CPEN 247
Software Assignment #1
Logistics: work in pairs

Deliverables:

- For Tasks 1 & 2, answer all questions, support your answers with screenshots and upload a .pdf file with your answers to Blackboard through the provided link for Assignment 1.
- The .java file that you will develop in Task 2 should be uploaded separately through the same Blackboard link used for submitting this assignment.

(30 points) Task 1 [Wireshark]:

1. Start up your web browser
2. Start up the Wireshark packet sniffer but don't yet begin packet capture. Enter http in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window
3. Wait a bit more than one minute then begin Wireshark packet capture
4. Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
5. Your browser should display the very simple, one-line HTML file
6. Stop Wireshark packet capture

Answer the following questions and ***support your answer with screenshots***:

a) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
b) What languages (if any) does your browser indicate that it can accept to the server?
c) What is the IP address of your computer? Of the gaia.cs.umass.edu server?
d) What is the status code returned from the server to your browser?
e) When was the HTML file that you are retrieving last modified at the server?
f) How many bytes of content are being returned to your browser?
g) By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

(70 points) Task 2 [UDP-based Ping Program]:

You are required to write the client ping program in Java – use of Eclipse Java editor is recommended. Your client will send a simple ping message to a server, receive a corresponding pong message from the server, and determine the delay between when the client sent the ping message and received the pong message. This delay is called the Round-Trip Time (RTT).

Your ping program is to send 10 ping messages to the target server over UDP.

- The content of each message is a string of characters written in the form:

`message = "Ping " + i + " " + now.toString();`
Here, `now` is an object of the Java class `Date` that captures the current date and time.
To create the `now` object you should first import the Date utility library from Java using:
```
import java.util.Date;
```
Then you can create the object as:
```
Date now = new Date();
```

- In order to send the 10 messages automatically, you will need to research how to realize a simple for loop or while loop on Java.

For each message, your client is to **determine** and **print** the RTT when the corresponding pong message is returned.

- Note that the server code is given to you in the file UDPPingServer.java
  In that code, if the server gets the message correctly, it will echo the message back to you.

- To compute the RTT of each sent message, use the Java system function `System.currentTimeMillis();` which captures the current time instant in milliseconds. You will need to record the time at the moment when the message is sent and when it is received and subtract the later from the former. Print it after it has been computed.

Because UDP is an unreliable protocol, a packet sent by the client or server may be *lost*. For this reason, the client cannot wait indefinitely for a reply to a ping message. You should have the client **wait** up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and **print** a "Packet-lost" message accordingly.

- The socket library allows you to set a timeout value in seconds. If your socket name is *`clientPing`*, then you can specify the timeout value through the command:
  `clientPing.setSoTimeout(1000);`
  Here, the 1000 is the number of milliseconds that pass before declaring a packet loss.
- To simulate message loss events, the server code (given to you) ignores the client message with probability 30%

- The timeout event at the client automatically creates an exception in the code. To handle this exception your procedure should follow this syntax in Java:

```
try{ // to catch the timeout exception
     clientPing.receive(response);
    // write here what should happen in the normal operation

} catch(IOException SoTimeout) {
   // write here what should happen in case of timeout
}
```

Close the socket after you are done.

It is recommended that you first study carefully the server code. You can then write your client code, liberally cutting and pasting lines from the server code. You can also use lines from the UDPClient code that we have covered in class.

You will need two hosts to test and verify your code. One host to run the client and another to run the server.

- You and your partner can use your computers here.

Recall:

1. You will need to include screenshots of your working client code both at the client and server stations
2. Upload the .java file of the client separately on Blackboard.