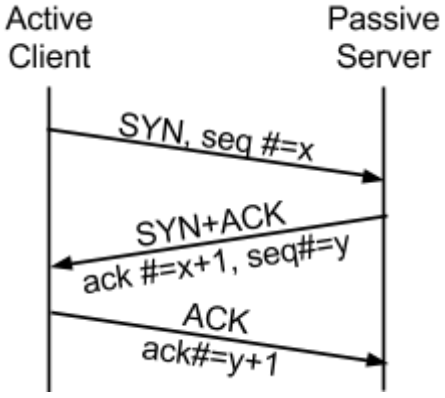
TCP Review

Outline

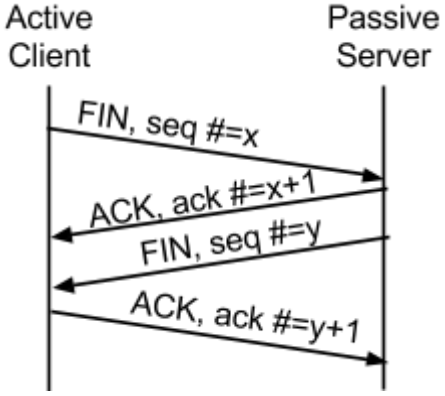
- Connection establishment and termination
- Flow control
- Congestion control

Connection Establishment and Termination

- Three-way handshake

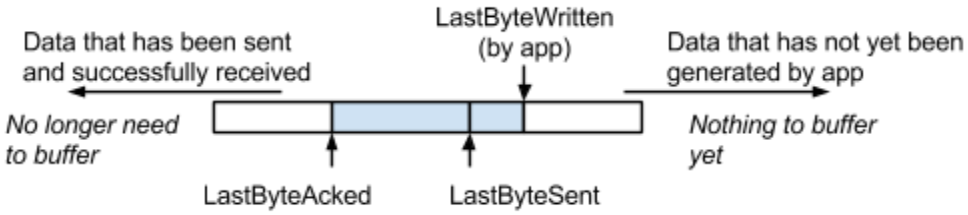


- Four-way handshake

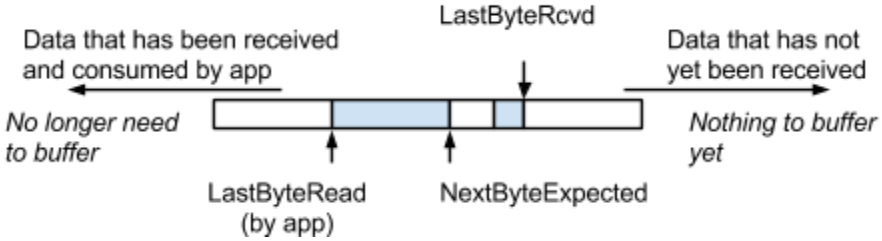


Flow control

- Buffers have finite size -- MaxSendBuffer and MaxRecvBuffer
    - Need to limit the amount of data a sender sends to the receiver -- otherwise the receiver won't be able to buffer it and the sender will need to retransmit it
- Sender buffer



    - LastByteAcked ≤ LastByteSent -- data that has not been sent cannot be acknowledged
      LastByteSent ≤ LastByteWritten -- data that has not been generated by the sending app cannot be sent
- Receiver buffer



    - LastByteRead < NextByteExpected -- receiving app cannot consume data that has not been received; cannot consume later data when earlier data is missing
    - NextByteExpected ≤ LastByteRcvd + 1 -- cannot expect data that is beyond the latest data that was received
- Advertised window (RWND) = amount of receiver buffer capacity remaining
  RWND = MaxRecvBuffer - ((NextByteExpected-1) - LastByteRead)
- Sender must adhere to RWND:
  EffectiveWindow = RWND - (LastByteSent - LastByteAcked)
- Don't send packets < MSS, unless app requires this, since headers add overhead
    - Send one byte of data per packet
        - Send at least 54 (14 + 20 + 20) bytes of header for each byte of data
        - 1Mbps link effective data throughput <0.017 Mbps
    - Send 1000 bytes of data per packet
        - 1Mbps link effective data throughput ≈0.948Mbps
- SIlly window syndrome
    - When receiver buffer fills, RWND = 0 and no data can be sent
    - As receiver buffer starts to empty, RWND will be > 0, but may be < MSS
    - If we sent a partial packet (<MSS), then receiver will ACK smaller number of bytes and again the window will only open by a small amount
- Nagle's algorithm: prevents the sender from sending TCP packets containing less than MSS (maximum segment size) bytes of data, unless there is no unACKed data in flight (i.e., all data the sender has sent has been acknowledged)

Congestion Control

- Congestion Window (CWND) -- limits amount of data in flight based on network conditions
    - MaxWin = min(CWND, RWND)
    - EffectiveWindow = MaxWindow - (LastByteSent - LastByteAck'd)
- Additive Increase/Multiplicative Decrease (AIMD) -- decrease aggressively and increase conservatively because having CWND too large is much worse than having CWND too small
    - Start CWND = 1 pkt
    - Increase CWND by 1 pkt every RTT -- CWND += 1
    - On timeout, divide CWND in half -- CWND = CWND / 2 (never go below 1 pkt)
- Slow Start -- reach maximum bandwidth faster
    - Begin with CWND = 1, SSTHRESH = infinity
    - Double CWND every RTT -- CWND = CWND * 2
    - Switch to additive increase (AI) when CWND >= SSTHRESH
    - On loss, set SSTHRESH = CWND/2 (keep SSTHRESH >= 1) and CWND = 1
- Fast retransmit -- no longer need to wait for timeout before loss is detected and recovered; means more data can be sent sooner
    - Treat three duplicate ACKs (i.e., ACKs with same ACK #) as sign of loss
    - Retransmit packet starting with seq # contained in duplicate ACKs
- Fast recovery -- avoid running slow start every time a loss occurs; it takes too long to ramp up when bandwidth is high
    - When fast retransmit occurs
        - Set CWND = SSTHRESH when loss occurs
        - Then do additive increase
    - Use slow start only at connection start or when timeout occurs