CSCE 463/612
Networks and Distributed Processing
Spring 2024

Transport Layer V

Dmitri Loguinov
Texas A&M University

March 8, 2024

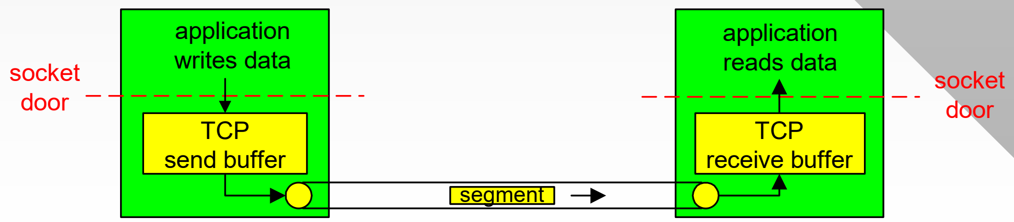
Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer
3.5 Connection-oriented transport: TCP
- Segment structure
- Reliable data transfer
- Flow control
- Connection management

3.6 Principles of congestion control
3.7 TCP congestion control

TCP: Overview [RFCs: 793, 1122, 1323, 2001, 2018, 2581, 3390, 5681]

- Point-to-point (unicast):
  - One sender, one receiver
- Reliable, in-order *byte stream:*
  - Packet boundaries are not visible to the application
- Pipelined:
  - TCP congestion and flow control set window size
- Send & receive buffers



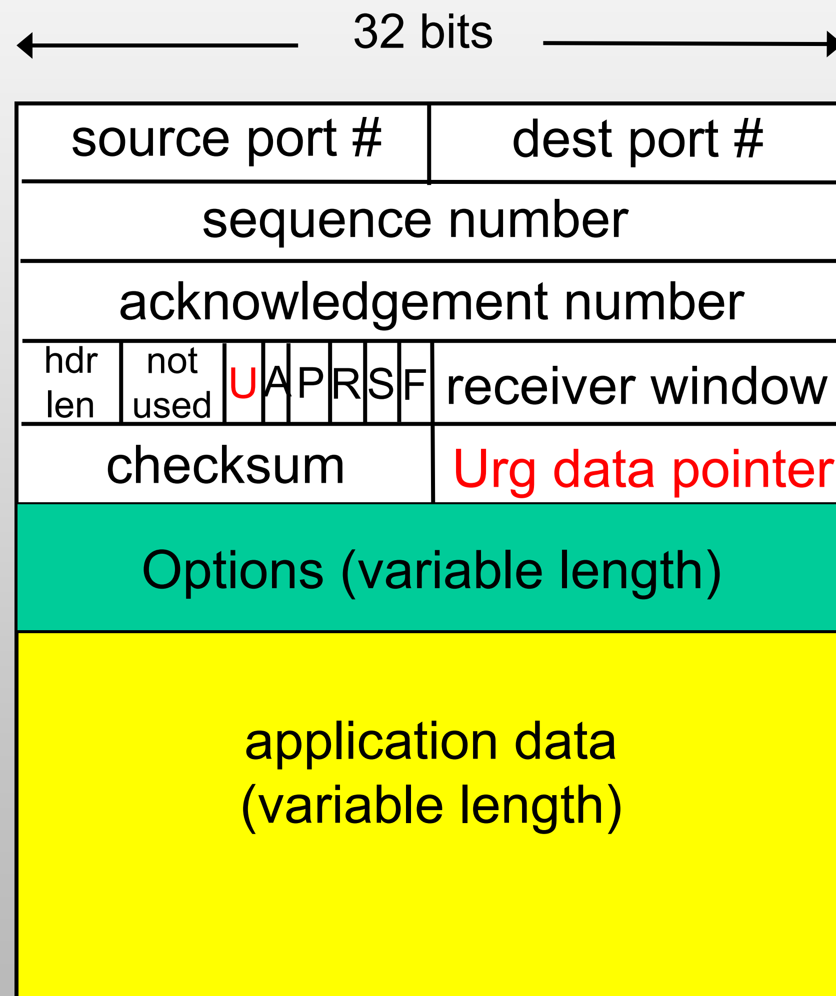
- Full duplex data:
  - Bi-directional data flow in same connection
- MSS: maximum segment size (excluding headers)
- Connection-oriented:
  - Handshaking (exchange of control msgs) initializes sender/receiver state before sending data
- Flow controlled:
  - Sender will not overwhelm receiver

Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer
3.5 Connection-oriented transport: TCP
- Segment structure
- Reliable data transfer
- Flow control
- Connection management

3.6 Principles of congestion control
3.7 TCP congestion control

TCP Segment Structure



- Sequence/ACK numbers
  - Count bytes, not segments
  - ACKs piggybacked on data packets
- Flags (U-A-P-R-S-F)
  - Urgent data (not used)
  - ACK field is valid
  - PUSH (reduce latency)
  - RST (reset connection)
  - SYN (connection request)
  - FIN (connection close)
- Hdr length in DWORDs (4-bit field)
  - Normally 20 bytes, but longer if options are present

TCP Seq. #'S and ACKs

Seq. #'s:

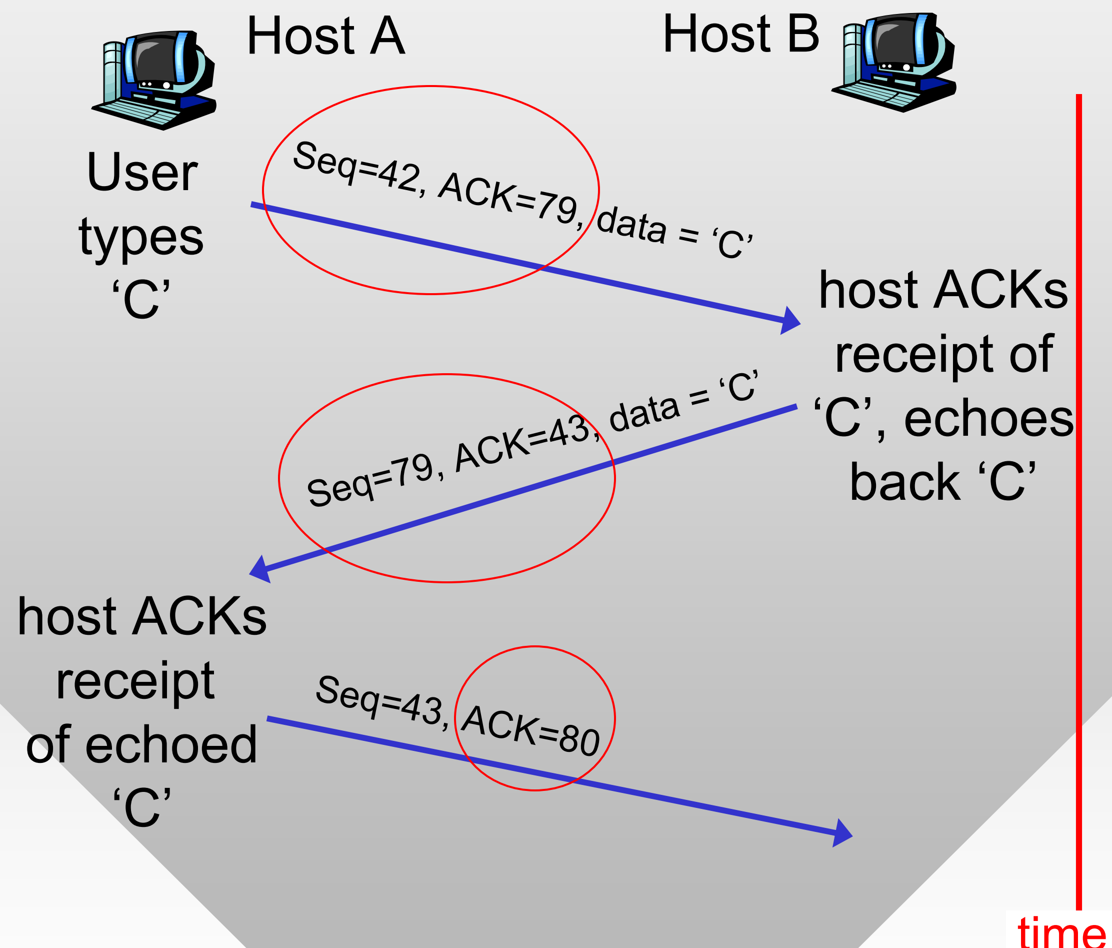
- Sequence number of the first byte in segment's data

ACKs:

- Seq # of next byte expected from sender
- Cumulative ACK

Q: how receiver handles out-of-order segments?

A: TCP spec doesn't say, up to implementor



Simple telnet scenario

TCP Round Trip Time and Timeout

Q: how to set TCP timeout value (RTO)?

- Want it slightly larger than the next RTT
  - But the RTT varies
- Too short: premature timeout
  - Unnecessary retransmissions
- Too long: slow reaction to segment loss
  - Protocol may stall, exhibit low performance

- Idea: dynamically measure RTT, average these samples, then add safety margin
- `SampleRTT`: measured time from segment transmission until ACK receipt
  - Ignore retransmissions, why?
- `SampleRTT` will vary, want estimated RTT "smoother"
  - Average several recent measurements, not just current **`SampleRTT`**

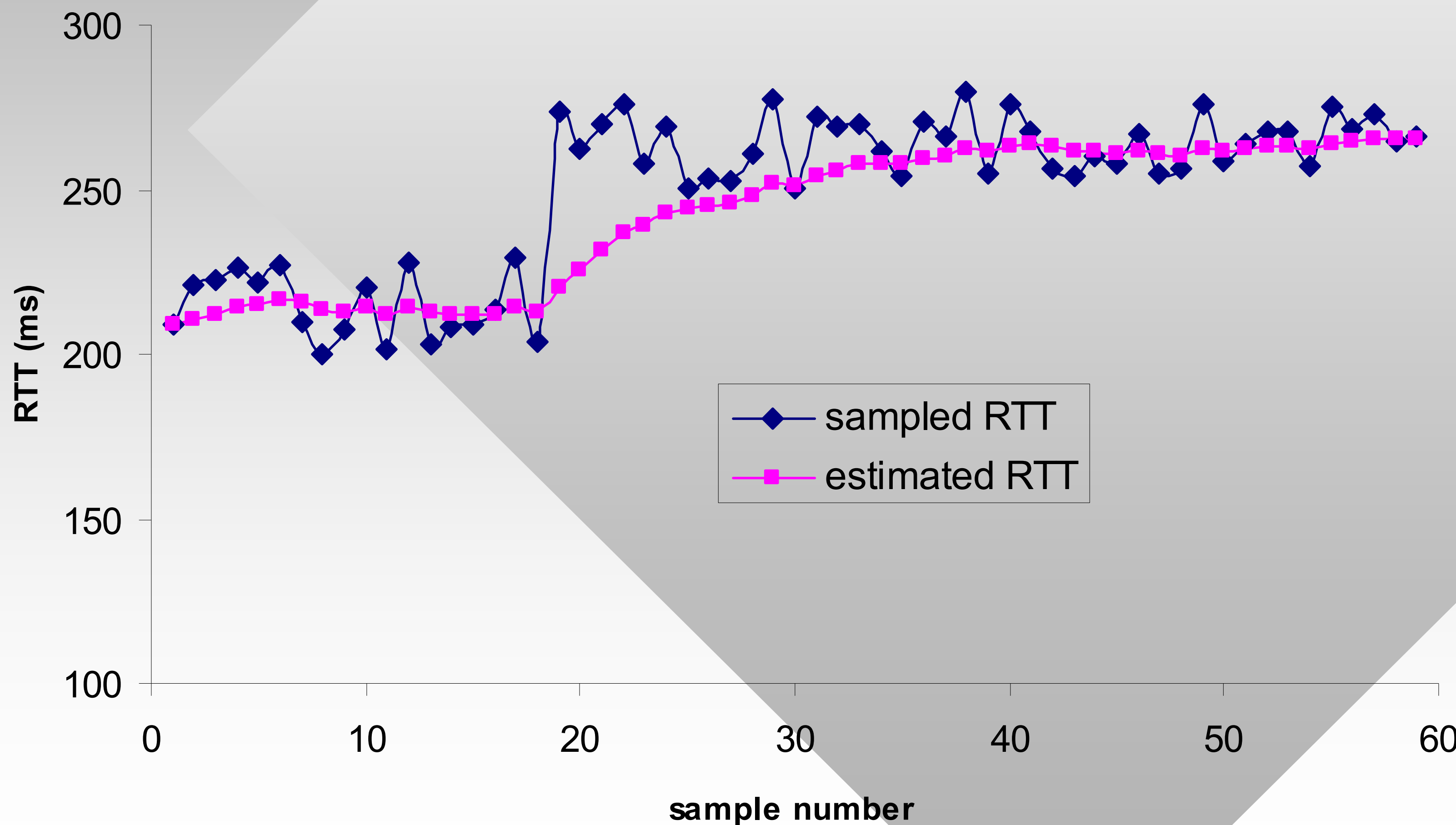
TCP Round Trip Time and Timeout

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

- Exponentially weighted moving average (EWMA)
  - Influence of past sample decreases exponentially fast
  - Typical value: $\alpha = 1/8$
- Task: derive a non-recursive formula for `EstimatedRTT`(n)
  - Assume `EstimatedRTT(0) = SampleRTT(0)`
  - Let $Y(n) =$ `EstimatedRTT`(n) and $y(n) =$ `SampleRTT`(n)

$$Y(n) = (1-\alpha)^n y(0) + \alpha \sum_{i=0}^{n-1} (1-\alpha)^i y(n-i)$$

Example RTT Estimation:



TCP Round Trip Time and Timeout

- Setting the timeout:
- `EstimatedRTT` plus a "safety margin"
  - Larger variation in `EstimatedRTT` → larger safety margin
- First estimate how much SampleRTT deviates from EstimatedRTT (typically, $\beta = 1/4$):

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

Then set retransmission timeout (RTO):

```
RTO(n) = EstimatedRTT(n) + 4*DevRTT(n)
```

Example Timeout Estimation: