HTTP/3 on QUIC on Streams

draft-kazuho-quic-quic-on-streams
draft-kazuho-httpbis-http3-on-streams

Kazuho Oku, Lucas Pardue

QUIC is (becoming) a huge success

- better than TCP / TLS:
  - faster handshake
  - path migration
  - no head-of-line blocking
  - better loss recovery
  - better preservation of privacy
  - future proof (prevents ossification by middleboxes)
- broad adoption:
  - support from major web browsers
  - supported by 29.2% of all websites as of Mar 2024 (source: w3techs)

QUIC is (becoming) a huge success

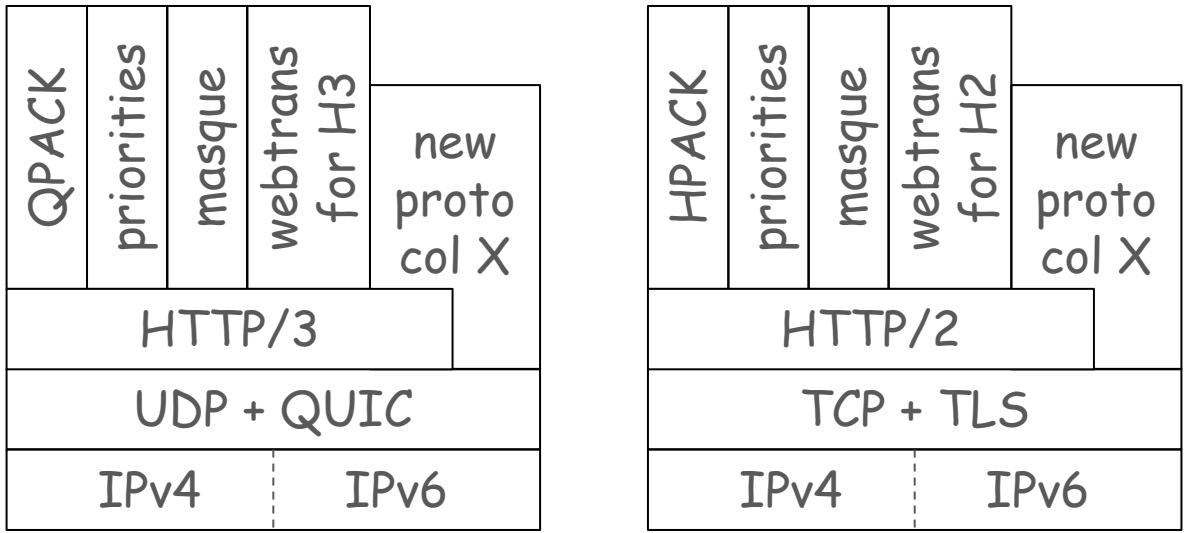
- but TCP continues to be used (at least) as a fallback
  - HTTP/2 (and HTTP/1.1)
  - MoQ
  - WebTransport
  - MASQUE
  - DNS

HTTP/3 projects will continue to want a TCP fallback

- Example: A new MASQUE-based proxying service
- It makes sense to start with QUIC via HTTP/3
  - Streams and unreliable datagrams
- But some users can't use this service because QUIC is not available
  - A fallback option is needed
  - Performance regressions are acceptable
- Adding TCP fallback, via HTTP/2, adds a large new dependency with a set of legacy problems

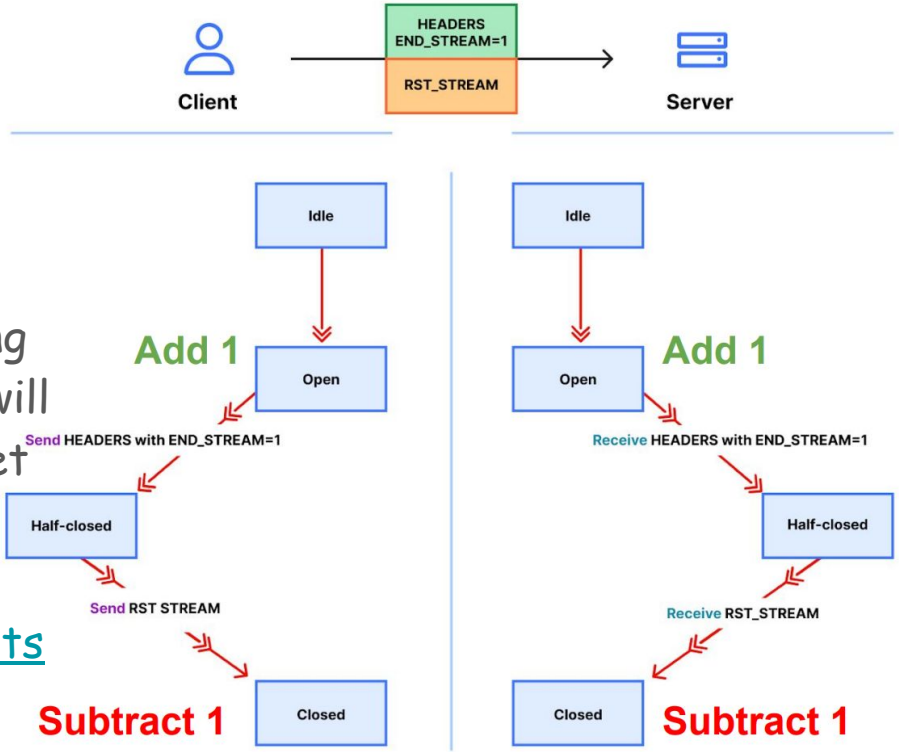
Sad state of application protocols

We now have to develop and maintain two different sets of stacks.



HTTP/2 concurrency & rapid reset

- HTTP/2 stream concurrency is broken
- Default: "unlimited"
- De-facto client default: 100
- Servers picking < 100 break behaving clients
- Despite concurrency limits, misbehaving clients can create / reset streams at will
- HTTP/3 design is immune to rapid reset
- Proposal to port QUIC concurrency to HTTP/2 as an extension: [draft-thomson-httpbis-h2-stream-limits]
  - Not much appetite



What if we could make an HTTP/2.1?

Activation energy for change depends on the value it delivers. What else would we fix about HTTP/2 if we are going to do anything: https://github.com/httpwg/admin/issues/56

1. Better Stream concurrency control
2. Remove server push
3. Remove RFC 7540 priorities (recommend RFC 9218?)
4. Restrict SETTINGS to once per connection at the start
5. Mandate extended CONNECT
6. Use QUIC variable-length integers (expand from 32-bit to 64-bit integers)
7. Grease every extension code point
8. Remove frame flags field - replace this with frame type instead (similar to QUIC and HTTP/3)
9. Remove CONTINUATION frames (larger frame sizes obviate them)
10. Mandate TLS; remove cleartext and upgrade (and hence avoid complications related to cleartext)
11. Remove prior knowledge and the preface stuff. (ALPN is enough)

We are trying to add more stuff to HTTP/2

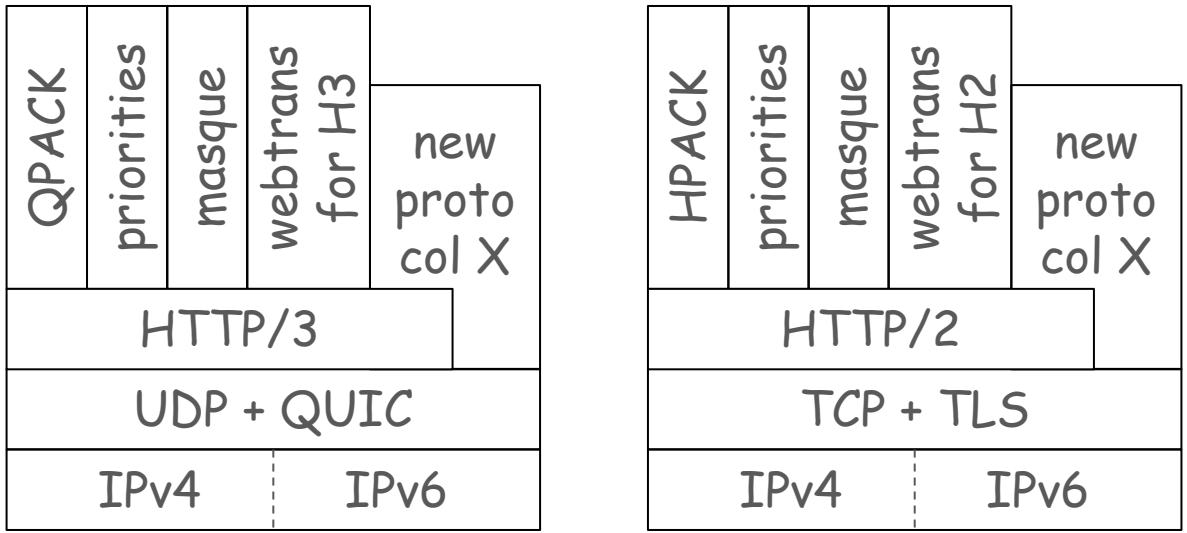
- secondary-server-certs - WG adoption in Mar 2024
- hpack / qpack static table replacement

Everything we add, an intermediary needs to be able to translate, e.g.,

- DATAGRAM frames to capsules on HTTP streams
- WebTransport over H2 to WebTransport over H3
- RFC 7540 priorities to RFC 9218 priorities

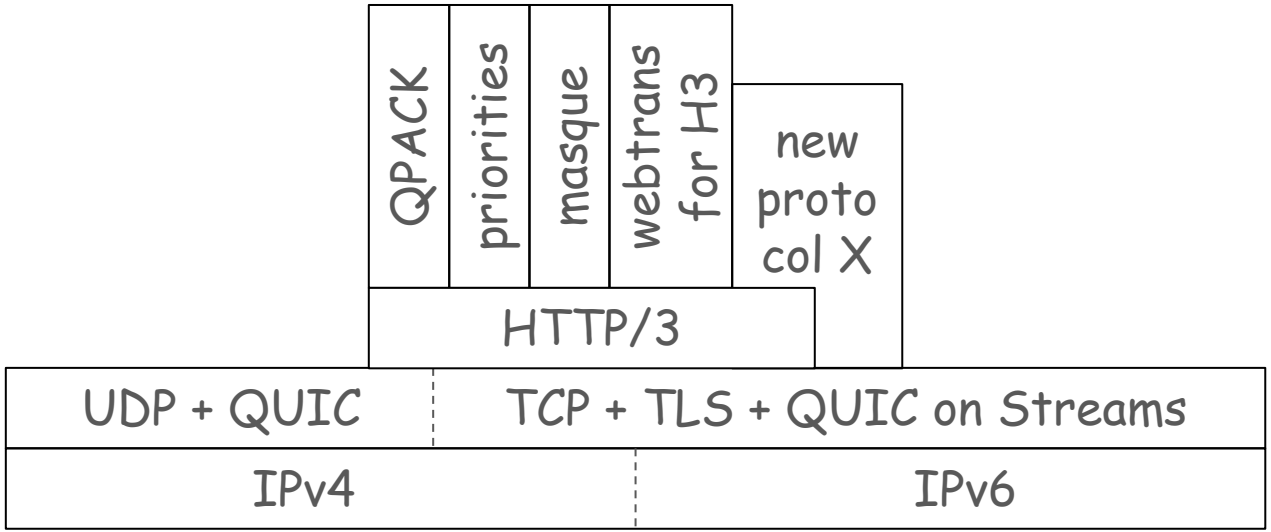
Sad state of application protocols

TCP (UDP) supports both IPv4 and IPv6.
Can QUIC support both UDP and TCP?



QUIC on Streams

Backport the QUIC API contract (i.e., QUIC streams) to TCP. Then, it is possible to run applications written for QUIC and HTTP/3 everywhere.



Wish	HTTP/2	HTTP/2.1	HTTP/3	HTTP/3 on streams
Better Stream concurrency control	No	Yes	Yes	Yes
No RFC 7540 priorities	No	Maybe	Yes	Yes
Restrict SETTINGS to once per connection	No	Maybe	Yes	Yes
Use QUIC variable-length integers	No	Maybe	Yes	Yes
Grease every extension code point	No	Maybe	Yes	Yes
No frame flags field	No	Maybe	Yes	Yes
Remove CONTINUATION frames	No	Maybe	Yes	Yes
Mandate TLS; remove cleartext and upgrade	No	Maybe	Yes	Yes
No prior knowledge / preface stuff. Use ALPN.	No	Maybe	Yes	Yes
No server push	No	Maybe	No	No
Mandate extended CONNECT	No	Maybe	No	Maybe

Our goals and non-goals

- Goals
  - eliminate the need to develop new things on top of two protocols
    - run unmodified HTTP/3 on top of QUIC on Streams
  - eliminate the need to deploy two HTTP versions
    - when you control both sides, this is immediate
  - reuse existing QUIC and HTTP/3 implementations
  - expose underlying transport properties for applications
- Non-goals
  - do not spend time optimizing TCP (e.g., solve HoL blocking) or QUIC frames. QUIC over UDP works in most cases and performs better, so QUIC on Streams is a fallback

Design of draft -00

- new ALPN: <insert bikeshed here>
- send minimal set of QUICv1 frames on top of TCP / TLS
  - only stream, datagram, and associated operations (flow control)
  - no ACK frames, CIDs, etc.
- Transport Parameters are exchanged using 1st frame called QS_TRANSPORT_PARAMETERS
- minimum of maximum frame size is 16KB (matches max. TLS record size)

※Working PoC for quicly created in 1/2 day. Took another 1/2 day to integrate that into H2O to run H3 client / server over QUIC on Streams.

Concerns

- Are application layer protocols going to suffer if they're assuming UDP and get TCP?
  - Performance versus HTTP/2
- Yet another way to do HTTP, one more protocol variant
  - Introducing a whole new set of [security] problems
- Possibility that networks will no longer feel the pressure to make QUIC/UDP work

Our goals and non-goals

- Goals
  - eliminate the need to develop new things on top of two protocols
    - run unmodified HTTP/3 on top of QUIC on Streams
  - eliminate the need to deploy two HTTP versions
    - when you control both sides, this is immediate
  - expose underlying transport properties for applications
  - reuse existing QUIC and HTTP/3 implementations
- Non-goals
  - do not spend time optimizing TCP (e.g., solve HoL blocking) or QUIC frames. QUIC over UDP works in most cases and performs better, so QUIC on Streams is a fallback