


# TCP糊墙小史—— 从Tcp Fast Open 到 QUIC



1024041144

朱倍江

2024年9月26日

“ /kwɪk/ ”



# TCP?

- 面向连接的
- 字节流的
- 可靠的

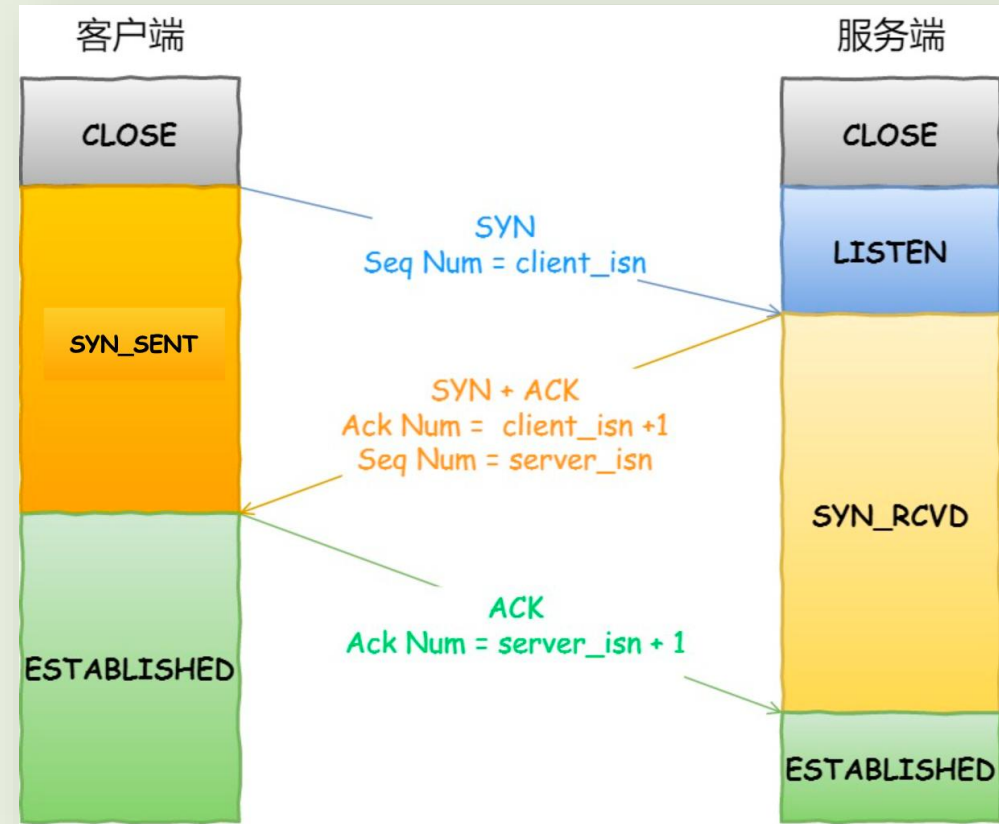
## 传输层通信协议

1974 -> 至今



# TCP 三次握手

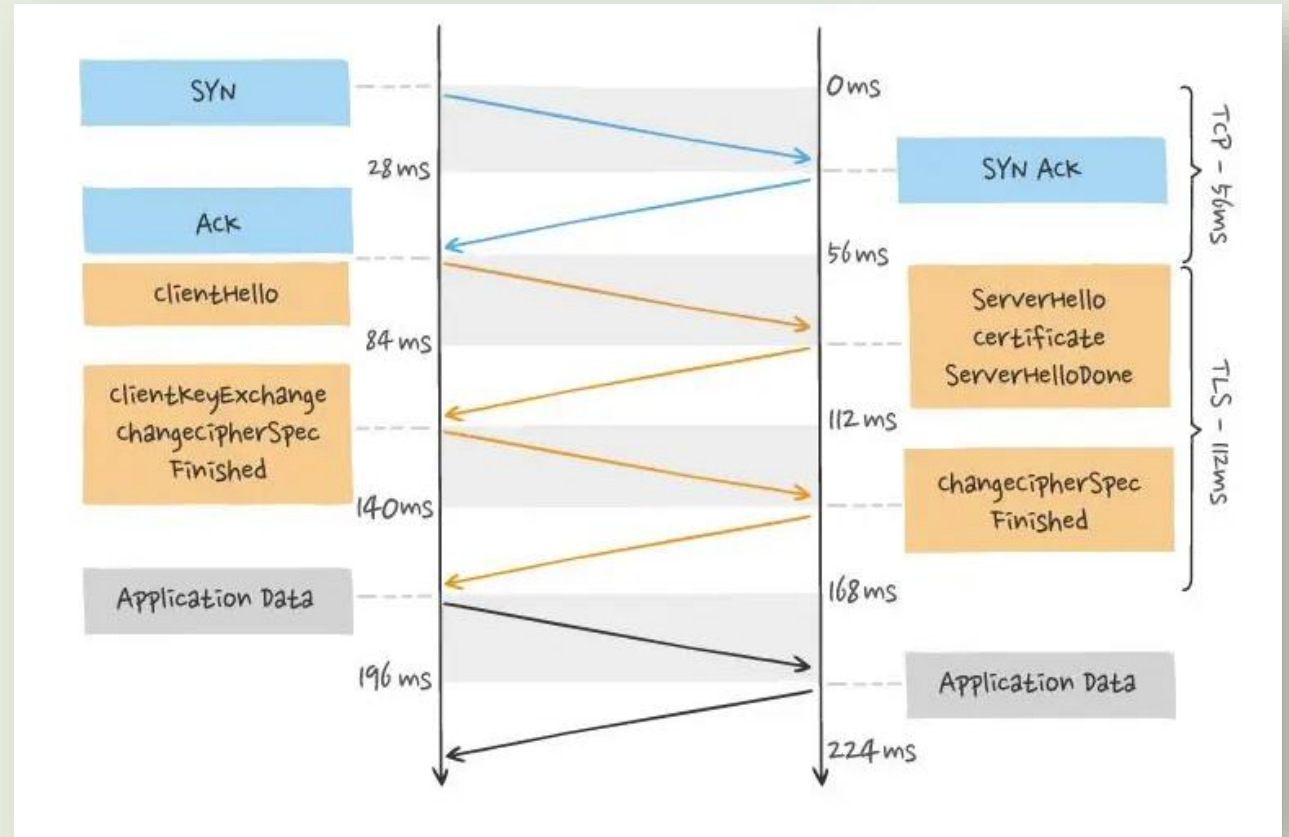
- 避免历史连接
- 同步双方初始序列号
- 减少资源浪费





# TLS 时代?

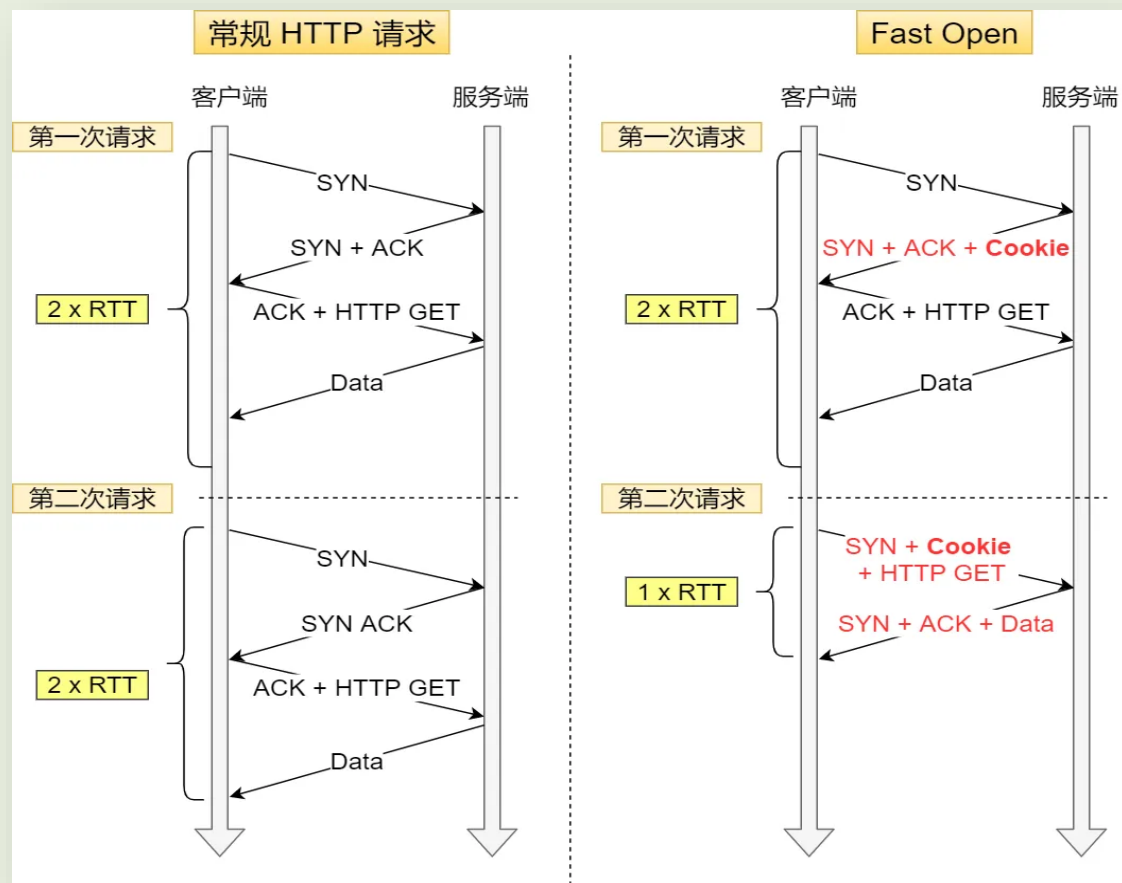
- 3次TCP握手
- 4次TLS v1.2握手（第一次与ACK同发）
- 无法加密的TCP包头
- 沉重的握手累赘





# TCP Fast Open

- 服务端加密Cookie发送给客户端
- 客户端缓存Cookie, 并在之后的请求中通过Cookie请求直接返回数据





# TCP Fast Open升级难点



在Bugzilla上见证历史：

为什么火狐 **彻底移除** TCP FastOpen支持？

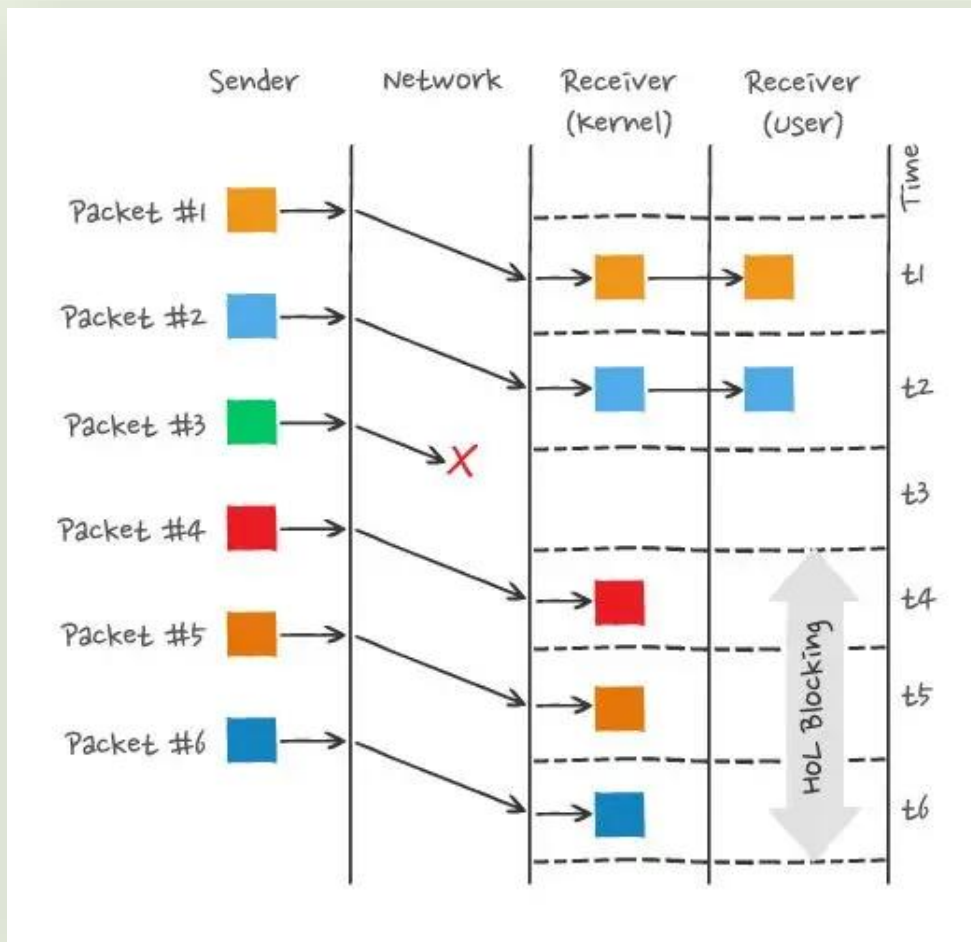
- TCP协议栈实现存在于Kernel中，甚至firmware中，升级需要考虑与其他内核功能的兼容性，需要巨量成本进行确认与测试。
- 现有设备存量巨大，且许多设备无法升级。
- Chromium默认关闭TCP FastOpen，Firefox亦然。



# 队头阻塞问题



- TCP 层必须保证收到的字节数据是完整且有序的。
- 如果序列号较低的 TCP 段在网络传输中丢失了，即使序列号较高的 TCP 段已经被接收了，应用层也无法从内核中读取到这部分数据。





# 源地址变化问题

- 源地址
- 源端口
- 目的地址
- 目的端口

## TCP四元组

确定唯一连接





# 源地址变化问题

- 通勤中：Wifi——基站网络——Wifi
- 公司里：Wifi AP1——Wifi AP2——Wifi AP3
- 高铁上：数秒钟变化一个基站
- 每次移动变化一个源地址，每次四元组变化需要重新建立连接，由于不可靠数据必须重传，用户必须动不动卡一下



# Why not UDP?

“字字一面”

## 如何基于 UDP 实现可靠传输？



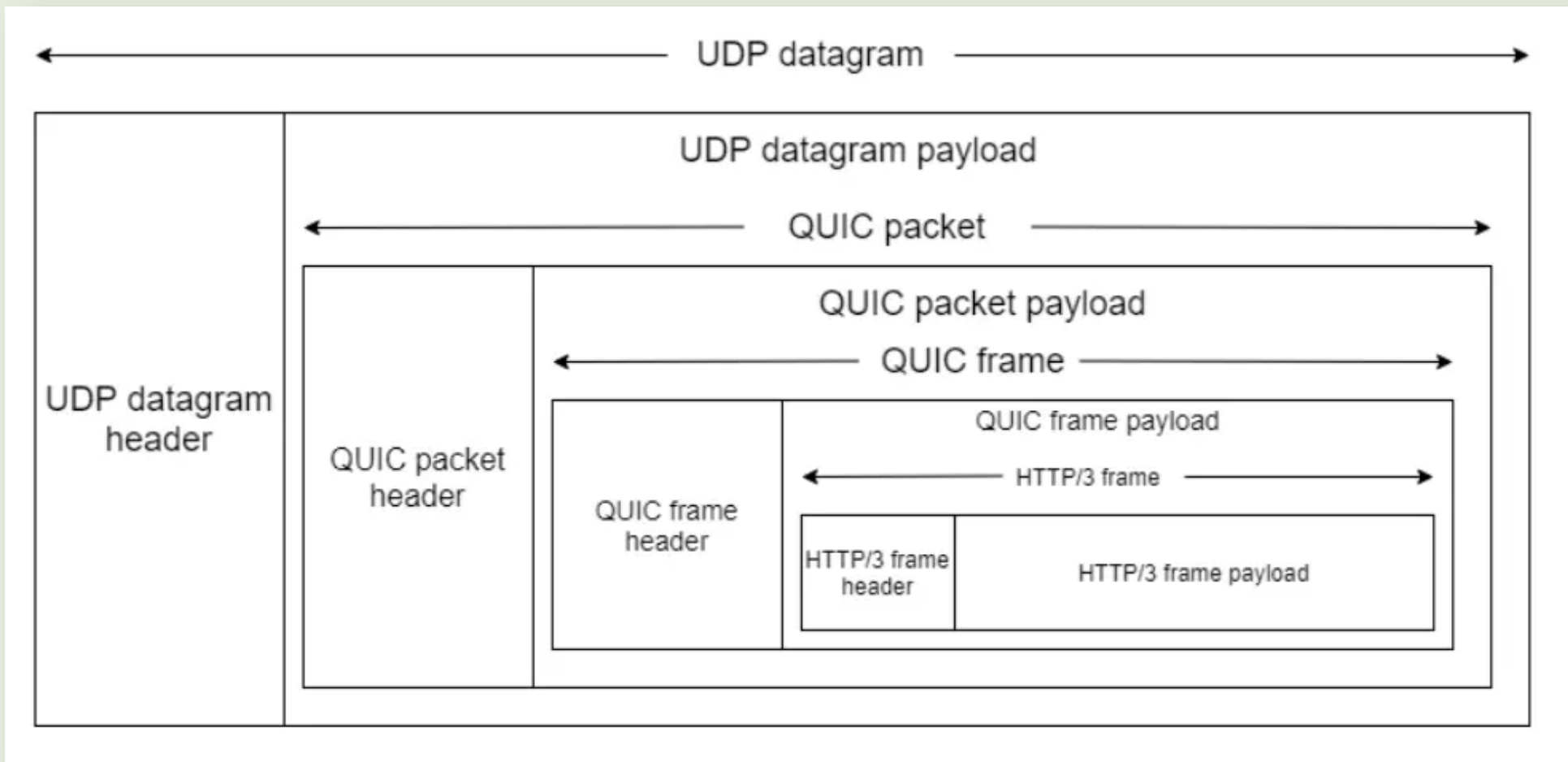
# QUIC

- 解决TCP的痛点：
  - TCP 升级工作困难
  - TLS时代 复杂握手延迟
  - TCP 队头阻塞问题
  - 移动互联网时代 源地址变化问题

HTTP/3 采用  
QUIC + TLS 1.3

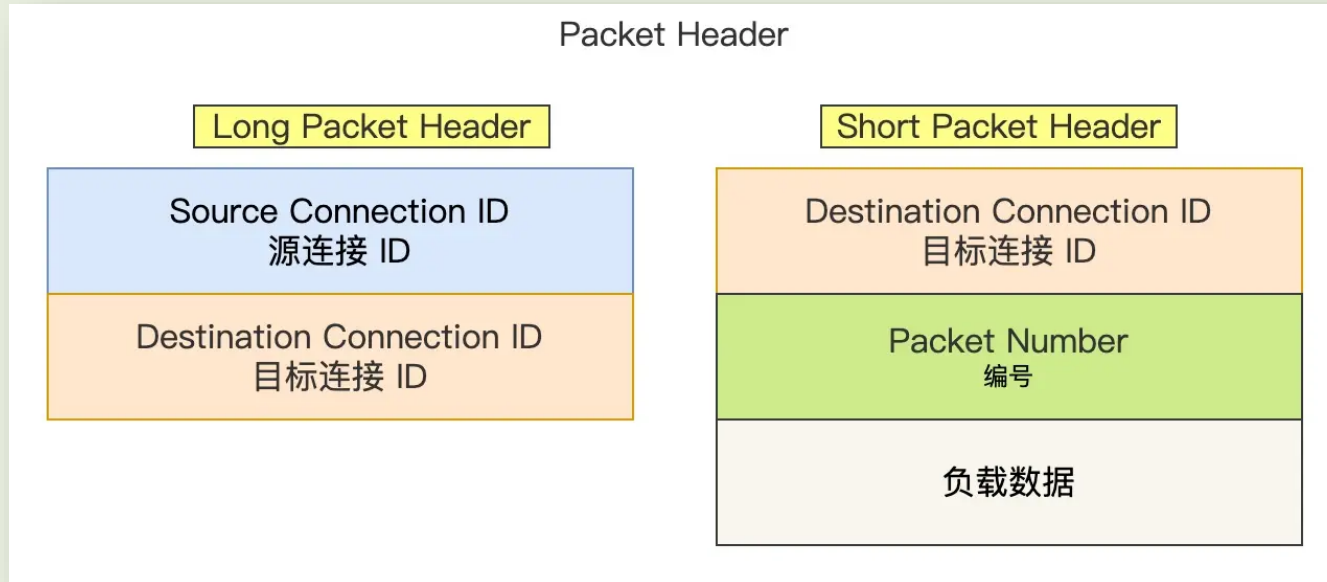


# QUIC 结构





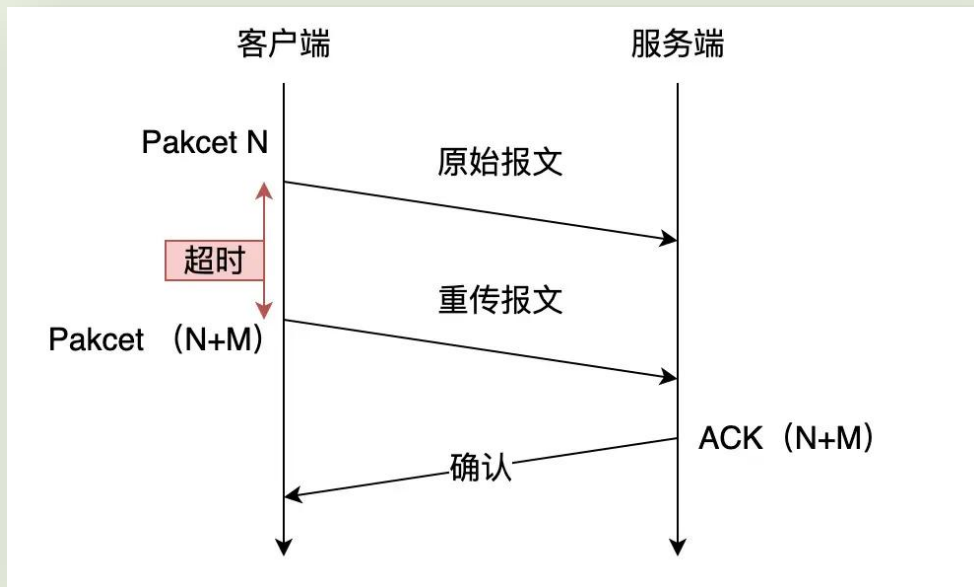
# Packet Header



- 首次连接：长包头，建立握手ID
- 日常连接：短包头，固定ID，实现不同源地址连接迁移



# Short Packet Header

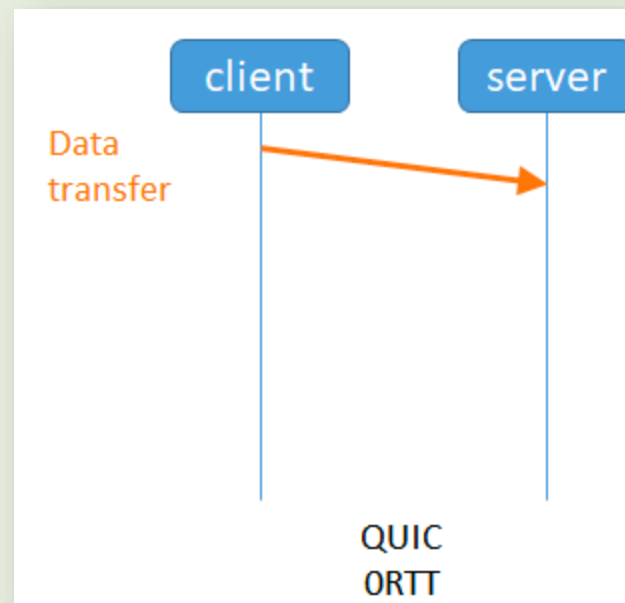
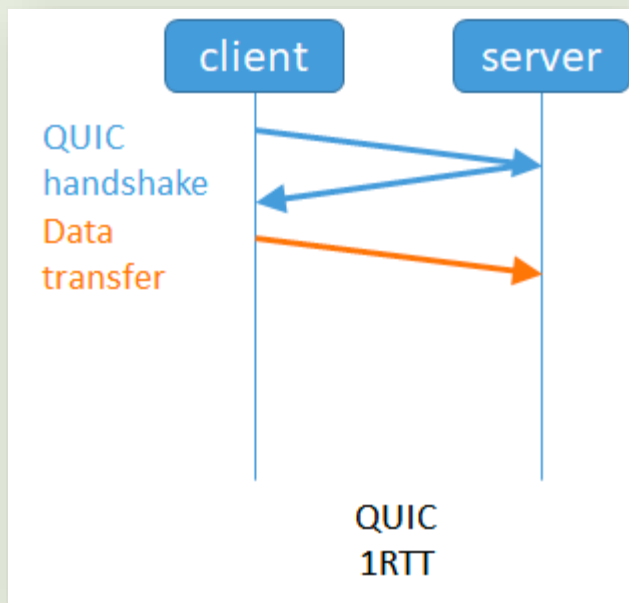


- 每个报文独一无二的编号 Packet Number, 严格递增;
- 即便重传也要增加编号大小。



# 0-RTT传输

- QUIC包体中包含了TLS 1.3信息，首次连接1-RTT
- 后续连接0-RTT

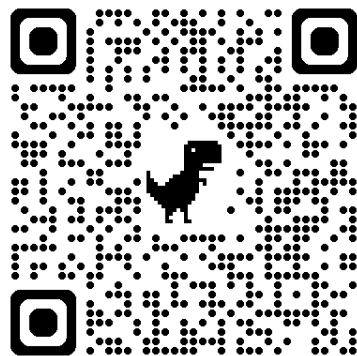




# QUIC更新

- 拥塞控制算法应用层实现，操作系统无关
- 跟随浏览器更新





扫码更多了解QUIC



1024041144  
朱倍江  
2024年9月26日

“Thanks”