National University of Singapore
School of Computing

CS2105 **Tutorial 4** Semester 2 AY18/19

To students:

Due to time constraint, not all the questions may be discussed in class. Your tutor has the discretion to choose the questions to discuss (or you may request your tutor to discuss certain questions). Please do go through those left-over questions after class.

Please be reminded that our **Mid-term test** is on **11 Mar 2019, Mon 2:10-3:30pm** at **MPSH1**. The scope of this test is everything taught before recess week. Two past year question papers can be found in IVLE Files.

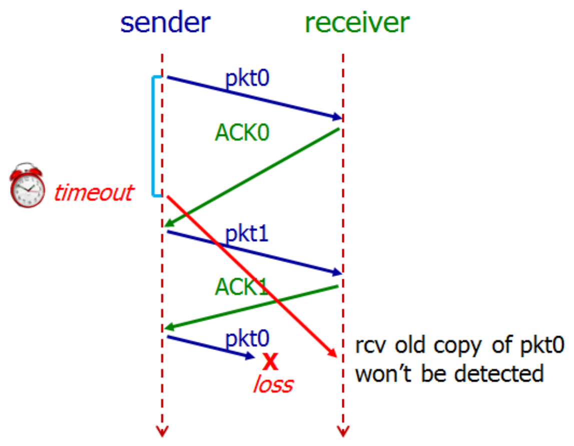
1. **[KR, Chapter 3, R6]** Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, how?

   Note: this is exactly what you are supposed to do in Assignment 2. ☺

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

IP Header	UDP Header	App Header fields + data + Trailer

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.



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 acknowledgement 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?

   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, TCP use large sequence number field (32-bit) to lower the chance a sequence number is to be reused. Secondly, a packet cannot "live" in the network forever. For example, IP protocol specifies TTL in packet header to ensure that datagrams do not circulate infinitely 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.

   GBN: Host A sends 9 segment. They are initially sent segments 1, 2, 3, 4, 5 and later resent segments 2, 3, 4 and 5. Host B sends 8 ACKs. They are 4 ACKs with seq # 1 and 4 ACKs with seq # 2, 3, 4 and 5.

   SR: Host A sends 6 segment. They are initially sent segments 1, 2, 3, 4, 5 and later resent segment 2. Host B sends 5 ACKs. They are 4 ACKs with seq # 1, 3, 4, 5 and 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?

      92 – 65 = 27 bytes

   b) Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

      65. Note that TCP acknowledgment 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.

   1. 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.

      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.

   2. 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.

      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 * 64 = 4,831,838,208 bytes
      Transmission delay = 4831838208 * 8 / (155 * 10^6) ≈ 249 seconds

7. Wireshark: TCP

   Do the following:

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

2. Next, go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.
3. Use the Browse button to enter the full path name on your computer containing Alice in Wonderland. Don't yet press the "Upload alice.txt file" button.
4. Startup Wireshark and begin packet capture.
5. Returning to your browser, press the "Upload alice.txt file" button. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
6. Stop Wireshark packet capture.

Answer the following questions:

1. 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 & src-port) of the TCP packet used to carry this HTTP message.
2. 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 explore the details of the TCP packet used to carry this HTTP message. port: 80.