# HTTP/3

How did we get here?

# GET /

```
GET / HTTP/1.0
Accept: text/html
Accept-Language: en
Accept-Encoding: x-gzip
User-Agent: libwww/2.5
Referer: http://www.example.com/
```

```
GET / HTTP/1.0
Accept: text/html
Accept-Language: en
Accept-Encoding: x-gzip
User-Agent: libwww/2.5
Referer: http://www.example.com/
Host: www.example.net
Connection: keep-alive
```

✅ Connection reuse

✅ Hosting

```
GET / HTTP/1.1
Accept: text/html
Accept-Language: en
Accept-Encoding: x-gzip
User-Agent: libwww/2.5
Referer: http://www.example.com/
Host: www.example.net
Transfer-Encoding: chunked
```

✅ Chunked Transfer Codings

⛔ Compression Transfer Codings

⛔ Pipelining

“The protocol is to deliver multiplexed bidirectional reliable ordered message streams over a bidirectional reliable ordered byte stream protocol (such as TCP). Message streams may be initiated by either side, once the underlying byte stream connection is established.

The length of a message is unrestricted... and the payload of a message is also unrestricted; such a message can be used directly, e.g., as a request or a response in an application-level request/response protocol.

Within each message stream, the messages are delivered reliably and in order (as are bytes in TCP).

Each message may be passed as a series of chunks, so that the multiplexing does not introduce unnecessary synchronization between streams.

The protocol will be layered on top of bidirectional reliable ordered byte stream protocols (such as, but not limited to, TCP), and multiplex many message streams over a single byte stream connection.

It should be possible for there to be multiple message chunks in one IP packet.”

# HTTP-NG

# SPDY → HTTP/2

✅ Binary framing

✅ Multiplexing

✅ Header Compression

⚠️ Prioritisation

⛔ Server Push

# What have we learned?

- Incremental changes that exploit layering tend to work
- Changing the fundamental model of the protocol, or implementation assumptions, often doesn't work well
- It's very tempting to over-engineer things
- Implementation mindshare is key, but resources are finite
- An active community is invaluable

QUIC
Internet-Draft
Intended status: Standards Track
Expires: August 2, 2019

M. Bishop, Ed.
Akamai

January 29, 2019

# Hypertext Transfer Protocol Version 3 (HTTP/3)

draft-ietf-quic-http-latest

**Abstract**

The QUIC transport protocol has several features that are desirable in a transport for HTTP, such as stream multiplexing, per-stream flow control, and low-latency connection establishment. This document describes a mapping of HTTP semantics over QUIC. This document also identifies HTTP/2 features that are subsumed by QUIC, and describes how HTTP/2 extensions can be ported to HTTP/3.

HTTP/2 built a stream layer because we needed multiplexing on top of TCP.

HTTP/3 gets multiplexing from QUIC.

# HTTP/3

# Has One Job

# Transport Head-of-Line Blocking

Inter-stream ordering is not guaranteed

# 1. SETTINGS

- Frames arriving after SETTINGS may have been sent before it
- ... so it's hard to reason about them
- SETTINGS sent once; cannot change
- Many settings superseded by QUIC transport parameters

# 2. Prioritisation

- HTTP/2 prioritisation relies upon changes to the dependency tree being applied by both peers in the same order
- HTTP/3 addresses this by sending all priority changes on one control stream
- Exclusive prioritisation is not possible

# 3. Header Compression

- HPACK is effectively a stream of commands:
  - Use this literal value
  - Use the value indexed at #5
  - Insert this value into index #5 and use it

# 3. Header Compression

- HPACK is effectively a stream of commands:
  - Use this literal value
  - Insert this value into index #5 and use it
  - Use the value indexed at #5

QUIC
Internet-Draft
Intended status: Standards Track
Expires: August 2, 2019

C. Krasic
Netflix
M. Bishop
Akamai Technologies
A. Frindell, Ed.
Facebook
January 29, 2019

# QPACK: Header Compression for HTTP over QUIC

draft-ietf-quic-qpack-latest

**Abstract**

This specification defines QPACK, a compression format for efficiently representing HTTP header fields, to be used in HTTP/3. This is a variation of HPACK header compression that seeks to reduce head-of-line blocking.

- Dynamic table is updated with a special, one-way stream
- Encoder keeps state about references until headers are ack'd
  - References can block insertion; fall back to literals
- Insert count used to track what state is required to decompress

“Personally, I give us one chance in three.”

*- Captain Ramius, The Hunt for Red October*

# What's Next for HTTP?

- HTTP/1.0: extensibility (headers)
- HTTP/1.1: utilisation of transport (multi-homing, conn reuse)
- HTTP-NG: utilisation of transport (HTTP HOL blocking)
- HTTP/2: utilisation of transport (HTTP HOL blocking)
- HTTP/3: utilisation of transport (TCP HOL blocking)

- Connection Coalescing
- Structured Headers
- Semantic Evolution
- CDN Standardisation
- ...

# HTTP/4?

# HTTP.