COMP416: Computer Networks
Project 2

Due: April 14, 11:59pm (Late submissions will not be accepted).

Submission of the project deliverables is via Blackboard.

This is an individual project. You are not allowed to share your codes with each other.

Transport Layer Protocols Comparison and Analysis with Wireshark

This project is about the **transport layer** of the network protocol stack.

The focus is on the **SSL, TCP and UDP protocols**. For this purpose, you are asked to modify the provided SSL client/server codes, implement TCP client/server codes for delay comparison as specified below, experiment with TCP and explore UDP features. You should use the **WireShark network protocol analyzer** tool to answer transport layer related questions.

WireShark is the world's foremost network protocol analyzer, and is the **de facto standard** across many industries and educational institutions. It can be downloaded freely at **http://www.wireshark.org/download.html**. WireShark allows users to trace the network activity by capturing all the packets that hit your network interface. It tags the information of each layer by parsing the given byte stream according to the corresponding protocol.

You should read this project document carefully before starting your tasks.

Part 1.A. SSL Implementation and Experiments

Figure 1 illustrates an overview of SSL protocol. Recall that, you are provided an SSL client/server code that performs echoe on top of an SSL socket. The corresponding SSL practical content codes and slides are available through the course web site.

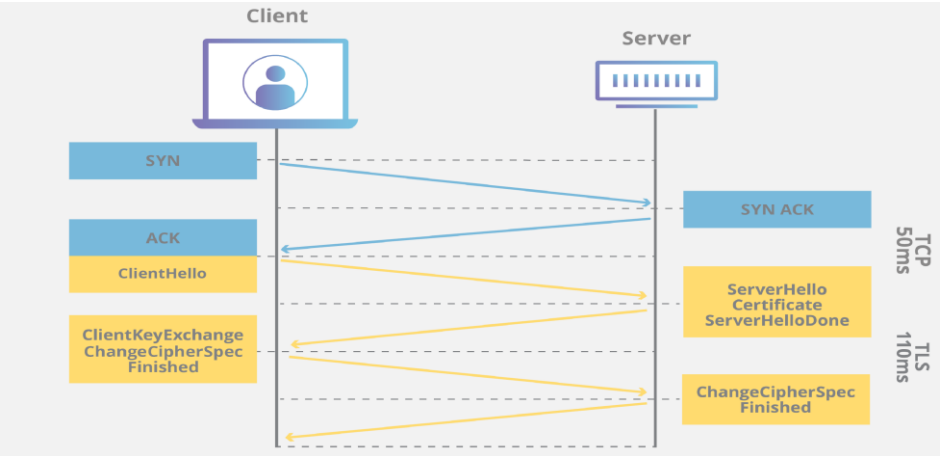

Figure 1. SSL Protocol Overview

As presented in the figure, the certificate is sent by the server to the user at the start of the session. The user adds this certificate to the local key store and uses it for authentication. The code is provided to add the certificate to the local key store, but the part where the server sends the certificate to the user is missing.

In this part, you are asked to modify the provided code as follows:

- First, the server side should listen to the port with your KUSIS ID number (look at the first question). If your ID number is larger than the possible maximum port number value, you can just divide it by 3 and indicate this in the answer of the first question.
- Second, your client should receive the digits of your KUSIS ID number one by one in separate messages with a **non-persistent manner** and should print the number it receives for every new connection. Then, it should print the final merged number.

Important Notes:

- Your modified SSL codes must be submitted along with your project report. In your project report, you should explain your answers and provide your WireShark outputs for each question, in order to get credit.
- On Windows operating systems, you might not be able to capture the Loopback interface, which is the traffic inside your operating system. When you run your server and client software in a single operating system, in order to capture incoming and outgoing packets, you need to capture the Loopback interface.

After running your code, answer the following questions:

1. How many TCP packets are transmitted in total while your KUSIS ID number is exchanged one by one with non-persistent connections?
2. How many cipher suites does your client support? Where can these details be found within the traffic you just monitored?
3. Does the client support any other version of the TLS/SSL? If yes, how many? If not, what may be the reason/s for this?
4. Locate the message with the TLS Protocol ‘Server Hello’. Can you identify the Cipher which the server will be using during this connection? Find the hex dump of the key and report it in your answer.

Part 1.B. SSL vs TCP: Delay measurements

In this part, you are asked to evaluate the performance of SSL and TCP using time delay between the client generating a request and receiving the answer as a metric. This part can be performed by using the sample TCP codes provided in the first practical content and SSL code provided for this project.

First, the client will send the character "1" to the server as a query and as a response the server will send your KUSIS ID + KUSIS Username + the date of the day (DDMMYYY) as a single string back to the client. Measure the time delay between sending the request and receiving the server response when communicating over a TCP socket vs an SSL socket.

Second, the client will send the character "2" to the server as a query and as a response the server will send your KUSIS ID + KUSIS Username + the date of the day (DDMMYYY) back to the client in a non-persistent fashion. Measure the time delay between the request and receiving the server final character response when communicating over a TCP socket vs an SSL socket.

5. Report both delays for 5 different executions and present the measurements as a single graph. Briefly describe the reasons for the results you have obtained.

Part 2. TCP Experiments

Before beginning the exploration of TCP, you need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You are asked to do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.htm then transfer the file to a Web server using the HTTP POST method. We are using the POST method rather than the GET method as we would like to transfer a large amount of data *from* your computer to another computer. Of course, you need to run Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer. Perform the following:

- Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-labs/alice.txt and retrieve an ASCII copy of *Alice in Wonderland*. Store this file somewhere on your computer.
- Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html. You should see a screen that looks like Figure 2:
- Use the *Browse button* in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). Don't yet press the *"Upload alice.txt file"* button.

▪ Now start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we will not need to select any options here).

▪ Returning to your browser, press the *"Upload alice.txt file"* button to upload the file to the gaia.cs.umass.edu.server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.

▪ Stop Wireshark packet capture.

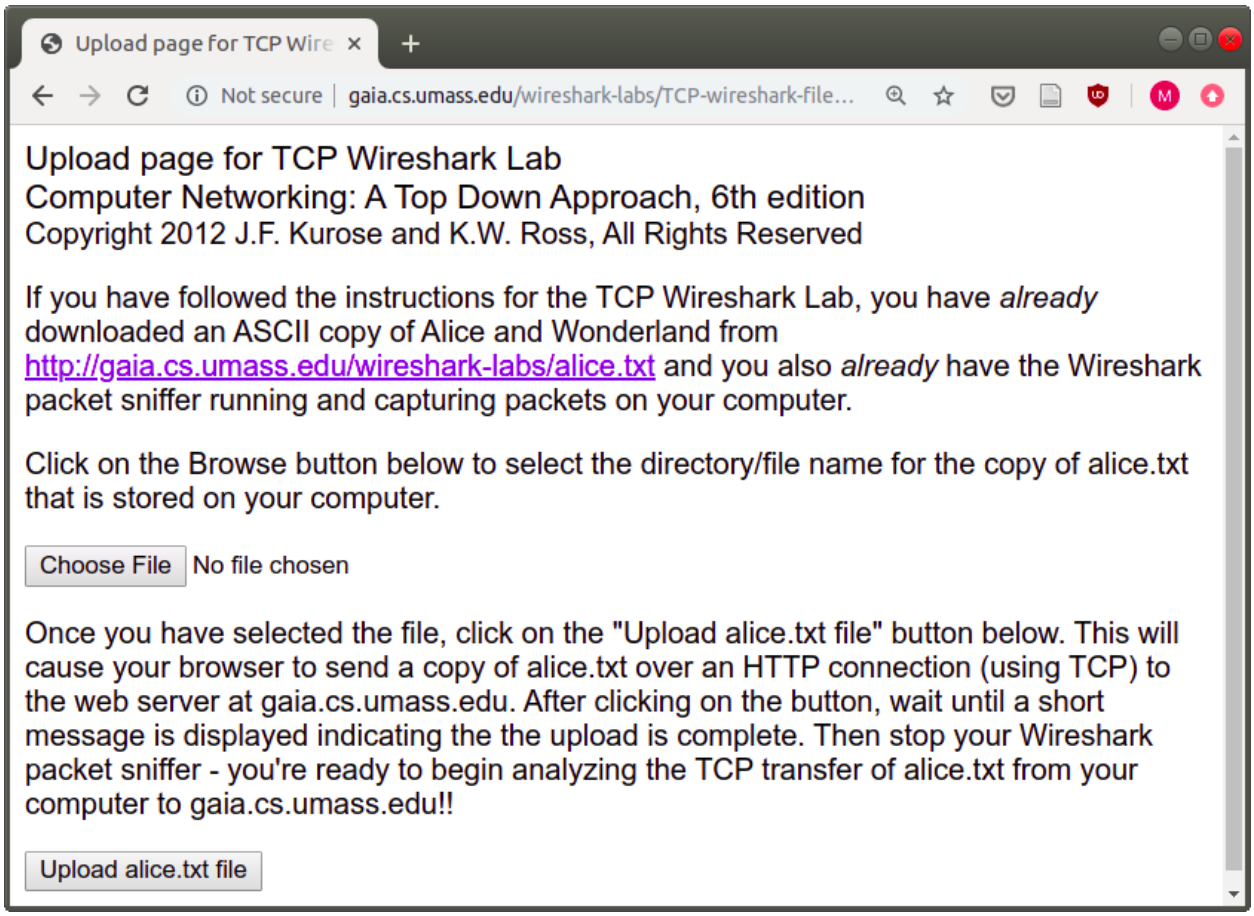

Figure 2. Text File upload Screen

Answer the following questions for the TCP segments:

6. What is the Acknowledgement Number and the 'relative ack number' in the TCP header? What does it signify?

7. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

8. What is the TTL value for the TCP packets originating at your end? What is the TTL value

for the packets received from the server? Can you confirm the number of nodes in between using another method?

9. Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the *EstimatedRTT* value (see Section 3.5.3 in the textbook) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation (Section 3.5.3 in the textbook) for all subsequent segments.

10. What does the stream index in the TCP header signify? Are the packets being transmitted during the experiment all belonging to the same stream index? What does a same or different stream index mean in the context of this experiment?

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: *Statistics->TCP Stream Graph- >Round Trip Time Graph.*

Part 3. UDP Experiments

In this part, you are assigned a **unique** URL to work with. The list is provided in file Project2_URL_List.pdf, and you must use the URL assigned you. You should provide the appropriate screenshots and work on the correct domain, in order to get credit. *nslookup* command works as an IP address resolver. When you provide a domain name as argument, it will return the IP address of that domain. Now, take the following steps provided and answer the questions accordingly.

▪ Start your Wireshark software and start capturing packets from the appropriate interface. ▪ Use nslookup command in order to resolve the IP address of the URL that is assigned to you.

▪ Stop packet capturing in the Wireshark and apply the appropriate **display filter**.

11. What is the source and destination socket address for the query packet? Under which header can this information be found?

12. Locate the header under which the name of the website you have requested the IP address for is mentioned?

13. Can you derive the local DNS server you connected work in iterative or recursive manner? Please provide a detailed explanation. Besides, explain the advantages and disadvantages of an iterative and recursive approach over each other.

14. How many replies were received in response to the DNS query? What was the type of the IP you have received and what does it mean? Identify another type and describe what that corresponds to.

15. What are types of DNS Records (name them in the report but you may be asked about their significance during the demo? How can you specify the type of DNS Record when using the 'nslookup' command? Share the results for using the nslookup with any '3' DNS Record Types.

Project Deliverables:

Important Note: You are expected to submit a project report, in PDF format, that documents and explains all the steps you have performed in order to achieve the assigned tasks of the project. A full grade report is one that clearly details and illustrates the execution of the project. Anyone who follows your report should be able to reproduce your performed tasks without effort. Use screenshots to illustrate the steps and provide clear and precise textual descriptions as well. All reports would be analyzed for plagiarism. Please be aware of the KU Statement on Academic Honesty.

The name of your project .zip file must be <surname>-<KUSIS-id>.zip
You should turn in a single .zip file including:

- Source codes: Containing the source codes of client and server for both SSL and TCP.
- Project.pdf file (Your report, should include answers and the corresponding Wireshark screenshots)
- Saved capture files from the Wireshark.

Figures in your report should be scaled to be visible and clear enough. All figures should have captions, should be numbered according to their order of appearance in the report, and should be referenced and described clearly in your text. All pages should be numbered, and have headers the same as your file naming criteria.

If you employ any (online) resources in this project, you must reference them in your report. There is no page limit for your report.

Good Luck!