Chapter 5 End-to-End Protocols

TCP (continued)

- Triggering Transmission
  - It is up to TCP to decide that it has enough bytes to send a segment
  - Assume the window is wide open, TCP has 3 mechanisms to trigger the transmission of a segment
    - TCP maintains a variable called *maximum segment size* (MSS) and sends a segment as soon as it has collected MSS bytes from the sending process
      - MSS is usually set to the size of the largest segment TCP can send without causing local IP to fragment, i.e., MSS = MTU of directly connected network – TCP header size - IP header size
    - Sending process has explicitly asked TCP to send it using the *push* operation
    - When a timer fires (see Nagle’s algorithm below)
      - Resulting segment contains as many bytes as are currently buffered for transmission
  - Early implementations of TCP aggressively take advantage of any available window and suffer the Silly Window Syndrome (SWS)
    - SWS is the transmission of small segments because either the receiver advertises a small window or the sender transmits a small segment
    - SWS makes data transmission extremely inefficient
Receive-side SWS avoidance: Clark’s solution
- Receiver closes the window until the buffer is half empty or the available buffer space is equal to MSS.

Send-side SWS avoidance: Nagle’s algorithm
- If available data is less than MSS, then wait some amount of time before sending the available data
- But how long? Nagle introduced an elegant *self-clocking* solution
  - As long as TCP has any data in flight, the sender will eventually receive an ACK
  - This ACK can be treated like a timer firing, triggering the transmission of more data

Nagle’s algorithm:
When the application produces data to send
if both the available data and the window $\geq$ MSS
send a full segment
else
  if there is unACKed data in flight
    buffer the new data until an ACK arrives
  else
    send all the new data now

Adaptive Retransmission
- Given the range of possible RTTs between any pair of hosts in the Internet, as well as the variation in RTT between the same two hosts over time, TCP uses an adaptive retransmission mechanism
  - Timeout value is set as a function of the estimated RTT between a pair of hosts
• Original algorithm
  • Measure SampleRTT for each segment/ACK pair
  • Compute weighted average between the previous estimate and the new sample:
    \[ \text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1 - \alpha) \times \text{SampleRTT} \]
    \( \alpha \) between 0.8 and 0.9
  • TimeOut = 2 x EstimatedRTT

• Problem of the original algorithm: When a segment is retransmitted and then an ACK arrives at the sender, it is impossible to decide if this ACK should be associated with the first or the second transmission for measuring the sample RTT

• Karn/Partridge Algorithm
  • Do not measure SampleRTT when retransmitting
  • Doubles timeout after each retransmission
    • Motivation: TCP source should not react too aggressively to a timeout since congestion is the most likely cause of lost segments

• Jacobson/Karels Algorithm
  • Take the variance of the sample RTTs into account
    • If the variance among SampleRTTs is small, the Estimated RTT can be better trusted and there is no need to multiply it by 2 to compute the timeout
    • On the other hand, a large variance in SampleRTTs suggest that timeout value should not be tightly coupled to the Estimated RTT
Calculating the timeout:

- Difference = SampleRTT – EstimatedRTT
- EstimatedRTT = EstimatedRTT + (δ x Difference)
  - 0 < δ < 1, typically set to 1/8
- Deviation = Deviation + δ (|Difference| - Deviation)
- TimeOut = µ x EstimatedRTT + φ x Deviation
  - µ typically set to 1 and φ set to 4.
  - When the variance is small, TimeOut is close to EstimatedRTT; a large variance causes the deviation term to dominate the calculation

TCP Extensions

- There are extensions to TCP that are realized as options that can be added to the TCP header
  - Two hosts may agree to use the options during TCP’s connection establishment phase
- RTT Measurement Option
  - Used to accurately measure RTT
  - TCP reads the system clock when it is about to send a segment, and puts this time (a 32-bit timestamp) in the segment’s header
  - Receiver echoes the timestamp back in its ACK
  - Sender subtracts the timestamp from the current time to measure the RTT
- Protect against Wrapped Sequence Numbers Option
  - Uses the 32-bit timestamp to effectively extend the sequence number space
- Window Scale Option
  - Allows TCP to advertise a larger window
Selective Acknowledgment (SACK) Option

- Allows TCP to augment its cumulative ACK with selective ACK of any additional segments that have been received but aren’t contiguous with all previously received segments
- Without SACK, there are only two reasonable strategies for a sender
  - The **pessimistic strategy** responds to a timeout by retransmitting not just the segment that timed out, but any segments transmitted subsequently
  - The **optimistic strategy** responds to a timeout by retransmitting only the segment that timed out
- With the SACK option, sender can retransmit just the segments that fill the gaps between the segments that have been selectively ACKed