COMPUTER NETWORKS (CS F303)
Second Semester 2017-2018
LAB-SHEET – 2

- **The motto of this lab session is learning by doing and thinking.**
- **Do NOT consult a book or a website or a senior – if you do so, you are not learning :(**
- **However, you are encouraged to consult/discuss/argue with other students in your class.**
- **Some of the material is taken up from TCP/IP Illustrated, Volume 1, The Protocols, by W. Richard Stevens**

This lab is divided into three parts.
In part one, we will continue doing the experiments with Wireshark to analyze and understand various application layers' protocols. In this lab our focus would be on another very important application layer protocol called File Transfer Protocol.

In the second part of the lab, we will try to understand the working principles of domain name system with the help of Wireshark capture and useful networking command called "nslookup".

Finally, in the third part, we will learn another useful networking tool called traceroute, which is very much required by any networking professional to manage the network and also to troubleshoot the problems during the time when the network is down or some site is not accessible.

Part 1: Understanding the functioning of File Transfer Protocol (FTP) using Wireshark

FTP: File Transfer Protocol
FTP is a commonly used application over the Internet for file transfer. The file transfer provided by FTP copies a complete file from one system to another system. To use FTP, we need an account to login on the FTP server.

FTP differs from the other applications such as HTTP and DNS that we did in the previous lab in the sense that it uses two TCP connections to transfer a file. The ***control connection*** is established in the normal client-server fashion. The server listens on the well-known port for FTP (21) and waits for a client connection. The client initiates a connection to TCP port 21 of the server to establish the control connection. The control connection stays up for the entire time that the client communicates with this server. This connection is used for commands from the client to the server and for the server's replies. A ***data connection*** is created each time a file is transferred between the client and server. Figure 1, shows the arrangement of the client and server and the two connections between them.

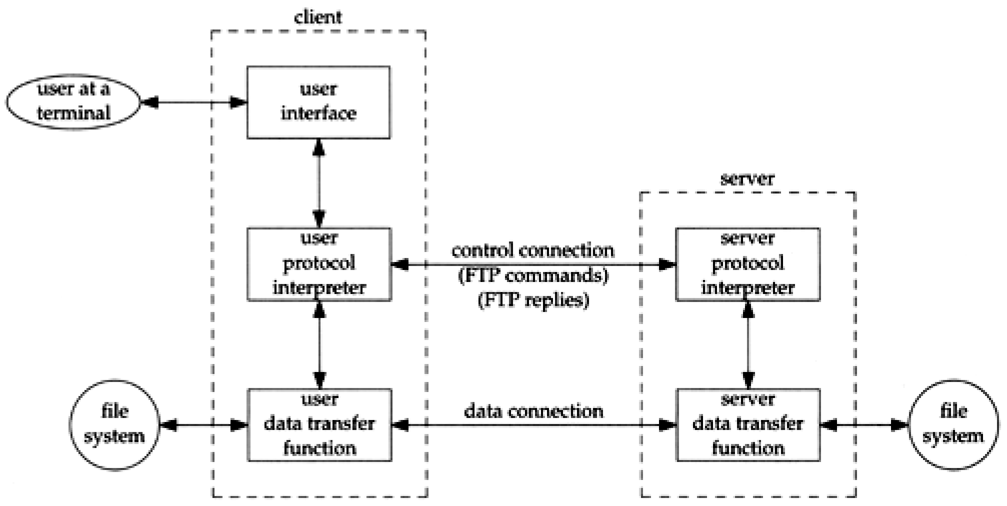

Fig. 1

FTP Commands

The commands and replies sent across the control connection between the client and server are in NVT ASCII (7 bit ASCII). This requires a CR (carriage return), LF (line feed) pair at the end of each line (i.e., each command or each reply). The commands are 3 or 4 bytes of uppercase ASCII characters, some with optional arguments. More than 30 different FTP commands can be sent by the client to the server. Table below shows some of the commonly used commands.

Command	Description
ABOR	abort previous FTP command and any data transfer
LIST *filelist*	list files or directories
PASS *password*	password on server
PORT *n1*,*n2*,*n3*,*n4*,*n5*,*n6*	client IP address *(nl.n2.n3.n4)* and port (*n5* x 256 + *n6*)
QUIT	logoff from server
RETR *filename*	retrieve (get) a file
STOP *filename*	store (put) a file

You will find that sometimes there is a one-to-one correspondence between what the interactive user types and the FTP command sent across the control connection, but for some operations a single user command (**what you type in the terminal**) results in multiple FTP commands across the control connection.

The FTP server sends file listings back across the data connection, rather than as multiline replies across the control connection. Unlike control connection which stays up for the duration of the client-server connection, the data connection can come and go, as required. So, the question is, how are the port numbers chosen for the data connection, and who does the active open and passive open?

The usual procedure is as follows:

1. The creation of the data connection is under control of the client, because it's the client that issues the command that requires the data connection (get a file, put a file, or list a directory).
2. The client normally chooses an **ephemeral port number** on the client host for its end of the data connection. The client issues a passive open from this port or starts listening on this port
3. The client sends this port number to the server across the control connection using the **PORT command**.
4. The server receives the port number on the control connection, and issues an active open (try to connect) to that port on the client host. The server's end of the data connection always uses port 20.

Lab Exercise-1: An Interactive FTP Session

1. To start an FTP interactive session type "ftp" from a DOS Command window.

C:\> ftp

The DOS prompt should be replaced with the FTP prompt. The FTP program is now running on the local system. A connection (or session) to a remote system has not been established.

2. The help command or ? (question mark) may be executed without being attached to a remote system and will do a print (usually to the screen) of the FTP commands. The following is an example of an FTP Command to display the FTP Help information.

ftp> help

3. The following FTP Command will perform the FTP OPEN (make the connection) and display the following messages.

ftp> open **172.22.1.129**
Username: **csis**, Password: **csis**

4. The following FTP Command will copy a file from the local system to the remote system and display the information.

ftp> put C:\Users\Dell\Documents\readme.txt (This file could be anything from your local computer)

ftp> quit

Lab Exercise-2: Understanding FTP protocol details using Wireshark

Identify TCP Header Fields and Operation Using a Wireshark FTP Session Capture

Step 1: Start a Wireshark capture.

a) Close all unnecessary network traffic, such as the web browser, to limit the amount of traffic during the Wireshark capture.
b) Start the Wireshark capture.

Step 2: Log on to the FTP server

a) From the command prompt, enter **ftp 127.22.1.129**
b) Enter Username: **csis**
c) Enter Password: **csis**

Step 3: Locate and download the *readme.txt* file from the server.

Step 4: ftp quit

Step 5: Stop the Wireshark capture.

Step 6: View the Wireshark Main Window.

Wireshark would have captured many packets during the FTP session. To limit the amount of data for analysis, type **tcp and ip.addr = = 172.22.1.129** in the Filter: entry area and click Apply. The IP address, **172.22.1.129**, is the address for our FTP server.

Analyze the TCP fields.

After the TCP filter has been applied, the first three frames in the packet list pane (top section) displays the transport layer protocol TCP creating a reliable session. The sequence of [SYN], [SYN, ACK], and [ACK] illustrates the three-way handshake. (Let us not get into details of SYN, SYN, ACK…., we will learn about them in due course of time!!!)

TCP is routinely used during a session to control datagram delivery, verify datagram arrival, and manage window size. For each data exchange between the FTP client and FTP server, a new TCP session is started. At the conclusion of the data transfer, the TCP session is closed. Finally, when the FTP session is finished, TCP performs an orderly shutdown and termination.

In Wireshark, detailed TCP information is available in the packet details pane (middle section).

Highlight the first TCP datagram from the host computer, and expand the TCP record.

Lab Exercise-3: Understand the working of FTP

Answer the following questions.

1. Did you see an OPTS request going from client to server? What is the purpose of this command?
2. Locate the PORT command in the Wireshark trace and try to correlate it with the command entered by you in the ftp terminal. Analyze the parameters of PORT command.
3. How the server is able to send PORT command successful, even when the three-way handshake required to complete the TCP connection that got initiated due to PORT command has not been completed.
4. The FTP server sends file listings back across the data connection, rather than as multiline replies across the control connection. Try to verify this fact by listing the remote directory using "dir" command and analyzing the Wireshark packet capture.

Part 2: Understanding the functioning of Domain Name System DNS Protocol using Wireshark and *nslookup* command.

Domain Name System

DNS i.e. Domain Name System is the method that the Internet uses to attach human-usable domain names to the machine-usable IP addresses that are almost never seen by the users. It is a hierarchical, decentralized naming system which is distributed in nature. Although TCP/IP uses IP addresses to locate and connect to hosts (computers and other TCP/IP network devices), users typically prefer to use friendly names. The Domain Name System (DNS), as defined in RFCs 1034 and 1035, is used on the Internet to provide a standard naming convention for locating IP-based computers. In early days, Network administrators entered names and IP addresses into Hosts file, and computers used the file for name resolution. Though it was effective but was not scalable. DNS replaced the Hosts file with a distributed database that implements a hierarchical naming system. This naming system allows for growth on the Internet and the creation of names that are unique throughout the Internet and private TCP/IP-based.

To understand the process of IP address resolution in DNS with the help of Wireshark lets perform the following experiment.

Lab Exercise-4:

Step 1: Start a Wireshark capture.

a) Close all unnecessary network traffic, such as the web browser, to limit the amount traffic during the Wireshark capture.
b) Start the Wireshark capture.

Step 2: Run the following command from the command prompt

```
nslookup  bits-pilani.ac.in
```

Step 3: Stop the Wireshark capture.

Step 4: View the Wireshark Main Window.

Try to answer the following questions based on observations of the traffic captured by Wireshark:

1. Locate the DNS query and response messages. Are they sent over UDP or TCP?
2. To what IP address is the DNS query message sent?
3. What is the destination port for the DNS query message?
4. What is the source port of DNS response message?
5. Examine the DNS response message.

Repeat the same experiment by running the following command at step 2
nslookup -type=NS bits-pilani.ac.in and try to answer the same questions.

Lab Exercise-5:

Run the following commands from the command prompt. We have provided here the corresponding output for each command. The output in your terminal may not be exactly same as the output provided here. Understand and describe the meaning/purpose of the commands and their corresponding output. (e.g., Different DNS records, i.e. A, NS, MX, CNAME and their purpose.)

Command	Output
C:\Users\Dell>nslookup bits-pilani.ac.in a0.in.afilias-nst.info	Server: UnKnown Address: 199.7.87.1 Name: bits-pilani.ac.in Served by: ns1.bits-pilani.ac.in 202.78.175.200
C:\Users\Dell>nslookup google.com	Server: UnKnown Address: 172.24.2.71 Non-authoritative answer: Name: google.com Addresses: 2404:6800:4009:807::200e 216.58.199.174
C:\Users\Dell>nslookup -type=NS google.com	Server: UnKnown Address: 172.24.2.71 Non-authoritative answer: google.com nameserver = ns2.google.com google.com nameserver = ns3.google.com google.com nameserver = ns4.google.com google.com nameserver = ns1.google.com ns3.google.com internet address = 216.239.36.10
C:\Users\Dell>nslookup google.com ns2.google.com	Server: ns2.google.com Address: 216.239.34.10 Name: google.com Addresses: 2404:6800:4009:807::200e 216.58.199.174
C:\Users\Dell>nslookup 8.8.8.8	Server: UnKnown Address: 172.24.2.71 Name: google-public-dns-a.google.com Address: 8.8.8.8
C:\Users\Dell>nslookup bits-pilani.ac.in google-public-dns-a.google.com	Server: google-public-dns-a.google.com Address: 8.8.8.8 Non-authoritative answer: Name: bits-pilani.ac.in Address: 202.78.175.227

Part 3: Traceroute Program- A handy debugging tool for network administrators

The Traceroute program, written by Van. It lets us see the route that IP datagrams follow from one host to another.

Although there are no guarantees that two consecutive IP datagrams from the same source to the same destination follow the same route, most of the time they do.

Traceroute uses ICMP (*don't worry if you do not understand the ICMP protocols at this point in time*) and the TTL field in the IP header. The TTL field (time-to-live) is an 8-bit field that the sender initializes to some value. the sender initializes to some value. Each router that handles the packet (datagram) is required to decrement the TTL by either one. The purpose of the TTL field is to prevent datagrams from ending up in infinite loops, which can occur during routing transients.

When a router gets an IP datagram whose TTL is either 0 or 1 it must not forward the datagram. (A destination host that receives a datagram like this can deliver it to the application, since the datagram does not have to be routed. Normally, however, no system should receive a datagram with a TTL of 0.) Instead the router throws away the datagram and sends back to the originating host an ICMP "time exceeded" message. **The key to Traceroute is that the IP datagram containing this ICMP message has the router's IP address as the source address.**

We can now guess the operation of Traceroute. It sends an IP datagram with a TTL of 1 to the destination host. The first router to handle the datagram decrements the TTL, discards the datagram, and sends back the ICMP time exceeded. This identifies the first router in the path. Traceroute then sends a datagram with a TTL of 2, and we find the IP address of the second router. This continues until the datagram reaches the destination host. But even though the arriving IP datagram has a TTL of 1, the destination host won't throw it away and generate the ICMP time exceeded, since the datagram has reached its final destination. **How can we determine when we've reached the destination?**

Traceroute sends UDP datagrams to the destination host, but it chooses the destination UDP port number to be an unlikely value (larger than 30,000), making it improbable that an application at the destination is using that port. This causes the destination host's UDP module to generate an ICMP "port unreachable" error when the datagram arrives. All Traceroute needs to do is differentiate between the received ICMP messages-time exceeded versus port unreachable-to know when it's done.

We're now ready to run traceroute and see the output. To perform the experiment, We'll use a **Traceroute** available online at the website http://traceroute.nmonitoring.com/

Following is the output we got when, we performed the traceroute for "**google.com**". You can directory write the domain name (like google.com) or the IP address of that domain to run the Traceroute program.

```
traceroute to 172.217.23.238 (172.217.23.238), 30 hops max, 60 byte packets
 1  praha-4d-c1-vl55.masterinter.net (77.93.199.253)  3.650 ms  3.718 ms  3.838 ms
 2  vl1387.cr3.r1-8.dc1.4d.prg.masterinter.net (83.167.254.150)  0.148 ms  0.357 ms  0.366 ms
 3  72.14.214.168 (72.14.214.168)  0.387 ms  0.376 ms  0.375 ms
 4  108.170.245.49 (108.170.245.49)  0.285 ms  0.295 ms 108.170.245.33 (108.170.245.33)  0.274 ms
 5  108.170.238.155 (108.170.238.155)  0.365 ms 108.170.238.157 (108.170.238.157)  0.337 ms
108.170.238.155 (108.170.238.155)  0.313 ms
 6  prg03s06-in-f14.1e100.net (172.217.23.238)  0.192 ms  0.185 ms  0.235 ms
```

Lab Exercise-6: Once you run the Traceroute from your computer, the output received by you may not be exactly same as above. (Note: It is suggested that you try it yourself.)
Now try to answer the following.

1. Explain the meaning of first line of the output.
2. The next two lines in the output begin with the TTL, followed by the name of the host or router its IP address and three different time values. What these time values signifies
3. How we can calculate the per hop time value?

Repeat the above experiment for the BITS Pilani web site and **iitd.ac.in** and observe the output

Lab Exercise-7:

Answer the following:

1. Did you observe the character * in few lines of the output for any of the trace route? If yes, then what does it mean?
2. Did you see the last hop in your output as the destination you are looking for? If no, then what could be the reason for this?

*****