CS 725/825 & IT 725
Lecture 16
Transport Layer

October 23, 2024

Retransmission Timeout

RFC 6298

Initialization:

RTO ← 1 sec

After the first measurement:

SRTT ← R

RTTVAR ← R/2

RTO ← SRTT + max (G, K * RTTVAR)

After subsequent measurements:

RTTVAR ← (1 - beta) * RTTVAR + beta * |SRTT - R'|

SRTT ← (1 - alpha) * SRTT + alpha * R'

RTO ← SRTT + max (G, K * RTTVAR)

Where:

R - first RTT measurement
R' - subsequent RTT measurement
RTTVAR - RTT variance
SRTT - smoothed RTT estimate
RTO - retransmission timeout
G - clock granularity

Recommended values:

alpha=1/8, beta=1/4, K=4

Exponential Back-off

RTO after a timeout:

Recommended value: $q = 2$

$$RTO \leftarrow q * RTO$$

This a congestion control mechanism since retransmissions are delayed after packet loss detected. The delay is increasing exponentially with more packet losses.

Transmission Window

- Network provides no explicit indication of congestion
- Source observes RTT and packet loss and adjusts transmission rate according to its estimate of the congestion state of the network
- Transmission window size is proportional to the maximum transmission rate
- Additive Increase Multiplicative Decrease (AIMD)
  - better safe than sorry

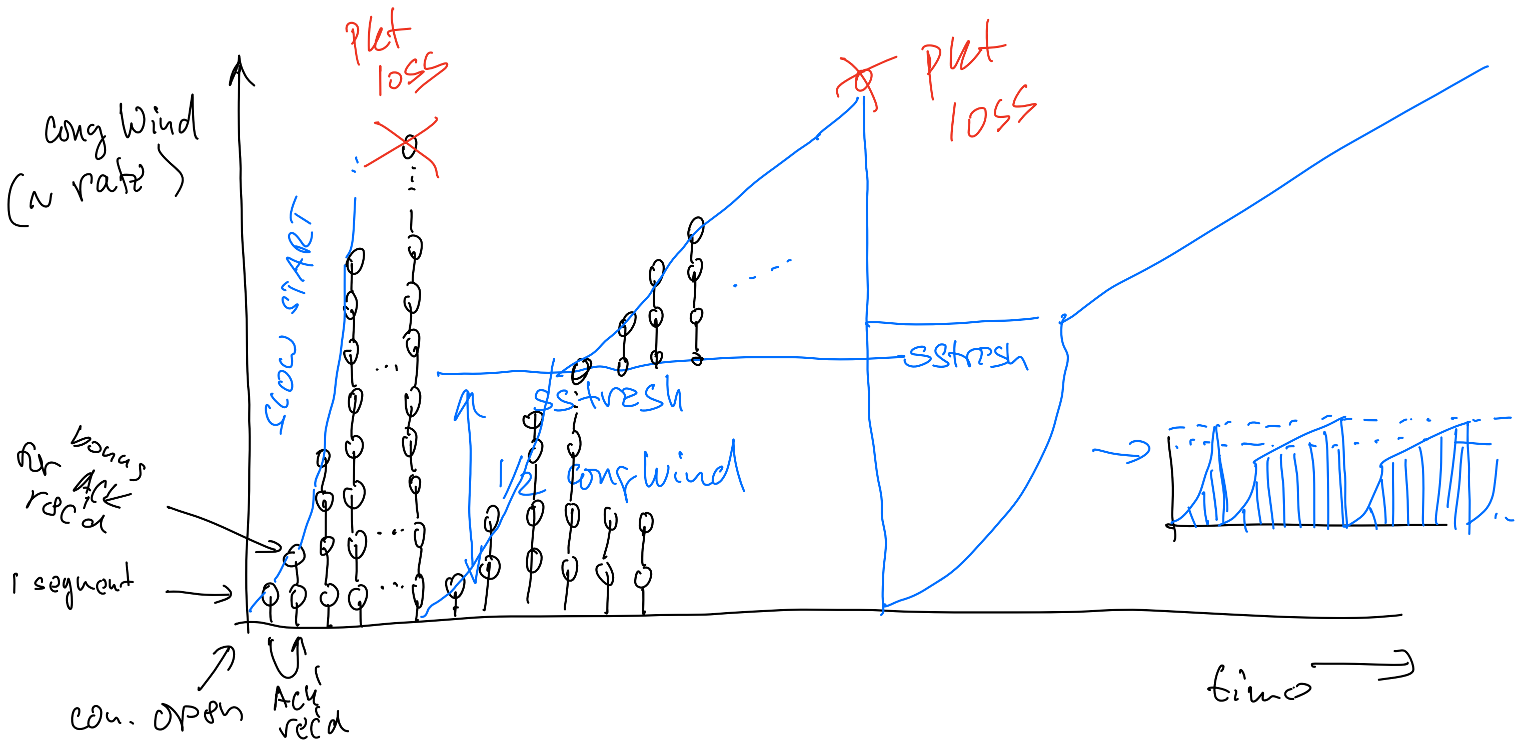
Network Congestion Control

- **Method**:

  TransWind = min(*RecvWind*, *CongWind*)
  EffectiveWind = *TransWind* - (*LastByteSent* - *LastByteAckd*)

- *EffectiveWind* - used in transmission
- *RecvWind* - from Window Size field
- *CongWind* - transmitter's estimate of how many unacknowledged packets can be pushed onto the network without causing congestion

TCP CONGESTION WINDOW



Congestion Window (original)

- Components algorithms of TCP network congestion control (RFC 2001):
  - Slow Start - initial growth of CongWind
  - Congestion Avoidance - AIMD-based “search” for optimal rate
  - Fast Retransmit - quick recovery from isolated packet losses
  - Fast Recovery - undoing congestion control steps under Fast Recovery

Variants of TCP (examples)

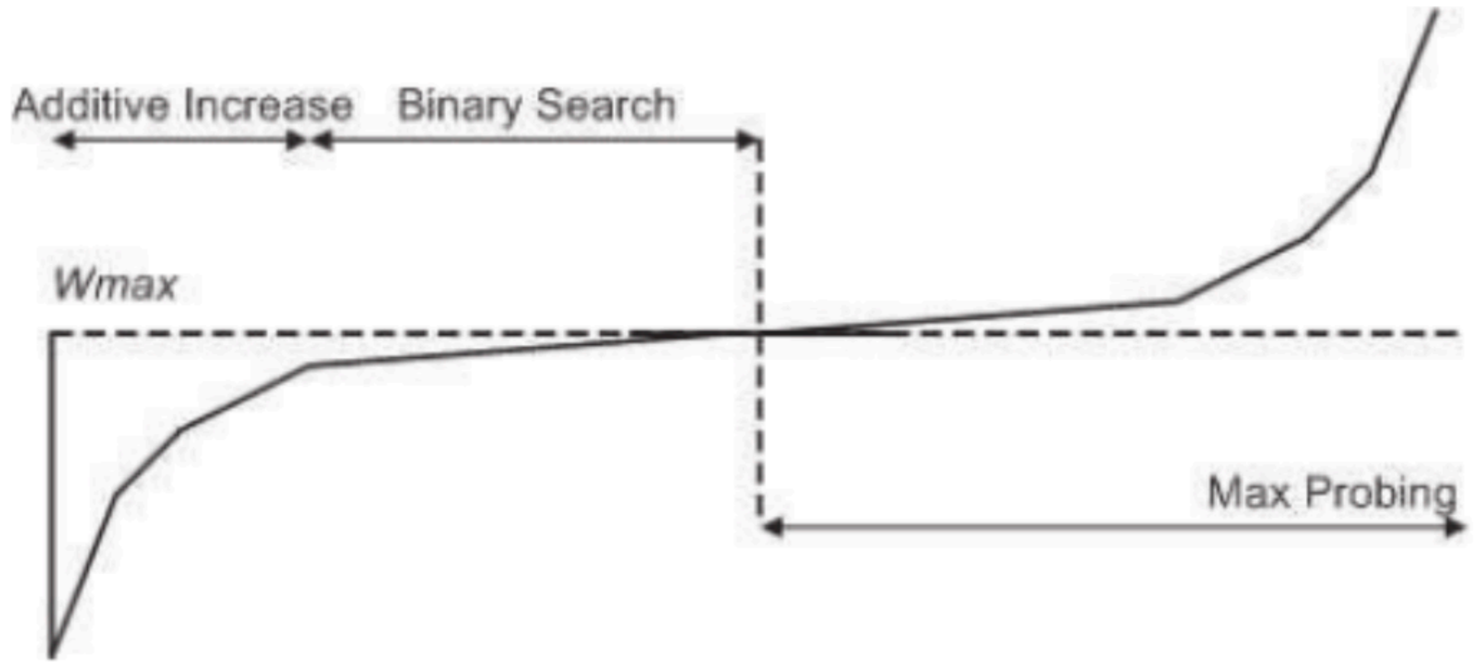
- Original TCP (RFC1122)
- TCP Tahoe (adds Fast Retransmit)
- TCP Reno (adds Fast Recovery)
- TCP Vegas (RTT-based)
- TCP BIC and CUBIC (Linux up to kernel 3.2)
- Compound TCP (Windows since Vista)
- TCP Proportional Rate Reduction (PRR) (Linux)
- TCP Bottleneck Bandwidth and Round-trip propagation time (BBR) (RTT-based, developed by Google)

TCP Vegas

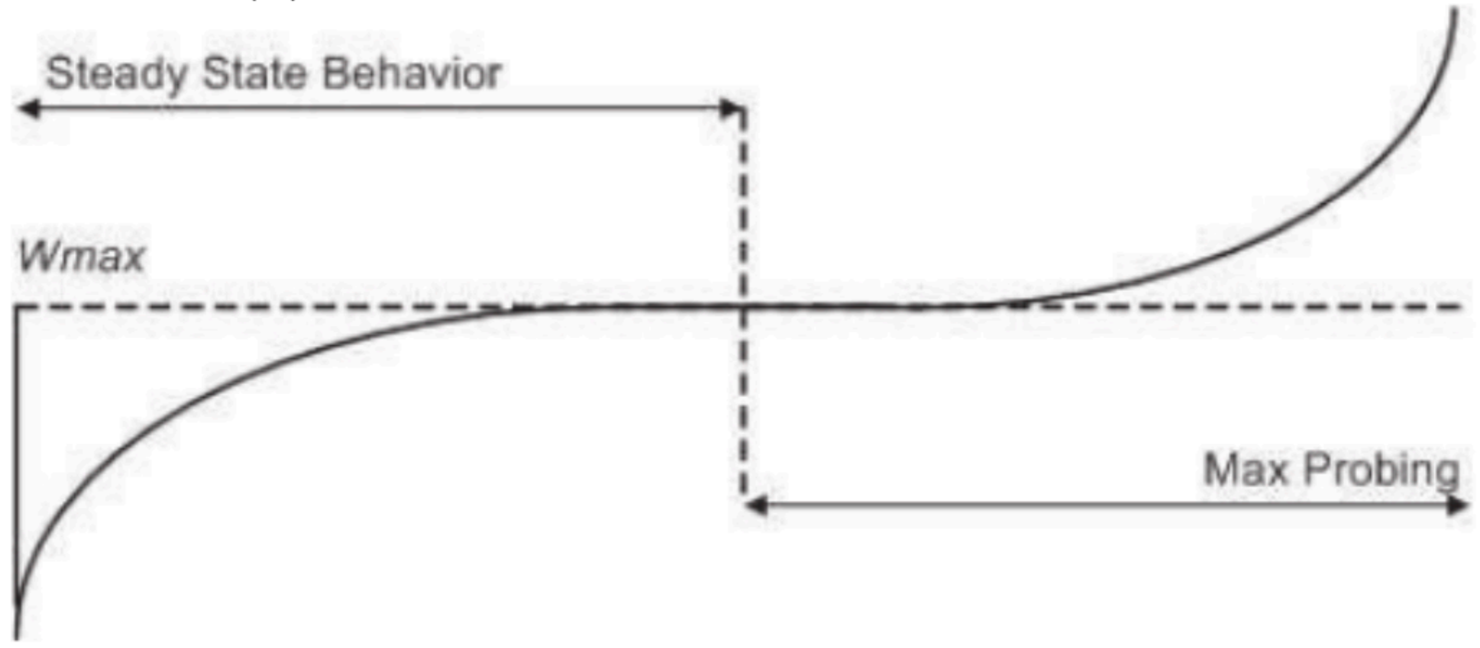
- RTT observed
- An increase in RTT indicates congestion
  - reduce transmission rate
- Steady RTT measurements indicate underutilization
  - slowly increase transmission rate until RTT starts increasing

TCP CUBIC

- An update of TCP BIC (Binary Increase Congestion control)
- “modifies the linear window growth function of existing TCP standards to be a cubic function in order to improve the scalability of TCP over fast and long distance networks”



(a) BIC-TCP window growth function.



(b) CUBIC window growth function.

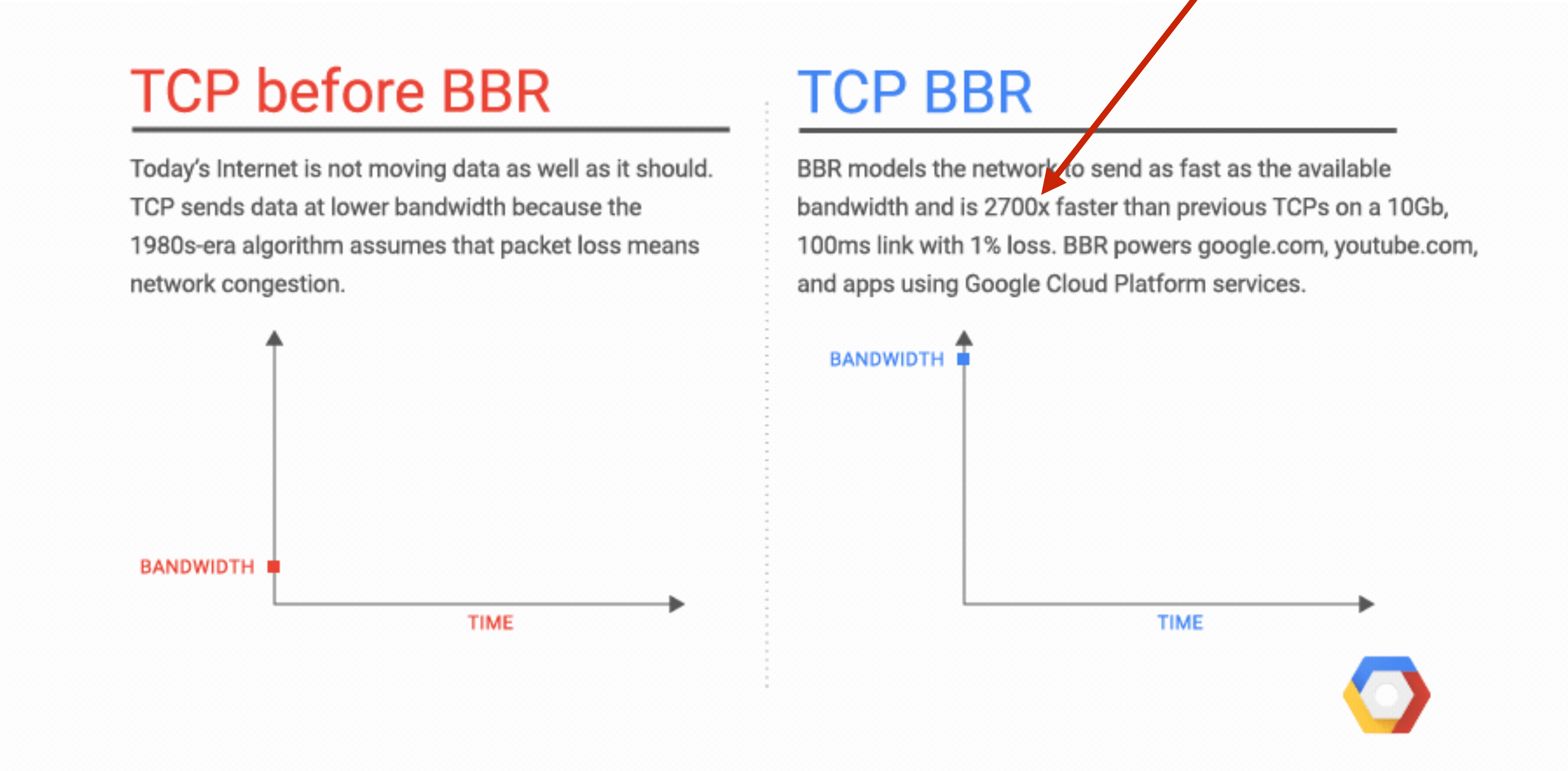
From: Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42, 5 (July 2008), 64–74. DOI:https://doi.org/10.1145/1400097.1400105

TCP BBR

- Bottleneck Bandwidth and Round-trip propagation time
- Designed by Google (~2016)
  - with YouTube as the motivating use case
  - available in Linux kernel 4.9+
- As the protocol name suggests:
  - "BBR congestion control computes the sending rate based on the delivery rate (throughput) estimated from ACKs" (comment in tcp-bbr.c in Linux kernel)

TCP BBR

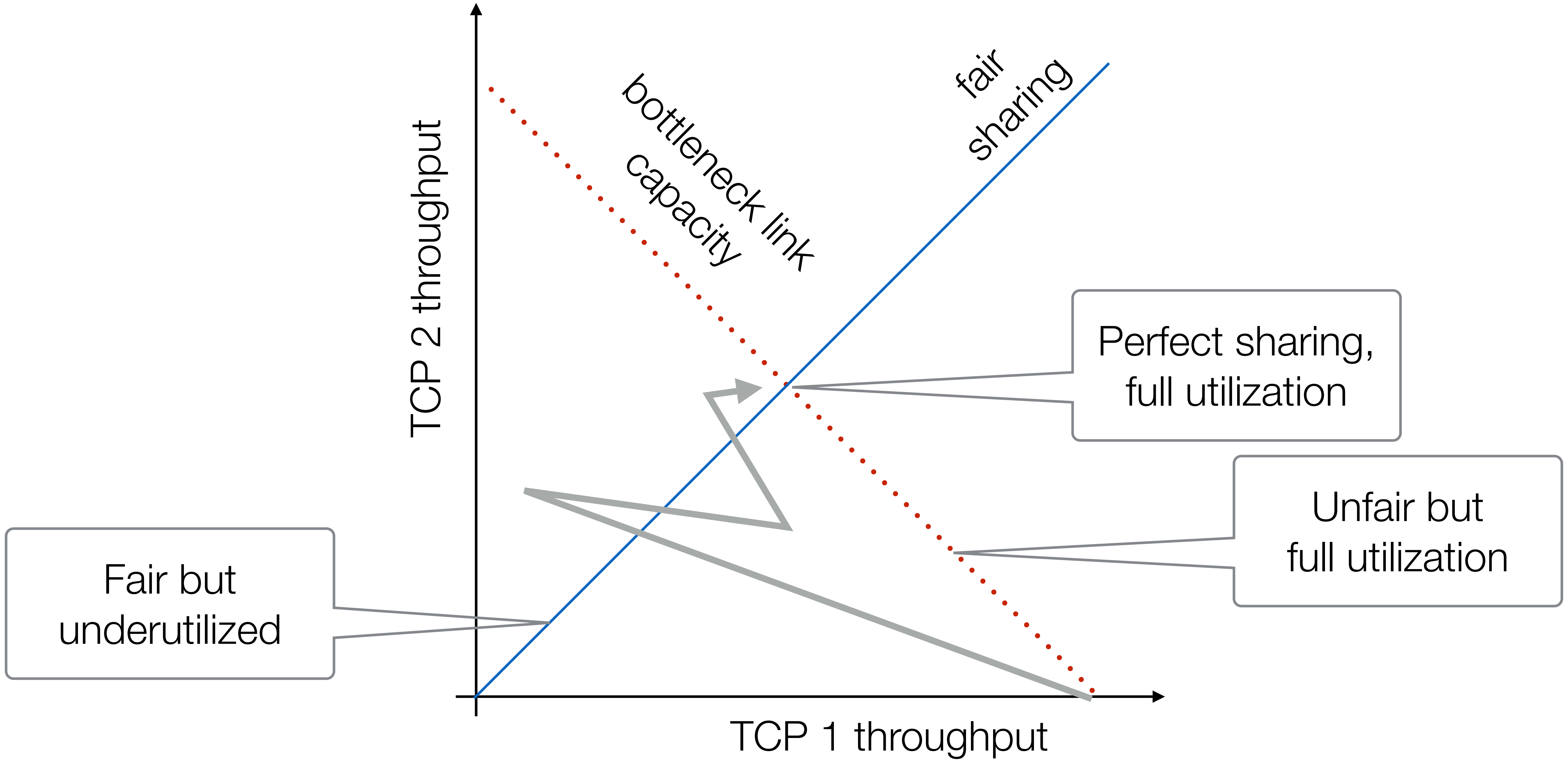
- One has to be careful when making claims:



From: https://cloud.google.com/blog/products/networking/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster (interestingly, the link no longer works, a copy of the article is still available at https://www.googblogs.com/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster/)

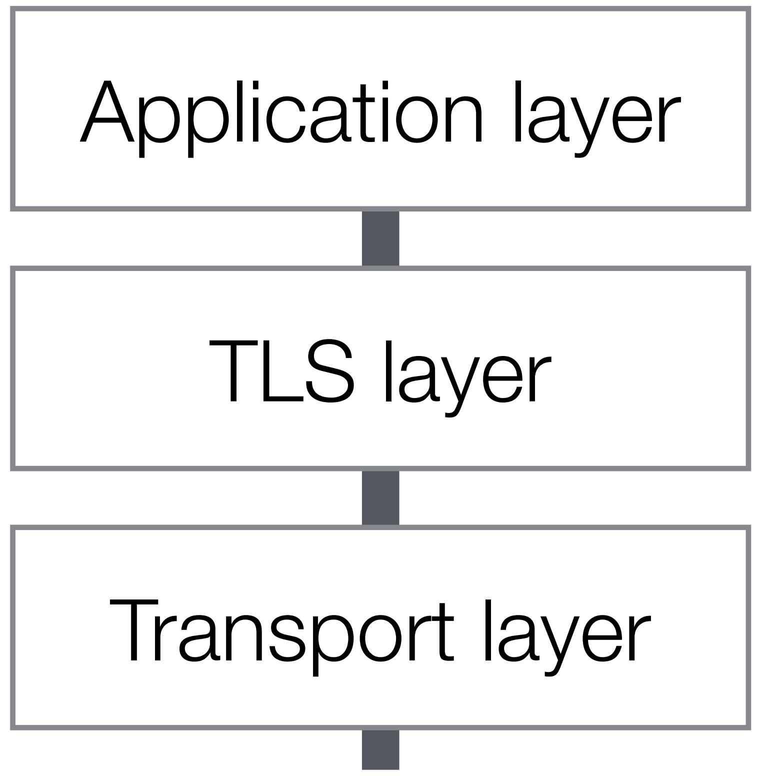
TCP Fairness

- Example: two TCP connections competing with each other on a bottleneck link:



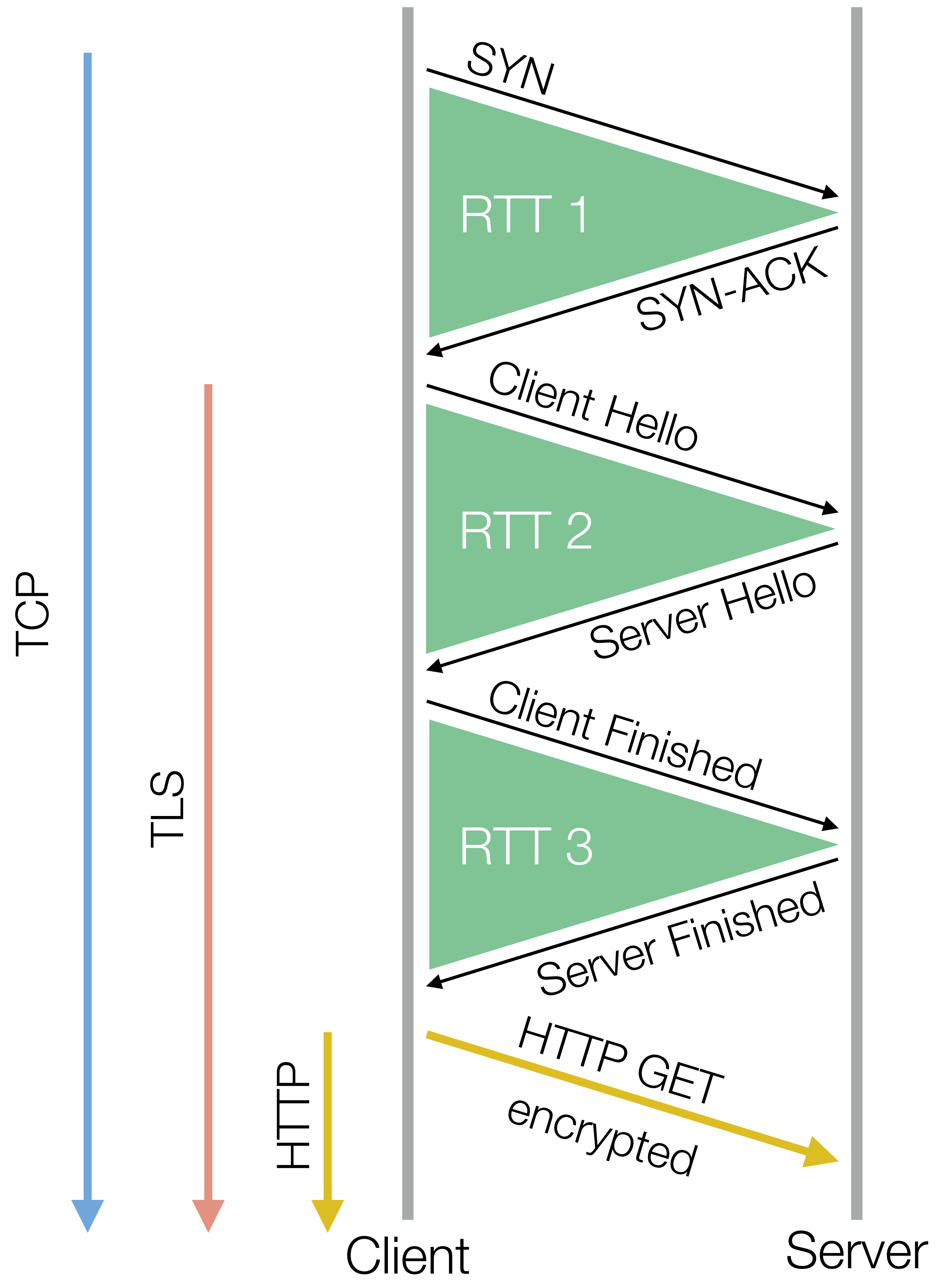
Transport Layer Security

- Transport Layer Security (TLS) - cryptographic protocols that to provide privacy (encryption) and data integrity protection
- … earlier versions known as SSL (Secure Socket Layer) is now deprecated but the term is widely used as a synonym for TLS
- Most used version TLS 1.2 (2008)
- Current version: TLS 1.3



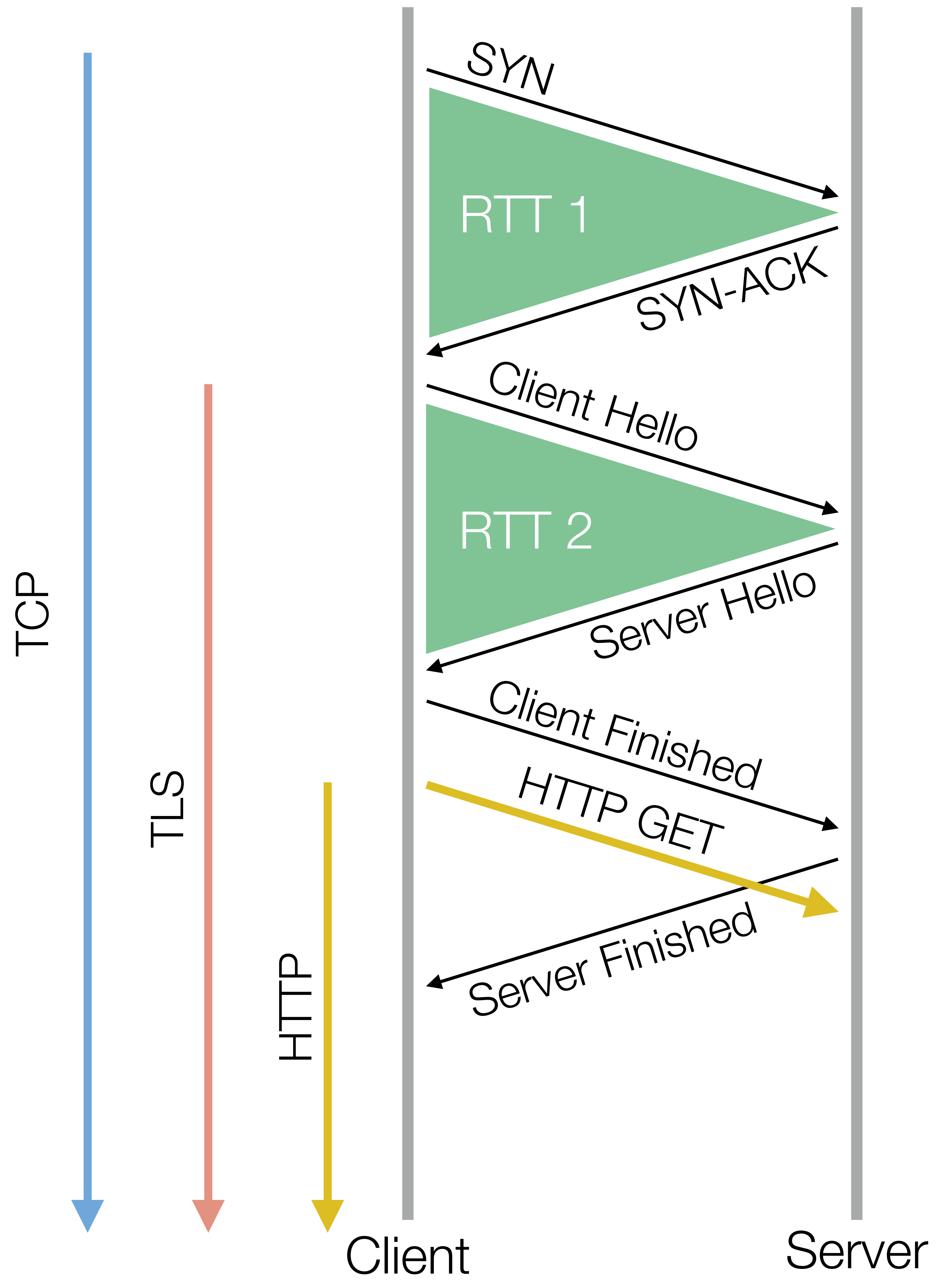
TLS connection latency

- TLS 1.2
  - 3 RTTs required to establish a secure connection



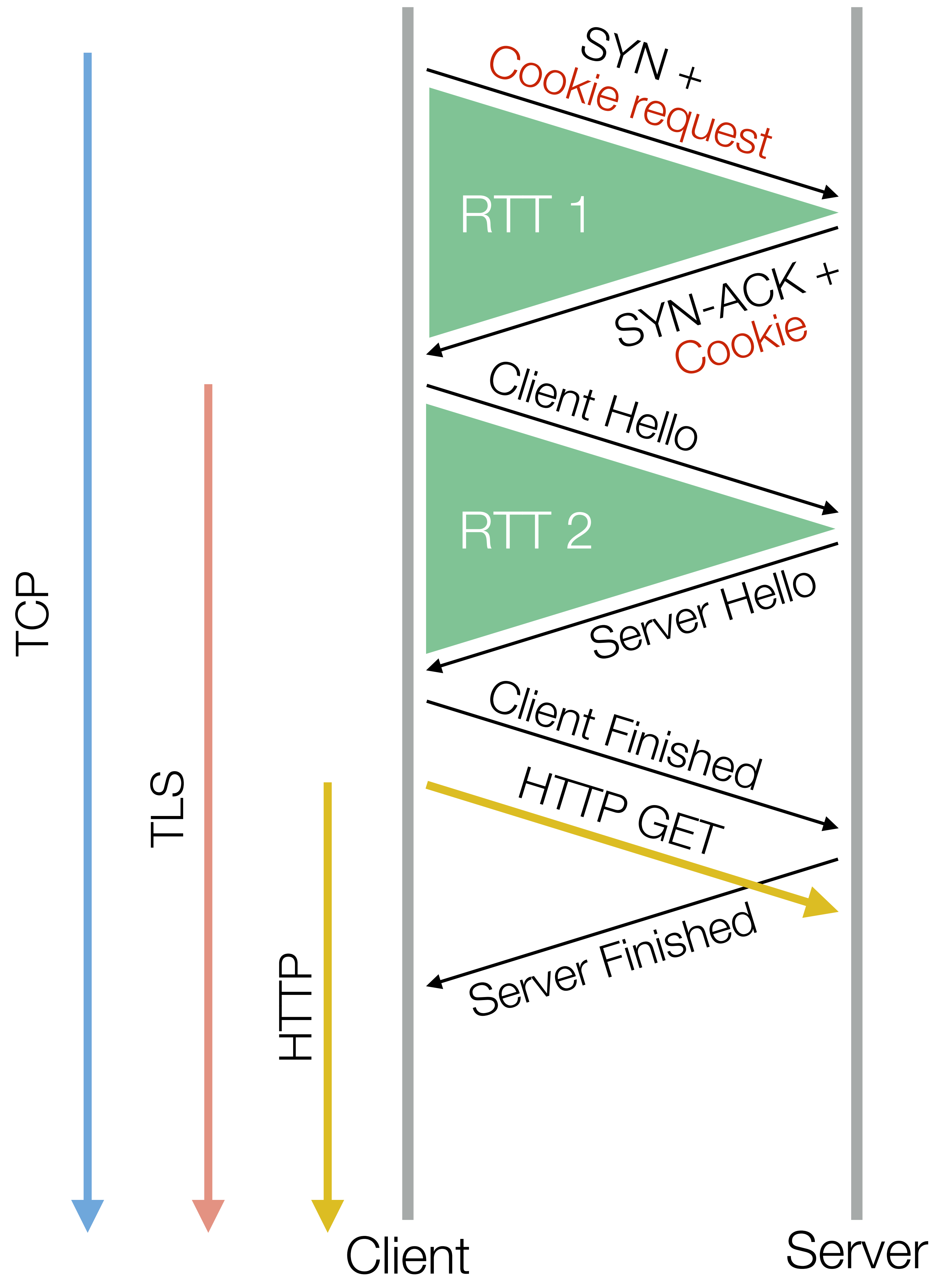
TLS connection latency

- TLS False Start option
  - 2 RTTs required to establish a secure connection



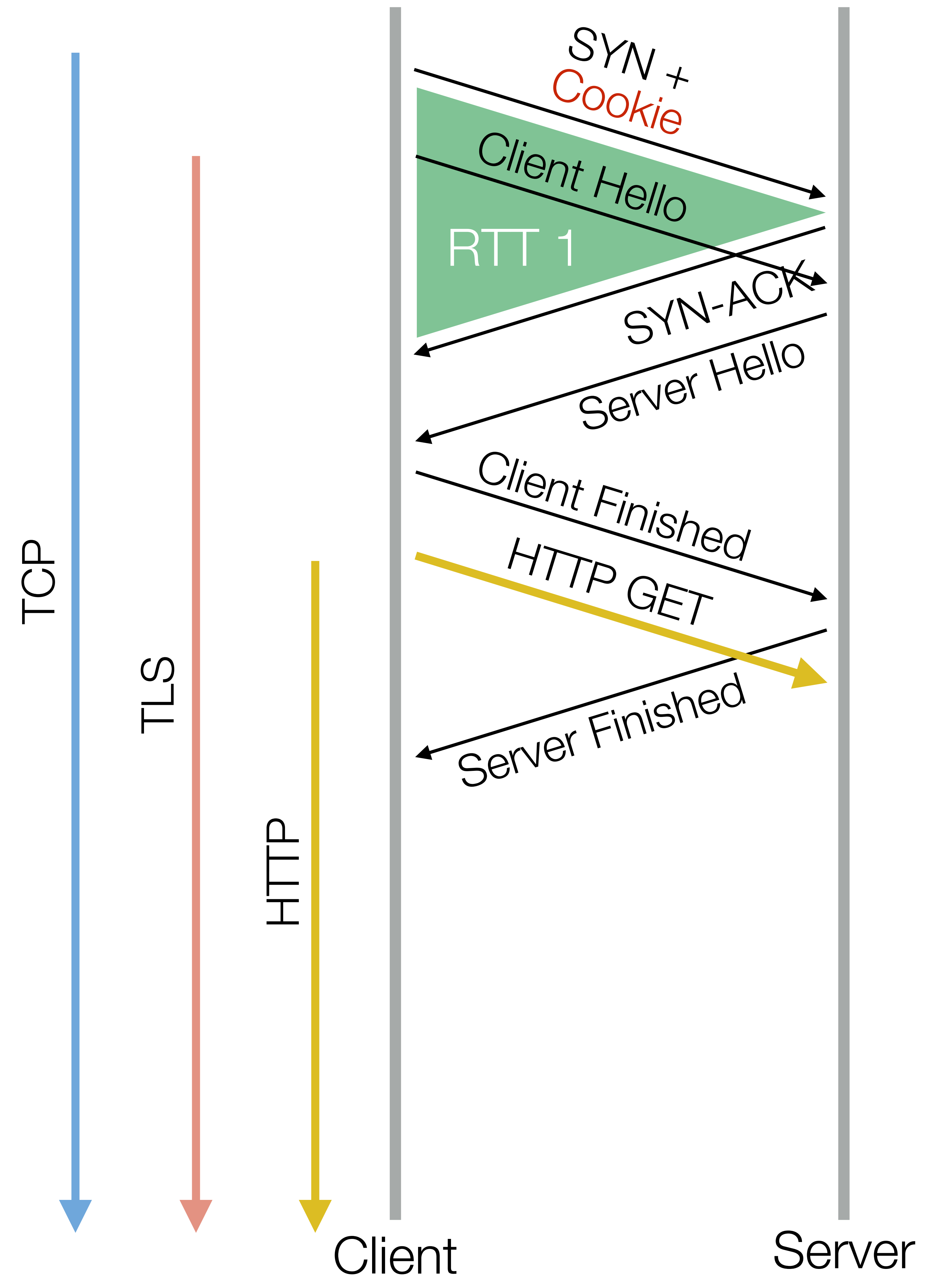
TLS connection latency

- TLS Fast Open option
  - when client connects for the first time, 2 RTTs are still required to establish a secure connection
  - server provides Fast Open Cookie to be used to speed-up subsequent connections



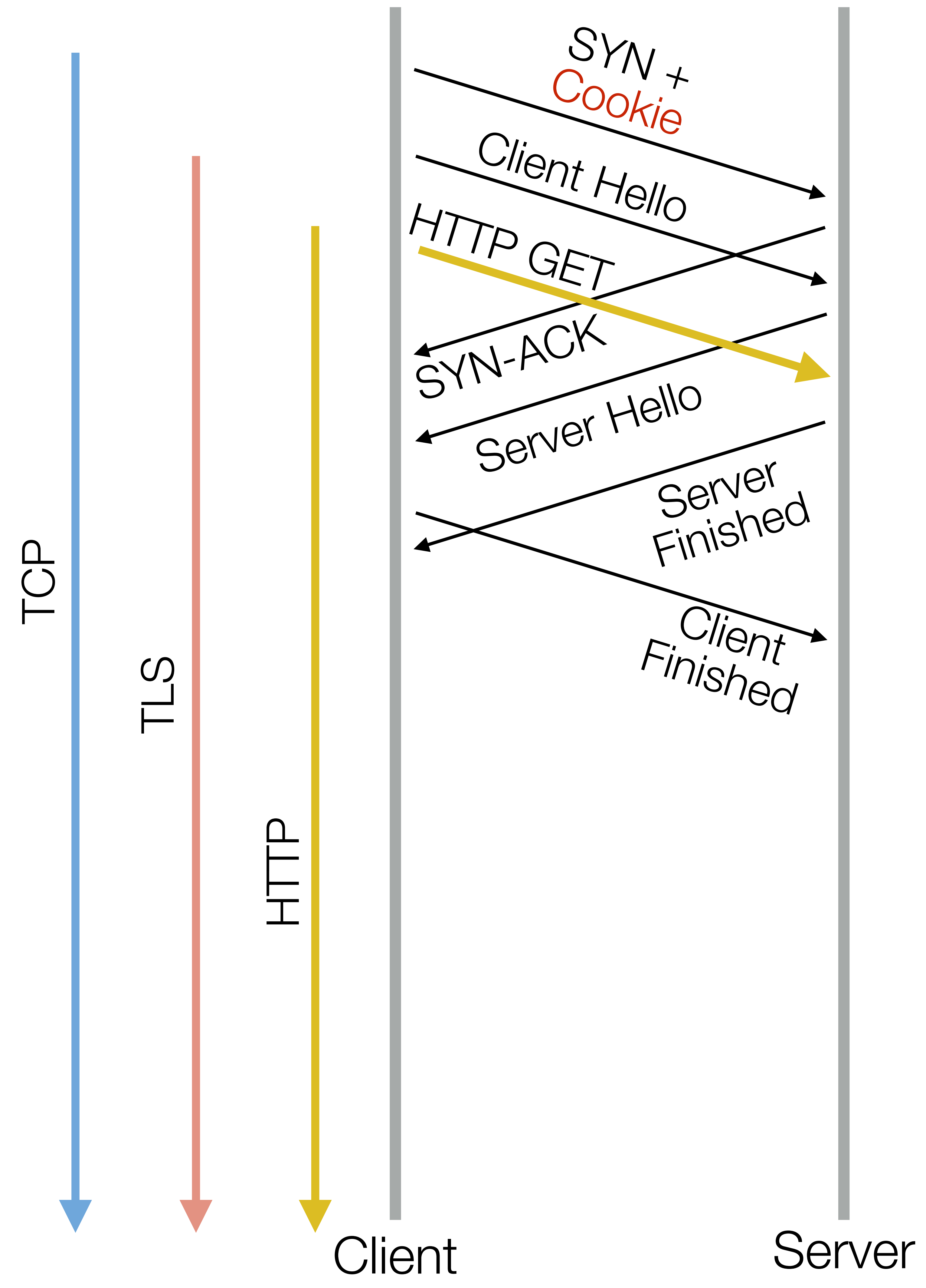
TLS connection latency

- TLS Fast Open option
  - for subsequent connections, only one RTTs required to establish a secure connection
  - client sends previously received Fast Open Cookie



TLS connection latency

- 0-RTT with TLS 1.3
  - for subsequent connections (using Fast Open Cookie), HTTP command is set before the TLS connection is fully established
  - However, the initial data sent to the server is susceptible (e.g., replay attack)



UDP

- User Datagram Protocol (RFC 768)
  - A wrapper protocol for IP to add port numbers
  - 8 bytes

Source Port	Destination Port
Length	Checksum