Lesson 3.4: **Transport Layer**

CSC450 – COMPUTER NETWORKS | WINTER 2019-20

DR. ANDREY TIMOFEYEV

OUTLINE

- TCP timer management.
- TCP flow control.

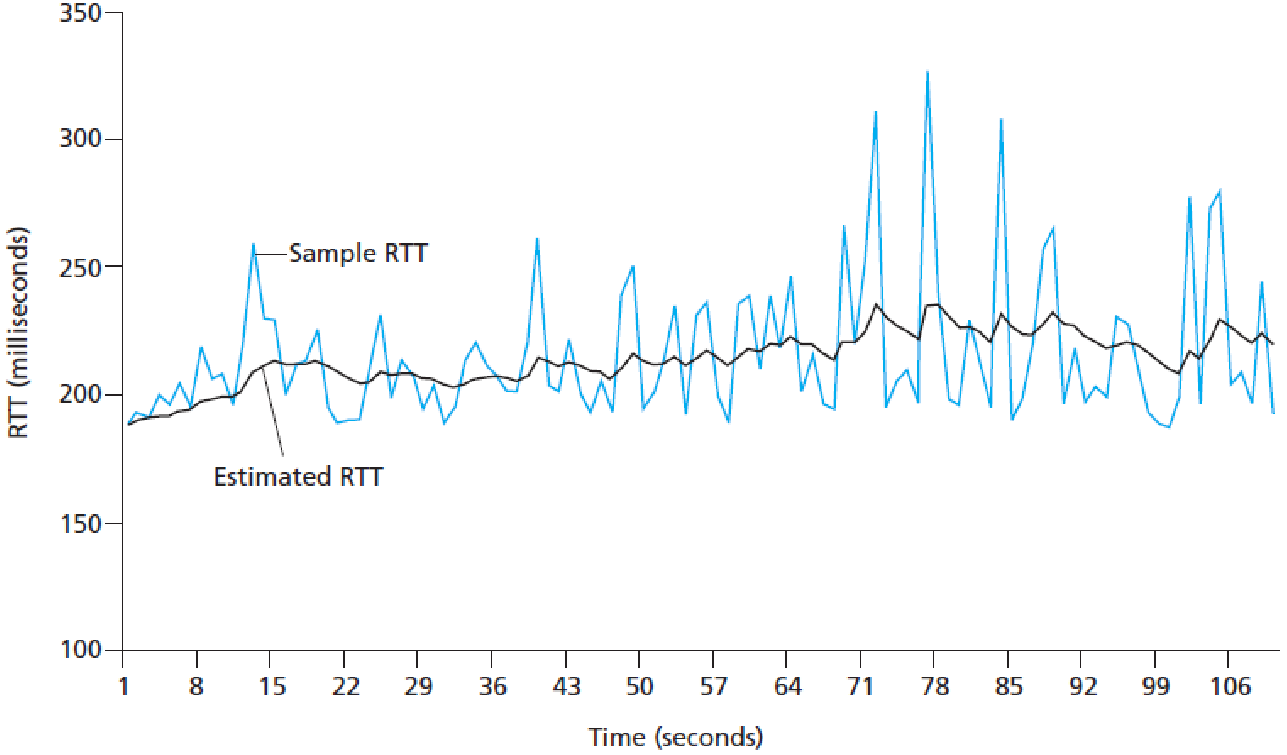
TCP TIMER MANAGEMENT: RTT ESTIMATION (1)

- **Timeout** of **TCP timer** has to be **longer** than Round-Trip Time (**RTT**).
  - Too **short** – **premature** timeout & **unnecessary** retransmission.
  - Too **long** – **slow** reaction to **lost** segments.

- **RTT estimation** in TCP:
  - *SampleRTT* – measured **time** from **segment transmission** until **ACK receipt**.
    - Varies from segment to segment, need something “smoother”.
  - $EstimatedRTT = (1 - \alpha) \times EstimatedRTT + \alpha \times SampleRTT$
    - **Weighted average** of SampleRTT values.
    - Recommended $\alpha = 0.125$.
  - $DevRTT = (1 - \beta) \times DevRTT + \beta \times |SampleRTT - EstimatedRTT|$
    - **Weighted average** of difference between SampleRTT and EstimatedRTT.
    - Recommended $\beta = 0.25$.

TCP TIMER MANAGEMENT: RTT ESTIMATION (2)

- **Timeout** of **TCP timer** has to be **longer** than Round-Trip Time (**RTT**).
  - Too **short** – **premature** timeout & **unnecessary** retransmission.
  - Too **long** – **slow** reaction to **lost** segments.



RTT samples and RTT estimates

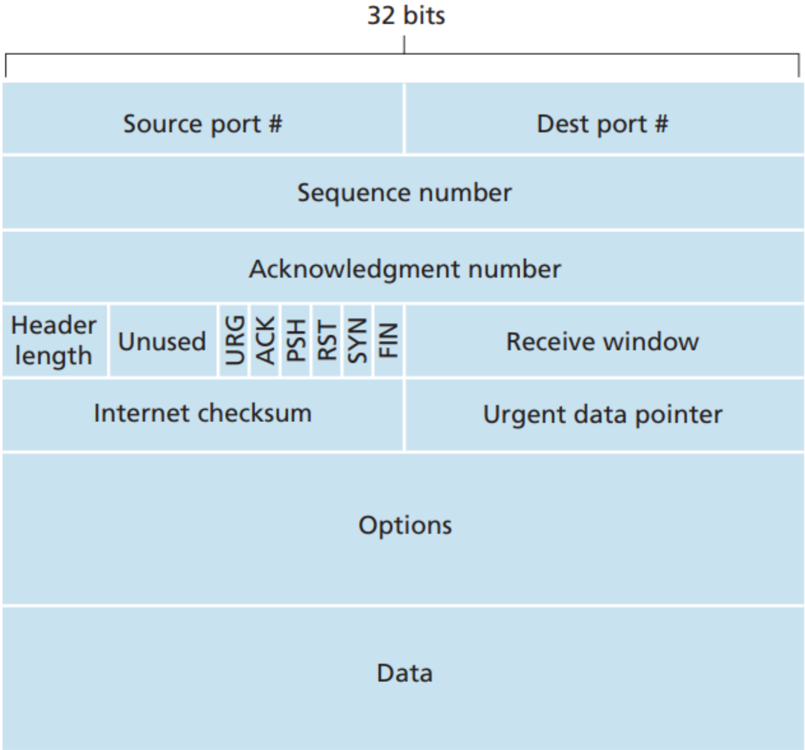
TCP TIMER MANAGEMENT: TIMEOUT INTERVAL

- **Timeout interval** should be **≥** *EstimatedRTT*, but not **>>** *EstimatedRTT*.
- **Timeout interval** = *EstimatedRTT* + "*safety margin*".
  - **High** deviation → **large** margin.
  - **Low** deviation → **small** margin.
- *TimeoutInterval = EstimatedRTT + 4*DevRTT.*
  - Initial *TimeoutInterval* is set to **1 second**.
    - **Updated** once segment received and *EstimatedRTT* is **calculated**.

TCP FLOW CONTROL (1)

- **Flow control** – **service** provided by TCP (to applications) to **eliminate** the possibility of sender **overflowing** the receivers buffer.
  - **“Speed matching”** service – **matching** the rate at which **sender** application is **sending** and **receiver** application is **reading** the byte stream.

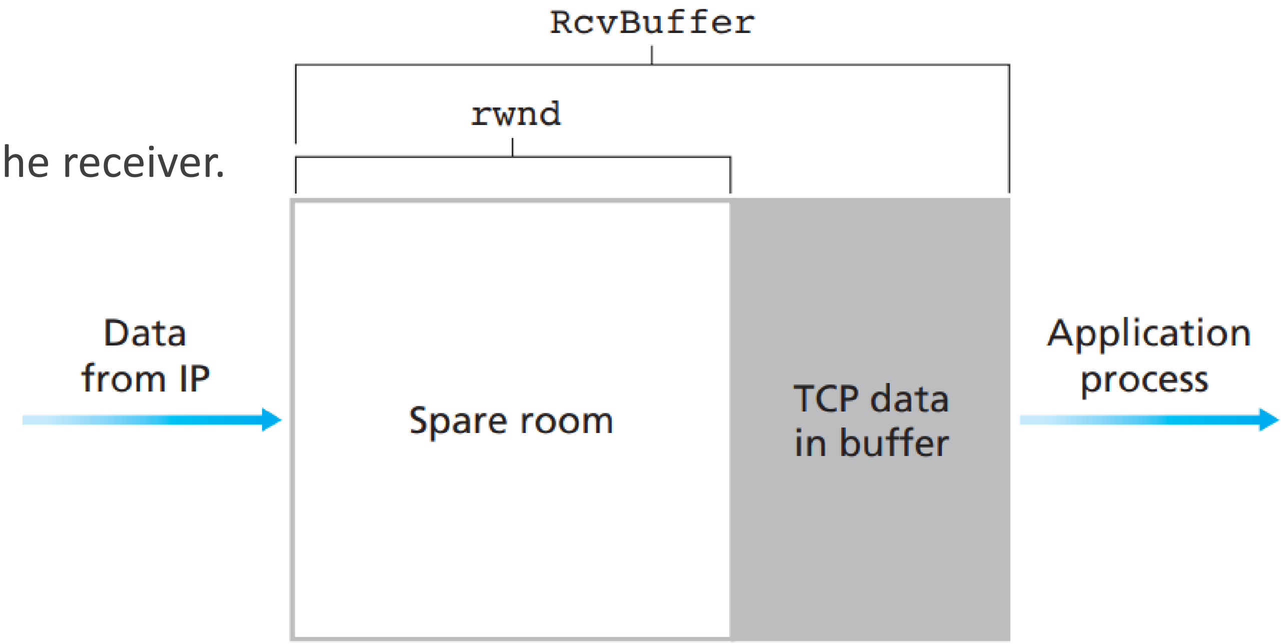
- **TCP flow control** overview:
  - **Receiver** side:
    - ***RcvBuffer*** – **size** of the receive **buffer**.
      - Set via socket options or dynamically by the OS.
    - ***Rwnd*** – **advertised** free buffer **space**.
      - **Receive Window** header field in receiver-to-sender segment.
  - **Sender** side:
    - **Limits** its window size (number of unACKed segments) to ***rwnd*** value.
    - **Guarantees** receive buffer will not overflow.



TCP segment structure

TCP FLOW CONTROL (2)

- **TCP flow control** details:
  - **Receiver** side:
    - ***LastByteRead*** – number of last byte read by the application process from receive buffer.
    - ***LastByteRcvd*** – number of last byte arrived from network and placed in receive buffer.
    - $\mathit{LastByteRcvd} - \mathit{LastByteRead} \leq \mathit{RcvBuffer}$
    - $\mathit{rwnd} = \mathit{RcvBuffer} - [\mathit{LastByteRcvd} - \mathit{LastByteRead}]$
  - **Sender** side:
    - ***LastByteSent*** – number of last byte sent by the sender.
    - ***LastByteAcked*** – number of last byte acknowledged by the receiver.
    - $\mathit{LastByteSent} - \mathit{LastByteAcked} \leq \mathit{rwnd}$



Receive buffer

TCP FLOW CONTROL (3)

- **Scenario**:
  - Receive **buffer** is **full** (*rwnd = 0 / Receive Window = 0*) and receiver has **nothing** to send.
  - Sender is **blocked** and cannot send more **application** data.
    - How will sender eventually know when the buffer is **free** again?
  - **Solution**: sender is required to send **control segment** with 1 byte of data.
  - Receiver **acknowledges** this control segment and **updates** *Receive Window* header field.
    - Once *rwnd* $\neq$ *0, Receive Window* is updated and the sender can send more application data.

TCP Flow Control animation

SUMMARY

- Sample RTT.
- Estimated RTT.
- Deviation RTT.
- Timeout interval.
- Receive buffer size.
- Advertised free buffer space.