lecture 5/ Transport layer

- Overview
  - Application layer (web, mail, etc)
  - → Transport layer (process-to-process data transfer)
  - Network layer (transports pkts between nodes in network) IP 4/6
  - Link Layer (transports pkts between neighboring nodes) encoding, framing, shared medium, reliable data transfer
  - Physical layer (transports bits between neighboring nodes) info theory, physical media, Shannon capacity
- Types of end-to-end services and protocols
  - unreliable pkt datagram delivery → UDP
  - reliable bytestream service → TCP
  - request/reply service → RPC (Remote Procedure Call)
  - real-time service (no guarantee of timely delivery) → → RTP (Real-time Transport Protocol)
- UDP
  - Trivial host-to-host protocol (on top of IP)
  - Each application pkt is sent as an independent IP pkt no connection setup
  - IP's best effort service: pkts may be lost/reordered
  - Packet size limited to IP capacity
  - Provides multiplexing/demultiplexing
  - Provides error detection using simple checksum
  - message format:
    - src 16 · length 16 · data 32
    - dst 16 · checksum 16

    (UDP pkt)
  - connectionless demultiplexing

TCP

- Provides reliable, in-order delivery (using ACKs & retransmissions)
- Provides multiplexing/demultiplexing
- Full-duplex
- Connection-oriented (different sockets per each connection)
- Flow-control: prevents sender from overrunning receiver
- Congestion-control: prevents sender from overloading the network
- Byte-oriented: apps write/receive stream of bytes
  (although network transports them in packets)
  framing in app. layer

End-to-end issues

- At link layer, we used retransmissions to make it reliable
- Differences from end-to-end protocols:
  - RTT is not known beforehand (varies and changes)
  - pkts may get reordered in the Internet (TTL=120s)
  - bandwidth-delay product and buffer space at peer are not known beforehand
  - max. packet size on entire path is unknown

TCP Header

- src port 16
- dst port 16
- seq. num (first byte in segment) 32
- ~~Ack num~~
- seq num (of)

TCP Header

- src port 16
- dest. port 16
- seq num 32 (of first byte of this segment)
- Ack num 32 (seq num of next byte)
- header length/4 4
- unused 6
- flags:
  - URG 1
  - ACK 1
  - PSH 1
  - RST 1
  - SYN 1
  - FIN 1
- advertised window size 16
- TCP checksum 16
- urgent pointer 16
- options (var length) 3
- data

TCP sequence numbers and ACKs

~~byte stream~~

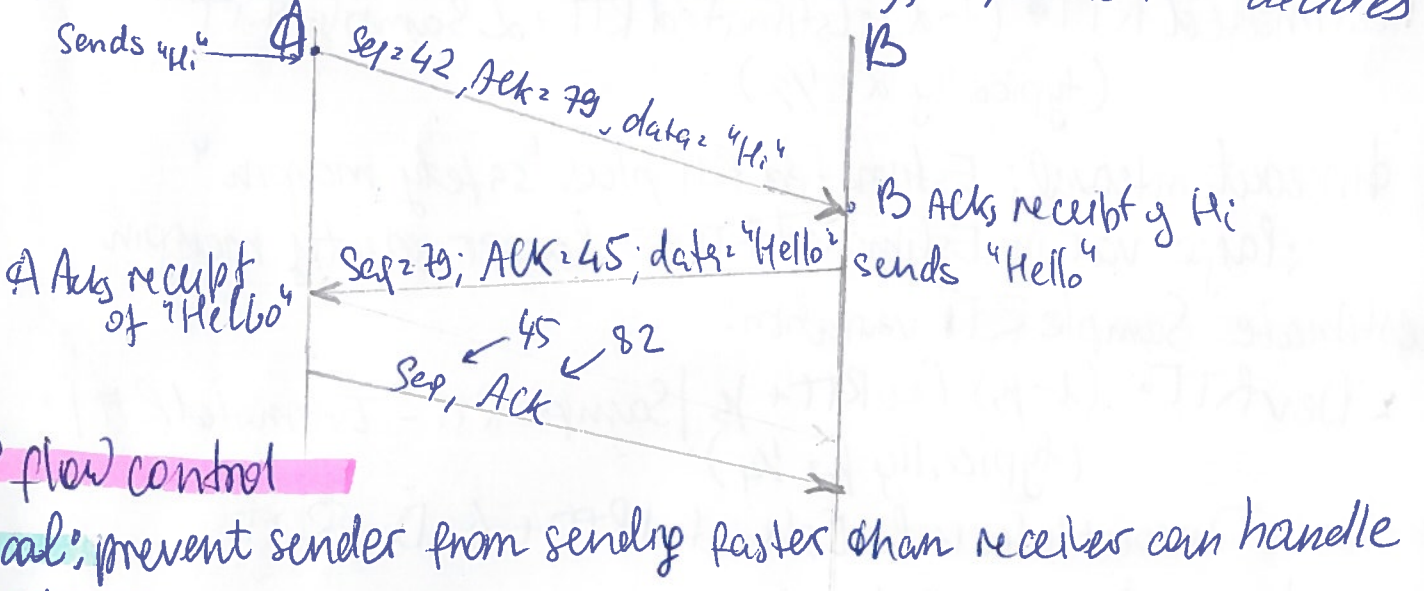
sequence numbers:

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

ACKs

- seq. nr. of next byte expected from the other side
- ACKs are cumulative

out-of-order segments? — TCP does not say; implementor decides

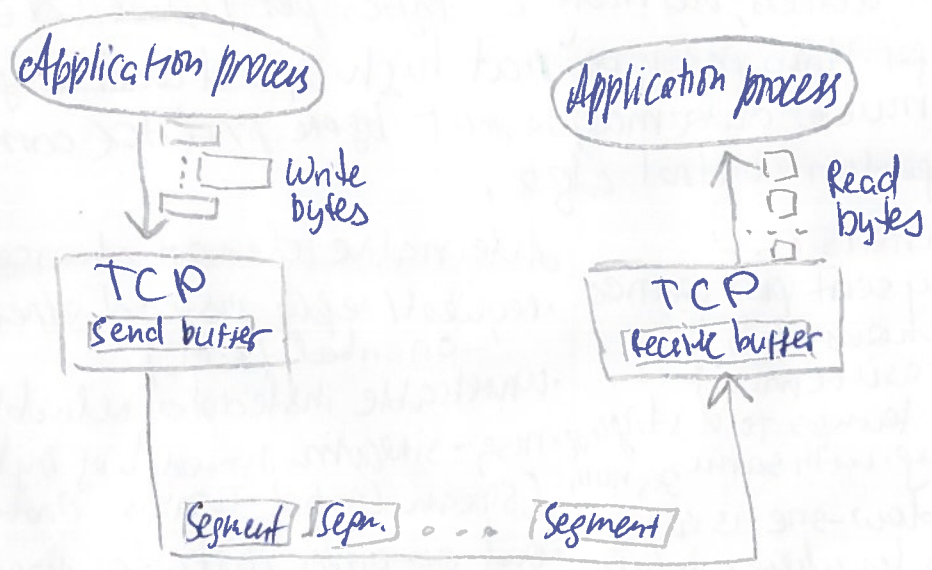


TCP flow control

- Goal: prevent sender from sending faster than receiver can handle

Principle:

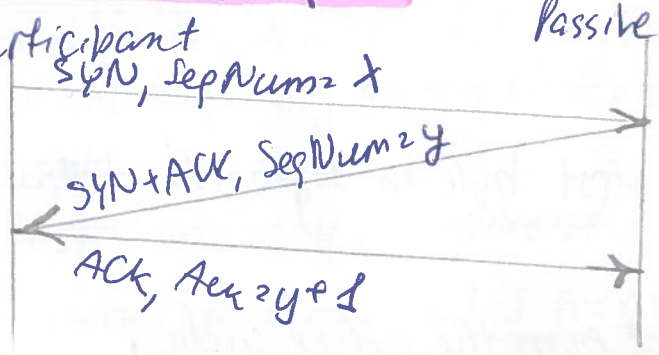
- Receiving TCP advertises remaining buffer space ("advertised window" in TCP Header)
- Sender is not allowed to send more data than fits in receiver's window



if receiving app stops reading:

- receive buffer fills up
- advertised window decreases, to zero
- sending TCP cannot send data
- send buffer fills up
- sending application will be blocked if no buffer space left ("backpressure")

- TCP connection setup



refer to TCP state-transition diagram

"three-way handshake"

- When to transmit?
  - Estimated RTT = $(1-\alpha)$ · Estimated RTT + α Sample RTT
    (typically $\alpha = 1/8$)

  timeout interval: Estimated RTT plus "safety margin"
  - large var in Estimated RTT → larger safety margin
  - estimate Sample RTT variation:

    DevRTT = $(1-\beta)$ DevRTT + β |SampleRTT − EstimatedRTT|
    (typically $\beta = 1/4$)
  - then: Timeout Interval = Estimated RTT + 4 · DevRTT
  - do not use retransmitted pkt for RTT measurements
- Problems with TCP at high speed
  - Sequence number wrap: TCP/IP ttl is 120s. Seq num has 32 bits. To guarantee uniqueness, no more 2^{32} bytes per 120sec (256 Mbps)
  - window too small for "long fat pipe": at high speed and large transport delay much data may be sent before first ACK comes in
  - problems with congestion control algo
- TCP Header extensions
  - Timestamp: every sent pkt carries timestamp, ACK echoes this; accurate RTT measurement
  - PAWS: same timestamps to distinguish old from new seq. with same seq num
  - Window scaling: window-size is not in bytes, but as 2^k bytes, where k is new.
  - Selective Ack (SACK): ack for data that has been received out of sequence.

Alternative design choices
- request/reply instead stream-oriented (RPC)
- unreliable instead of reliable (RTP)
- msg-stream instead of byte-stream (Stream Control Trans. Protocols)
- out-of-order instead of in-order delivery (SCTP)
- no explicit set-up/teardown phases
- rate-based instead of window-based