

Lesson 05-05: QUIC

CS 326E Elements of Networking

Mikyung Han

mhan@cs.utexas.edu

Example Protocols

FTP, HTTP, SMTP

Application

TCP, UDP

Transport

IP

Network

Ethernet, WiFi

Link

802.3 PHY

Physical

Responsible for

application specific needs

process to process data transfer

host to host data transfer across different network

data transfer between physically adjacent nodes

bit-by-bit or symbol-by-symbol delivery

Internet Reference Model

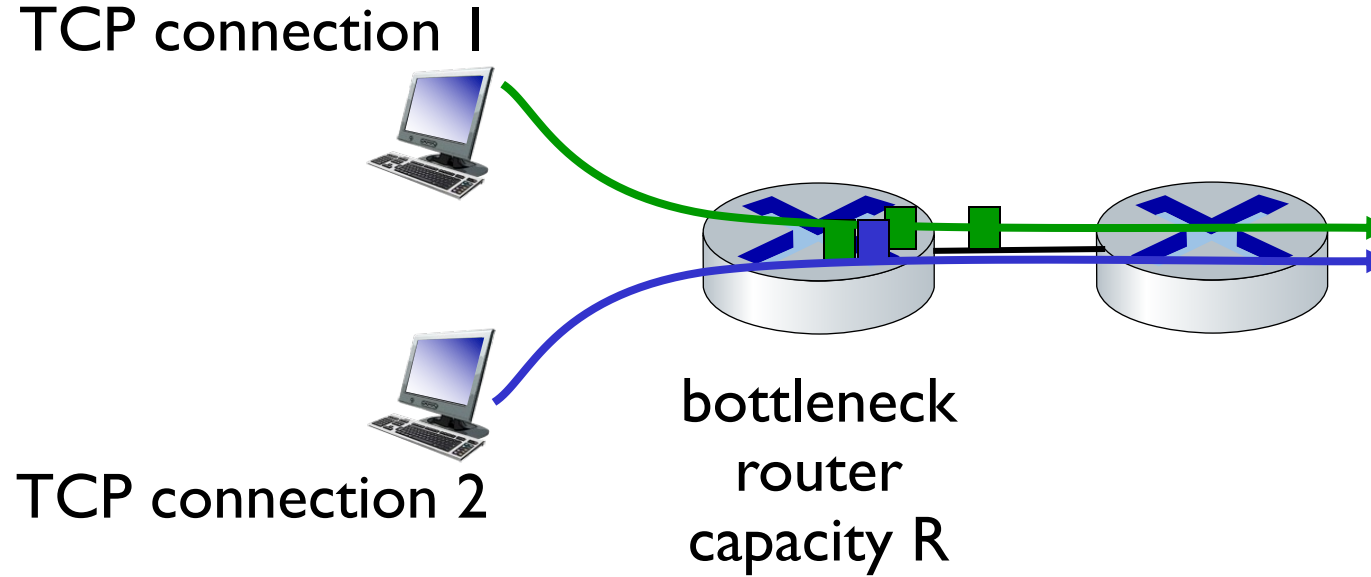


Outline

👉 I. Is TCP fair?

TCP fairness

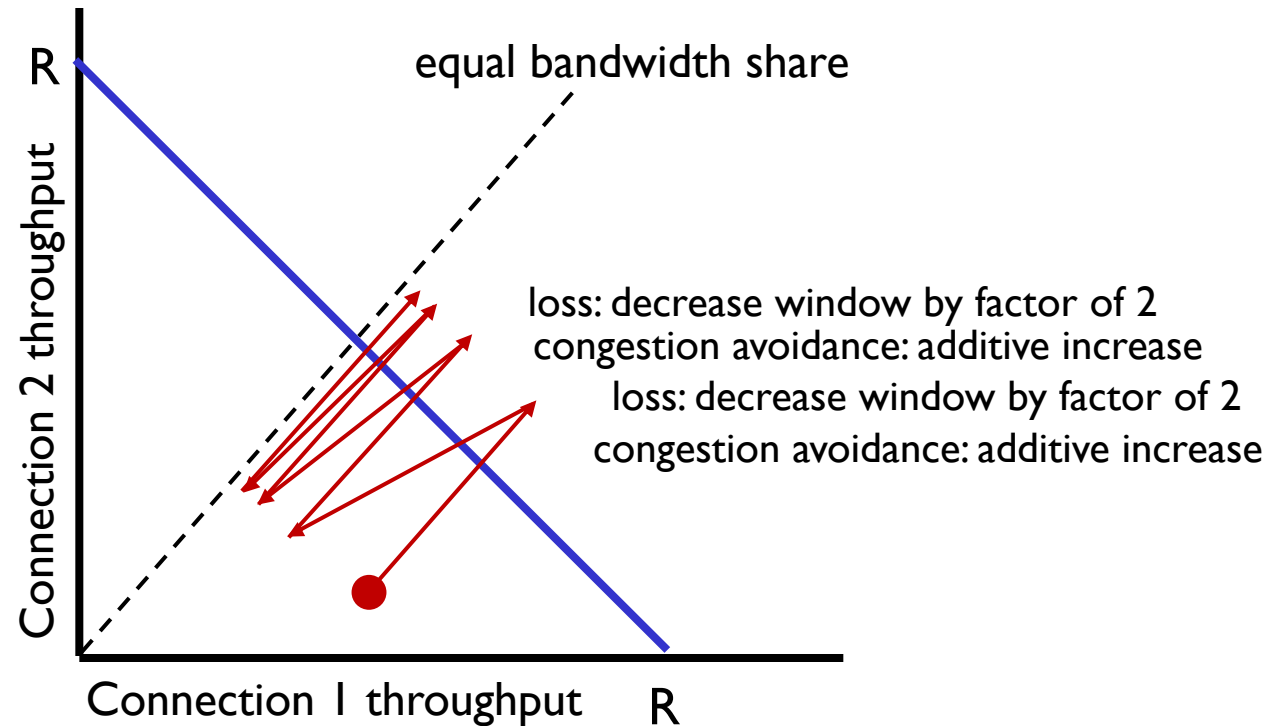
Fairness goal: if K TCP sessions share same bottleneck link of bandwidth R , each should have average rate of R/K



Q: is TCP Fair?

Example: two competing TCP sessions:

- additive increase gives slope of 1, as throughput increases
- multiplicative decrease decreases throughput proportionally



Is TCP fair?

A: Yes, under idealized assumptions:

- same RTT
- fixed number of sessions only in congestion avoidance

Fairness: must all network apps be “fair”?

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

Fairness, 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 $R/2$

There is NO “Internet police” policing fairshare

Outline

1. Is TCP fair?

 2. Evolution in Transport layer: QUIC

Evolving transport-layer functionality

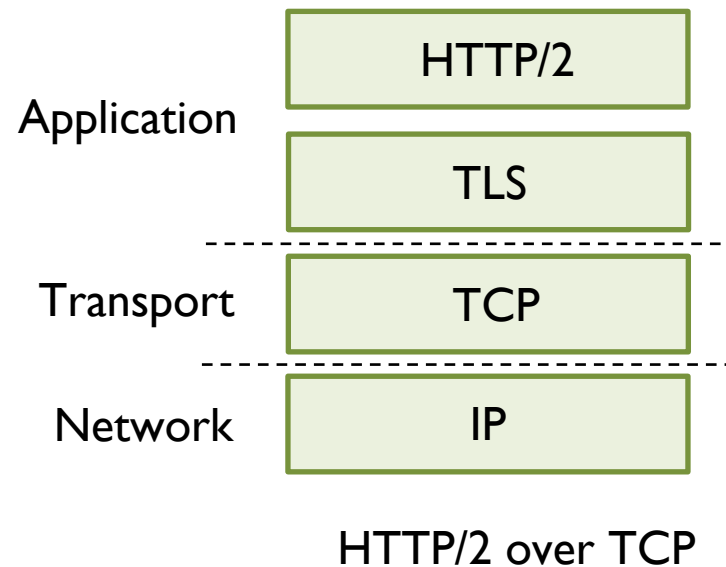
- TCP, UDP: principal transport protocols for 40 years
- different “flavors” of TCP developed, for specific scenarios:

| Scenario | Challenges |
|--|--|
| Long, fat pipes (large data transfers) | Many packets “in flight”; loss shuts down pipeline |
| Wireless networks | Loss due to noisy wireless links, mobility; TCP treat this as congestion loss |
| Long-delay links | Extremely long RTTs |
| Data center networks | Latency sensitive |
| Background traffic flows | Low priority, “background” TCP flows |

- moving transport-layer functions to application layer, on top of UDP
 - HTTP/3: QUIC

QUIC: Quick UDP Internet Connections

- application-layer protocol, on top of UDP
 - increase performance of HTTP
 - deployed on many Google servers, apps (Chrome, mobile YouTube app)

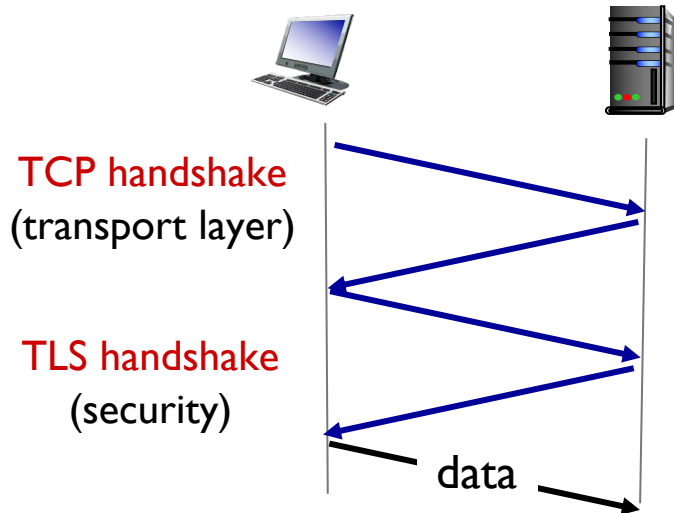


QUIC: Quick UDP Internet Connections

adopts approaches we've studied in this chapter for connection establishment, error control, congestion control

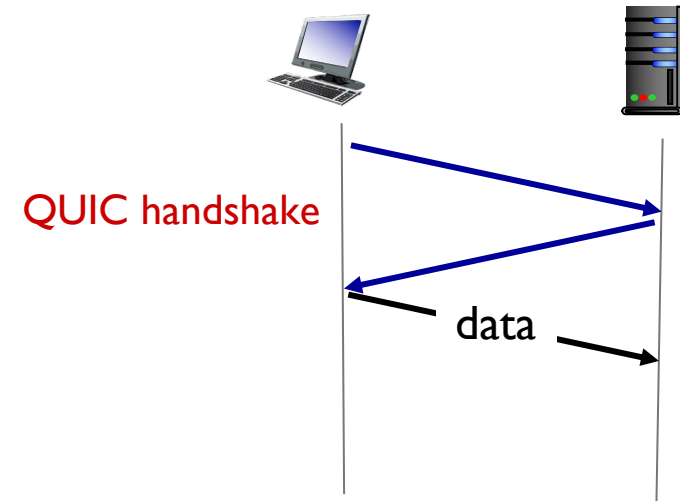
- **error and congestion control:** “Readers familiar with TCP’s loss detection and congestion control will find algorithms here that parallel well-known TCP ones.” [from QUIC specification]
- **connection establishment:** reliability, congestion control, authentication, encryption, state established in one RTT
- multiple application-level “streams” multiplexed over single QUIC connection
 - separate reliable data transfer, security
 - common congestion control

QUIC: Connection establishment



TCP (reliability, congestion control state)
+ TLS (authentication, crypto state)

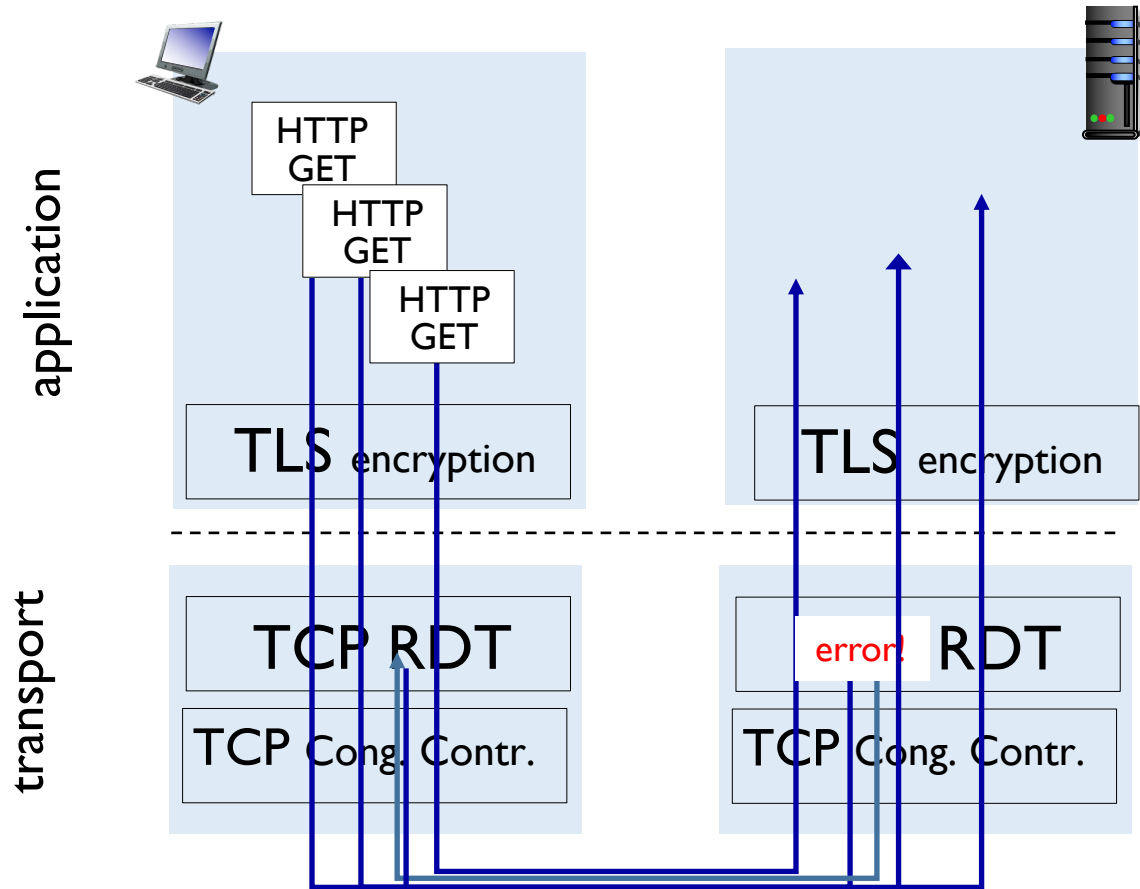
- 2 serial handshakes



QUIC: reliability, congestion control,
authentication, crypto state

- 1 handshake

QUIC: streams: parallelism, no HOL blocking



(a) HTTP 1.1

Acknowledgements

Slides are adopted from Kurose' Computer Networking Slides