COMP 3721
Introduction to Data Communications

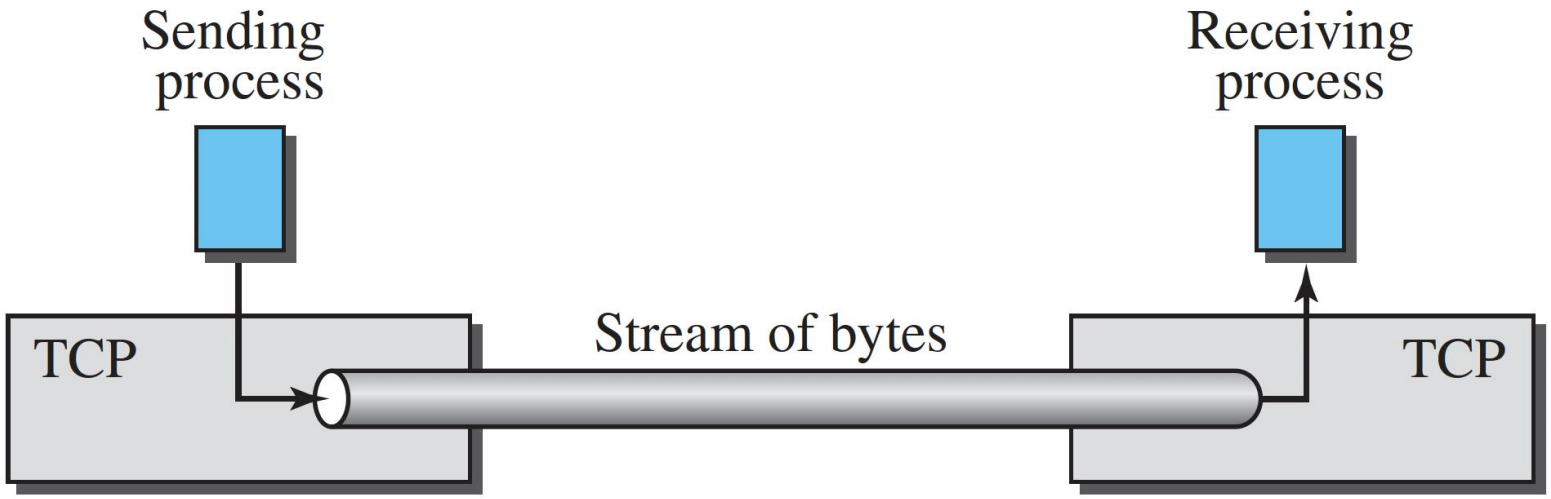
13 - Week 13

Learning Outcomes

- By the end of this lecture, you will be able to
  - Explain how TCP works and what are its characteristics.

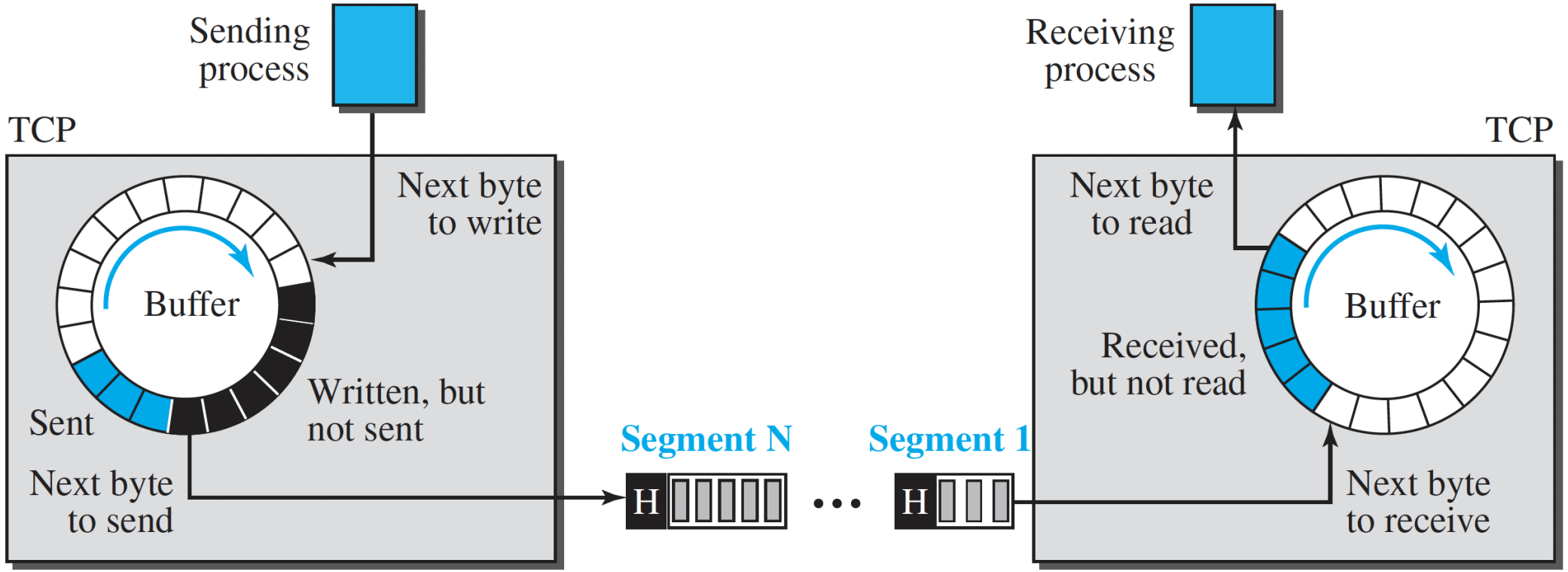
TCP – A Stream-Oriented Protocol

- TCP, unlike UDP, is a **stream-oriented** (**byte-oriented**) **protocol**.
  - The sending process delivers data as **a stream of bytes** and the receiving process obtains data as a stream of bytes



TCP – Sending and Receiving Buffers

Buffers are necessary for **flow control** and **error control**



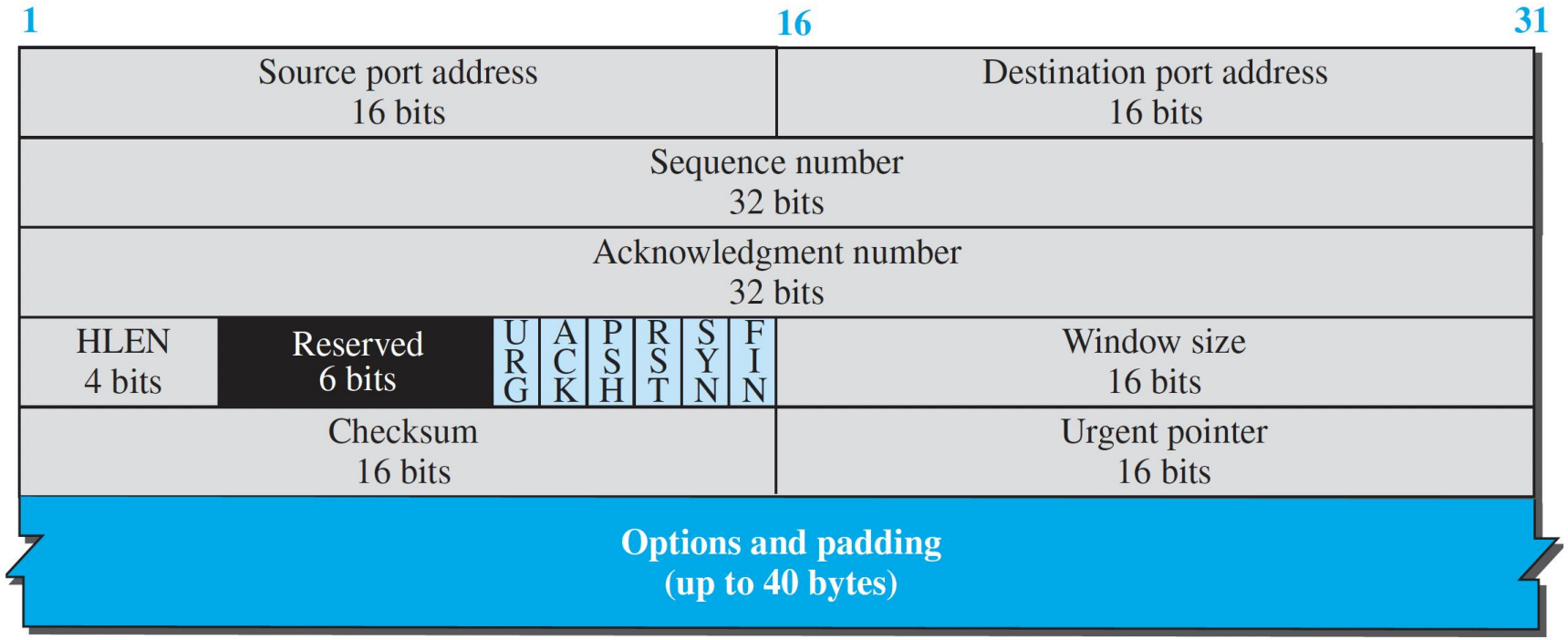
Note that segments are not necessarily all the same size.

TCP – Numbering System

- TCP numbers all data bytes (octets) that are transmitted in a connection.
- Numbering is **independent** in each direction.
- Byte numbering is used for **flow and error control**.
- Two fields in the segment header
  - **Sequence number**
  - **Acknowledgment number**
  - Each of the above fields refer to a **byte number** and not a segment number.

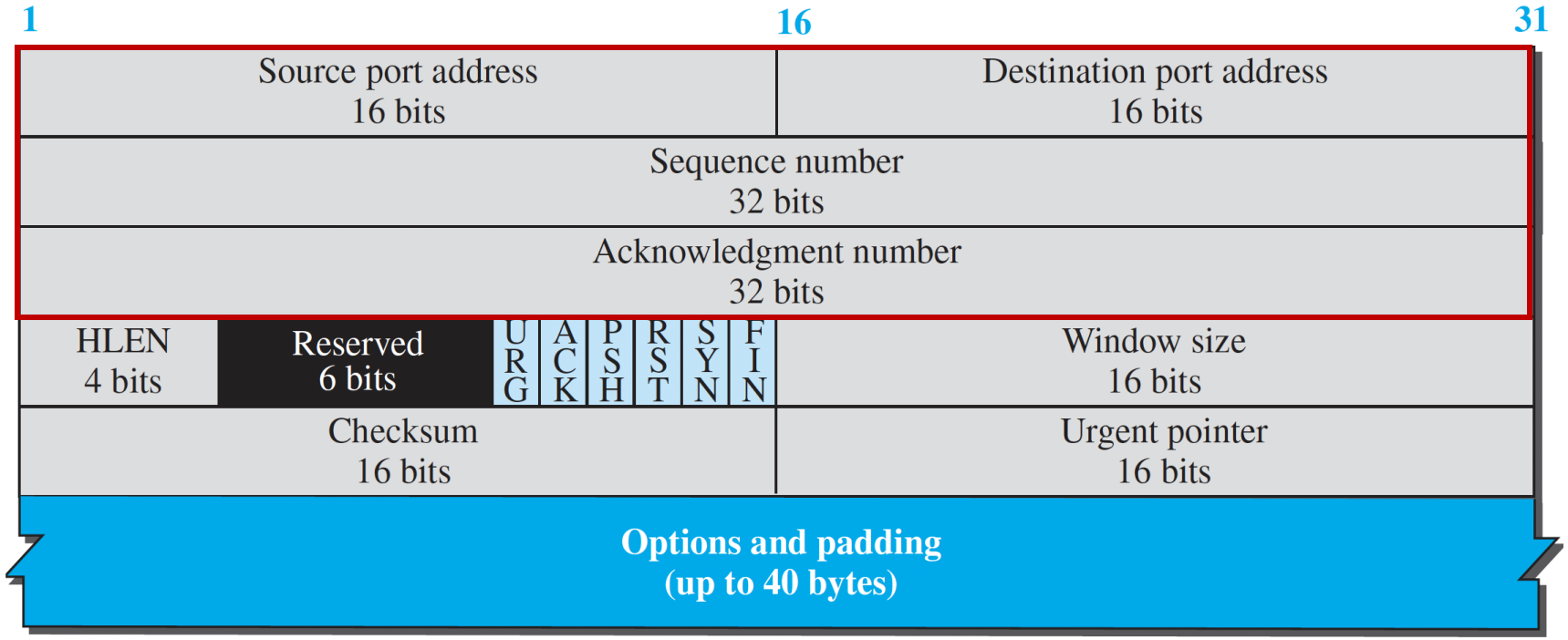
TCP Segment – Format

a. Segment



b. Header

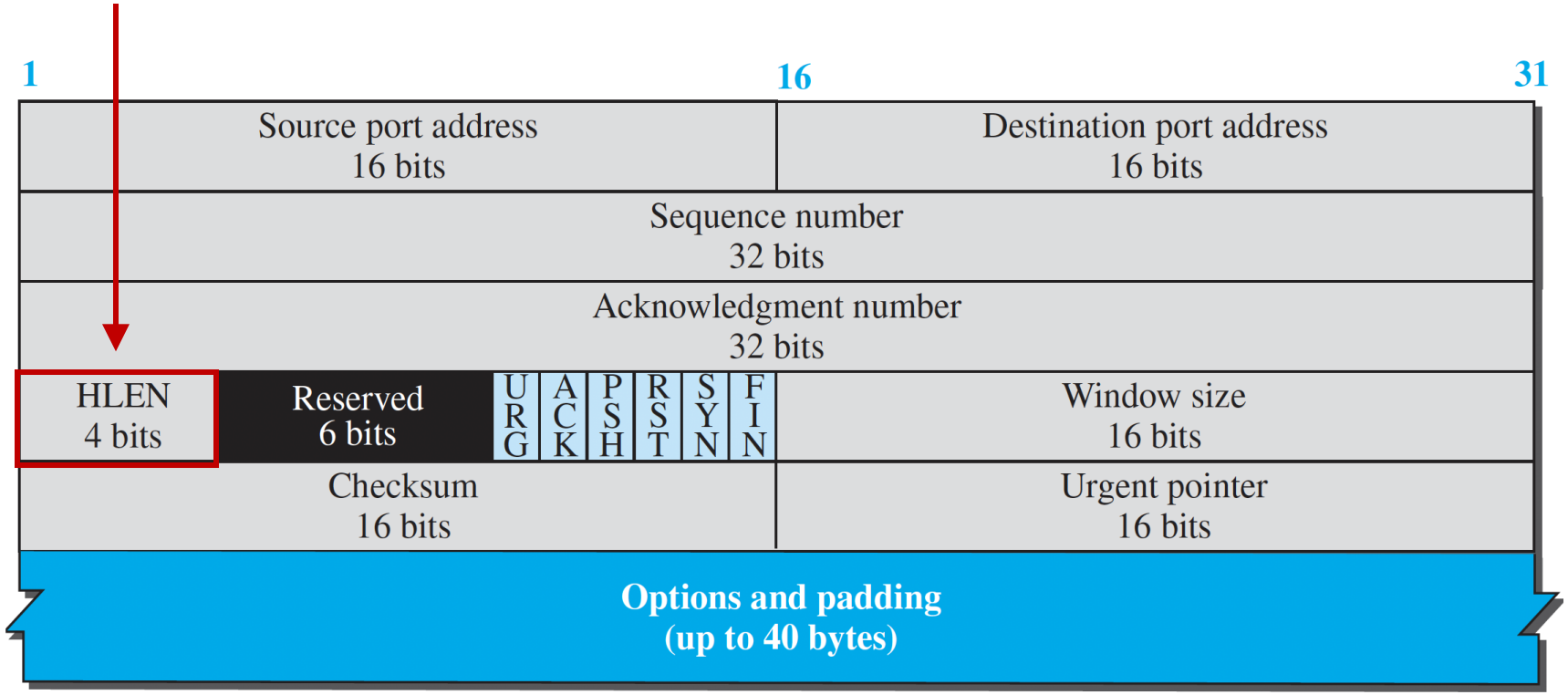
TCP Segment – Format



b. Header

TCP Segment – Format

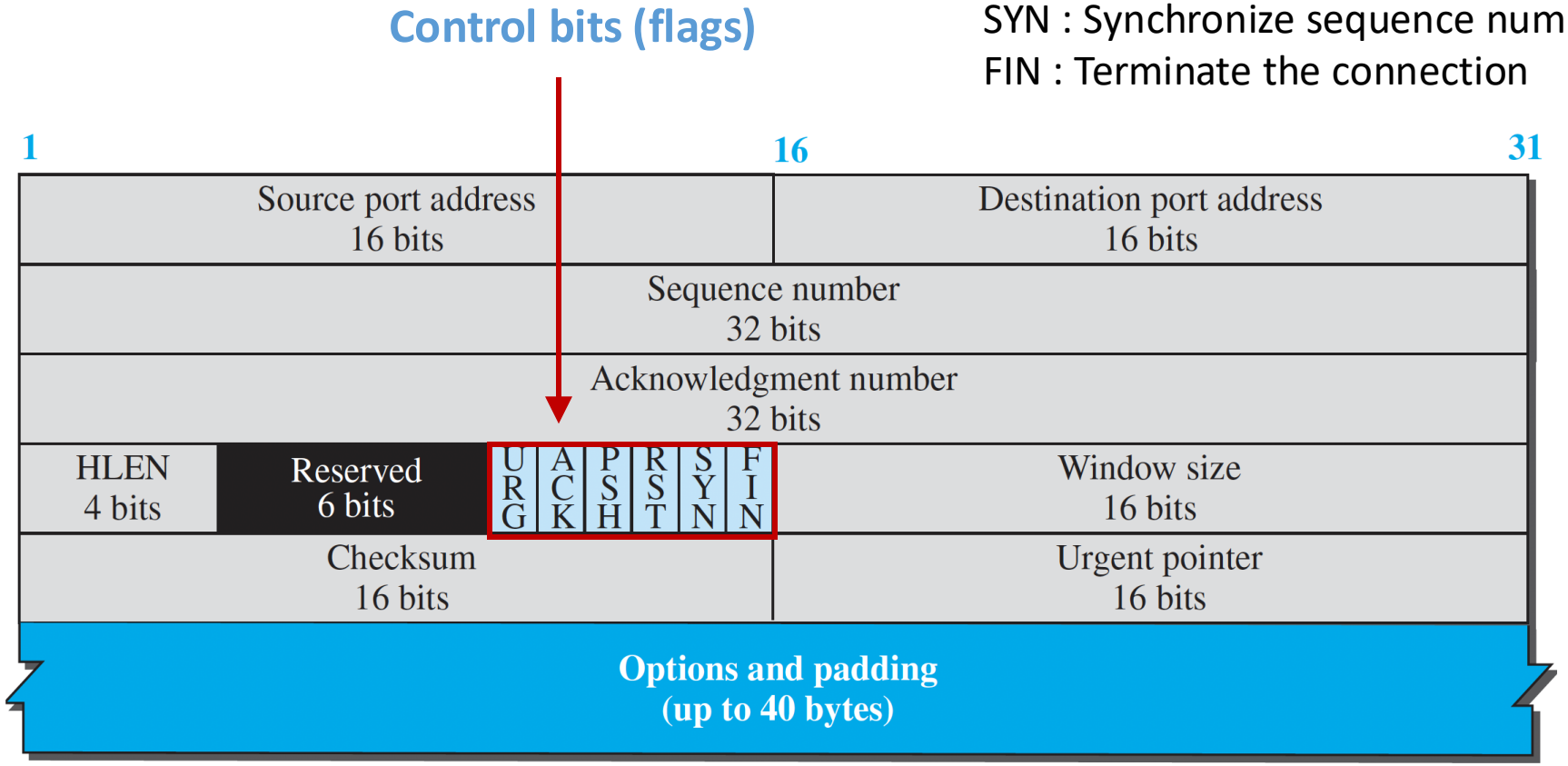
The length of the TCP header in 4-byte words



b. Header

TCP Segment – Format

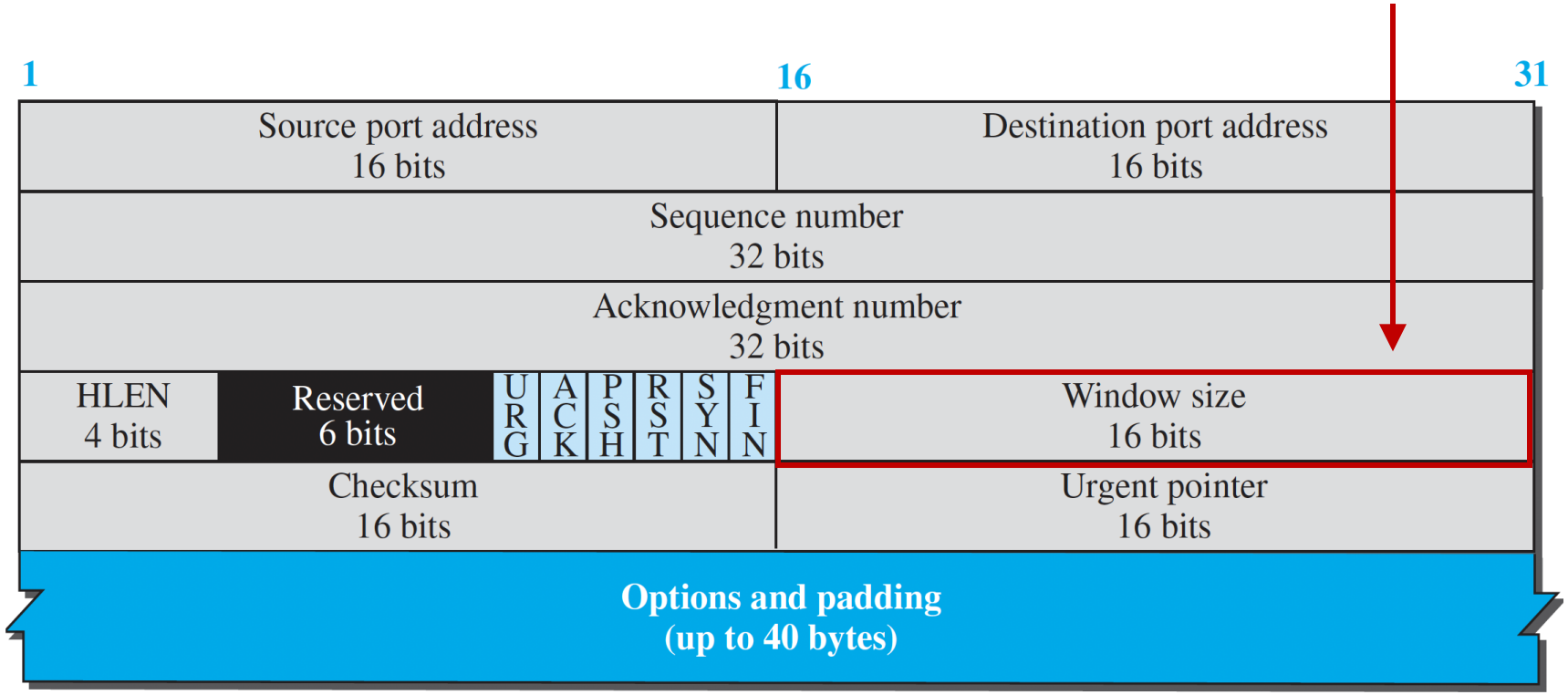
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH : Request for push
RST : Reset the connection
SYN : Synchronize sequence numbers
FIN : Terminate the connection



b. Header

TCP Segment – Format

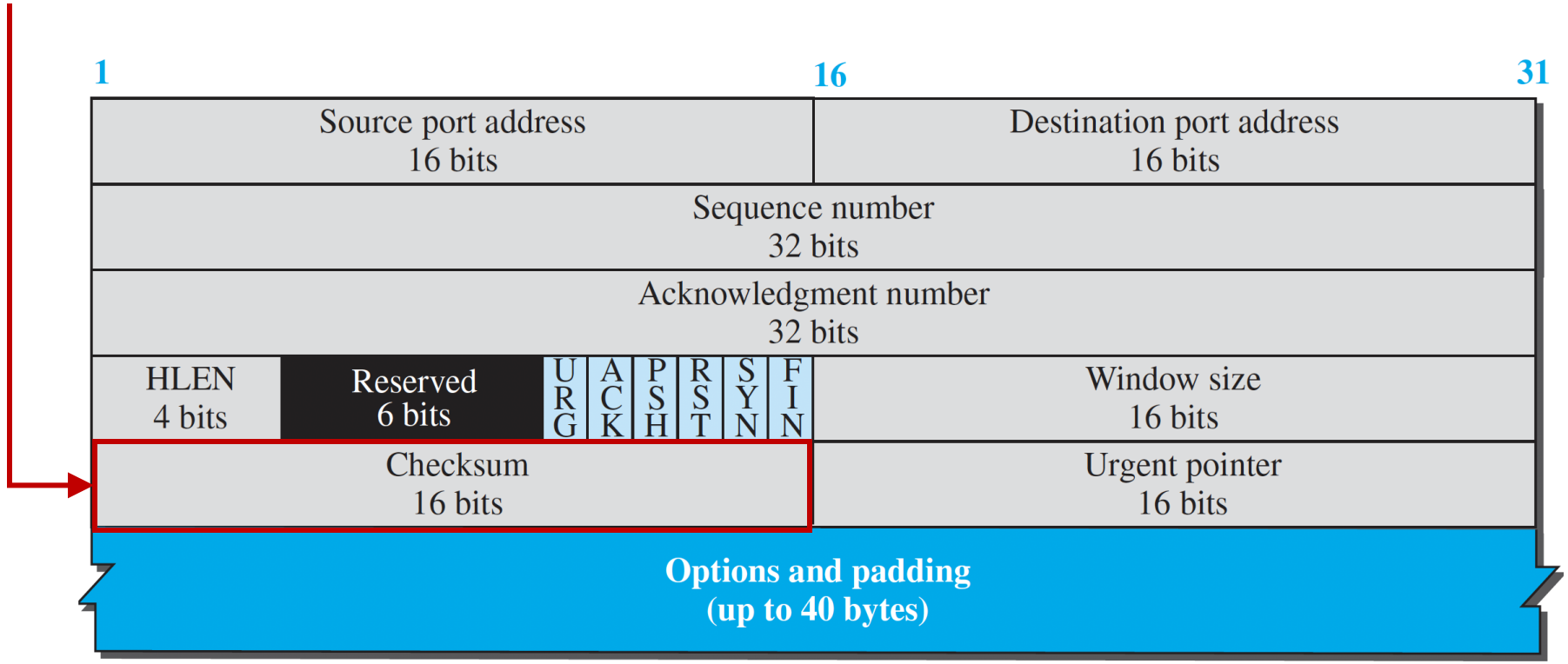
Receiving window size (rwnd) and determined by the receiver. It is used for flow control. (The num of bytes the receiver is willing to accept)



b. Header

TCP Segment – Format

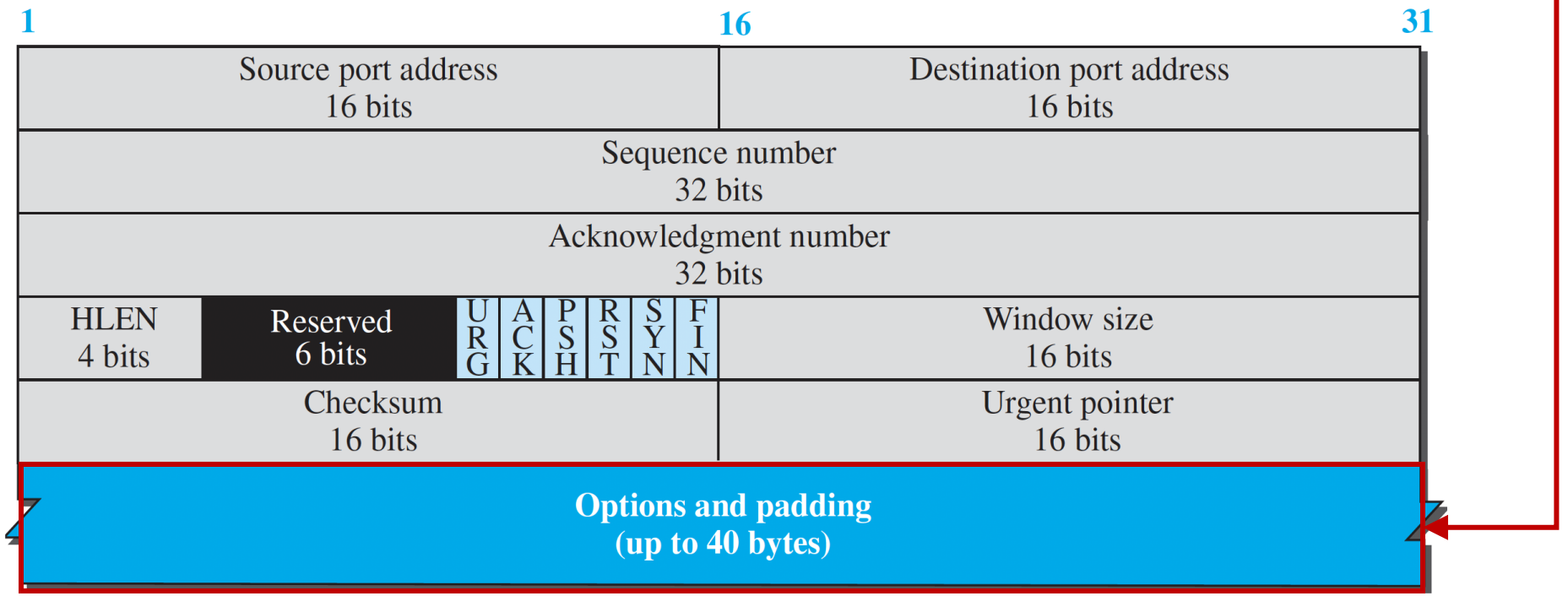
Unlike UDP, the use of checksum is mandatory in TCP.



b. Header

TCP Segment – Format

Optional field, e.g., when a sender and receiver negotiate the maximum segment size (MSS)

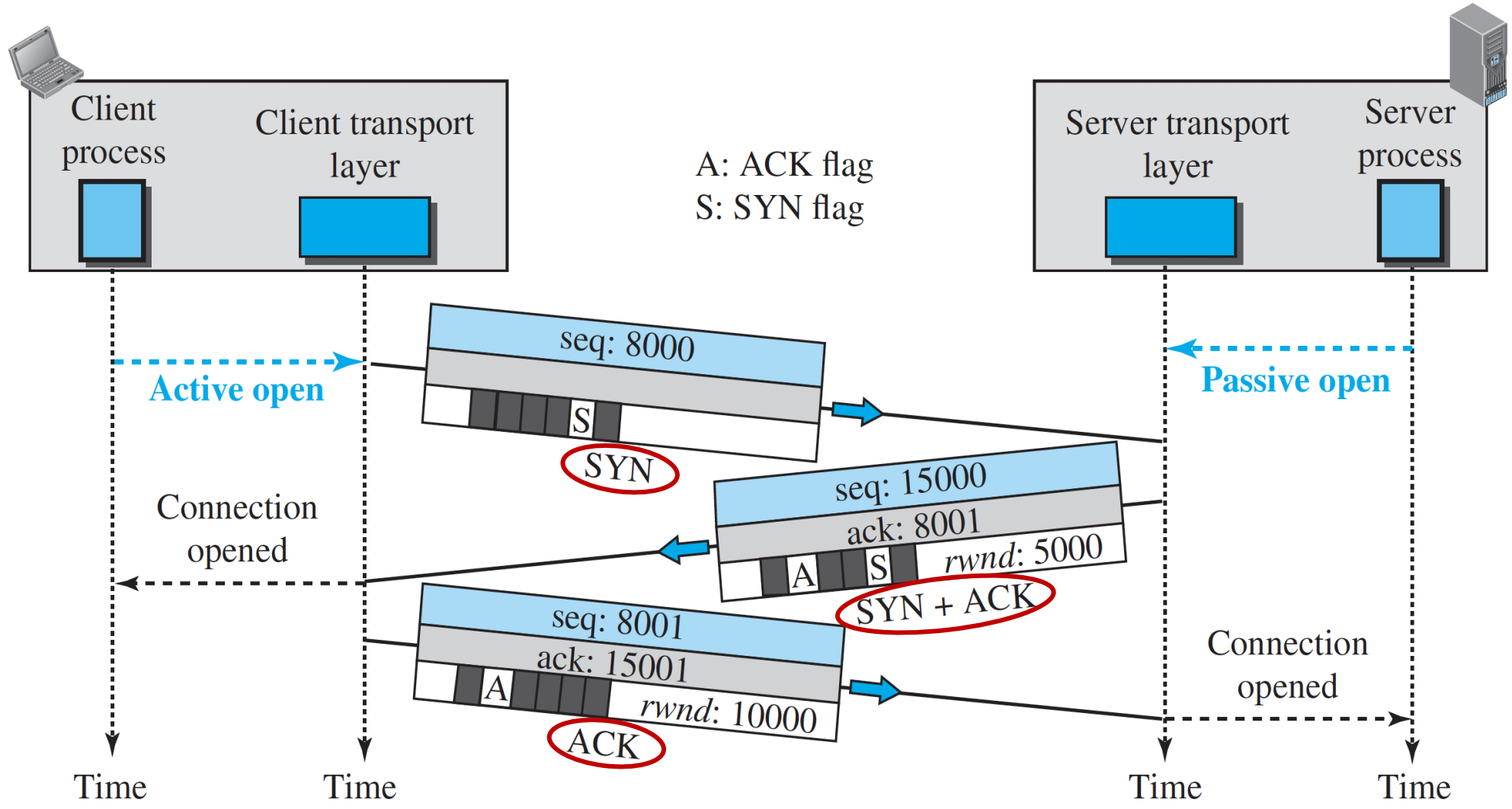


b. Header

TCP (Transmission Control Protocol)

- TCP is a **connection-oriented** protocol.
- TCP explicitly defines **connection establishment**, **data transfer**, and **connection teardown** phases (**connection-oriented service**).
  - A **single logical pathway** is used for the segments belonging to the same message.
- The connection is **logical**, and data is exchanged in **both directions** (**full-duplex service**).

TCP – Connection Establishment

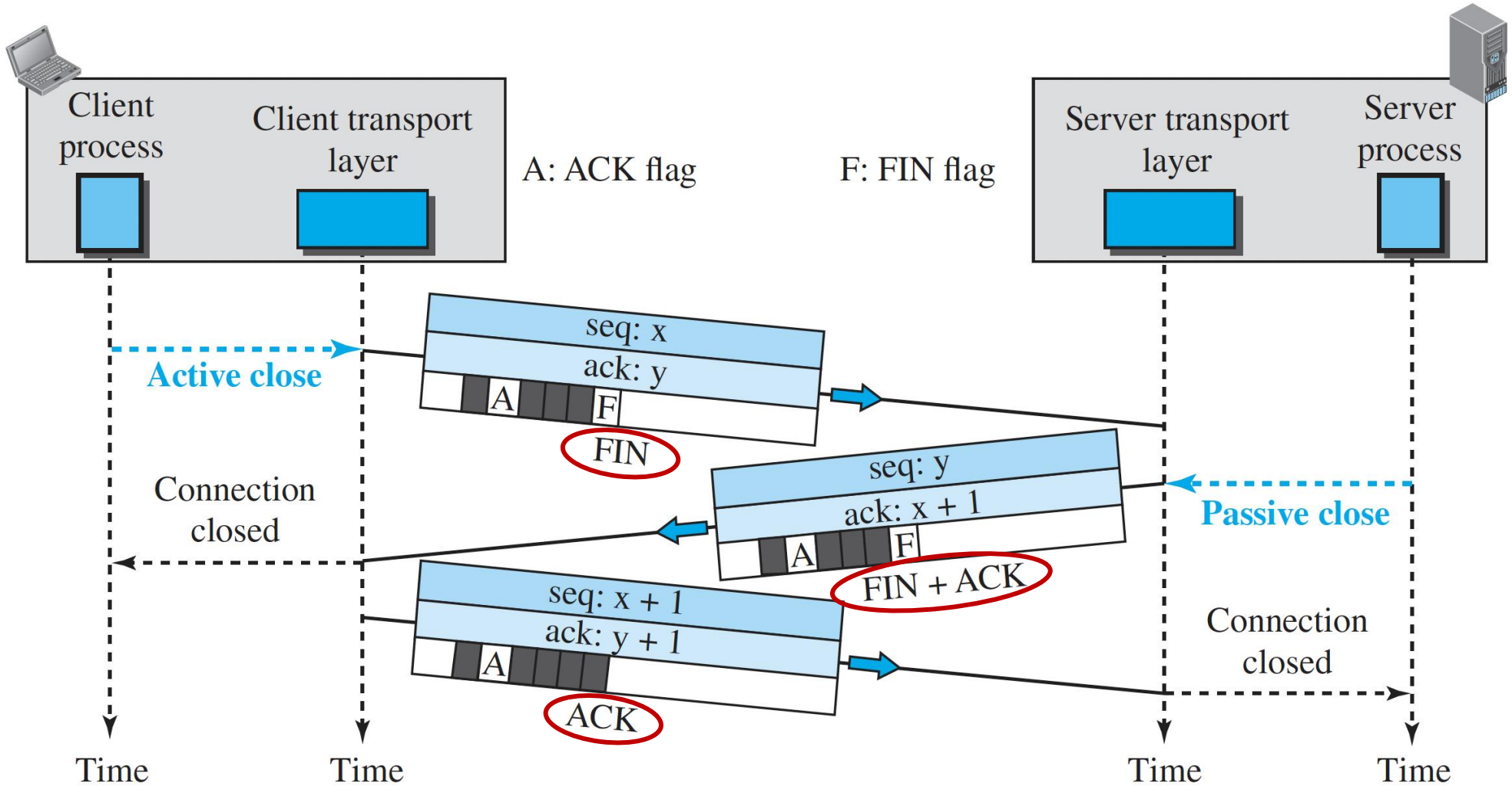


TCP connection-establishment procedure is often referred to as a **three-way handshake.**

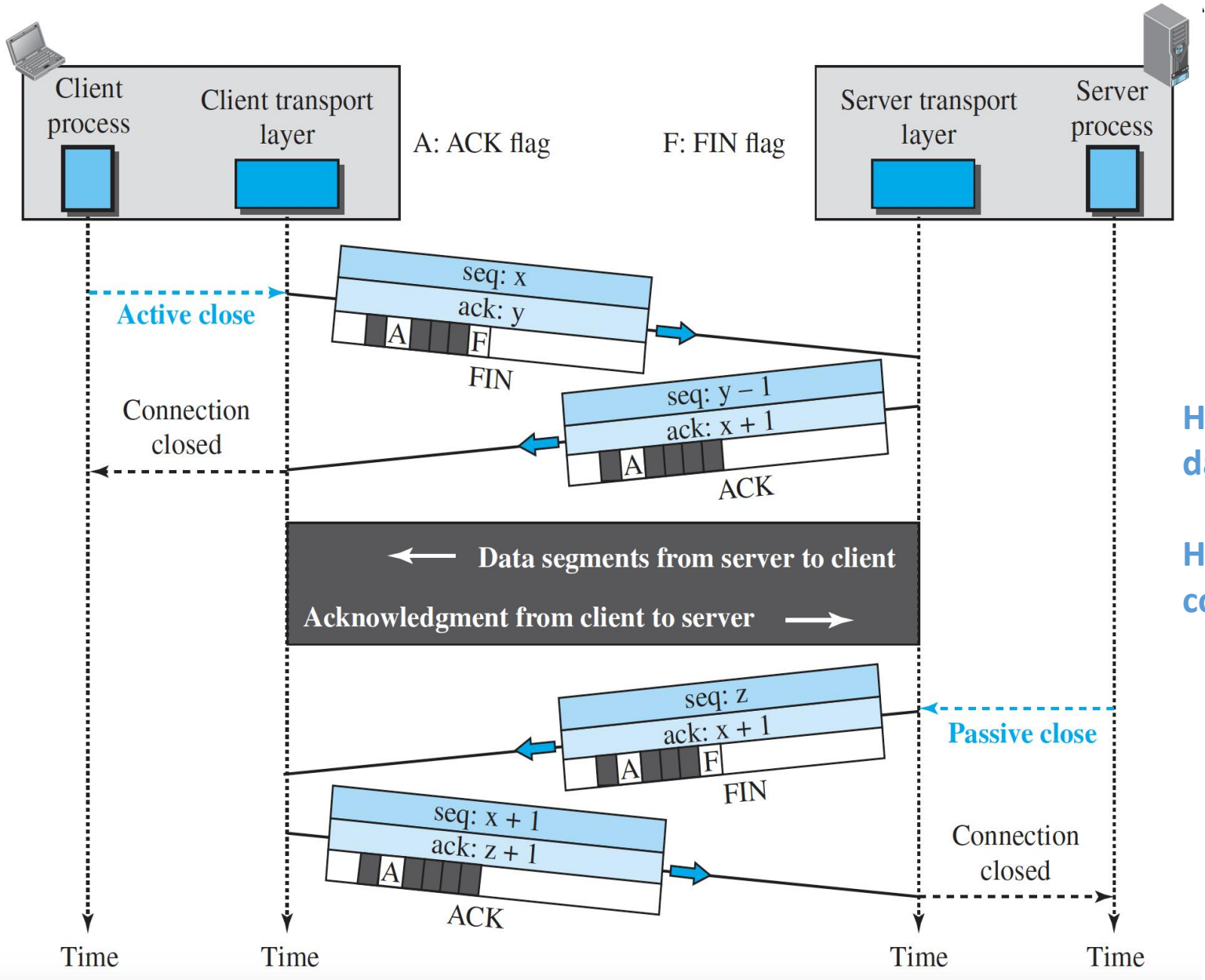
TCP – Connection Termination

- Either of the two parties involved in exchanging data (client or server) can close the connection, although it is **usually initiated by the client**.
- Most implementations today allow two options for **connection termination**:
  1. Three-way handshaking
  2. Four-way handshaking with a half-close option

TCP – Connection Termination



TCP – Connection Termination (Half-Close)



Half-close: one end can stop sending data while still receiving data.

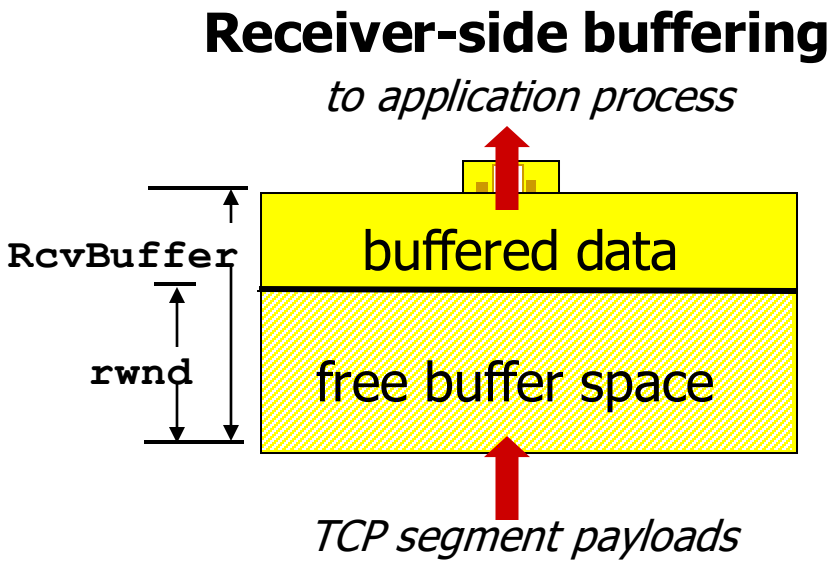
Here, the client half-closes the connection by sending a FIN segment

TCP Flow Control

- **Flow control** is used to **eliminate the possibility** of the sender **overflowing** the receiver's buffer.
- **TCP sender** (at each side of the connection) maintains a variable called **receive window** (**rwnd**).
  - Gives the sender an idea of how much free buffer space is available at the receiver.
  - Initial value of **rwnd** is equal to the **receive buffer** size (RcvBuffer).

TCP Flow Control

- **Receiver** "advertises" **free buffer space** by including **rwnd** value in TCP header of receiver-to-sender segments.
  - **RcvBuffer** size set via socket options (typical default is 4096 bytes).
  - Many operating systems auto-adjust RcvBuffer.



TCP Flow Control

- **Sender** limits amount of unACKed ("in-flight" or "outstanding") data to receiver's rwnd value (LastByteSent – LastByteAcked <= rwnd)
  - Guarantees receive buffer will not overflow
- At **receiver**: rwnd = RcvBuffer – [LastByteRcvd – LastByteRead]

TCP Error Control

- **TCP** is a **reliable** transport-layer protocol
- An application program that delivers a stream of data to TCP relies on TCP to deliver the entire stream to the application program on the other end **in order**, **without error**, and **without any part lost** or **duplicated**.

TCP Error Control Mechanisms

1. Acknowledgements
2. Sequence Numbers
3. Retransmission
4. Checksum

Acknowledgement

- An acknowledgement **confirms the receipt of data segments**.
- The value of the **acknowledgment** field in a segment defines the **byte number** of the **next byte** a party expects to receive.
- The acknowledgment number is **cumulative**.
  - The party takes the **last byte number** that it has received, safe and sound, **adds 1 to it**, and announces this sum as the **acknowledgment number**.

- TCP only **acknowledges** bytes up to the **first missing byte** in the stream.

Sequence Number

- The value in the **sequence number** field of a segment indicates the number assigned to the **first data byte** contained in that segment.
- The sequence number, in each direction, is indicated as follows:
  1. **ISN** (**initial sequence number**): The sequence number of the first segment (a random number between 0 and $2^{32}-1$).
  2. The sequence number of any other segment is the **sequence number** of the **previous segment** plus the **number of bytes** carried by the previous segment.
- Some segments, when carrying only **control information**, need a sequence number to allow an acknowledgment from the receiver.
  - Segments for connection establishment, termination, or abortion.
  - Each of these segments consume **one sequence number** as though it carries one byte, but there are no actual data.

Sequence Number – Example

- Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

Sequence Number – Example

- Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

Segment 1	→	Sequence Number:	10001	**Range:**	10001	to	11000
Segment 2	→	Sequence Number:	11001	**Range:**	11001	to	12000
Segment 3	→	Sequence Number:	12001	**Range:**	12001	to	13000
Segment 4	→	Sequence Number:	13001	**Range:**	13001	to	14000
Segment 5	→	Sequence Number:	14001	**Range:**	14001	to	15000

Sequence Number (Cont.)

- Detection of a **lost** segment
  - **Gaps** in the sequence numbers of the received segments.
- Detection of a **duplicate** segment
  - Received segments with **duplicate sequence numbers**.
- Detection of **out-of-order** segments
  - The receiver flags the out-of-order segments and stores them temporarily until the missing segments arrive (TCP **guarantees** that data are delivered to the process **in order**).

Retransmission

- The heart of TCP error control.
- Two types of retransmissions in TCP:
  1. Retransmission after **Time-out**
  2. Retransmission after **Three Duplicate ACK Segments**

Retransmission after Time-out

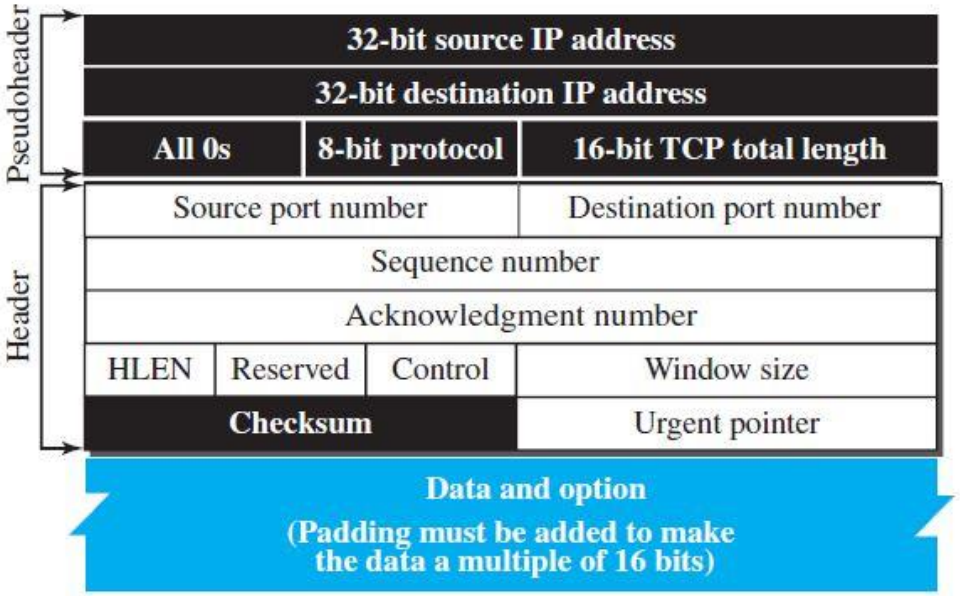
- The sending TCP maintains one **retransmission time-out** for each connection.
- When time-out occurs,
  - The segment that caused the time-out is retransmitted
  - Timer is restarted.
- The value of **retransmission time-out** is **dynamic** and updated based on **RTT** (**round-trip time**).
  - RTT is the time needed for a segment to reach a destination and for an acknowledgment to be received.

Retransmission after Three Duplicate ACKs

- This is also called **fast retransmission**.
- If three duplicate acknowledgments (i.e., an **original ACK** plus **three exactly identical copies**) arrive for a segment, the next segment is retransmitted without waiting for the time-out.
  - Three or more duplicate ACKs are received in a row, it is a strong indication that **a segment has been lost**.

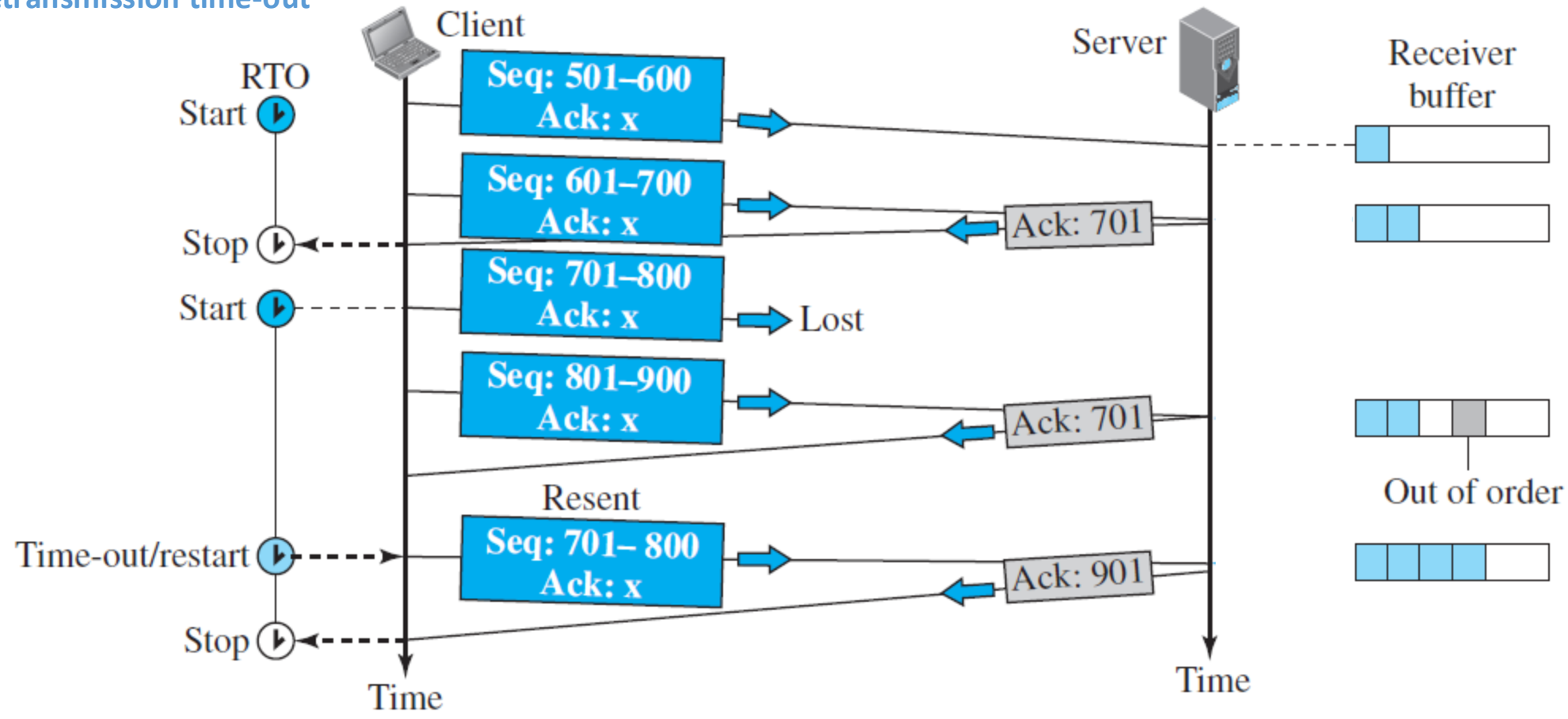
Checksum

- Detection of **corrupted segments**.
- It is a **16-bit** checksum.
- It is calculated using a **pseudoheader** (why?), the **TCP header**, and the **data** coming from the application layer.
- Difference with **network layer checksum**?
  - Network-layer checksum is only calculated over the **header** of IP datagram.
- Difference with **UDP checksum**?
  - UDP checksum is **optional**.

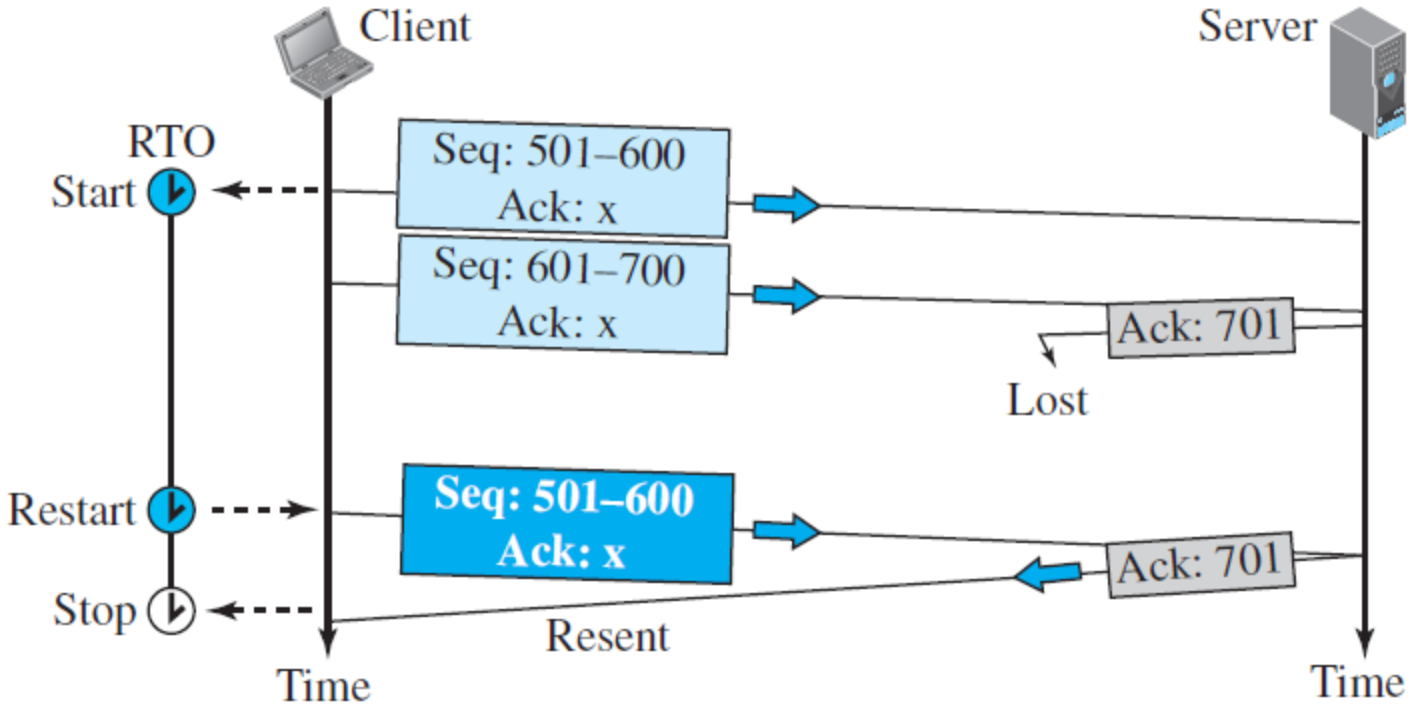


Retransmission after Time-out – Lost Segment

RTO: Retransmission time-out



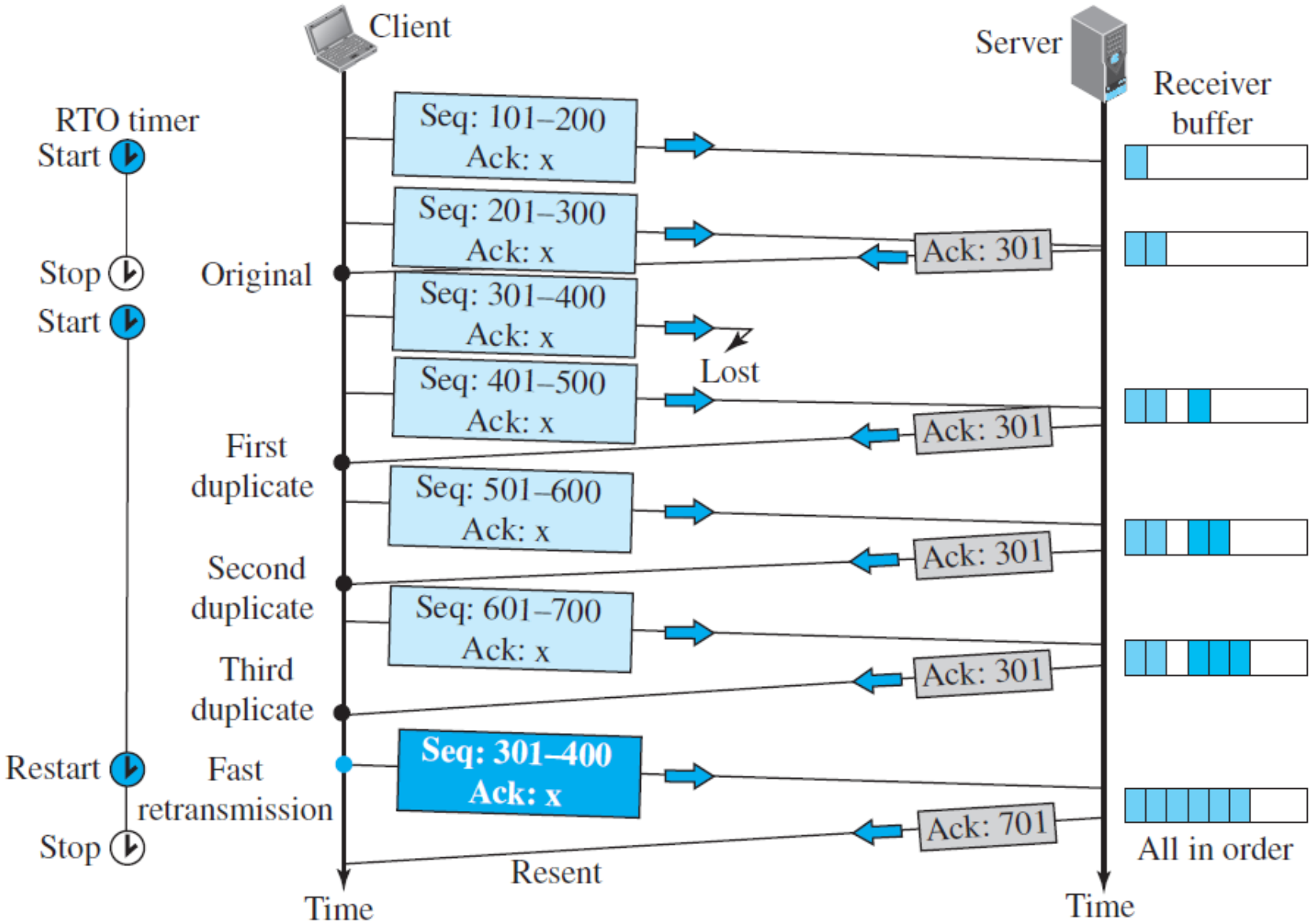
Retransmission after Time-out – Lost Acknowledgement



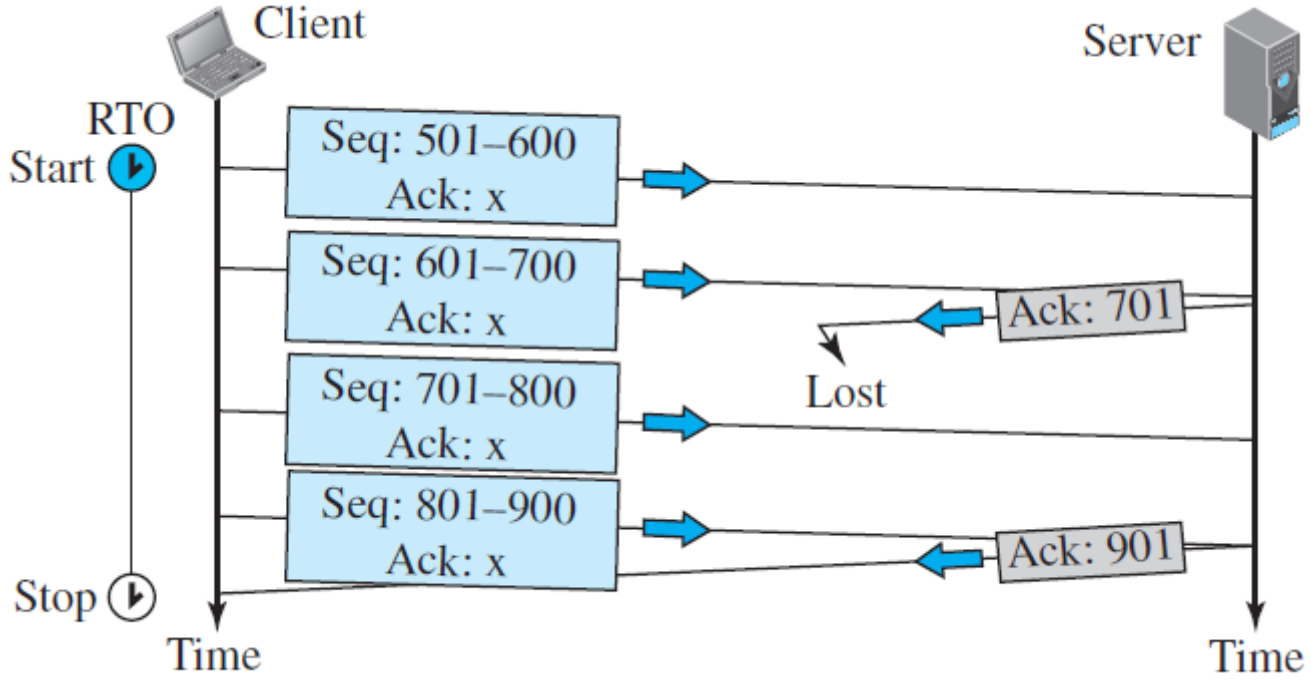
Duplicate segment, discarded

Lost acknowledgment is corrected by resending a segment.

Fast Retransmission (Three Duplicate ACKs)



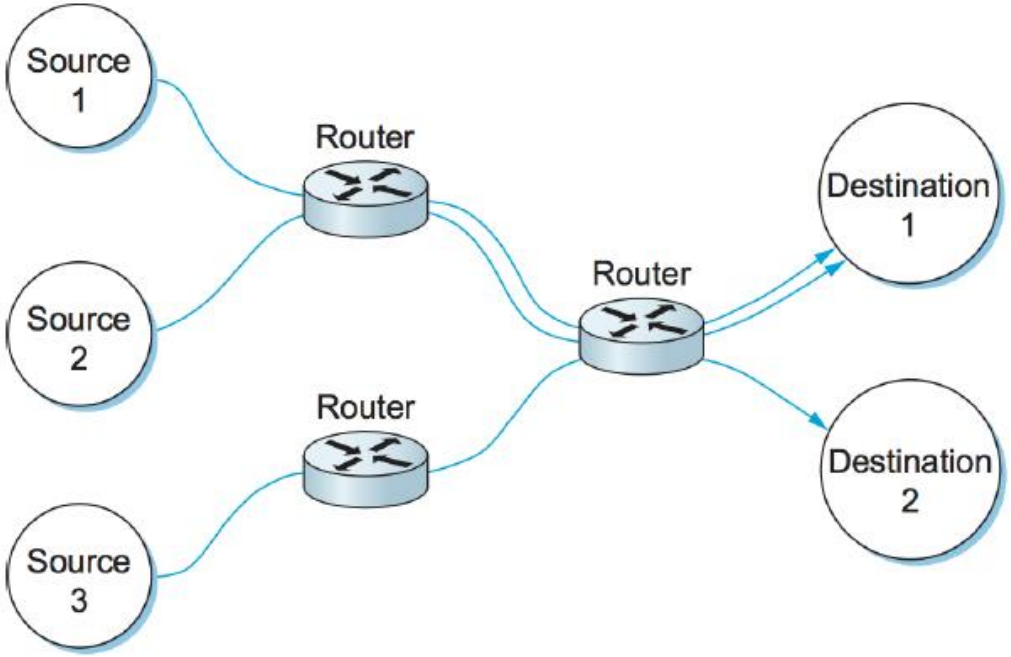
Cumulative Acknowledgement– Lost Acknowledgement



The next acknowledgment automatically corrects the loss of the previous acknowledgment.

Congestion in a Network

- When load on the network is more than its capacity
- Two main results of congestion:
  1. Increased end-to-end delay
  2. Packet loss (due to buffer/queue overflow)



TCP Congestion Detection

- TCP congestion control is **feedback-based**.
  - To **detect congestion**, TCP sender uses an implicit feedback from the other end, i.e., **acknowledgements** (**ACKs**).
- Two events are considered as signs of congestion in the network:
  1. **Time-out**: sign of a **strong** congestion
  2. Receiving **three duplicate ACKs**: sign of a **weak** congestion

TCP Congestion Control

- TCP provides **end-to-end congestion control**.
- End host (TCP sender) **adjusts its sending rate** according to the network condition.

TCP Congestion Control is Window-based

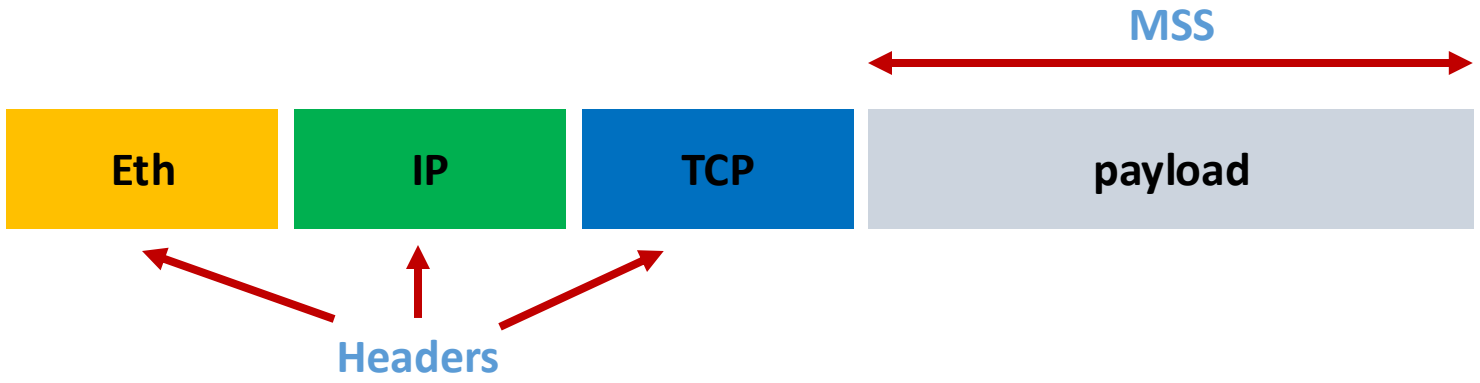
- TCP maintains a state variable for each connection called **congestion window** (**cwnd**).
  - Imposes a constraint on the rate at which a TCP sender can send traffic into the network.
- The maximum number of unacknowledged (in-flight) data bytes ≤ cwnd
  - Assume that the receive window size (rwnd) is much larger than cwnd.

TCP Congestion Window

- TCP source sets the **congestion window** (**cwnd**) based on the level of congestion it perceives to exist in the network.
- This involves **decreasing** the congestion window when the level of **congestion goes up** and **increasing** the congestion window when the level of **congestion goes down**.

TCP Congestion Window and MSS

- What is Maximum Segment Size (MSS)?
  - The maximum amount of application layer data in the segment.



- MSS is used in TCP congestion control algorithm to **adjust** the sending rate (i.e., cwnd ≥ MSS) → negotiated during the **connection establishment**

TCP Congestion Control Algorithms

- Three main components:
  1. **Slow start**
  2. **Congestion Avoidance**
  3. **Fast Recovery**
     - New versions of TCP use it.
     - It starts when three duplicate ACKs arrive (light congestion).

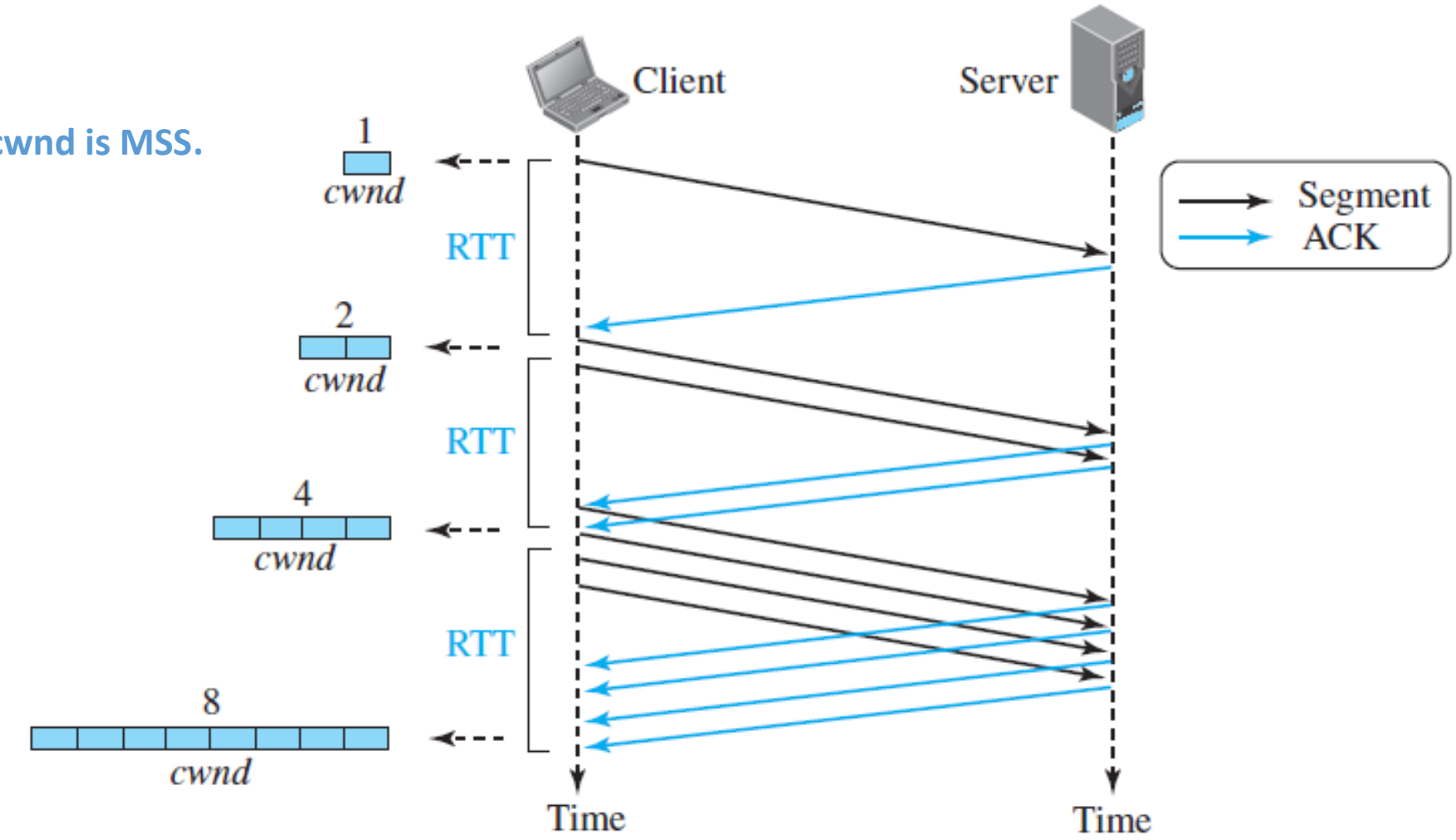
} mandatory

Slow Start: Exponential Increase

- The size of the congestion window (**cwnd**) begins at **1 MSS** and is **doubled** per **RTT** (i.e., sending rate is doubled per RTT).
- The exponential increase:
  - **Continues** until it reaches a threshold called **slow start threshold** (**ssthresh**).
  - Or is **restarted** if **congestion** is detected.

Slow Start: Exponential Increase

Unit of cwnd is MSS.

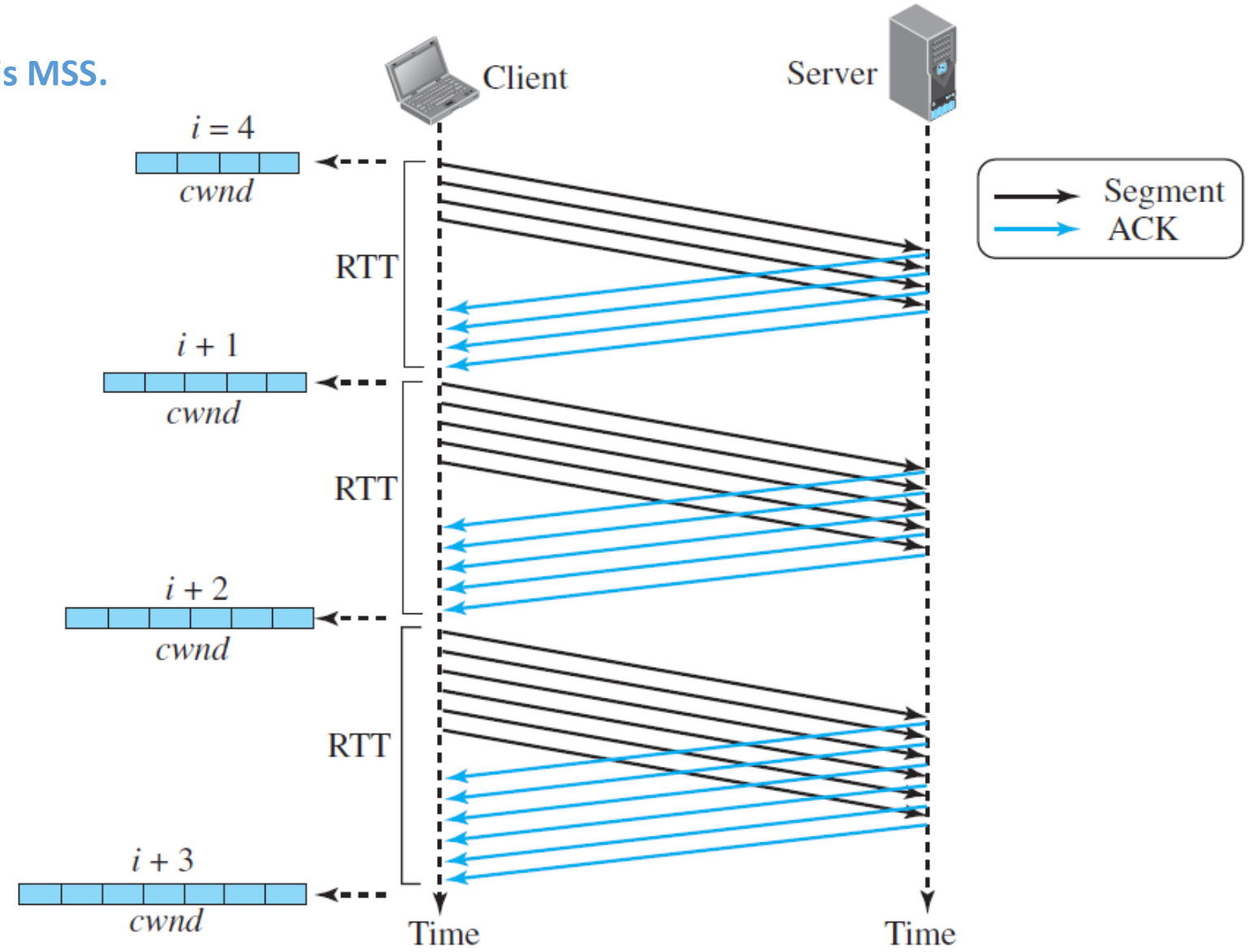


Congestion Avoidance

- A conservative approach.
- The size of the congestion window increases **additively** (**linearly**) until congestion is detected.

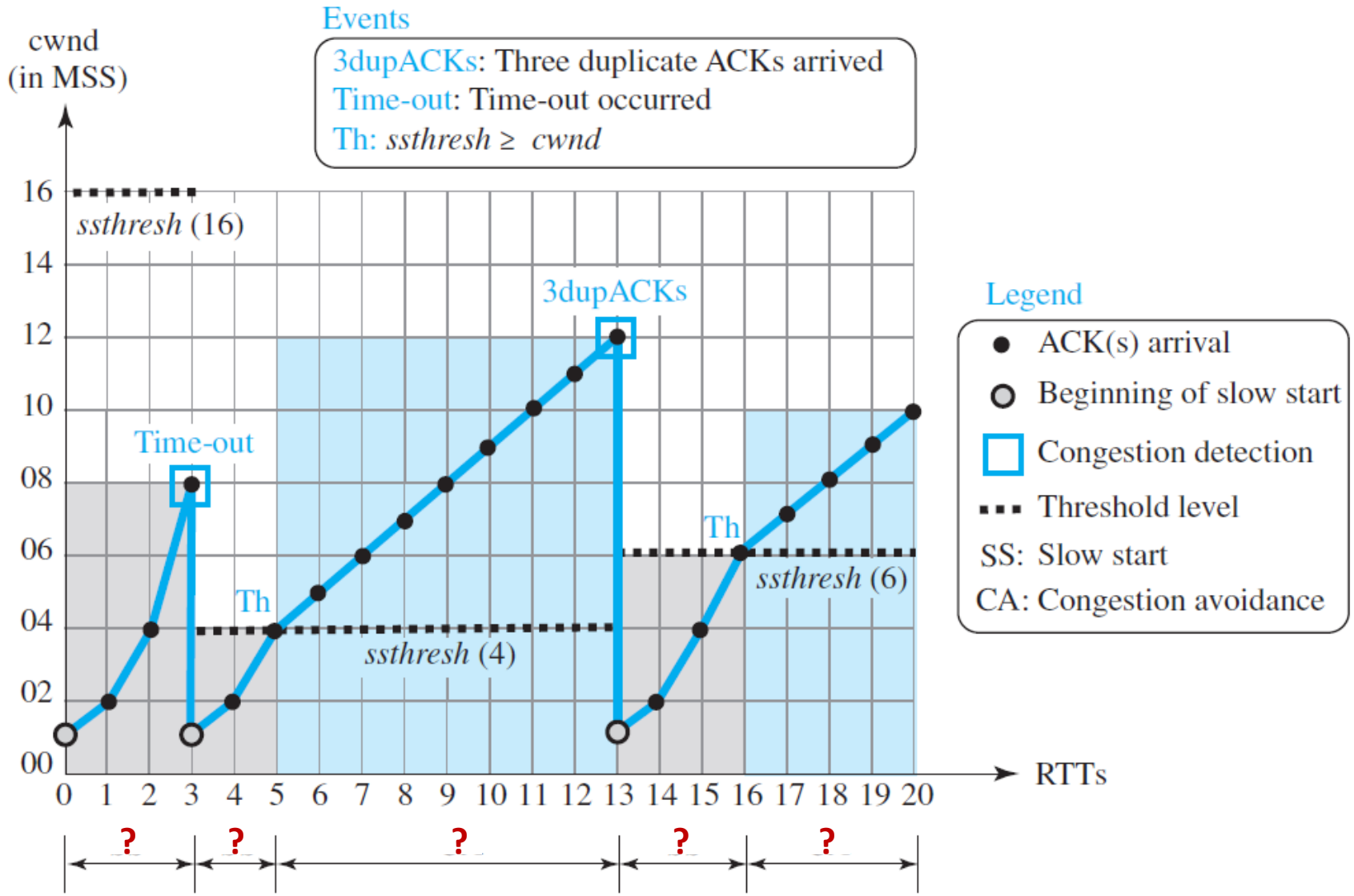
Congestion Avoidance

Unit of cwnd is MSS.



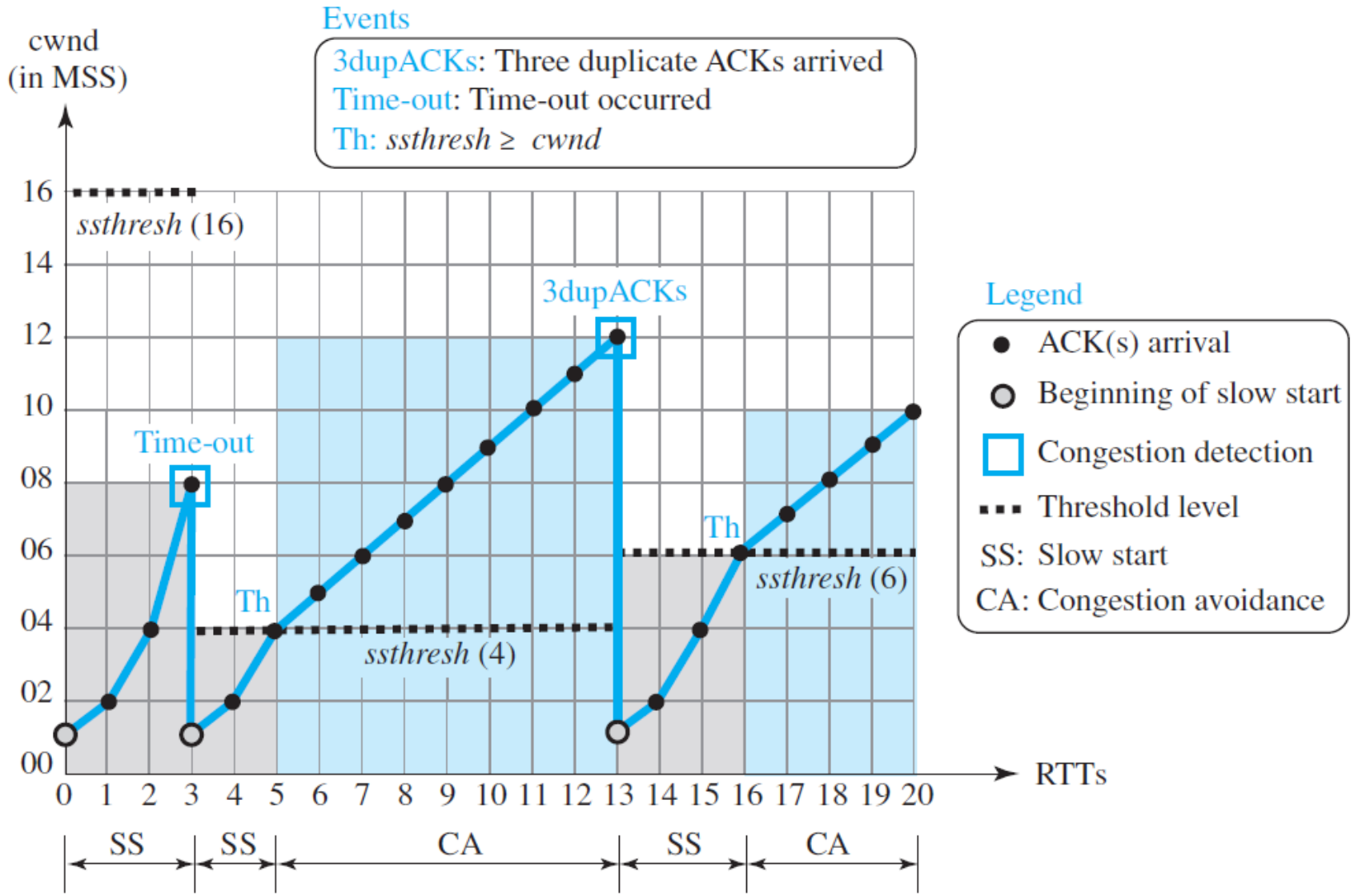
Tahoe TCP – Example

Tahoe is an early version of TCP.



Tahoe TCP – Example

Tahoe is an early version of TCP.



TCP Congestion Control vs. TCP Flow Control

- Avoid **overwhelming** the **network** vs. Avoid **overwhelming** the **receiver**.
- TCP sender maintains **cwnd** , TCP receiver: maintains **rwnd**.
- The maximum number of **unacknowledged** (in-flight) **data bytes** ≤ **min**{**cwnd**, **rwnd**}

Summary

- TCP is a **reliable connection-oriented** protocol.
- TCP explicitly defines **connection establishment**, **data transfer**, and **connection teardown** phases (**connection-oriented service**).
- The **connection** is **logical** and **full-duplex**.
- TCP packets are called **segments**.
- TCP connection establishment is called **three-way handshake**.
- TCP connection is **point-to-point** (i.e., between a single sender and a single receiver).

Summary (Cont.)

- TCP flow control is implemented using **sliding window**.
- TCP uses error control to provide a **reliable service**.
- **Checksum** is used to detect corrupted segments.
- **Cumulative acknowledgements** are used to confirm the receipt of data segments.
- **Sequence numbers** are helpful in detection of lost, out-of-order, and duplicate segments.
- **Retransmission** is the heart of TCP error control (two types).
  - **Retransmission time-out** interval calculation based on RTT.
- TCP **congestion control** algorithm has three phases.

References

[1] Behrouz A.Forouzan, Data Communications & Networking with TCP/IP Protocol Suite, 6th Ed, 2022, McGraw-Hill companies.

[2] J.F. Kurose, K.W. Ross, Computer Networking: A Top-Down Approach, 7th Ed, 2017, Pearson Education, Inc.

Reading

- Chapter 9 of the textbook, section 9.4
- Chapter 9 of the textbook, section 9.7 (Practice Test)