TCP Flow Control

Outline
- Flow control
- Triggering transmissions

Flow control
- Sender buffer
  - Data that has been sent and successfully received
  - Data that has not yet been generated by app
  - LastByteAcked ≤ LastByteSent
  - LastByteSent ≤ LastByteWritten

- Receiver buffer
  - Data that has been received and consumed by app
  - Data that has not yet been received
  - LastByteRead < NextByteExpected
  - NextByteExpected ≤ LastByteRcvd+1

- Sliding window -- amount of data that can be sent without waiting for an ACK from the receiver
  - To avoid overflowing receiver buffer:
    LastByteRcvd - LastByteRead ≤ MaxRecvBuffer
  - Amount of buffer capacity remaining:
    RecvBufRemain = MaxRecvBuffer - ((NextByteExpected-1) - LastByteRead)
    • ACKs include AdvertisedWindow -- receiver buffer capacity remaining
  - Increment LastByteRcvd when data is received, so the amount of buffer remaining is reduced
    • If app is consuming data as fast as it arrives, then full receiver buffer is always available -- i.e.,
      AdvertisedWindow = MaxRecvBuffer
  - Sender must adhere to AdvertisedWindow:
    LastByteSent - LastByteAcked ≤ AdvertisedWindow
    EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)
    • EffectiveWindow must be > 0 before source can send more data
    • Also sending application cannot overflow sender buffer:
      LastByteWritten - LastByteAcked ≤ MaxSendBuffer
Triggering Transmissions

- Sender can transmit as soon as it has buffered enough bytes to fill a packet
  - Maximum segment size (MSS) = the maximum data bytes that can fit in one packet -- based on maximum packet size physical layer allows minus IP and TCP header size
- App can explicitly request transmission of buffered data
  - Needed for highly interactive apps -- e.g., SSH
  - PSH flag is set in TCP header
- 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, AdvertisedWindow = 0 and no data can be sent
  - As receiver buffer starts to empty, AdvertisedWindow 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
  - The problem can continue to exist indefinitely
- Nagle’s algorithm
  - Application writes data to buffer
    - If unsent data in buffer ≥ MSS and advertised window ≥ MSS
      - Send packet with MSS bytes
    - Else
      - If unacked data in buffer
        - Do not send any data now
      - Else
        - Send all new data now
  - When ACK arrives
    - If unsent data in buffer ≥ MSS and advertised window ≥ MSS
      - Send packet with MSS bytes
    - Else
      - Do not send any data now
  - Can disable Nagle’s algorithm for interactive apps -- e.g., SSH

**Exercises**

- In-class scenario 1: receiver consumes data faster than sender generates data
  - MSS = 1000 bytes
  - RTT = 50 ms
  - Sender generates 1000 bytes every 50ms
  - Receiver consumes 1000 bytes every 25ms
  - Receiver’s buffer size = 3000 bytes
In-class scenario 2: receiver consumes data slower than sender generates data
- MSS = 1000 bytes; RTT = 50 ms
- Sender generates 1000 bytes every 25ms
- Receiver consumes 500 bytes every 25ms
- Receiver’s buffer size = 2000 bytes
- Timeline with and without Nagle’s algorithm

Scenario 3:
- MSS = 1000 bytes; RTT = 50 ms
- Sender generates 1000 bytes every 25ms
- Receiver consumes 500 bytes every 25ms
- Receiver’s buffer size = 3000 bytes
○ Timeline **without** Nagle’s algorithm

- (EffWin=3000) Generate 1000 bytes (EffWin=2000)
- (EffWin=2000) Generate 1000 bytes (EffWin=1000)
- (EffWin=1000) Generate 1000 bytes (EffWin=0)
- (EffWin=500) Generate 1000 bytes (EffWin=0)
- (EffWin=500) Generate 1000 bytes (EffWin=0)
- (EffWin=500) Generate 1000 bytes (EffWin=0)

Timeline **with** Nagle’s algorithm

- (EffWin=3000) Generate 1000 bytes (EffWin=2000)
- (EffWin=2000) Generate 1000 bytes (EffWin=1000)
- (EffWin=1000) Generate 1000 bytes (EffWin=0)
- (EffWin=500) Generate 1000 bytes (EffWin=0)
- (EffWin=1000) Generate 1000 bytes (EffWin=0)
- (EffWin=0) Generate 1000 bytes (EffWin=0)
- (EffWin=1000) Generate 1000 bytes (EffWin=0)

Buffer Available=3000

-Nagle's algorithm