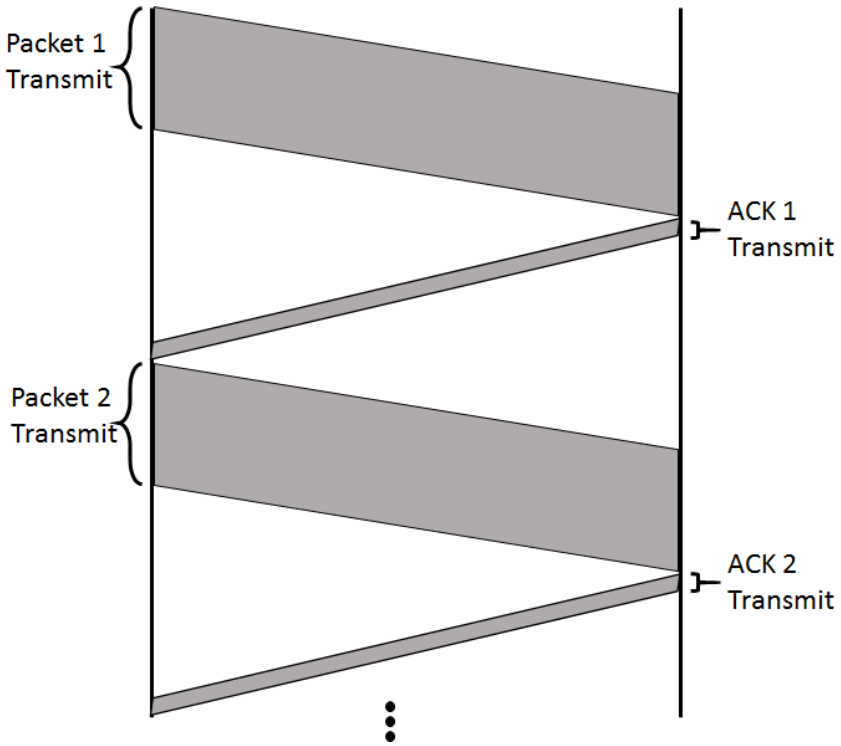
CS 372 Introduction to Computer Networks
Self-Check Exercises: Lecture 18 **Solutions**

1) What are some tradeoffs for the implementation of RDT in TCP
   Additional overhead
   Possible delays due to waiting for acknowledgements
   More packets on the network (because of ACKs) – leading to congestion issues
   (more answers possible)

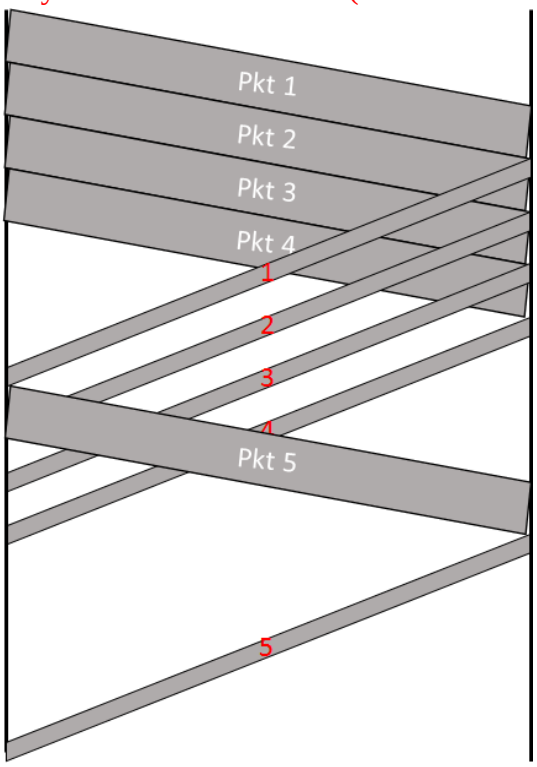
2) With a stop-and-wait implementation
   a. How long would it take to send five 1500-byte packets, subject to the following conditions:
   - R = 20Mbps
   - Network end-to-end time (after initial transmission) = 20 ms
   - ACK packet L = 20 bytes

   Initial Transmit = 1500 * 8 / 20,000,000 = 0.6 ms.
   Propagation = 20 ms
   ACK transmit = 20 * 8 / 20,000,000 = 0.008 ms
   Propagation = 20ms
   First packet total = 0.6 + 20 + 0.008 + 20 = 40.608ms (Just the first packet)
   Repeat 5x = 203.04ms



b. What is the utilization?
Time spent transmitting = 5*(0.6 + 0.008) = 3.04ms
Total time until finished = 203.04ms (from above)
3.04ms / 203.04ms = approx. 1.5 %

3) For the situation in question 2, but with a pipelined implementation…
   a. How long would it take to send five 1500-byte packets? Assume the pipeline will accommodate 6000 bytes from sender to receiver at any given time.
   6000 / 1500 = 4 packets will fit in the pipeline. We can send the 5th packet when we receive the ACK for the first packet. See diagram below.
   So we transmit the 5th packet at time:
   1500 * 8 / 20,000,000 + 20ms + 20 * 8 / 20,000,00 + 20ms = 40.608 ms
   and we receive the ACK for the 5th packet after another…
   1500 * 8 / 20,000,000 + 20ms + 20 * 8 / 20,000,000 + 20ms = 40.608 ms.
   So we're entirely finished at 81.216 ms (much better than the previous 203.04ms)

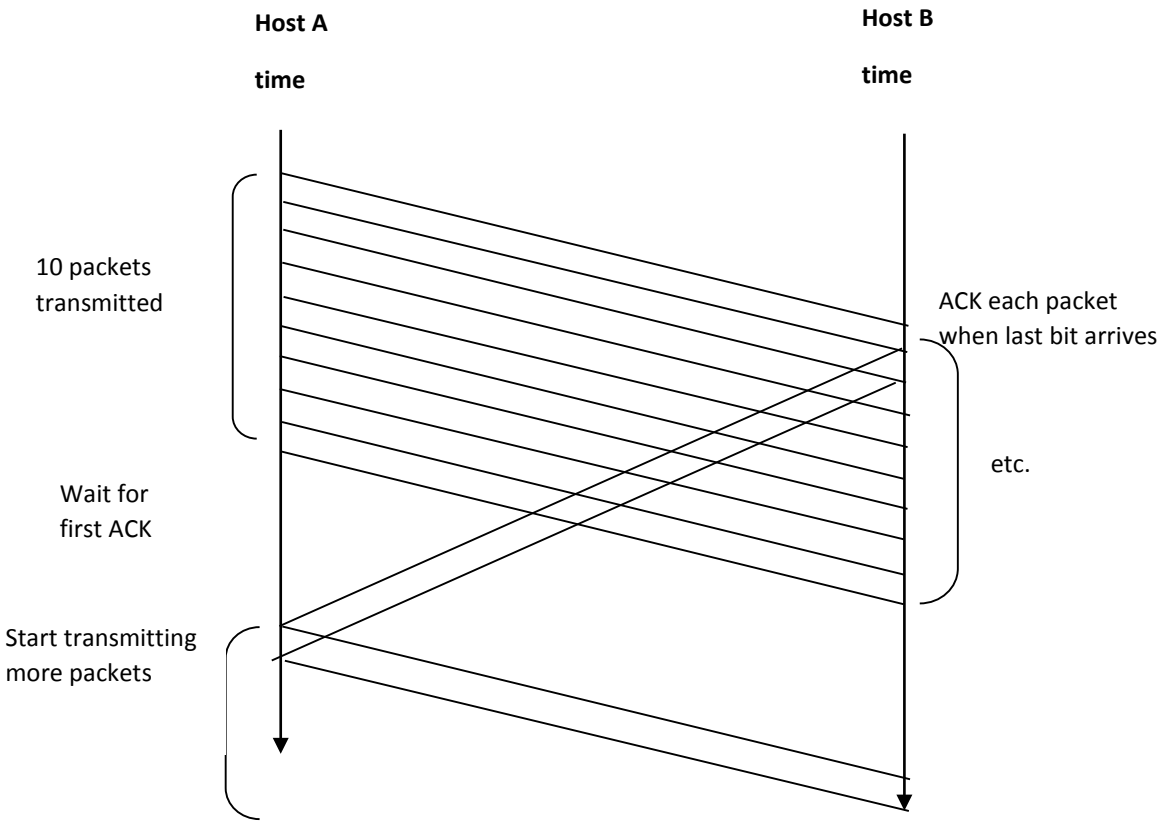


   b. What is the utilization?
   Time spent transmitting = 5*(0.6 + 0.008) = 3.04ms
   Total time until finished = 81.216 ms (from above)
   Utilization = 3.04 / 81.216 = approx. 3.74%, much better than before!
   NOTE: The reason this looks different from lecture is that, in the lecture slides, there is a continuous stream of packets (infinite packets). In this problem there are a limited number of packets.

4) How does the receiver handle multiple incoming packets?
An input buffer is filled as data comes in, and the application layer drains the buffer as it needs the data contained therein.

5) What happens if the sender sends information faster than the receiver can process?
Packets would be dropped. With a receive window implementation, the receive window size will shrink over time, eventually limiting the sender's output rate to that which can be processed by the receiver (flow control).

6) *HostA* is sending fixed-size packets to remote *HostB* on a 100 Mbps link. Each packet's length is 1200 bytes. The propagation delay is 1 ms.; the processing/queuing delays are negligible. Given a sliding window of 12,000 bytes, and assuming no pipeline errors, what is *HostA*'s utilization?
Approximately 45.8%



The time to transmit one packet is:
L/R = (1200 x 8 bits) / 10^8 bps = 0.000096 sec = 0.096 ms.
So it takes 0.96 ms to transmit 10 packets.
The RTT is 2 ms. (i.e., propagation delay x 2)
Total time before we can send packet # 11: 2.096 ms.
Percentage of time actually transmitting: 0.96 / 2.096 = approximately 0.458 = 45.8%