NGINX

Nginx支持QUIC/HTTP3的实现路径和实践思考

陶辉

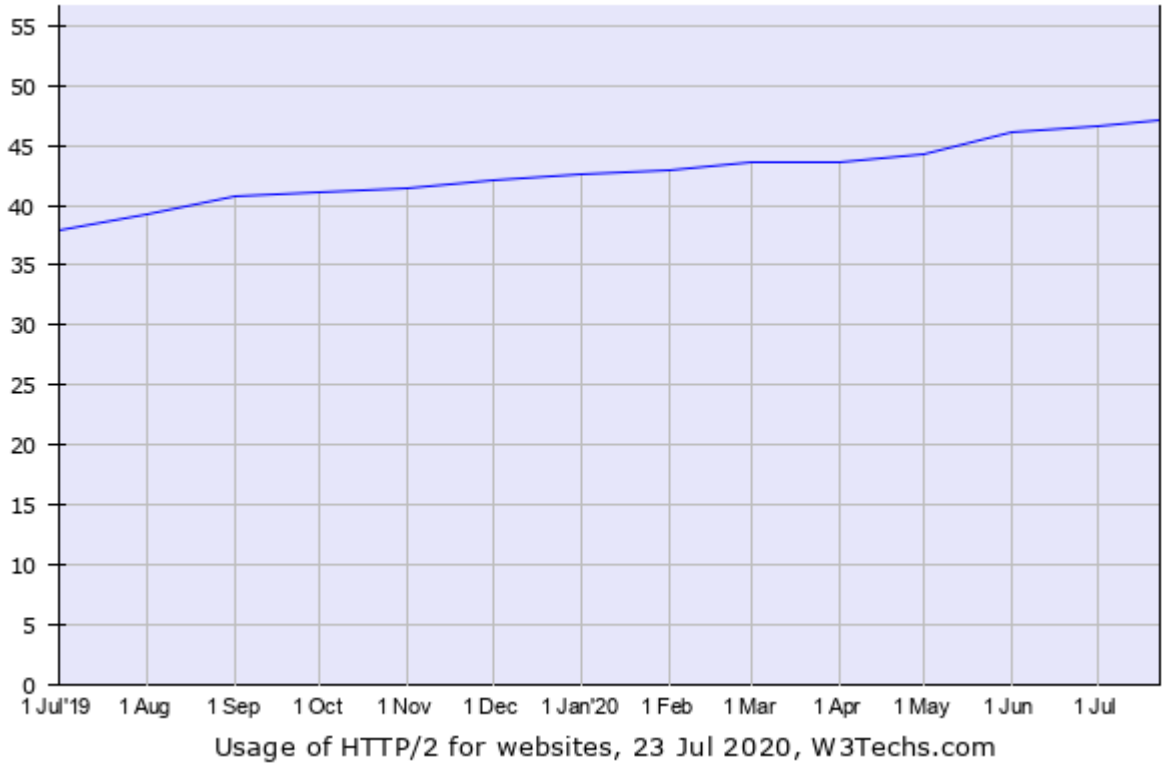
目录

HTTP3协议的概念与细节

- HTTP2协议遗留了哪些问题?
- HTTP3如何在UDP协议之上实现了连接迁移?
- 通过QUIC Frame、HTTP3 Frame，如何实现多路复用?
- QPACK是怎样解决队头阻塞问题的?
- HTTP3如何处理丢包问题?

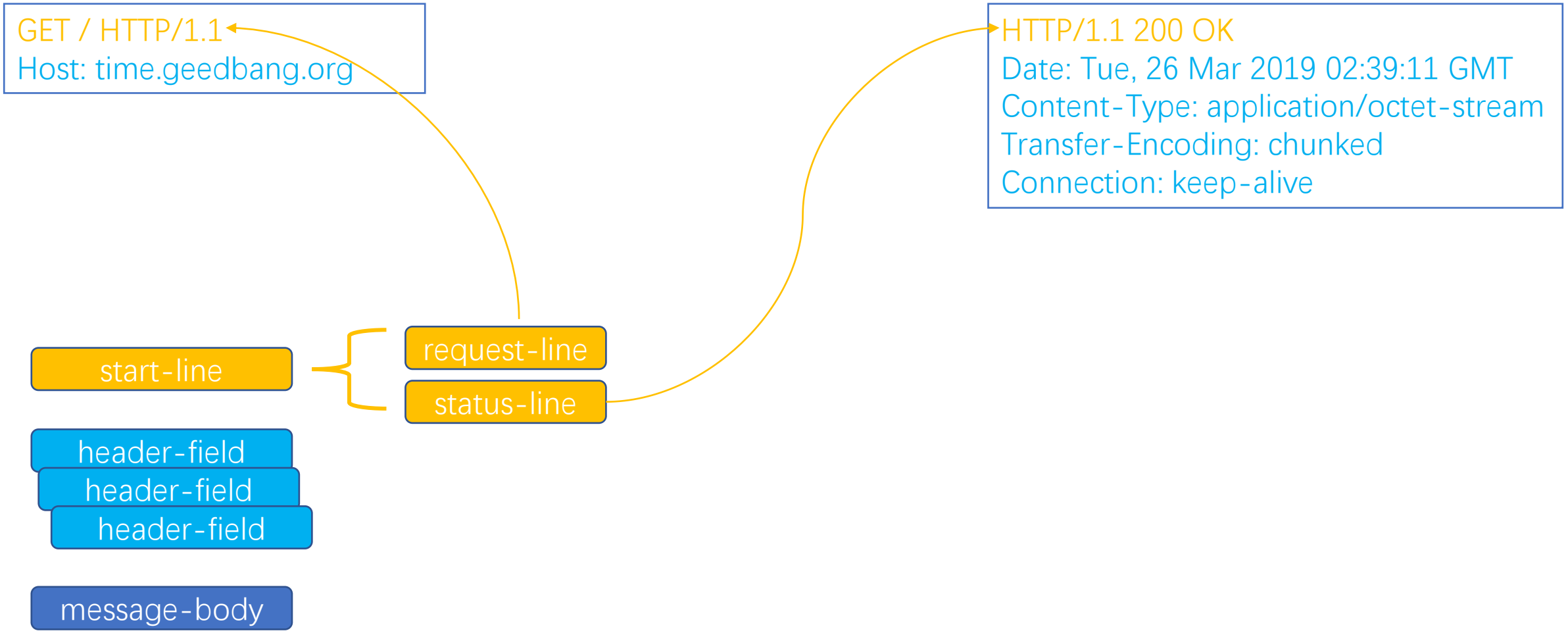
HTTP协议的发展历史

- 1991 HTTP/0.9
- 1996 HTTP/1.0
- 1999 HTTP/1.1
- 2015 HTTP/2
- 2020 HTTP/3



Usage of HTTP/2 for websites, 23 Jul 2020, W3Techs.com

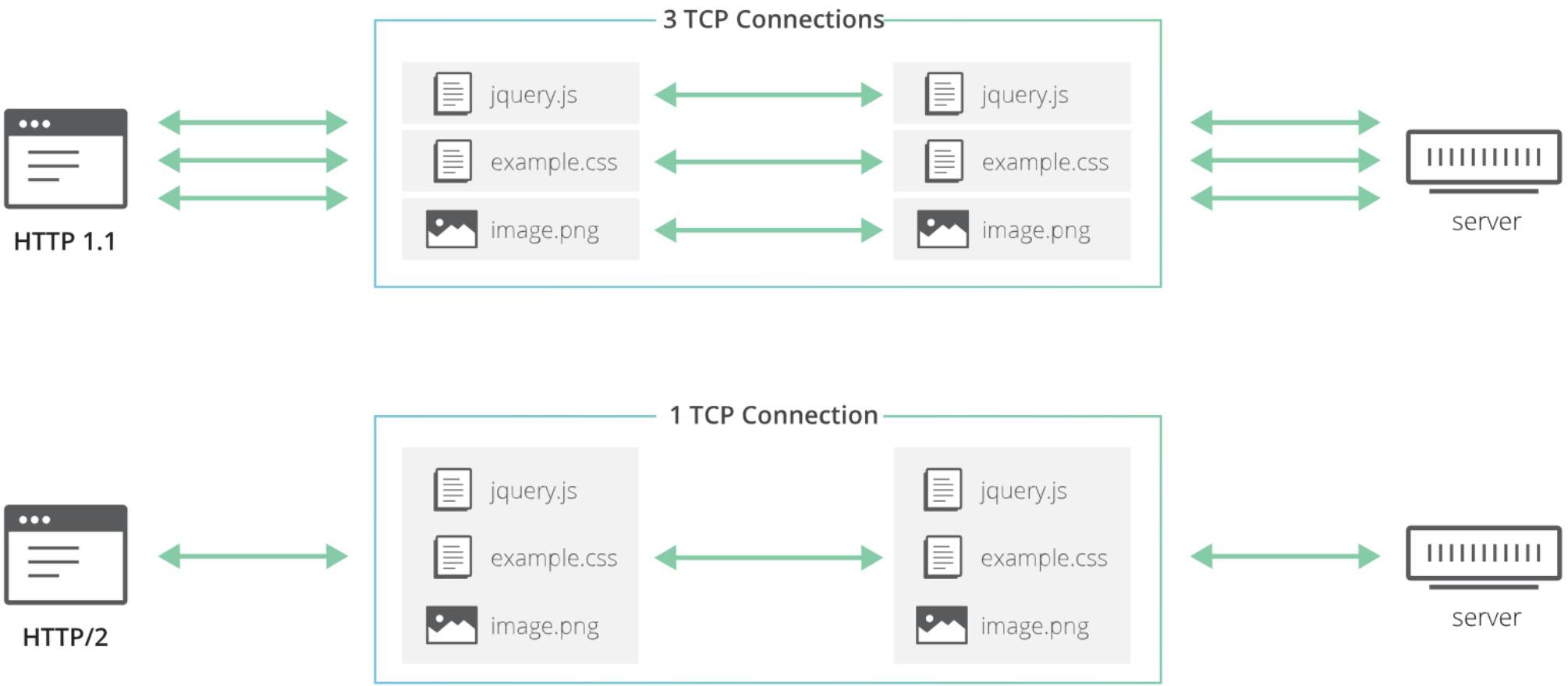
HTTP/1协议格式


GET / HTTP/1.1
Host: time.geedbang.org
HTTP/1.1 200 OK
Date: Tue, 26 Mar 2019 02:39:11 GMT
Content-Type: application/octet-stream
Transfer-Encoding: chunked
Connection: keep-alive
start-line
request-line
status-line
header-field
header-field
header-field
message-body

HTTP1语义及HTTP1协议问题

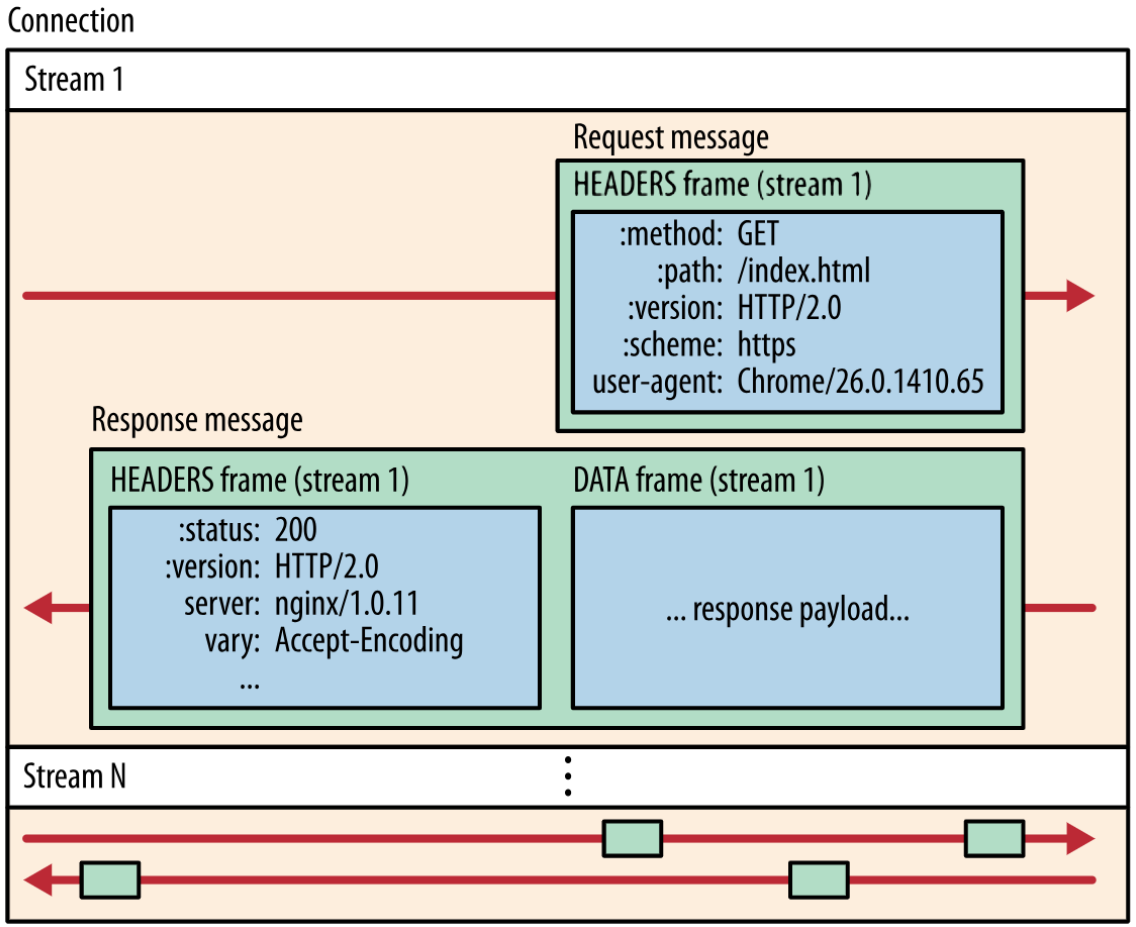
- **HTTP1语义**
  - Client Request / Server Response机制
  - header + body编码格式
- **HTTP1协议的问题**
  - header编码效率低：stateless
  - 多路复用成本高
    - 慢启动
    - 建连接
    - 异步编程开发效率
  - 长连接传输中无法中止请求
  - 不支持服务器推送
  - HTTP层不支持流控

多路复用

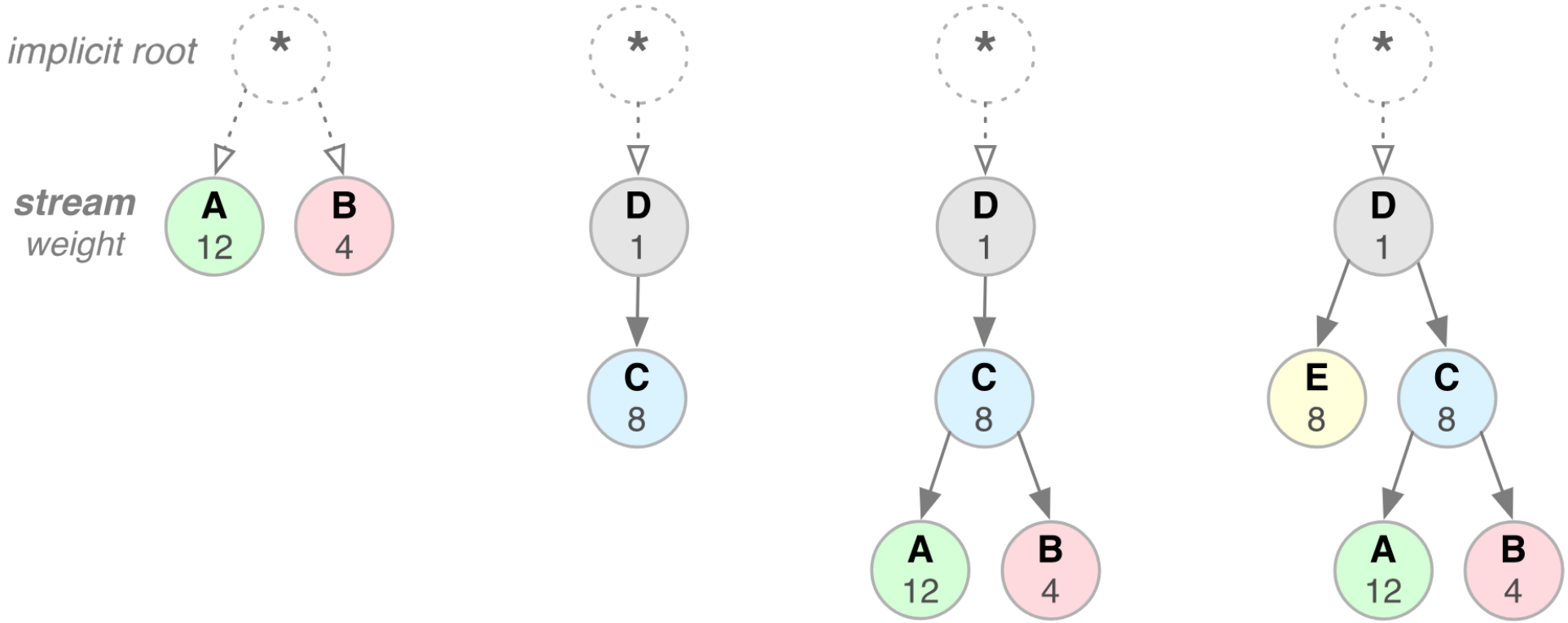

3 TCP Connections
jquery.js
example.css
image.png
jquery.js
example.css
image.png
server
HTTP 1.1
1 TCP Connection
jquery.js
example.css
image.png
jquery.js
example.css
image.png
server
HTTP/2

HTTP2协议

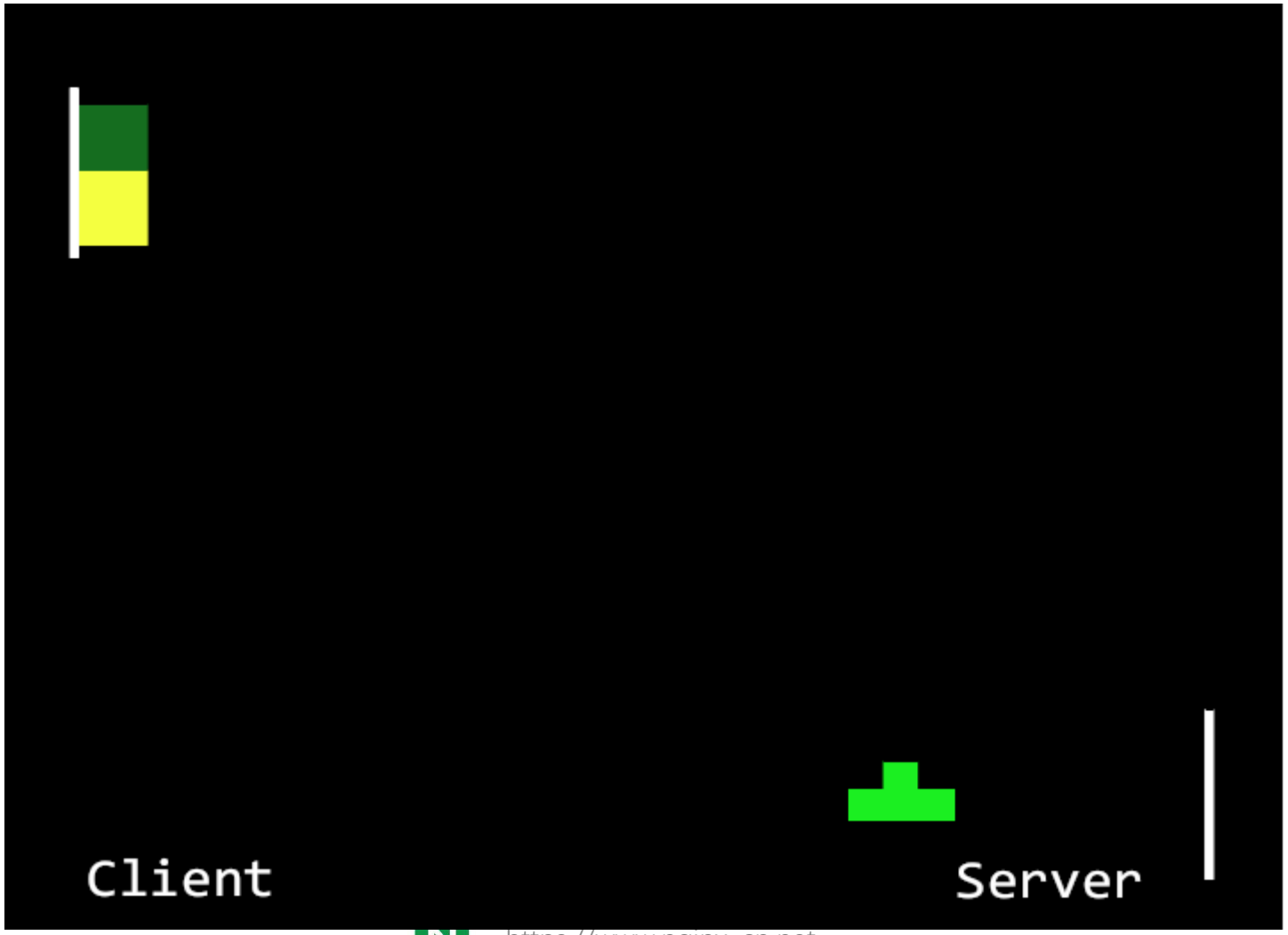
- 多路复用
  - 连接、Stream级流控
  - 带权重、依赖的优先级
  - Stream Reset置位
- HPACK头部编码
- 服务器消息推送



HTTP2的优先级与流控

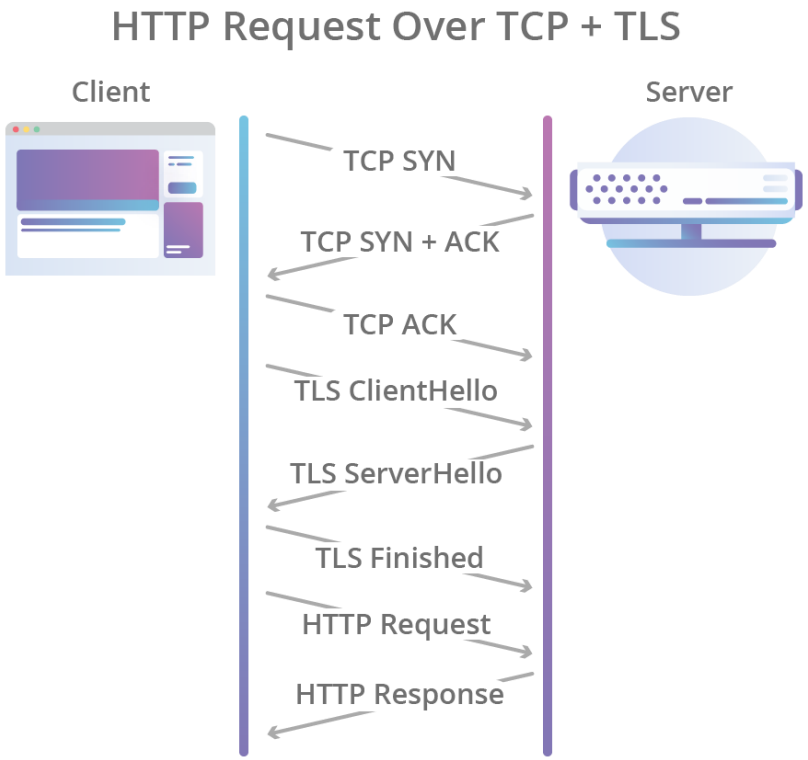


HTTP2协议的遗留问题（1）：队头阻塞



HTTP2协议的遗留问题（2）：建连接

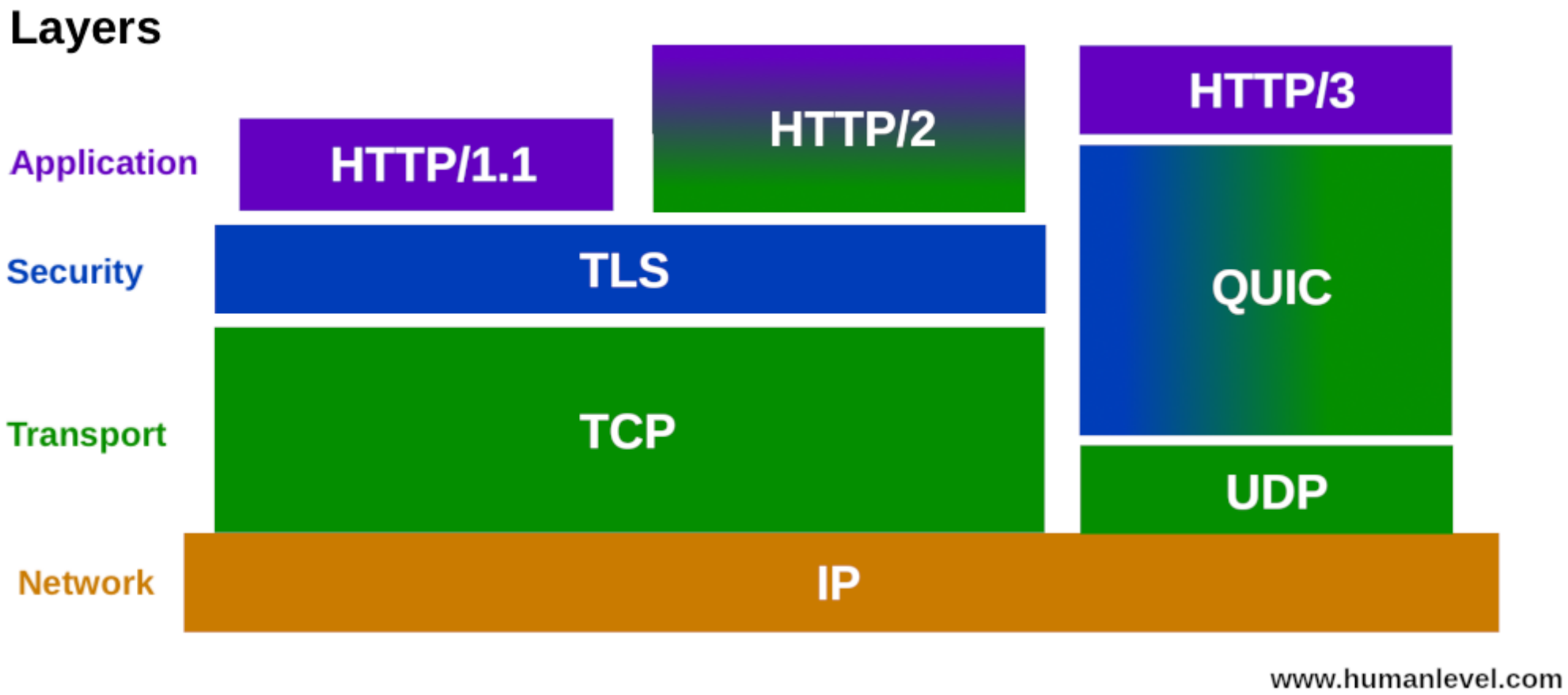
- 2次握手
  - 传输层握手
  - TLS层握手



HTTP3协议的概念与细节

- HTTP2协议遗留了哪些问题?
- HTTP3如何在UDP协议之上实现了连接迁移?
- 通过QUIC Frame、HTTP3 Frame，如何实现多路复用?
- QPACK是怎样解决队头阻塞问题的?
- HTTP3如何处理丢包问题?

HTTP3协议

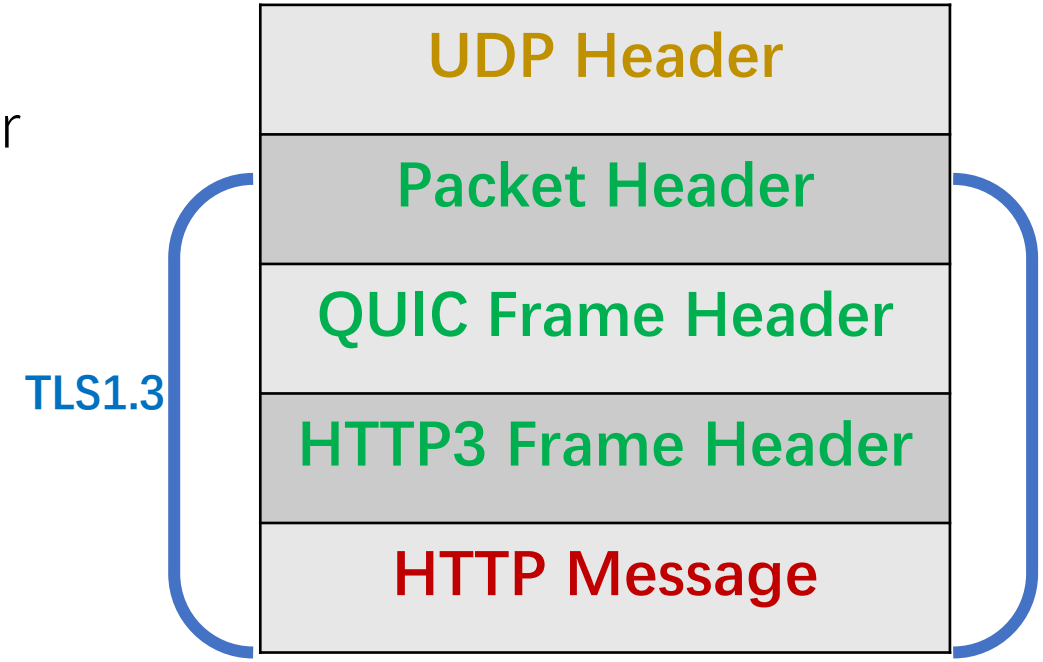

Layers
Application
Security
Transport
Network
HTTP/1.1
HTTP/2
HTTP/3
TLS
QUIC
TCP
UDP
IP
www.humanlevel.com

HTTP3协议

- RFC
  1. QUIC: https://tools.ietf.org/html/draft-ietf-quic-transport-27
  2. QUIC+TLS: https://tools.ietf.org/html/draft-ietf-quic-tls-27
  3. 丢包重传: https://tools.ietf.org/html/draft-ietf-quic-recovery-27
  4. QPACK: https://tools.ietf.org/html/draft-ietf-quic-qpack-14
  5. HTTP3: https://tools.ietf.org/html/draft-ietf-quic-http-27

核心概念

- **Packet： UDP Payload**
  - Long Packet Header/Short Packet Header
  - 含有1个或者多个QUIC Frame
- **QUIC Frame**
  - 不可跨越Packet
  - QUIC Stream有序字节流
    - 双向Stream
    - 单向Stream
      - 传输QPACK动态表、设置、服务器推送
- **HTTP3 Frame**
  - 可跨越多个Packet
- **Message**

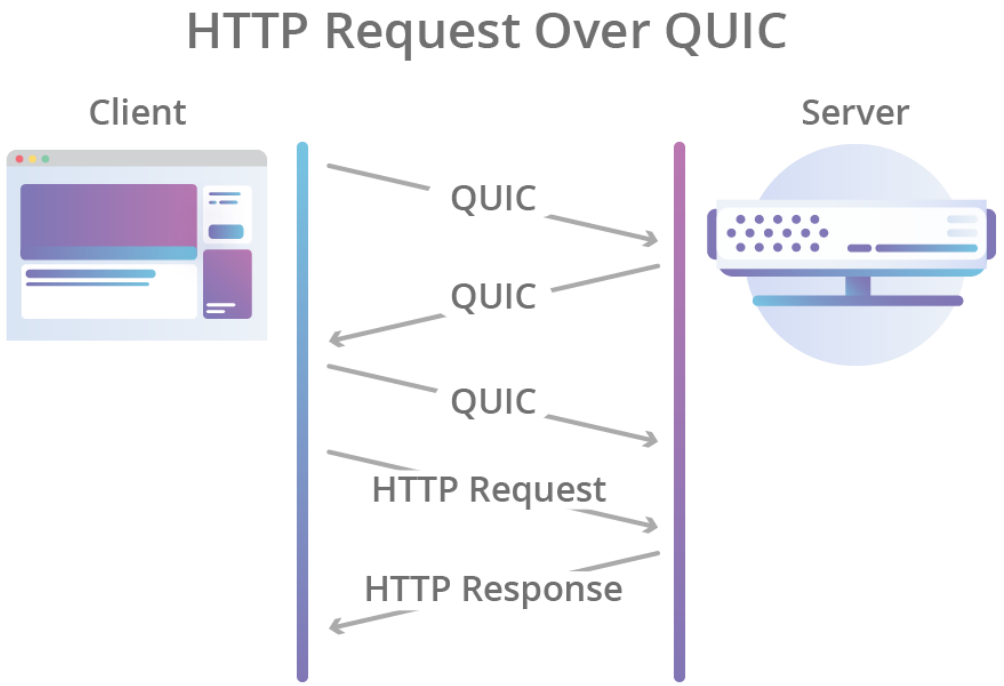


连接ID

- 不再以四元组定连接（源IP、源端口、目的IP、目的端口）
- 防止四层、七层负载均衡修改连接四元组信息
- 连接迁移
- 源连接ID，用于生成目的连接ID
  - 生成连接ID的一方，放在Packet的源连接ID中发送给对方
- 目的连接ID，用于连接的维持
  - 服务器根据Packet中的目的连接ID，决定Packet的归属，或者新建连接
  - 客户端首个目的连接ID是不可预测的8字节以上随机值

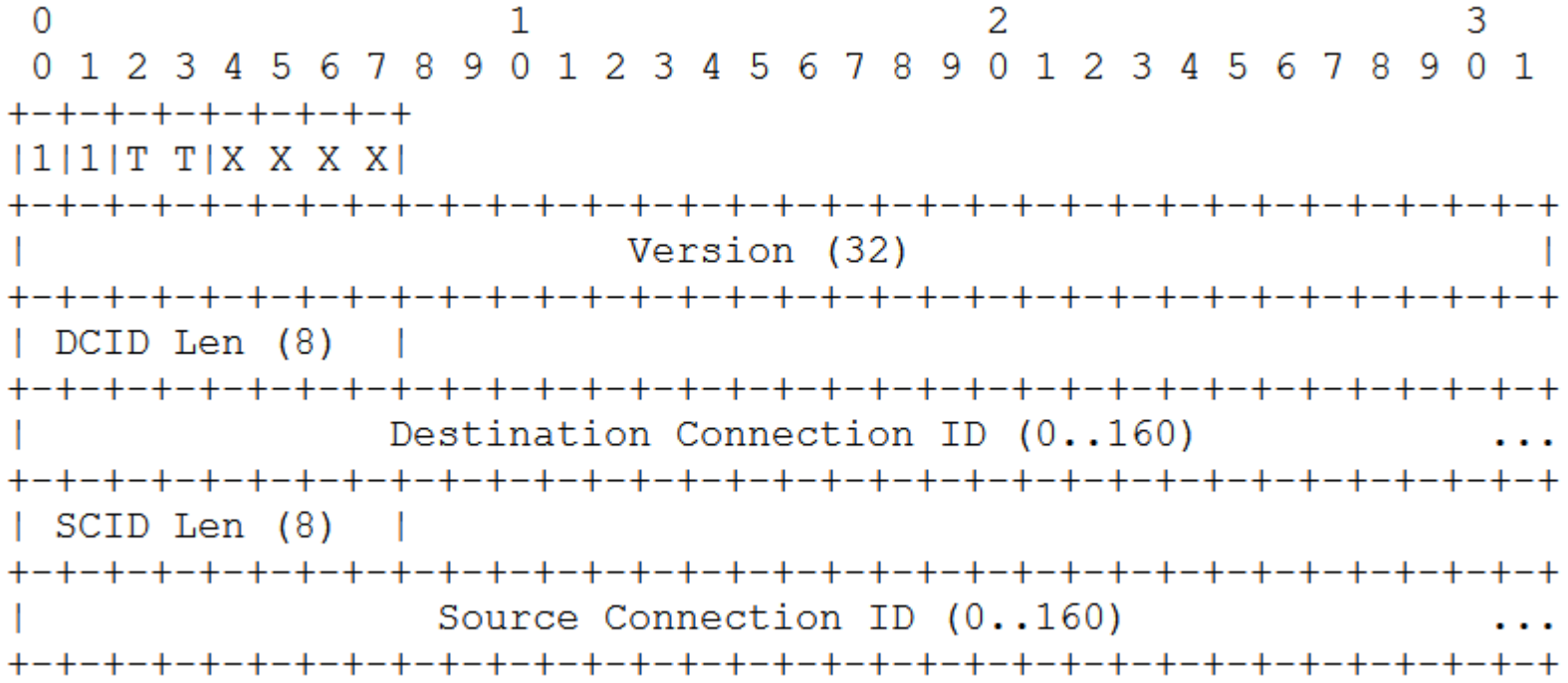
HTTP3的握手

- 1次握手
  - 版本协商
  - 传输层握手
  - TLS握手

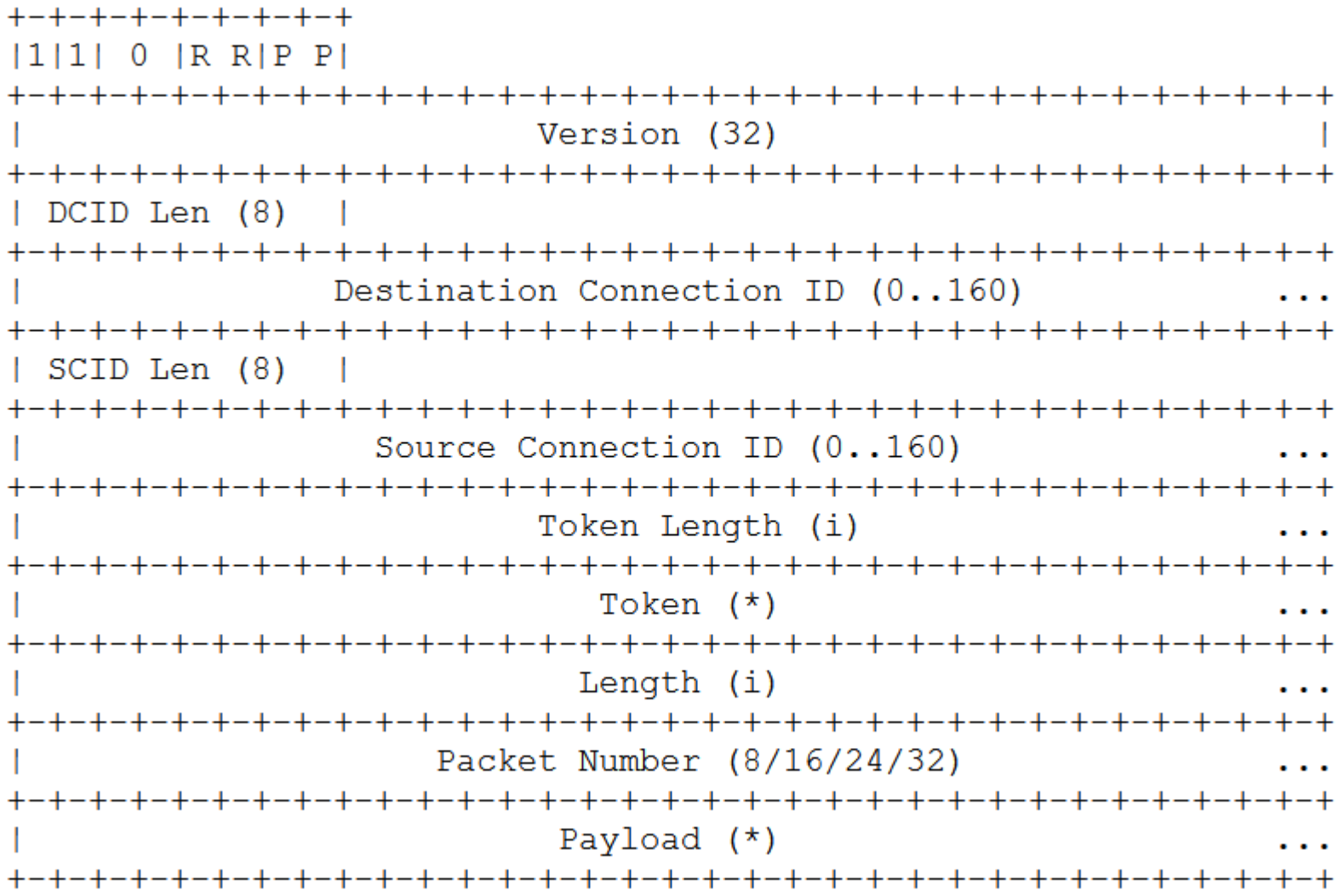


Long Header Packets：建立连接

- Header Form
  - 1 Long
  - 0 Short
- Fixed bit
- 2位Type
  - Initial
  - 0-RTT
  - HandleShake
  - Retry
- 4位Type-Specific
- Version
- DCID
- SCID



Initial Packet格式



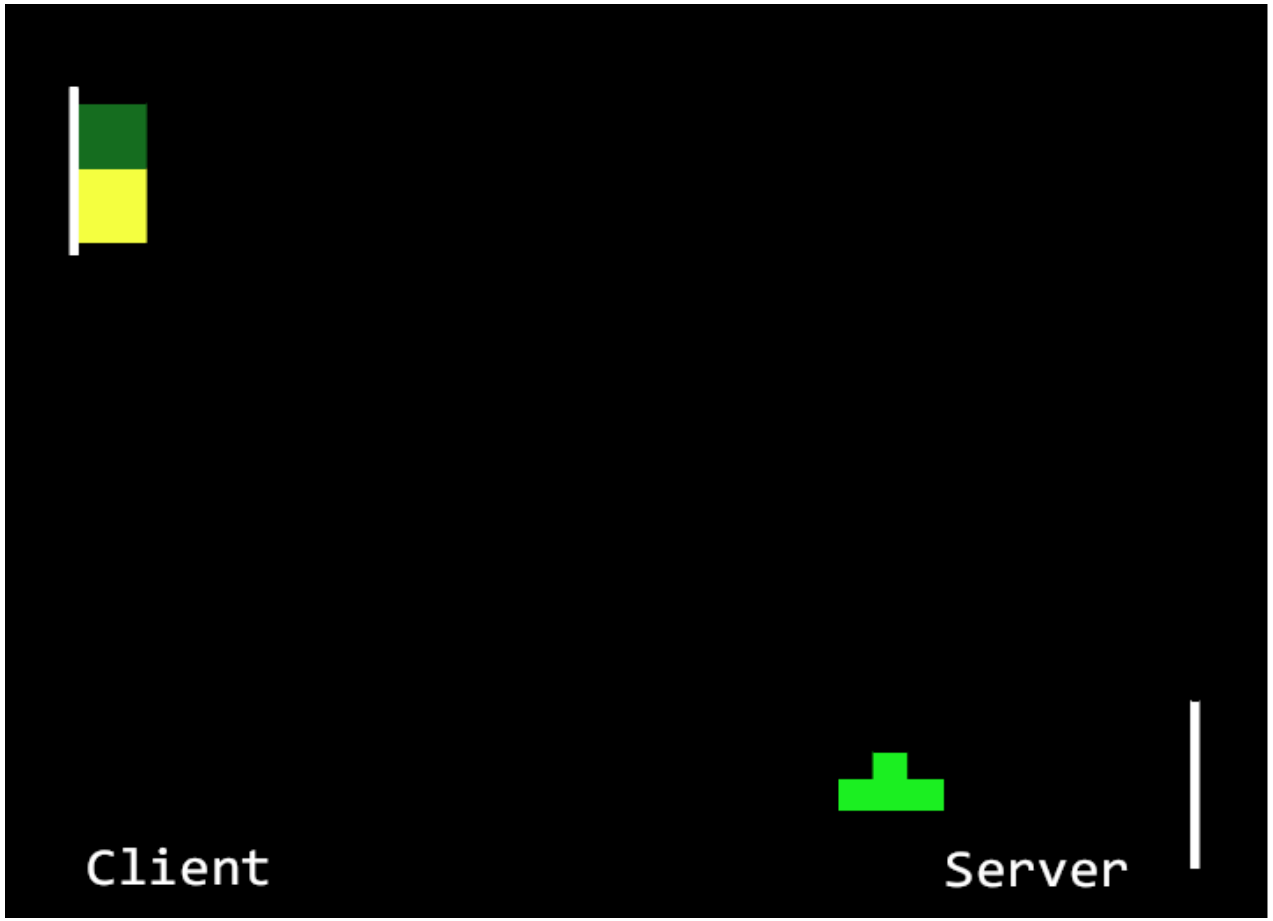
连接迁移

- **协议升级**
  - Draft27
    - Alt-Svc: h3-27=":50781"
- **连接迁移**
  - 基于固定的连接ID实现
  - 新路径验证
    - 加密的PATH_CHALLENGE Frame
      - 附带PMTU验证功能
    - PATH_RESPONSE Frame
  - 连接迁移后拥塞控制清零重来

HTTP3协议的概念与细节

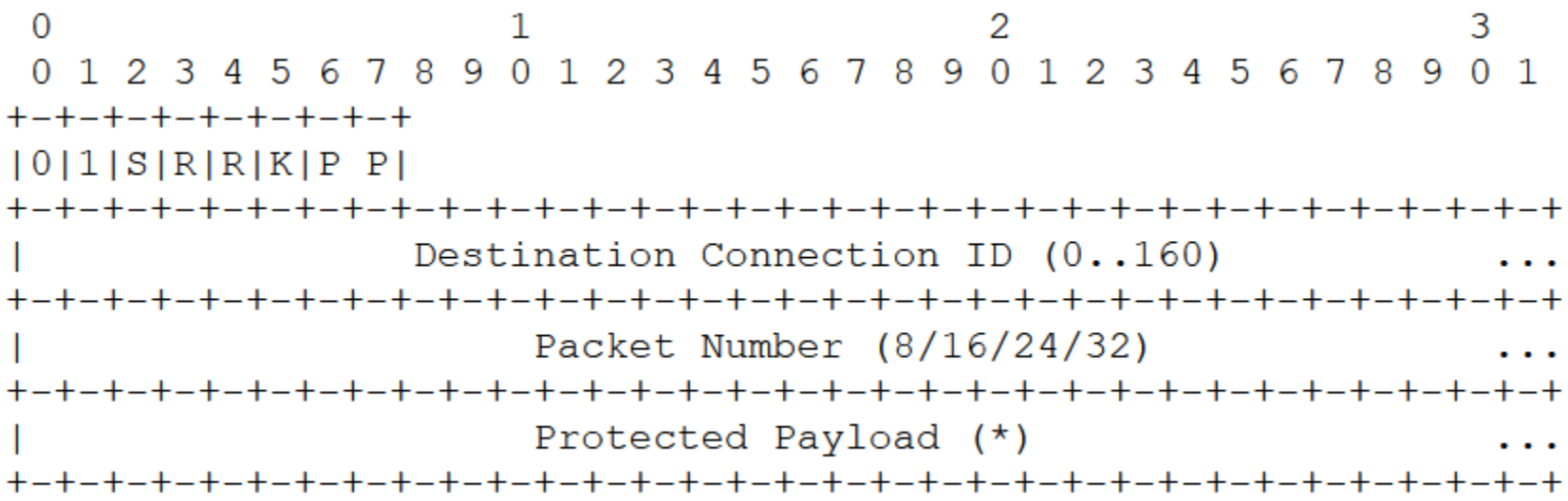
- HTTP2协议遗留了哪些问题?
- HTTP3如何在UDP协议之上实现了连接迁移?
- 通过QUIC Frame、HTTP3 Frame，如何实现多路复用?
- QPACK是怎样解决队头阻塞问题的?
- HTTP3如何处理丢包问题?

HTTP3解决队头阻塞问题的方案



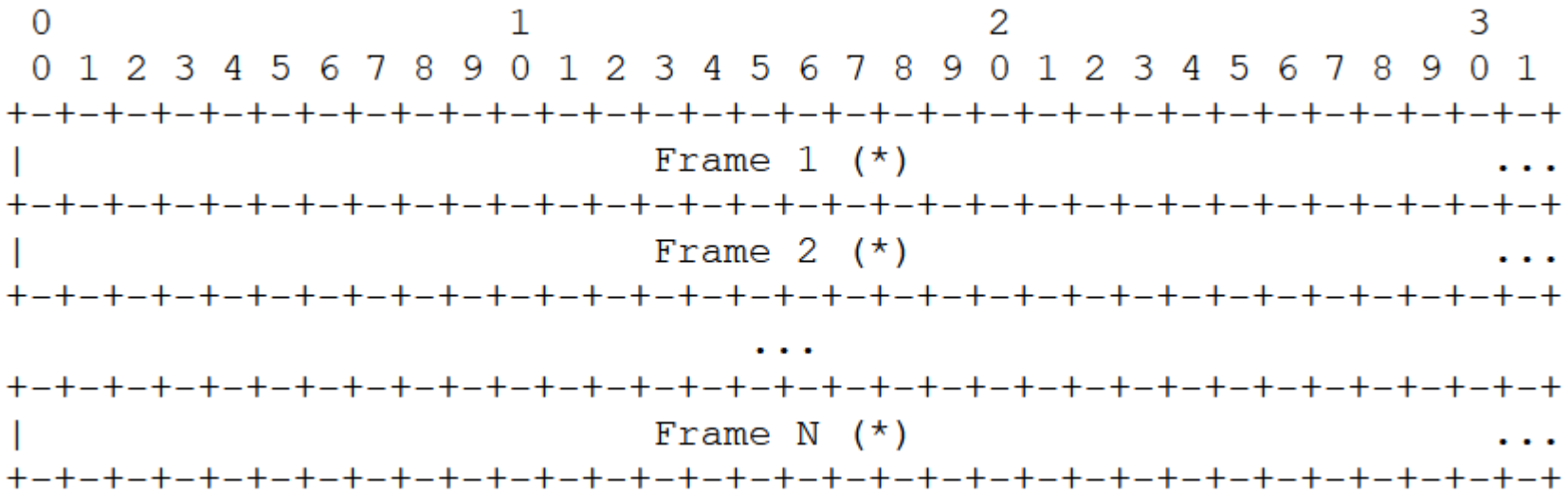
Short Header Packets

- Header Form
- Fixed Bits
- Spin Bits
- Reverse Bits
- Key Phase
- Packet Number Len
- DCID
- Packet Number
- Payload

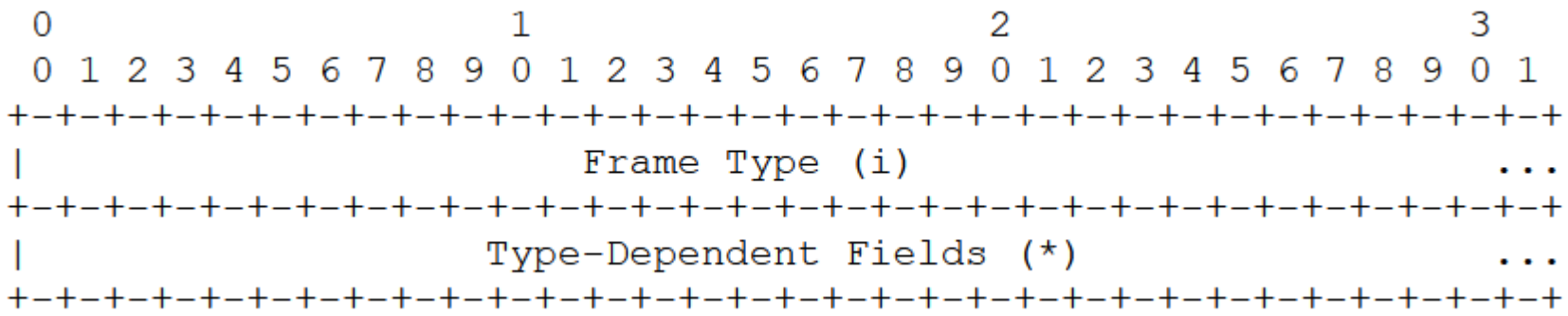


QUIC Frame格式

- Packet Payload



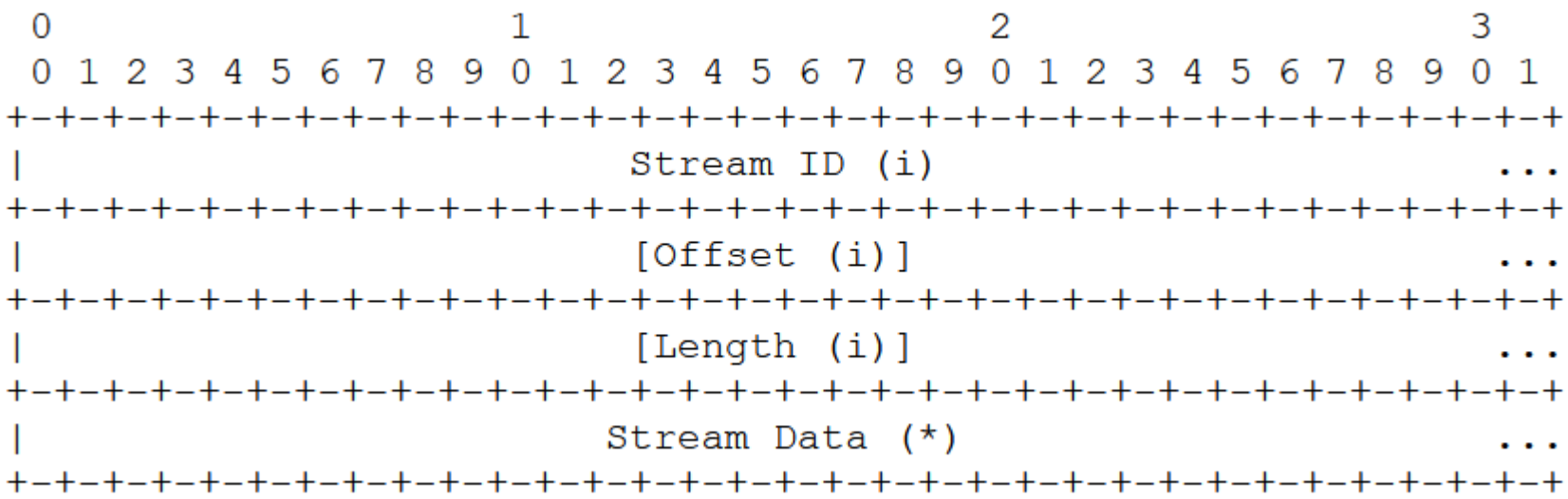
QUIC Frame类型



Value	Name	Value	Name		
0x00	PADDING	0x02 – 0x03	ACK	0x15	STREAM_DATA_BLOCKED
0x01	PING	0x08 – 0x0f	STREAM	0x18	NEW_CONNECTION_ID
0x04	RESET_STREAM	0x12-0x13	MAX_STREAMS	0x19	RETRY_CONNECTION_ID
0x05	STOP_SENDING	0x16-0x17	STREAM_BLOCKED	0x1a	PATH_CHALLENGE
0x06	CRYPTO	0x1c-0x1d	CONNECTION_CLOSE	0x1b	PATH_RESPONSE
0x07	NEW_TOKEN	0x11	MAX_STREAM_DATA	0x1e	HANDSHAKE_DONE
0x10	MAX_DATA	0x14	DATA_BLOCKED		

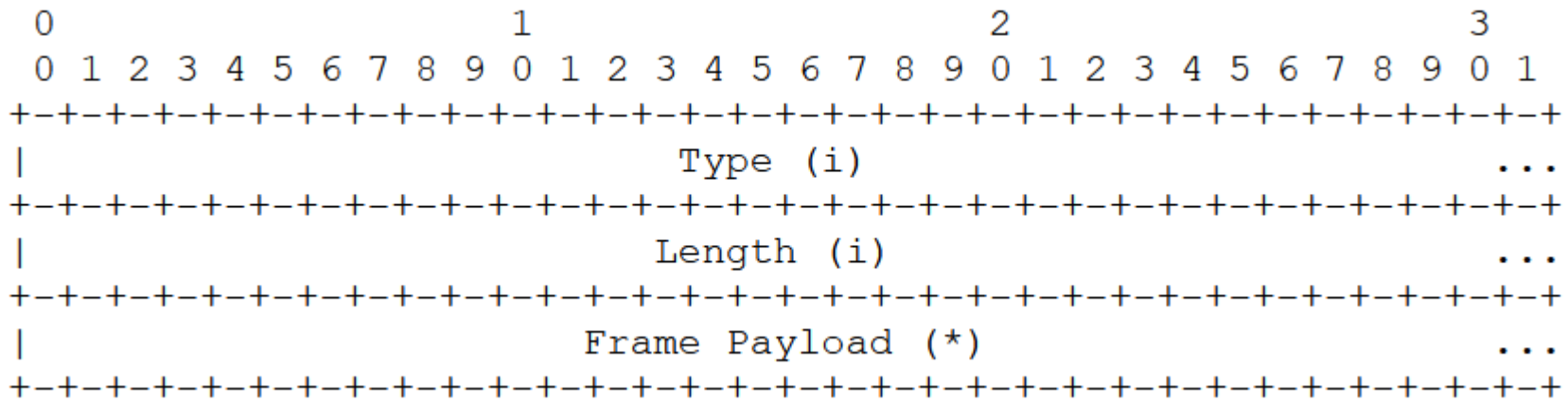
Stream Frame

- **FrameType低3位（0x08 – 0x0f）**
  - OFF bit
    - 是否有Offset
  - Len bit
    - 是否有Len
  - FIN bit
- **Offset**
- **Length**



基于QUIC Stream的HTTP3 Frame

- Type
  - 0x00：DATA
  - 0x01：HEADERS
  - 0x03：CANCEL_PUSH
  - 0x04：SETTINGS
  - 0x05：PUSH_PROMISE
  - 0x07：GOAWAY
  - 0x0d：MAX_PUSH_ID
- Length
- Payload



HTTP3协议的概念与细节

- HTTP2协议遗留了哪些问题?
- HTTP3如何在UDP协议之上实现了连接迁移?
- 通过QUIC Frame、HTTP3 Frame，如何实现多路复用?
- QPACK是怎样解决队头阻塞问题的?
- HTTP3如何处理丢包问题?

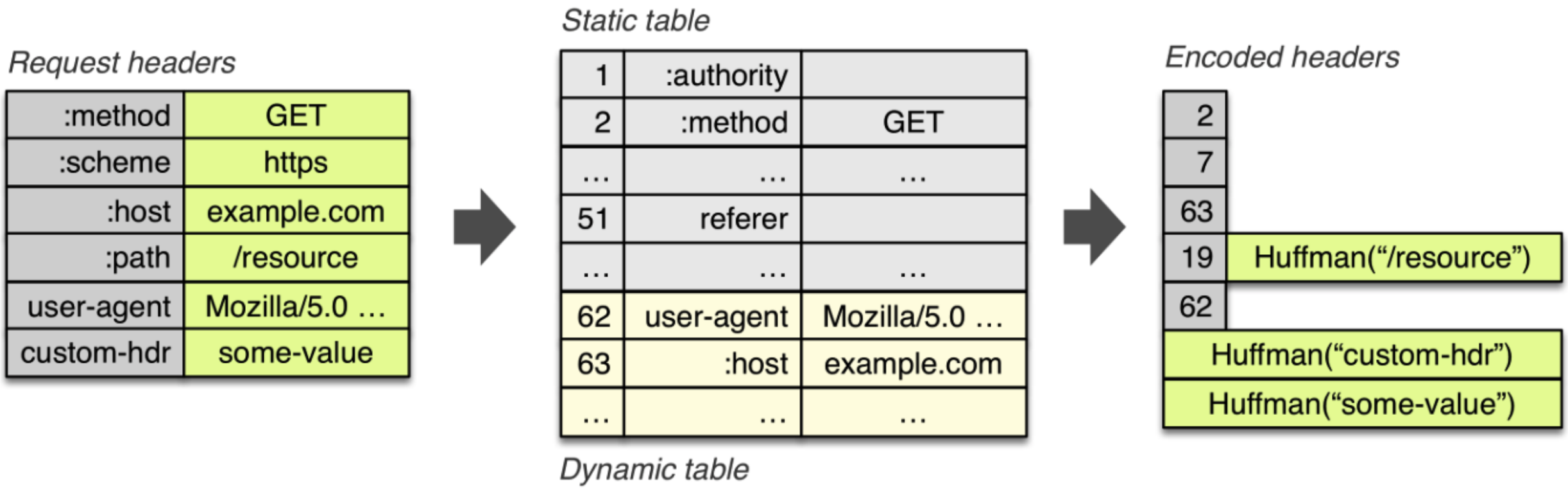
QPACK 静态表

- 静态表项数
  - HPACK：61项
    - ngx_http_v2_static_table
  - QPACK：98项
    - ngx_http_v3_static_table
      - https://tools.ietf.org/html/draft-ietf-quic-qpack

Index	Header Name	Header Value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
...	...	...
32	cookie	
...	...	...
60	via	
61	www-authenticate	

HPACK

- 动态表的队头阻塞问题



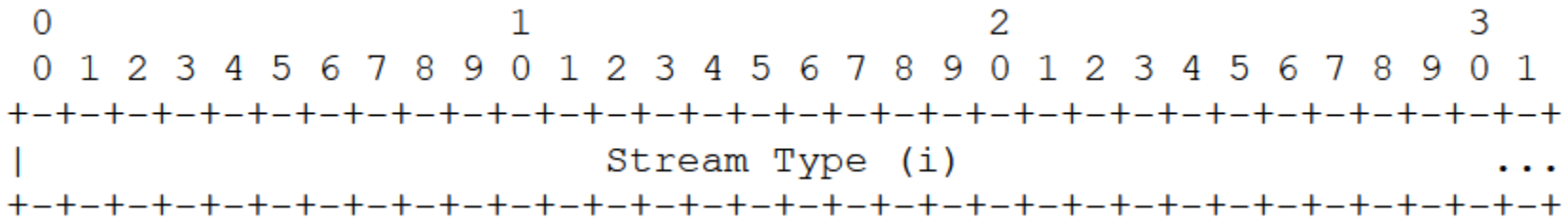
Stream ID的类型

- 最低位
  - 0：Client
  - 1：Server
- 倒数第2位
  - 0：双向
  - 1：单向
- 例如：客户端建立的双向Stream
  - 0，4，8，12，…

Bits	Stream Type
0x0	Client-Initiated, Bidirectional
0x1	Server-Initiated, Bidirectional
0x2	Client-Initiated, Unidirectional
0x3	Server-Initiated, Unidirectional

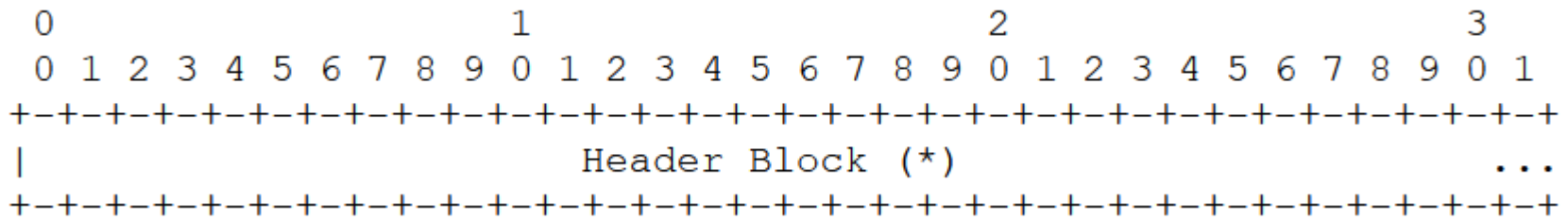
单向Stream的类型

- 0x00：Control Stream
- 0x01：Push Stream
- 0x02：QPACK Encoder
- 0x03：QPACK Decoder



动态表的更新

- QPACK动态表
  - FIFO时序依赖 VS STREAM间的完全无序
  - 每一端都有至多1个以下Stream
    - encoder 单向Stream：Type 0x02
    - decoder 单向Stream：Type 0x03
- HTTP3 HEADERS Frame

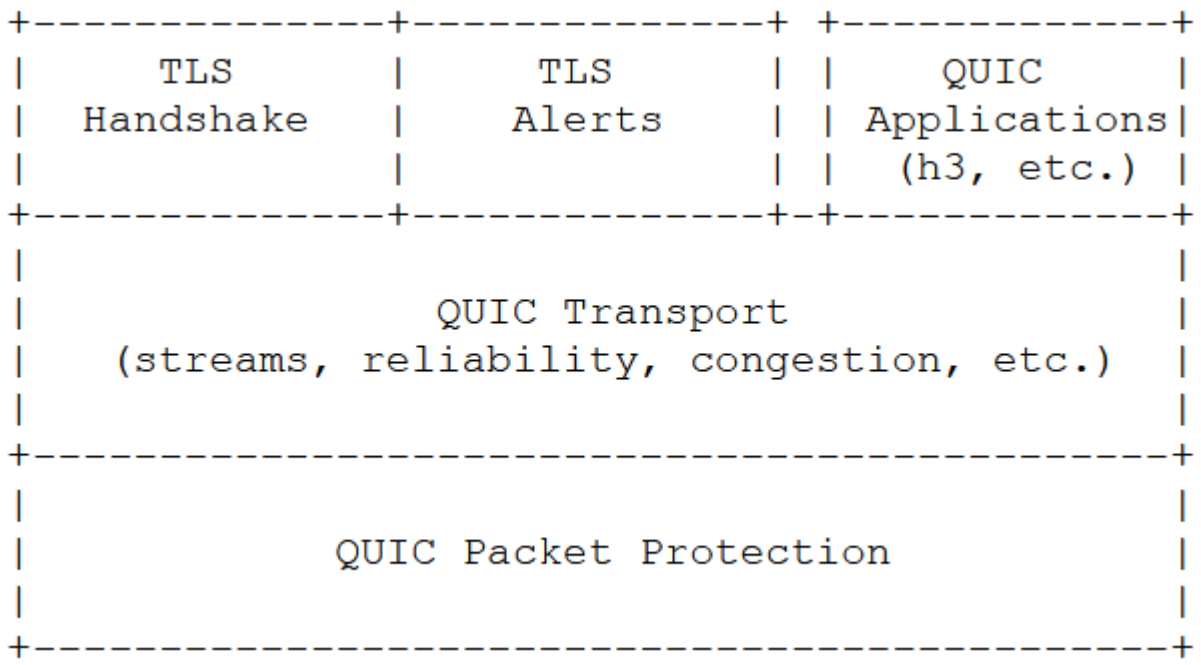


HTTP3协议的概念与细节

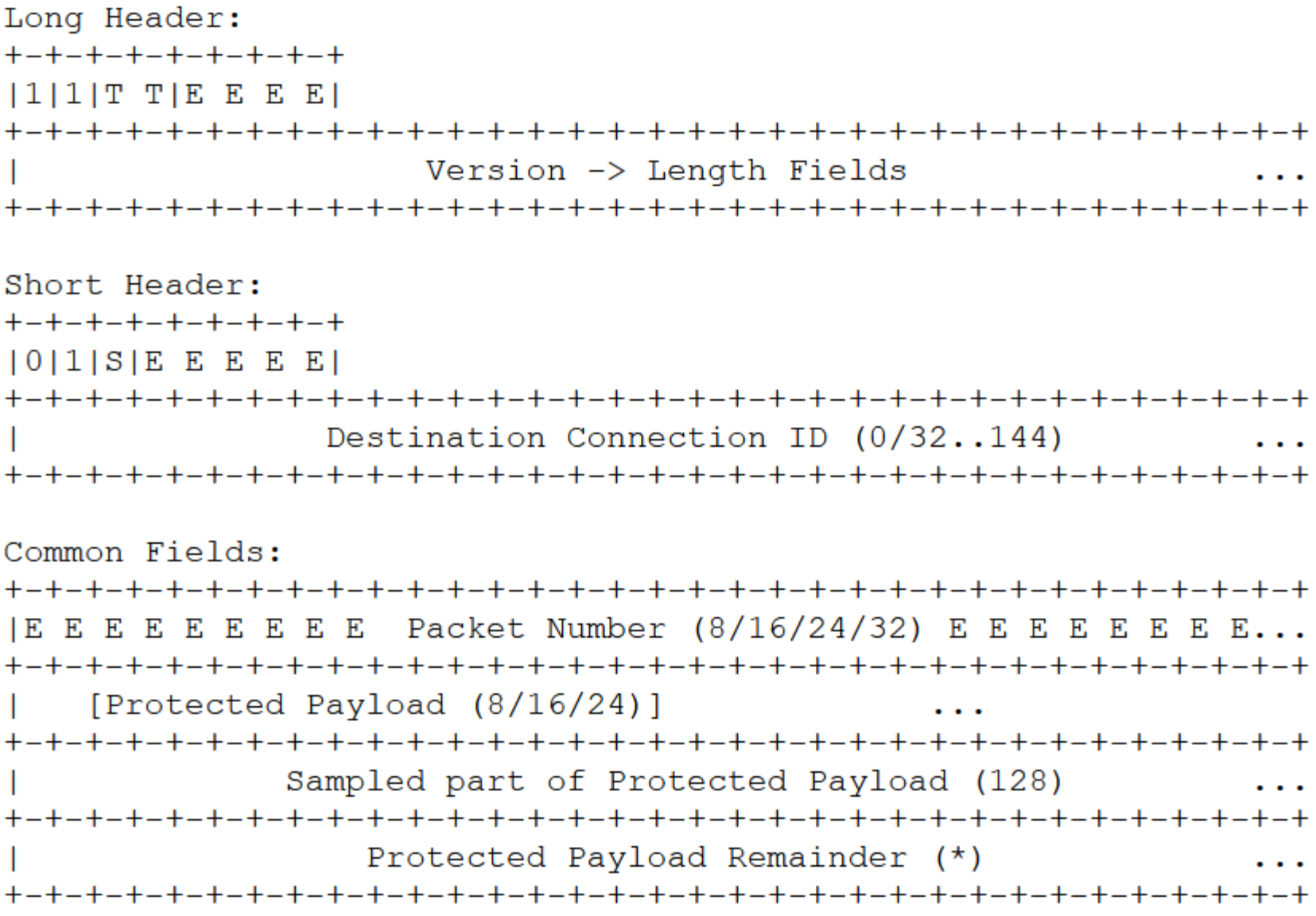
- HTTP2协议遗留了哪些问题?
- HTTP3如何在UDP协议之上实现了连接迁移?
- 通过QUIC Frame、HTTP3 Frame，如何实现多路复用?
- QPACK是怎样解决队头阻塞问题的?
- **HTTP3如何处理丢包问题?**

QUIC+TLS

- Packet Number等信息都是加密在TLS中的



TLS Payload

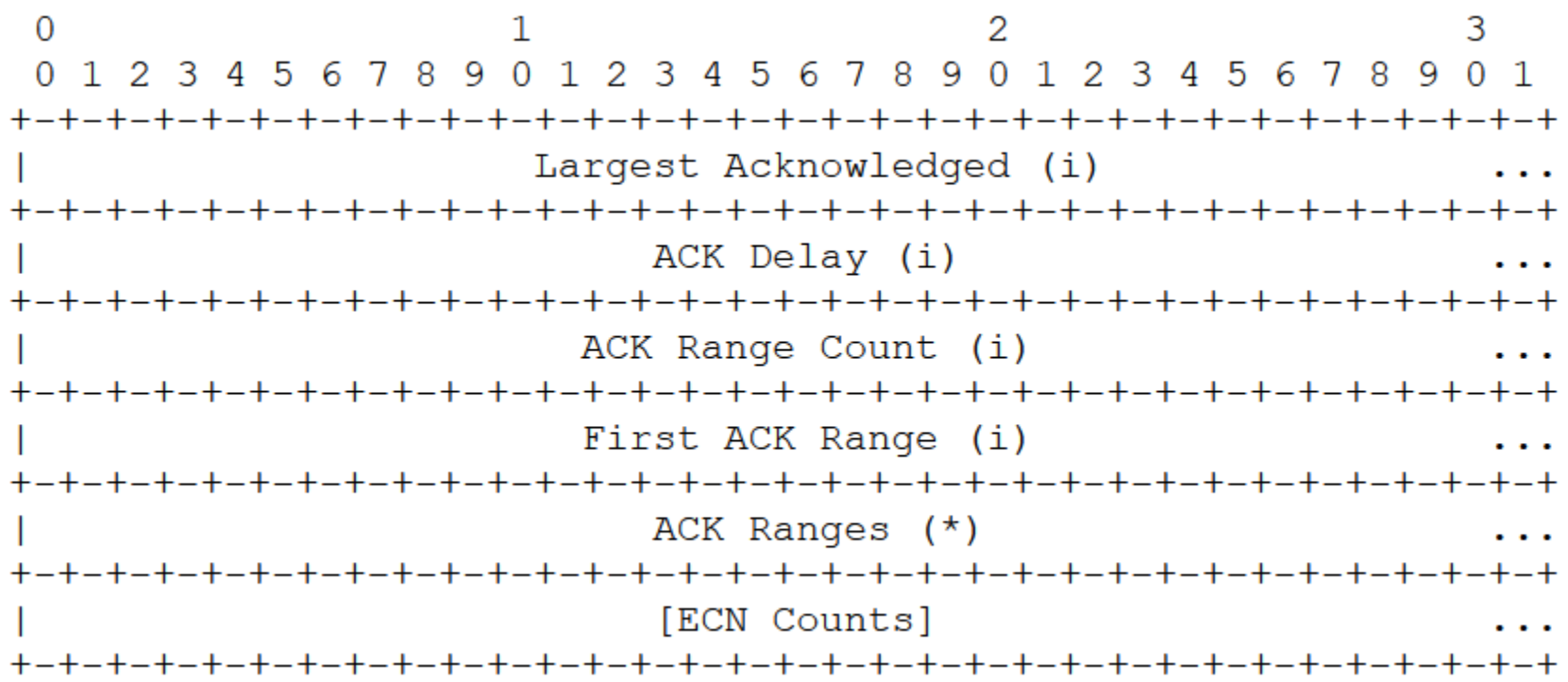

Long Header:
|1|1|T T|E E E E|
Version -> Length Fields
Short Header:
|0|1|S|E E E E E|
Destination Connection ID (0/32..144)
Common Fields:
|E E E E E E E E E E Packet Number (8/16/24/32) E E E E E E E E...
[Protected Payload (8/16/24)]
Sampled part of Protected Payload (128)
Protected Payload Remainder (*)

丢包处理机制

- 基于Packet Number
- 确认与重传
  - RTT测量
- 拥塞控制：Reno算法
  - 慢启动
  - 拥塞避免
  - 快速重传
  - 快速恢复

Ack Frame

- Largest Ack
  - 接收方最大Packet Number
- ACK Delay
  - 发送ACK与接收到最大已确认Packet的时间差，单位毫秒
- ACK Range Count
- ACK Range



目录

使用Nginx搭建HTTP3 Web Server

- **编译Nginx**
  - 获取官方quic分支源代码
  - 编译google boringssl：master分支
  - 编译Nginx: --with-http_v3_module
- **运行Nginx**
  - 各配置指令的含义
- **测试Nginx**
  - chrome：./chrome.exe --enable-quic --quic-version=h3-27
  - wireshark：3.2版本以上

目录

Nginx流控指令（1）- http{} server{}

- quic_initial_max_data
  - 连接上飞行中的报文数量，默认为16*65536，可以由MAX_DATA帧改变
- quic_initial_max_stream_data_bidi_local
  - 双向stream本地端的初始流控值，默认65536
- quic_initial_max_stream_data_bidi_remote
  - 双向Stream远端的初始流控值，默认65536
- quic_initial_max_stream_data_uni
  - 单向Stream的初始流控值，默认65536
- quic_initial_max_streams_bidi
  - 双向Stream的最大并发数，默认16，可以由MAX_STREAM帧改变
- quic_initial_max_streams_uni
  - 单向Stream的最大并发数，默认16

Nginx QUIC 指令（2）- http{} server{}

- quic_max_idle_timeout
  - 最大空闲时间，单位毫秒，Nginx默认60000，0表示关闭功能
- quic_max_ack_delay
  - 延迟确认的最大值，默认（也是RFC推荐）25毫秒
- quic_max_packet_size
  - 默认65527，不能小于1200字节
- quic_ack_delay_exponent
  - 将ACK确认帧中，将ACK时延按2的倍数扩大。默认值3（RFC推荐），不允许超过20
- quic_active_migration
  - 是否支持客户端迁移连接，默认为1表示开启
- quic_retry
  - 防止流量放大攻击时，可以通过retry功能强制要求客户端开启TOKEN地址验证功能。默认关闭off

Nginx指令（3）- http{} server{}

- **http3_max_field_size**
  - HTTP QPACK头部的大小限制
- **http3_max_table_capacity**
  - 设置动态表容量
  - SETTINGS_QPACK_MAX_TABLE_CAPACITY帧可以修改其值
    - 与HTTP2中的SETTINGS_HEADER_TABLE_SIZE帧功能一致
- **http3_max_blocked_streams**
  - QPACK动态表会阻塞Stream，该值可以定义允许最大阻塞住的Stream数量，一旦超出，会立刻向客户端返回QPACK解压失败错误

HTTP3挑战

- 硬件调整
- 升级慢
- 中间的代理服务器无法控制
- 二等公民UDP

谢谢