Assignment 3: TCP Sockets

Deadline: February 3, 2025 EOD

SUBMISSION INSTRUCTIONS

Prepare a detailed observation and analysis report and the appropriate screenshots from the Wireshark and Terminal for listed questions with specific details asked in individual tasks. Submit your report in PDF format with <NAME>_<ROLL NO>_Report.pdf. The pcap file should be uploaded to a public Google drive folder and the link needs to be given within the code. Zip all these files, include the source code of the two programs into a single zip file <NAME>_<ROLL NO>.zip and submit on MS Teams.

NOTES:

1. You should mention your name (as per ERP), roll number and the Assignment number within the comment line at the beginning of each program. A sample header will look as follows.
   ==========
   Assignment 3 Submission
   Name: <Your_Name>
   Roll number: <Your_Roll_Number>
   Link of the pcap file: <Google_Drive_Link_of_the_pcap_File>
   ==========
2. The code should have proper documentation and indentation. Unreadable codes will not be evaluated.
3. The code should get executed to the lab machines. If we get a compilation error or runtime error during executing your code on the lab machines, appropriate marks will be deducted.
4. Any form of plagiarism will incur severe penalties. You should not copy the code from any sources. You may consult online sources or your friend, but at the end, you should type the program yourself. We may ask you to explain the code, and if you fail to do so, you'll be awarded zero marks.

Objective:

The objective of this assignment is to get familiar with TCP sockets using POSIX C programming and to analyze the behavior of the communication between client and server using **Wireshark**. The target is to establish communication between two processes (client and server) using a TCP socket and observe the network traffic using Wireshark.

Problem Statement:

Consider a simple Substitution Cipher encryption scheme in which each letter of a plaintext file is replaced by a fixed letter in the same set as described by the key. Here is the sample key as-

Plain Text Letter	**Mapped Cipher Text Letter**	**Plain Text Letter**	**Mapped Cipher Text Letter**
A	D	N	G
B	E	O	H
C	F	P	Q
D	P	Q	S
E	R	R	I
F	T	S	U
G	V	T	J
H	W	U	X
I	L	V	K
J	M	W	B
K	Z	X	C
L	A	Y	N
M	Y	Z	O

The same mapping will be taken for lower case letters as well.

In this assignment, a client will send a 'plaintext' file and a key like 'DEFPRTVWLMZAYGHQSIUJXKBCNO' (as per the above sample key: 'A' is mapped to 'D', 'B' is mapped to 'E', etc.) to an encryption server using a TCP socket. The server will encrypt the file using the Substitution Cipher scheme and send it back to the client using the same TCP socket. The file is a text file of arbitrary size (> 0 bytes). You can assume (no need to check) that the file will contain only alphabets (uppercase and lowercase), space, and new line characters. Only letters will be encrypted and space or new line is left as is and not encrypted.

Here is the content of a sample file:

```
HELLO
ABC
CSE
IIT Kharagpur
```

The encrypted version of the above file using the example key ‘DEFPRTVWLMZAYGHQSIUJXKBCNO’ is as follows.

```
WRAAH
DEF
FUR
LLJ Zwdidvqxi
```

The transfer of the contents of the file works using a communication protocol as follows.

1. The client establishes a connection to the server.
2. The client reads the filename to be sent from the user (keyboard). If the file does not exist, an error message NOTFOUND <FILENAME> is printed and the user is asked to enter an available filename again.
3. If the entered filename exists, then the user is asked to enter the value of the key k. If the key contains less than 26 characters, then the client throws an error and asks for the key again.
4. The client first sends the key k to the server, and then sends the file content, in that order. Since the file can be arbitrarily long, the client cannot send the entire file in a single send() call, and sends the file in small chunks using multiple send() calls until the entire file is transferred. The chunk size used by the client is not known to the server.
5. The server reads the file contents sent by the client and copies it into a temporary file in the current directory. The file should be named as <IP.port of client>.txt where IP is the IP address of the client and port is the port of the client. For example, if the client has IP address 192.168.0.20 and port 5000, the file should be named 192.168.0.20.5000.txt.
6. After the complete file is received, the server encrypts the file and stores the encrypted file in another file named X.enc, where X is the name of the original file. For example, for the above example, the encrypted file will be stored in a file named 192.168.0.20.5000.txt.enc.
7. The server sends the file to the client, once again in chunks. The chunk size used by the server is not known to the client.
8. The client stores the file in the same directory as the original file with the filename Y.enc, where Y is the original filename of the file. For example, if the filename was dummy.txt, the encrypted file is named dummy.txt.enc.
9. The client then prints a message stating the file is encrypted, giving the filenames of the original and the encrypted file.
10. The Client will prompt the user to confirm whether any additional files need to be encrypted. If the message received from the user is: **“No”**, The Client will immediately initiate a connection closure procedure to gracefully close the

connection. Otherwise, the Client will repeat Steps 2 to 9 to continue encrypting additional files.

Part-1: Socket Programming

Your task will be to write the two programs - one program corresponding to the server process and another program corresponding to the client process.

1. doencfileserver.c (server program): The server listens for a multiple file encryption requests from the client, encrypts them, and sends them back to the client.

   Marks:20

2. retrieveencfileclient.c (client program): The client receives the encrypted files from the server and prints a message stating the files are encrypted, giving the filenames of the original and the encrypted files.

   Marks:15

Note:-

I. You need to ensure the following in your code:
   a) No buffer of size > 100 can be used in either client or server.
   b) The client cannot make more than one pass over the file, nor can it find the size of the file in any other way. Same goes for the server. More strictly, you cannot use the size of thefile anywhere in your code.

Part-2: Wireshark Analysis

Use **Wireshark** to capture and analyze the communication between the client and server. Perform the following:

1. What are the source and destination IP addresses and ports? Share the screenshots to justify your answer. *Marks:2*
2. Inspect the Three-way handshaking procedure and capture all packets exchanged for it. Attach the necessary screenshots to demonstrate it. *Marks:4*
3. Inspect the connection closure procedure and capture all packets exchanged for it. Attach the necessary screenshots to demonstrate it. *Marks:4*

4. Inspect the traffics and count the number of packets exchanged for the transfer of a file(related to data only) between client and server. Plot a graph ‘file size vs the number of packets’ clearly based on your observation. *Marks:2*
5. Measure the total time taken for the file transfer , its encryption and send back it from server to the client. Plot a graph ‘file size vs time’ clearly based on your observation and also attach the necessary screenshots. *Marks:3*
6. Calculate the average size packet exchanged during the data communication? Take reference of the plotted graph in the above question. *Marks:2*