# ICNW-532C
# Assignment 1(Venkat Sir)

**Name : Nishit Gupta**
**Roll No : IIT2014502**
**Course : B.Tech (IT)**
**Section : 'B'**

---

**Solutions:**

1. **HTTP 1.1 has a required Host header** by spec. whereas HTTP 1.0 does not officially require a Host header it is optional.
   HTTP 1.1 also allows to have persistent connections which means that you can have more than one request/response on the same HTTP connection
   whereas
   In HTTP 1.0 you had to open a **new connection for each request/response pair**. And after each response the connection would be closed.

   eg: Suppose two sites www.abcdefgh.com and www.xyzbag.com point to same IP then extra info (Host header) can tell the client/server machine where it can direct the machine browser.

2. Folder named "Que2" with screenshots of how message is sent from PackETH and captures using Wireshark.
   Step 1: Open Packeth, Write the Source IP and Destination IP in their respective fields.
   Meanwhile, open Wireshark and start capturing packets on the network.
   Step 2: Choose TCP and write the source and destination port
   Step 3: Write message in the Message Box and choose payload checkbox.
   Step 4: Choose the interface
   Step 5: Send Message
   Step 6: Apply TCP Filter and check if message has been captured or not.

3. Folder named "Que3" with screenshots of how message is sent from PackETH and captures using Wireshark.
   Step 1: Open Packeth, Write the Source IP and Destination IP in their respective fields.
   Meanwhile, open Wireshark and start capturing packets on the network.
   Step 2: Choose UDP and write the source and destination port
   Step 3: Write message in the Message Box and choose payload checkbox.
   Step 4: Choose the interface
   Step 5: Send Message
   Step 6: Apply UDP Filter and check if message has been captured or not.

4. **TCP**
   a. Open 2 Terminal Tabs (or you can choose two diiferent systems)
   b. For Server Side:
   Type: nc -l #port_no# > #new_filename#
   For Client Side:
   Type: nc #server_ip# -port_no- < #filename#

**UDP**

a. Open 2 Terminal Tabs (or you can choose two diiferent systems)
b. For Server Side:
Type: nc -u -l #port_no# > #new_filename#
For Client Side:
Type: nc -u #server_ip# -port_no- < #filename#

Folder Que4 contains the screenshot of file shared using TCP Protocol.

5. A server socket listens on a single port. All established client connections on that server are a ssociated with that same listening port on the server side of the connection. An established connection is uniquely identified by the combination of client-side and server-side IP/Port pairs. Multiple connections on the same server can share the same server-side IP/Port pair as long as they are associated with different client-side IP/Port pairs, and the server would be able to handle as many clients as available system resources allow it to.

On the client-side, it is common practice for new outbound connections to use a random client-side port, in which case it is possible to run out of available ports if you make a lot of connections in a short amount of time.

a-> On a server, a process is listening on a port. Once it gets a connection, it hands it off to **another thread**. The communication never hogs the listening port.

b-> Connections are uniquely identified by the OS by the following **5-tuple: (local-IP, local-port, remote-IP, remote-port, protocol)**. If any element in the tuple is different, then this is a completely independent connection.

c-> When a client connects to a server, it picks a random, unused high-order source port. This way, a single client can have up to ~64k connections to the server for the same destination port.

6. TCP : The absolute limitation -> **64K (65535 bytes).**
TCP deals with segments instead of packets.
Each TCP segment has a sequence number which is contained inside a TCP header.
The actual data sent in a TCP segment is variable.
However it can be calculated by function "**tcp_maxseg**" function in socket.h .

UDP : The correct maximum UDP message size is 65507 bytes, as determined by the following formula:
**0xffff - (sizeof(IP Header) + sizeof(UDP Header)) = 65535-(20+8) = 65507**