National University of Singapore

School of Computing

CS2105: Introduction to Computer Networks

Semester 1, 2018/2019

Tutorial 4

These questions will be discussed during the next week's discussion group meetings. **Due to time constraint, not all the questions may be discussed in class.** Please be prepared so you can request questions to discuss. Otherwise, it will be left to your tutor's discretion to choose.

Some of the questions are taken from the textbook, so please bring it along for reference.

1. [**KR, Chapter 3, R6**] Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

   Note: this will be assignment 2 :).

   Ans: One would have to implement reliability checking and recovery mechanisms (ACK, seq #, checksum, timeout, re-transmission, etc.) at the application layer. For example, sender needs to include relevant header/trailer fields in every packet.

2. Show an example that if the communication channel between the sender and receiver can reorder messages (i.e., two messages are received in different order they are sent), then protocol rdt3.0 will not work correctly.

   Ans:
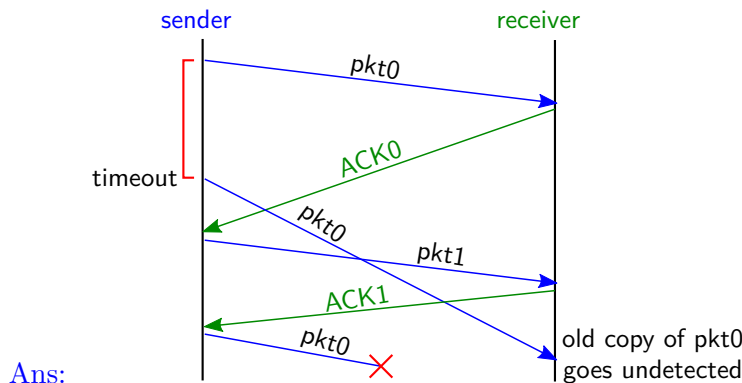
3. [**KR, Chapter 3, P29**] It is generally a reasonable assumption, when sender and receiver are connected by a single wire, that packets cannot be reordered within the channel between the sender and receiver. However, when the "channel" connecting the two is a network, packet reordering may occur.

   One manifestation of packet reordering is that old copies of a packet with a sequence or acknowledgment number of x can appear, even though neither sender's nor receiver's window contains x. With packet reordering, the channel can be thought of as essentially buffering packets and spontaneously emitting these packets at any point in the future.

   What is the approach taken in practice to guard against such duplicate packets?

   Ans: The approach taken in practice is to ensure that a sequence number is not reused until the sender is "sure" that any previously sent packets with the same sequence number are no longer in the network.

   Firstly, a large range of sequence numbers can be used to lower the change a sequence number is to be reused (TCP uses a 32-bit field). Secondly, a packet cannot "live" in the network

forever. For example, the IP protocol specifies TTL in packet header to ensure that datagrams do not circulate indefinitely in the network. This field is decreased by one each time the datagram arrives at a router along the end-to-end path. If TTL field reaches 0, router will discard this datagram.

In practice, a maximum packet lifetime of approximately three minutes is assumed in the TCP extensions for high-speed networks.

4. [**Modified from KR, Chapter 3, P37**] Host A is sending data segments to Host B using a reliable transport protocol (either GBN or SR). Assume timeout values are sufficiently large such that all data segments and their corresponding ACKs can be received (if not lost in the channel) by Host B and the Host A respectively.

   Suppose Host A sends 5 data segments to Host B and the 2nd data segment is lost. Further suppose retransmission is always successful. In the end, all 5 data segments have been correctly received by Host B.

   How many segments has Host A sent in total and how many ACKs has Host B sent in total if either GBN or SR protocol is used? What are their sequence numbers? Answer this question for both protocols.

   Ans: **GBN:** Host A sent 9 segments in total. It initially sent segments 1, 2, 3, 4, 5 and later resent segments 2, 3, 4 and 5. Host B sent 8 ACKs. They are 4 ACKs with seq #1 and 4 ACKs with seq #2, 3, 4 and 5.

   SR: Host A sent 6 segment. It initially sent segments 1, 2, 3, 4, 5 and later resent segment 2. Host B sent 5 ACKs. They are 4 ACKs with seq #1, 3, 4, 5 and later 1 ACK with seq #2 (for resent segment).

5. [**KR, Chapter 3, R15**] Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 65; the second has sequence number 92.

   (a) How much data is in the first segment?
       Ans: $92 - 65 = 27$ bytes

   (b) Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number?
       Ans: 65. Note that TCP acknowledgement is cumulative and states the expected in-order sequence number.

6. [**KR, Chapter 3, P26**] Consider transferring an enormous file of L bytes from Host A to Host B. Assume an MSS of 512 bytes.

   (a) What is the maximum value of L such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field is 32 bits.
       Ans: TCP sequence number doesn't increase by one with each TCP segment. Rather, it increases by the number of bytes of data sent. Therefore the size of the MSS is irrelevant here—the maximum size file that can be sent from A to B without exhausting TCP sequence number is simply 2^{32} bytes.

   (b) For the L you obtain in (a), find how long it takes to transmit this file. Assume that a total of 64 bytes of transport, network, and data-link header are added to each packet before the resulting packet is sent out over a 155 Mbps link. Ignore flow control, congestion control and assume Host A can pump out all segments back to back and continuously.

Ans: Number of packets $= L/MSS = 2^{32}/512 = 8,388,608$
64 bytes of headers will be added to each packet. Therefore,
Total bytes sent $= 2^{32} + 8388608 \times 64 = 4,831,838,208$ bytes
Transmission delay $= 4,831,838,208 \times 8/(155 \times 10^6) \approx 249$ seconds

7. **Wireshark: TCP**

   Do the following:

   (a) Start up your web browser. Go to `http://gaia.cs.umass.edu/wiresharklabs/alice.txt` and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.

   (b) Next, go to `http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html`.

   (c) Use the Browse button to enter the full path name of the file you just downloaded. Do not yet click the "Upload alice.txt file" button yet.

   (d) Start up Wireshark and begin packet capture.

   (e) Returning to your browser, click the "Upload alice.txt file" button. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.

   (f) Stop the Wireshark capture.

   Answer the following questions:

   (a) What is the IP address and TCP port number used by the client computer (source) that is transferring the file to `gaia.cs.umass.edu`?
   Ans: select an HTTP message sent from your computer to `gaia.cs.umass.edu` and explore the details (src-ip & sec-port) of the TCP packet used to carry this HTTP message.

   (b) What is the IP address of `gaia.cs.umass.edu`? On what port number is it sending and receiving TCP segments for this connection?
   Ans: IP: select an HTTP message and explore the details of the TCP packet used to carry this HTTP message.
   Port: 80.