School of Computing and Information Systems COMP30023: Computer Systems Tutorial Week 6

Flow Control, Congestion Control

1. What is the difference between congestion control and flow control? What methods (as discussed in the lectures) are used in TCP to provide each functionality?

- congestion control: associated with ability of network to handle current traffic, e.g. not losing packets along the route by overflowing router buffers
- flow control: associated with rate at which receiver is able to receive data
- congestion control is implemented via the congestion window, which is maintained by the sender, and is adjusted in response to a segment loss (timeout or triple duplicate ACK), and is increased when ACKs are received. This enables TCP to adjust the rate depending on perceived network conditions
- flow control is implemented by the receive window, which quantifies the amount of data the receiver has remaining in its receive buffer. This is signalled to the sender on every ack via the receive window header.

2. How can a deadlock occur in the TCP Sliding Window protocol? What is the countermeasure used in TCP to prevent this scenario happening?

- deadlock can occur if the receiver's receive window is 0, and it has no data to send. the sender knows the receive window is 0 from the final ACK, and as the application eventually reads data from the receive window such that the receive window is now > 0, there is no way for the receiver to notify the sender that it is available for further transmission. this is the deadlock
- the countermeasure used is a zero window probe: the sender sends a one byte segment, "probing" the receiver to see if it is able to receive more data. the receiver will acknowledge these segments and once the receive window starts to free up, the sender will be notified and normal transmission can resume.

3. How does the TCP Slow Start algorithm detect potential congestion in the network? What other methods (possibly with the explicit assistance of the network layer) could be used to infer congestion in the network?

- potential congestion is detected by triple duplicate ACKs or a segment timeout
- routers experiencing heavy load can signal congestion via setting Explicit Congestion Notification header in the IP datagram. Then the transport layer sends an ECN Echo (i.e. sets the ECE bit in the TCP header), which gets sent back in the ACK, and subsequently the sender halves its congestion window.

- another method may be to use increasing round-trip delay to indicate increasing network congestion

4. Suppose that the network is not congested. How does the congestion window size cwnd get increase in each transmission round in the Slow Start, Congestion Avoidance and Fast Recovery algorithms, respectively?

- Slow start: cwnd increases by 1 MSS per ACK, i.e. doubles every RTT
- Congestion avoidance: cwnd increases by 1 MSS per RTT i.e. linear increase w.r.t. RTT
- Fast recovery: cwnd increass by 1 for each duplicate ACK for the missing segment that caused TCP to enter fast recovery

5. When will the congestion control enter into the Congestion Avoidance algorithm from the Slow Start algorithm?

- when cwnd $\geq$ ssthresh

6. How does the Slow Start algorithm handle segment loss events?

- timeout: restart slow start, with less agressive upper bound
  - cwnd <- 1 MSS
  - ssthresh <- 1/2 ssthresh
- triple duplicate ACK: transition to fast recovery
  - cwnd <- ssthresh + 3
  - ssthresh <- ssthresh/2

7. How does the Congestion Avoidance algorithm handle segment loss events?
8. For the fast retransmission event, why does it set `cwnd = ssthresh + 3 MSS` rather than `cwnd = ssthresh`? Fast retransmit is caused by receipt of 3 Duplicate ACKs, which imply at least 3 segments got through. This is used to avoid diminishing `cwnd` too drastically: ideally after fast recovery, you re-enter congestion avoidance state, (assuming no packet timeout occurs) so this is an appropriate level for cwnd
9. How does the Fast Recovery algorithm perform?

- for each duplicate ACK, increase cwnd by 1 MSS
- timeout: ssthresh <-cwnd/2, cwnd=1; enter slow start
- ACK received: cwnd <- ssthresh, enter congestion avoidance

10. True or False?

(a) The size of the TCP receive window never changes throughout the duration of the connection.

- False: it changes as the receive window fills up while waiting to be read by application and it is communicated on every segment to the sender

(b) Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive window.

- True

(c) Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m + 1.

- False. Only true if the segment contained 1 byte

(d) The TCP segment has a field in its header for the receive window size.

- True. Window size field refers to receive window size

2