



# QUIC PROTOCOL

HTTP/3

罗龙君(luolongjuna@gmail.com)

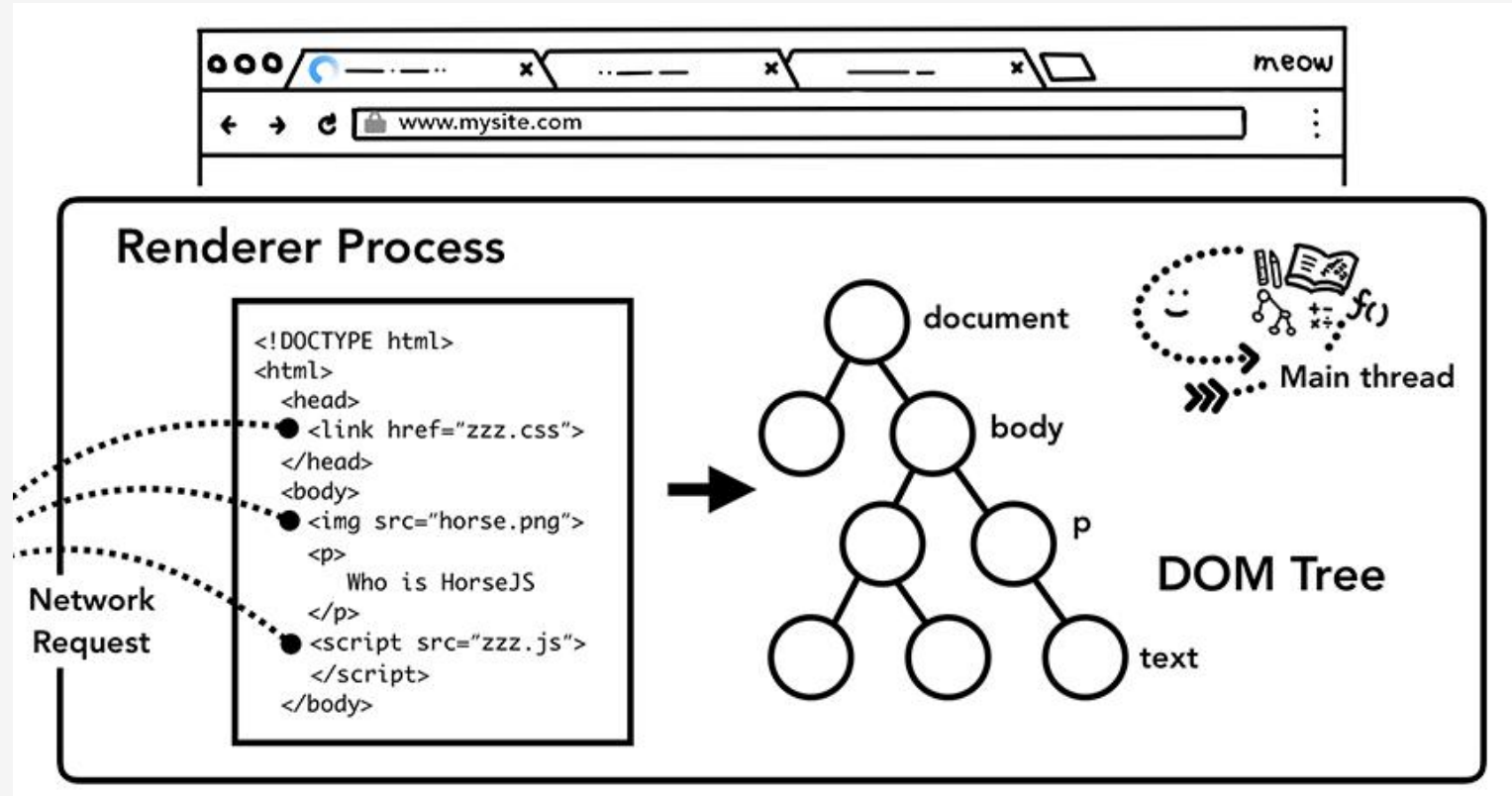
# Contents of a website

The image shows a screenshot of the Tsinghua University website (tsinghua.edu.cn) with the Chrome DevTools Network tab open. The website features a navigation menu with categories like '清华新闻', '学校概况', '院系设置', '教育教学', '科学研究', '招生就业', '合作交流', and '走进清华'. A sidebar on the left lists '本科生招生', '研究生招生', '国际学生招生', '在线教育', and '终身教育'. The main content area displays a large image of a person in a dark, futuristic setting. The DevTools Network tab shows a list of 120 requests, including various CSS, JavaScript, and image files. The 'Filter' dropdown is set to 'All', and the 'Waterfall' view is active, showing the timing of each request.

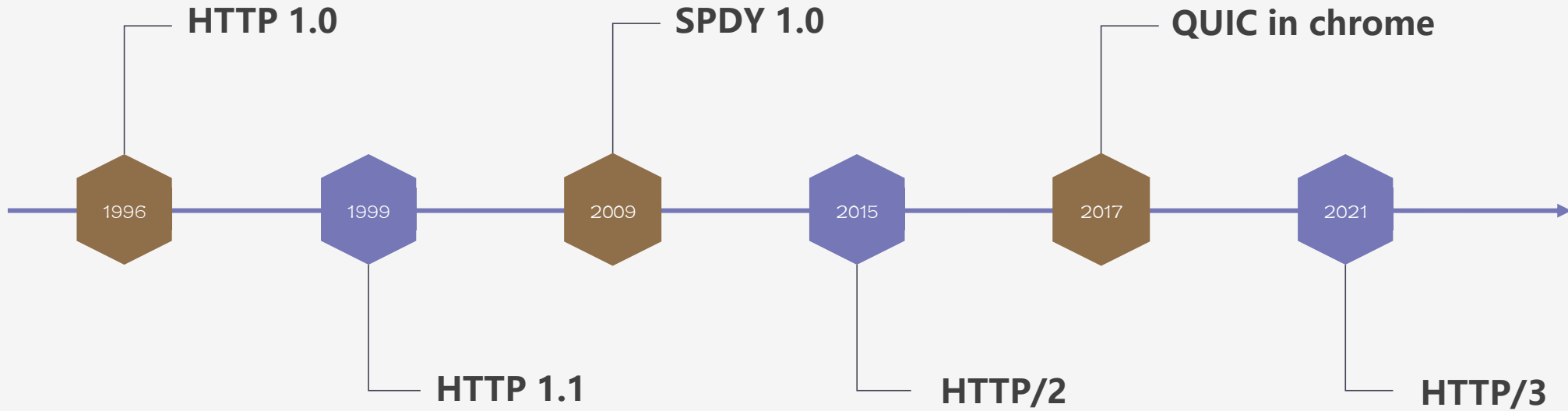
Name	Stat...	Type	Initiator	Size	Time	Waterfall
www.tsinghua.edu...	200	doc...	Other	31.6...	65 ...	
index.css	200	styl...	(index)	13.8...	55 ...	
jquery.min.js	200	script	(index)	50.4...	147 ...	
set.js	200	script	(index)	1.2 ...	28 ...	
_sitegray_d.css	200	styl...	(index)	168 B	29 ...	
_sitegray.js	200	script	(index)	223 B	79 ...	
index.vsb.css	200	styl...	(index)	837 B	79 ...	
counter.js	200	script	(index)	1.0 ...	27 ...	
dynclicks.js	200	script	(index)	1.1 ...	28 ...	
openlink.js	200	script	(index)	388 B	27 ...	
base64.js	200	script	(index)	1.3 ...	38 ...	
formfunc.js	200	script	(index)	576 B	38 ...	
centerCutimg.js	200	script	(index)	1.2 ...	77 ...	
ajax.js	200	script	(index)	1.7 ...	77 ...	
mp4video.js	200	script	(index)	1.6 ...	65 ...	
engine.js	200	script	(index)	20.2...	44 ...	
util.js	200	script	(index)	17.8...	57 ...	
PageCounterDWR.js	200	script	(index)	403 B	44 ...	
js.js	200	script	(index)	2.3 ...	72 ...	
swiper.min.css	200	styl...	(index)	4.1 ...	72 ...	
swiper.jquery.min.js	200	script	(index)	34.4...	114 ...	
TweenMax.min.js	200	script	(index)	61.0...	689 ...	
EB3E9119838E80...	200	jpeg	(index)	437 ...	11.1...	
FD94488AE39103...	200	jpeg	(index)	145 ...	307 ...	
0B7A68914CCA79...	200	jpeg	(index)	413 ...	10.4...	
5346AFF0485D62...	200	jpeg	(index)	145 ...	492 ...	
D7ED83C8ABFE5...	200	jpeg	(index)	350 ...	15.1...	
E686ACCEA33698...	200	jpeg	(index)	127 ...	7.01 s	

120 requests | 19.6 MB transferred | 19.9 MB resources | Finish: 49.14 s | DOMContentLoaded: 974

# Contents of a website



# Timeline of HTTP protocol



# HTTP 1.0 VS HTTP 1.1

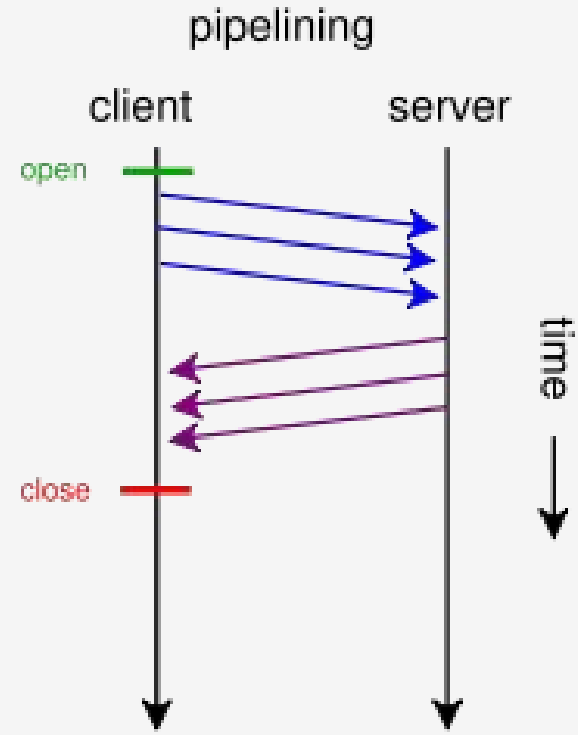
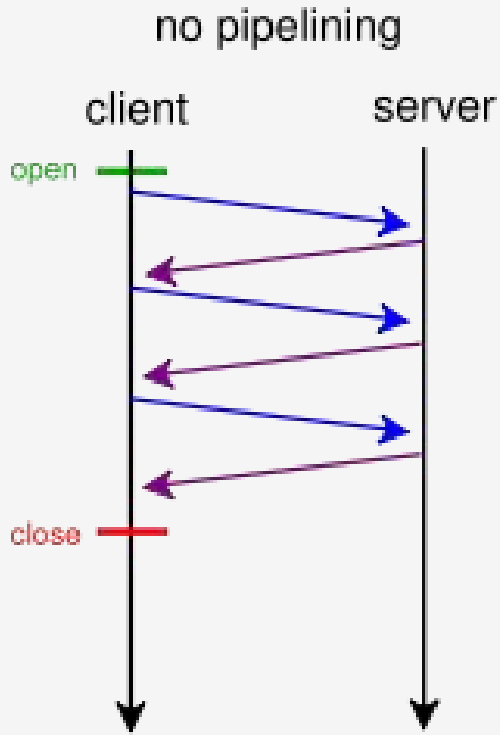
## HTTP 1.0

- Single request/reponse per connection
- Host header optional
- Limited support for caching
- Simple status code

## HTTP 1.1

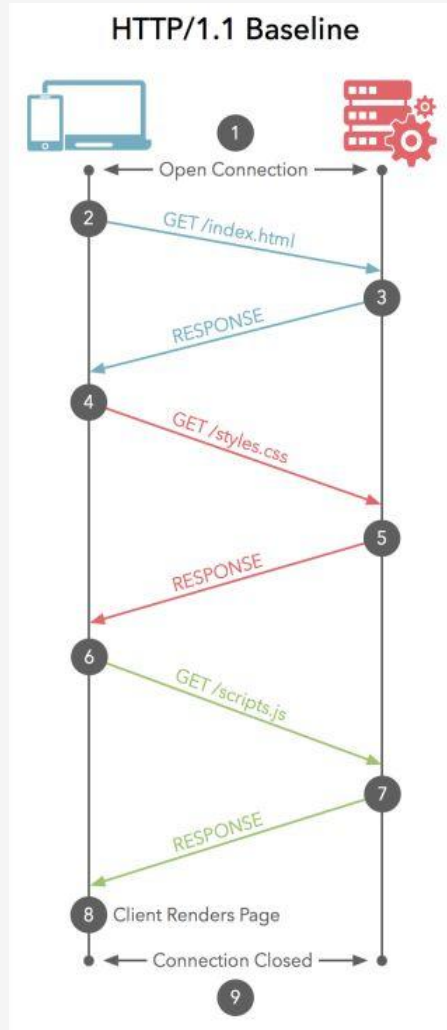
- Persistent and pipelined connection
- Need host filed
- Caching support
- HTTP status code
- Chunked transfers / byte-range transfers
- Compression/decompression
- More .....

# HTTP 1.1 pipeline

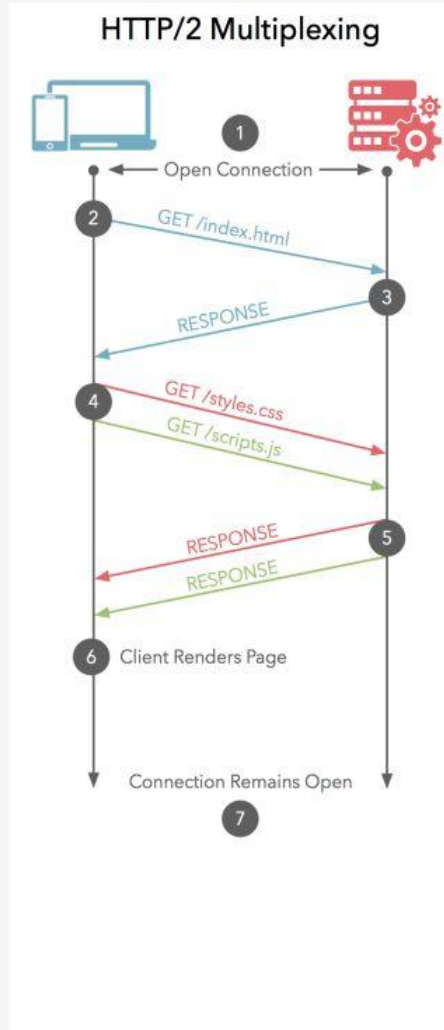


- Solve HOL problem partly (FIFO)
- Hard to use (Idempotent request)
- Not be used in default

# SPDY and HTTP/2

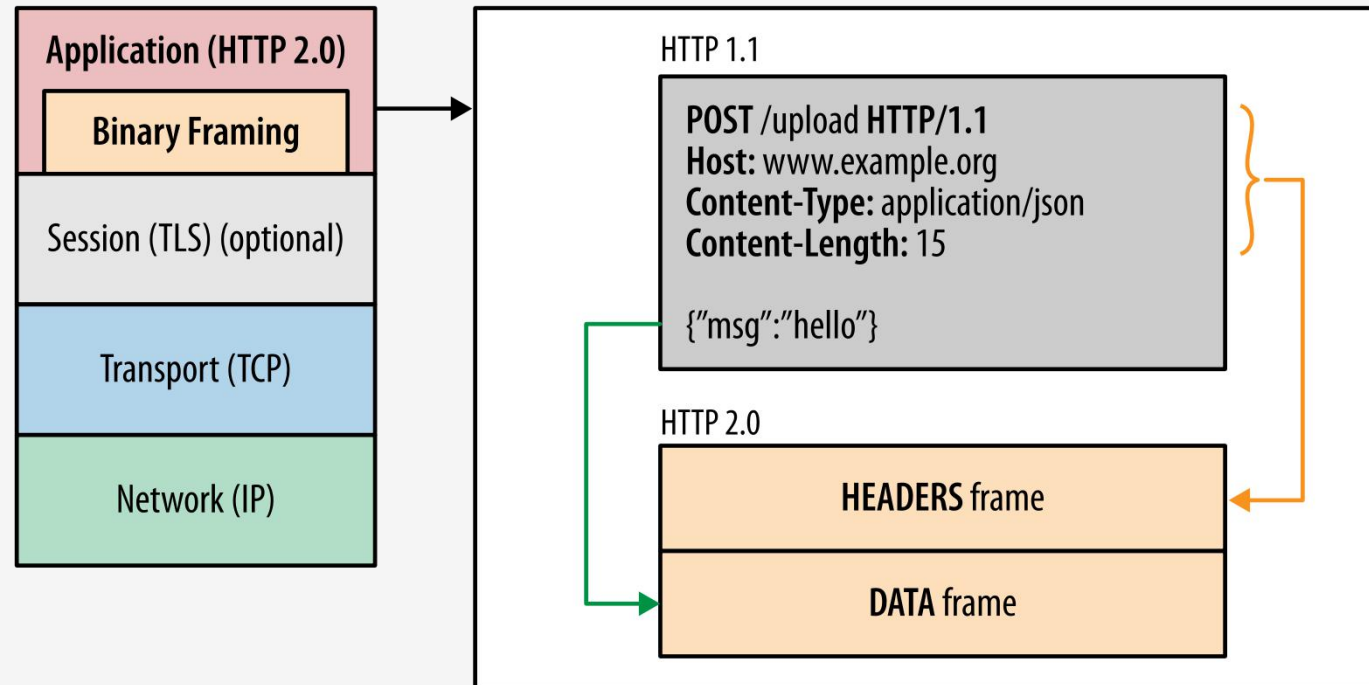


Time



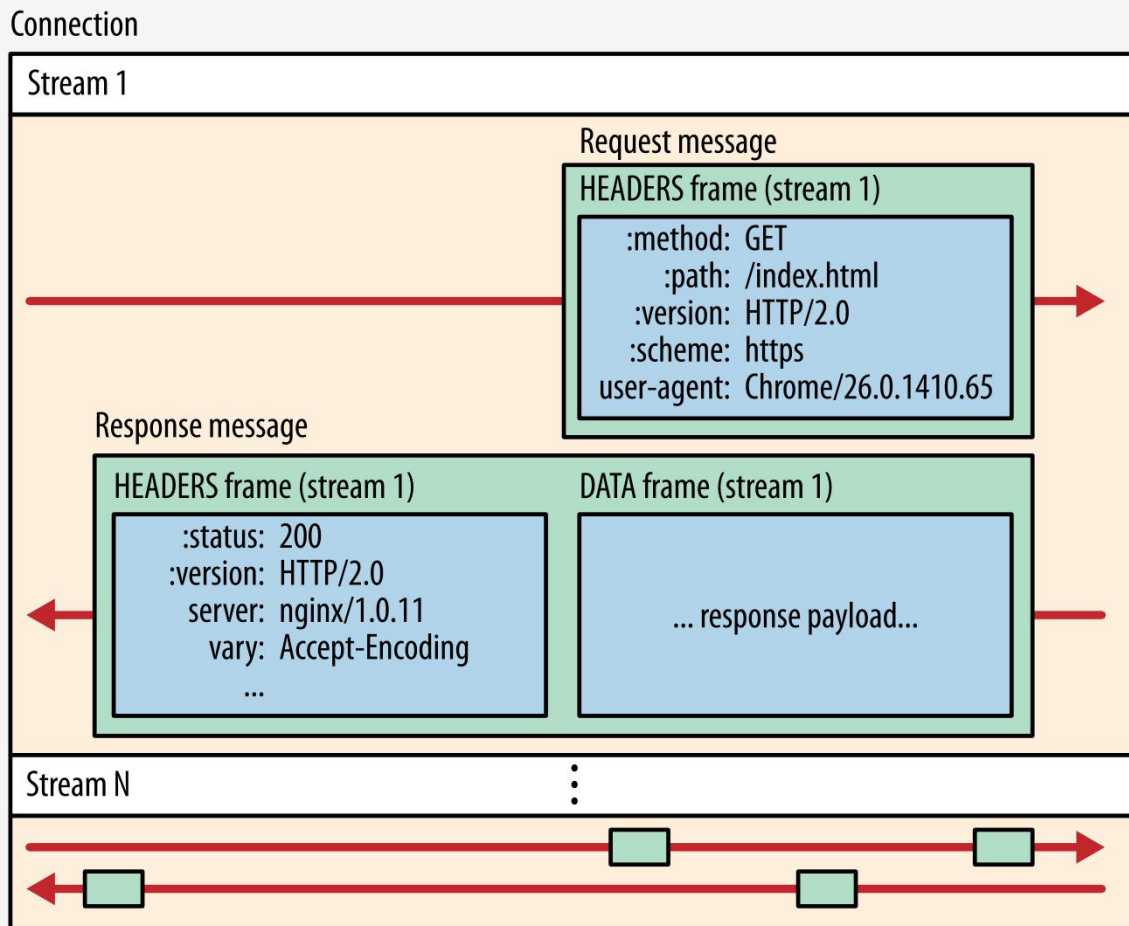
- Multiplexing
- Introduce priority
- flow control
- Server push
- Hpack

# A frames protocol





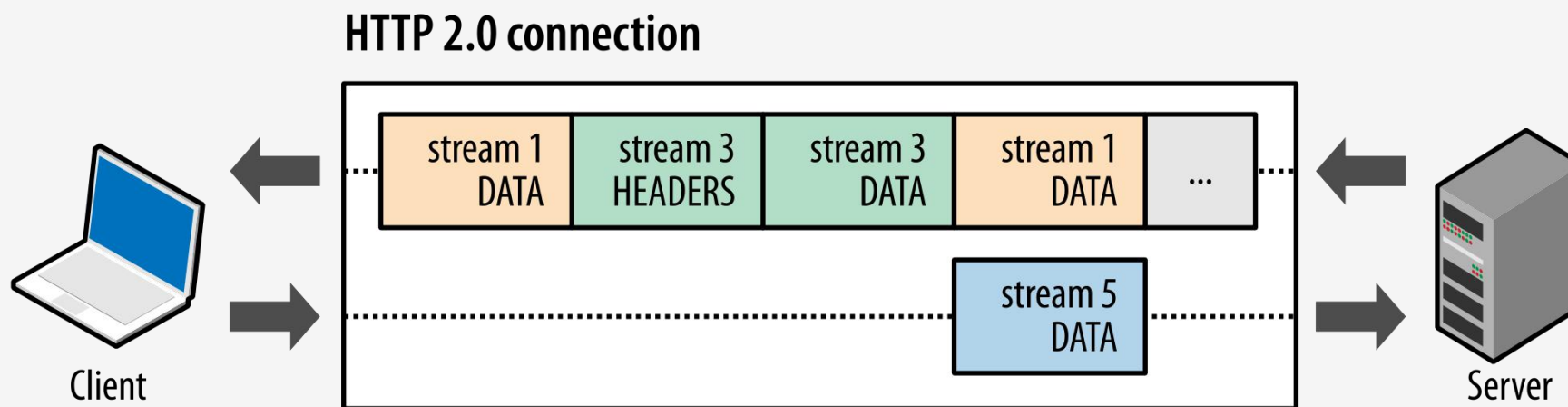
# A frames protocol



- Stream
  - 已建立的连接内的双向字节流，可以承载一条或多条消息。
- Message
  - 每个数据流都有一个唯一的标识符和可选的优先级信息，用于承载双向消息。
- Frame
  - 帧是最小的通信单位，承载着特定类型的数据，例如 HTTP 标头、消息负载等等。来自不同数据流的帧可以交错发送，然后再根据每个帧头的数据流标识符重新组装

# Multiplexing

- 并行交错地发送多个请求，请求之间互不影响。
- 并行交错地发送多个响应，响应之间互不干扰。
- 使用一个连接并行发送多个请求和响应。
- 消除不必要的延迟和提高现有网络容量的利用率，从而减少页面加载时间。



# Multiplexing

https://http2.akamai.com/demo

The screenshot shows a Wireshark capture of HTTP/2 traffic. The main pane displays a list of 28 requests, all of which are HEADERS for GET requests to various tile images. The requests are multiplexed, as they all originate from the same source IP (192.168.5.105) and are destined for the same server IP (104.127.205.211). The bottom pane shows a hex dump of a TLS frame, which is a decrypted TLS frame (48 bytes) containing a Hypertext Transfer Protocol 2: Protocol frame (124 bytes).

No.	Time	Source	Destination	Protocol	Length	Info
5439	41.387156	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[43]: GET /demo/tile-62.png
5440	41.387185	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[45]: GET /demo/tile-65.png
5441	41.387216	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[47]: GET /demo/tile-34.png
5442	41.387247	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[49]: GET /demo/tile-55.png
5443	41.387277	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[51]: GET /demo/tile-28.png
5444	41.387304	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[53]: GET /demo/tile-17.png
5445	41.387332	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[55]: GET /demo/tile-60.png
5446	41.387365	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[57]: GET /demo/tile-10.png
5447	41.387396	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[59]: GET /demo/tile-80.png
5448	41.387423	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[61]: GET /demo/tile-47.png
5449	41.388713	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[63]: GET /demo/tile-36.png
5450	41.388751	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[65]: GET /demo/tile-68.png
5451	41.388780	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[67]: GET /demo/tile-58.png
5452	41.388813	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[69]: GET /demo/tile-52.png
5453	41.388841	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[71]: GET /demo/tile-32.png
5454	41.388870	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[73]: GET /demo/tile-16.png
5455	41.388906	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[75]: GET /demo/tile-26.png
5456	41.388935	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[77]: GET /demo/tile-78.png
5457	41.388962	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[79]: GET /demo/tile-61.png
5458	41.388996	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[81]: GET /demo/tile-53.png
5459	41.389025	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[83]: GET /demo/tile-70.png
5460	41.389052	192.168.5.105	104.127.205.211	HTTP2	122	HEADERS[85]: GET /demo/tile-14.png
5461	41.389086	192.168.5.105	104.127.205.211	HTTP2	123	HEADERS[87]: GET /demo/tile-49.png

```
0000 f0 2f 74 b3 69 97 54 a7 03 3b 71 3a 08 00 45 00  ./t.i.T. .;q:...E.
0010 00 6e 89 f0 40 00 2f 06 c5 35 68 7f cd d3 c0 a8  n.@./- 5h....
0020 05 69 01 bb f3 84 0d ed 15 d5 7d 04 9a e8 50 18  .i.....}...P.
0030 01 f5 3f 2b 00 00 17 03 03 00 41 08 11 0d 0d a9  ..?+.....A.....
0040 7f 4f 96 1a 6b 2b a9 83 72 e5 4d af d8 ad 18 f0  .O.k+...r.M.....
```

Frame (124 bytes) | Decrypted TLS (48 bytes)

HyperText Transfer Protocol 2: Protocol | 分组: 86392 · 已显示: 194 (0.2%) | 配置: Default

# Multiplexing

<https://http2.akamai.com/demo>

Wireshark · 分组 3798 · 以太网 5

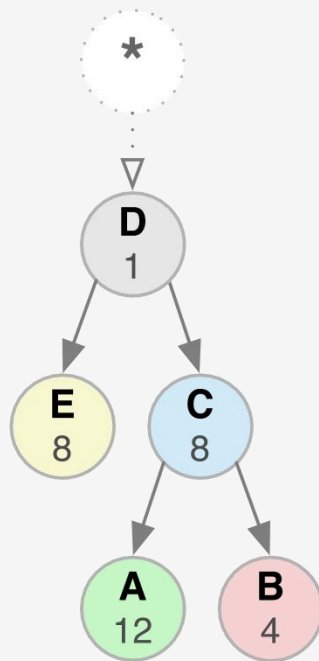
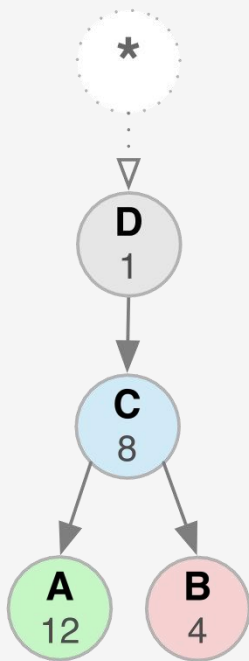
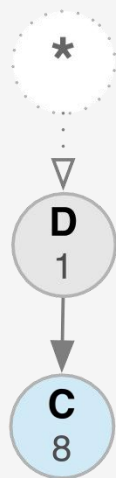
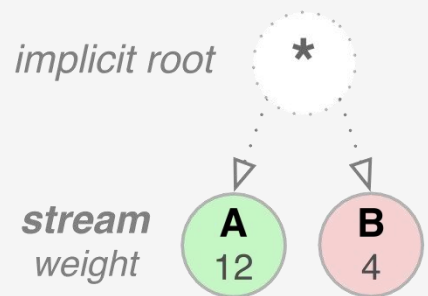
> Frame 3798: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface \Device\NPF\_{79080F3B-63BE-4C8B-9AA9-B08D5948853E}, id 0  
> Ethernet II, Src: Tp-LinkT\_3b:71:3a (54:a7:03:3b:71:3a), Dst: ASUSTekC\_b3:69:97 (f0:2f:74:b3:69:97)  
> Internet Protocol Version 4, Src: 104.127.205.211, Dst: 192.168.5.105  
> Transmission Control Protocol, Src Port: 443, Dst Port: 62340, Seq: 552, Ack: 1339, Len: 70  
> Transport Layer Security  
▼ HyperText Transfer Protocol 2  
 > Stream: SETTINGS, Stream ID: 0, Length 30  
 > Stream: SETTINGS, Stream ID: 0, Length 0

```
0000 f0 2f 74 b3 69 97 54 a7 03 3b 71 3a 08 00 45 00  ./t-i-T.;q:-E
0010 00 6e 89 f0 40 00 2f 06 c5 35 68 7f cd d3 c0 a8  .n-@./-5h-----
0020 05 69 01 bb f3 84 0d ed 15 d5 7d 04 9a e8 50 18  .i.....}...P-
0030 01 f5 3f 2b 00 00 17 03 03 00 41 08 11 0d 0d a9  ..?+.....A.....
0040 7f 4f 96 1a 6b 2b a9 83 72 e5 4d af d8 ad 18 f0  .O..k+...n-M.....
0050 c9 1f f1 d0 56 90 2a e1 3c a7 69 1d 5d 25 2e 81  ....V.*.<-i.]%..
0060 6d bb 67 70 65 22 76 dd b8 ab 8d 20 f5 8c 73 64  m-gpe"v...-sd
0070 7c d1 ec 1a 69 08 d5 a2 4b ba 29 fd          ]...i...K.)
```

Frame (124 bytes)    Decrypted TLS (48 bytes)

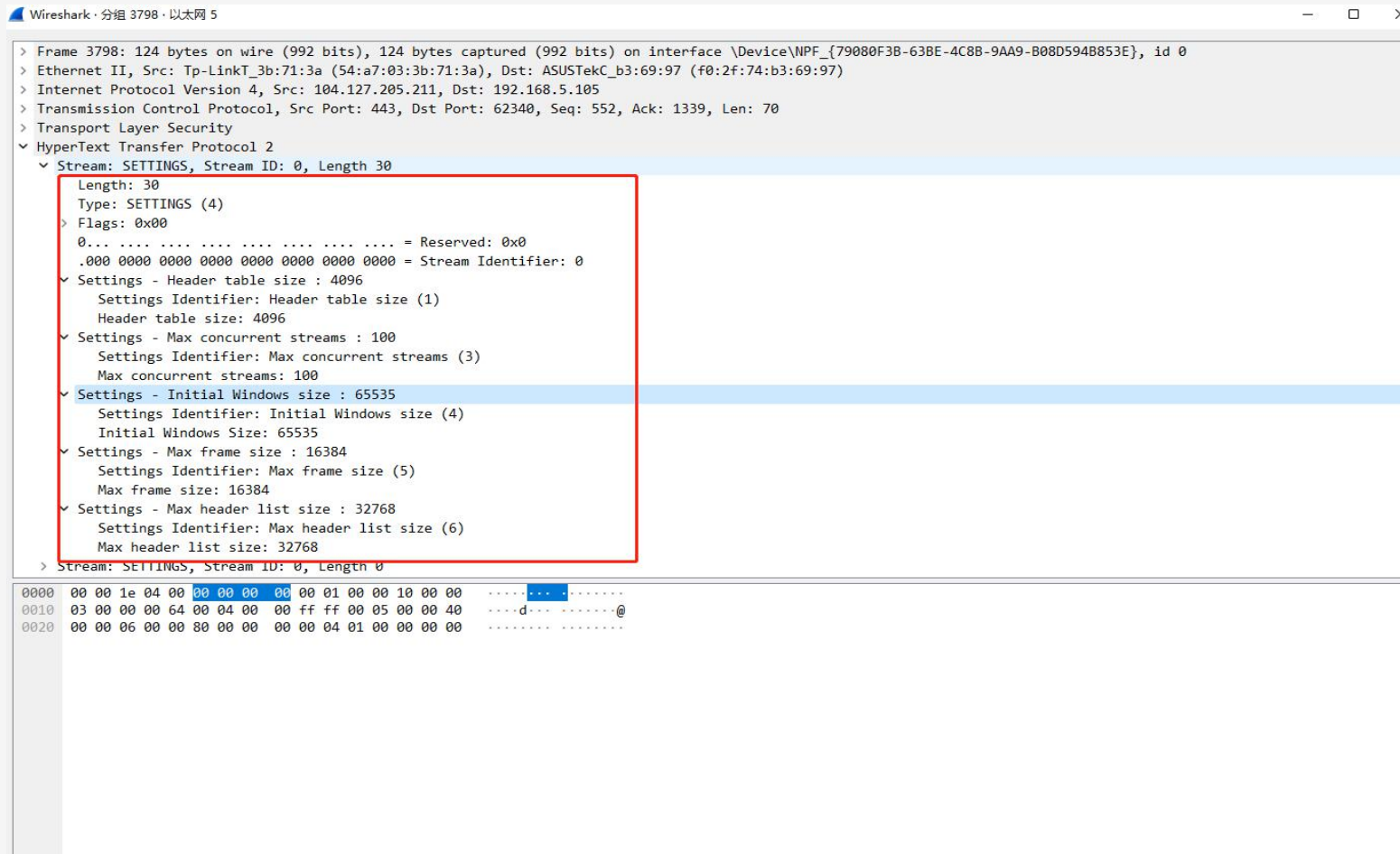
Close    Help

# Introduce priority



- 每个流都分配一个1~256优先级
- 流之间存在依赖关系

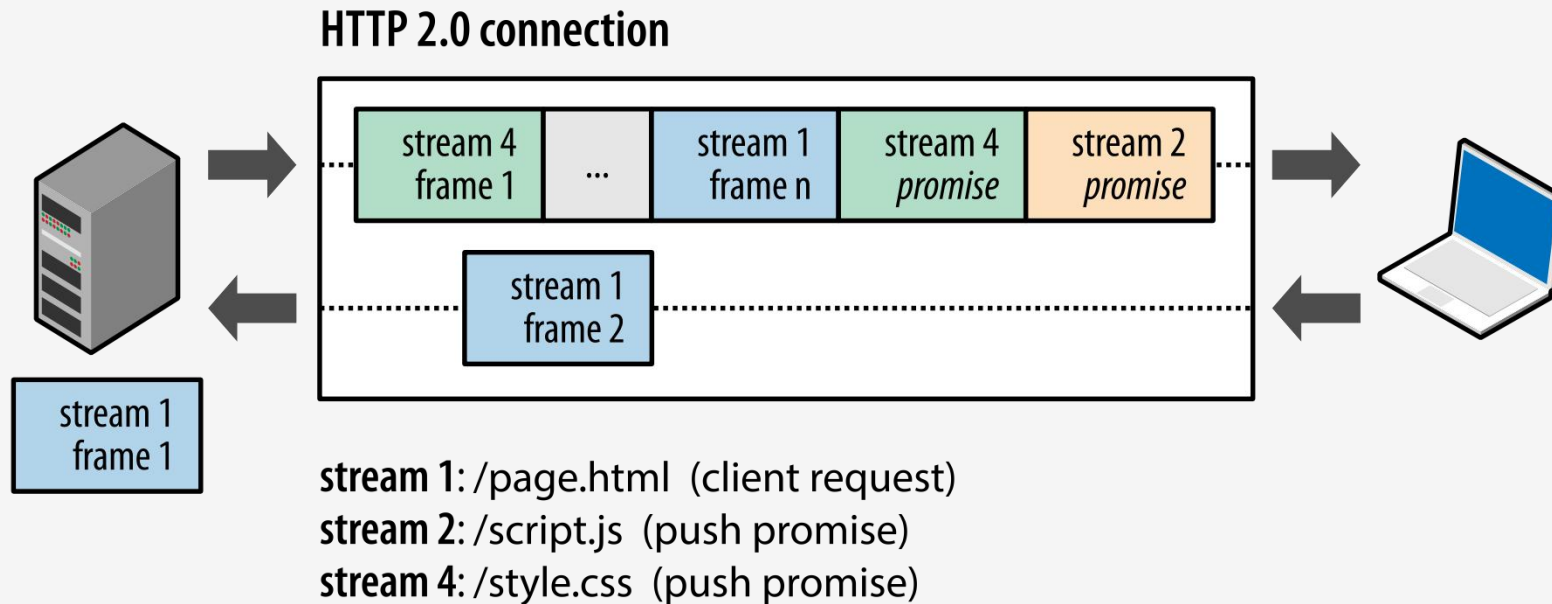
# flow control



```
Wireshark · 分组 3798 · 以太网 5
> Frame 3798: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface \Device\NPF_{79080F3B-63BE-4C8B-9AA9-B08D594B853E}, id 0
> Ethernet II, Src: Tp-LinkT_3b:71:3a (54:a7:03:3b:71:3a), Dst: ASUSTekC_b3:69:97 (f0:2f:74:b3:69:97)
> Internet Protocol Version 4, Src: 104.127.205.211, Dst: 192.168.5.105
> Transmission Control Protocol, Src Port: 443, Dst Port: 62340, Seq: 552, Ack: 1339, Len: 70
> Transport Layer Security
> HyperText Transfer Protocol 2
  > Stream: SETTINGS, Stream ID: 0, Length 30
    Length: 30
    Type: SETTINGS (4)
    Flags: 0x00
    0... .. = Reserved: 0x0
    .000 0000 0000 0000 0000 0000 0000 0000 = Stream Identifier: 0
  > Settings - Header table size : 4096
    Settings Identifier: Header table size (1)
    Header table size: 4096
  > Settings - Max concurrent streams : 100
    Settings Identifier: Max concurrent streams (3)
    Max concurrent streams: 100
  > Settings - Initial Windows size : 65535
    Settings Identifier: Initial Windows size (4)
    Initial Windows Size: 65535
  > Settings - Max frame size : 16384
    Settings Identifier: Max frame size (5)
    Max frame size: 16384
  > Settings - Max header list size : 32768
    Settings Identifier: Max header list size (6)
    Max header list size: 32768
  > Stream: SETTINGS, Stream ID: 0, Length 0
0000 00 00 1e 04 00 00 00 00 00 01 00 00 10 00 00 .....
0010 03 00 00 00 64 00 04 00 00 ff ff 00 05 00 00 40 .....d.....@
0020 00 00 06 00 00 80 00 00 00 00 04 01 00 00 00 00 .....
```

- 流控制具有方向性
- 仅针对data frame
- 可针对连接限制(stream id = 0)
- 可针对流限制(stream id > 0)

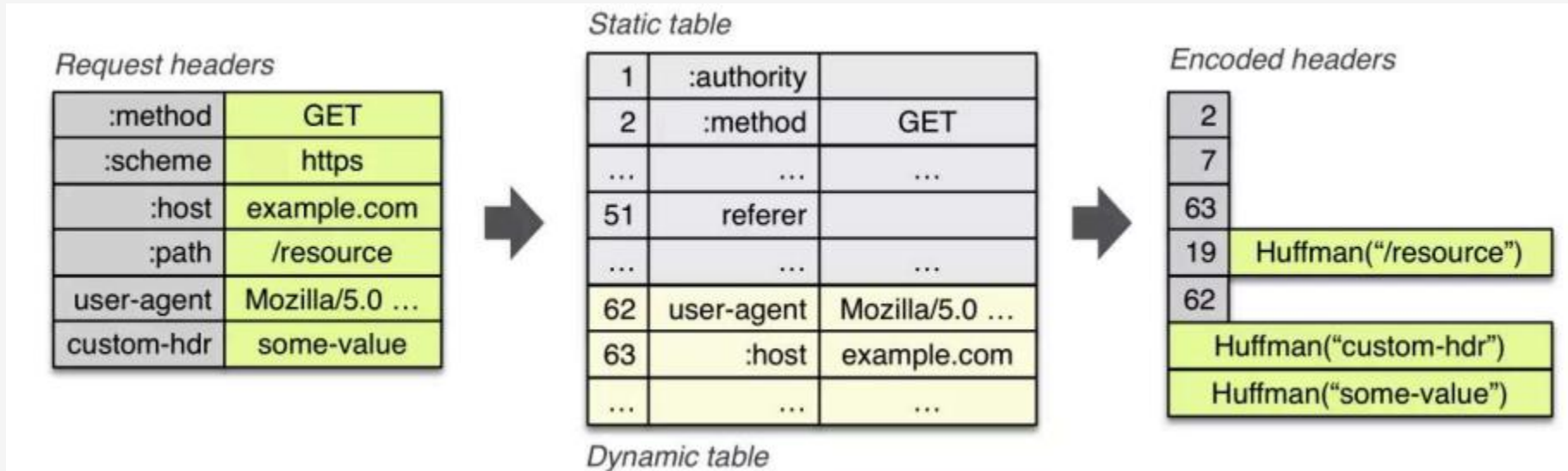
# server push



- 预知client的资源需求
- 由客户端缓存
- 不同页面间进行复用
- 由服务器设定优先级
- 客户端可以拒绝
- 帧类型为PUSH\_PROMISE, 新建stream



# Hpack



- RFC中规定了静态的表(最常用的)
- 传输时, 维护了一个动态表
- 如果命中, 仅需要传递索引号
- 使用HPACK算法压缩 (RFC7541)



# HTTP2 问题 (TCP ?)

## HTTP/1.1

Latency: 118ms  
Load time: 7.17s



Demo concept inspired by Golang's Gophertiles

## HTTP/2

Latency: 124ms  
Load time: 17.42s



[Return to Akamai's HTTP/2 page](#)

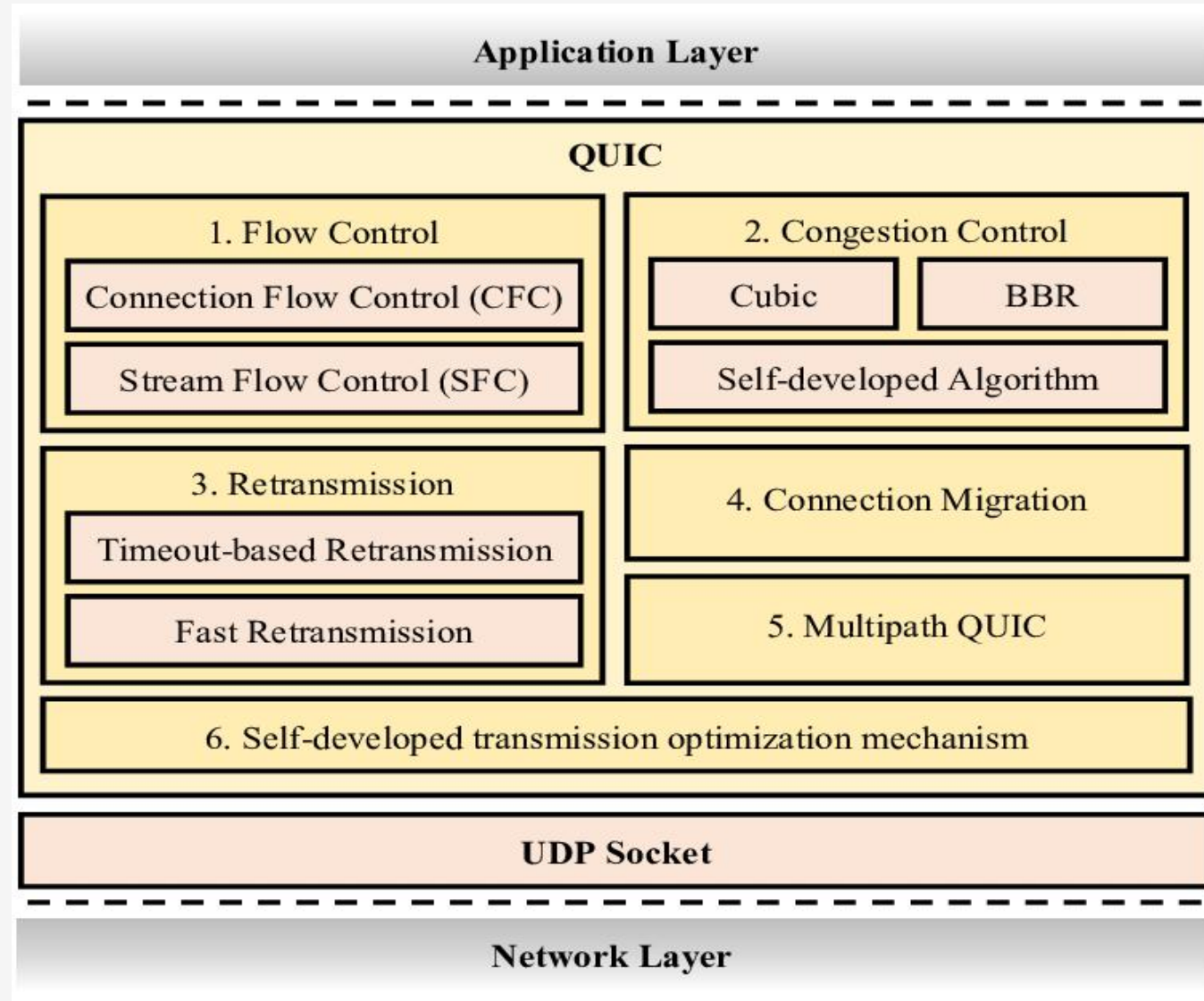
[Blog explaining how this demo works](#)

- 频繁丢包的慢网环境下, http2 差于 http1.1
- TCP 的丢包重传带来了 HOL 问题
- TCP 的拥塞控制导致整个链接传输窗口变小
- TCP 需要三次握手, TLS 也需要握手
- 移动网络变化 IP 地址
- SCTP/RTP ....

# QUIC



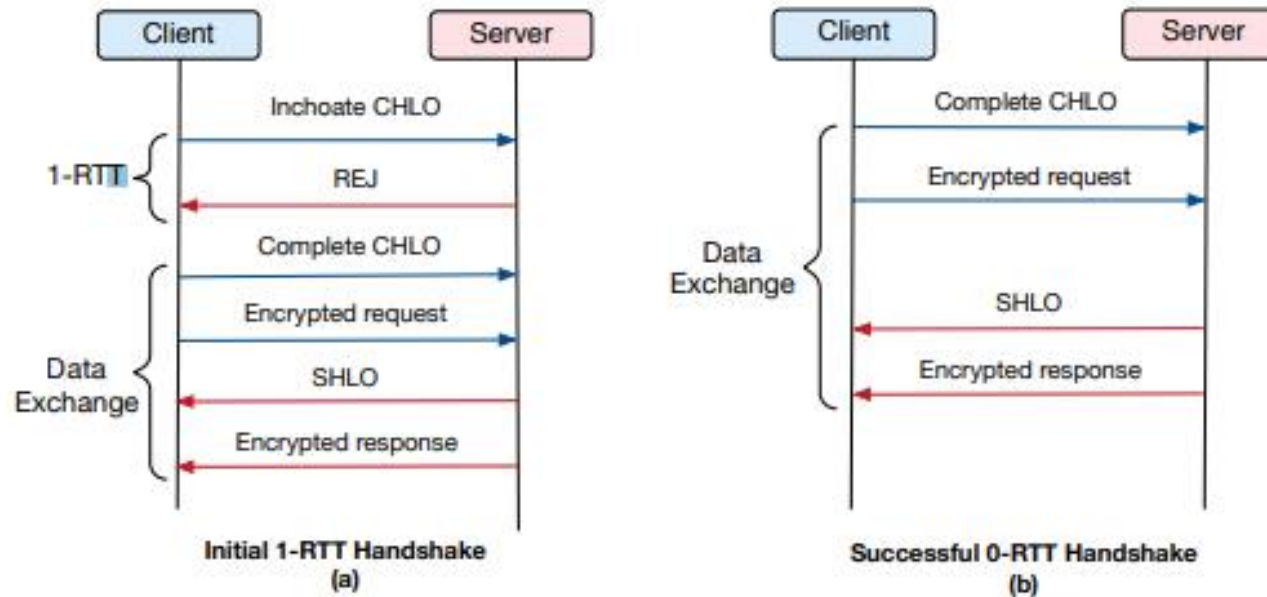
# loss detection / congestion control (RFC9002)



# 1 RTT (RFC 9001)

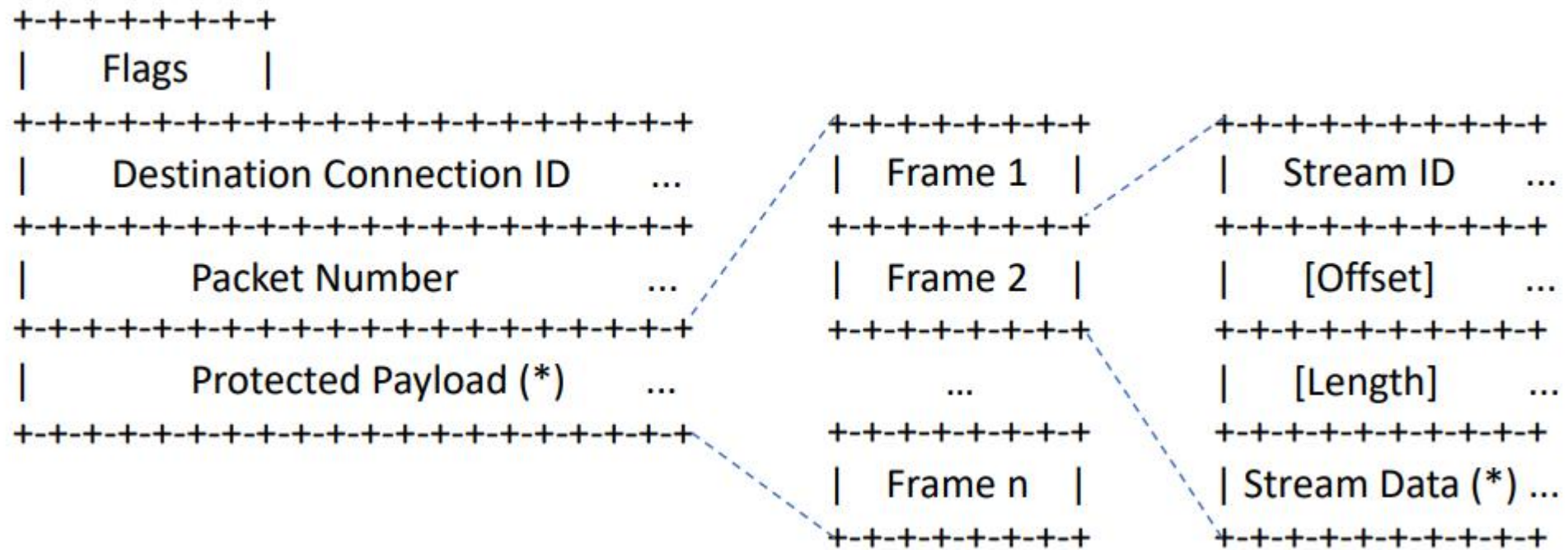


# 0 RTT (RFC 9001)

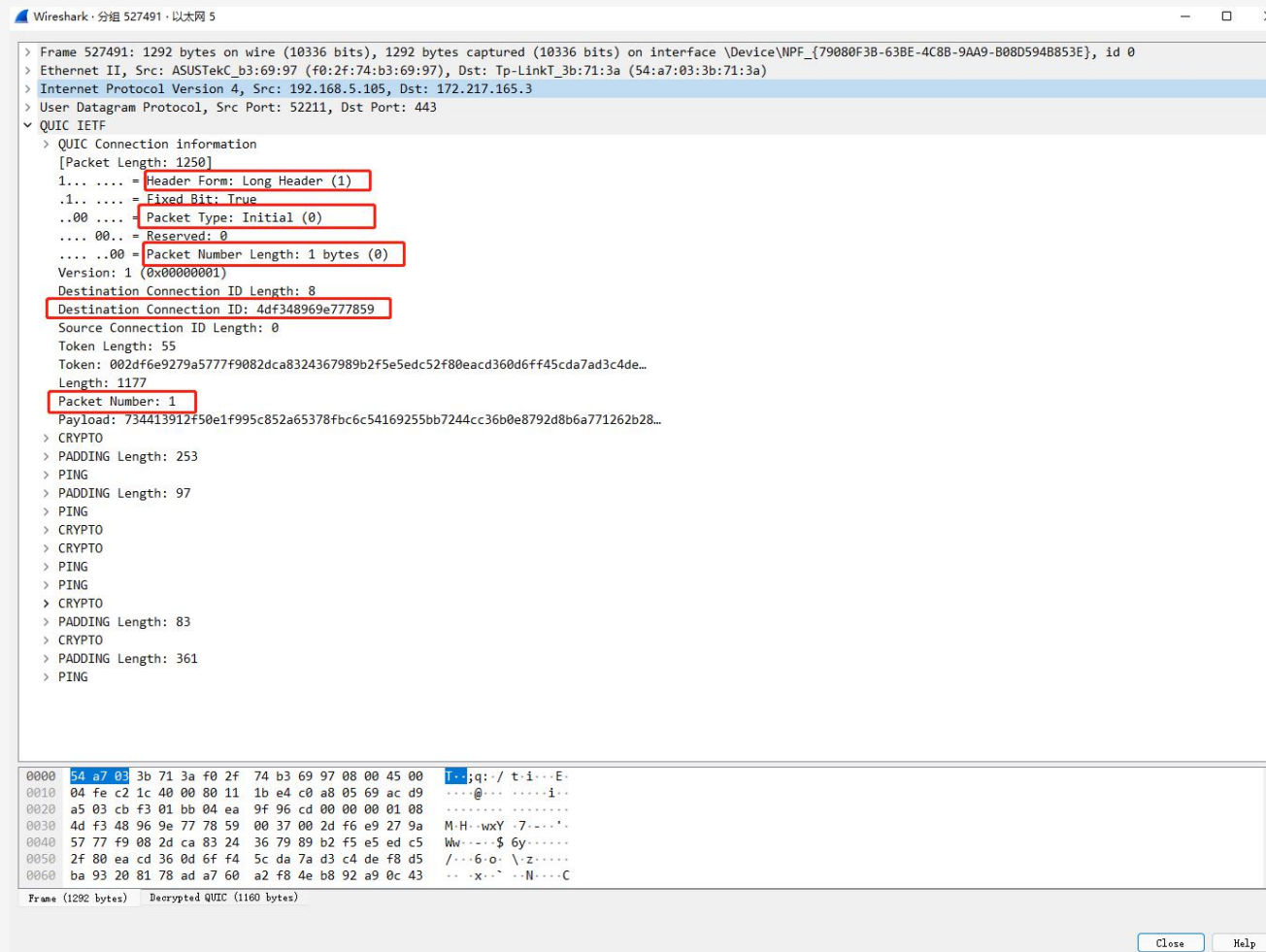


# QUIC connection/stream/frame (RFC 9000)

## P12.3



# QUIC connection/stream/frame (RFC 9000)



```
Wireshark - 分组 527491 - 以太网 5
> Frame 527491: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface \Device\NPF_{79080F3B-63BE-4C8B-9AA9-B08D594B853E}, id 0
> Ethernet II, Src: ASUSTekC_b3:69:97 (f0:2f:74:b3:69:97), Dst: Tp-LinkT_3b:71:3a (54:a7:03:3b:71:3a)
> Internet Protocol Version 4, Src: 192.168.5.105, Dst: 172.217.165.3
> User Datagram Protocol, Src Port: 52211, Dst Port: 443
< QUIC IETF
  < QUIC Connection information
    [Packet Length: 1250]
    1... .... = Header Form: Long Header (1)
    .1... .... = Fixed Bit: True
    ..00 .... = Packet Type: Initial (0)
    .... 00.. = Reserved: 0
    .... ..00 = Packet Number Length: 1 bytes (0)
    Version: 1 (0x00000001)
    Destination Connection ID Length: 8
    Destination Connection ID: 4df348969e777859
    Source Connection ID Length: 0
    Token Length: 55
    Token: 002df6e9279a5777f9082dca8324367989b2f5e5edc52f80eacd360d6ff45cda7ad3c4de...
    Length: 1177
    Packet Number: 1
    Payload: 734413912f50e1f995c852a65378fbc6c54169255bb7244cc36b0e8792d8b6a771262b28...
  < CRYPTO
  < PADDING Length: 253
  < PING
  < PADDING Length: 97
  < PING
  < CRYPTO
  < PING
  < PING
  < CRYPTO
  < PADDING Length: 83
  < CRYPTO
  < PADDING Length: 361
  < PING
0000  54 a7 03 3b 71 3a f0 2f 74 b3 69 97 08 00 45 00  |.;q:/ t.i...E
0010  04 fe c2 1c 40 00 80 11 1b e4 c0 a8 05 69 ac d9  |...@... ..i...
0020  a5 03 cb f3 01 bb 04 ea 9f 96 cd 00 00 00 01 08  |.....
0030  4d f3 48 96 9e 77 78 59 00 37 00 2d f6 e9 27 9a  |M.H.wxY -7....
0040  57 77 f9 08 2d ca 83 24 36 79 89 b2 f5 e5 ed c5  |Ww...$ gy....
0050  2f 80 ea cd 36 0d 6f f4 5c da 7a d3 c4 de f8 d5  |/...6.o. \z....
0060  ba 93 20 81 78 ad a7 60 a2 f8 4e b8 92 a9 0c 43  |...x... -N....C
Frame (1292 bytes) | Decrypted QUIC (1160 bytes)
Close Help
```



# QUIC connection/stream/frame (RFC 9000)

The image shows a Wireshark packet capture window titled "Wireshark · 分组 517288 · 以太网 5". The selected packet is Frame 517288, which is a QUIC IETF packet. The packet details pane shows the following structure:

- Frame 517288: 1322 bytes on wire (10576 bits), 1322 bytes captured (10576 bits) on interface \Device\NPF\_{79080F3B-63BE-4C8B-9AA9-B08D594B853E}, id 0
- Ethernet II, Src: Tp-LinkT\_3b:71:3a (54:a7:03:3b:71:3a), Dst: ASUSTekC\_b3:69:97 (f0:2f:74:b3:69:97)
- Internet Protocol Version 4, Src: 151.101.125.57, Dst: 192.168.5.105
- User Datagram Protocol, Src Port: 443, Dst Port: 49888
- QUIC IETF
  - QUIC Connection information
    - [Connection Number: 46]
    - [Packet Length: 1280]
  - QUIC Short Header PKN=417
    - 0... .. = Header Form: Short Header (0)
    - .1.. .... = Fixed Bit: True
    - ..0. .... = Spin Bit: False
    - ...0 0... = Reserved: 0
    - .... 0.. = Key Phase Bit: False
    - ..... 01 = Packet Number Length: 2 bytes (1)
    - Packet Number: 417
    - Protected Payload: 7f670150a1a30068933264d4683b4e4b081949128214260d82aae5f269428fa2be0c8818...
  - STREAM id=64 fin=0 off=61 len=1257 dir=Bidirectional origin=Client-initiated
    - Frame Type: STREAM (0x000000000000000c)
    - Stream ID: 64
      - .....0 = Stream initiator: Client-initiated (0)
      - .....0 = Stream direction: Bidirectional (0)
      - Offset: 61
      - Stream Data: 71a003579afe472c72bed88923458db13eeb3248ec8dbed331b6f842cbbfcfeeeeae05696...

The packet bytes pane shows the raw data of the frame, with the first few bytes highlighted in blue:

```
0020 05 69 01 bb c2 e0 05 08 4f 11 4a cd 88 7f 67 01  -i.....0-J...g-
0030 50 a1 a3 00 68 93 32 64 d4 68 3b 4e 4b 08 19 49  P...h-2d-h;NK-I
0040 12 82 14 26 0d 82 aa e5 f2 69 42 8f a2 be 0c 88  ...&....iB....
0050 18 37 2c 92 63 c2 93 e0 e0 bd b2 36 b8 f4 d1 05  -7,-c...-6...-
0060 6d 73 bb d8 d1 b6 d7 27 4d f3 68 8f 38 3b b3 a4  ms.....M-h-8;..
0070 20 3a 4f fb 0c 80 2e 53 5b 32 d0 58 ac 56 cf 09  :0....S [2-X-V..
0080 97 a6 64 4a bc b1 8c c0 5c a2 53 12 fe c4 57 85  -dJ....\S...W-
0090 54 00 44 fa a8 78 ca a6 c0 d3 26 28 9d 4a 65 af  T-D-x...-&(-Je-
```

At the bottom of the window, there are buttons for "Close" and "Help".



# QUIC encryption level (RFC 9001)

## 12.3. Packet Numbers

The packet number is an integer in the range 0 to  $2^{62}-1$ . This number is used in determining the cryptographic nonce for packet protection. Each endpoint maintains a separate packet number for sending and receiving.

Packet numbers are limited to this range because they need to be representable in whole in the Largest Acknowledged field of an ACK frame (Section 19.3). When present in a long or short header, however, packet numbers are reduced and encoded in 1 to 4 bytes: see Section 17.1.

Version Negotiation (Section 17.2.1) and Retry (Section 17.2.5) packets do not include a packet number.

Packet numbers are divided into three spaces in QUIC:

Initial space: All Initial packets (Section 17.2.2) are in this space.

Handshake space: All Handshake packets (Section 17.2.4) are in this space.

Application data space: All 0-RTT (Section 17.2.3) and 1-RTT (Section 17.3.1) packets are in this space.

As described in [QUIC-TLS], each packet type uses different protection keys.

Conceptually, a packet number space is the context in which a packet can be processed and acknowledged. Initial packets can only be sent with Initial packet protection keys and acknowledged in packets that are also Initial packets. Similarly, Handshake packets are sent at the Handshake encryption level and can only be acknowledged in Handshake packets.

This enforces cryptographic separation between the data sent in the different packet number spaces. Packet numbers in each space start at packet number 0. Subsequent packets sent in the same packet number space MUST increase the packet number by at least one.

## 5.1. Packet Protection Keys

QUIC derives packet protection keys in the same way that TLS derives record protection keys.

Each encryption level has separate secret values for protection of packets sent in each direction. These traffic secrets are derived by TLS (see Section 7.1 of [TLS13]) and are used by QUIC for all encryption levels except the Initial encryption level. The secrets for the Initial encryption level are computed based on the client's initial Destination Connection ID, as described in Section 5.2.

The keys used for packet protection are computed from the TLS secrets using the KDF provided by TLS. In TLS 1.3, the HKDF-Expand-Label function described in Section 7.1 of [TLS13] is used, using the hash function from the negotiated cipher suite. All uses of HKDF-Expand-Label in QUIC use a zero-length Context.

Note that labels, which are described using strings, are encoded as bytes using ASCII [ASCII] without quotes or any trailing NUL byte.

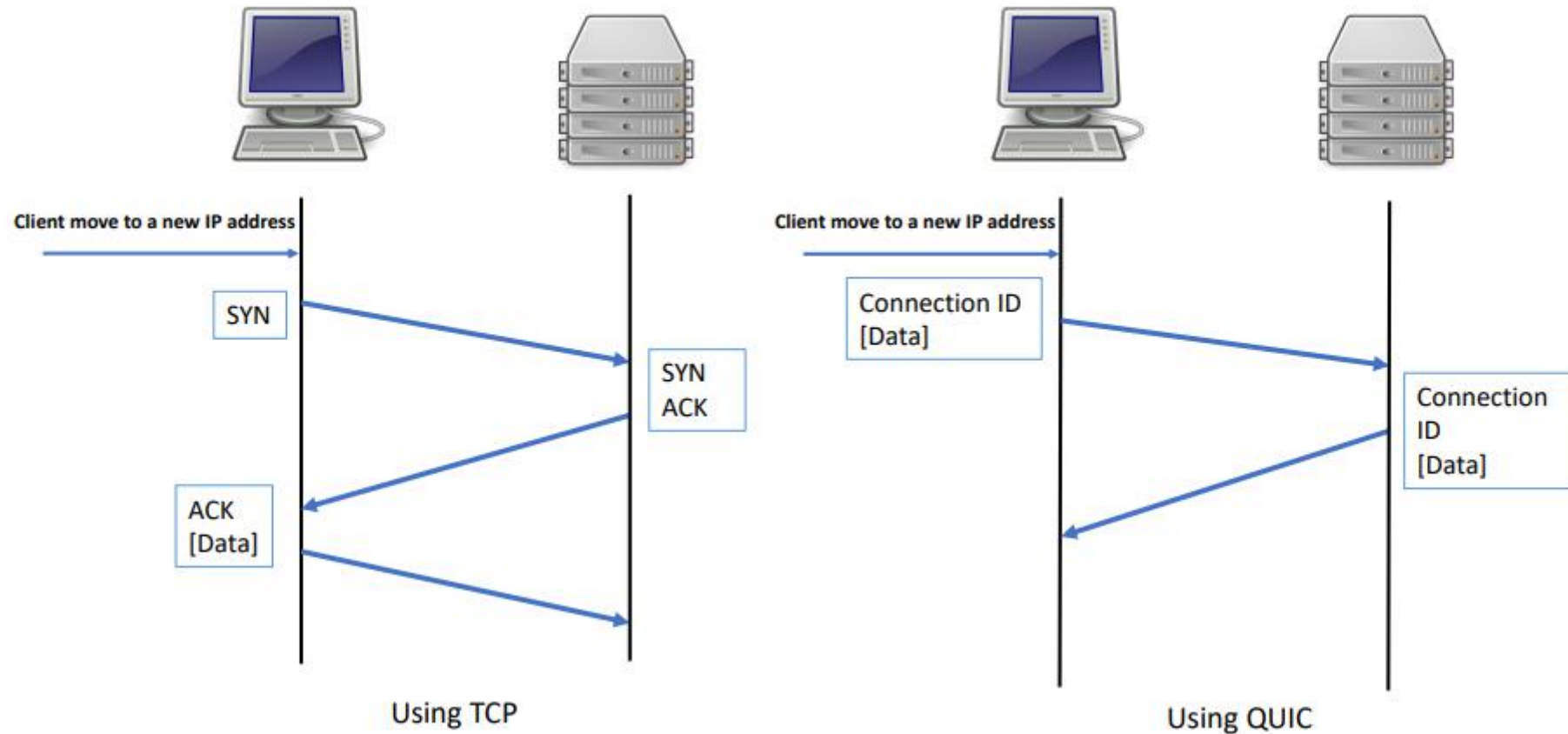
Other versions of TLS **MUST** provide a similar function in order to be used with QUIC.

The current encryption level secret and the label "quic key" are input to the KDF to produce the AEAD key; the label "quic iv" is used to derive the Initialization Vector (IV); see Section 5.3. The header protection key uses the "quic hp" label; see Section 5.4. Using these labels provides key separation between QUIC and TLS; see Section 9.6.

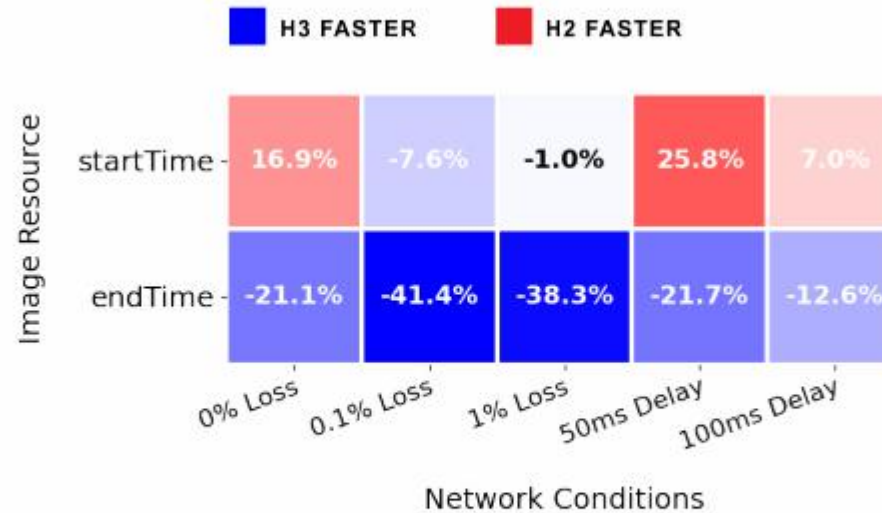
Both "quic key" and "quic hp" are used to produce keys, so the Length provided to HKDF-Expand-Label along with these labels is determined by the size of keys in the AEAD or header protection algorithm. The Length provided with "quic iv" is the minimum length of the AEAD nonce or 8 bytes if that is larger; see [AEAD].

The KDF used for initial secrets is always the HKDF-Expand-Label function from TLS 1.3; see Section 5.2.

# connection migration

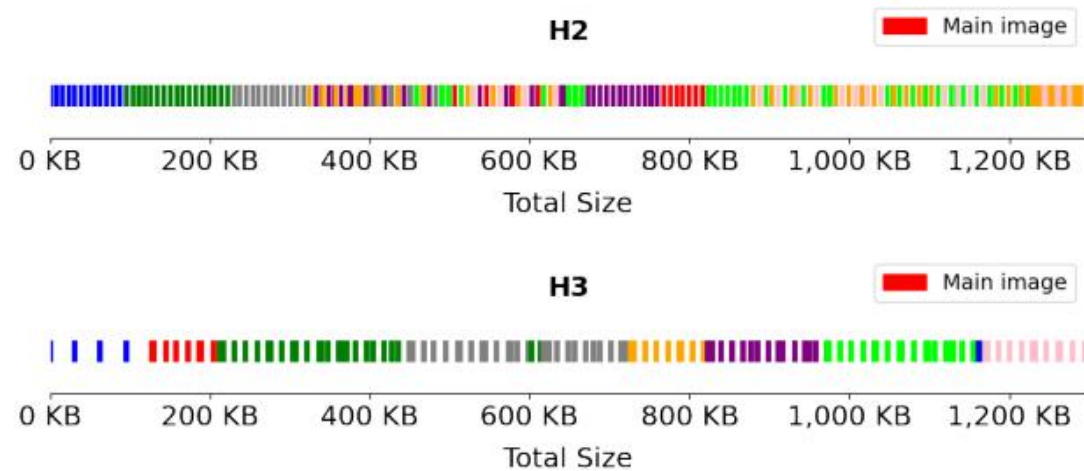


# HTTP 2 VS HTTP 3 performance



**Figure 15: H2 vs H3 performance for the start and end times of the 'main' image resource of our large-sized Cloudflare web-page.**

# HTTP 2 VS HTTP 3 performance



**Figure 16: HTTP frames received by Chrome for 8 simultaneously-requested image/gif resources from our large-sized Cloudflare web-page. Each color represents a unique HTTP stream and each bar represents a chunk of data read by the application layer. Notice that H3 not only received the main image's<sup>5</sup> frames earlier, but also received them exclusively (H2 started receiving main image frames around the 500KB mark).**

# Problems

- UDP rate limiting and blocking
- More cpu usage
- DDOS
- Replay attack



# debug time: QUIC complete process (chrome://net-internals/#events)(https://quic.cloud/)

The screenshot displays the netlog-viewer interface for a Chrome net-export-log.json file. The left sidebar shows a list of events, with the '2128: QUIC\_SESSION' event highlighted. The main panel shows the detailed log for this session, starting at 2021-12-16 02:27:47.589. The log includes the following key events:

- 2128: QUIC\_SESSION** (Start Time: 2021-12-16 02:27:47.589)
- t=69547 [st=23859]: QUIC\_CHROMIUM\_CLIENT\_STREAM\_SEND\_REQUEST\_HEADERS**
  - method: GET
  - authority: quic.cloud
  - scheme: https
  - path: /
  - pragma: no-cache
  - cache-control: no-cache
  - sec-ch-ua: "Not A;Brand",v="99", "Chromium",v="96", "Google Chrome",v="96"
  - sec-ch-ua-mobile: ?0
  - sec-ch-ua-platform: "Windows"
  - dt: 1
  - upgrade-insecure-requests: 1
  - user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36
  - accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9
  - sec-fetch-site: cross-site
  - sec-fetch-mode: navigate
  - sec-fetch-user: ?1
  - sec-fetch-dest: document
  - referrer: https://www.google.com/
  - accept-encoding: gzip, deflate, br
  - accept-language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
  - cookie: [337 bytes were stripped]
  - quic\_priority = 0
  - quic\_stream\_id = 209
- t=69547 [st=23859]: QUIC\_SESSION\_STREAM\_FRAME\_SENT**
  - fin = false
  - length = 102
  - offset = 10723
  - stream\_id = 1
- t=69547 [st=23859]: QUIC\_SESSION\_PACKET\_SENT**
  - encryption\_level = "ENCRYPTION\_FORWARD\_SECURE"
  - packet\_number = 248
  - sent\_time\_us = 104670570493
  - size = 132
  - transmission\_type = "NOT\_RETRANSMISSION"
- t=69776 [st=24088]: QUIC\_SESSION\_PACKET\_RECEIVED**
  - peer\_address = "3.225.219.165:443"
  - self\_address = "192.168.5.105:64056"
  - size = 1250
- t=69776 [st=24088]: QUIC\_SESSION\_UNAUTHENTICATED\_PACKET\_HEADER\_RECEIVED**
  - connection\_id = "24fdd85b902634b"
  - header\_format = "IETF\_QUIC\_SHORT\_HEADER\_PACKET"
  - packet\_number = 1047
- t=69776 [st=24088]: QUIC\_SESSION\_PACKET\_AUTHENTICATED**
- t=69776 [st=24088]: QUIC\_SESSION\_STREAM\_FRAME\_RECEIVED**
  - fin = false
  - length = 49
  - offset = 7454
  - stream\_id = 1
- t=69776 [st=24088]: QUIC\_SESSION\_STREAM\_FRAME\_RECEIVED**
  - fin = false
  - length = 1174
  - offset = 0
  - stream\_id = 209
- t=69776 [st=24088]: QUIC\_SESSION\_PACKET\_RECEIVED**
  - peer\_address = "3.225.219.165:443"
  - self\_address = "192.168.5.105:64056"
  - size = 1250
- t=69776 [st=24088]: QUIC\_SESSION\_UNAUTHENTICATED\_PACKET\_HEADER\_RECEIVED**
  - connection\_id = "24fdd85b902634b"
  - header\_format = "IETF\_QUIC\_SHORT\_HEADER\_PACKET"
  - packet\_number = 1048
- t=69776 [st=24088]: QUIC\_SESSION\_PACKET\_AUTHENTICATED**
- t=69776 [st=24088]: QUIC\_SESSION\_STREAM\_FRAME\_RECEIVED**
  - fin = false
  - length = 1907

# Reference

- <https://web.cs.ucla.edu/~lixia/papers/UnderstandQUIC.pdf>
- <https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-34>
- <https://medium.com/jspoint/how-the-browser-renders-a-web-page-dom-cssom-and-rendering-df10531c9969>
- <https://developers.google.com/web/fundamentals/performance/http2>
- <https://www.slideshare.net/lmacvittie/http2-changes-everything>
- <http://xiaorui.cc/static/http2quic.pdf>
- <https://arxiv.org/pdf/1810.07730.pdf>
- [https://cs.brown.edu/~tab/papers/QUIC\\_WWW21.pdf](https://cs.brown.edu/~tab/papers/QUIC_WWW21.pdf)