CSEE 4119: Computer Networks — Due on Tuesday, March 11th, 2014

Assignment #3 – Transport Layer (with solutions)

P. Prakash, W. An, P. Nirantar, R. Yu, X. Zhu (TAs) A. Chaintreau (instructor),

How to read this assignment : Exercise levels are indicated as follows

($\rightarrowtail$) "elementary": the answer is not strictly speaking obvious, but it fits in a single sentence, and it is an immediate application of results covered in the lectures.

Use them as a checkpoint: it is strongly advised to go back to your notes if the answer to one of these questions does not come to you in a few minutes.

($\curvearrowright$) "intermediary": The answer to this question is not an immediate translation of results covered in class, it can be deduced from them with a reasonable effort.

Use them as a practice: how far are you from the answer? Do you still feel uncomfortable with some of the notions? which part could you complete quickly?

($\nrightarrow$) "tortuous": this question either requires an advanced notion, a proof that is long or inventive, or it is still open.

Use them as an inspiration: can you answer any of them? does it bring you to another problem that you can answer or study further? It is recommended to work on this question only AFTER you are done with the rest!

Exercise 1: Quick questions (12 pt)

1. ($\rightarrowtail$)Why should a server bind to a port number whereas it is not required for a client?

For the following questions, we will ask you how many sockets (i.e. a local door open on one machine to exchange information with the network interface) are open. Sockets that are communicating but sit on different machines will be related but should be counted separately.

2. ($\rightarrowtail$)Detail how many sockets are simultaneously open in every machine when 10 clients connect to a webserver using persistent HTTP to request 5 objects each, and the webserver answers those queries by sending web-objects.

3. ($\rightarrowtail$)Same question for 10 clients connect through 2 different applications to a local DNS server.

For the following two questions, we wish to compare protocols. You can either claim that the two protocols are exactly identical (in which case you should justified in your answer why this is the case), or you could claim they are different by providing a counterexample.

4. ($\curvearrowright$)How does rdt 2.1 or rdt 2.2 (the alternating 0-1 protocol with NAK or duplicate ACK, and time-out) compare with GO-BACK-N when the window is set to $W = 1$?

5. ($\curvearrowright$)Same question with selective repeat?

Exercise 2: Are sequence numbers needed? (12 pt, including 1pt for question 4)

Motivation: This exercise was part of the 2013 Midterm. Imagine you give your credit card number on the phone but your cell phone is noisy. It's annoying to give a sequence number to every digit (something like "the first digit is 0, the second digit is 7"). That's why many of us like to give the information as is and say "I'll repeat" in case something seemed wrong and we repeated a number (to avoid it is written twice). In this exercise, you will see if the same idea works between computers or whether it is in general confusing.

In order to detect duplicates while avoiding using sequence numbers, one may think of adding a different field in the header of any packet. Two alternatives are:

retransmission flagging: A unique bit is added to the header of the packet. This bit is a "0" if this is the first time this particular data is sent, or a "1" if this is not the first time it is sent.

retransmission counting: An integer value is added to the header of the packet, that indicates the number of previous transmissions of this data. This number will be "0" when this is the first time. Later, in future retransmissions it will be changed to "1", "2" etc.

You may conceptually think of it as a "sequence number" associated with all packets transmitting the same data. Note that we assume that the receiver, having no particular information on the packet it acknowledges, simply answer either with a "ACK" or a "NACK" at the reception of a packet. We also assume that there is no pipelining used.

1. ($\rightarrowtail$) Prove using a simple counter example that none of these schemes can correctly operate if packets can get reordered.

For the following three questions, we assume that packets are not reordered. However, they may be delayed or loss. The sender implements a time-out to avoid deadlock.

2. ($\curvearrowright$) Prove that **retransmission counting** does not correctly operate with packet losses. To do that, you will provide a counter-example that uses the minimum number of packets and packet loss.

For the following two questions, we assume that no packet loss occurs. However, packets can be corrupted. We assume that any corruption is detected using a checksum

3. ($\curvearrowright$) Prove that **transmission flagging** does not correctly operate even in this case?

4. ($\not\rightarrowtail$) Do you think that **retransmission counting** can correctly operate if no packets are lost?

Exercise 3: Designing a reliability protocol for two receivers (15 pt + 1pt for ($\not\rightarrow$))

This exercise is taken from 2012 Midterm. A host wishes to transmit message over a broadcast channel to two receivers. Unfortunately, losses may occur on this channel for any of the two receivers independently, so the sender will have to ensure reliability by receiving some signal from each receiver separately.

More precisely, consider a scenario in which a Host A wants to simultaneously send messages to Hosts B and C. A is connected to B and C via a broadcast channel: a packet sent by A (e.g., in a single unreliable_send() operation) is carried by the channel to both B and C.

We make the following assumption on the channel connecting A, B, and C:

- It can independently lose and corrupt messages from A to B and C (and so, for example, a message sent by A might be correctly received at B but not at C).
- It has a maximum bounded delay of D (i.e., if a message is sent by A, it will either be lost or arrive at B and/or C within D time units).
- any control message (e.g., an ACK or NAK) sent by B or C to A will only be received by A and not any other node, but like others they can be lost or corrupted.
- We do not use any pipelining in this channel.

1. ($\curvearrowright$) Design a stop-and-wait-like error-control protocol for reliably transferring a packet from A to B and C, such that A will not get new data from the upper layer until it knows that both B and C have correctly received the current packet. Give a FSM description for A and C (assuming the FSM for B is similar, if it is not similar give the FSM for B as well). Also, give a description of the packet format used.

2. ($\not\rightarrow$) If we assume that each channel ($A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow A$) may create a loss or a corruption independently from the others with probability p_e, and has the same propagation delay d_p. We assume that the packet has a transmission delay d_t and we neglect the transmission delay of the ACK, compute the utilization of this channel (the utilization corresponds to the amount of new useful information that could be sent from the source and received by unit of time, as compared with the capacity of the link).