CS120: Computer Networks

Lecture 19. Other Topics in Transportation Layer

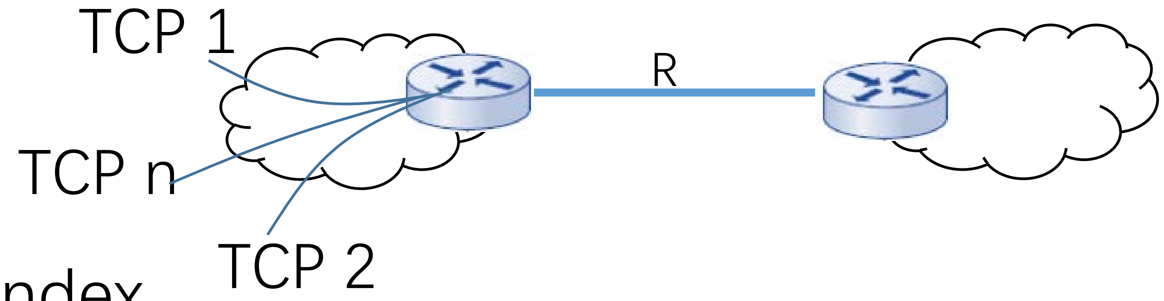
Zhice Yang

Outline

- TCP Fairness
- QUIC
- QoS

Evaluation Criteria

- Defining fairness is hard
  - In terms of a host, a TCP link, or an application ?
- TCP fairness goal: if n TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/n



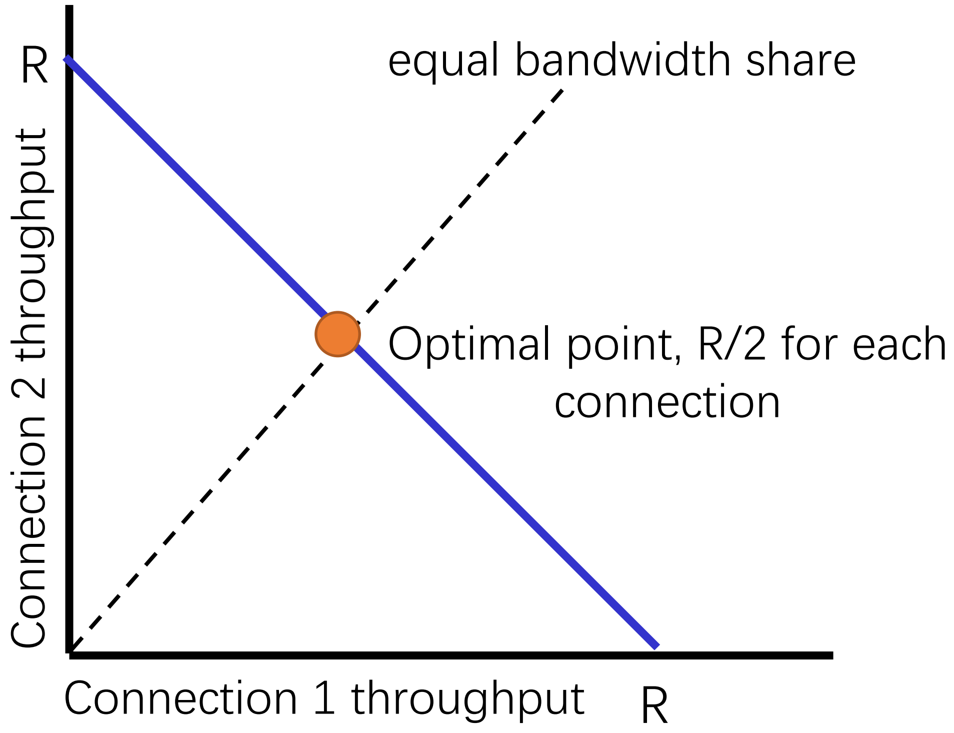
- Fairness Index
  - $f(x_1 \ldots x_n) = \frac{(\sum x_i)^2}{n * \sum x_i^2}$

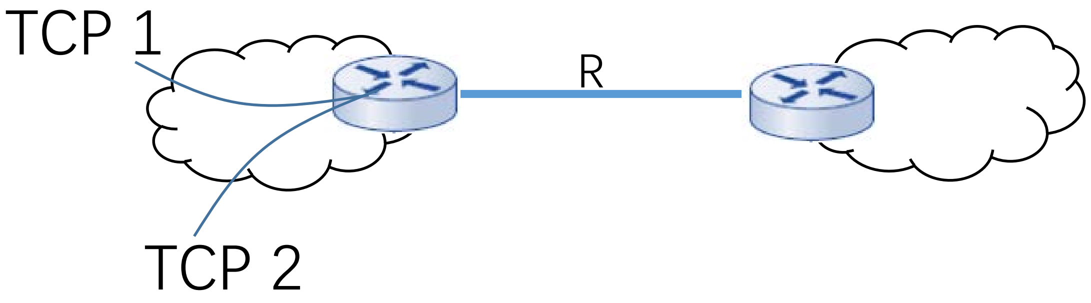
Fairness in TCP

- Consider the steady state, TCP uses a (linear) scheme to adjust its window cwnd
  - $cwnd' = b * cwnd + a$
- Possible Designs
  - Additive increase, additive decrease
  - Additive increase, multiplicative decrease (AIMD)
  - Multiplicative increase, additive decrease
  - Multiplicative increase, multiplicative decrease

Fairness in TCP

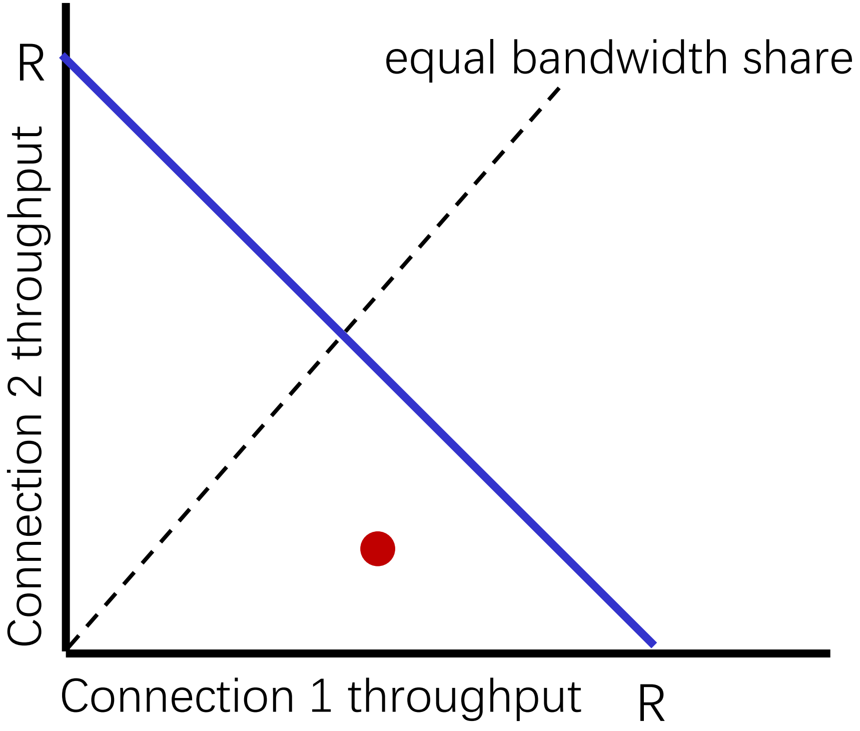
- Consider a case with two TCP connections





Fairness in TCP

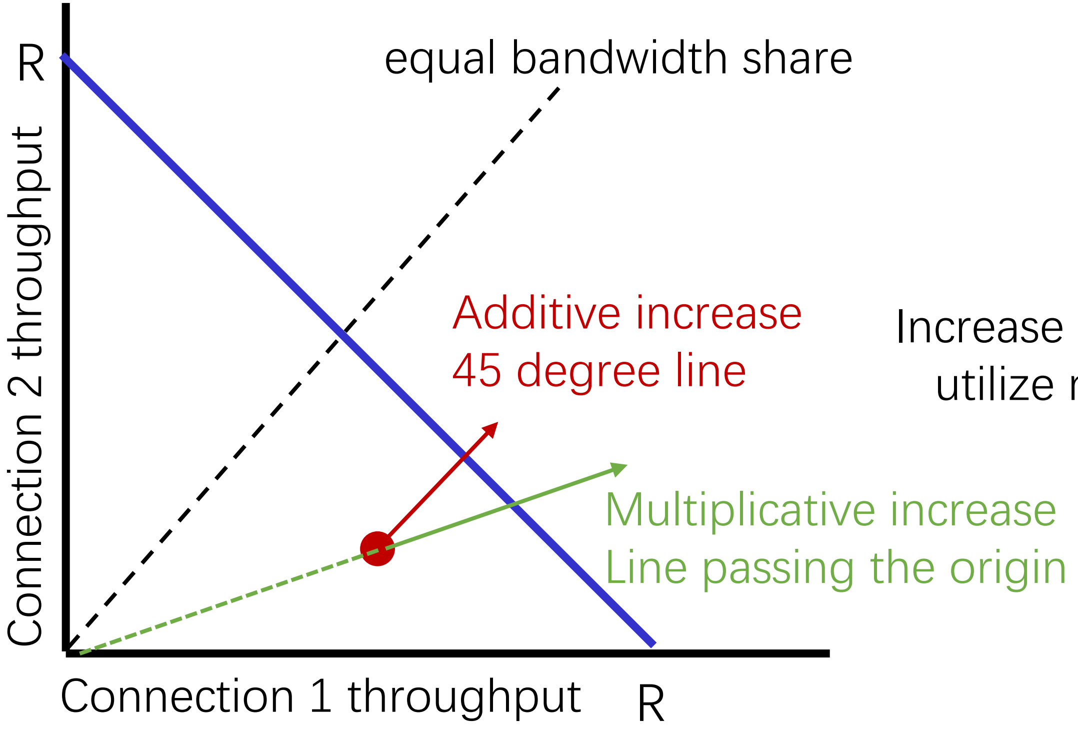
- Consider a case with two TCP connections



Increase the throughput to utilize more bandwidth

Fairness in TCP

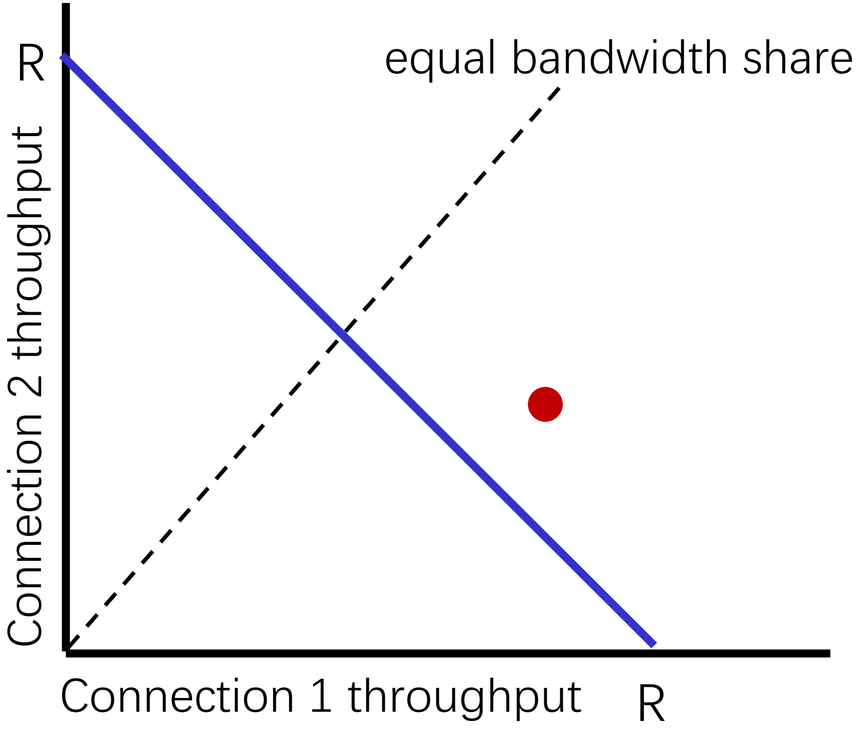
- Consider a case with two TCP connections



Increase the throughput to utilize more bandwidth

Fairness in TCP

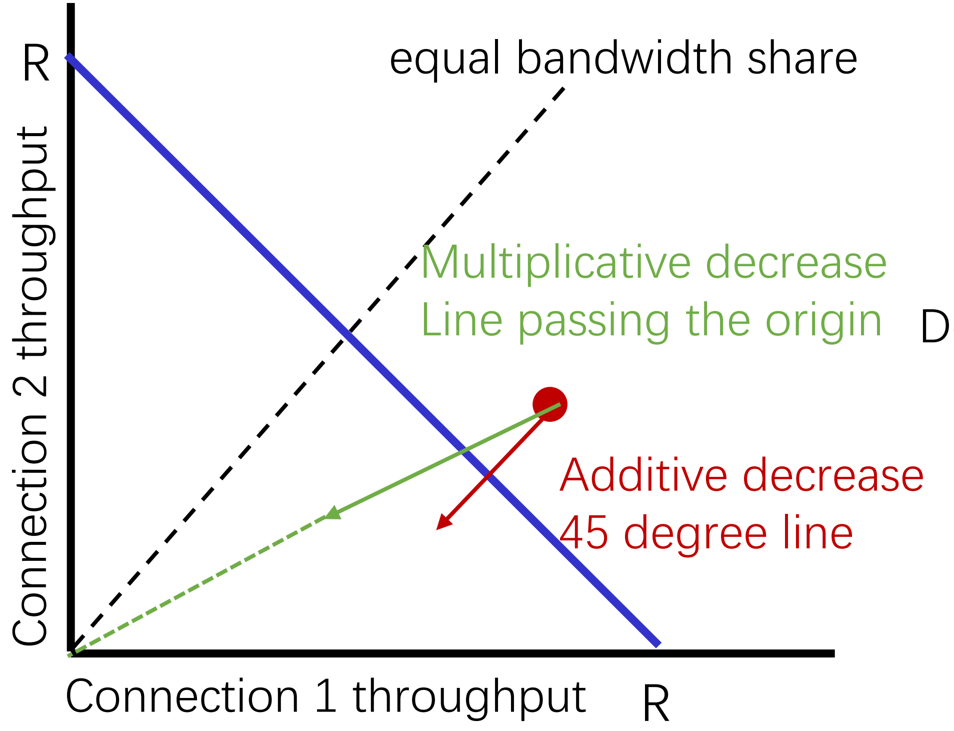
- Consider a case with two TCP connections



Decrease the throughput
to avoid congestion

Fairness in TCP

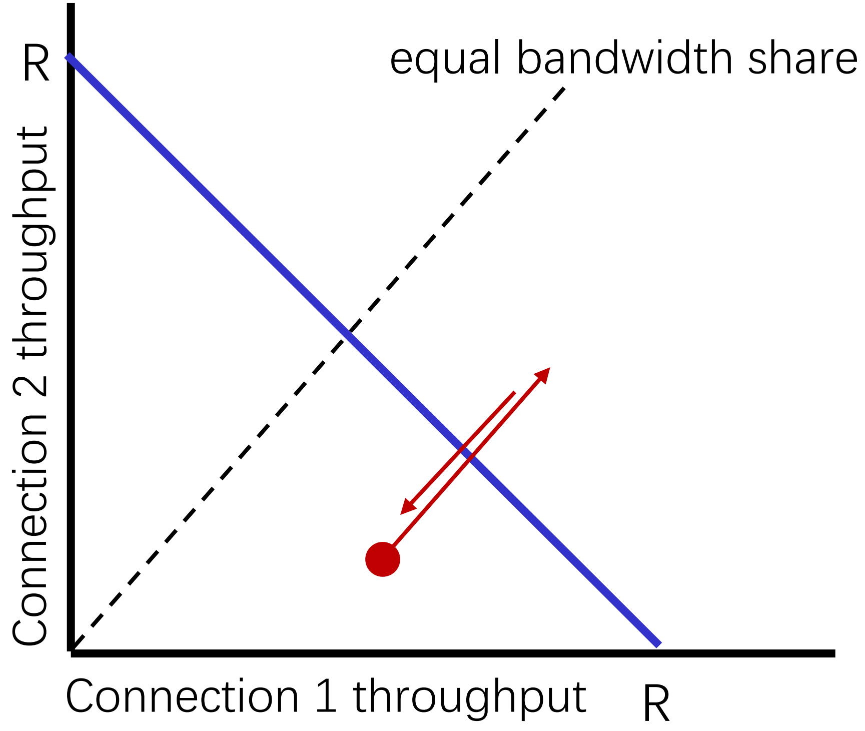
- Consider a case with two TCP connections



Decrease the throughput to avoid congestion

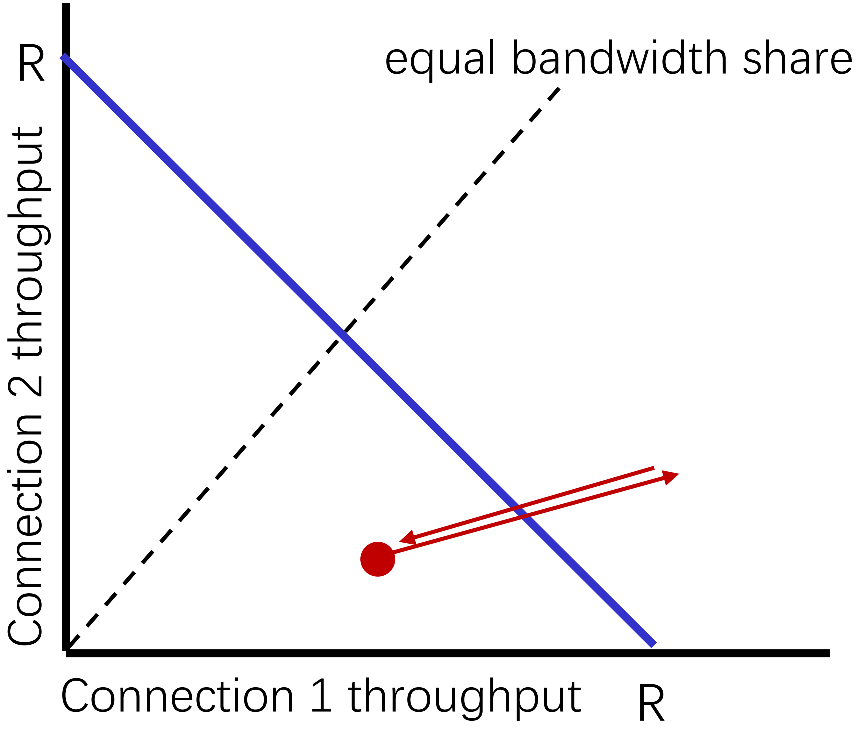
Fairness in TCP

- Consider a case with two TCP connections
  - Behavior of additive increase additive decrease
    - Stable but not fair



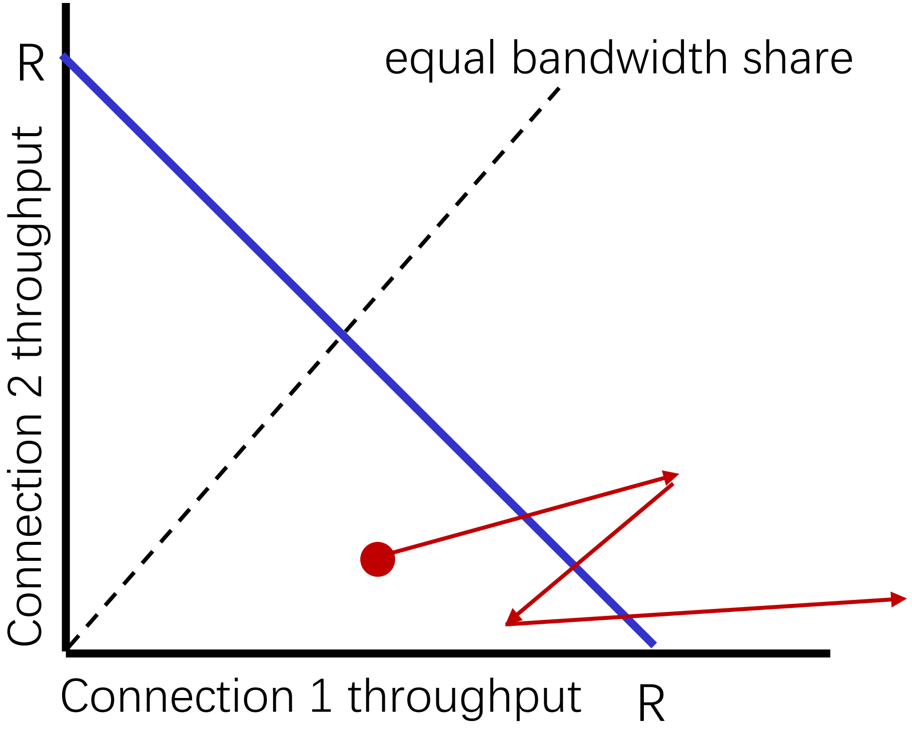
Fairness in TCP

- Consider a case with two TCP connections
  - Behavior of multiplicative increase multiplicative decrease
    - Stable but not fair



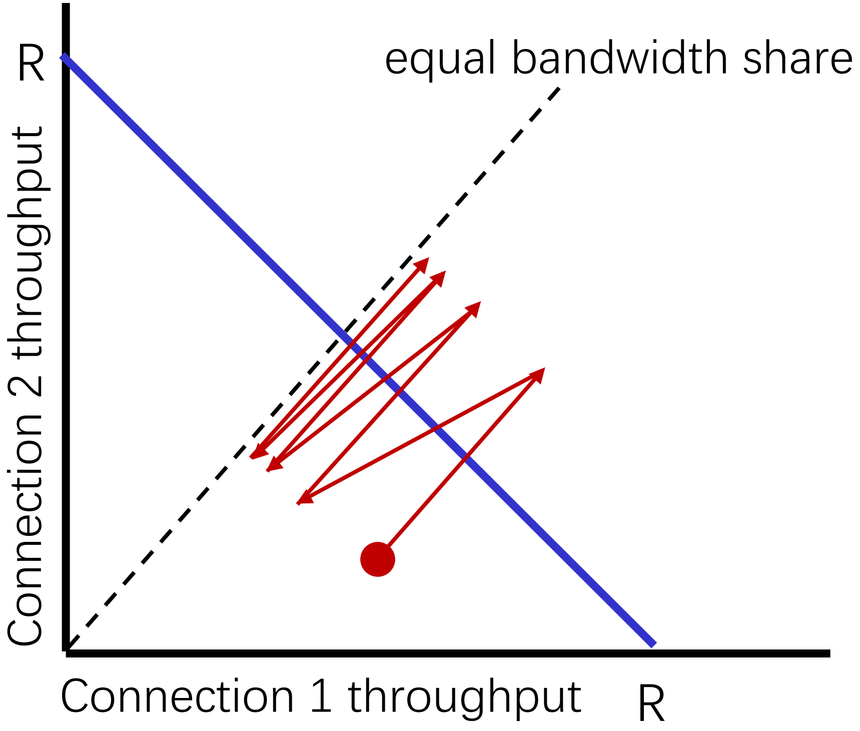
Fairness in TCP

- Consider a case with two TCP connections
  - Behavior of multiplicative increase additive decrease
    - Not stable



Fairness in TCP

- Consider a case with two TCP connections
  - Behavior of AIMD
    - Stable and faire



Fairness and RTT

- TCP connation with smaller RTT occupies more bandwidth
  - When congestion happens, they recover more quickly
    - TCP adjust cwnd in RTT basis

Fairness and Parallel TCP Connections

- Application can open multiple parallel connections between two hosts
  - web browsers do this , e.g., link of rate R with 9 existing connections:
    - new app asks for 1 TCP, gets rate R/10
    - new app asks for 11 TCPs, gets R/2

Fairness and UDP

- Multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- Instead use UDP:
  - send audio/video at constant rate, tolerate packet loss
- There is no “Internet police” policing use of congestion control

Outline

- TCP Fairness

➢QUIC

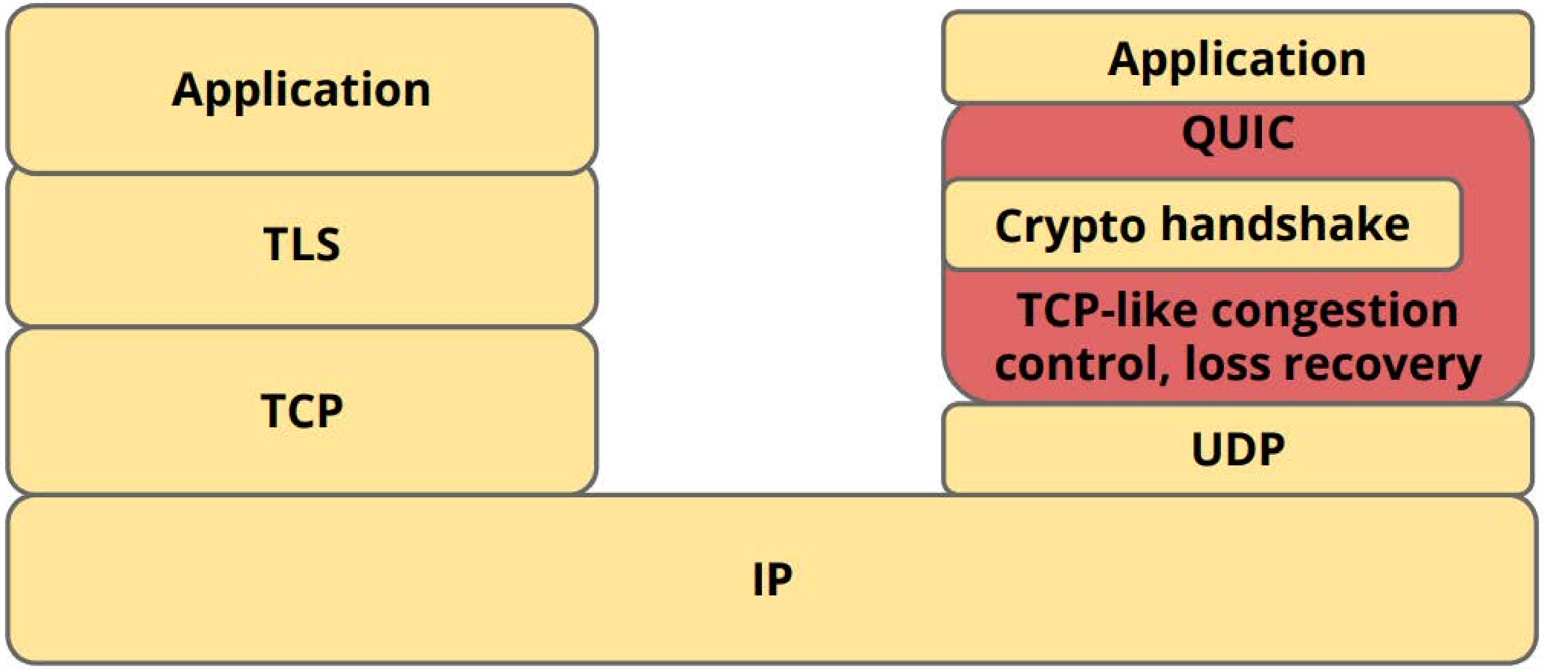
- QoS

QUIC

- QUIC: Quick UDP Internet Connections
- Application-layer protocol, on top of UDP
    - Deployed by Google staring at 2014
        - Deployed on many Google servers, apps (Chrome, mobile YouTube app)
    - QUIC working group formed in Oct 2016
- Initial goal: increase performance of HTTP

QUIC

- Protocol Stack

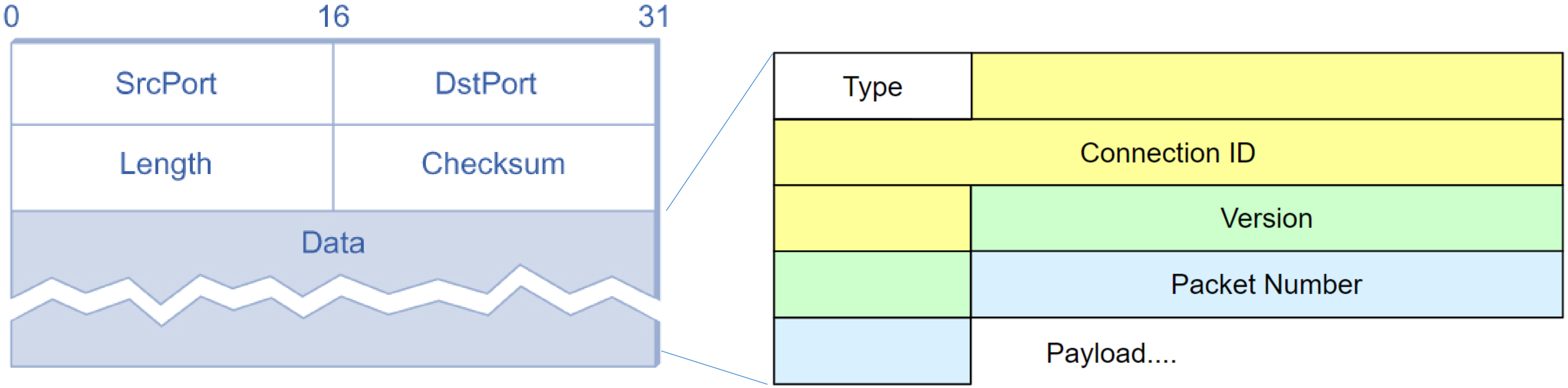


https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf

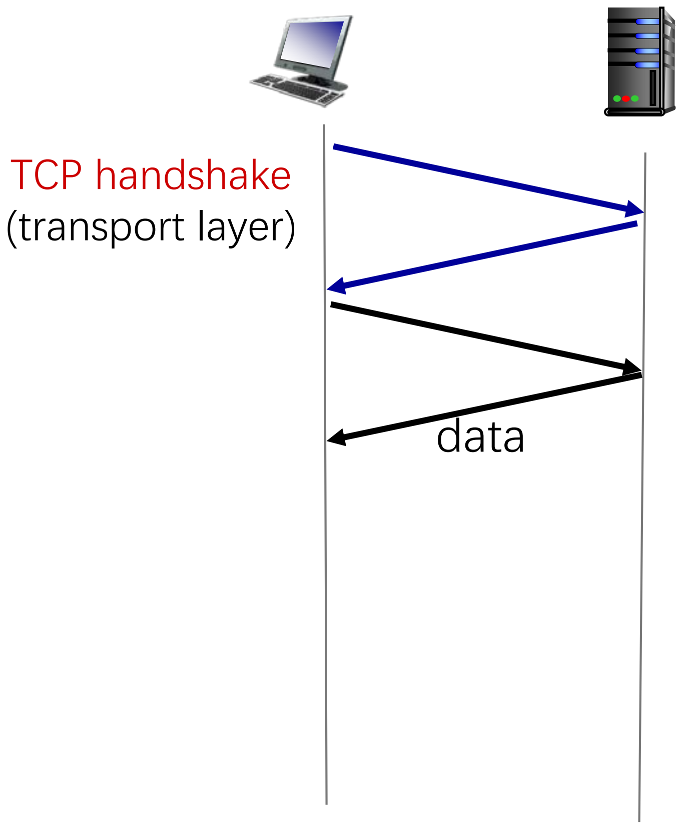
QUIC

- Key features
  - Always encrypted
  - 0-RTT connection establishment
  - Connection migration
  - Congestion control
  - Parallel Streams

QUIC - Header


0
16
31
SrcPort
DstPort
Length
Checksum
Data
Type
Connection ID
Version
Packet Number
Payload....

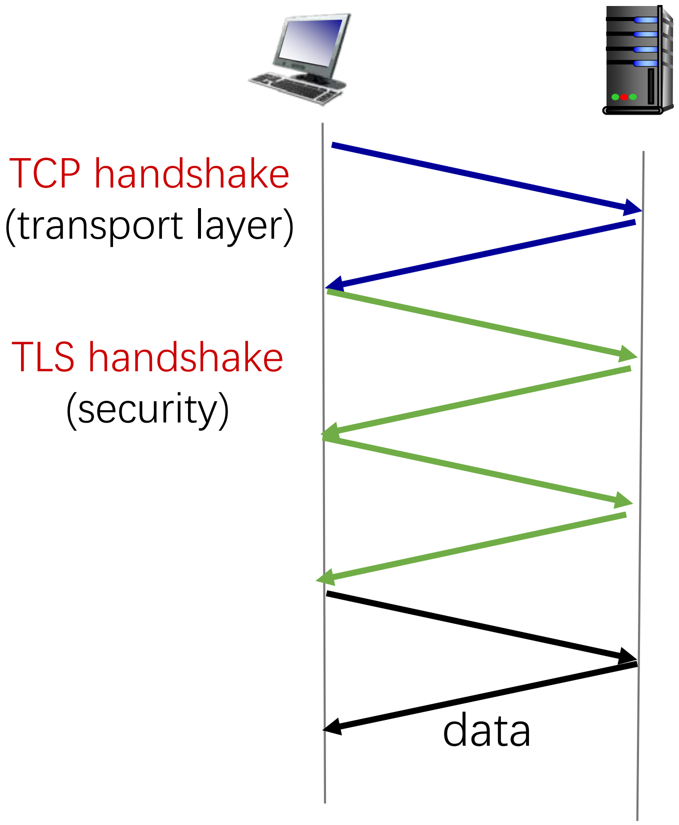
QUIC Connection Establishment



TCP handshake
(transport layer)

data

TCP
(2RTT)

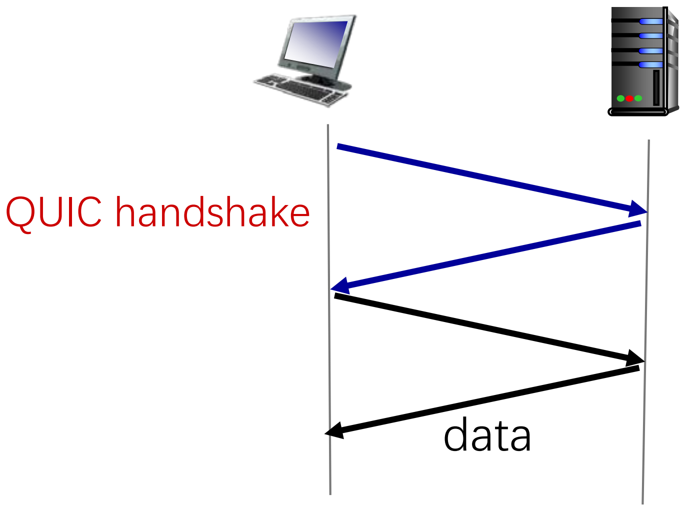


TCP handshake
(transport layer)

TLS handshake
(security)

data

TCP+TLS 1.2
(new 4RTT
resumed 3RTT)



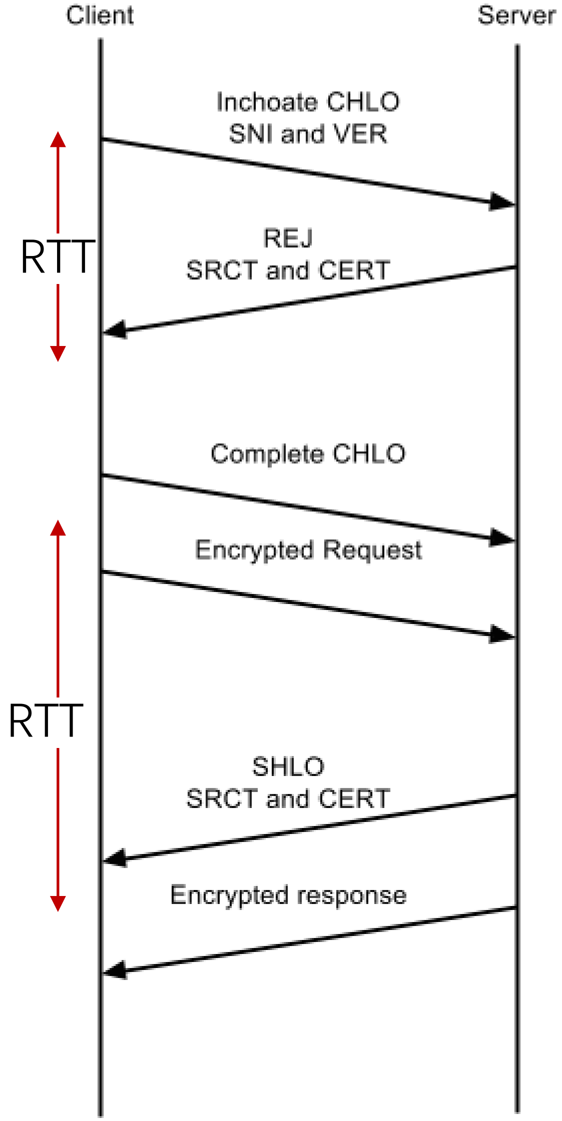
QUIC handshake

data

QUIC
(new 2RTT
Resumed 1RTT)

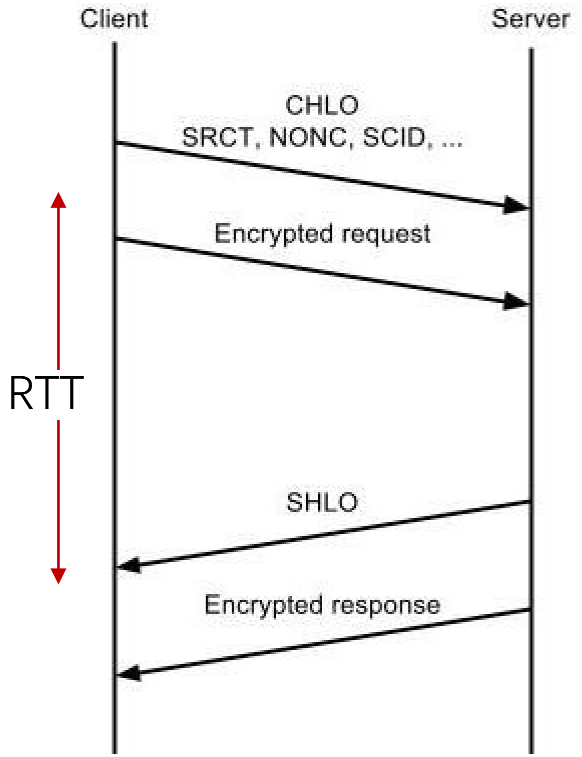
QUIC Connection Establishment

- 1-RTT (First-ever connection)
  - No cached information available
  - First CHLO is inchoate (empty)
    - Simply includes version and server name
  - Server responds with REJ
    - Includes server config, certs, etc.
    - Allows client to make forward progress
  - Second CHLO is complete
    - Followed by initially encrypted request data
  - Server responds with SHLO
    - Followed immediately by forward-secure encrypted response data



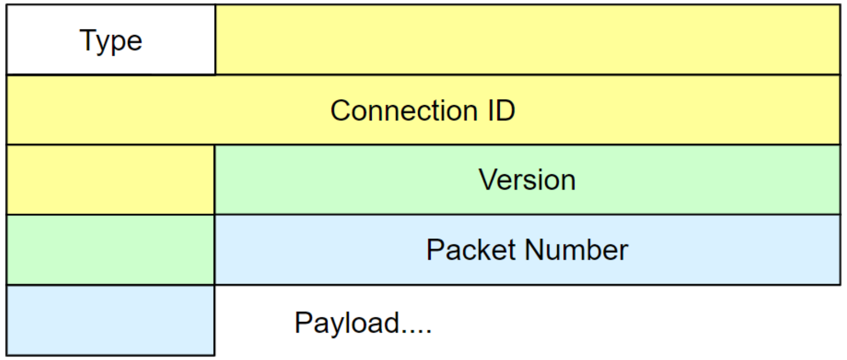
QUIC Connection Establishment

- 0-RTT (Subsequent connection)
  - Motivation: client can cache information about the *origin* it connected to
  - First CHLO is complete
    - Based on information from previous connection
    - Followed by initially encrypted data.
  - Server responds with SHLO
    - Followed immediately by forward-secure encrypted data



QUIC Connection Migration

- NAT Rebinding
  - NATs remaps port
    - Frequency (~ mins)
    - Why ? to release unused ports
      - According to TCP connection state (if they are closed)
    - UDP does not have connection state, QUIC state is encrypted
- Mobility
  - Switching between different IP
    - Wi-Fi and cellular network
- Connection Migration
  - Keep QUIC connections alive even if port and IP are chang
  - Detect connection path changes via Connection ID and IP/port
    - Connection is identified by connection ID rather than <IP, port>
    - 64-bit connection ID
    - randomly chosen by client



Type

Connection ID

Version

Packet Number

Payload....

QUIC Congestion Control

- Incorporates TCP best practices
  - TCP Cubic, Fast Retransmission, Selective ACK, etc.
- Better signaling than TCP
  - Each packet carries a monotonically increasing packet number
    - Better RTT measurement
    - Even ACK
  - Retransmitted packets also consume new sequence numbers
    - no retransmission ambiguity
- More verbose ACK
  - support 256 NAK ranges (vs. TCP's 3SACK ranges)

QUIC - Parallel Streams

- Handle HOL blocking

Outline

- TCP Fairness
- QUIC

➢QoS