下一代Internet技术与协议

张冬梅
zhangdm@bupt.edu.cn

QUIC

主要内容

- 概述
- 主要原理
- 进展及应用

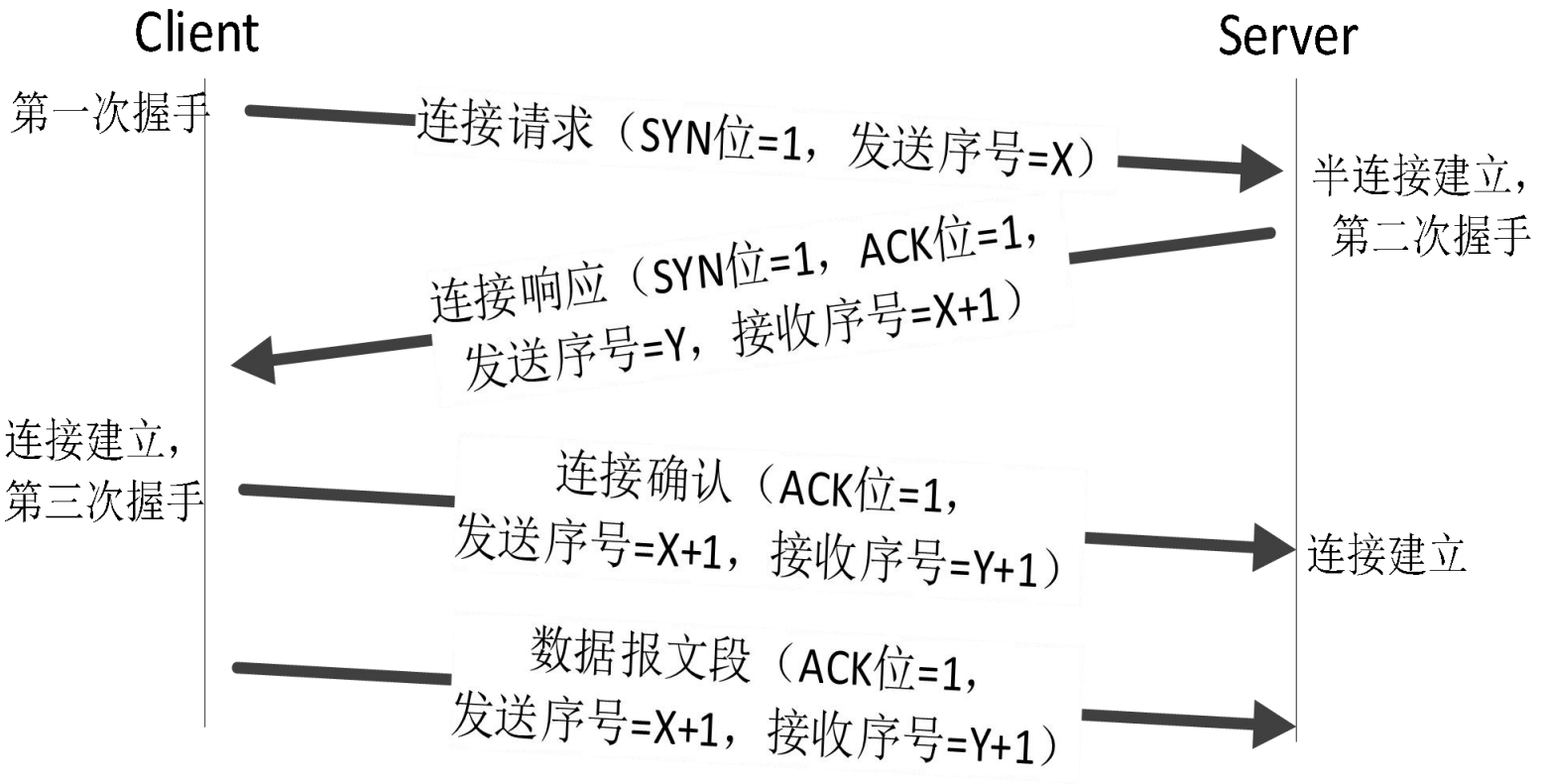
QUIC概述

- Quick UDP Internet Connection（快速UDP互联网连接）
- 新型传输协议，在UDP之上（Server Port 443）
  - 比TCP更简单、更快速建立连接
  - 安全性可与TCP+TLS匹敌
- 由Google开发，2013年在Chrome浏览器中实现
- 2018 年，基于 QUIC 协议的 HTTP（HTTP over QUIC）—— HTTP/3，正式被确定为下一代网络规范。

现有网络的问题

- 常用协议
  - 网络层IPv4/IPv6：提供路由选择和IP包转发功能
  - 传输层TCP：提供可靠的面向连接的进程-进程通信
  - 传输层SSL/TLS ：提供传输安全
  - 应用层DNS： 进行域名解析，域名→IP地址
  - 应用层HTTP/HTTPS：实现最广泛的应用——Web
- 存在的主要问题
  - 协议历史悠久，导致中间设备僵化；
  - 依赖于操作系统的实现，导致协议本身僵化；
  - 建立连接，导致端到端延迟大
  - 队头阻塞

TCP的连接建立：三次握手



UDP的通信过程

- 无需连接，Client直接发送数据给Server
- 报头简单：无序号
- 功能：基于端口号的复用、无连接不可靠

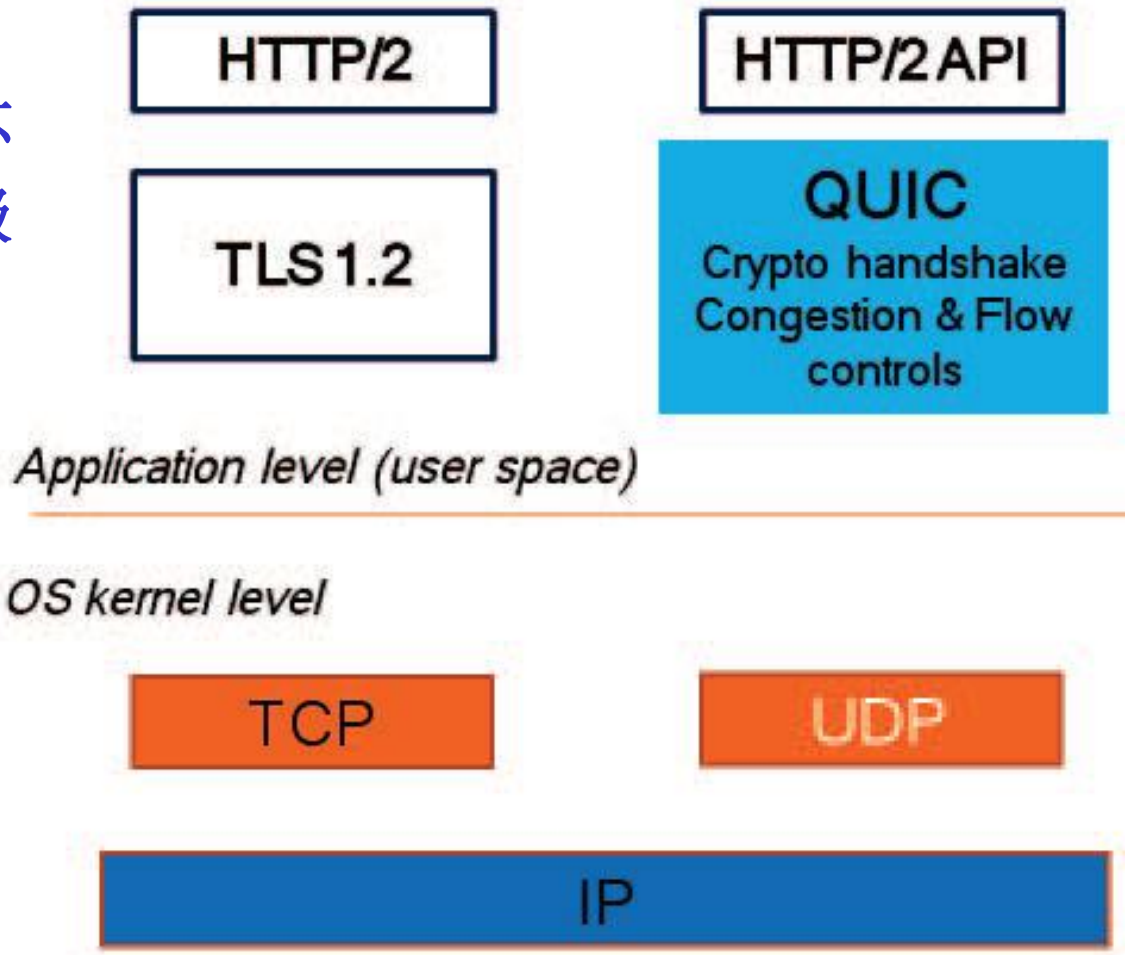
队头阻塞

- Head of Line blocking
- TCP确保将数据按顺序上交应用层，在出现数据报文段丢失时，后续数据必须缓存，等待丢失的报文端重传到达
- 每个 TCP 连接一个只能处理一个请求-响应，浏览器按 FIFO 原则处理请求；如果上一个请求的响应没返回，后续请求-响应都会受阻。
- Web用户感知：页面加载时间长，体验差

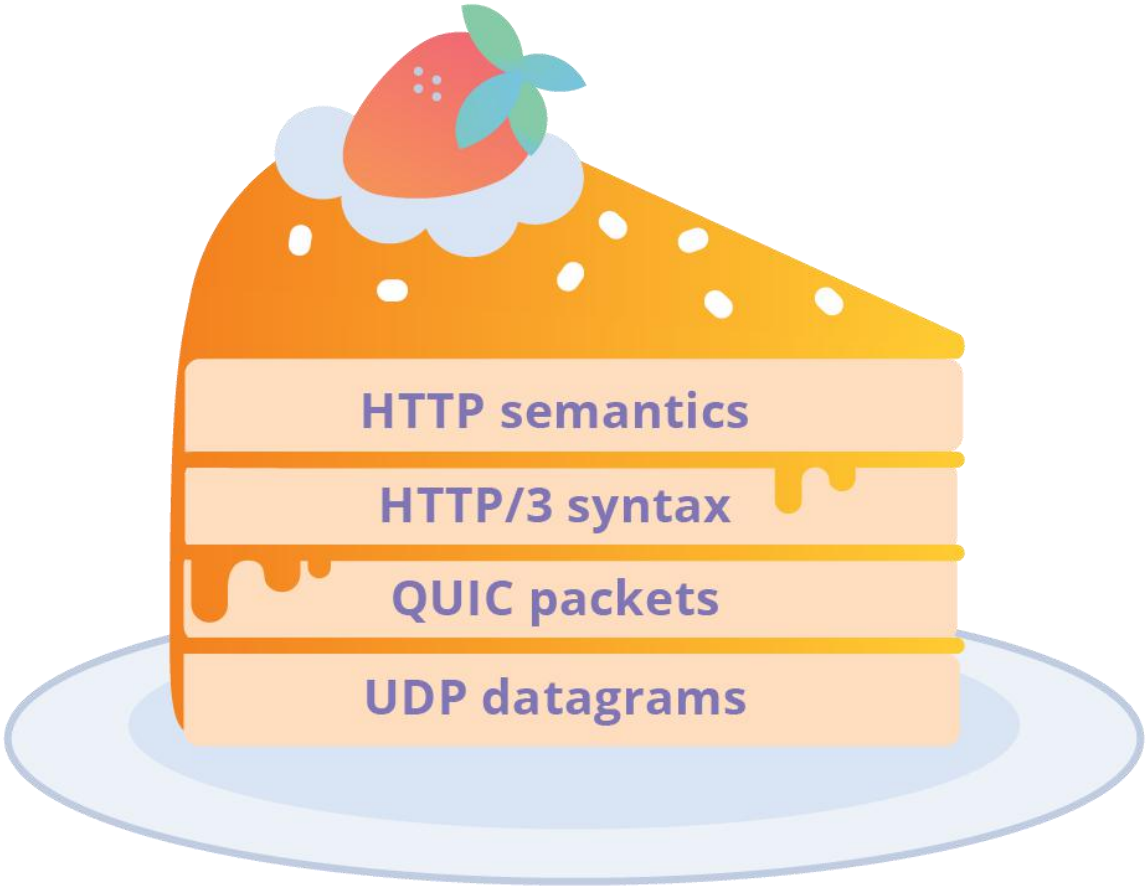
QUIC：用户级协议

QUIC在用户级实现，而不是系统内核级

→无需改动**OS**和中间设备， 容易实现和 部署



HTTP/3的分层



相关术语

- QUIC连接：Client和Server之间的通信关系，Client发起连接，Server接受连接
- 流(Stream)：一个QUIC连接内，单向或双向的有序字节流
  - 一个QUIC连接可以同时包含多个Stream
- 帧(Frame)：QUIC连接内的最小通信单元
  - 一个QUIC数据包（packet）中的数据部分包含一个或多个帧

QUIC的主要功能

- 流复用
- 连接建立低时延
- 灵活的拥塞控制
- 流级和连接级流量控制
- 连接迁移
- 数据包头和包内数据的身份认证和加密

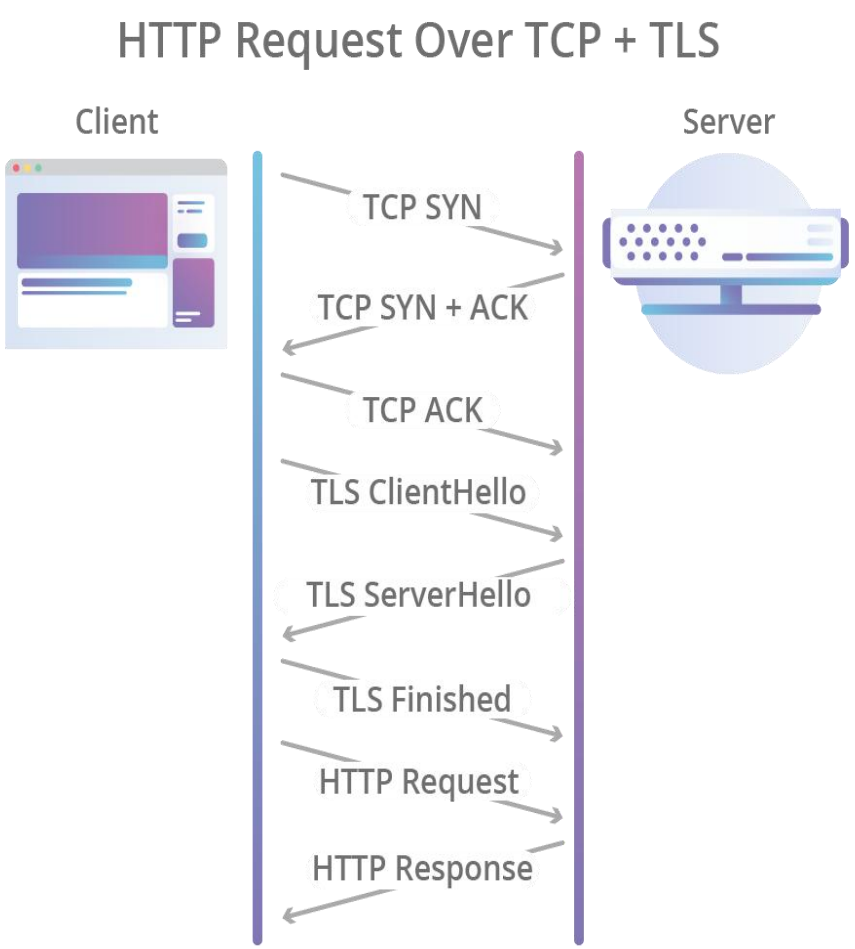
HTTP1.1

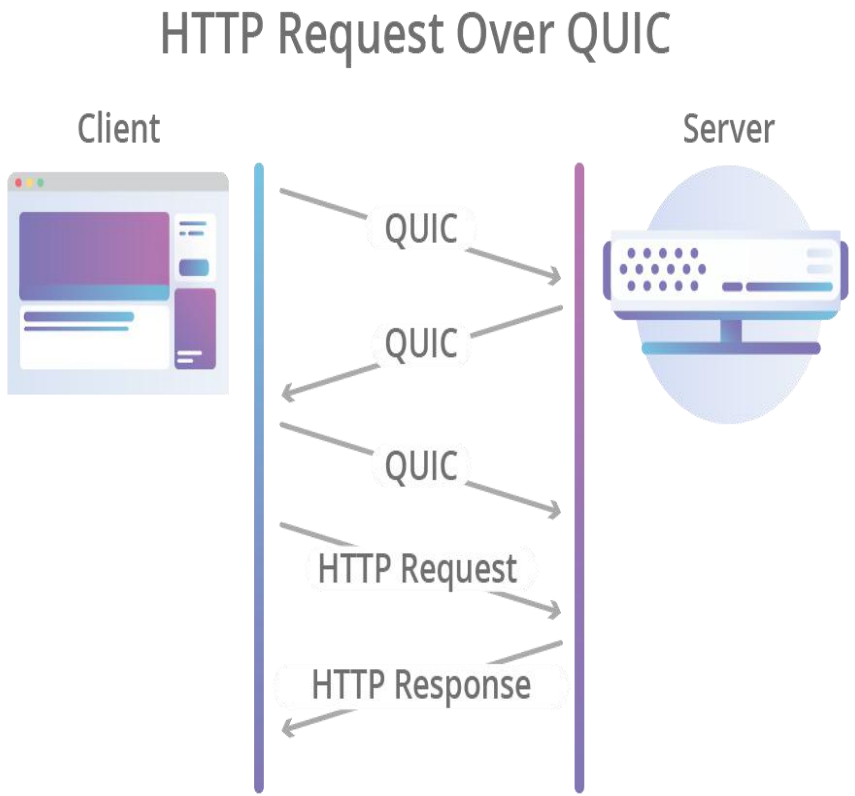
- 1999年发布
- 基于TCP，三次握手建立连接
- 每个 TCP 连接同时只能处理一个请求 - 响应，为提高响应速度，需要同时创建多个连接。
- TCP 协议头部没有经过任何加密和认证，在传输过程中很容易被中间网络设备窃听和篡改
- HTTP1.1 一发一收，也就是第一个数据没响应之前不能发第二个请求。

HTTP/2

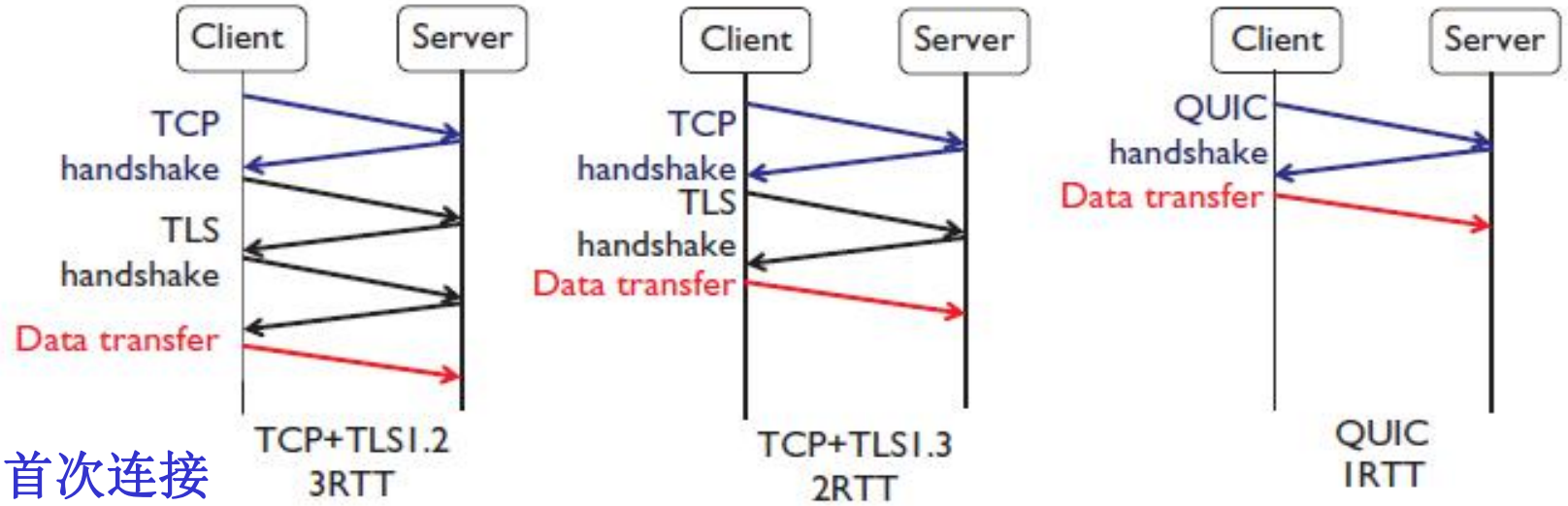
- HTTP/2：2015年2月，IETF标准 [RFC 7540]
- 优点：
  - 流(Stream)复用：通过同一个TCP连接发送多个逻辑数据流，实现了单连接之上的多个请求/响应的复用
  - 帧：以二进制传输代替HTTP1.1的明文传输，HTTP1.1报文消息被划分为更小的数据帧
    - 帧头包含“流标识符”字段，接收方会把具有相同流标识符的帧重新组装成完整消息报文
- 缺点：
  - 队头阻塞，在包丢失时，同一连接上的后续包被阻塞（即使属于不同流），用户体验较差
  - 建立TCP连接需要3个RTT（其中2个RTT用于TLS）

HTTP over TCP+TLS vs. HTTP over QUIC





QUIC：连接建立（1）

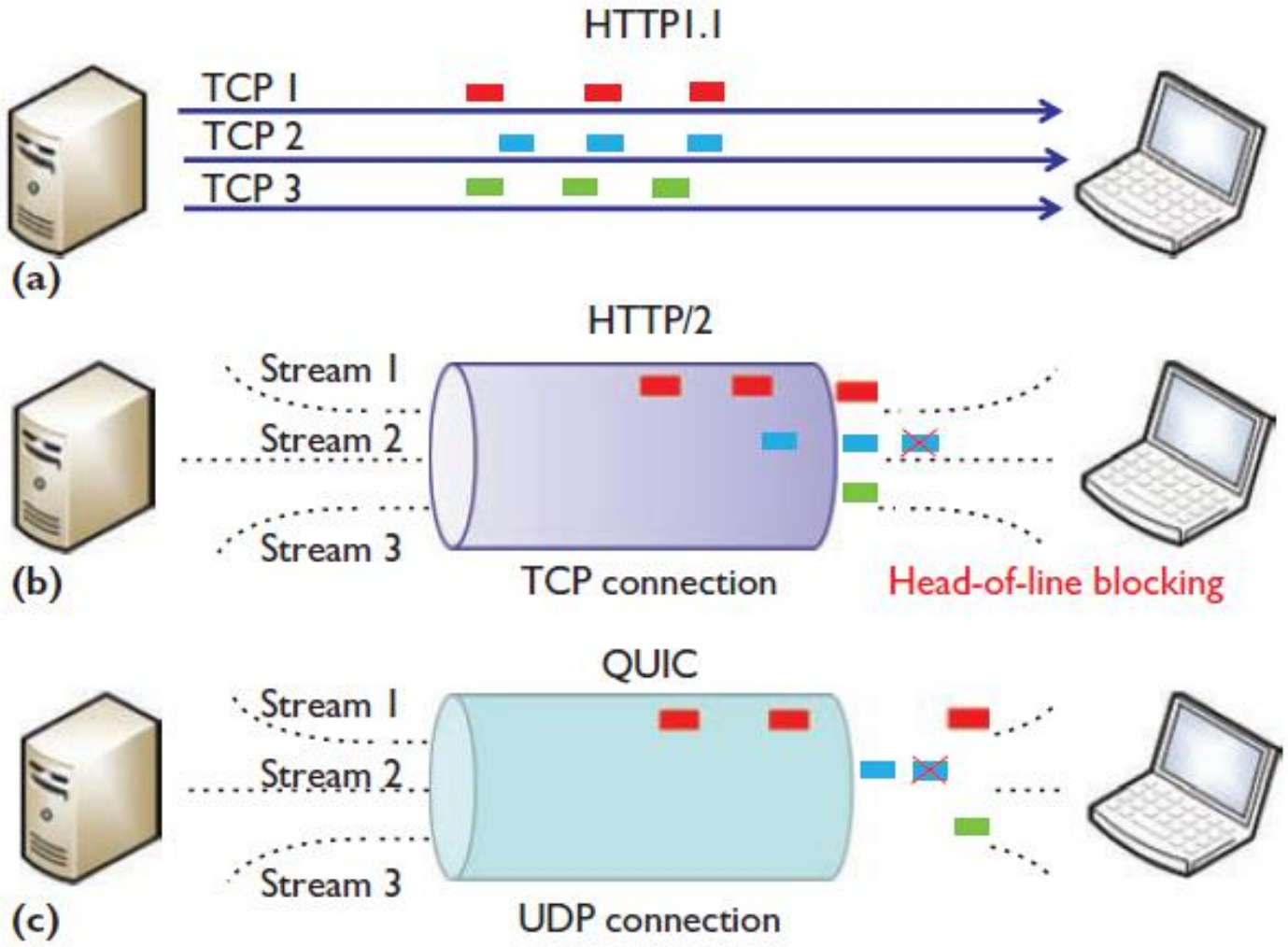


首次连接

QUIC：连接建立（2）

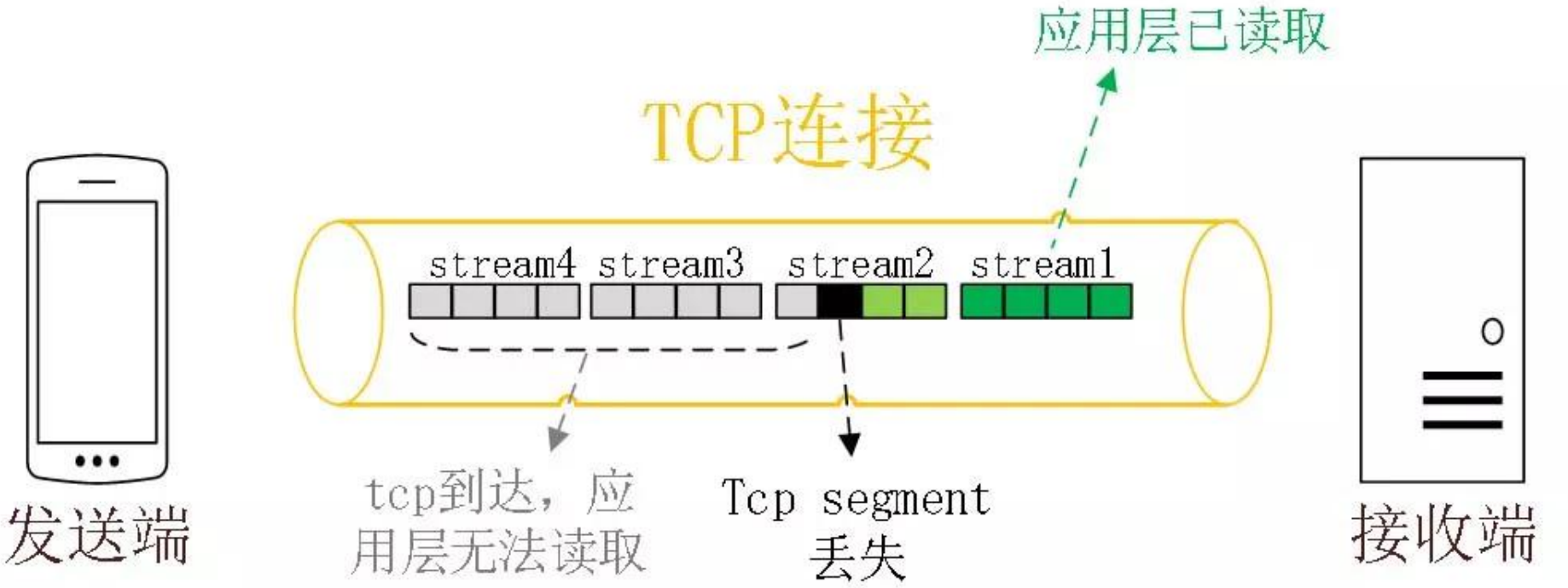
- 使用Diffie-Hellman算法协商初始密钥
- 首次连接：1RTT
  - Client向Server发送消息，请求传输配置参数和加密相关参数
  - Server回复其配置参数
- 再次连接：0 RTT
  - Client本地已有Server的全部配置参数，据此计算出初始密钥，直接发送加密的数据包

QUIC：复用


HTTP1.1
TCP 1
TCP 2
TCP 3
(a)
HTTP/2
Stream 1
Stream 2
Stream 3
TCP connection
Head-of-line blocking
(b)
QUIC
Stream 1
Stream 2
Stream 3
UDP connection
(c)

HTTP2的队头阻塞问题

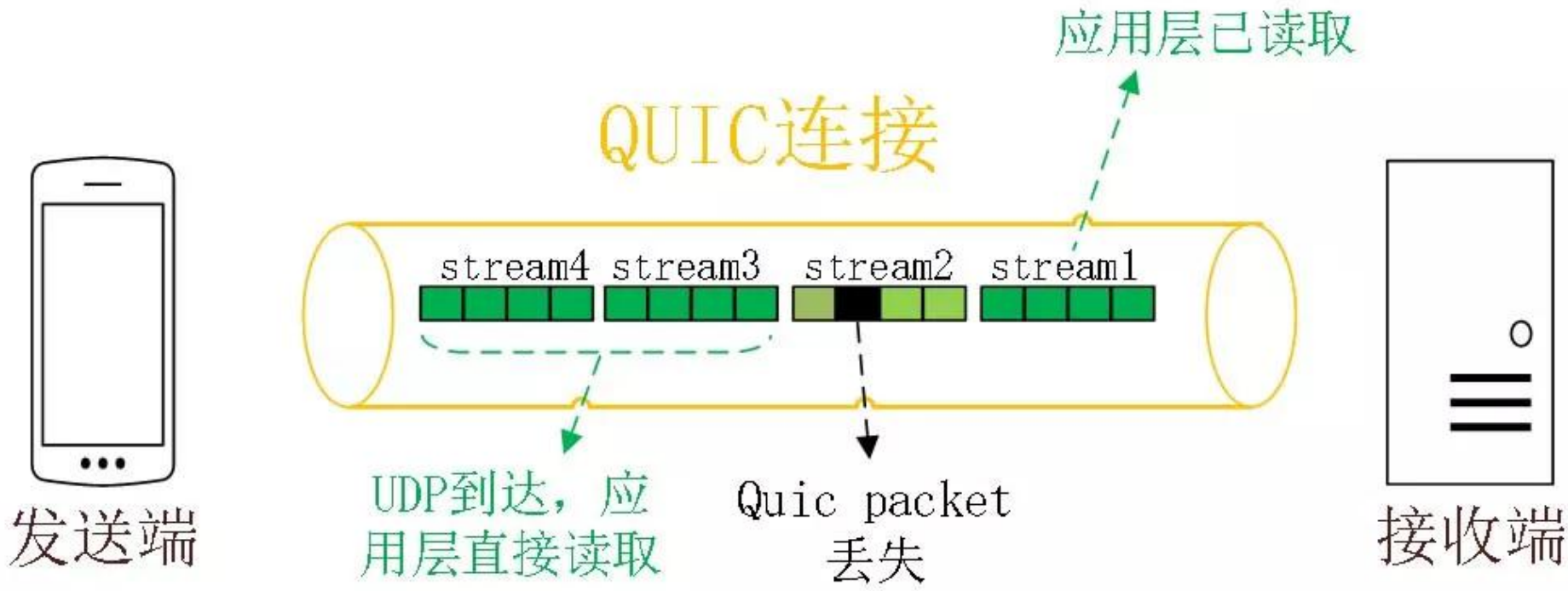
- 一个流的数据丢失会阻塞整个TCP连接



- TLS也有队头阻塞问题

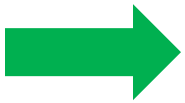
QUIC：无队头阻塞

- UDP的各个数据报独立
- QUIC的各个流独立



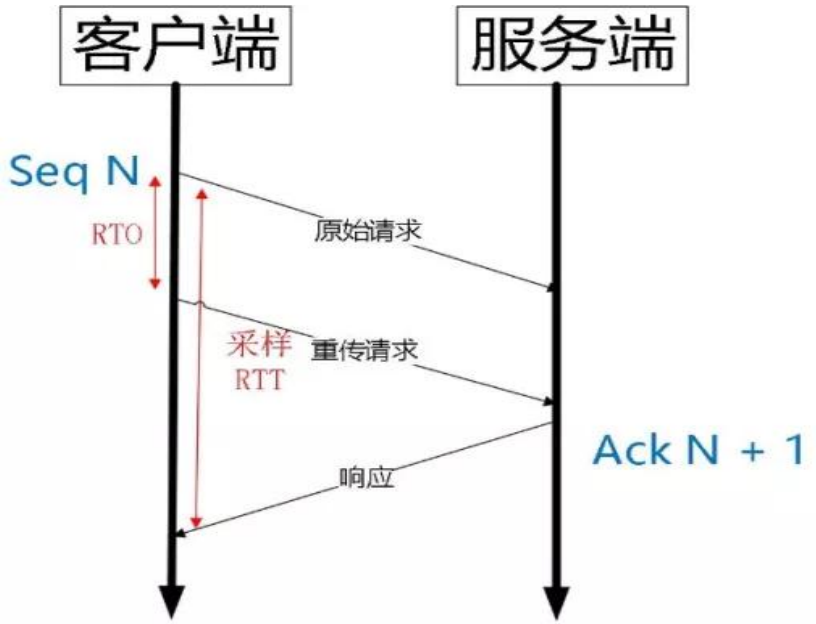
QUIC：拥塞控制

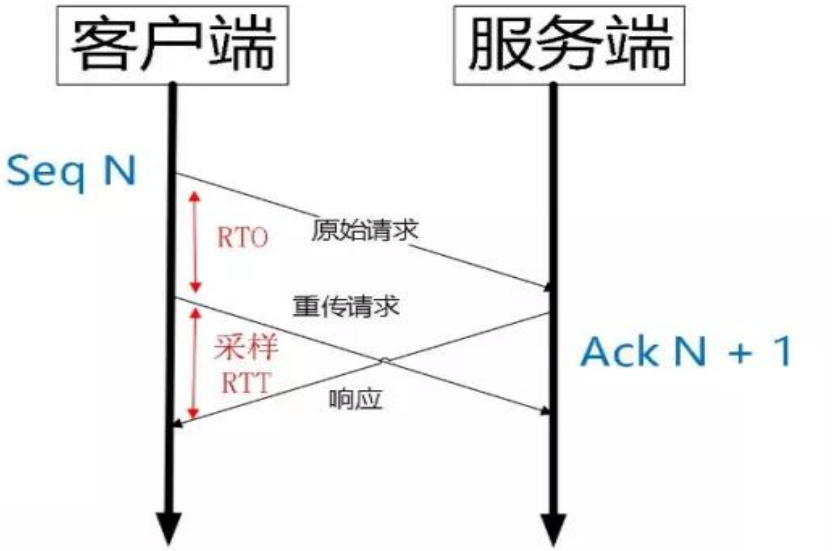
- 拥塞控制算法：默认使用CUBIC
  - 可插拔：可根据场景，切换选择不同方法，如CubicBytes、Reno、RenoBytes、BBR、PCC 等
- 重传的数据包的号码≠ 原数据包的号码
  - 解决了重传歧义问题
- ACK携带了收到数据包和发送确认之间的时延
  - 源主机估算RTT更准确
- 数据包重排序更灵活
  - NACK替代了TCP SACK



优于TCP的拥塞控制

TCP：重传歧义问题

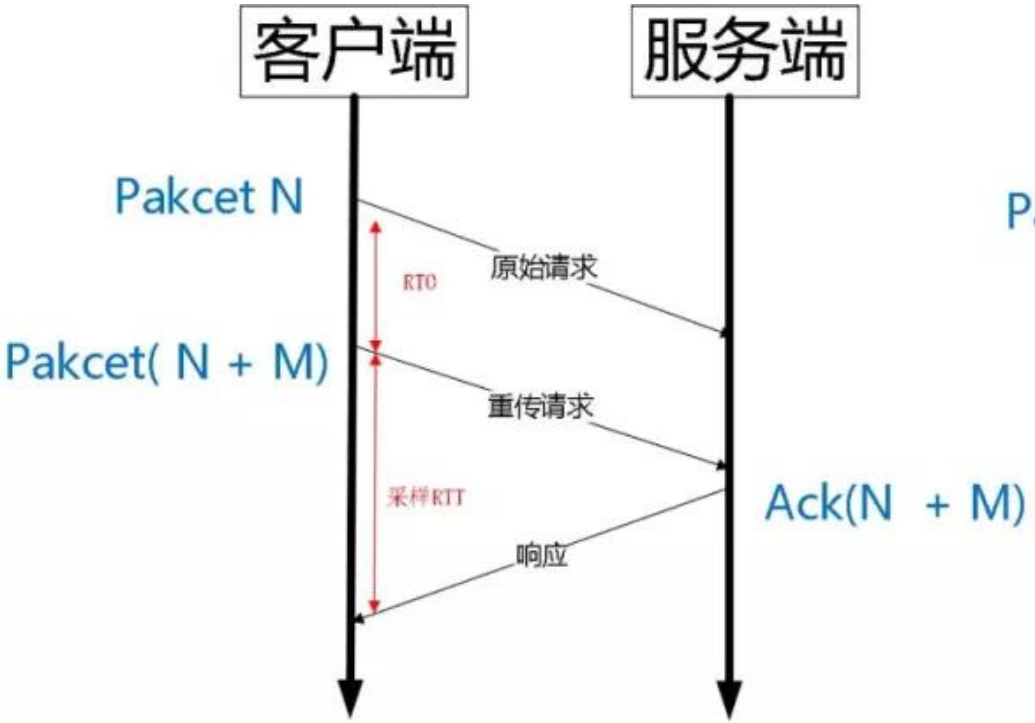


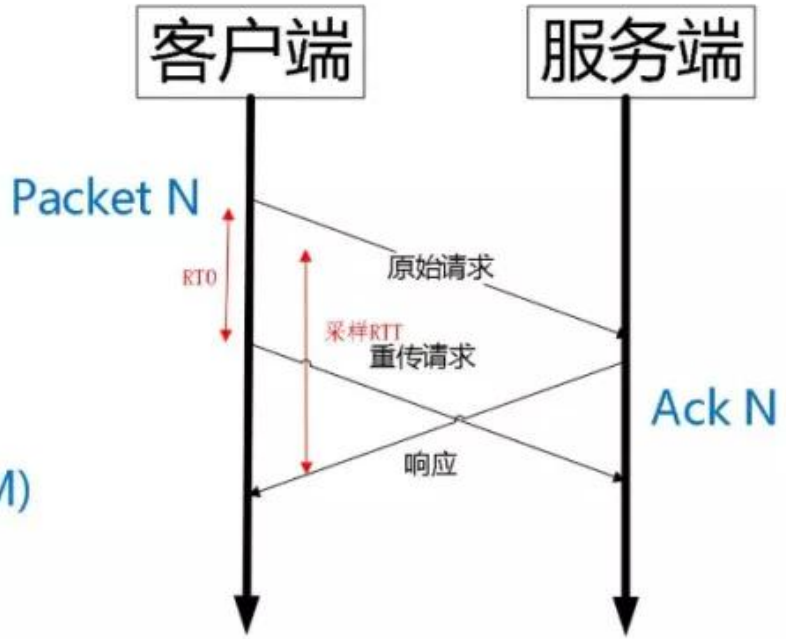


- ACK是响应原始请求（左图）还是重传请求（右图）？
- 采样RTT=？

QUIC：无重传歧义

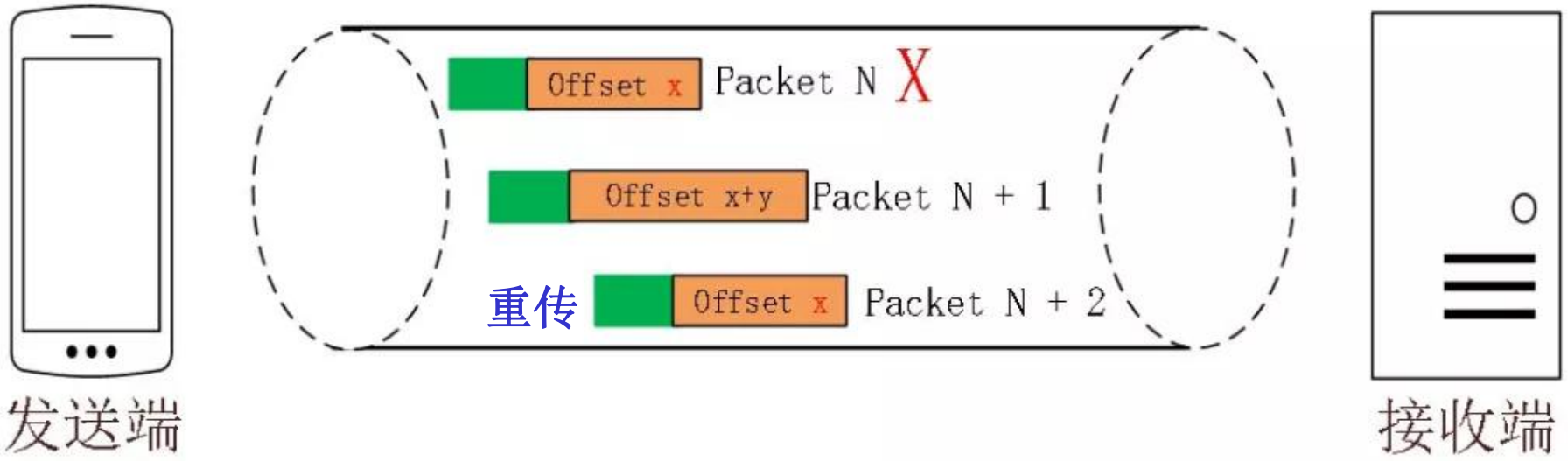
- 重传数据包号码与原始数据包号码不同
- 采样RTT估算更准确





重传数据的顺序化

- 通过数据包中的流偏移量（Stream offset）来排序

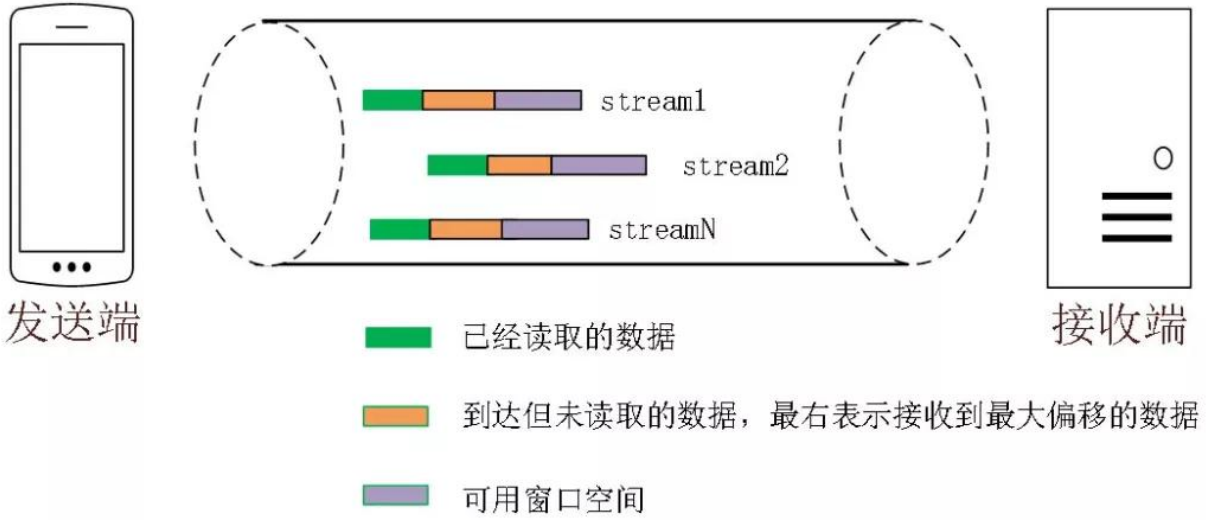


差错控制

- 基于异或（XOR）的FEC（前向纠错）
- 每个数据包除本身数据外，会带有其他数据包的部分数据，少量丢包时，可以使用其他数据包的冗余数据完成数据组装而无需重传，从而提高数据的传输速度。

流量控制

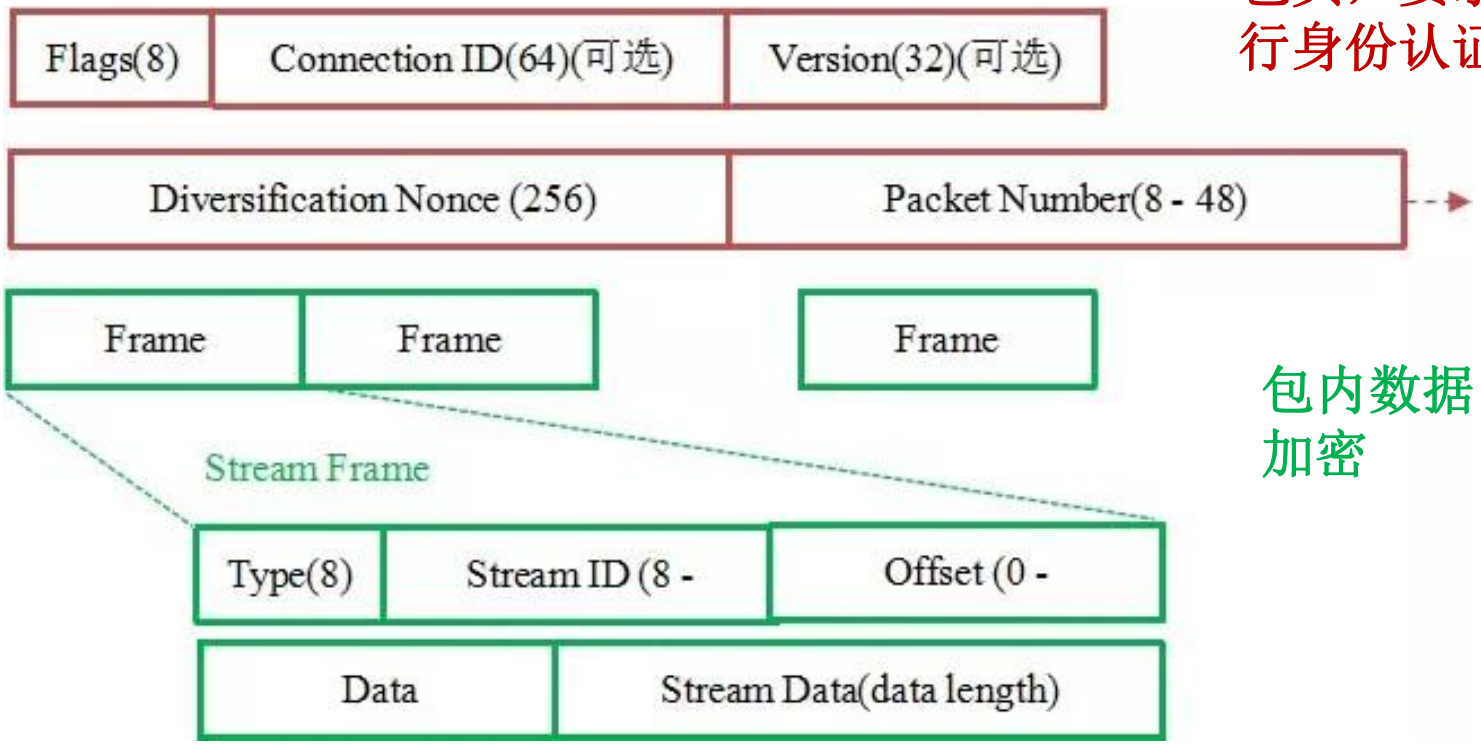
- 两级流量控制（类似HTTP/2)
  - 连接级流量控制：针对整个连接
  - 流级流量控制，防止一个流消耗过多缓存，而导致其他流堵塞



内置的安全性

- 恶意环境下性能与TLS类似，友好环境下优于TLS
  - TLS使用一个会话密钥，而QUIC使用两个密钥：初始密钥和会话密钥
  - QUIC提供密码保护，而TLS不提供
- QUIC的包头经过身份认证，包内数据加密
  - 接收端可以发现恶意修改，降低了安全风险

QUIC的包格式



包头，要求进行身份认证

包内数据：加密

连接迁移

- TCP 连接标识：四元组（源 IP，源端口，目的 IP，目的端口）
  - Client在 WIFI 和 4G 移动网络切换时，IP 地址会发生变化，需要重新建立和Server的TCP 连接
- QUIC 连接以一个 64 位的随机数作为 ID (Connection ID）
  - IP地址或者端口号发生变化时，只要 ID 不变，依然能够维持原有连接，上层业务逻辑感知不到变化，不会中断

QUIC的标准化进程

- Active Internet Drafts(10)
  - Hypertext Transfer Protocol Version 3 (HTTP/3)
  - QUIC: A UDP-Based Multiplexed and Secure Transport
  - Applicability of the QUIC Transport Protocol
  - QPACK: Header Compression for HTTP/3
  - QUIC Loss Detection and Congestion Control
  - …

https://datatracker.ietf.org/wg/quic/documents/

QUIC的部署情况

- 微软将开源自己的内部 QUIC 库 —— MsQuic，很快将成为其大多数产品（Windows、.NET、Microsoft 365等）的一部分

谢 谢！