

# TCP

## □ Transport protocol:

- Communication between applications
- API: sockets
- Uses IP as network protocol
- De-/Multiplexing via port numbers

## □ Point-to-point:

- One sender, one receiver

## □ Full duplex data:

- MSS: maximum segment size
  - IP is packet switching
- Bi-directional data flow in same connection
  - Bi-directional byte stream

# TCP (cont.)

## □ Pipelined

- Multiple packet in flight
- Controlled via sliding window of size n:
  - Can send up to n bytes without ack
  - When data acked window slides forward

## □ Flow controlled

- Sender will not overwhelm receiver
- Use receiver side window
- Receiver explicitly informs sender of (dynamically changing) amount of free buffer space
- Depends on consuming application
- Persist timer
  - If  $rwnd = 0$
  - Exponentially backed off (up to 60 s)

# TCP (cont.)

- ❑ **Reliable, in-order byte stream**
  - No “message boundaries”
  - Sequence numbers (per byte)
  - Acknowledgements (per byte)
    - Cumulative
    - Selective
    - Delayed
      - Max 2 packets or 200 ms
      - Always ACK out of order data
  - Retransmissions
    - Timeout based on RTT estimation
    - Three duplicated ACKs

# TCP (cont.)

- **Reliable, in-order byte stream (cont.)**
  - RTT (round trip time) estimation:
    - Smoothed RTT estimation
      - $RTT = a * RTT + (1-a) * \text{measured RTT}$
    - Single timer for all connections
      - Typically every 500 ms
    - Traditional:
      - Single packet per window
      - Invalid by retransmitted packets
    - New:
      - Timestamp option for every window
  - RTO (recovery time objective):
    - Static:  $RTO = b * RTT$  ( $b=2$ )
    - Dynamic:  $RTO = RTT + 4 * D$ 
      - D = smoothed RTT deviation

# TCP (cont.)

- ❑ **Reliable, in-order byte stream (cont.)**
  - Small packets == silly window syndrome:
    - Sender side (Nagle)
      - Only one partial packet outstanding
    - Receiver side (Clark)
      - Only advertise reasonable window changes
      - $\text{Min}(\text{MSS}, \frac{1}{2} \text{ of receiver buffer space})$

# TCP (cont.)

## □ Connection-oriented

- Handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- Control flags:
  - SYN: connection establishment
  - FIN: connection close
  - RST: connection reset
  - SYN, FIN use one byte of segment space enables reuse of existing mechanisms
- Connection establishment:
  - 3-way handshake
- Connection teardown
  - 4-way handshake
- Initial sequence number: best unpredictable
- Receiver state: for flow control
- Time wait state: avoid reuse of sockets

# TCP (cont.)

## □ TCP congestion control

- Sender will not overwhelm network
- End-to-end control
- Congestion detection
  - Lost packets
  - Marked packets
- Use sender side window
  - Cwnd
- AIMD for window size control
  - Additive increase
  - Multiplicative decrease

# TCP (cont.)

## □ TCP congestion control (cont.)

- Selfclocking
  - ACK clocking
- Two stages
  - Reaching equilibrium
    - Slow start
  - Adapting to resource availability
    - Congestion avoidance

# TCP (cont.)

## □ TCP congestion control (cont.)

### ○ Slow start

- Init:
  - $\text{cwnd} = \text{MSS}$
  - $\text{ssthresh} = 64\text{K}$
- ACK:
  - $\text{cwnd} += \text{MSS}$
  - If ( $\text{cwnd} > \text{ssthresh}$ )  
congestion avoidance
- Timeout:
  - $\text{cwnd} = \text{MSS}$
  - $\text{RTO} = \min(2 * \text{RTO}, 64 \text{ s})$
  - restart

# TCP (cont.)

## □ TCP congestion control (cont.)

- Congestion avoidance
  - ACK:
    - $\text{cwnd} += \text{MSS}/\text{cwnd}$
  - Lost packet indication:
    - $\text{ssthresh} = \max(\min(\text{rwnd}, \text{cwnd})/2, 2*\text{MSS})$
    - $\text{RTO} = \min(2*\text{RTO}, 64 \text{ s})$
    - Cont or switch to slow start

# TCP (cont.)

## □ TCP congestion control (cont.)

- Retransmissions
  - Fast retransmit
    - Receiver acks out-of-order segments immediately
    - $\geq 3$  duplicate ACKs  $\Leftrightarrow$  lost packet
    - Retransmit packet
    - Switch to slow start
  - Fast recovery
    - Fast retransmit
    - Congesting avoidance
    - (Allowed to transmit packet for every dup ACK)
  - Partial ACK
    - Not all outstanding data is Acked after retransmission
    - Retransmit next packet