CS 39006: Networks Lab
Assignment 1: Using Wireshark, UDP sockets
Submission Deadline: January 12, 2024, 2:00 PM

PART A: Exploring Packet Sniffer and Packet Analyzer

The objective of this part of the assignment is to understand the network packet structures and different fields present at the packet headers corresponding to different layers of the protocol stack. For this assignment, you need to install Wireshark on your machine and keep your machine connected with the Internet.

To start with, execute the following steps.

1) Ensure that no browsing activity is going on in your machine. Close all the browser tabs.
2) Download and run Wireshark.
3) Start capturing packets over your default (active) network interface.
4) Open a browser tab and open the website http://iitkgp.ac.in/.
5) Wait for 1 minutes.
6) Now on the same browser tab, open the website https://www.cornell.edu/.
7) Close the browser tabs.
8) Stop packet capture.
9) Save the pcap file from Wireshark.

Now answer the following questions by analyzing the packet traces. Do not use any other tools other than wireshark, use its features to answer the questions to learn them better.

1. How many packets do you see for the following protocols?
    a. TCP and UDP together
    b. IPv4 and IPv6?
2. What is the total amount of data being received for the following two cases?
    a. When you access http://iitkgp.ac.in
    b. When you access https://www.cornell.edu
3. How many DNS packets have you observed in total?
    a. Create a <Domain Name, IP> table by exploring the queries and the answers in those DNS packets. The Domain Name will be the domain for which you see a query, and the IP address will be the address that is being returned against the corresponding query.
    b. Can you find out the IP of the DNS servers by exploring the DNS packets?
4. Answer the following when you access the site http://iitkgp.ac.in.

a. How many HTTP GET requests do you observe? List down the GET requests.
b. For each of the HTTP GET requests you see above, find out (ii) the total number of TCP segments being received, and (ii) the total amount of data being received in the corresponding HTTP Response message.

Submission Instructions:

Upload the pcap file that you have used in a password-less shareable Google drive folder (do not upload the file directly in moodle). Create a doc file named <roll_number>_Assignment1_A.doc (replace <roll_number> with your roll number) with your answers. The header of the doc file should be as follows.

==========
Assignment 1 (Part A) Submission
Name: <Your_Name>
Roll number: <Your_Roll_Number>
Link of the pcap file: <Google_Drive_Link_of_the_pcap_File>
==========

You should put the header at the beginning, and then type the answers to the above questions. Submit the doc file through Moodle by the deadline.

PART B: Simple Datagram Socket using POSIX C

The objective of this assignment is to get familiar with datagram sockets using POSIX C programming. The target is to establish a communication between two computers (processes) using a datagram socket. A datagram socket uses a simple communication paradigm to transfer short messages between two computers (processes) without ensuring any reliability.

Your task will be to write two programs - one program corresponding to the server process and another program corresponding to the client process. The client process requests for the content of a file (by providing the file name) and the server process sends the contents of that file to the client.

For simplicity, we assume that the file is a simple text file that contains a set of words, with the first work being HELLO and the last word being END. We assume that HELLO and END are two special keywords that do not come anywhere except at the first line (HELLO) and at the last line (END). The content of a sample file looks as follows.

```
HELLO
SOCKET PROGRAMMING
```

```
COMPUTER
NETWORK
TCP UDP
END
```

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

1. The client first sends the file name to the server (assume the name has les than 100 characters).
2. The server looks for the file in the local directory, if the file is not there it sends back with a message NOTFOUND <FILENAME>, where <FILENAME> is the name of the file requested by the client. By receiving this message, the client prints an error message “File <FILENAME> Not Found” and exits.
3. If the file is present, the server reads the first line of the file, which contains HELLO, and sends this message to the client.
4. After receiving HELLO, the client creates a local file (a different file name from the requested one) and sends a message WORD1 to the server. This message indicates that the client is requesting for the first word.
5. After receiving the message WORD1, the server sends the first word (after HELLO) to the client. The client writes this word to the local file after receiving it and sends the message WORD2 to request for the next word. This procedure continues for each word.
6. On receiving WORDi, the server sends the i-th word to the client. This process continues until the client receives the keyword END.
7. Once the client receives the keyword END, it closes the local file after writing the last word to the file.

Is this a good file transfer protocol?

Submission Instruction:

You should write two C programs - wordserver.c (contains the server program) and wordclient.c (contains the client program). Keep these two files in a single compressed folder (zip or tar.gz) having the name <roll number>_Assignment1_B.zip or <roll number>_Assignment1_B.tar.gz. Upload this compressed folder at the Moodle course page by the deadline.