COSC 650 Project Fall 2012

**Due Thursday December 6, 11:59 pm.**

This is a group project. All members should contribute. The items indicated below should be submitted by uploading to Blackboard on or before the due date. Please submit only one set of answers (files) for your group.

**1.** Determine how to send an HTTP Get request (to port 80) from a TCP Java socket client program to obtain two distinct files f1 and f2 that are stored on the same Web server. The client allows the user two options 1 and 2.

If option 1 is chosen,

* It sends Get requests to a Web server and retrieves the two files f1 and f2.
* The client selects the server and the files (the user does not choose the server or the files).
* The client displays the contents of each file received and the time it took between sending the request and receiving the complete file.
* The client should send the requests for f1 and f2 (without waiting till f1 is received). Whichever file retrieval is completed first, it is displayed first.
* The contents of the files should not be displayed in an interleaved manner. If f1 is large, f2 may be received first and it will be displayed first.
* After receiving and printing a file, the client prints the time that it took to receive the file.

If option 2 is chosen,

* The program works the same way except that the user is allowed to input the server name and the two files to be retrieved.
* The program should display example input so that the user knows the format in which the input is to be entered.

Upload the following to Blackboard.

a) The client code named groupnameTCPClient.java

b) A readme file (MS Word) named groupnameTCPReadme with instructions on how to compile/run the program

c) A file named groupnameTCPResult containing the output from a trial run to show what works. Keep it small.

**2.** Write a UDP Java socket program to do the following.

* The client sends a packet with the system time tc1, integer sequence number s (use 0, 1, 2, etc.), and a one-line message m to the server.
* The server sends an ack back to the client with the system time ts at the server when the message is received, the system time tc1, and the next sequence number expected (one more than the sequence number of the last message received).
* The server then prints the following: system time tc1, system time at the server ts, sequence number x, and message m
* When the client gets the ack, it prints: system time tc2 when the client gets the ack, the time tc1 the message was sent, the time ts when the server received the message, and the next sequence number expected.
  + If the ack is not received, within a certain timeout period (to be initially specified by the user), the client prints the message “timeout”.
  + It then doubles the timeout period and sends the message again.
  + It repeats this process 3 times until an ack is received (otherwise it quits).
  + Once an ack is received, the client resets the timeout to the original value and sends another message and so on.
  + If the server has printed a message, but the client resends the message due to a timeout, the server does not print the message again.

Submit the output at the client and the server for the following cases:

(i) Specify a time that is large enough so that a timeout does not occur.

(ii) Specify a time that is small enough so that a timeout occurs and the message in a) has to be re-sent at least once.

The timeout process should be managed so that the client can simultaneously start the “timer” and also receive acks. Similarly, the server should be able to receive incoming messages while it is processing a message.

Submit the following:

a) Code for the server and client named groupnameUDPClient (or UDPServer)

b) A readme file (MS Word) named groupnameUDPReadme with instructions on how to compile/run the program

c) A file named groupnameUDPResult containing the output for the above cases to show what works. Keep it small.

**3.** Capture a DHCPv6 packet using Wireshark.

For the DHCPv6 packet, make a table showing the value of the fields in the DHCP packet and briefly (but clearly) explaining the meaning (or significance) of that particular value.

Submit the following:

a) The actual Wireshark capture .pcap file that includes the packet you used (named groupnameWS)

b) The table you made saved in MS Word named groupnameTable.