Computer Networks

Fall 2017/18

Exercise 2

Submission by Friday **8-12-2017.** Submission is done by uploading your work to the course website (Moodle). >>> No late submissions will be accepted.

Problem 1.

Fill in a table like this, determining for each statement whether it is true or false. Do no supply explanations.

A	B	C	D	E

a. A user requests a Web page that consists of some text and three images, using HTTP/1.1. For this page, the client will send one request message and receive four response messages.
b. Two distinct Web pages (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.
c. With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.
d. The `Date:` header in the HTTP response message indicates when the object in the response was last modified.
e. HTTP response messages never have an empty message body.

Problem 2.

Suppose within your web browser you click on a link to obtain a web page. The IP for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that the first DNS server visited responds with the requested IP, and this visit is done in RTT_{dns} time.
Suppose the web page associated with the link contains a small html file, referring to 6 small images on the same server. Suppose querying and getting a response from the web server takes RTT_{web} time.
Assume zero transmission time of all objects.
How much time elapses from when the client clicks on the link until the client receives the complete web page with…

a. Non-persistent HTTP/1.0 with no parallel TCP connections?
b. Non-persistent HTTP/1.0 with the browser configured to a maximum of 4 parallel connections?

c. Persistent HTTP/1.1 without pipelining and with no parallel connections?
d. Persistent HTTP/1.1 with pipelining and with up to 4 parallel connections?
e. Suppose 2 of the 6 images are on two different servers. Suppose the servers' IP address is not cached on your local host, but again the first DNS server accessed with the appropriate query reveals their IP. Assume persistent HTTP/1.1 with pipelining and up to 8 parallel connections. Assume getting a reply from the other web servers also takes RTT_{web} time. How much time elapses before the client fully receives the web page?

Problem 3.

The goal of this question is to get you comfortable retrieving and reading some RFCs.

a. Take a look at RFC 5034.
   How should a server react when receiving an AUTH command after a previous AUTH command was accepted successfully?
   What other restriction is there on AUTH commands?
b. Take a look at RFC 1939.
   At which section is the TOP command defined?
   What are its arguments?
   What does it do?
   How should a server react on `TOP 1 10000` if there are 10 messages on the queue and they are all of size 500?
c. Take a look at RFC 5321 for SMTP. You might want to focus on sub-sections 2.3 and 3.7.
   What does MTA stand for? Explain its role in the message delivery in no more than 2 lines.
   Consider the following received spam email. Assuming only the originator of the spam email is malicious and all other hosts are honest, identify the malicious host that has generated this spam email.

```
From - Fri Nov 07 13:41:30 2008
Return-Path: <tennis5@pp33head.com>
Received: from barmail.cs.umass.edu (barmail.cs.umass.edu [128.119.240.3]) by
cs.umass.edu (8.13.1/8.12.6) for <hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:10
-0500
Received: from asusus-4b96 (localhost [127.0.0.1]) by barmail.cs.umass.edu
(Spam Firewall) for <hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:07 -0500 (EST)
Received: from asusus-4b96 ([58.88.21.177]) by barmail.cs.umass.edu for
<hg@cs.umass.edu>; Fri, 07 Nov 2008 13:27:07 -0500 (EST)
Received: from [58.88.21.177] by inbnd55.exchangeddd.com; Sat, 8 Nov 2008
01:27:07 +0700
From: "Jonny" <tennis5@pp33head.com>
To: <hg@cs.umass.edu>
Subject: How to secure your savings
```

Problem 4.

You can download the windows version of **ncat** from: http://nmap.org/ncat/ (Mac OS users: you might have nc).
Use **ncat** to perform the following:

1. Get **ncat** listening on port 1717 over **UDP** (this is the server side).
2. On a different or the same computer, run a client **ncat** to connect to the **ncat** server that is listening on port 1717 over **UDP** (127.0.0.1 is IP of the "same computer").
3. From the client **ncat** send the string "`<your name>: Easy come, easy go, will you let me go?`" – make sure that the string is being received by the server **ncat**.
4. From the server **ncat** send back the string "`CS staff: No, we will not let you go`" – make sure that the string is being received by the client **ncat**.
5. Repeat steps 1-4 over **TCP**.

Tip: use "ncat –help" to see the available command lines and/or read the readme.txt file.

A. Take screen captures of all **4** windows (for both UDP and TCP). The screen shots must contain the **ncat** commands used and the strings that have been sent and received. Make sure you state which screen shot is of the client and which is of the server.
B. Suppose you run the **ncat** client over UDP before you run **ncat** server over UDP (that is, replace the order of step 1 and step 2 above). What happens? Why?
C. Suppose you run the **ncat** client over TCP before you run **ncat** server over TCP. What happens? Why?
D. What happens if you use different ports numbers for the client and server side, which run over TCP?
E. Suppose you run the **ncat** server over TCP as before and the **ncat** client over UDP. What happens? Why?

Problem 5.

Write a server program which runs over TCP listening to port 2017. The server first prints onto the server's standard output the message "`Exercise 2. November 2017. <Your name here>`". The server accepts lines of input from a client and then prints the lines onto the server's standard output. (You can do this by modifying the TCP server code you've seen in recitation). Compile and execute your program.
On any other machine that contains a web browser, use it to access http://search.lores.eu/indexo.htm. **After** you do that, set the proxy server in the browser to the host that is running your server program. (Hint: Don't forget to configure the port number appropriately in your browser's settings). Using the bowser try to access again http://search.lores.eu/indexo.htm.
Your browser should now send its HTTP GET request message to your server, and your server should display the messages on its standard output. Use this platform to determine whether your browser generates conditional GET messages for objects that are locally cached. Write down your analysis.

Attach your server's code (do not attach compiled files) and a screen capture of the full output.

Problem 6.

Re-consider the algorithm for binary consensus in the case of byzantine failures exactly as it appears in slide 21 of the presentation.

a. Suppose $n = 21$ and $t = 8$. (Note that in this scenario it does not hold that $n \geq 4t + 1$). Is it possible that all proper processes will reach an agreement at the end, but that the value of the agreement will not be a valid one? Explain your answer.
b. Do not submit!: Suppose $n = 21$ and $t = 8$. (Note that in this scenario it does not hold that $n \geq 4t + 1$). Is it possible that the proper processes will not reach an agreement at the end? Explain your answer.
c. Suppose $n = 21$ and $t = 5$, but now we change the number of iterations from $t + 1$ to t. Is it possible that all proper processes will reach an agreement at the end, but that the value of the agreement will not be a valid one? Explain your answer.