

# National University of Computer & Emerging Sciences

## CS 3001 - COMPUTER NETWORKS

### Lecture 13 Chapter 3

5<sup>th</sup> October, 2023

Nauman Moazzam Hayat  
[nauman.moazzam@lhr.nu.edu.pk](mailto:nauman.moazzam@lhr.nu.edu.pk)

Office Hours: 02:30 pm till 06:00 pm (Every Tuesday & Thursday)

# Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- **Connection-oriented transport: TCP**
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- Principles of congestion control
- TCP congestion control

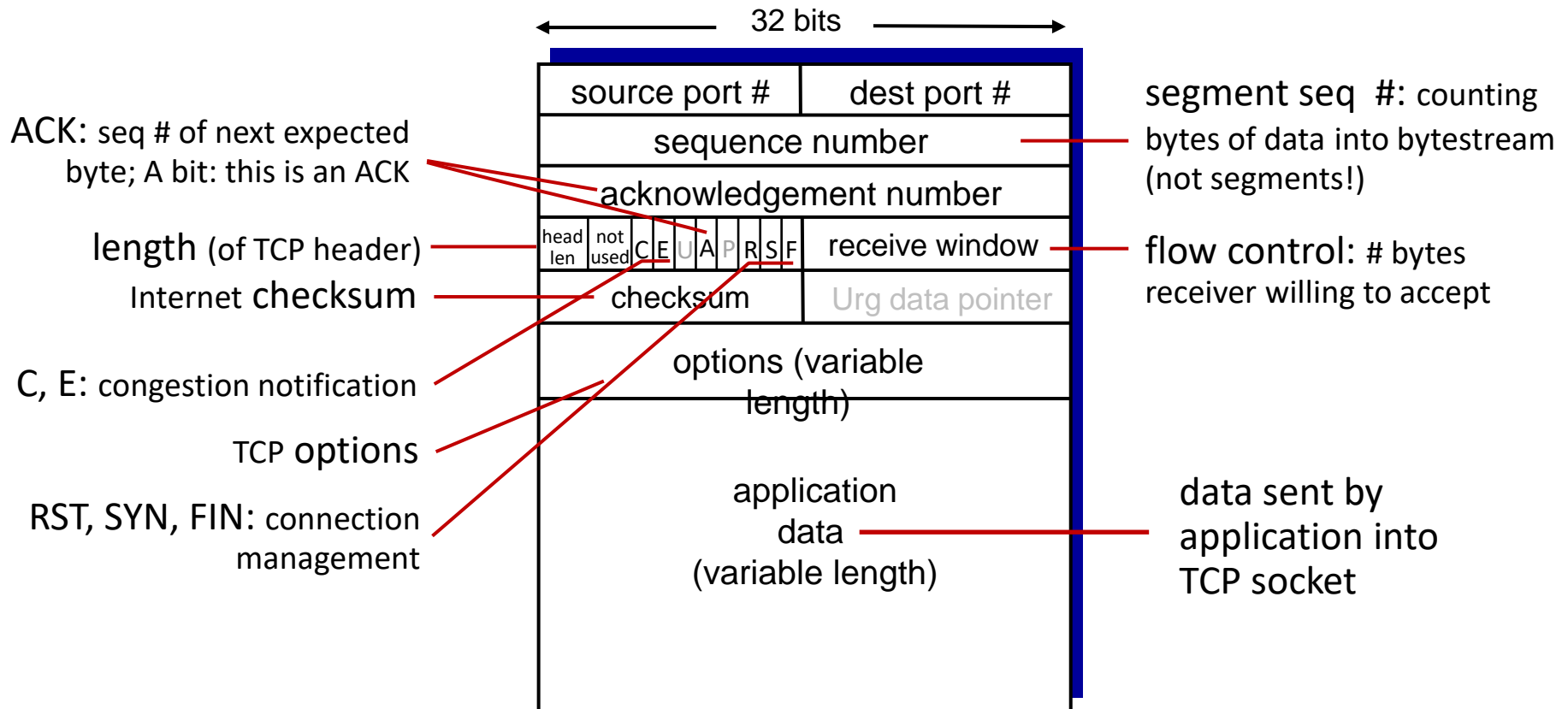


# TCP: overview

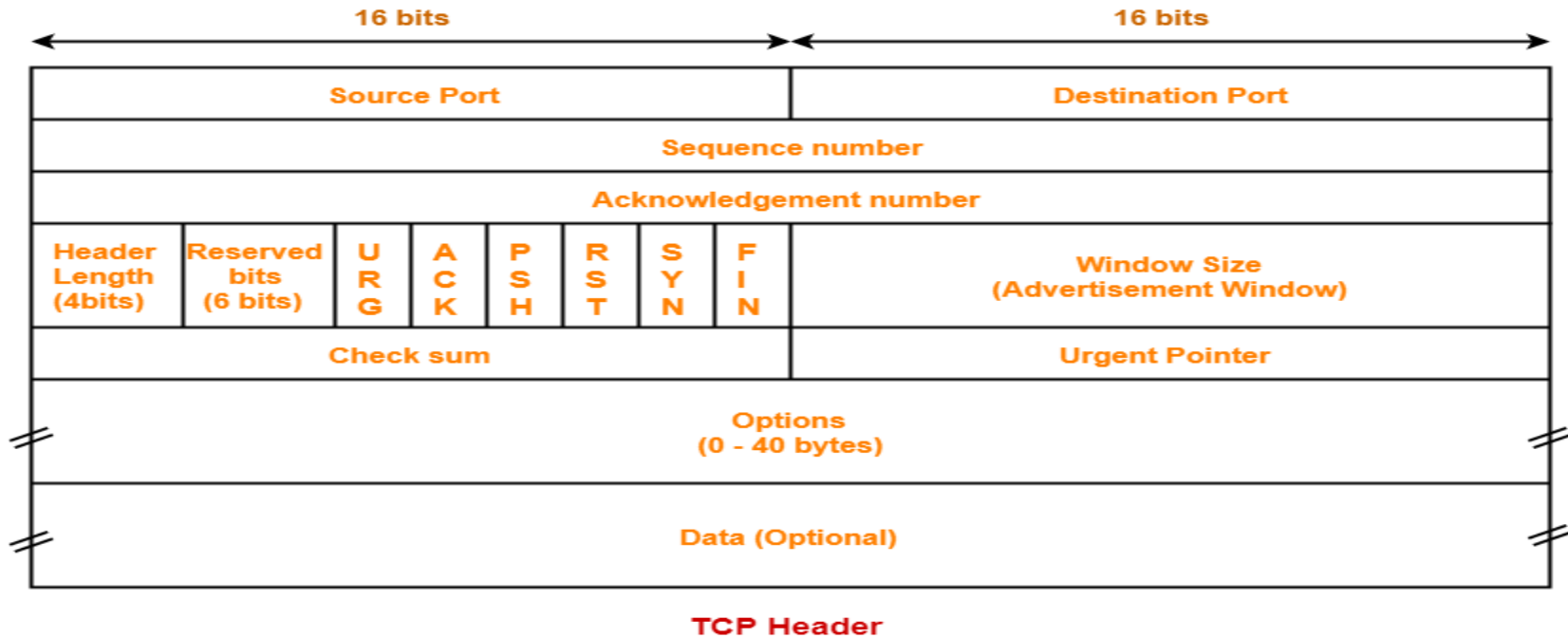
RFCs: 793, 1122, 2018, 5681, 7323

- **point-to-point:**
  - one sender, one receiver
- **reliable, in-order *byte stream*:**
  - no “message boundaries”
- **full duplex data:**
  - bi-directional data flow in same connection
  - MSS: maximum segment size **is based on Path MTU. (It is the max. amount of app layer data in the segment, not the max size of TCP segment including the TCP header.)**
- **cumulative ACKs**
- **pipelining:**
  - TCP congestion and flow control set window size
- **connection-oriented:**
  - **3-way** handshaking (exchange of control messages) initializes sender, receiver state before data exchange
- **flow controlled:**
  - sender will not overwhelm receiver

# TCP segment structure



# TCP Header (cont'd)



- Header length is a 4 bit field.
- It contains the length of TCP header.
- It helps in knowing from where the actual data begins.
- [Minimum and Maximum Header length](#)

The length of TCP header always lies in the range: [20 bytes (minimum) till 60 bytes (maximum)]

- The initial 5 rows of the TCP header are always used.
- The size of the 6th row representing the Options field vary as it can be used or not used.
- The size of Options field can go from 0 bytes till 40 bytes.
- Header length is a 4-bit field, thus it can have a min value of 0000 (0 in decimal) till a max value of 1111 (15 in decimal)
- But the range of header length is [20, 60].
- So, to represent the header length, we use a scaling factor of 4.
- Thus, in general:

$$\text{Header length} = \text{Header length field value} \times 4 \text{ bytes}$$

# Difference between PUSH & Urgent Flags in TCP Header

| PSH  | URG   |
|--|---|
| --> All data in buffer to be pushed to NL(sender)/ AL(receiver). | --> Only the urgent data to be given to AL immediately. |
| --> Data is delivered in sequence.                               | --> Data is delivered out of sequence.                  |

(marked by the Urgent Data Pointer field.)

# TCP sequence numbers, ACKs

## Sequence numbers:

- byte stream “number” of first byte in segment’s data

## Acknowledgements:

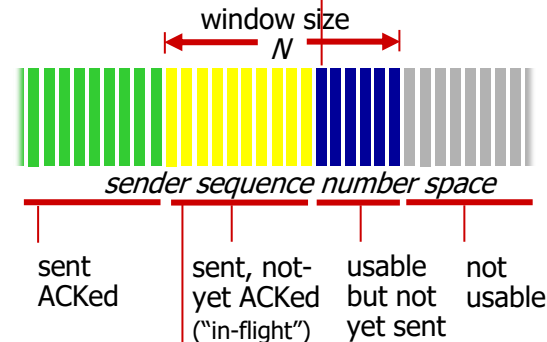
- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn’t say, - up to implementor (two basic choices, i.e. either discard out-of-order segment or buffer it.)

outgoing segment from sender

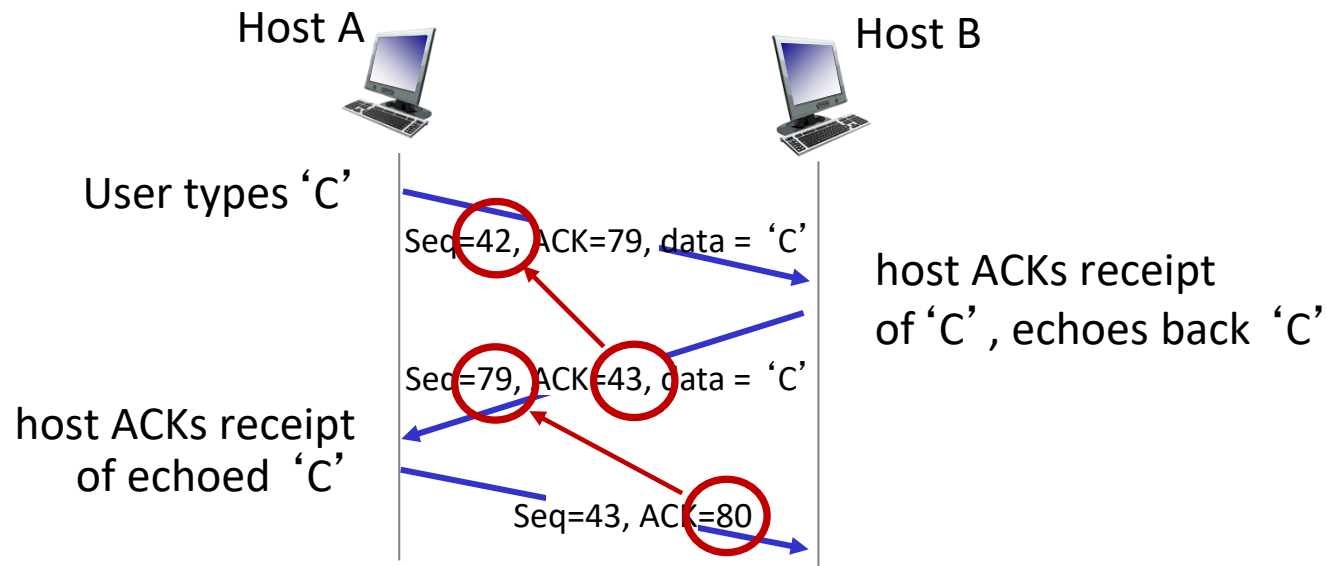
|                 |             |
|-----------------|-------------|
| source port #   | dest port # |
| sequence        |             |
| acknowledgement |             |
| number          | rwnd        |
| checksum        | urg pointer |



outgoing segment from receiver

|                 |             |
|-----------------|-------------|
| source port #   | dest port # |
| sequence        |             |
| acknowledgement |             |
| A               | rwnd        |
| checksum        | urg pointer |

# TCP sequence numbers, ACKs



simple telnet scenario



# TCP round trip time, timeout

Q: how to set TCP timeout value?

- longer than RTT, but RTT varies!
- *too short*: premature timeout, unnecessary retransmissions
- *too long*: slow reaction to segment loss

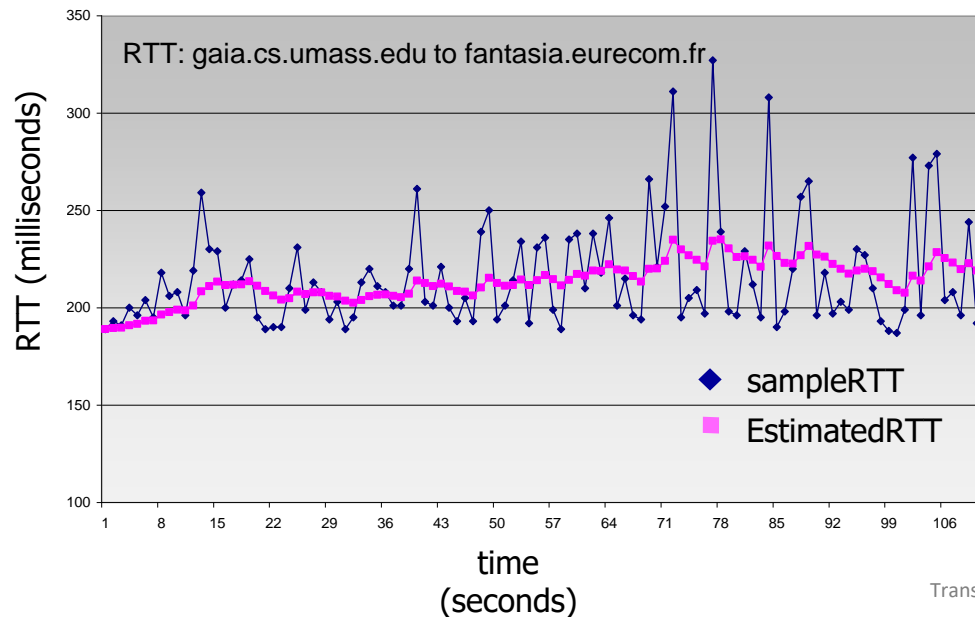
Q: how to estimate RTT?

- `SampleRTT`: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- `SampleRTT` will vary, want estimated RTT “smoother”
  - average several *recent* measurements, not just current `SampleRTT`

# TCP round trip time, timeout

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- exponential weighted moving average (EWMA)
- influence of past sample decreases exponentially fast
- typical value:  $\alpha = 0.125$



# TCP round trip time, timeout

- timeout interval: **EstimatedRTT** plus “safety margin”
  - large variation in **EstimatedRTT**: want a larger safety margin

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$



↑  
estimated RTT

↑  
“safety margin”

- **DevRTT**: EWMA of **SampleRTT** deviation from **EstimatedRTT**:

$$\text{DevRTT} = (1 - \beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

(typically,  $\beta = 0.25$ )

\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# Doubling the Timeout Interval

This modification provides a limited form of congestion control

- In this modification, whenever the timeout event occurs, each time TCP retransmits, it sets the next timeout interval to twice the previous value, rather than deriving it from the last **EstimatedRTT** and **DevRTT**
- Thus the intervals grow **exponentially** after each retransmission
- However, whenever the timer is started after either of the two other events (that is, data received from application above and/or ACK received), the TimeoutInterval is derived from the most recent values of **EstimatedRTT** and **DevRTT**

# Midterm 1 Solution Discussion



*Midterm 1 Solution  
was discussed*