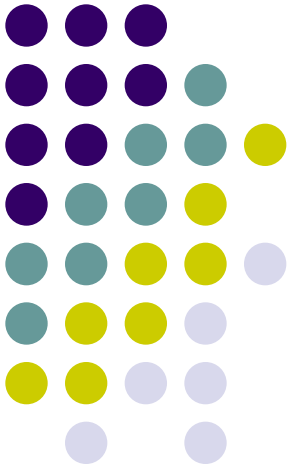
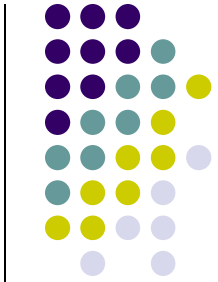
ELEN 4017

Network Fundamentals

Lecture 18



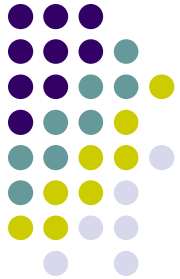
Purpose of lecture



Chapter 3: Transport Layer

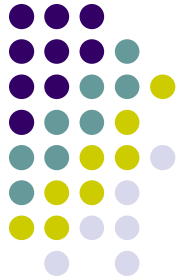
- Transmission Control Protocol Basics
- Timeout in TCP

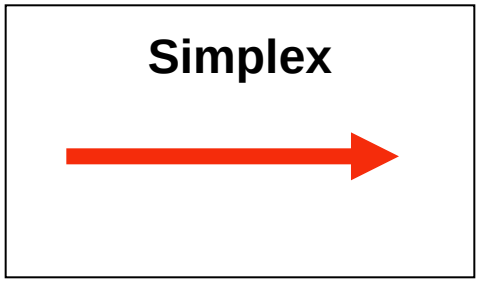
Basics of TCP

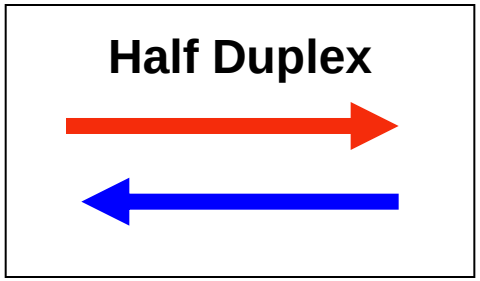


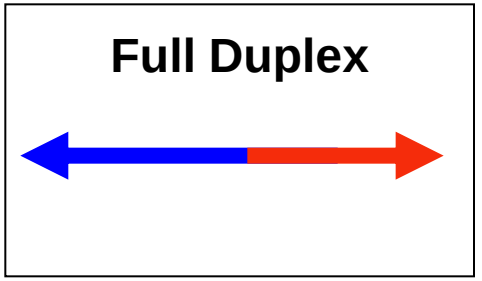
- Reliable data transfer
- Connection-oriented (handshake)
- Not an end-to-end TDM/FDM connection.
- Not a virtual circuit (resource and path not reserved)
- Full duplex (bi-directional)
- Point-to-point i.e. communications between a single sender and receiver.

Uni and bi-directional communication



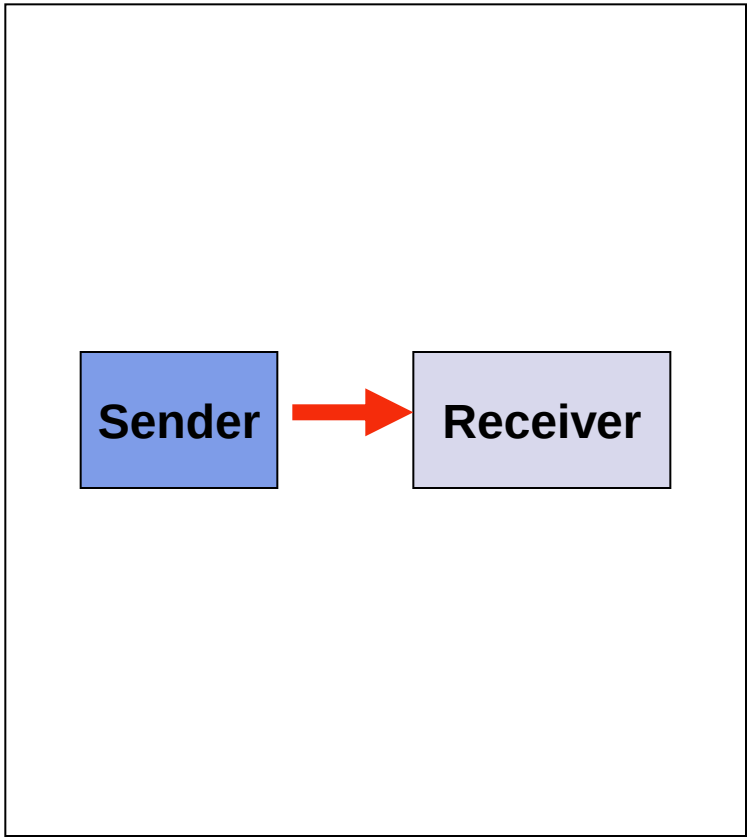


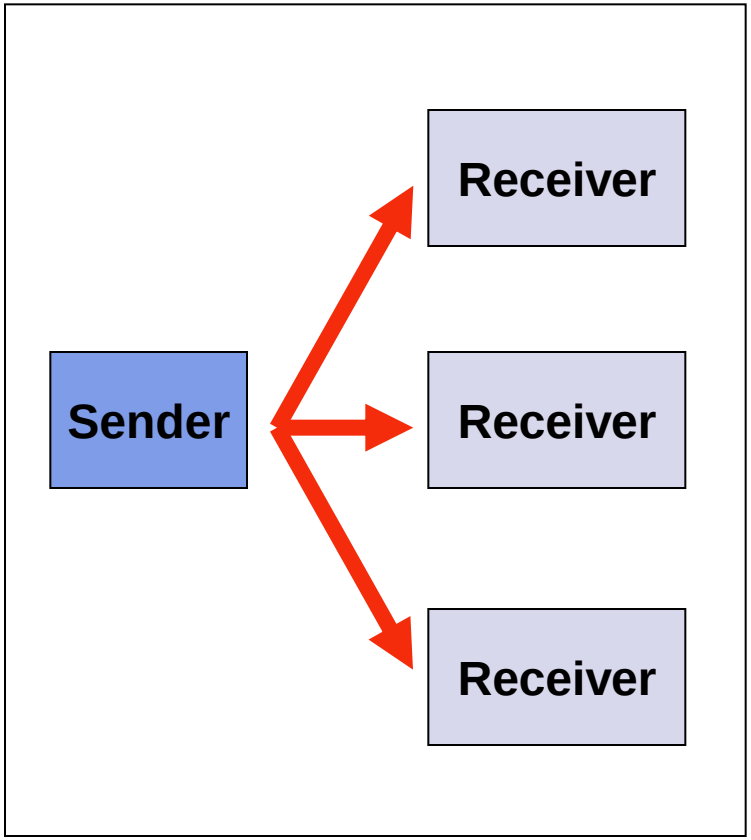




Examples ?

Point-to-point vs multipoint





Examples ?

Virtual circuit vs Datagram

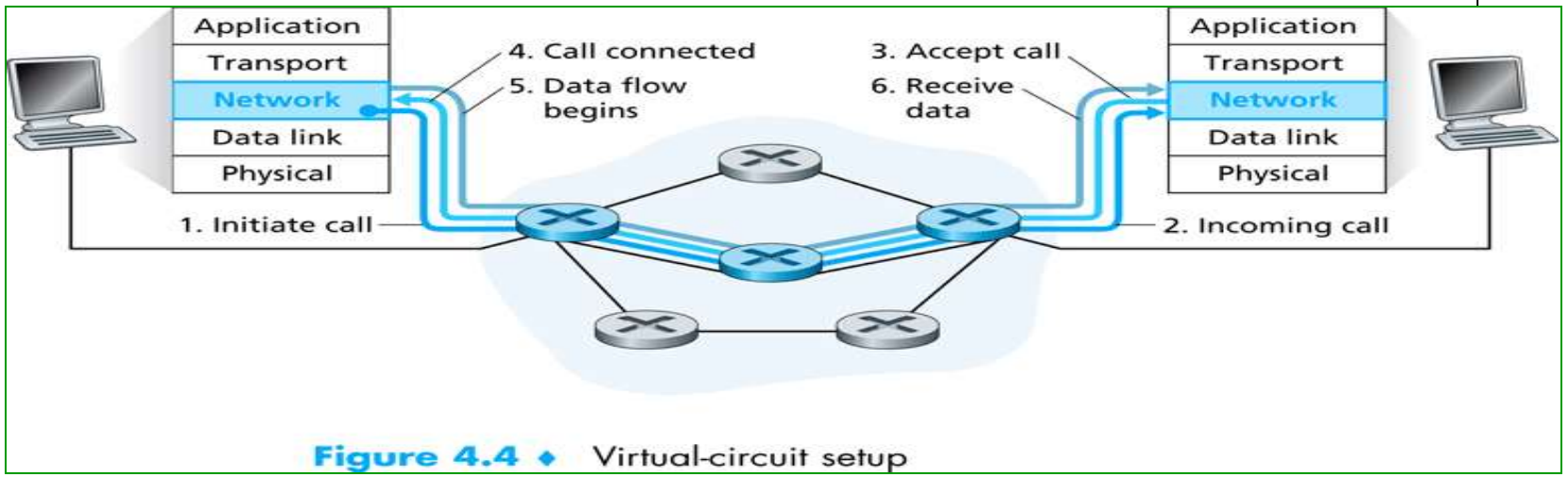


Figure 4.4 ♦ Virtual-circuit setup

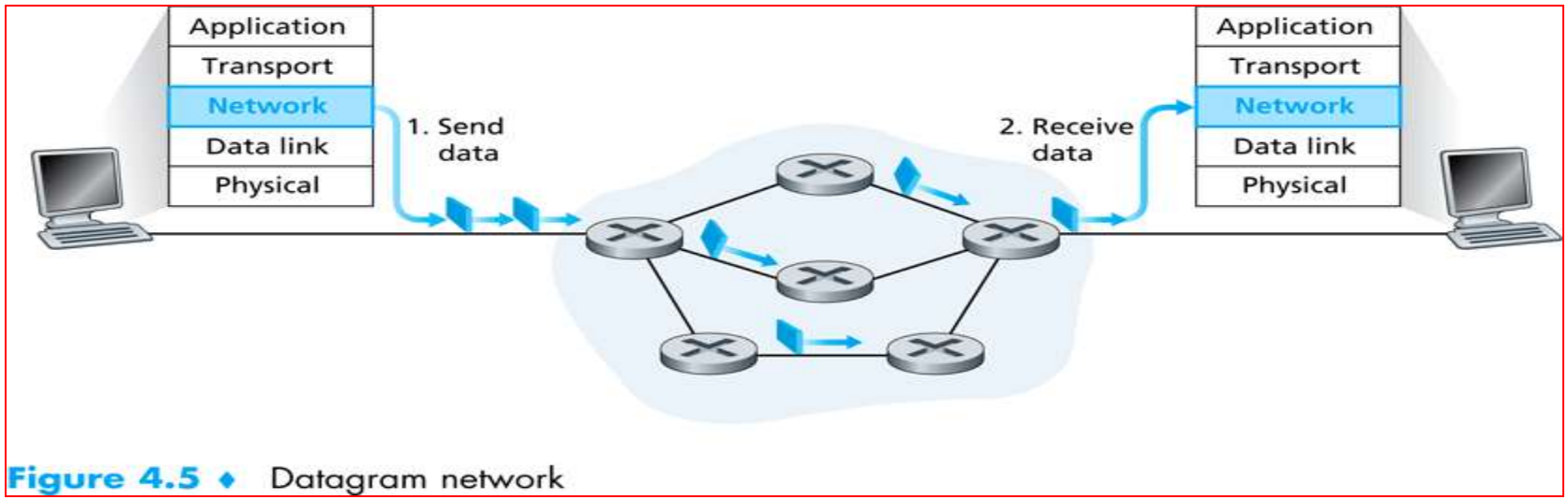


Figure 4.5 ♦ Datagram network

Buffers

- TCP employs buffers to store data during transmission.
- Buffers are assigned following a successful handshake.

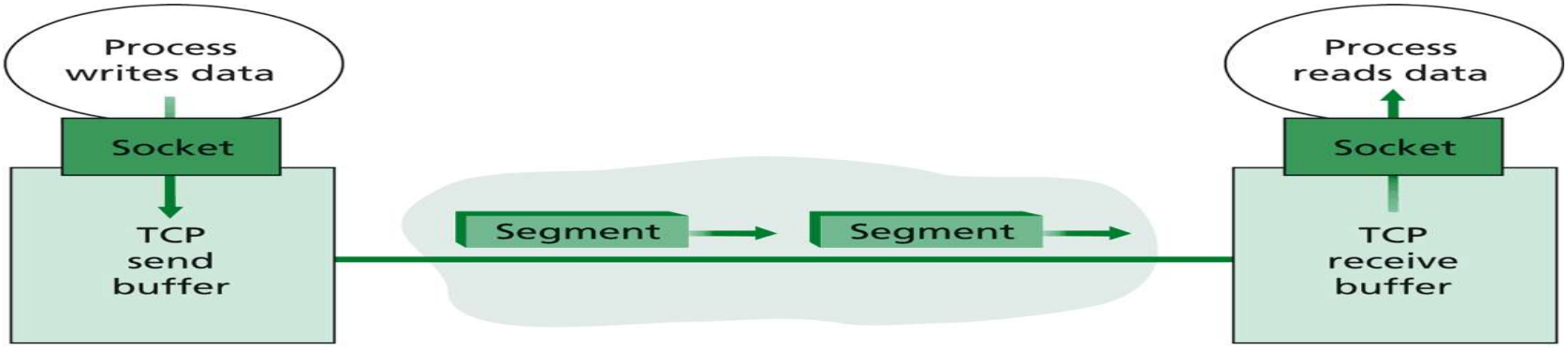
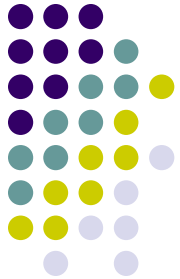


Figure 3.28 ◆ TCP send and receive buffers

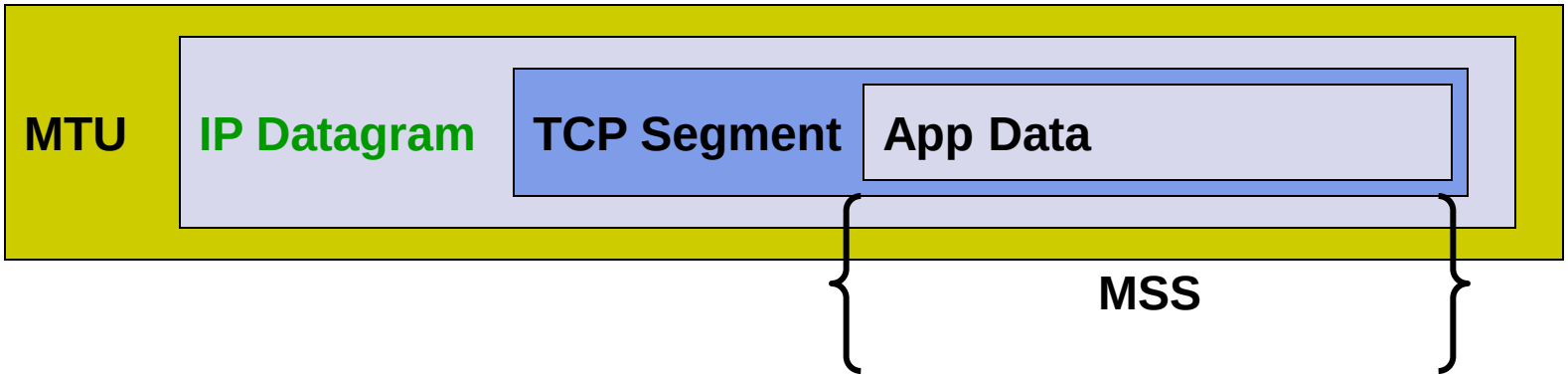
Maximum segment size (MSS)



- **MSS** defines the maximum amount of data that can be grabbed and placed in a segment.
- This is usually dimensioned using the length of largest link-layer frame that can be sent by the local host - called Maximum Transmission Unit (MTU).
- There have been proposals to set the MSS based on the path MTU value. Why ?

MSS

- Determine MTU (i.e. link layer frame)
- Fit a TCP segment into this MTU.
- What is the size of the application data → this determines MSS



TCP Segment

- Source and destination ports
- 32 bit sequence and ACK no.
- Receive window (flow control)
- Header length
- Flag field:
  - RST, SYN, FIN are used for connection setup/teardown.

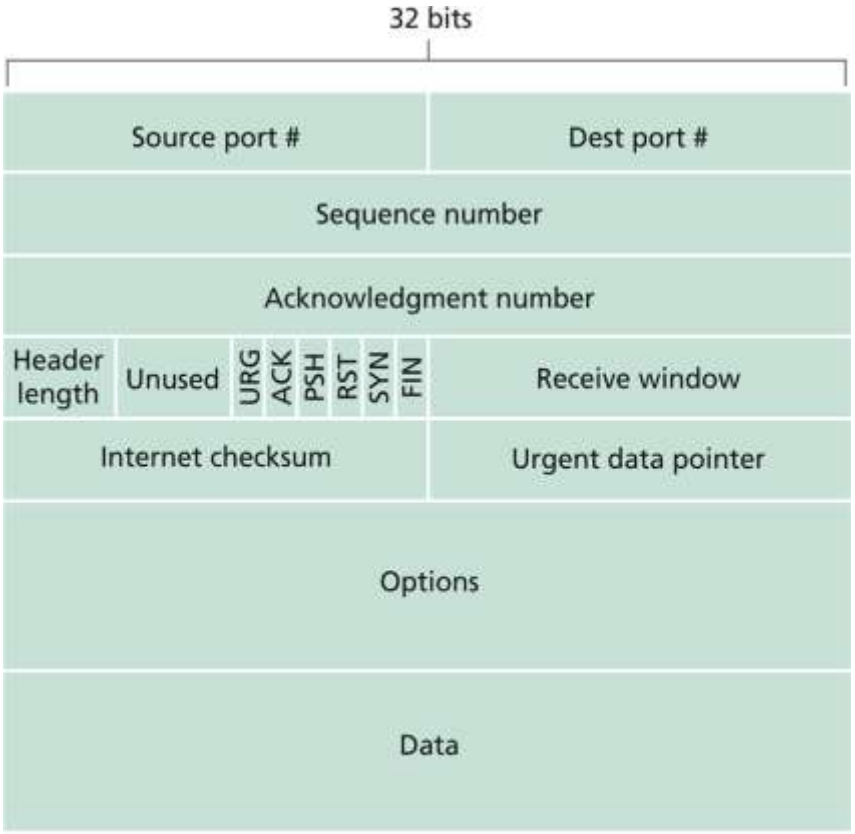
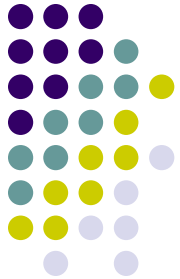


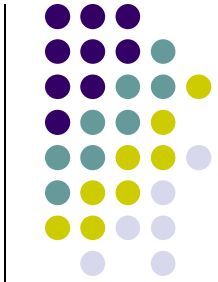
Figure 3.29 ♦ TCP segment structure

TCP Reliable transfer



- TCP uses a **hybrid of GBN and SR**.
- It also **piggy-backs messages** i.e. the acknowledgment for **client-to-server** data, is put into the same segment as **server-to-client data**.
- The sequence numbers are assigned on **byte position**, not packet number (more later).

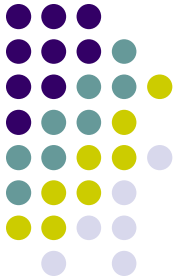
Purpose of lecture



Chapter 3: Transport Layer

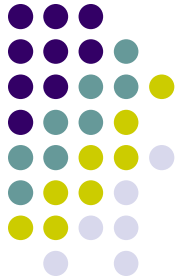
- Transmission Control Protocol Basics
- Timeout in TCP

Timeout handing



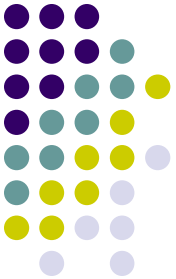
- How do we estimate an optimised timeout value ?
- We don’t want a timeout that it is too long since it will delay the pipelining once the window is full.
- We don’t want a timeout that is too short, since it means unnecessary retransmissions.

Estimating the RTT



- RTT is the time between sending a segment to IP and receiving the ACK.
- TCP does not measure this for every segment, usually just 1 segment at a time, thus we usually have 1 sample every RTT.
- TCP also does not compute the RTT for a retransmitted segment.
- These values are averaged to smooth out fluctuations.

Exponential Weighted Moving Average (EWMA)



EstimatedRTT = (1- α).EstimatedRTT + α. SampleRTT

- α → recommended = 0.125
- Weighted average, more weight assigned to recent samples.
- Weight of a given sample decays exponentially fast as updates proceed.

RTT sample and RTT estimates

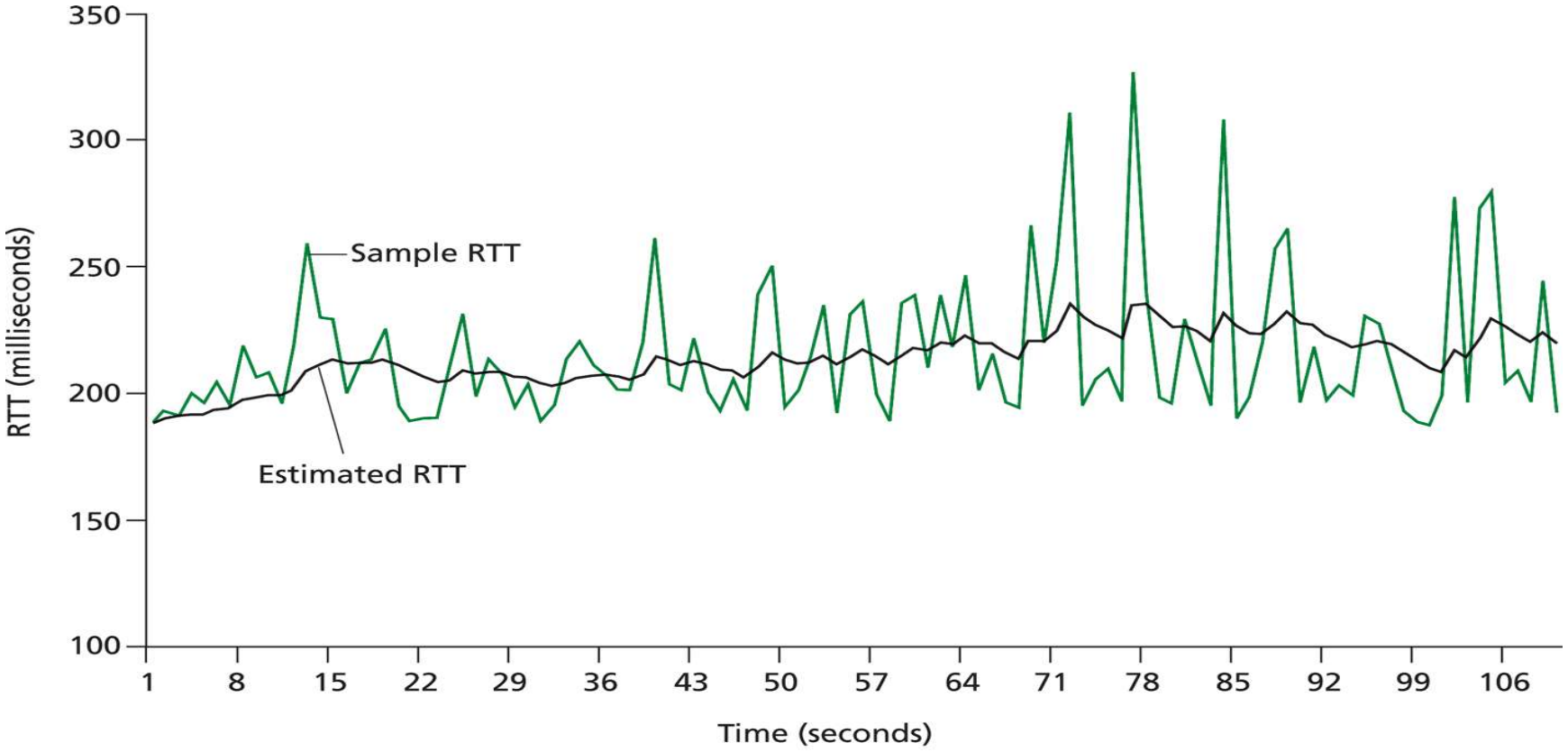


Figure 3.32 ◆ RTT samples and RTT estimates

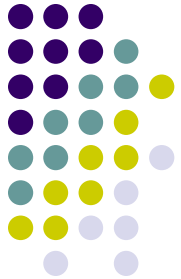
Variability of RTT (deviation)

$$DevRTT = (1- \beta).\ DevRTT + \beta.|\ SampleRTT - EstimatedRTT|$$

- β recommended = 0.25
- Weighted moving average of deviation from mean.
- In case of low fluctuation, DevRTT is small.
- TCP thus uses the following timeout interval

$$TimeoutInterval = EstimatedRTT + 4.DevRTT$$

Special features of TCP – Doubling timeout interval



- In case of retransmits, TCP sets the timeout interval = twice the currently used interval.
- Thus the interval will grow exponentially for each retransmit.
- This provides a limited form of congestion control.

Fast retransmit

- A problem with **timeout triggered retransmissions** is that the timeout period can be relatively long.
- The sender can often detect packet loss well before the time-out event occurs, by noting so-called duplicate ACKS.
- A **duplicate ACK** is an ACK that re-acknowledges a segment for which the sender already received an earlier ACK.
- If sender receives 3 duplicate ACKS back-to-back, it assumes that the next packet in sequence was lost, and re-transmits.