National University of Computer & Emerging Sciences

CS 3001 – COMPUTER NETWORKS

Lecture 12

Chapter 3

6^{th} March, 2025

Nauman Moazzam Hayat

nauman.moazzam@lhr.nu.edu.pk

Office Hours: 11:30 am till 01:00 pm (Every Tuesday & Thursday)

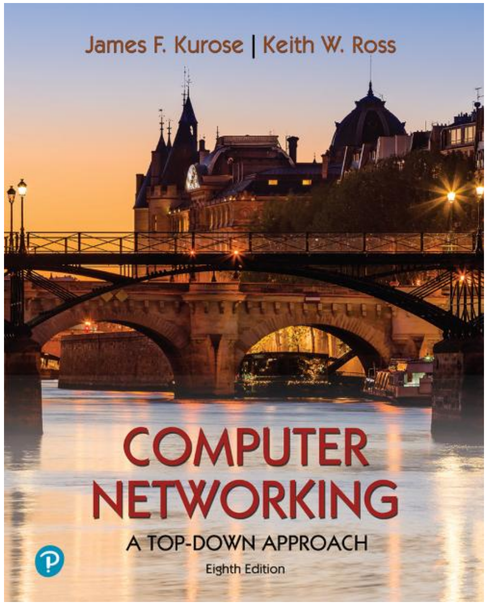
Chapter 3
Transport Layer

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

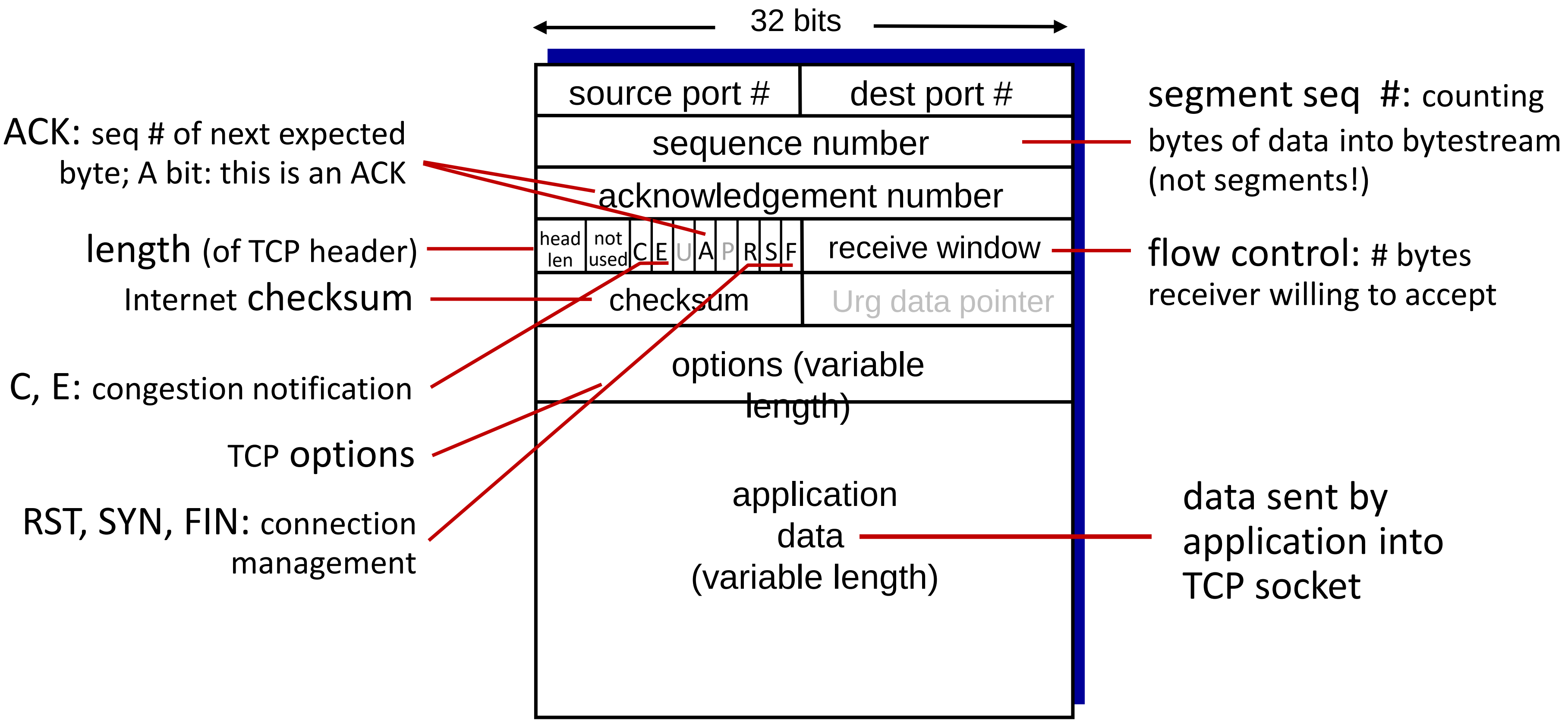
Chapter 3: roadmap

- 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

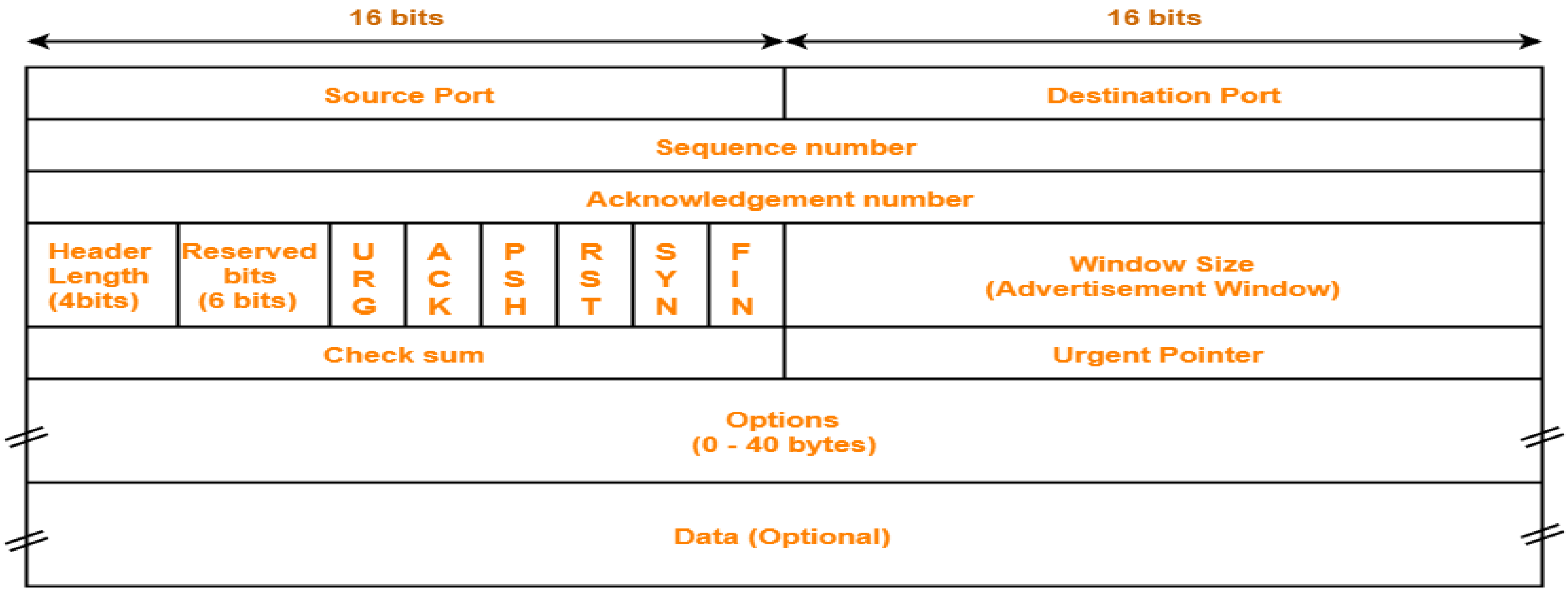
TCP: overview RFCs: 793,1122, 2018, 5681, 7323

- 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)



TCP Header

- 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:

Header length = Header length field value x 4 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:

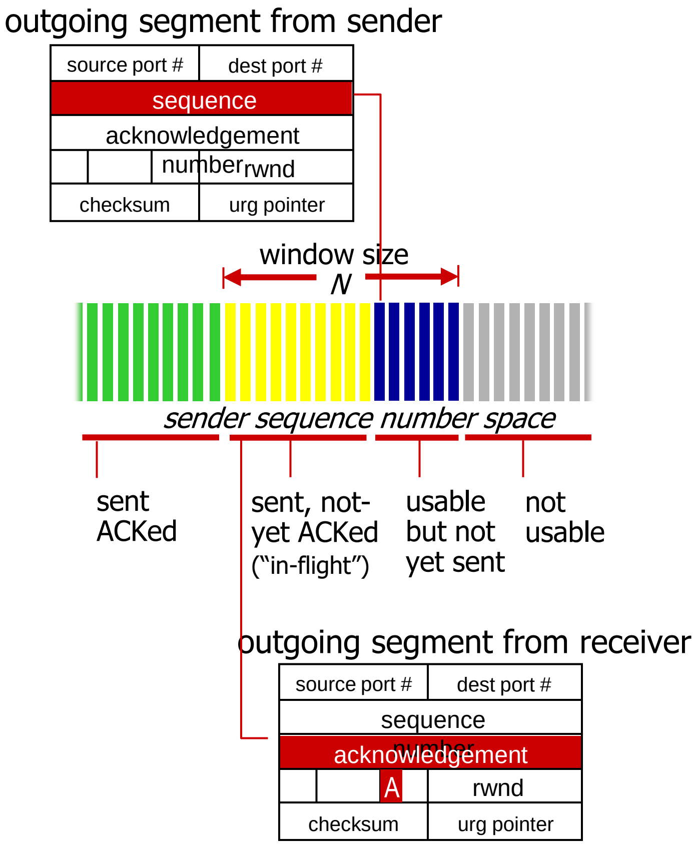
- byte stream "number" of first byte in segment's data

Acknowledgements:

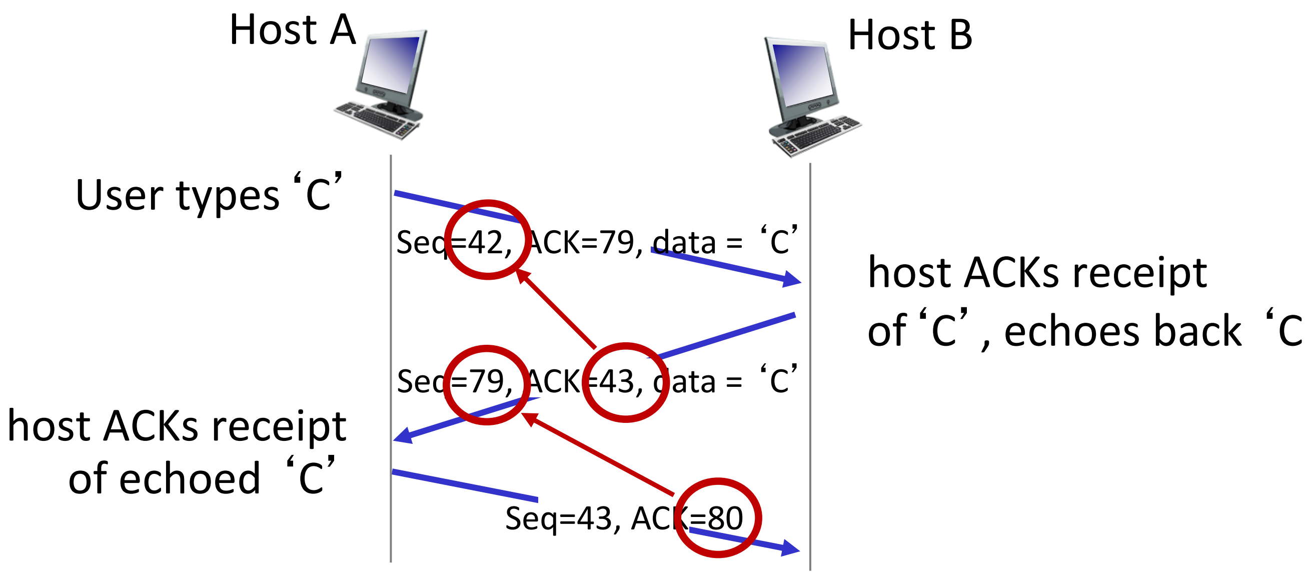
- seq # of next byte expected from other side
- cumulative ACK

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.)



TCP sequence numbers, ACKs



simple telnet scenario

TCP round trip time, timeout

Q: how to set TCP timeout value?

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

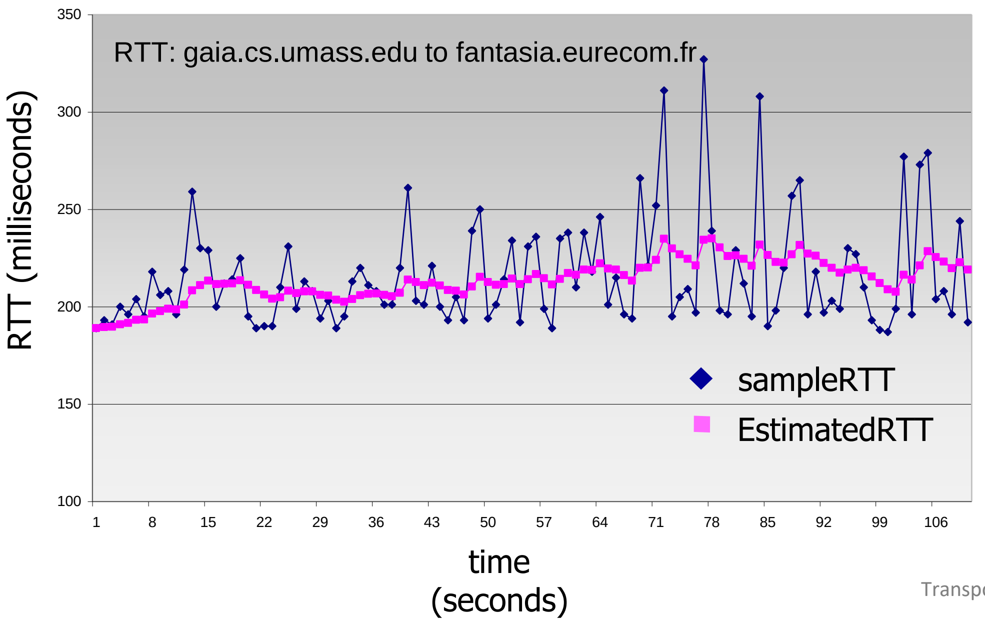
Q: how to estimate RTT?

- `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

```
EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT
```

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



TCP round trip time, timeout

- timeout interval: **`EstimatedRTT`** plus "safety margin"
  - large variation in **`EstimatedRTT:`** want a larger safety margin

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

estimated RTT

"safety margin"

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

```
DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|
```

(typically, β = 0.25)

* Check out the online interactive exercises for more examples: 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