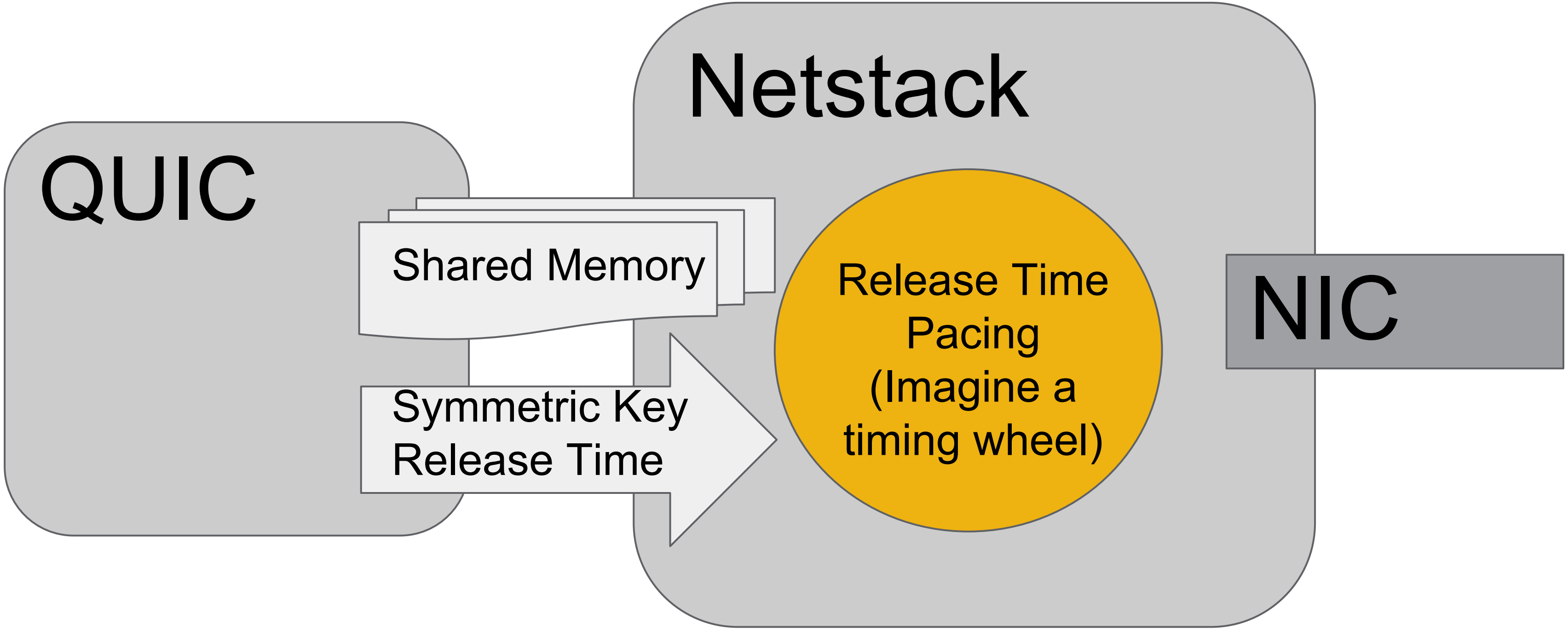
Minimum release time based pacing is ideal

Easy to integrate with congestion control, including BBR, vs rate-based pacing. Allows QUIC to share a socket among flows.

Disabling pacing saves up to 30% CPU in some locations (Carousel, SIGCOMM 2017), but also increases retransmit rates over 50%.

TXTIME patch, FQ in-progress, Chromium pacing offload

The Sending Dream



What remains to be done

UDP receive-side optimizations - UDP GRO?

Crypto offload API and support - Both send and receive

API to allow pacing of multi-datagram UDP sends (ie: GSO)
… solve the tradeoff between packet loss and CPU usage

Thanks!

Willem de Bruijn, Eric Dumazet, Jesus Sanchez-Palencia, all others who’ve improved Linux UDP and pacing in the past few years.

Also, thanks to Tom Herbert for SO_REUSEPORT!

IETF drafts: transport-13, recovery-13, tls-13, http-13
Chromium QUIC Code: cs.chromium.org