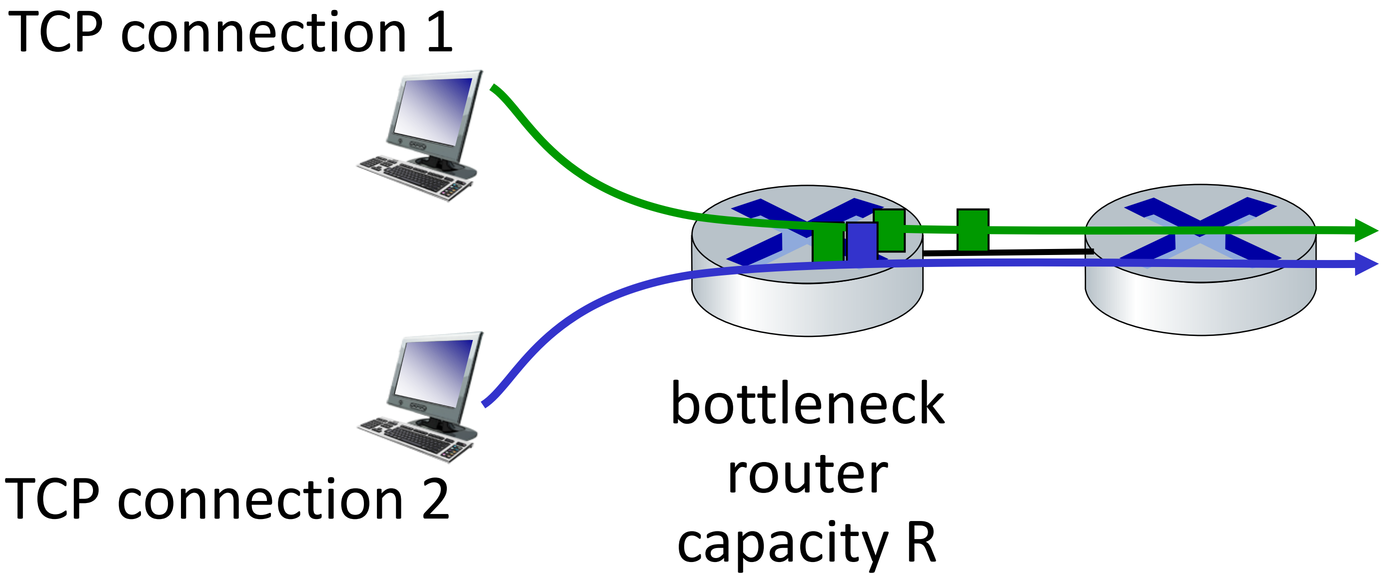
Transport Layer

Anand Baswade

anand@iitbhilai.ac.in

TCP fairness

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



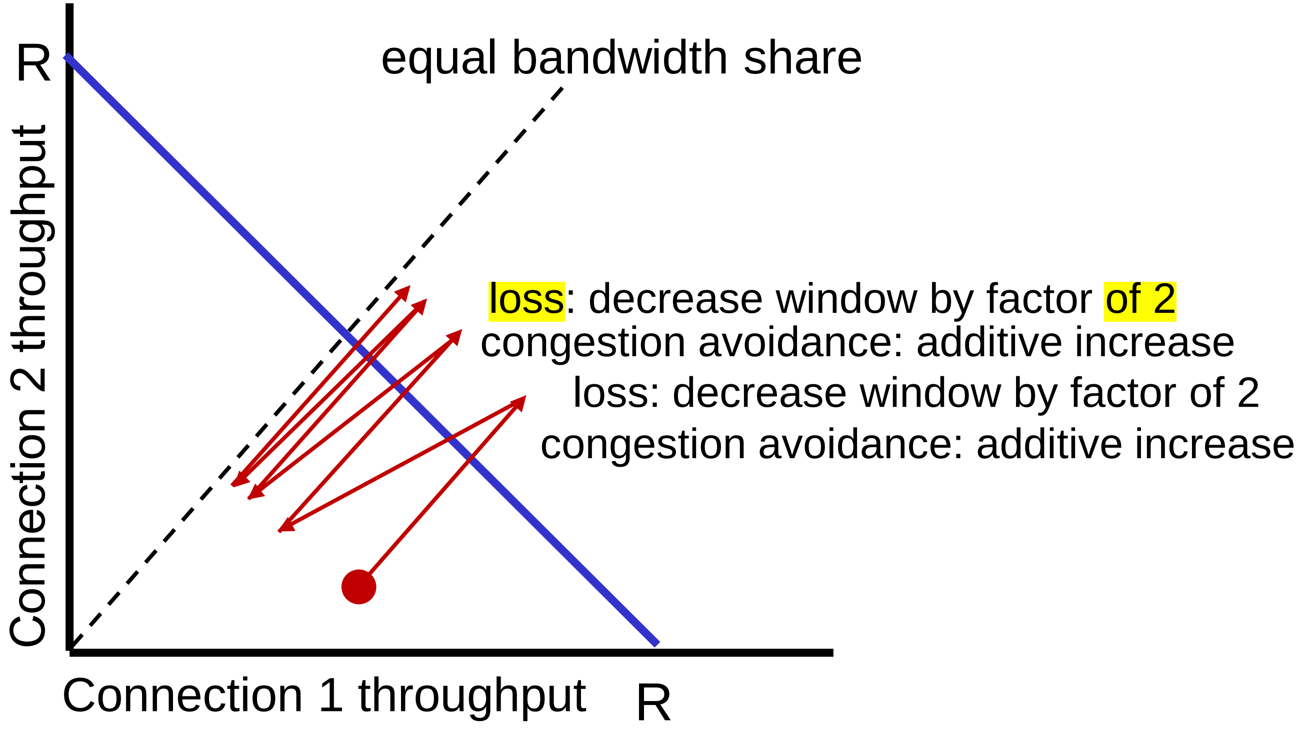
Goal:

1. Fairness: Every user should get a fair share of link
2. Capacity: Network link bandwidth should be well utilized

Q: is TCP Fair?

Example: two competing TCP sessions:

- additive increase – Increases Efficiency
- multiplicative decrease decreases throughput proportionally - Increases Fairness



Is TCP fair?

A: Yes, under idealized assumptions:

- same RTT
- fixed number of sessions

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
- there is no "Internet police" policing use of congestion control

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

TCP Options

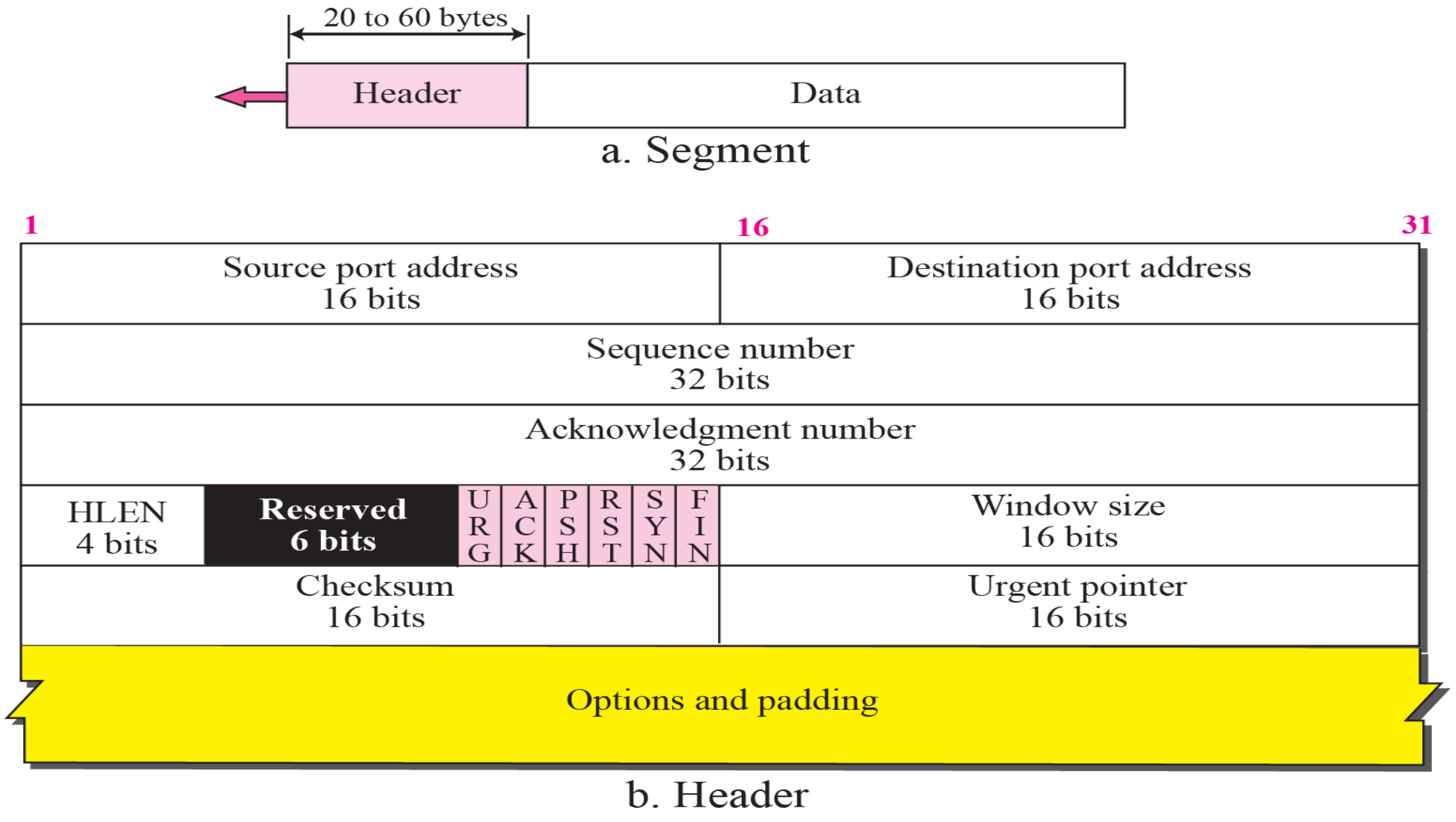
- The TCP header can have up to 40 bytes of optional information. Options convey additional information to the destination.

- We can define two categories of options: 1-byte options and multiple-byte options.

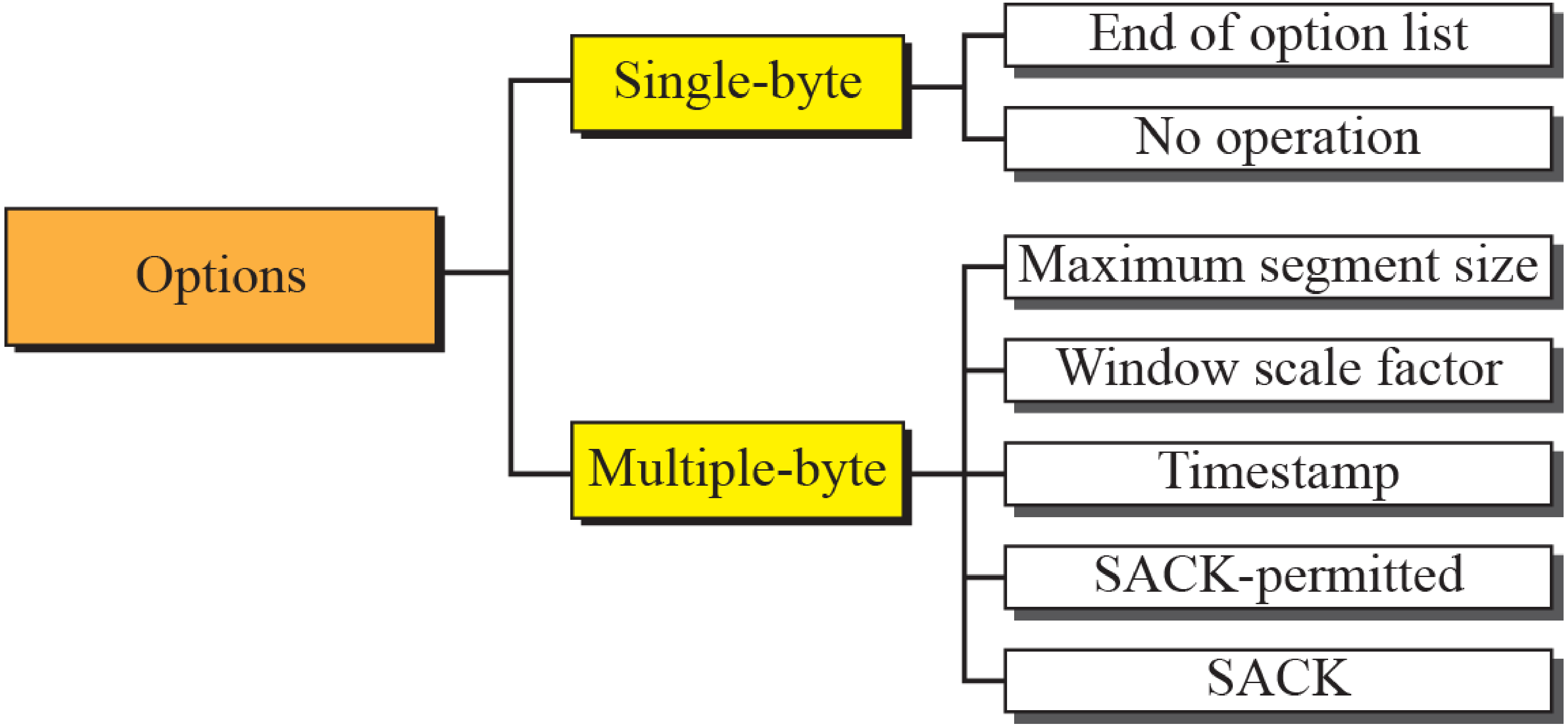
- The first category contains two types of options: end of option list and no operation.

- The second category, in most implementations, contains five types of options: maximum segment size, window scale factor, timestamp, SACK-permitted, and SACK.

TCP segment with options format

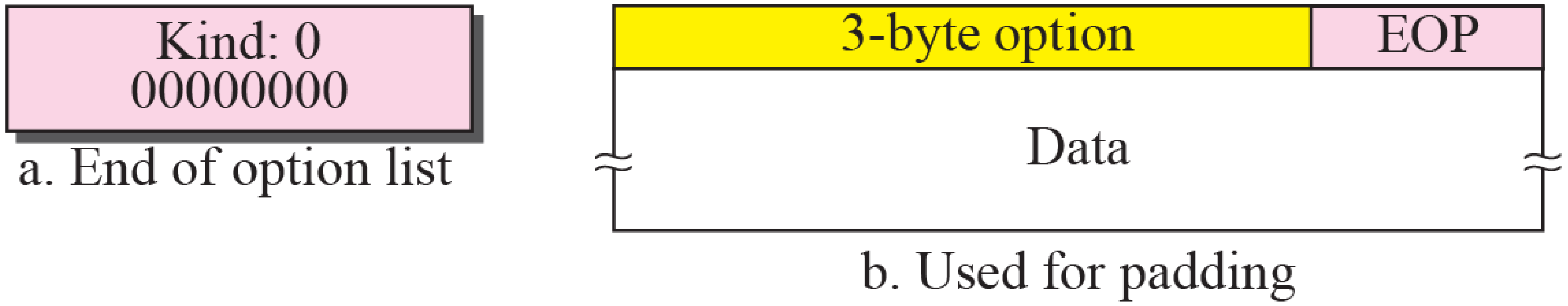


Options



Options
Single-byte
Multiple-byte
End of option list
No operation
Maximum segment size
Window scale factor
Timestamp
SACK-permitted
SACK

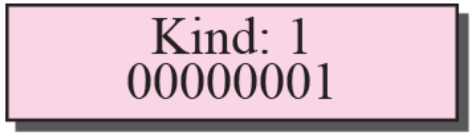
End-of-option option



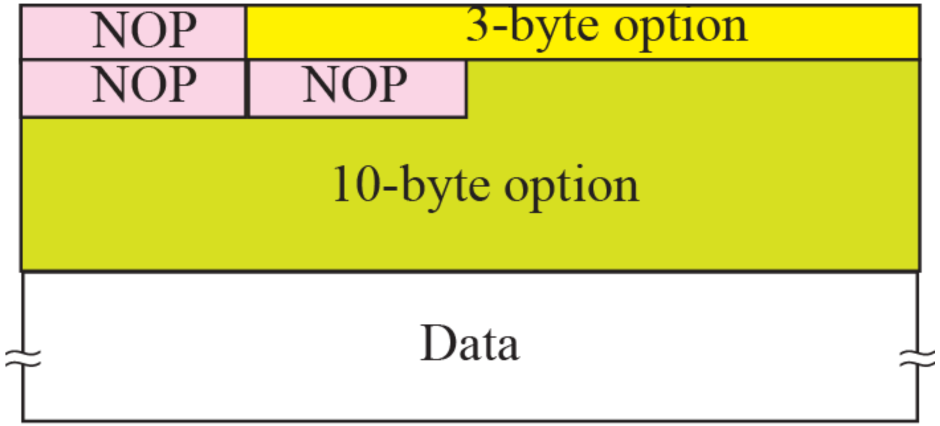
EOP can be used only once.

No-operation option

- It is used to separate the different options used within the TCP Option field.



a. No operation option



b. Used to align beginning of an option

NOP can be used more than once.

Minimum-segment-size option

1) The MSS (Maximum Segment Size) is defined as the largest block of data that a sender using TCP will send to the receiver.
2) When a connection is initiated a SYN segment is sent, in the process of sending a SYN segment, the sender has the option of announcing its MSS.
3) If a sender doesn't use the options field to declare the MSS then TCP assumes a default of 536bytes (minus the 20 byte TCP header).

Kind: 2 00000010	Length: 4 00000100	Maximum segment size
1 byte	1 byte	2 bytes

The value of MSS is determined during connection establishment and does not change during the connection.

Window-scale-factor option

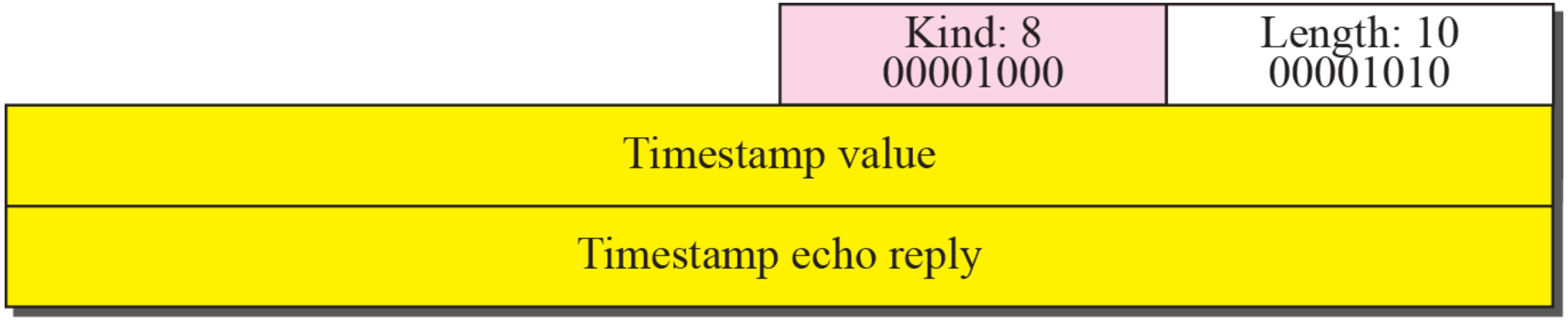
1. For lines with high bandwidth, high delay or high bandwidth-delay product, 64KB is not enough to keep the sender busy all the time.
2. In RFC 1323, the use of the options field is proposed permitting the use of a window scale factor.
3. This scale factor permits shifting the window size up to 14 bits to the left therefore permitting window sizes of up to $2^{(16+14)} = 2^{30}$ bytes.

Example window size =65535, and scale factor = 2 (2^ 2 = 4 multiply by 4 to window size to calculate the actual window size)

Kind: 3 00000011	Length: 3 00000011	Scale factor
1 byte	1 byte	1 byte

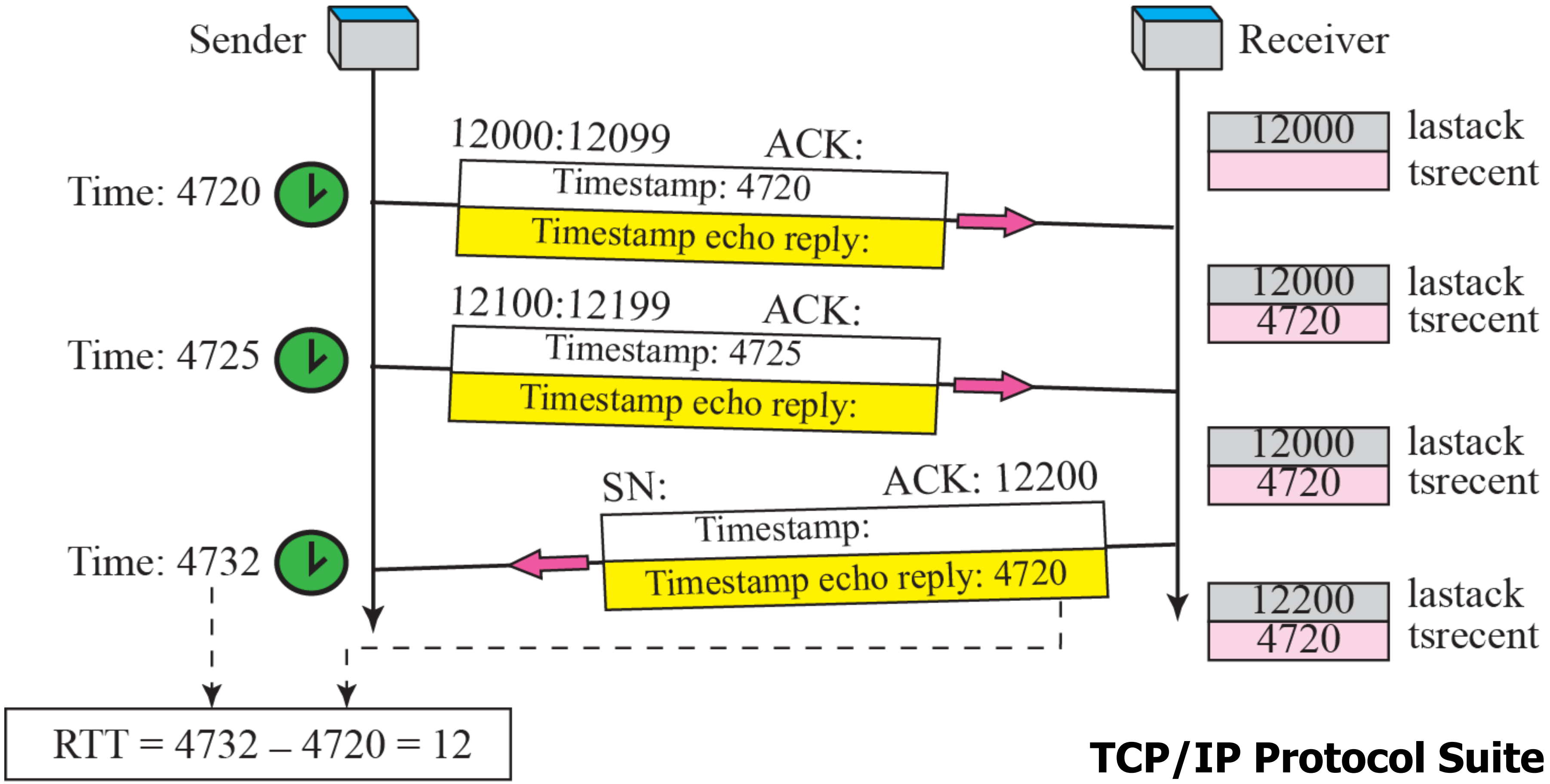
The value of the window scale factor can be determined only during connection establishment; it does not change during the connection.

Timestamp option

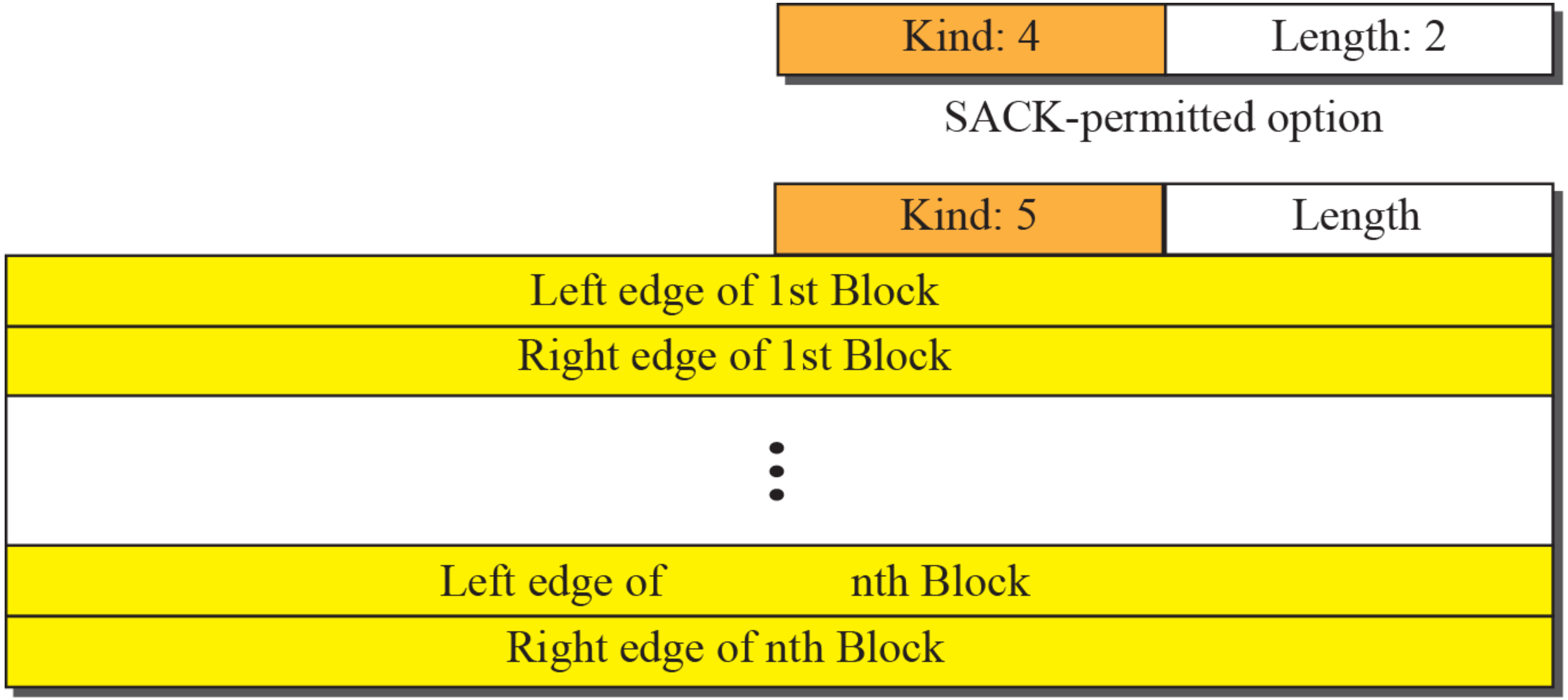


One application of the timestamp option is the calculation of round-trip time (RTT).

Figure shows an example that calculates the round-trip time for one end. Everything must be flipped if we want to calculate the RTT for the other end.

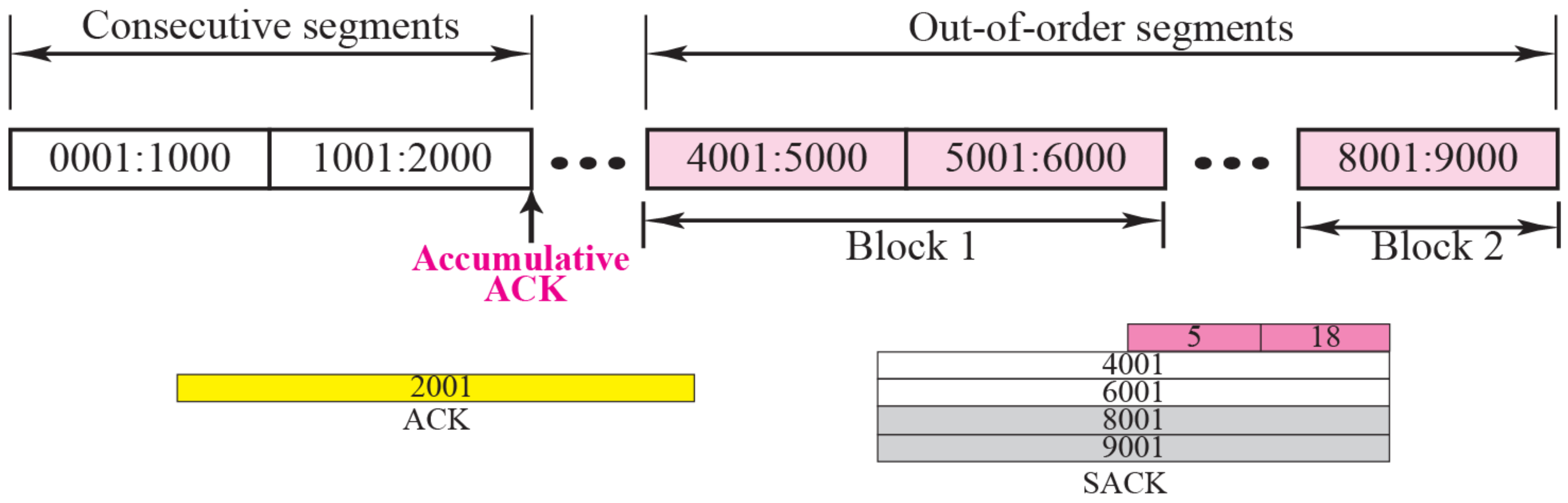


Selective ACK (*SACK)*



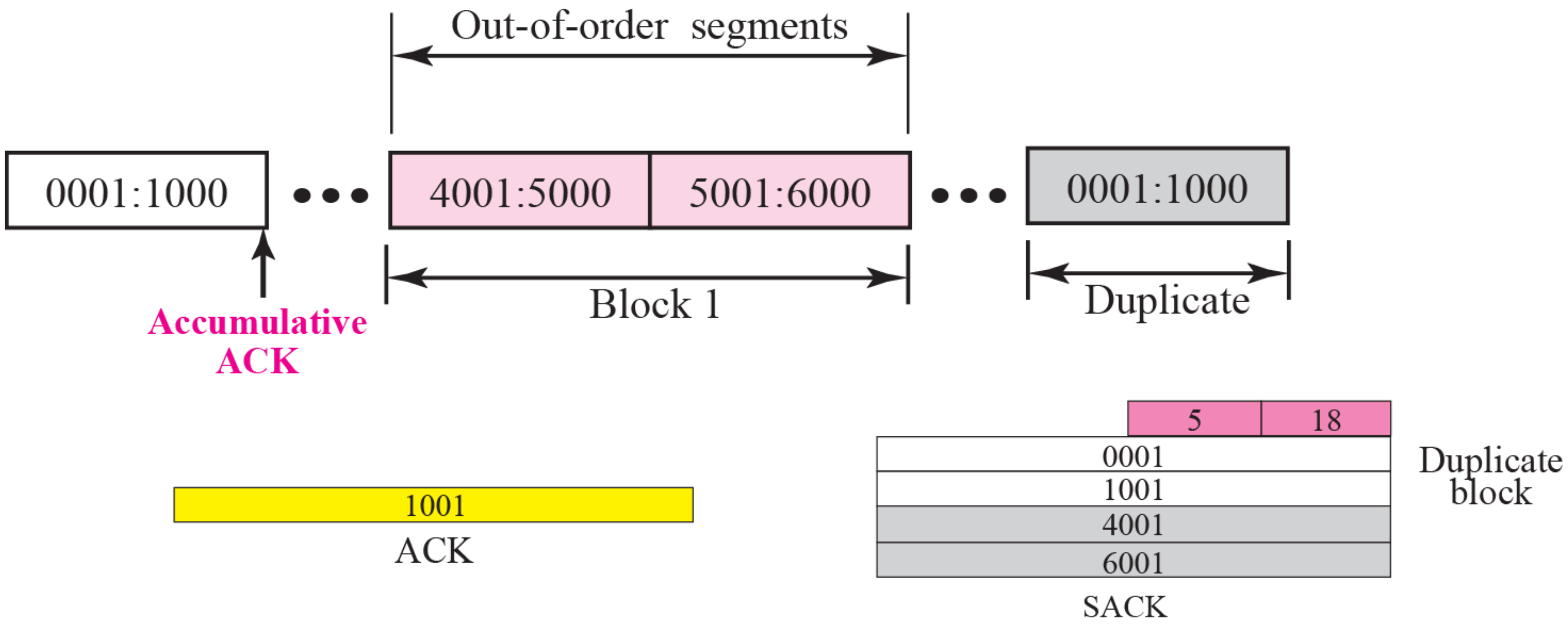
SACK option

Let us see how the SACK option is used to list out-of-order blocks. In Figure an end has received five segments of data.



TCP/IP Protocol Suite

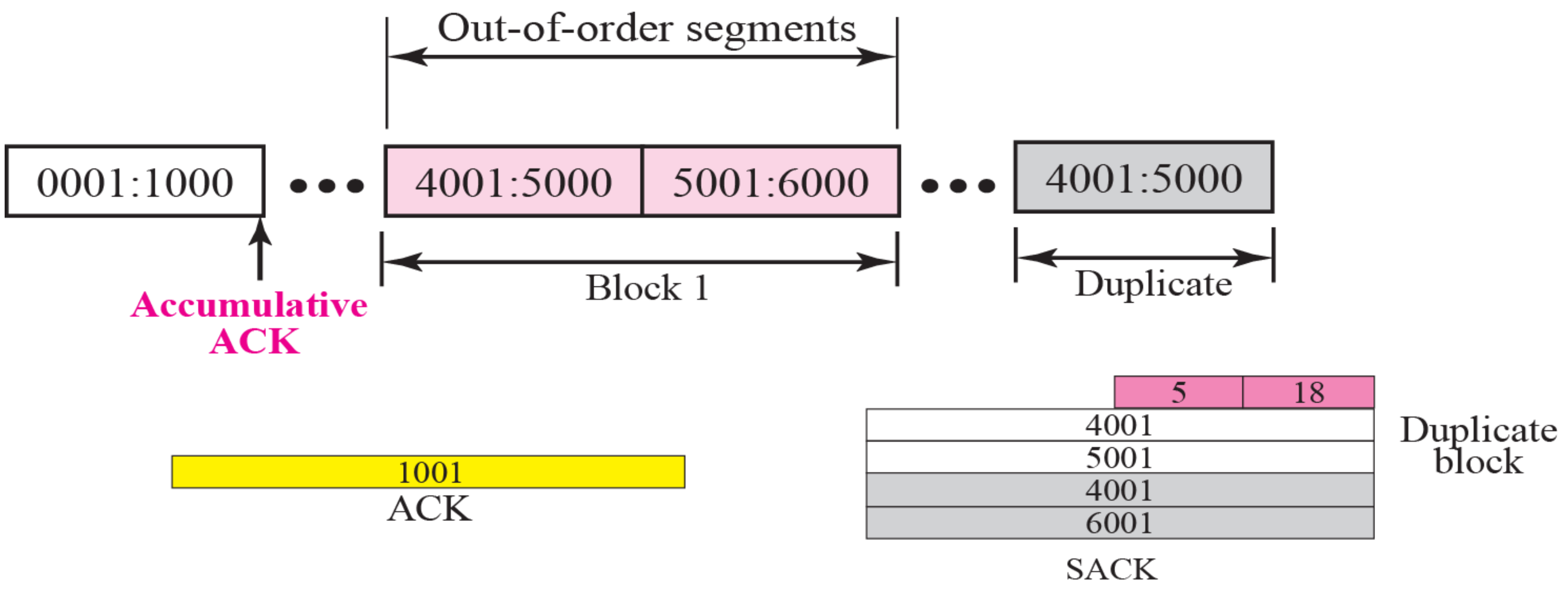
Figure shows how a duplicate segment can be detected with a combination of ACK and SACK.



TCP/IP Protocol Suite

Figure shows what happens if one of the segments in the out-of-order section is also duplicated. In this example, one of the segments (4001:5000) is duplicated.

The SACK option announces this duplicate data first and then the out-of-order block.



A New Transport Protocol

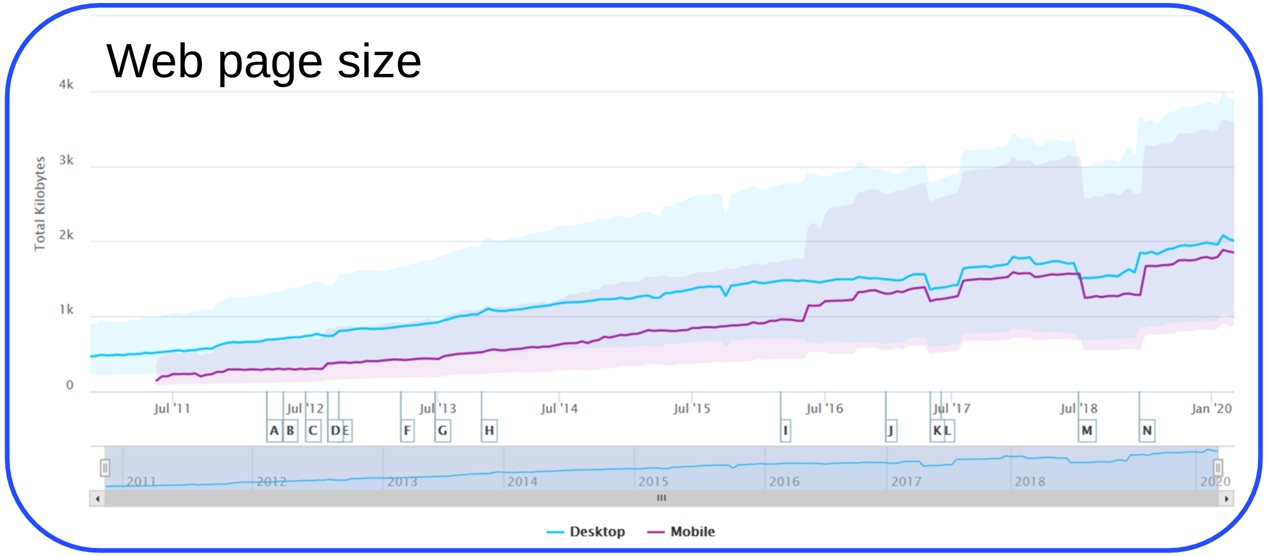
QUIC: Quick UDP Internet Connections

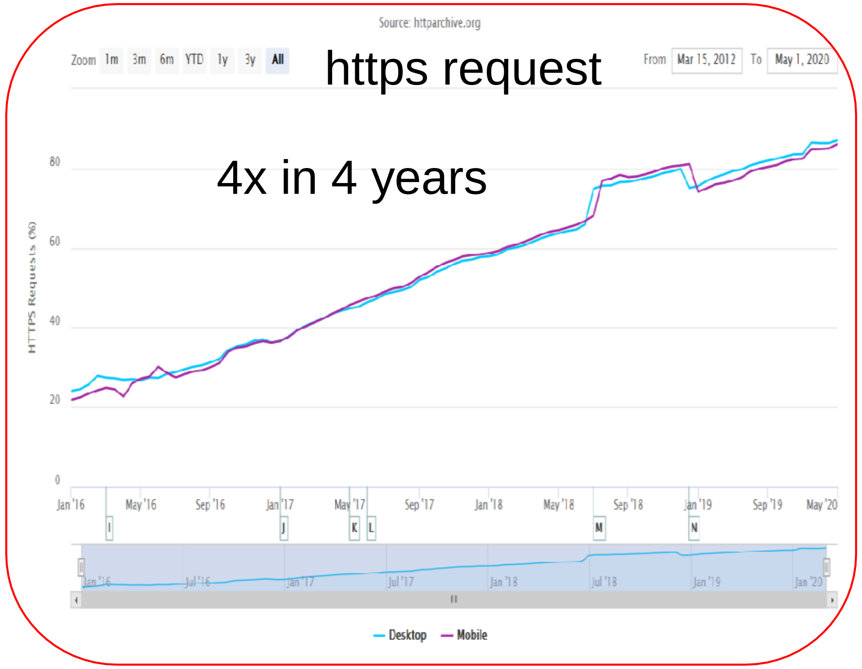
HTTP/3: HTTP over QUIC is next Generation

On 6 June 2022, IETF published HTTP/3 as a Proposed Standard in RFC 9114

Introduction: Change

- **Increasing scale of .... everything**
  - Flow size changes
  - Flow count increases (e.g., web pages)
  - Flow diversity increase (e.g., web pages)
    - Multiple connections





HTTP Network Stack

HTTP
TLS 1.2+
TCP
IP

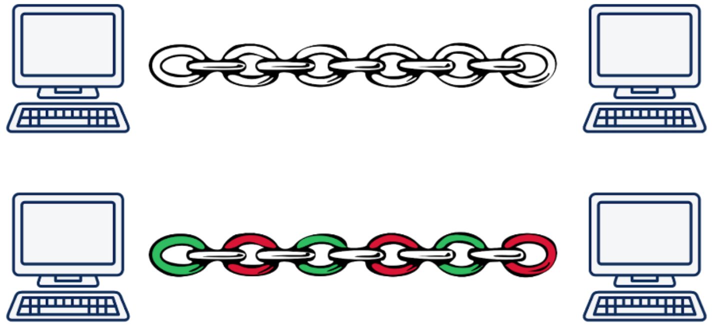
HTTP / 1.1

- January 1997
- **Many parallel TCP connection (6 connections per host name)**
- **HTTP head of line blocking**

HTTP / 2

- May 2015
- **Using Single connection per host**
- **Many parallel streams**
- **TCP head of line blocking**

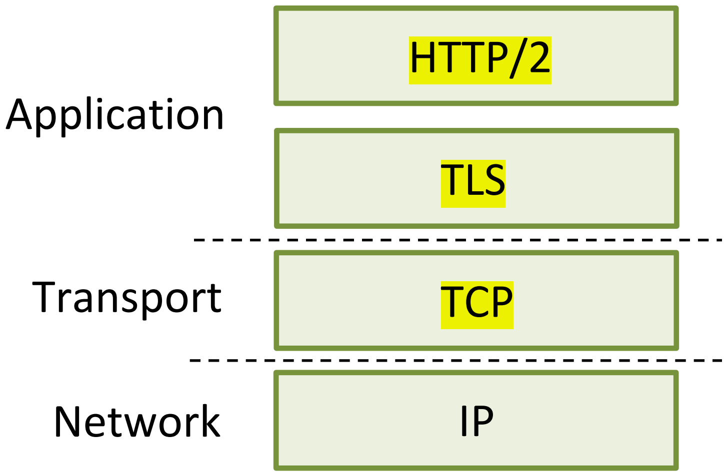
TLS – Transport Layer Security
TCP – Transport Control Protocol
IP – Internet Protocol



https://http3-explained.haxx.se/en/why-quic

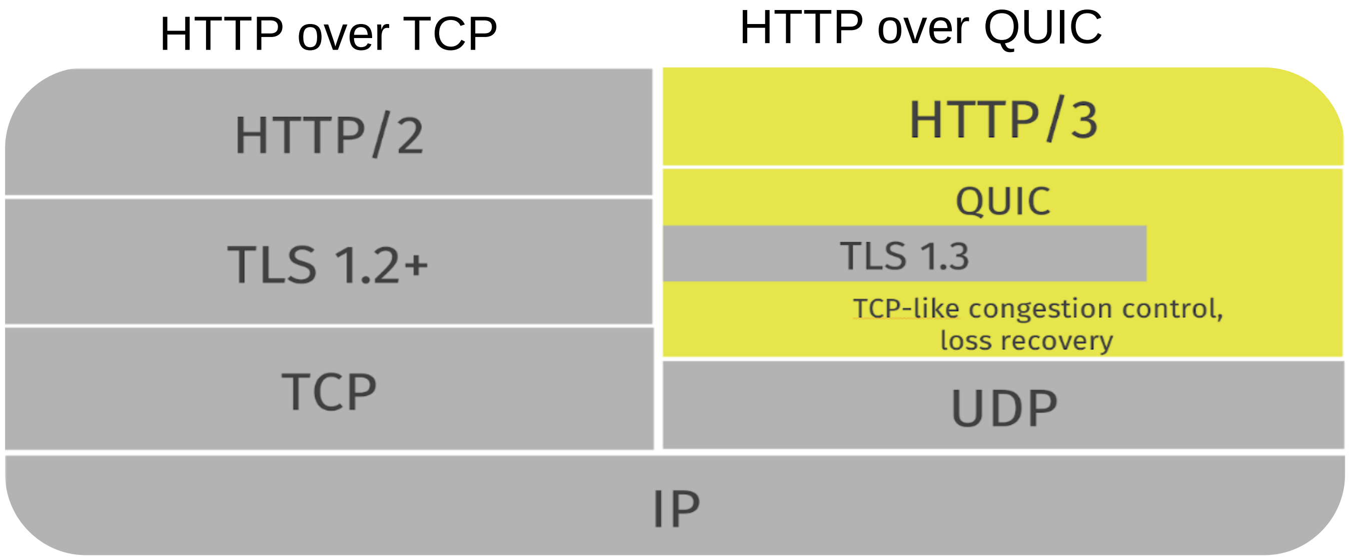
QUIC: Quick UDP Internet Connections

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



HTTP/2 over TCP

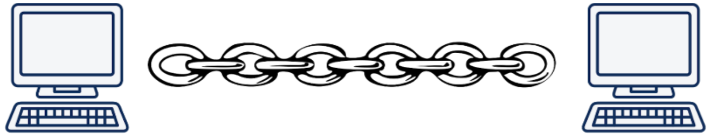
HTTP Over QUIC Network Stack



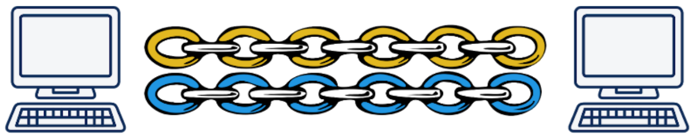
HTTP / 3

Workgroup:	QUIC
Internet-Draft:	draft-ietf-quic-http-latest
Published:	9 June 2020
Intended Status:	Standards Track
Expires:	11 December 2020
Author:	M. Bishop, Ed.
	Akamai

- **No - TCP head of line blocking**
  - **streams are independent to each other**
- **Faster handshake**
  - **Earlier data**
- **More encryption, always**
- **Over UDP** (***Connection less, No resend, No flow control***)



- Steam impact not connection impact



https://http3-explained.haxx.se/en/why-quic

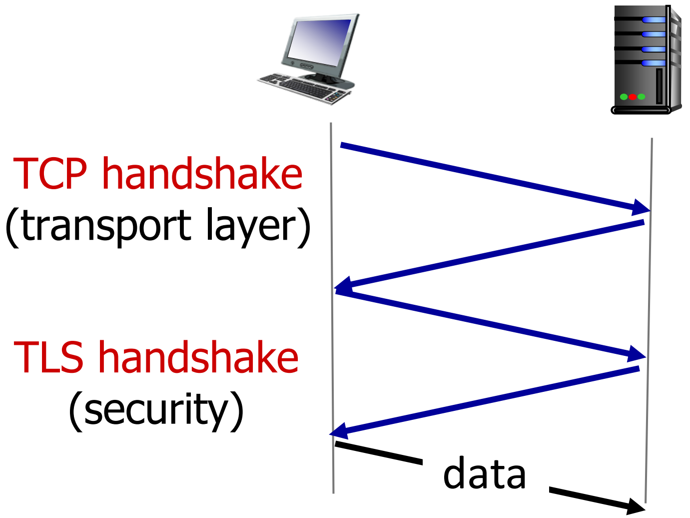
QUIC: Quick UDP Internet Connections

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

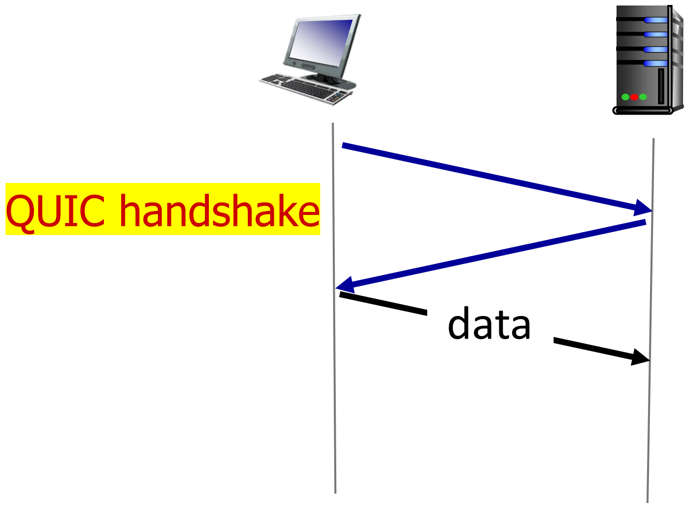
- multiple application-level "streams" multiplexed over single QUIC connection
  - separate reliable data transfer, security
  - common congestion control

QUIC: Connection establishment



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

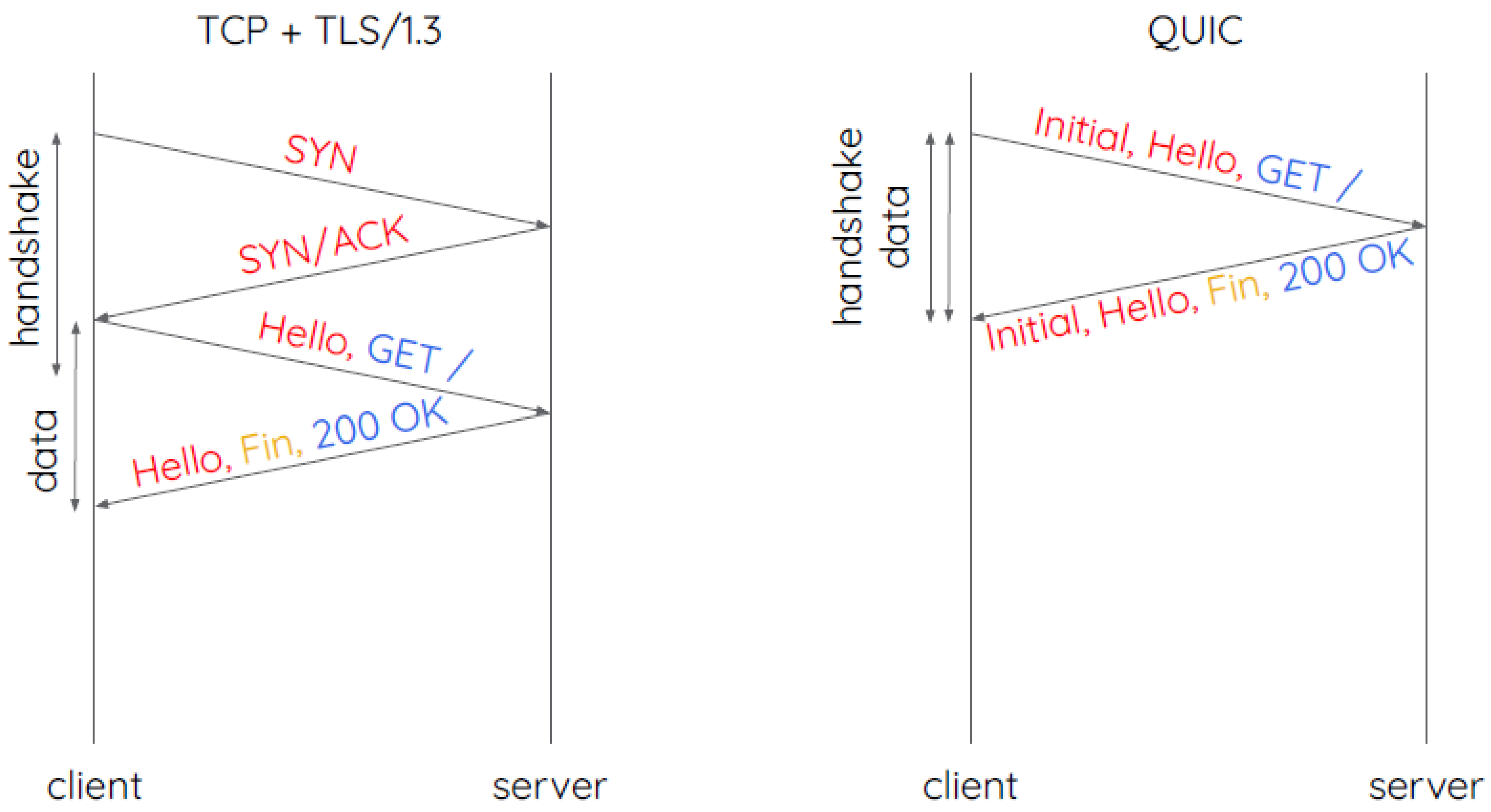
- 2 serial handshakes



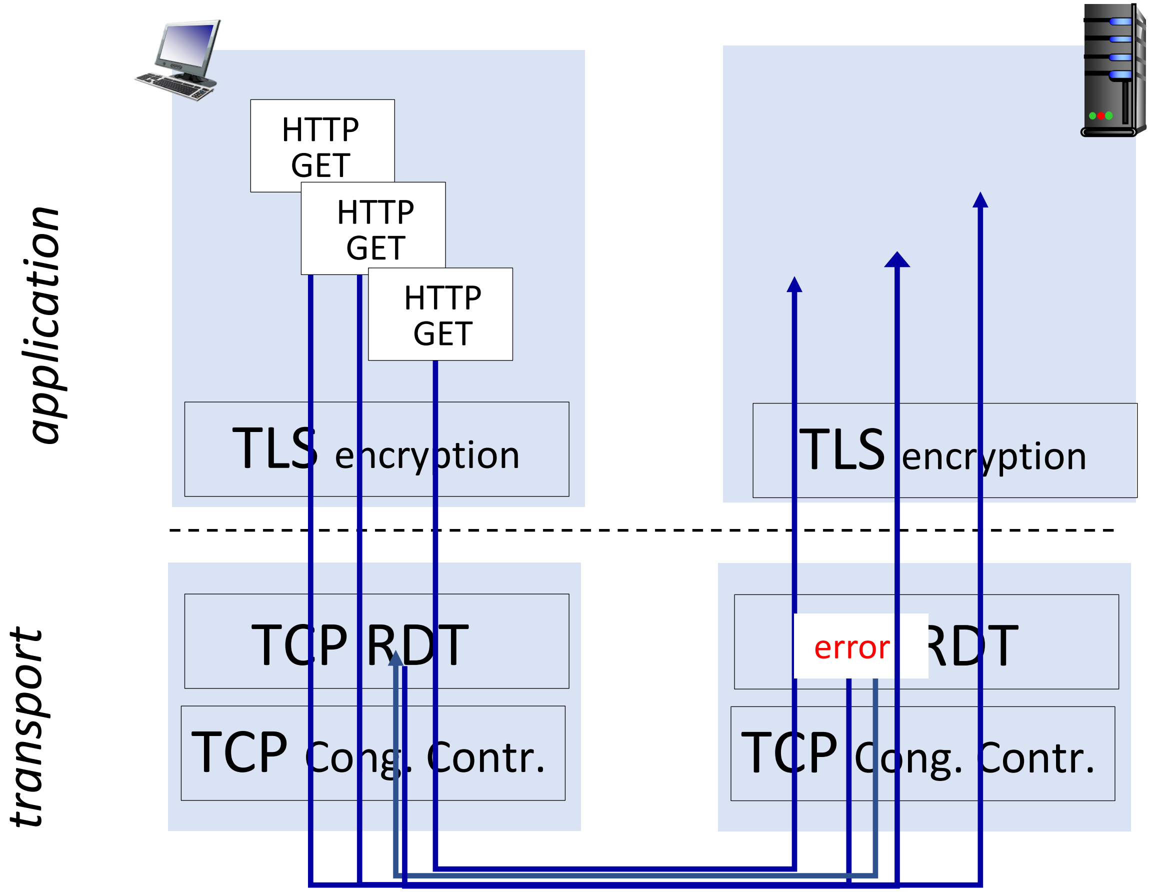
QUIC: reliability, congestion control, authentication, crypto state

- 1 handshake

Subsequent Connection to the same server



QUIC: streams: parallelism, no HOL blocking



(a) HTTP 1.1

QUIC Status

- In May 2021, the IETF standardized QUIC in RFC 9000.

Implementations:
Apple, Facebook, Fastly, Firefox, F5, Google, Microsoft ...

Server deployments have been going on for a while
Akamai, Cloudflare, Facebook, Fastly, Google ...

Clients are at different stages of deployment
Chrome, Firefox, Edge, Safari
iOS, MacOS

Source: SIGCOMM 2020, QUIC Tutorial, https://conferences.sigcomm.org/sigcomm/2020/tutorial-quic.html