Week #1

Study and understand the basic networking tools - Wireshark, Tcpdump, Ping, Traceroute and Netcat.

Learn and Understand Network Tools

1. Wireshark

- Perform and analyze Ping PDU capture
- Examine HTTP packet capture
- Analyze HTTP packet capture using filter

2. Netcat

- Establish communication between client and server
- Transfer files

3. Tcpdump

- Capture packets

4. Ping

- Test the connectivity between 2 systems

5. Traceroute

- Perform traceroute checks

6. Nmap

- Explore an entire network

IMPORTANT INSTRUCTIONS:

- This manual is written for Ubuntu Linux OS only. You can also execute these experiments on VirtualBox or VMWare platform.
- For few tasks, you may need to create 2 VMs for experimental setup.
- Perform **sudo apt-get update** before installing any tool or utility.
- Install any tool or utility using the command **sudo apt-get install name_of_the_tool**
- Take screenshots wherever necessary and upload it to Edmodo as a single PDF file. (Refer general instructions for submission requirements).
- Instructors will give information, to define an IP address for your machine (e.g., Section – 'a' & Serial number is 1, then your IP address should be 10.0.1.1. Section – 'h' & & Serial number is 23, then your IP address should be 10.0.8.23)

Task 1: Linux Interface Configuration (ifconfig / IP command)

Step 1: To display status of all active network interfaces.

ifconfig (or) **ip addr show**

Analyze and fill the following table:

ip address table:

Interface name	**IP address (IPv4 / IPv6)**	**MAC address**	

Step 2: To assign an IP address to an interface, use the following command.

sudo ifconfig interface_name 10.0.your_section.your_sno netmask 255.255.255.0 (or)

sudo ip addr add 10.0.your_section.your_sno /24 dev interface_name

Step 3: To activate / deactivate a network interface, type.

sudo ifconfig interface_name down

sudo ifconfig interface_name up

Step 4: To show the current neighbor table in kernel, type

ip neigh

Task 2: Ping PDU (Packet Data Units or Packets) Capture

Step 1: Assign an IP address to the system (Host).

Note: IP address of your system should be 10.0.your_section.your_sno.

Step 2: Launch Wireshark and select 'any' interface

Step 3: In terminal, type **ping 10.0.your_section.your_sno**

Observations to be made

Step 4: Analyze the following in Terminal

- TTL
- Protocol used by ping
- Time

Step 5: Analyze the following in Wireshark

On Packet List Pane, select the first echo packet on the list. On Packet Details Pane, click on each of the four "+" to expand the information. Analyze the frames with the first echo request and echo reply and complete the table below.

Details	First Echo Request	First Echo Reply
Frame Number		
Source IP address		
Destination IP address		
ICMP Type Value		
ICMP Code Value		
Source Ethernet Address		
Destination Ethernet Address		
Internet Protocol Version		
Time To Live (TTL) Value		

Task 3: HTTP PDU Capture

Using Wireshark's Filter feature

Step 1: Launch Wireshark and select 'any' interface. On the Filter toolbar, type-in 'http' and press enter

Step 2: Open Firefox browser, and browse www.flipkart.com

Observations to be made

Step 3: Analyze the first (interaction of host to the web server) and second frame (response of server to the client). By analyzing the filtered frames, complete the table below:

Details	First Echo Request	First Echo Reply
Frame Number		
Source Port		
Destination Port		
Source IP address		
Destination IP address		
Source Ethernet Address		
Destination Ethernet Address		

Step 4: Analyze the HTTP request and response and complete the table below.

HTTP Request		HTTP Response	
Get		Server	
Host		Content-Type	
User-Agent		Date	
Accept-Language		Location	
Accept-Encoding		Content-Length	
Connection		Connection	

Using Wireshark's Follow TCP Stream

Step 1: Make sure the filter is blank. Right-click any packet inside the Packet List Pane, then select 'Follow TCP Stream'. For demo purpose, a packet containing the HTTP GET request "GET / HTTP / 1.1" can be selected.

Step 2: Upon following a TCP stream, screenshot the whole window.

Task 4: Capturing packets with tcpdump

Step 1: Use the command **tcpdump -D** to see which interfaces are available for capture.

sudo tcpdump -D

Step 2: Capture all packets in any interface by running this command:

sudo tcpdump -i any

Note: Perform some pinging operation while giving above command. Also type www.google.com in browser.

Observation

Step 3: Understand the output format.

Step 4: To filter packets based on protocol, specifying the protocol in the command line. For example, capture ICMP packets only by using this command:

sudo tcpdump -i any -c5 icmp

Step 5: Check the packet content. For example, inspect the HTTP content of a web request like this:

sudo tcpdump -i any -c10 -nn -A port 80

Step 6: To save packets to a file instead of displaying them on screen, use the option -w:

sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80

Task 5: Perform Traceroute checks

Step 1: Run the traceroute using the following command.

sudo traceroute www.google.com

Step 2: Analyze destination address of google.com and no. of hops

Step 3: To speed up the process, you can disable the mapping of IP addresses with hostnames by using the *-n* option

sudo traceroute -n www.google.com

Step 4: The -I option is necessary so that the traceroute uses ICMP.

sudo traceroute -I www.google.com

Step 5: By default, traceroute uses icmp (ping) packets. If you'd rather test a TCP connection to gather data more relevant to web server, you can use the -T flag.

sudo traceroute -T www.google.com

Task 6: Explore an entire network for information (Nmap)

Step 1: You can scan a host using its host name or IP address, for instance.

nmap www.pes.edu

Step 2: Alternatively, use an IP address to scan.

nmap 163.53.78.128

Step 3: Scan multiple IP address or subnet (IPv4)

nmap 192.168.1.1 192.168.1.2 192.168.1.3

Task 7 a): Netcat as Chat tool

a) Intra system communication (Using 2 terminals in the same system)

Step 1: Open a terminal (Ctrl+Alt+T). This will act as a Server.

Step 2: Type **nc -l any_portnum** (For eg., nc -l 1234)

Note: It will goto listening mode

Step 3: Open another terminal and this will act as a client.

Step 4: Type nc <your-system-ip-address> portnum

Note: portnum should be common in both the terminals (for eg., nc 10.0.2.8 1234)

Step 5: Type anything in client will appear in server

Note: For the below tasks, create 2 virtual machines. Use VMware Workstation or VirtualBox.

b) Inter system communication

Setup a simple switched network of 2 PCs with one acting as Web server. Assign IP addresses for both PCs. Set the capture option as described above.

Step 1: Open terminal on Server machine (Machine 1).

Step 2: Type **nc -l any_portnum**

Step 3: Open terminal on the Client machine (Machine 2)

Step 4: Type nc <server-ip-address> portnum

Step 5: Type anything in client will appear in the server terminal

Task 7 b): Use Netcat to Transfer Files

The netcat utility can also be used to transfer files.

Step 1: At the server side, create an empty file named 'test.txt'

sudo nc -l 555 > test.txt

Step 2: At the client side, we have a file 'testfile.txt'. Add some contents to it.

Step 3: Run the client as:

sudo nc 10.0.2.8 555 < testfile.txt

here, 10.0.2.8 is the IP address of server and 555 is the port number.

Step 4: At server side, verify the file transfer using the command

cat test.txt

Task 7 c): Other Commands

1) To test if a particular TCP port of a remote host is open.

nc -vn 10.0.2.8 555

2) Run a web server with a static web page.
   Step 1: Run the command below on local host (e.g. 10.0.2.8) to start a web server that serves test.html on port 80.
   while true; do sudo nc -lp 80 < test.html; done

   Step 2: Now open **http://10.0.2.8/test.html** from another host to access it.

   Step 3: Observe the details on the terminal

Questions on above observations:

1) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server?
2) When was the HTML file that you are retrieving last modified at the server?
3) How to tell ping to exit after a specified number of ECHO_REQUEST packets?
4) How will you identify remote host apps and OS?

Exercises:

1) Capture and Analyze IPv4 / IPv6 packets

IPv4 / IPv6 packet header

GET	
HOST	
USER-AGENT	
ACCEPT-LANGUAGE	
CACHE-CONTROL	
PRAGMA	
CONNECTION	

2) Explore various other network configuration, troubleshooting and debugging tools such as Route, Netstat, etc.

Computer Networks Lab – UE18CS304

Ruchika Shashidhara – PES1201800046 – Sem 5 Sec D

Week #1 - Understanding Basic Networking Tools – 06 Sept 2020

OBJECTIVE: To study & understand the basic networking tools – Wireshark, Tcpdump, Ping, Traceroute

Task 1: Linux Interface Configuration (ifconfig / IP command)

Display status of all active network interfaces:

$ ifconfig

```
ubuntu1@ubuntu1-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::4e89:5c22:ea08:58b4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e0:93:48  txqueuelen 1000  (Ethernet)
        RX packets 1511  bytes 1981030 (1.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 667  bytes 87656 (87.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 199  bytes 17062 (17.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 199  bytes 17062 (17.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

IP address table

Interface name	IP address (IPv4 / IPv6)	MAC address	Description
enp0s3	10.0.2.15 fe80::4e89:5c22:ea08:58b4	08:00:27:e0:93:48	Ethernet Network Peripheral
lo	127.0.0.1 ::1	-	Loopback

Loopback interface is not connected on any physical media, hence these is no purpose for a link local address with MAC address. MAC addresses(link local layer 2 addess) is used to distinguish induvidual devices on the media.

Assigning IP address – 10.0.2.4 (Section D, Roll No. 2) to the enp0s3 (Ethernet Network Peripheral) interface

$ sudo ifconfig enp0s3 10.0.4.2 netmask 255.255.255.0

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo ifconfig enp0s3 10.0.4.2 netmask 255.255.255.0
[sudo] password for ubuntu1:
ubuntu1@ubuntu1-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.4.2  netmask 255.255.255.0  broadcast 10.0.4.255
        inet6 fe80::4e89:5c22:ea08:58b4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e0:93:48  txqueuelen 1000  (Ethernet)
        RX packets 8662  bytes 8597473 (8.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5703  bytes 709498 (709.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1605  bytes 168127 (168.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1605  bytes 168127 (168.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Interface name	IP address (IPv4 / IPv6)	MAC address	Description
enp0s3	10.0.4.2	08:00:27:e0:93:48	Ethernet Network Peripheral
lo	127.0.0.1 ::1	-	Loopback

IP address of loopback interface is always 127.0.0.1 (IPv4) and ::1 (IPv6) even if we change the enp0s3 Ethernet Network Interface.

$ sudo ifconfig enp0s3 down

$ sudo ifconfig enp0s3 up

$ ip neigh

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo ifconfig enp0s3 down
ubuntu1@ubuntu1-VirtualBox:~$ sudo ifconfig enp0s3 up
ubuntu1@ubuntu1-VirtualBox:~$ ip neigh
10.0.2.1 dev enp0s3 lladdr 52:54:00:12:35:00 REACHABLE
```

ip neigh command shows the neighbour/arp table. It shows the list of neighbour entries. They establish bindings between protocol address and link layer address for hosts sharing the same link

Task 2: Ping PDU (Packet Data Units or Packets) Capture

$ ping -c 5 10.0.4.2

```
ubuntu1@ubuntu1-VirtualBox:~$ ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.4.2  netmask 255.255.255.0  broadcast 10.0.4.255
        inet6 fe80::4e89:5c22:ea08:58b4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e0:93:48  txqueuelen 1000  (Ethernet)
        RX packets 10271  bytes 10351956 (10.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6742  bytes 809200 (809.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
ubuntu1@ubuntu1-VirtualBox:~$ ping -c 5 10.0.4.2
PING 10.0.4.2 (10.0.4.2) 56(84) bytes of data.
64 bytes from 10.0.4.2: icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from 10.0.4.2: icmp_seq=2 ttl=64 time=0.090 ms
64 bytes from 10.0.4.2: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 10.0.4.2: icmp_seq=4 ttl=64 time=0.035 ms
64 bytes from 10.0.4.2: icmp_seq=5 ttl=64 time=0.070 ms

--- 10.0.4.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4072ms
rtt min/avg/max/mdev = 0.026/0.062/0.092/0.027 ms
```

Ping command sends an ICMP echo request & receives active/inactive responses ICMP echo replies. It also displays the round-trip delay when communicating with the host with packet losses.

- **TTL: 64**
- **Protocol used by ping: ICMP – Internet Control Message Protocol**
- **Time: 4072ms**

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) request id=0x0006, seq=1/256, ttl=64 (reply in 2)
2	0.000010185	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) reply id=0x0006, seq=1/256, ttl=64 (request in 1)
3	0.999727625	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) request id=0x0006, seq=2/512, ttl=64 (reply in 4)
4	0.999751552	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) reply id=0x0006, seq=2/512, ttl=64 (request in 3)
5	2.024329831	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) request id=0x0006, seq=3/768, ttl=64 (reply in 6)
6	2.024353851	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) reply id=0x0006, seq=3/768, ttl=64 (request in 5)
7	3.047921341	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) request id=0x0006, seq=4/1024, ttl=64 (reply in 8)
8	3.047929543	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) reply id=0x0006, seq=4/1024, ttl=64 (request in 7)
9	4.071712927	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) request id=0x0006, seq=5/1280, ttl=64 (reply in 10)
10	4.071732757	10.0.4.2	10.0.4.2	ICMP	100	Echo (ping) reply id=0x0006, seq=5/1280, ttl=64 (request in 9)

```
▸ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture
▾ Internet Protocol Version 4, Src: 10.0.4.2, Dst: 10.0.4.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x12ef (4847)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x0bb7 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.4.2
    Destination: 10.0.4.2
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xafa5 [correct]
    [Checksum Status: Good]
    Identifier (BE): 6 (0x0006)
    Identifier (LE): 1536 (0x0600)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 2]
    Timestamp from icmp data: Sep  5, 2020 22:16:58.000000000 IST
    [Timestamp from icmp data (relative): 0.221248270 seconds]
  ▸ Data (48 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ···· ···· ····· ····
0010  45 00 00 54 12 ef 40 00  40 01 0b b7 0a 00 04 02   E··T··@· @·· ····
0020  0a 00 04 02 08 00 af a5  00 06 00 01 02 c1 53 5f   ···· ···· ····· ··S_
0030  00 00 00 00 30 60 03 00  00 00 00 00 10 11 12 13   ····0`·· ····· ····
0040  14 15 16 17 18 19 1a 1b  1c 1d 1e 1f 20 21 22 23   ···· ···· ···· ···· !"#
0050  24 25 26 27 28 29 2a 2b  2c 2d 2e 2f 30 31 32 33   $%&'()*+ ,-./0123
0060  34 35 36 37                                       4567
```

```
▸ Frame 2: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
▸ Linux cooked capture
▾ Internet Protocol Version 4, Src: 10.0.4.2, Dst: 10.0.4.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x12f0 (4848)
  ▸ Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x4bb6 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.4.2
    Destination: 10.0.4.2
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xb7a5 [correct]
    [Checksum Status: Good]
    Identifier (BE): 6 (0x0006)
    Identifier (LE): 1536 (0x0600)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Request frame: 1]
    [Response time: 0.010 ms]
    Timestamp from icmp data: Sep  5, 2020 22:16:58.000000000 IST
    [Timestamp from icmp data (relative): 0.221258455 seconds]
  ▸ Data (48 bytes)
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ···· ···· ····· ····
0010  45 00 00 54 12 f0 00 00  40 01 4b b6 0a 00 04 02   E··T···· @·K· ····
0020  0a 00 04 02 00 00 b7 a5  00 06 00 01 02 c1 53 5f   ···· ···· ····· ··S_
0030  00 00 00 00 30 60 03 00  00 00 00 00 10 11 12 13   ····0`·· ····· ····
0040  14 15 16 17 18 19 1a 1b  1c 1d 1e 1f 20 21 22 23   ···· ···· ···· ···· !"#
0050  24 25 26 27 28 29 2a 2b  2c 2d 2e 2f 30 31 32 33   $%&'()*+ ,-./0123
0060  34 35 36 37                                       4567
```

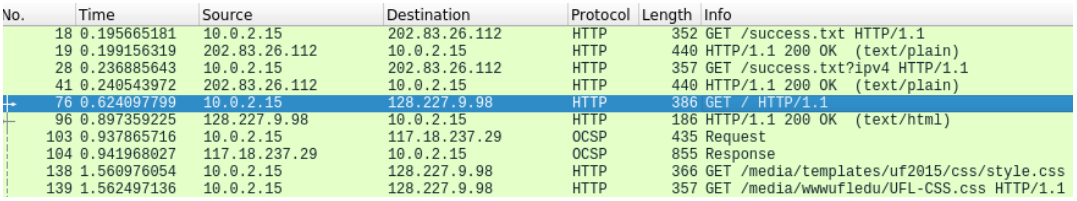
Details	First Echo Request	First Echo Reply
Frame Number	1	2
Source IP Address	10.0.4.2	10.0.4.2
Destination IP Address	10.0.4.2	10.0.4.2
ICMP Type Value	8 (Echo (ping) request)	0 (Echo (ping) reply)
ICMP Code Value	0	0
Source Ethernet Address	08:00:27:e0:93:48	08:00:27:e0:93:48
Destination Ethernet Address	08:00:27:e0:93:48	08:00:27:e0:93:48
Internet Protocol Version	4	4
Time to Live (TTL) Value	64	64

Task 3: HTTP PDU Capture using Wireshark's Filter feature

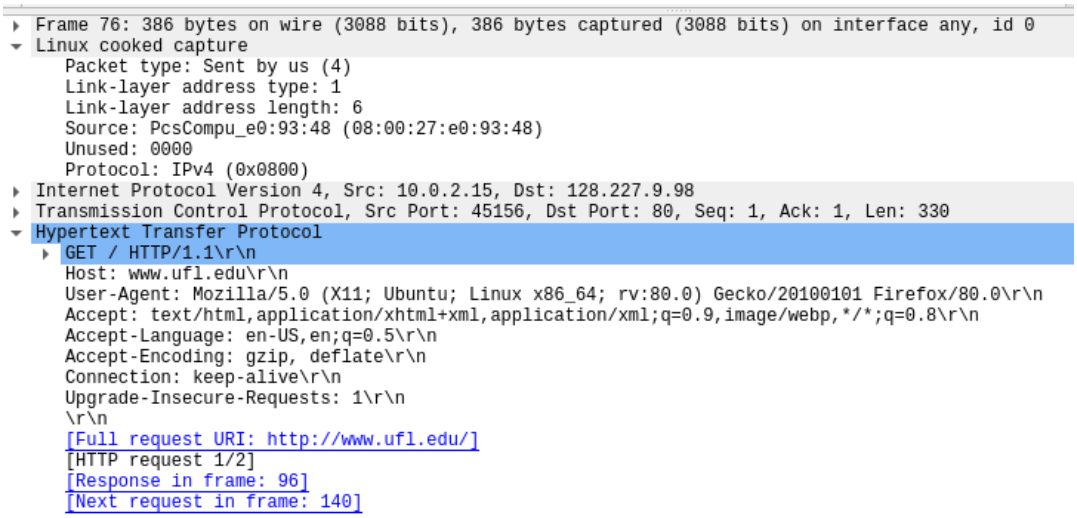
$ sudo wireshark

$ firefox www.ufl.edu

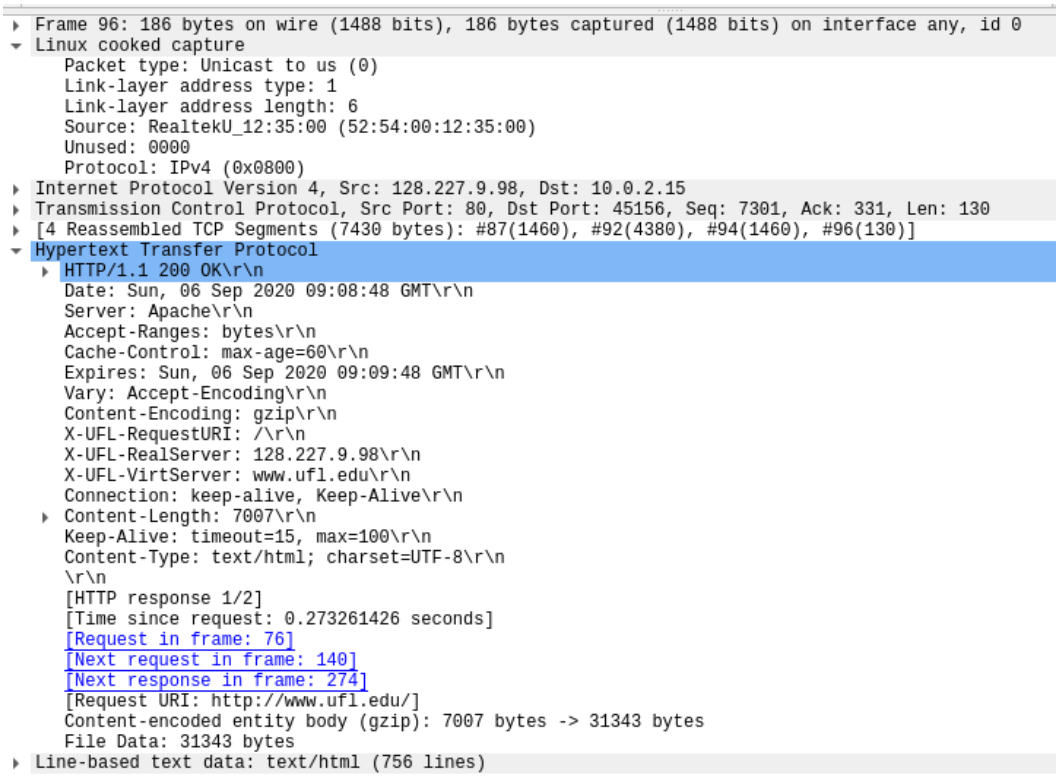
(Used example www.ufl.edu (http website) because in the packet capture of www.flipkart.com, 301 HTTP, URL moved permanently issue was being displayed)



No.	Time	Source	Destination	Protocol	Length	Info
18	0.195665181	10.0.2.15	202.83.26.112	HTTP	352	GET /success.txt HTTP/1.1
19	0.199156319	202.83.26.112	10.0.2.15	HTTP	440	HTTP/1.1 200 OK (text/plain)
28	0.236885643	10.0.2.15	202.83.26.112	HTTP	357	GET /success.txt?ipv4 HTTP/1.1
41	0.240543972	202.83.26.112	10.0.2.15	HTTP	440	HTTP/1.1 200 OK (text/plain)
76	0.624097799	10.0.2.15	128.227.9.98	HTTP	386	GET / HTTP/1.1
96	0.897359225	128.227.9.98	10.0.2.15	HTTP	186	HTTP/1.1 200 OK (text/html)
103	0.937865716	10.0.2.15	117.18.237.29	OCSP	435	Request
104	0.941968027	117.18.237.29	10.0.2.15	OCSP	855	Response
138	1.560976054	10.0.2.15	128.227.9.98	HTTP	366	GET /media/templates/uf2015/css/style.css
139	1.562497136	10.0.2.15	128.227.9.98	HTTP	357	GET /media/wwwufledu/UFL-CSS.css HTTP/1.1



```
▸ Frame 76: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface any, id 0
▾ Linux cooked capture
    Packet type: Sent by us (4)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: PcsCompu_e0:93:48 (08:00:27:e0:93:48)
    Unused: 0000
    Protocol: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 128.227.9.98
▸ Transmission Control Protocol, Src Port: 45156, Dst Port: 80, Seq: 1, Ack: 1, Len: 330
▾ Hypertext Transfer Protocol
  ▸ GET / HTTP/1.1\r\n
    Host: www.ufl.edu\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://www.ufl.edu/]
    [HTTP request 1/2]
    [Response in frame: 96]
    [Next request in frame: 140]
```



```
▸ Frame 96: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface any, id 0
▾ Linux cooked capture
    Packet type: Unicast to us (0)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: RealtekU_12:35:00 (52:54:00:12:35:00)
    Unused: 0000
    Protocol: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 128.227.9.98, Dst: 10.0.2.15
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 45156, Seq: 7301, Ack: 331, Len: 130
▸ [4 Reassembled TCP Segments (7430 bytes): #87(1460), #92(4380), #94(1460), #96(130)]
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Date: Sun, 06 Sep 2020 09:08:48 GMT\r\n
    Server: Apache\r\n
    Accept-Ranges: bytes\r\n
    Cache-Control: max-age=60\r\n
    Expires: Sun, 06 Sep 2020 09:09:48 GMT\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    X-UFL-RequestURI: /\r\n
    X-UFL-RealServer: 128.227.9.98\r\n
    X-UFL-VirtServer: www.ufl.edu\r\n
    Connection: keep-alive, Keep-Alive\r\n
  ▸ Content-Length: 7007\r\n
    Keep-Alive: timeout=15, max=100\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.273261426 seconds]
    [Request in frame: 76]
    [Next request in frame: 140]
    [Next response in frame: 274]
    [Request URI: http://www.ufl.edu/]
    Content-encoded entity body (gzip): 7007 bytes -> 31343 bytes
    File Data: 31343 bytes
▸ Line-based text data: text/html (756 lines)
```

Details	First Echo Request	First Echo Reply
Frame Number	76	97
Source Port	45156	80
Destination Port	80	45156
Source IP Address	10.0.2.15	128.227.9.98
Destination IP Address	128.227.9.98	10.0.2.15
Source Ethernet Address	08:00:27:e0:93:48	52:54:00:12:35:00
Destination Ethernet Address	-	-

HTTP Request		HTTP Response	
Get	/ HTTP/1.1	Server	Apache
Host	www.ufl.edu	Content-Type	text/html; charset=UTF-8
User-Agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0	Date	Sun, 06 Sept 2020 09:08:48 GMT
Accept-Language	en-US, en, q=0.5	Location	-
Accept-Encoding	gzip, deflate	Content-Length	7007
Connection	keep-alive	Connection	keep-alive

Wireshark's Follow TCP Stream

Task 4: Capturing packets with tcpdump

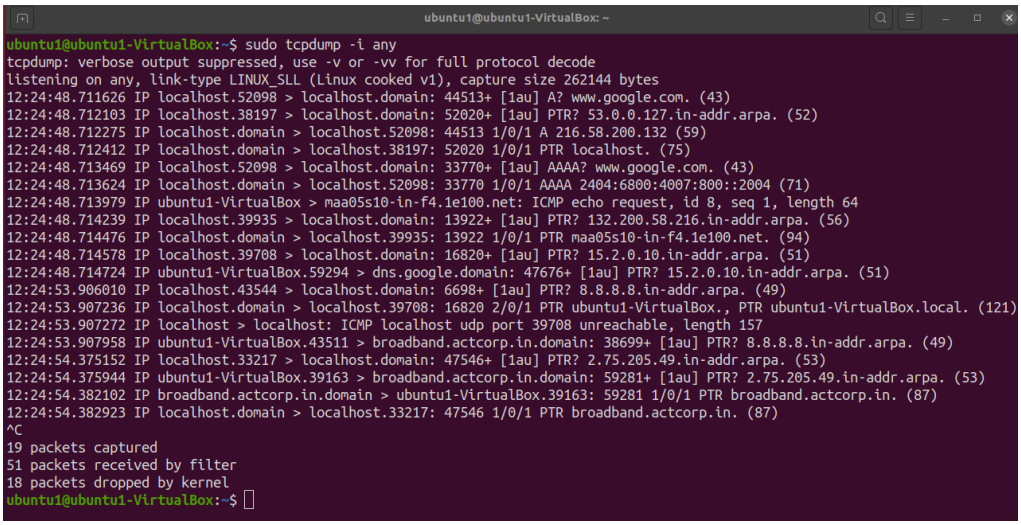
$ sudo tcpdump -D

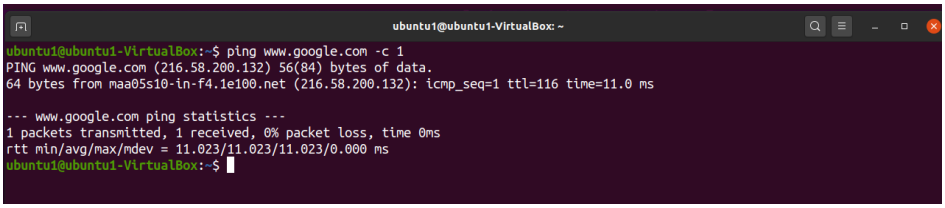
```
ubuntu1@ubuntu1-VirtualBox:~$ sudo tcpdump -D
[sudo] password for ubuntu1:
1.enp0s3 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
```

tcpdump -D lists the network interfaces available on the system on which tcpdump can capture packets. For each network interface. On my system, Ethernet Network Peripheral(enp0s3), Loopback(lo) and Any (any – allows capturing of any active interface) interfaces are all up and running and can capture packets.

$ sudo tcpdump -i any

$ ping www.google.com -c 1

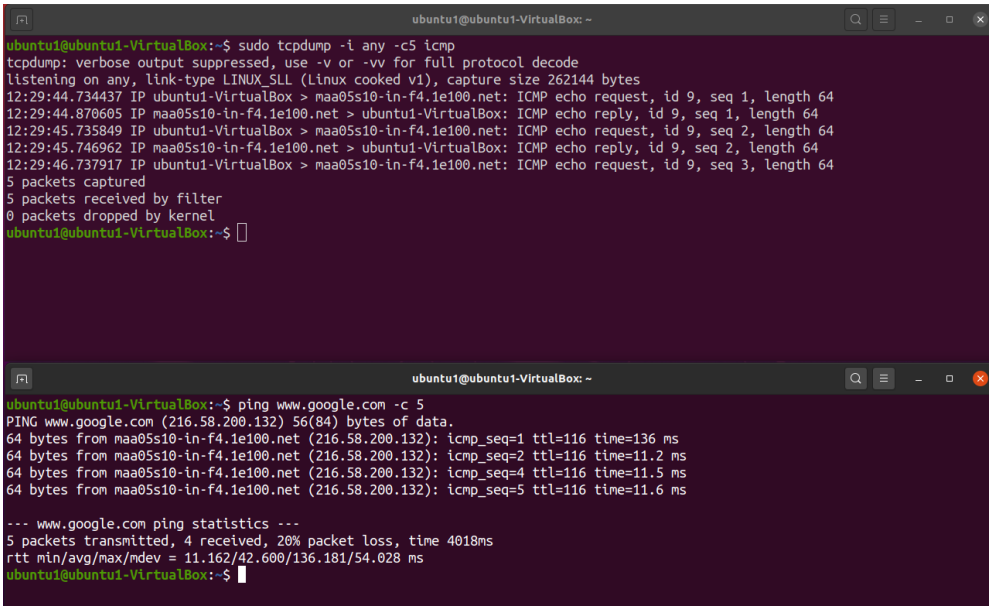




tcpdump -i any captures all packets in any interface until it is manually terminated (Ctrl + C). -i option is the interface on which tcpdump should listen, here interface specified is any active interface.

$ sudo tcpdump -i any -c5 icmp

$ ping www.google.com -c 5



To filter packets based on protocol, we can specify it with the command. Here, we specify icmp to only capture ICMP captures. Here icmp packet is captured from any active interface for 5 packets.

$ sudo tcpdump -i any -c10 -nn -A port 80

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo tcpdump -i any -c10 -nn -A port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
13:25:26.515077 IP 10.0.2.15.50892 > 202.83.26.112.80: Flags [S], seq 1709231326, win 64240, optio
ns [mss 1460,sackOK,TS val 632290636 ecr 0,nop,wscale 7], length 0
E..<..@.@...
....S.p...Pe.....................
%..L........
13:25:26.519858 IP 202.83.26.112.80 > 10.0.2.15.50892: Flags [S.], seq 36634, ack 1709231327, win
32768, options [mss 1460], length 0
E..,.......s.S.p
....P......e...`....L........
13:25:26.519888 IP 10.0.2.15.50892 > 202.83.26.112.80: Flags [.], ack 1, win 64240, length 0
E..(..@.@...
....S.p...Pe.......P.......
13:25:26.526399 IP 10.0.2.15.50892 > 202.83.26.112.80: Flags [P.], seq 1:297, ack 1, win 64240, le
ngth 296: HTTP: GET /success.txt HTTP/1.1
E..P..@.@...
....S.p...Pe.......P.......GET /success.txt HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cache-Control: no-cache
Pragma: no-cache
Connection: keep-alive
```

```
13:25:26.530537 IP 202.83.26.112.80 > 10.0.2.15.50892: Flags [P.], seq 1:385, ack 297, win 32472,
length 384: HTTP: HTTP/1.1 200 OK
E............S.p
....P......e...P.~..H..HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 8
Last-Modified: Mon, 15 May 2017 18:04:40 GMT
ETag: "ae780585f49b94ce1444eb7d28906123"
Accept-Ranges: bytes
Server: AmazonS3
X-Amz-Cf-Id: 0n5m87AhdS46-O-aDyBovf89wpiK_DdJ1kZKuIteCQVcUMN1Vf2Vkw==
Cache-Control: no-cache, no-store, must-revalidate
Date: Sun, 06 Sep 2020 07:55:26 GMT
Connection: keep-alive

success
```

```
13:25:26.530557 IP 10.0.2.15.50892 > 202.83.26.112.80: Flags [.], ack 385, win 63856, length 0
E..(..@.@...
....S.p...Pe.......P..p....
13:25:26.585770 IP 10.0.2.15.50894 > 202.83.26.112.80: Flags [S], seq 3831717018, win 64240, optio
ns [mss 1460,sackOK,TS val 632290707 ecr 0,nop,wscale 7], length 0
E..<~.@.@...
....S.p...P.c\..................
%..........
13:25:26.589132 IP 202.83.26.112.80 > 10.0.2.15.50894: Flags [S.], seq 37394, ack 3831717019, win
32768, options [mss 1460], length 0
E..,.......q.S.p
....P.......c\.`.............
13:25:26.589171 IP 10.0.2.15.50894 > 202.83.26.112.80: Flags [.], ack 1, win 64240, length 0
E..(~.@.@...
....S.p...P.c\.....P.......
13:25:26.589424 IP 10.0.2.15.50894 > 202.83.26.112.80: Flags [P.], seq 1:302, ack 1, win 64240, le
ngth 301: HTTP: GET /success.txt?ipv4 HTTP/1.1
E..U~.@.@...
....S.p...P.c\.....P.......GET /success.txt?ipv4 HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache


10 packets captured
10 packets received by filter
0 packets dropped by kernel
ubuntu1@ubuntu1-VirtualBox:~$
```

tcpdump resolves IP addresses and ports into names by default. To disable this, and view the IP addresses and port numbers themselves, we can use -nn option

- A option lets us to print each packet in ASCII, this is helpful for capturing web page packets

Port filter option lets us to filter the packets based on the desired service or port. Here, the packets related to HTTP web service is captured with the filter port 80.

Content of a TCP packet captured example:

13:25:26.530537 IP 202.83.26.112.80 > 10.0.2.15.50892: Flags [P.], seq 1:385, ack 297, win 32472, length 384: HTTP: HTTP/1.1 200 OK

13:25:26.530537 specifies the timestamp of the received packet, as per the local clock, here (IST)

IP represents the network layer protocol, here it is IPv4

202.83.26.112.80 represents the source IP address & port

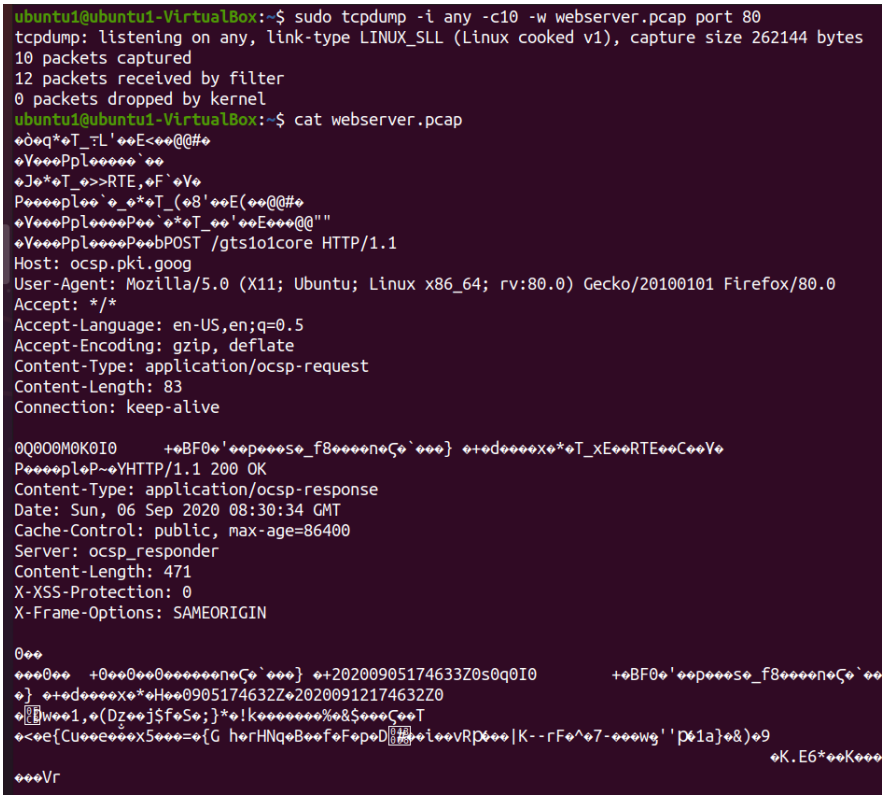
10.0.2.15.50892 represents the destination IP address & port

Flags [P.], seq 1:385, ack 297, win 32472, length 384 represents the TCP flags

- **seq 1:385 means that this packet contains bytes 1 – 385**
- **ack 297 means that we are receiving data from the server**
- **win 32472 represents the number of bytes available in the receiving buffer (TCP parameters)**
- **length 384 represents the length of the payload data in bytes**

HTTP: HTTP/1.1 200 OK represents the HTTP response, here it is running HTTP version 1.1 and we are receiving a successful 200 OK response.

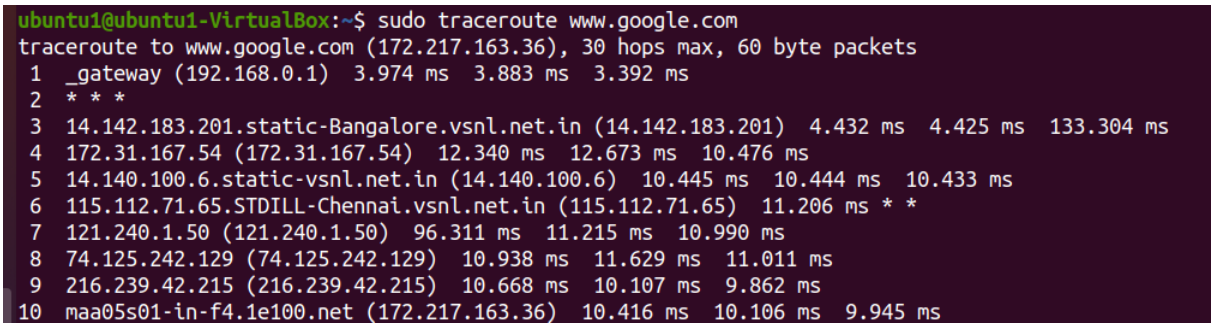
$ sudo tcpdump -i any -c10 -nn -w webserver.pcap port 80



We use the above command to save the packets into a file – webserver.pcap instead of displaying it on screen with the option -w <filename>. Here, we are saving the packets into a write-protected file, hence the above output is displayed with cat.

Task 5: Perform Traceroute checks

$ sudo traceroute www.google.com



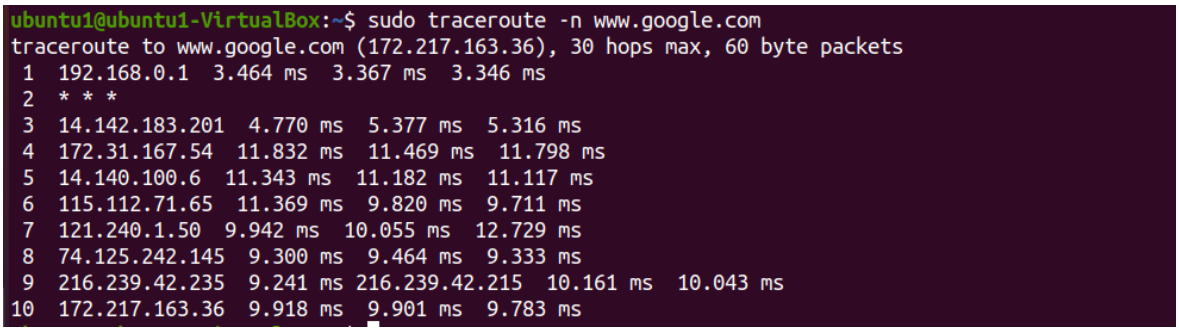
Traceroute is a network tool used to show the route taken by packets from the local machine across an IP network host, here www.google.com. It displays max hops (here 30) and packet size (here 60 bytes) used. It also displays each hop sequentially with hop number, roundtrip times, best time(ms), IP address, TTL, etc. for each.

Output for each traceroute for www.google.com:

Destination address of www.google.com: maa05s01-in-f4.1e100.net (172.217.163.36)

Number of hops: 10

$ sudo traceroute -n www.google.com



- n option when used, displays the same output for each hop but it does not resolve the IP addresses to their domain, it just displays each hops IPv4 addresses itself.

$ sudo traceroute -I www.google.com

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo traceroute -I www.google.com
traceroute to www.google.com (172.217.26.196), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  2.669 ms  2.586 ms  2.575 ms
 2  10.240.0.1 (10.240.0.1)  3.791 ms  3.788 ms  3.777 ms
 3  14.142.183.201.static-Bangalore.vsnl.net.in (14.142.183.201)  5.919 ms  6.432 ms  6.433 ms
 4  172.31.167.54 (172.31.167.54)  14.229 ms  14.216 ms  14.207 ms
 5  14.140.100.6.static-vsnl.net.in (14.140.100.6)  14.164 ms  14.172 ms  14.164 ms
 6  115.112.71.65.STDILL-Chennai.vsnl.net.in (115.112.71.65)  15.372 ms  10.728 ms  10.616 ms
 7  121.240.1.50 (121.240.1.50)  11.235 ms  14.366 ms  14.283 ms
 8  108.170.253.113 (108.170.253.113)  11.494 ms  11.504 ms  11.495 ms
 9  72.14.237.165 (72.14.237.165)  11.493 ms  11.488 ms  11.482 ms
10  maa03s23-in-f196.1e100.net (172.217.26.196)  11.475 ms  11.845 ms  11.846 ms
```

- I option when used, displays the output for each hop, but it uses ICMP ECHO for probes/tracerouting instead. ICMP is one of the protocols for TCP/IP suite and are also commonly used for ping messages

$ sudo traceroute -T www.google.com

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo traceroute -T www.google.com
traceroute to www.google.com (172.217.26.196), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  5.207 ms  5.100 ms  5.082 ms
 2  10.240.0.1 (10.240.0.1)  4.629 ms  4.104 ms  4.008 ms
 3  14.142.183.201.static-Bangalore.vsnl.net.in (14.142.183.201)  3.989 ms  6.246 ms  4.631 ms
 4  172.31.167.54 (172.31.167.54)  10.447 ms  10.402 ms  10.303 ms
 5  14.140.100.6.static-vsnl.net.in (14.140.100.6)  10.217 ms  10.202 ms  10.003 ms
 6  115.112.71.65.STDILL-Chennai.vsnl.net.in (115.112.71.65)  11.137 ms  12.076 ms  17.913 ms
 7  121.240.1.50 (121.240.1.50)  17.895 ms  17.867 ms  17.586 ms
 8  108.170.253.97 (108.170.253.97)  17.461 ms 108.170.253.113 (108.170.253.113)  9.188 ms  9.188 ms
 9  72.14.237.165 (72.14.237.165)  9.076 ms  9.054 ms  9.037 ms
10  maa03s23-in-f4.1e100.net (172.217.26.196)  8.951 ms  10.689 ms  10.703 ms
```

- T option when used, displays the output for each hop, but it uses TCP SYN for probes/tracerouting instead.

Task 6: Explore an entire network for information (Nmap)

Scan host www.pes.edu using hostname

$ nmap www.pes.edu

```
ubuntu1@ubuntu1-VirtualBox:~$ nmap www.pes.edu
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-06 18:38 IST
Nmap scan report for www.pes.edu (13.71.123.138)
Host is up (0.031s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 124.10 seconds
```

nmap is a security scanner used to list hosts and services on a computer network. It can be used for host detection, port scanning, version, OS identification and detection. It also displays the port table which displays the information of the port number and protocol

When we scan the host www.pes.edu, the following output is displayed:

- **Timestamp of scan – 2020-09-06 18:38 IST**
- **Host name – www.pes.edu**

- **IP address – 13.71.123.138**
- **Host is active with 0.031s latency**
- **998 filtered ports were found**
- **Port table – both http & https services are open states & their ports are 80/tcp and 443/tcp**
- **Total time taken for scan report – 124.10s**

Scan host 163.53.78.128 (www.flipkart.com) using IP address

$ nmap 163.53.78.128

```
ubuntu1@ubuntu1-VirtualBox:~$ nmap 163.53.78.128
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-06 18:42 IST
Nmap scan report for 163.53.78.128
Host is up (0.021s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 67.80 seconds
```

When we scan the IP address 163.53.78.128, the following output is displayed:

- **Timestamp of scan – 2020-09-06 18:42 IST**
- **IP address – 163.53.78.128**
- **Host is active with 0.021s latency**
- **998 filtered ports were found**
- **Port table – both http & https services are open states & their ports are 80/tcp and 443/tcp**
- **Total time taken for scan report – 67.80s**

Scan multiple IP address (192.168.1-3) or subnet (IPv4)

$ nmap 192.168.1.1 192.168.1.2 192.168.1.3

```
ubuntu1@ubuntu1-VirtualBox:~$ nmap 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-06 18:45 IST
Nmap done: 3 IP addresses (0 hosts up) scanned in 3.11 seconds
```

When we scan multiple hosts 192.168.1.1-3, the following output is displayed:

- **Timestamp of scan – 2020-09-06 18:45 IST**
- **All 3 IP addresses' hosts were not up**
- **Total time taken for scan report – 3.11s**

Questions on above based observations:

1) **Is your browser running HTTP version 1.0 or 1.1? What version of HTTP version is the server?**

Ans: As seen in Wireshark's HTTP PDU packet capture, both my browser (Firefox) and the server were each running **HTTP version 1.1.**

HTTP 1.1. allows to have persistent connections (i.e. more than one request/response can happen on the same HTTP connection) whereas HTTP 1.0 allows to have a new connection for each request/response pair. After each response the connection would be lost

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 8
Last-Modified: Mon, 15 May 2017 18:04:40 GMT
```

2) When was the HTML file you are retrieving last modified at the server?

Ans: For www.ufl.edu, seen in Wireshark's HTTP PDU packet capture, the HTML file retrieved was last modified on 15 May 2017 18:04:40 GMT.

3) How to tell ping to exit after a specific number of ECHO_REQUEST packets?

Ans: By adding an additional option **-c <num>** with the ping command, we can tell ping to exit after <num> ECHO_REQUEST packets.

$ ping -c 5 www.google.com

4) How will you identify remote host apps and OS?

Ans: nmap command in Linux OS can be used to detect remote running host apps and operating systems. Nmap probes the remote compute, based on its responses to TCP packets, it can also infer what OS it is using.

CONCLUSION:

Learnt & understood the basic networking tools:

1) Wireshark is used for performing & analysing Ping PDU captures, examining & analysing HTTP packet captures with filters, etc.
2) Tcpdump is used to see how packets are being captured
3) Ping is used to test the connectivity between 2 systems
4) Nmap is used for exploring an entire network

Week #2

Understanding Persistent and Non-persistent HTTP Connections

To understand persistent and non-persistent HTTP connections and corresponding performance impact.

Create a web page with N (e.g. 10) embedded images. Each image should be of minimum 2 MB size. Configure your browser (Firefox) with following settings (each setting requires repeat of experiment)

- Non persistent connection
- 2 persistent connections
- 4 persistent connections
- 6 persistent connections
- 10 persistent connections.

Observation: Note down the time taken to display the entire page in each of the settings. Ensure that (cache is cleared before starting the web request). Explain the response time differences. What is the optimal number of persistent connections for best performance? Explain your answer.

Introduction

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features including dynamically loadable modules, robust media support, and extensive integration with other popular software.

Objective: Understand persistent and non-persistent HTTP connections and corresponding performance impact.

Experiment: Create a web page with N (e.g. 10) embedded images. Each image should be of minimum 2 MB size. Configure your browser (Firefox) with following settings (each setting requires repeat of experiment)

a) Non-persistent connection
b) 2 persistent connections
c) 4 persistent connections
d) 6 persistent connections
e) 10 persistent connections

Note down the time taken to display the entire page in each of the settings. Ensure that cache is cleared before starting the web request. Explain the response time differences. What is the optimal number of persistent connections for best performance? Explain your answer.

Note: To install Apache server, use the following command,

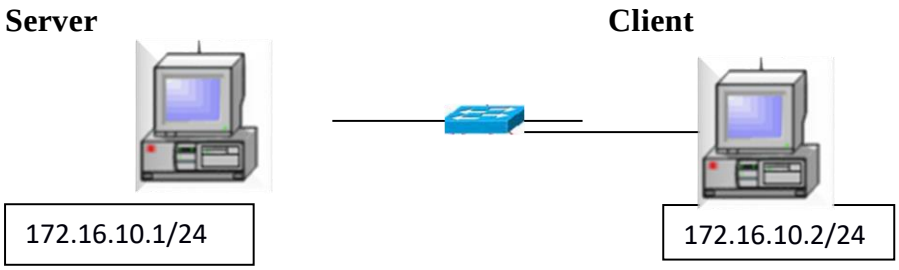
```
sudo apt-get install apache2
```

If there is any error during installation, update the package manager by issuing the command,

```
sudo apt-get update
```

EXECUTION STEPS

Step 1: Connect 2 desktops using switch and cables as shown below. (Use 2 VMs on Virtualbox or VMware instead of physical connections.)



Server Side:

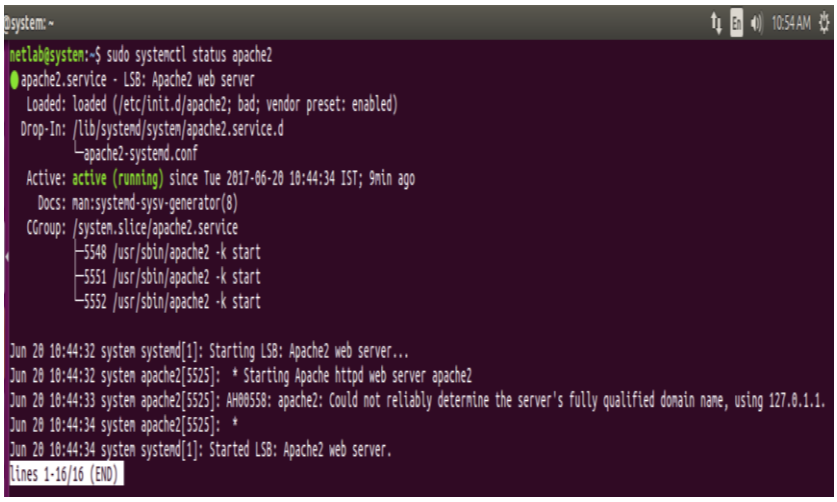
Step 2: Check your Web Server

At the end of the installation process, Ubuntu 16.04 starts Apache. The web server should already be up and running. We can check with the `systemctl` command to make sure the service is running by typing:

sudo systemctl status apache2

or

sudo service apache2 status



As you can see above, the service appears to have started successfully. However, the best way to test this is to actually request a page from Apache. You can access the default Apache landing page to confirm that the software is running properly. You can access this through your server's domain name or IP address.

Step 3: Server IP address can be set by the following command

$sudo ip addr add 172.16.10.1/24 dev enps0
$sudo ip addr

Note: If IP address fluctuates, kindly setup the IP address manually using ‘Edit connections’.

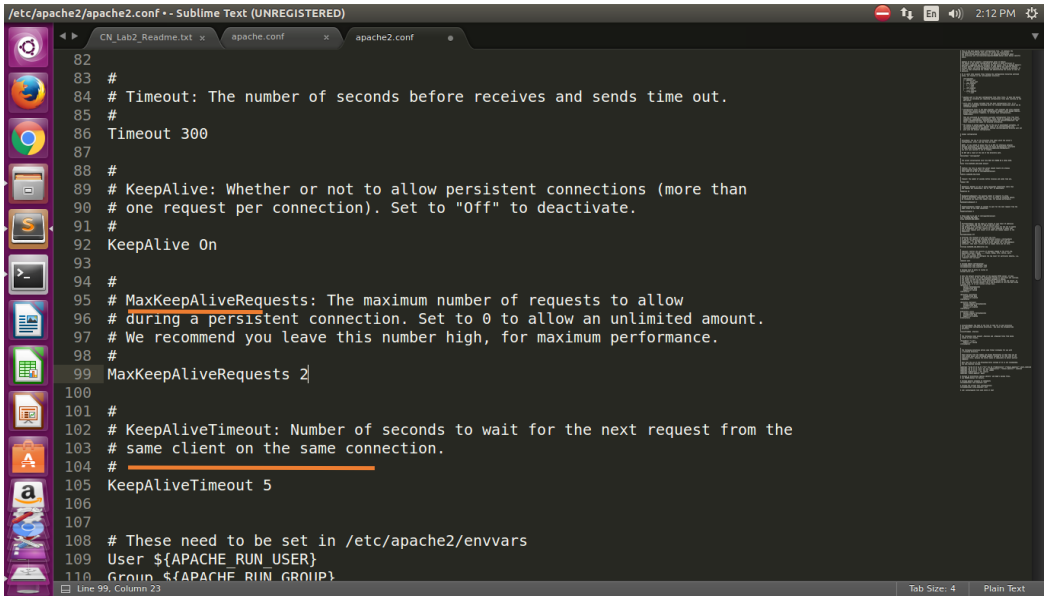
```
student@student-H81H3-I:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:ae:ed:a5:a5:a9 brd ff:ff:ff:ff:ff:ff
    inet 172.16.10.1/24 brd 172.16.10.255 scope global enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::c901:c994:4cf5:f837/64 scope link
       valid_lft forever preferred_lft forever
student@student-H81H3-I:~$
```

Step 4: The **apache2.conf** file present in the **etc/apache2** directory is modified as:

a) The **keep-alive** option was set (i.e. value was made **ON**)

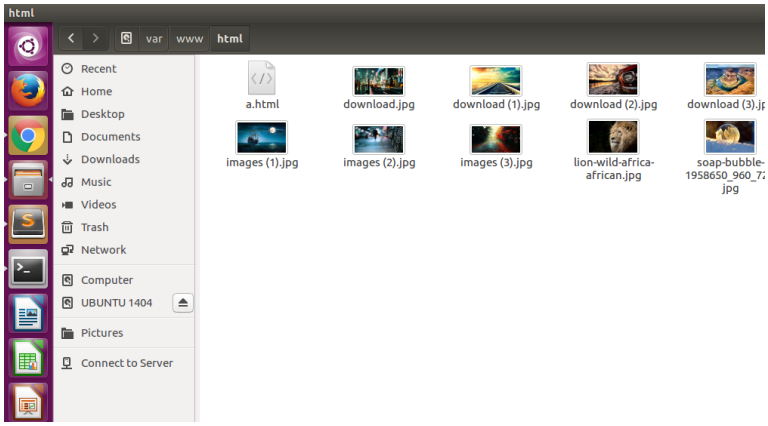
b) The **MaximumKeepAliveRequests** were set to **2**

$sudo nano /etc/apache2/apache2.conf



Step 5: Store images in the server path. A html page consisting of 10 images having size > 2MB were placed and accessed by the client. This html page is stored in the location - **/var/www/html/file_name.html**.

Note: Use the images provided by faculty incharges.



Step 6: Prepare a web page as shown below. The html file needs to add 10 images. (Kindly skip the style attribute in the below image)

a.html [Read-Only] (/var/www/html) - gedit

Open

```
<!DOCTYPE html>
<html>
<body>

<h2>Spectacular Mountain</h2>
<img src="images (1).jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="images (2).jpg" style="width:304px;height:228px;">
<img src="download (4).jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="images (3).jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="lion-wild-africa-african.jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="images.jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="download.jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="download (1).jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="soap-bubble-1958650_960_720.jpg" alt="Mountain View" style="width:304px;height:228px;">
<img src="download (2).jpg" alt="Mountain View" style="width:304px;height:228px;">

</body>
</html>
```

Client side:

Client IP address can be set by the following command.

$sudo ip addr add 172.16.10.2/24 dev enps0

$sudo ip addr

Note: If IP address fluctuates, kindly setup the IP address manually using ‘Edit connections’.

```
student@student-H81H3-I:~$ sudo ip addr add 172.16.10.2/24 dev enp2s0
student@student-H81H3-I:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether b8:ae:ed:a5:a6:32 brd ff:ff:ff:ff:ff:ff
    inet 172.16.10.2/24 scope global enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::8bf0:837a:849e:a79f/64 scope link
       valid_lft forever preferred_lft forever
student@student-H81H3-I:~$
```

There are broadly two parts of execution:

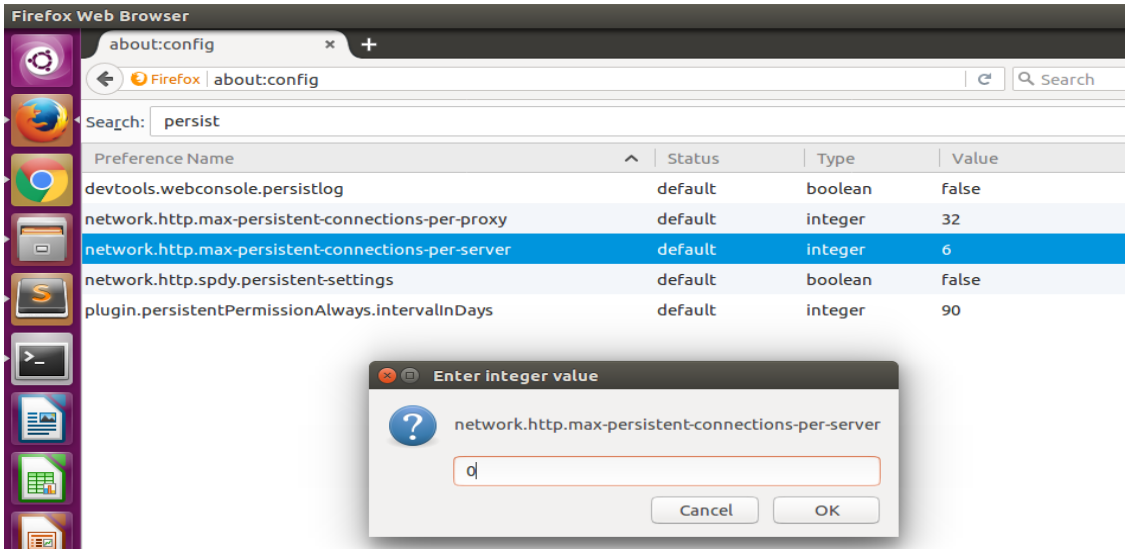
1. Dealing with non-persistent connections
2. Dealing with persistent connections

Open Firefox browser to configure for persistent option. Go to browser and type **about:config** and search for the term **'persistent'**

- While using non-persistent connection experiment, the **max-persistent-connections-per-server** has the value set to **0** and **persistent-settings** value set to false.
- While using persistent connection experiment, the **max-persistent-connections-per-server** should have value greater than 0 (depending on the number of persistent connections needed) and **persistent-settings** value set to true.

PART 1: NON-PERSISTENT CONNECTION

Step 1: This is done by setting the value of max-persistent-connection-per-server to 0 in the client computer.



Step 2: Access web page on client-side browser (Firefox)

The client could access the file as:

172.16.10.1/file_name.html where--> **172.16.10.1** is Server's IP

Here the file name is **a.html** present in server. So, by tying **172.16.10.1/a.html** in client browser, we will be able to open the requested web page.

Note 1: The wireshark should capture the packets between the client and the server while the file is accessed.

Note 2: The images in the HTML page should have all the permissions specified through the server for the proper access.

Step 3: Use wireshark**.** Open wireshark in the server computer while client is trying to access the server's local host webpage. Apply 'http' filter and note the time to capture all the 10 images.

http

No.	Time	Source	Destination	Protocol	Length	Info
25	0.211530105	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
27	2.070581279	172.16.10.2	172.16.10.1	HTTP	421	GET /a.html HTTP/1.1
28	2.070866155	172.16.10.1	172.16.10.2	HTTP	641	HTTP/1.1 200 OK (text/html)
30	2.117160769	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(1).jpg HTTP/1.1
35	2.117571913	172.16.10.1	172.16.10.2	HTTP	1200	HTTP/1.1 200 OK (JPEG JFIF image)
36	2.117753115	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(2).jpg HTTP/1.1
45	2.117944288	172.16.10.1	172.16.10.2	HTTP	463	HTTP/1.1 200 OK (JPEG JFIF image)
51	2.118574057	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(4).jpg HTTP/1.1
63	2.119058490	172.16.10.1	172.16.10.2	HTTP	242	HTTP/1.1 200 OK (JPEG JFIF image)
65	2.119487932	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(3).jpg HTTP/1.1
77	2.119784374	172.16.10.1	172.16.10.2	HTTP	565	HTTP/1.1 200 OK (JPEG JFIF image)
79	2.120323770	172.16.10.2	172.16.10.1	HTTP	359	GET /lion-wild-africa-african.jpg HTTP/1.1
94	2.121263792	172.16.10.2	172.16.10.1	HTTP	341	GET /images.jpg HTTP/1.1
110	2.122045168	172.16.10.1	172.16.10.2	HTTP	1226	HTTP/1.1 200 OK (JPEG JFIF image)
117	2.122719543	172.16.10.2	172.16.10.1	HTTP	343	GET /download.jpg HTTP/1.1
138	2.123847115	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(1).jpg HTTP/1.1
160	2.124700199	172.16.10.2	172.16.10.1	HTTP	362	GET /soap-bubble-1958650_960_720.jpg HTTP/1.1
164	2.124733805	172.16.10.1	172.16.10.2	HTTP	1017	HTTP/1.1 200 OK (JPEG JFIF image)
171	2.125125151	172.16.10.1	172.16.10.2	HTTP	711	HTTP/1.1 200 OK (JPEG JFIF image)
184	2.126599573	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(2).jpg HTTP/1.1
252	2.131056667	172.16.10.1	172.16.10.2	HTTP	114	HTTP/1.1 200 OK (JPEG JFIF image)
529	2.151487483	172.16.10.1	172.16.10.2	HTTP	73	HTTP/1.1 200 OK (JPEG JFIF image)
3834	2.429637133	172.16.10.1	172.16.10.2	HTTP	1124	HTTP/1.1 200 OK (JPEG JFIF image)

Here it is 2.429637133 - 2.070581279 = 0.359055854

PART 2: PERSISTENT CONNECTIONS

Step 1: For 2 persistent connections, set the value of **max-persistent-connection-per-server to 2** in the client computer.

Step 2: Repeat the **steps 1-3** in the previous section.

http

No.	Time	Source	Destination	Protocol	Length	Info
28	0.158495832	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
30	2.685888334	172.16.10.2	172.16.10.1	HTTP	421	GET /a.html HTTP/1.1
31	2.686488793	172.16.10.1	172.16.10.2	HTTP	641	HTTP/1.1 200 OK (text/html)
33	2.734091058	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(1).jpg HTTP/1.1
38	2.734592637	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(2).jpg HTTP/1.1
39	2.734696958	172.16.10.1	172.16.10.2	HTTP	1200	HTTP/1.1 200 OK (JPEG JFIF image)
48	2.735025557	172.16.10.1	172.16.10.2	HTTP	463	HTTP/1.1 200 OK (JPEG JFIF image)
49	2.735180365	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(4).jpg HTTP/1.1
66	2.736079156	172.16.10.1	172.16.10.2	HTTP	243	HTTP/1.1 200 OK (JPEG JFIF image)
68	2.736374643	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(3).jpg HTTP/1.1
82	2.736755733	172.16.10.1	172.16.10.2	HTTP	565	HTTP/1.1 200 OK (JPEG JFIF image)
85	2.737381832	172.16.10.2	172.16.10.1	HTTP	359	GET /lion-wild-africa-african.jpg HTTP/1.1
92	2.737840608	172.16.10.2	172.16.10.1	HTTP	341	GET /images.jpg HTTP/1.1
101	2.738335480	172.16.10.2	172.16.10.1	HTTP	343	GET /download.jpg HTTP/1.1
119	2.738809142	172.16.10.1	172.16.10.2	HTTP	1226	HTTP/1.1 200 OK (JPEG JFIF image)
121	2.739075438	172.16.10.1	172.16.10.2	HTTP	1016	HTTP/1.1 200 OK (JPEG JFIF image)
139	2.740900738	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(1).jpg HTTP/1.1
143	2.741014891	172.16.10.2	172.16.10.1	HTTP	362	GET /soap-bubble-1958650_960_720.jpg HTTP/1.1
148	2.741205777	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(2).jpg HTTP/1.1
179	2.742807473	172.16.10.1	172.16.10.2	HTTP	113	HTTP/1.1 200 OK (JPEG JFIF image)
190	2.743723330	172.16.10.1	172.16.10.2	HTTP	712	HTTP/1.1 200 OK (JPEG JFIF image)
402	2.764054977	172.16.10.1	172.16.10.2	HTTP	72	HTTP/1.1 200 OK (JPEG JFIF image)
3774	3.042252027	172.16.10.1	172.16.10.2	HTTP	1124	HTTP/1.1 200 OK (JPEG JFIF image)

Here it is 3.042252027 - 2.685888334 = 0.356363

Step 3: For 4 persistent connections, Set the value of **max-persistent-connection-per-server to 4** in the client computer.

Step 4: Repeat the **steps 1-3** in the previous section.

http

No.	Time	Source	Destination	Protocol	Length	Info
28	0.152642908	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
30	1.667969551	172.16.10.2	172.16.10.1	HTTP	421	GET /a.html HTTP/1.1
31	1.668311781	172.16.10.1	172.16.10.2	HTTP	641	HTTP/1.1 200 OK (text/html)
33	1.699473631	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(1).jpg HTTP/1.1
35	1.699692009	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(2).jpg HTTP/1.1
45	1.699908042	172.16.10.1	172.16.10.2	HTTP	463	HTTP/1.1 200 OK (JPEG JFIF image)
46	1.699913003	172.16.10.1	172.16.10.2	HTTP	1200	HTTP/1.1 200 OK (JPEG JFIF image)
47	1.700012712	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(4).jpg HTTP/1.1
63	1.700901747	172.16.10.1	172.16.10.2	HTTP	242	HTTP/1.1 200 OK (JPEG JFIF image)
69	1.701341018	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(3).jpg HTTP/1.1
70	1.701432635	172.16.10.2	172.16.10.1	HTTP	359	GET /lion-wild-africa-african.jpg HTTP/1.1
86	1.701888908	172.16.10.1	172.16.10.2	HTTP	565	HTTP/1.1 200 OK (JPEG JFIF image)
93	1.702192885	172.16.10.2	172.16.10.1	HTTP	341	GET /images.jpg HTTP/1.1
95	1.702219175	172.16.10.2	172.16.10.1	HTTP	343	GET /download.jpg HTTP/1.1
97	1.702228220	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(1).jpg HTTP/1.1
98	1.702233130	172.16.10.2	172.16.10.1	HTTP	362	GET /soap-bubble-1958650_960_720.jpg HTTP/1.1
122	1.703328136	172.16.10.1	172.16.10.2	HTTP	711	HTTP/1.1 200 OK (JPEG JFIF image)
126	1.703773424	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(2).jpg HTTP/1.1
157	1.705498971	172.16.10.1	172.16.10.2	HTTP	1227	HTTP/1.1 200 OK (JPEG JFIF image)
159	1.705614894	172.16.10.1	172.16.10.2	HTTP	113	HTTP/1.1 200 OK (JPEG JFIF image)
167	1.706637782	172.16.10.1	172.16.10.2	HTTP	1017	HTTP/1.1 200 OK (JPEG JFIF image)
414	1.724541388	172.16.10.1	172.16.10.2	HTTP	73	HTTP/1.1 200 OK (JPEG JFIF image)
3825	2.005934395	172.16.10.1	172.16.10.2	HTTP	1124	HTTP/1.1 200 OK (JPEG JFIF image)

Here is it 2.005934395 - 1.667969557 = 0.337964838

Step 5: For 6 persistent connections, set the value of **max-persistent-connection-per-server to 6** in the server computer.

Step 6: Repeat the **steps 1-3** in the previous section.

http

No.	Time	Source	Destination	Protocol	Length	Info
21	0.100232302	172.16.10.2	172.16.10.1	HTTP	306	GET /favicon.ico HTTP/1.1
22	0.100476138	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
24	0.184514911	172.16.10.2	172.16.10.1	HTTP	366	GET /favicon.ico HTTP/1.1
25	0.184789474	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
27	3.915242469	172.16.10.2	172.16.10.1	HTTP	421	GET /a.html HTTP/1.1
28	3.915930950	172.16.10.1	172.16.10.2	HTTP	641	HTTP/1.1 200 OK (text/html)
30	3.934519286	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(1).jpg HTTP/1.1
31	3.934703623	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(2).jpg HTTP/1.1
44	3.935084209	172.16.10.1	172.16.10.2	HTTP	1200	HTTP/1.1 200 OK (JPEG JFIF image)
45	3.935091751	172.16.10.1	172.16.10.2	HTTP	463	HTTP/1.1 200 OK (JPEG JFIF image)
50	3.935485109	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(4).jpg HTTP/1.1
68	3.936344013	172.16.10.1	172.16.10.2	HTTP	243	HTTP/1.1 200 OK (JPEG JFIF image)
74	3.936634551	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(3).jpg HTTP/1.1
75	3.936649737	172.16.10.2	172.16.10.1	HTTP	359	GET /lion-wild-africa-african.jpg HTTP/1.1
76	3.936654620	172.16.10.2	172.16.10.1	HTTP	341	GET /images.jpg HTTP/1.1
78	3.936684823	172.16.10.2	172.16.10.1	HTTP	343	GET /download.jpg HTTP/1.1
80	3.936696984	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(1).jpg HTTP/1.1
122	3.937371850	172.16.10.2	172.16.10.1	HTTP	362	GET /soap-bubble-1958650_960_720.jpg HTTP/1.1
123	3.937539442	172.16.10.1	172.16.10.2	HTTP	1227	HTTP/1.1 200 OK (JPEG JFIF image)
160	3.939256627	172.16.10.1	172.16.10.2	HTTP	1017	HTTP/1.1 200 OK (JPEG JFIF image)
167	3.940125154	172.16.10.1	172.16.10.2	HTTP	712	HTTP/1.1 200 OK (JPEG JFIF image)
183	3.941778538	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(2).jpg HTTP/1.1
229	3.946434434	172.16.10.1	172.16.10.2	HTTP	565	HTTP/1.1 200 OK (JPEG JFIF image)
233	3.946891865	172.16.10.1	172.16.10.2	HTTP	113	HTTP/1.1 200 OK (JPEG JFIF image)
441	3.964535410	172.16.10.1	172.16.10.2	HTTP	72	HTTP/1.1 200 OK (JPEG JFIF image)
3771	4.241013689	172.16.10.1	172.16.10.2	HTTP	1124	HTTP/1.1 200 OK (JPEG JFIF image)

Here it is 4.241013689 - 3.915242469 = 0.325771229

Step 7: For 10 persistent connections, set the value of **max-persistent-connection-per-server** **to** **10** in the client computer.

Step 8: Repeat the **steps 1-3** in the previous section.

http

No.	Time	Source	Destination	Protocol	Length	Info
27	0.192665375	172.16.10.1	172.16.10.2	HTTP	568	HTTP/1.1 404 Not Found (text/html)
29	1.556964626	172.16.10.2	172.16.10.1	HTTP	421	GET /a.html HTTP/1.1
30	1.557214715	172.16.10.1	172.16.10.2	HTTP	641	HTTP/1.1 200 OK (text/html)
32	1.575716934	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(1).jpg HTTP/1.1
33	1.575953704	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(2).jpg HTTP/1.1
46	1.576334520	172.16.10.1	172.16.10.2	HTTP	1200	HTTP/1.1 200 OK (JPEG JFIF image)
47	1.576343533	172.16.10.1	172.16.10.2	HTTP	463	HTTP/1.1 200 OK (JPEG JFIF image)
52	1.576760416	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(4).jpg HTTP/1.1
70	1.577515601	172.16.10.1	172.16.10.2	HTTP	243	HTTP/1.1 200 OK (JPEG JFIF image)
76	1.577834686	172.16.10.2	172.16.10.1	HTTP	347	GET /images%20(3).jpg HTTP/1.1
77	1.577847379	172.16.10.2	172.16.10.1	HTTP	359	GET /lion-wild-africa-african.jpg HTTP/1.1
78	1.577855269	172.16.10.2	172.16.10.1	HTTP	341	GET /images.jpg HTTP/1.1
80	1.577886802	172.16.10.2	172.16.10.1	HTTP	343	GET /download.jpg HTTP/1.1
82	1.577905312	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(1).jpg HTTP/1.1
118	1.578606528	172.16.10.2	172.16.10.1	HTTP	362	GET /soap-bubble-1958650_960_720.jpg HTTP/1.1
119	1.578639337	172.16.10.1	172.16.10.2	HTTP	1227	HTTP/1.1 200 OK (JPEG JFIF image)
146	1.580341669	172.16.10.1	172.16.10.2	HTTP	712	HTTP/1.1 200 OK (JPEG JFIF image)
169	1.582240704	172.16.10.2	172.16.10.1	HTTP	349	GET /download%20(2).jpg HTTP/1.1
187	1.583749770	172.16.10.1	172.16.10.2	HTTP	1017	HTTP/1.1 200 OK (JPEG JFIF image)
219	1.586862673	172.16.10.1	172.16.10.2	HTTP	113	HTTP/1.1 200 OK (JPEG JFIF image)
222	1.587108849	172.16.10.1	172.16.10.2	HTTP	565	HTTP/1.1 200 OK (JPEG JFIF image)
455	1.606226568	172.16.10.1	172.16.10.2	HTTP	72	HTTP/1.1 200 OK (JPEG JFIF image)
3814	1.882459413	172.16.10.1	172.16.10.2	HTTP	1124	HTTP/1.1 200 OK (JPEG JFIF image)

Here it is 1.882459413-1.556964626=0.325494787

OBSERVATIONS REQUIRED ON EDMODO:

Find out the time taken to load images for 2 4 6 persistent connections is lesser or greater than 10 persistent compared to non-persistent. Why? Find out the optimal persistent connections.

SCREENSHOTS REQUIRED FOR EDMODO:

1) Non-persistent connection wireshark capture (should include all 10 images)
2) Persistent connections wireshark capture – 2, 4, 6, 8 & 10 respectively (should include all 10 images).

Computer Networks Lab – UE18CS304

Ruchika Shashidhara – PES1201800046 – Sem 5 Sec D – Roll Num 2

Week #2 – Understanding Persistent and Non-persistent HTTP Connections - 15 Sept 2020

OBJECTIVE: To understand persistent and non-persistent HTTP connections and corresponding performance impact.

SETUP:

172.16.10.2/24 **172.16.10.3/24**

Server (ubuntu1) Virtual Machine's IP Address: **172.16.10.2** (Roll num: 2)

```
ubuntu1@ubuntu1-VirtualBox:~$ sudo ip addr flush dev enp0s3
[sudo] password for ubuntu1:
ubuntu1@ubuntu1-VirtualBox:~$ sudo ip addr add 172.16.10.2/24 dev enp0s3
ubuntu1@ubuntu1-VirtualBox:~$ sudo ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.10.2  netmask 255.255.255.0  broadcast 0.0.0.0
        ether 08:00:27:e0:93:48  txqueuelen 1000  (Ethernet)
        RX packets 55533  bytes 4903725 (4.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 501588  bytes 753701023 (753.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2105  bytes 162779 (162.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2105  bytes 162779 (162.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Also created a.html and embedded 10 images (i0.jpg – i9.jpg) of sizes 1MB-2MB inside /var/www/html

Client (ubuntu2) Virtual Machine's IP Address: **172.16.10.3**

```
ubuntu2@ubuntu2-VirtualBox:~$ sudo ip addr flush dev enp0s3
[sudo] password for ubuntu2:
ubuntu2@ubuntu2-VirtualBox:~$ sudo ip addr add 172.16.10.3/24 dev enp0s3
ubuntu2@ubuntu2-VirtualBox:~$ sudo ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.10.3  netmask 255.255.255.0  broadcast 0.0.0.0
        ether 08:00:27:71:24:5d  txqueuelen 1000  (Ethernet)
        RX packets 502702  bytes 753480379 (753.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57040  bytes 4453060 (4.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 10439  bytes 891875 (891.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10439  bytes 891875 (891.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

OBSERVATIONS: (on ubuntu2 – Client Machine)

No.	Time	Source	Destination	Protocol	Length	Info
98	41.732895050	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
100	41.733754192	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
102	42.679535267	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
244	42.782118395	172.16.10.2	172.16.10.3	HTTP	1982	HTTP/1.1 200 OK (JPEG JFIF image)
247	42.882044800	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
365	42.970847192	172.16.10.2	172.16.10.3	HTTP	1967	HTTP/1.1 200 OK (JPEG JFIF image)
372	43.014830445	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
566	43.052045745	172.16.10.2	172.16.10.3	HTTP	32462	HTTP/1.1 200 OK (JPEG JFIF image)
568	43.112091433	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
672	43.157143627	172.16.10.2	172.16.10.3	HTTP	119	HTTP/1.1 200 OK (JPEG JFIF image)
674	43.194795932	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
754	43.225463682	172.16.10.2	172.16.10.3	HTTP	17119	HTTP/1.1 200 OK (JPEG JFIF image)
761	43.236135765	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
1052	43.393745895	172.16.10.2	172.16.10.3	HTTP	10186	HTTP/1.1 200 OK (JPEG JFIF image)
1054	43.453839316	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
1180	43.518934304	172.16.10.2	172.16.10.3	HTTP	1275	HTTP/1.1 200 OK (JPEG JFIF image)
1182	43.528626272	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
1281	43.611561922	172.16.10.2	172.16.10.3	HTTP	21243	HTTP/1.1 200 OK (JPEG JFIF image)
1288	43.731075172	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
1461	43.748071183	172.16.10.2	172.16.10.3	HTTP	18001	HTTP/1.1 200 OK (JPEG JFIF image)
1463	43.810164236	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
1570	43.897800555	172.16.10.2	172.16.10.3	HTTP	22968	HTTP/1.1 200 OK (JPEG JFIF image)

Non Persistent Connection: Time taken to load: 43.897800555 – 41.732895050 = **2.164905505 s**

No.	Time	Source	Destination	Protocol	Length	Info
148	83.627666438	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
150	83.628285160	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
154	84.291002625	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
330	84.302423276	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
617	84.351718637	172.16.10.2	172.16.10.3	HTTP	12138	HTTP/1.1 200 OK (JPEG JFIF image)
663	84.365631453	172.16.10.2	172.16.10.3	HTTP	3430	HTTP/1.1 200 OK (JPEG JFIF image)
671	84.407357534	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
809	84.446330226	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
905	84.462761767	172.16.10.2	172.16.10.3	HTTP	4915	HTTP/1.1 200 OK (JPEG JFIF image)
947	84.489038262	172.16.10.2	172.16.10.3	HTTP	46455	HTTP/1.1 200 OK (JPEG JFIF image)
954	84.507929771	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
1207	84.574079111	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
1280	84.606440496	172.16.10.2	172.16.10.3	HTTP	7018	HTTP/1.1 200 OK (JPEG JFIF image)
1286	84.638244412	172.16.10.2	172.16.10.3	HTTP	33319	HTTP/1.1 200 OK (JPEG JFIF image)
1289	84.810500530	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
1426	84.915940905	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
1623	84.937034331	172.16.10.2	172.16.10.3	HTTP	7067	HTTP/1.1 200 OK (JPEG JFIF image)
1636	84.995707794	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
1747	85.014471798	172.16.10.2	172.16.10.3	HTTP	3902	HTTP/1.1 200 OK (JPEG JFIF image)
1755	85.046294715	172.16.10.2	172.16.10.3	HTTP	7830	HTTP/1.1 200 OK (JPEG JFIF image)
1765	85.144092341	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
1968	85.154629143	172.16.10.2	172.16.10.3	HTTP	20072	HTTP/1.1 200 OK (JPEG JFIF image)

2 Persistent Connections: Time taken to load: 85.154629143 – 83.627666438 = **1.526962705 s**

No.	Time	Source	Destination	Protocol	Length	Info
62	25.763645499	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
64	25.764231526	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
67	26.287966059	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
302	26.325931747	172.16.10.2	172.16.10.3	HTTP	17910	HTTP/1.1 200 OK (JPEG JFIF image)
313	26.334848115	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
314	26.334960764	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
315	26.335052757	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
951	26.430137730	172.16.10.2	172.16.10.3	HTTP	119	HTTP/1.1 200 OK (JPEG JFIF image)
1035	26.462158185	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
1120	26.467860987	172.16.10.2	172.16.10.3	HTTP	16534	HTTP/1.1 200 OK (JPEG JFIF image)
1174	26.483418038	172.16.10.2	172.16.10.3	HTTP	2002	HTTP/1.1 200 OK (JPEG JFIF image)
1178	26.492499270	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
1250	26.502956202	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
1347	26.525655067	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
1528	26.556040525	172.16.10.2	172.16.10.3	HTTP	2639	HTTP/1.1 200 OK (JPEG JFIF image)
1574	26.588543492	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
1706	26.612152620	172.16.10.2	172.16.10.3	HTTP	10186	HTTP/1.1 200 OK (JPEG JFIF image)
1789	26.631261129	172.16.10.2	172.16.10.3	HTTP	2454	HTTP/1.1 200 OK (JPEG JFIF image)
1791	26.647881345	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
1855	26.654655451	172.16.10.2	172.16.10.3	HTTP	2723	HTTP/1.1 200 OK (JPEG JFIF image)
1928	26.693786911	172.16.10.2	172.16.10.3	HTTP	625	HTTP/1.1 200 OK (JPEG JFIF image)
2146	26.880392679	172.16.10.2	172.16.10.3	HTTP	14245	HTTP/1.1 200 OK (JPEG JFIF image)

4 Persistent Connections: Time taken to load: 26.880392679 – 25.763645499 = **1.116747180 s**

No.	Time	Source	Destination	Protocol	Length	Info
10	7.585200389	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
12	7.585896120	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
14	7.809981945	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
188	7.846341439	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
336	7.873518493	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
662	7.957708903	172.16.10.2	172.16.10.3	HTTP	2054	HTTP/1.1 200 OK (JPEG JFIF image)
673	7.958705019	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
761	7.994385742	172.16.10.2	172.16.10.3	HTTP	7774	HTTP/1.1 200 OK (JPEG JFIF image)
870	8.024733910	172.16.10.2	172.16.10.3	HTTP	4898	HTTP/1.1 200 OK (JPEG JFIF image)
1114	8.090755960	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
1115	8.090922642	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
1522	8.135810312	172.16.10.2	172.16.10.3	HTTP	5570	HTTP/1.1 200 OK (JPEG JFIF image)
1727	8.172716230	172.16.10.2	172.16.10.3	HTTP	5842	HTTP/1.1 200 OK (JPEG JFIF image)
1729	8.177259166	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
1807	8.198854752	172.16.10.2	172.16.10.3	HTTP	1567	HTTP/1.1 200 OK (JPEG JFIF image)
1822	8.217686839	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
1916	8.239903468	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
2095	8.314376397	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
2176	8.337697731	172.16.10.2	172.16.10.3	HTTP	3521	HTTP/1.1 200 OK (JPEG JFIF image)
2184	8.337997839	172.16.10.2	172.16.10.3	HTTP	4136	HTTP/1.1 200 OK (JPEG JFIF image)
2277	8.442504756	172.16.10.2	172.16.10.3	HTTP	3902	HTTP/1.1 200 OK (JPEG JFIF image)
2311	8.494466657	172.16.10.2	172.16.10.3	HTTP	5592	HTTP/1.1 200 OK (JPEG JFIF image)

6 Persistent Connections: Time taken to Load: 8.494466657 – 7.585200389 = **0.909266268 s**

No.	Time	Source	Destination	Protocol	Length	Info
75	44.122493358	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
77	44.123112338	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
81	44.686497823	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
257	44.722876896	172.16.10.2	172.16.10.3	HTTP	7774	HTTP/1.1 200 OK (JPEG JFIF image)
282	44.733866575	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
283	44.733948554	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
284	44.734004696	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
285	44.734057812	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
286	44.734109536	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
291	44.734239503	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
293	44.734324854	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
912	44.823449245	172.16.10.2	172.16.10.3	HTTP	13082	HTTP/1.1 200 OK (JPEG JFIF image)
1692	44.881971337	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
1855	44.915723697	172.16.10.2	172.16.10.3	HTTP	2002	HTTP/1.1 200 OK (JPEG JFIF image)
1901	44.921203645	172.16.10.2	172.16.10.3	HTTP	7359	HTTP/1.1 200 OK (JPEG JFIF image)
1908	44.921334519	172.16.10.2	172.16.10.3	HTTP	28518	HTTP/1.1 200 OK (JPEG JFIF image)
2059	44.951053103	172.16.10.2	172.16.10.3	HTTP	11362	HTTP/1.1 200 OK (JPEG JFIF image)
2434	45.028987299	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
2603	45.044336519	172.16.10.2	172.16.10.3	HTTP	4171	HTTP/1.1 200 OK (JPEG JFIF image)
2698	45.103625329	172.16.10.2	172.16.10.3	HTTP	606	HTTP/1.1 200 OK (JPEG JFIF image)
2719	45.104213751	172.16.10.2	172.16.10.3	HTTP	4934	HTTP/1.1 200 OK (JPEG JFIF image)
2902	45.382878598	172.16.10.2	172.16.10.3	HTTP	2696	HTTP/1.1 200 OK (JPEG JFIF image)

8 Persistent Connections: Time taken to Load: 45.382878598 – 44.122493358 = **1.260385240 s**

No.	Time	Source	Destination	Protocol	Length	Info
45	50.968802453	172.16.10.3	172.16.10.2	HTTP	404	GET /a.html HTTP/1.1
47	50.969658557	172.16.10.2	172.16.10.3	HTTP	588	HTTP/1.1 200 OK (text/html)
52	52.185966428	172.16.10.3	172.16.10.2	HTTP	350	GET /i0.jpg HTTP/1.1
181	52.202976783	172.16.10.3	172.16.10.2	HTTP	350	GET /i1.jpg HTTP/1.1
190	52.203872404	172.16.10.3	172.16.10.2	HTTP	350	GET /i2.jpg HTTP/1.1
546	52.247967393	172.16.10.2	172.16.10.3	HTTP	44046	HTTP/1.1 200 OK (JPEG JFIF image)
549	52.248374972	172.16.10.3	172.16.10.2	HTTP	350	GET /i3.jpg HTTP/1.1
730	52.304662523	172.16.10.2	172.16.10.3	HTTP	534	HTTP/1.1 200 OK (JPEG JFIF image)
1021	52.390081003	172.16.10.2	172.16.10.3	HTTP	554	HTTP/1.1 200 OK (JPEG JFIF image)
1538	52.674803836	172.16.10.2	172.16.10.3	HTTP	7359	HTTP/1.1 200 OK (JPEG JFIF image)
1562	52.768787309	172.16.10.3	172.16.10.2	HTTP	350	GET /i4.jpg HTTP/1.1
1706	52.811661949	172.16.10.3	172.16.10.2	HTTP	350	GET /i5.jpg HTTP/1.1
1834	52.868204052	172.16.10.3	172.16.10.2	HTTP	350	GET /i9.jpg HTTP/1.1
1839	52.868464708	172.16.10.3	172.16.10.2	HTTP	350	GET /i8.jpg HTTP/1.1
1845	52.868722659	172.16.10.3	172.16.10.2	HTTP	350	GET /i7.jpg HTTP/1.1
1846	52.868846986	172.16.10.3	172.16.10.2	HTTP	350	GET /i6.jpg HTTP/1.1
2202	52.983807471	172.16.10.2	172.16.10.3	HTTP	23183	HTTP/1.1 200 OK (JPEG JFIF image)
2588	53.161894990	172.16.10.2	172.16.10.3	HTTP	2674	HTTP/1.1 200 OK (JPEG JFIF image)
3004	53.400482749	172.16.10.2	172.16.10.3	HTTP	7040	HTTP/1.1 200 OK (JPEG JFIF image)
3498	53.599624461	172.16.10.2	172.16.10.3	HTTP	6417	HTTP/1.1 200 OK (JPEG JFIF image)
4246	53.819496257	172.16.10.2	172.16.10.3	HTTP	6798	HTTP/1.1 200 OK (JPEG JFIF image)
4302	53.822776975	172.16.10.2	172.16.10.3	HTTP	5619	HTTP/1.1 200 OK (JPEG JFIF image)

10 Persistent Connections: Time taken to Load: 53.822776975 – 50.968802453 = **2.853974522 s**

CONCLUSION:

- Time taken to load images for 2,4,6 persistent connections is **lesser than** 10 persistent and non-persistent connections.
- Optimal number of persistent connections is **6** for best performance in Firefox browser.
- Time taken to load decreases from 2 to 6 persistent connections and then increases for 8, 10 persistent connections and non-persistent connection.

Reasons for the above trend:

A **persistent connection** remains open for a period of time, and can be reused for several requests, saving the need for a new TCP handshake, and utilizing TCP's performance enhancing capabilities. This connection stays open until the time set by Keep-Alive header in its configuration. So here, we can have multiple requests one after the other through the same channel in a single TCP connection.

A **non-persistent connection** establishes a connection with the sever for each image request. Hence it adds up time for each round-trip time-delay. So here, for each multiple request, it has to go through multiple channels with multiple TCP connections.

Browsers limit the number of HTTP connections with the same domain name. This restriction in the Firefox Browser's HTTP specification allow only 6 connections per domain. So, if the number of connections increases, the browser limit is exceeded and existing connections are closed to allow new ones to open. Hence, we got **6 optimal number of persistent connections** for Firefox browser.

Week: #3 Understand working of HTTP Headers

	Understand working of HTTP headers: Conditional Get: If-Modified-Since HTTP Cookies: Cookie and Set-Cookie Authentication: Auth-Basic Design a web page that has one embedded page (e.g. image) and sets a cookie and enables authentication. You are required to configure the web server (e.g. apache) with authentication mechanism. Show the behavior of conditional get when embedded objects is modified and when it is not (you can just change the create date of the embedded object). Decode the Basic-Auth header using Base64 mechanism as per the password setup. **Observation**: Show the behavior of browser when is cookie is set and when cookie is removed.

Week: 5

Understanding Working of HTTP Headers

Question: Understand working of HTTP headers

Conditional Get: If-Modified-Since

HTTP Cookies: Cookie and Set-Cookie

Authentication: Auth-Basic

Design a web page that has one embedded page (e.g. image) and sets a cookie and enables authentication. You are required to configure the web server (e.g. apache) with authentication mechanism. Show the behavior of conditional get when embedded objects are modified and when it is not (you can just change the create date of the embedded object). Decode the Basic-Auth header using Base64 mechanism as per the password setup.

Observation: Show the behavior of browser when is cookie is set and when cookie is removed.

Solution: Analyzing Basic Authentication and Cookies

The three parts of experiment are:

1. Password Authentication
2. Cookie Setting
3. Conditional get

Steps of Execution (for Password Authentication)

1. Executing the below commands on the terminal.

--> To update and integrate the existing softwares

sudo apt-get update

--> To install the apache utility

sudo apt-get install apache2 apache2-utils

```
osboxes@osboxes: ~
osboxes@osboxes:~$ sudo apt-get install apache2-utils
[sudo] password for osboxes:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  apache2-utils
1 upgraded, 0 newly installed, 0 to remove and 247 not upgraded.
Need to get 81.7 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 apache2-utils amd64 2.4.18-2ub
untu3.10 [81.7 kB]
Fetched 81.7 kB in 1s (45.5 kB/s)
(Reading database ... 217540 files and directories currently installed.)
Preparing to unpack .../apache2-utils_2.4.18-2ubuntu3.10_amd64.deb ...
Unpacking apache2-utils (2.4.18-2ubuntu3.10) over (2.4.18-2ubuntu3.9) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up apache2-utils (2.4.18-2ubuntu3.10) ...
osboxes@osboxes:~$
```

--> Provide username and password to set authentication

sudo htpasswd -c /etc/apache2/.htpasswd ANY_USERNAME

```
osboxes@osboxes:~$ sudo htpasswd -c /etc/apache2/.htpasswd netwo
New password:
Re-type new password:
Adding password for user netwo
osboxes@osboxes:~$ sudo cat /etc/apache2/.htpasswd
netwo:$apr1$6YdDa0Ti$ELrUaOlQ/jun9TTU1PYKu/
osboxes@osboxes:~$
```

Here “netwo” is the username. Also, password is entered twice.

--> View the authentication

sudo cat /etc/apache2/.htpasswd

2. To setup the authentication phase, execute the following commands. Configuring Access control within the Virtual Host Definition.

--> Opening the file for setting authentication

sudo nano /etc/apache2/sites-available/000-default.conf

```
<VirtualHost*:80>
        ServerAdmin webmaster@localhost
        DocumentRoot  /var/www/html
        ErrorLog ${APACHE_LOG_DIR}error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        <Directory "/var/www/html">
                AuthType Basic
                AuthName "RESTRICTED"
                AuthUserFile /etc/apache2/.htpasswd
                Require valid-user >
        </Directory>
</VirtualHost>
```

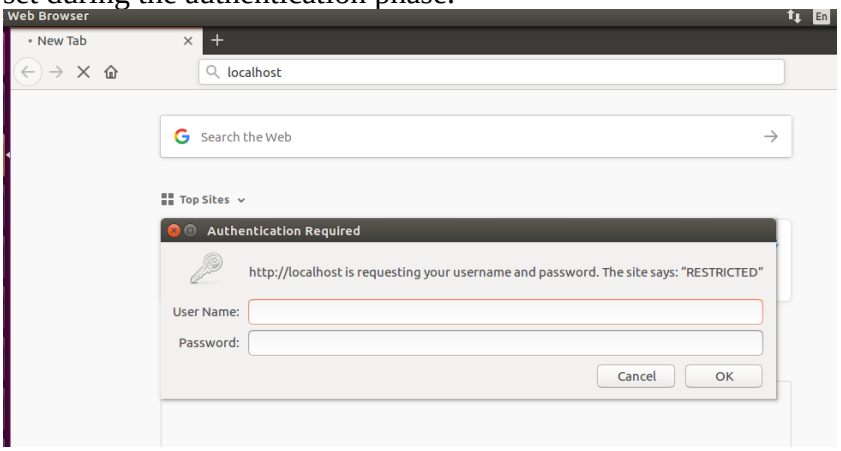
```
 GNU nano 2.5.3        File: /etc/apache2/sites-available/000-default.conf

<VirtualHost *:80>


        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

                <Directory "/var/www/html">
                        AuthType Basic
                        AuthName "RESTRICTED"
                        AuthUserFile /etc/apache2/.htpasswd
                        Require valid-user >
        </Directory>

</VirtualHost>
```

3. Password policy implementation is done by restarting the server as:

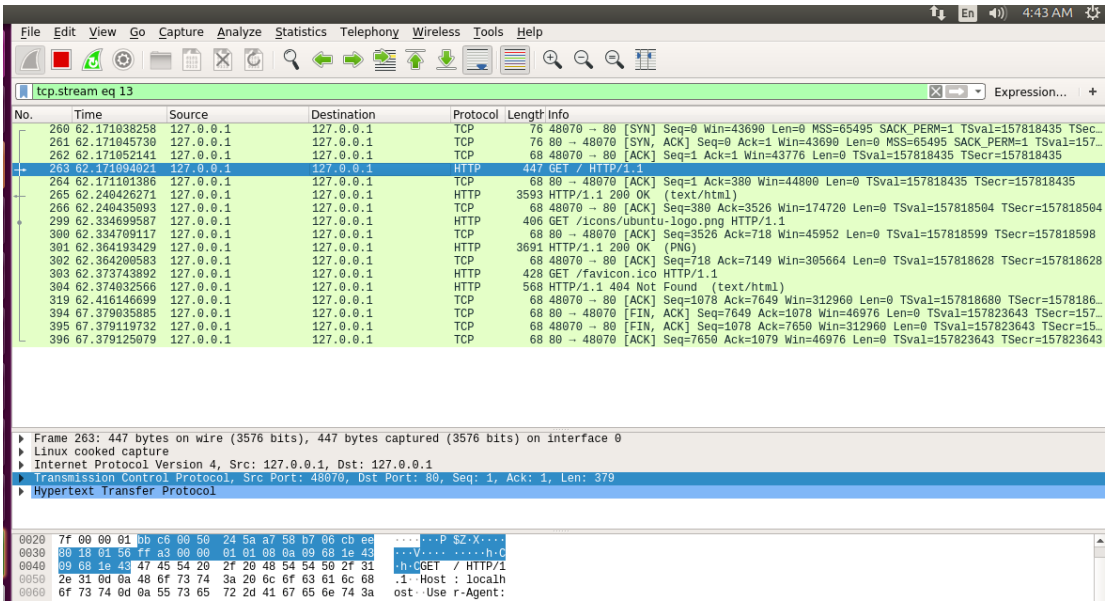
sudo service apache2 restart

```
osboxes@osboxes:~$ sudo service apache2 restart
osboxes@osboxes:~$
```

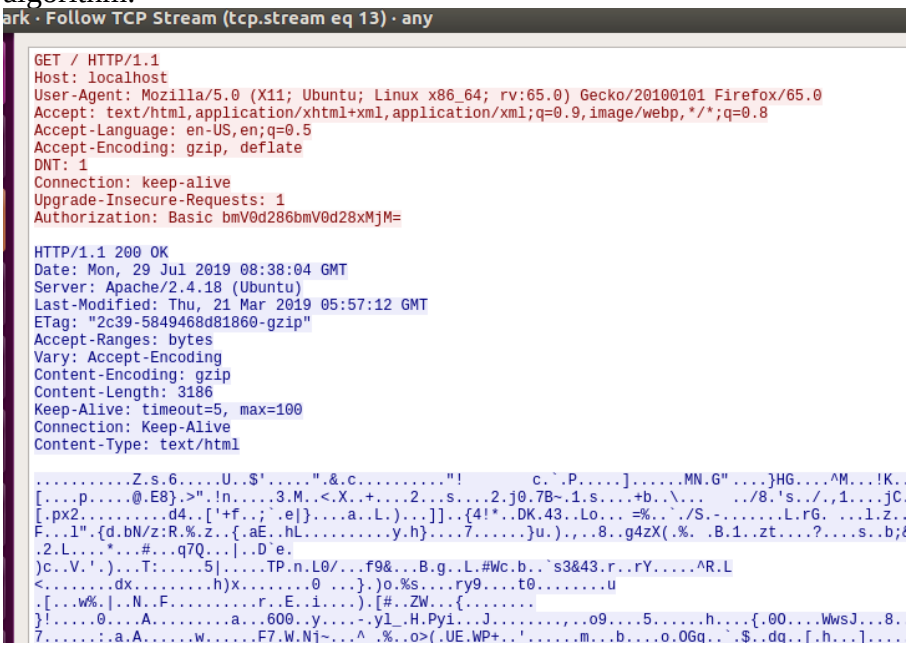
4. The localhost is then accessed using the Firefox browser requiring a username and a password set during the authentication phase.



5. Wireshark is used to capture the packets sent upon the network.



6. Using the “follow TCP stream” on the HTTP message segment the password was retrieved which was encrypted by the base64 algorithm and decryption could be done with same algorithm.



Steps of Execution (Cookie Setting)

1. A PHP file to set the cookie is created which also contains an image in it (placed under the HTML directory) to be accessed once the cookie is set. The following code helped to set the cookie:

<html>
<?php

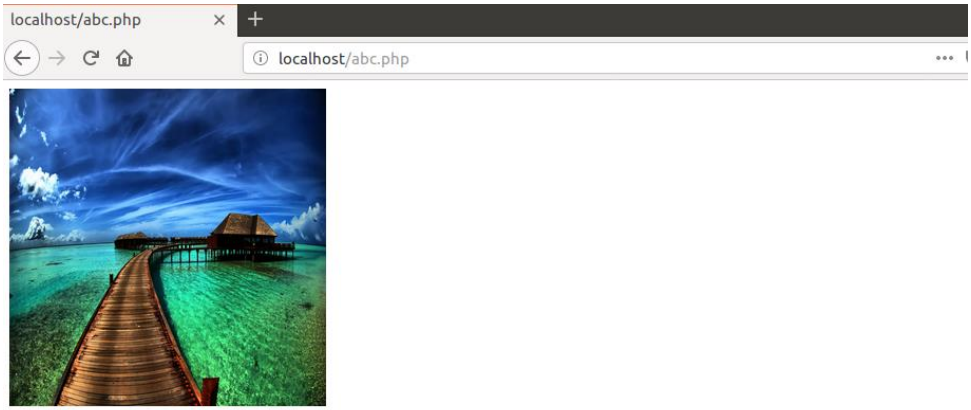
```
        setcookie("namecookie","netqwerty",time()+123);
        setcookie("nickname","work");
?>
<img src= “highres.png” width= “300” height= “300” title= “password” />
</html>
```

```
 GNU nano 2.5.3                File: abc.php

<html>
<?php
setcookie("namecookie","netqwerty",time()+123);
setcookie("nickname","work");
?>
<img src="highres.jpg" width="300" height="300" title="password"/>
</html>
```

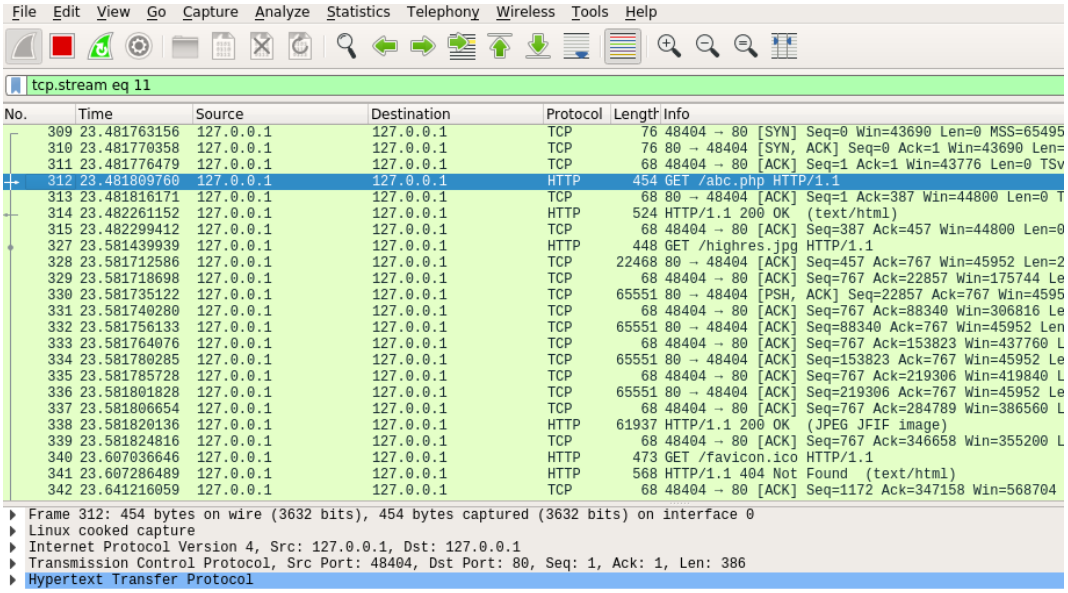
Note: Here you can add any image if required

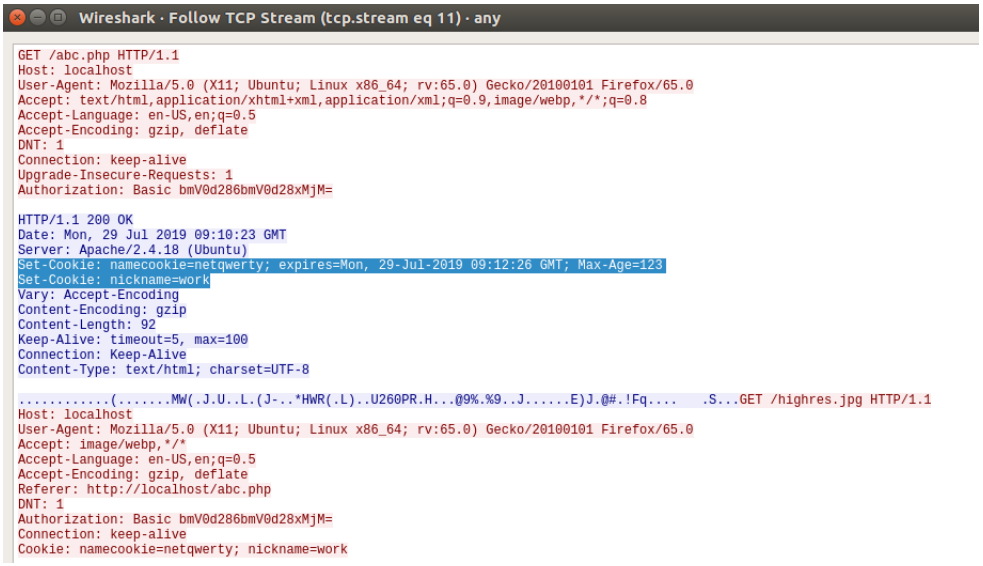
Note: You can capture Cookies mostly during the first time of web access. Hence keep wireshark capture ready before executing the task for the first time.

2. The combined file saved with a .php extension is placed under **/var/www/html** for accessing.



3. The packets are captured using Wireshark and using the “follow TCP stream” which checks for the set-cookie field whether the cookie is set or not set.





The cookie is set as shown in the above screenshot.

Observation: Understand and work out base 64 algorithm and write in your observation. Observe various parameters associated with Cookie in the wireshark capture.

Conditional Get: If-Modified-Since

Before performing the steps below, make sure your browser's cache is empty. (To do this under Firefox, select Tools -> Clear Recent History and check the Cache box). Now do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
- Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Observations:

- ✓ Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?
- ✓ Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
- ✓ Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
- ✓ What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Repeat the above task with some images on the server.

Attach screenshots wherever necessary.

Computer Networks Lab – UE18CS304

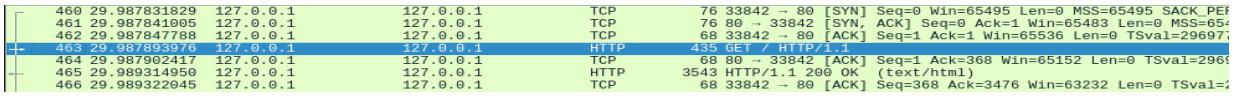
Ruchika Shashidhara – PES1201800046 – Sem 5 Sec D

Week #3 – Understanding Working of HTTP Headers – 22 Sept 2020

OBJECTIVE: Understand working of HTTP headers – Conditional Get, HTTP Cookies, Authentication

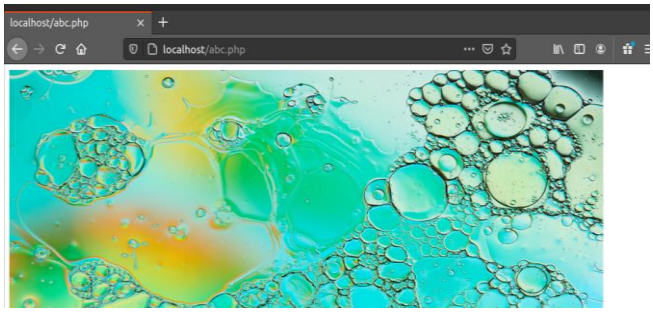
OBSERVATION:

1) Password Authentication: Auth-Basic



Authorization: Basic bmV0d286cXdlcnR5 is the encrytped username & password, explanation on page 3

2) Cookie Setting – HTTP Cookies and Set-Cookie




```
   125 24.477160170  127.0.0.1       127.0.0.1       TCP        76 33914 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PE
   126 24.477168436  127.0.0.1       127.0.0.1       TCP        76 80 → 33914 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65
   127 24.477175115  127.0.0.1       127.0.0.1       TCP        68 33914 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=29699
   128 24.477224204  127.0.0.1       127.0.0.1       HTTP      442 GET /abc.php HTTP/1.1
   129 24.477232624  127.0.0.1       127.0.0.1       TCP        68 80 → 33914 [ACK] Seq=1 Ack=375 Win=65152 Len=0 TSval=296
   130 24.478318373  127.0.0.1       127.0.0.1       HTTP      523 HTTP/1.1 200 OK  (text/html)
   131 24.478373764  127.0.0.1       127.0.0.1       TCP        68 33914 → 80 [ACK] Seq=375 Ack=456 Win=65152 Len=0 TSval=2
```

Wireshark · Follow TCP Stream (tcp.stream eq 13) · any

```
GET /abc.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Authorization: Basic bmV0d286cXdlcnR5

HTTP/1.1 200 OK
Date: Tue, 22 Sep 2020 05:18:23 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: namecookie=netqwerty; expires=Tue, 22-Sep-2020 05:20:26 GMT; Max-Age=123
Set-Cookie: nickname=work
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 93
Keep-Alive: timeout=5, max=2
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

............(.......MW(.J.U..MLO..*HWR(.L)..U.40PR.H.L.(.U2.qJ2KrRm.
......R.....C....V..P...GET /image.jpg HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic bmV0d286cXdlcnR5
Connection: keep-alive
Referer: http://localhost/abc.php
Cookie: namecookie=netqwerty; nickname=work

HTTP/1.1 200 OK
Date: Tue, 22 Sep 2020 05:18:24 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Tue, 22 Sep 2020 04:12:15 GMT
ETag: "3ba453-5afdf2c122ad4"
Accept-Ranges: bytes
Content-Length: 3908691
Keep-Alive: timeout=5, max=1
Connection: Keep-Alive
Content-Type: image/jpeg

......JFIF.....H.H......ICC_PROFILE.......lcms....mntrRGB XYZ .........).9acspAPPL.................................-
lcms....................................................
desc.......^cprt...
\....wtpt...h....bkpt...|....rXYZ........gXYZ........bXYZ........rTRC.......@gTRC.......@bTRC.......@desc........c2.................
.....................................................text....IX..XYZ ..............-XYZ ..........3....XYZ ......o...
8.....XYZ ......b.........XYZ ......$.........curv..............c...k...?.Q.4!.).2.;.F.Qw].kpz....|.i.}...
0.............................
```

Decrypting retrieved username & password using Base64 algorithm from:

Authorization: Basic bmV0d286cXdlcnR5

S1:	B	m	V	0	d	2	8	6	c	X	d	l	c	n	R	5
S2:	27	38	21	52	29	54	60	58	28	23	29	37	28	39	17	57

S3: 011011 100110 010101 110100 011101 110110 111100 111010 011100 010111 011101 100101 011100 100111 010001 111001

S4: 01101110 01100101 01110100 01110111 01101111 00111010 01110001 01110111 01100101 01110010 01110100 01111001

S5:	110	101	116	119	111	58	113	119	101	114	116	121
S6:	n	e	t	w	o	:	q	w	e	r	t	y

Decrypted Username-Password -> netwo:qwerty

netwo is the username and qwerty is the password

Base64 Decrypt Algorithm:

S1: Split each charcter of the encrypted string letter by letter

S2: Convert each Base64 charcater to its indicies using the Base64 Characters Table

S3: Convert each Base64 indicies from Binary to Decimal(8 digit) and remove the preceeding “00” in from each and combine them into one string of 96 digits (16 groups * 6 digits)

S4: Divide the above string into groups of 8 digits to get 12 binary numbers (96 digits / 8 digits)

S5: Convert each Binary number to Decimal

S6: Convert each Decimal number into its ASCII character and concaenate them

Parameters associated with Cookie in Wireshark Capture:

Set-Cookie: namecookie=netqwerty; expires = Tue, 22 Sep 2020, 05:20:25 GMT; Max-Age= 123

Cookie’s name, password: namecookie, netqwerty; Cookie set & expiry time stamp (123s)

Set-Cookie: nickname=work

Cookie’s name, password: nickname, work

3) Conditional-Get: If-Modified-Since

First HTTP GET Request

	Source	Destination	Protocol	Length	Info
85079	10.0.2.15	128.119.245.12	HTTP	432	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
76264	128.119.245.12	10.0.2.15	HTTP	786	HTTP/1.1 200 OK (text/html)
03161	10.0.2.15	128.119.245.12	HTTP	313	GET /favicon.ico HTTP/1.1
43584	128.119.245.12	10.0.2.15	HTTP	541	HTTP/1.1 404 Not Found (text/html)
32370	10.0.2.15	35.224.99.156	HTTP	143	GET / HTTP/1.1
43707	35.224.99.156	10.0.2.15	HTTP	204	HTTP/1.1 204 No Content
96240	10.0.2.15	128.119.245.12	HTTP	518	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
67859	128.119.245.12	10.0.2.15	HTTP	296	HTTP/1.1 304 Not Modified

```
▸ Frame 139: 432 bytes on wire (3456 bits), 432 bytes captured (3456 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 128.119.245.12
▸ Transmission Control Protocol, Src Port: 55228, Dst Port: 80, Seq: 1, Ack: 1, Len: 376
▾ Hypertext Transfer Protocol
  ▾ GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
    ▾ [Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n]
        [GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /wireshark-labs/HTTP-wireshark-file2.html
      Request Version: HTTP/1.1
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
    [HTTP request 1/1]
    [Response in frame: 141]
```

	Source	Destination	Protocol	Length	Info
85079	10.0.2.15	128.119.245.12	HTTP	432	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
76264	128.119.245.12	10.0.2.15	HTTP	786	HTTP/1.1 200 OK (text/html)
03161	10.0.2.15	128.119.245.12	HTTP	313	GET /favicon.ico HTTP/1.1
43584	128.119.245.12	10.0.2.15	HTTP	541	HTTP/1.1 404 Not Found (text/html)
32370	10.0.2.15	35.224.99.156	HTTP	143	GET / HTTP/1.1
43707	35.224.99.156	10.0.2.15	HTTP	204	HTTP/1.1 204 No Content
96240	10.0.2.15	128.119.245.12	HTTP	518	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
67859	128.119.245.12	10.0.2.15	HTTP	296	HTTP/1.1 304 Not Modified

```
▸ Frame 141: 786 bytes on wire (6288 bits), 786 bytes captured (6288 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 128.119.245.12, Dst: 10.0.2.15
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 55228, Seq: 1, Ack: 377, Len: 730
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 200 OK\r\n
    ▾ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        [HTTP/1.1 200 OK\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Date: Tue, 22 Sep 2020 05:53:12 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.10 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Tue, 22 Sep 2020 05:53:01 GMT\r\n
    ETag: "173-5afe094752865"\r\n
    Accept-Ranges: bytes\r\n
  ▸ Content-Length: 371\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.229991185 seconds]
    [Request in frame: 139]
    [Request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]
    File Data: 371 bytes
▾ Line-based text data: text/html (10 lines)
    \n
    <html>\n
    \n
    Congratulations again!  Now you've downloaded the file lab2-2.html. <br>\n
    This file's last modification date will not change.  <p>\n
    Thus  if you download this multiple times on your browser, a complete copy <br>\n
    will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
    field in your browser's HTTP GET request to the server.\n
    \n
    </html>\n
```

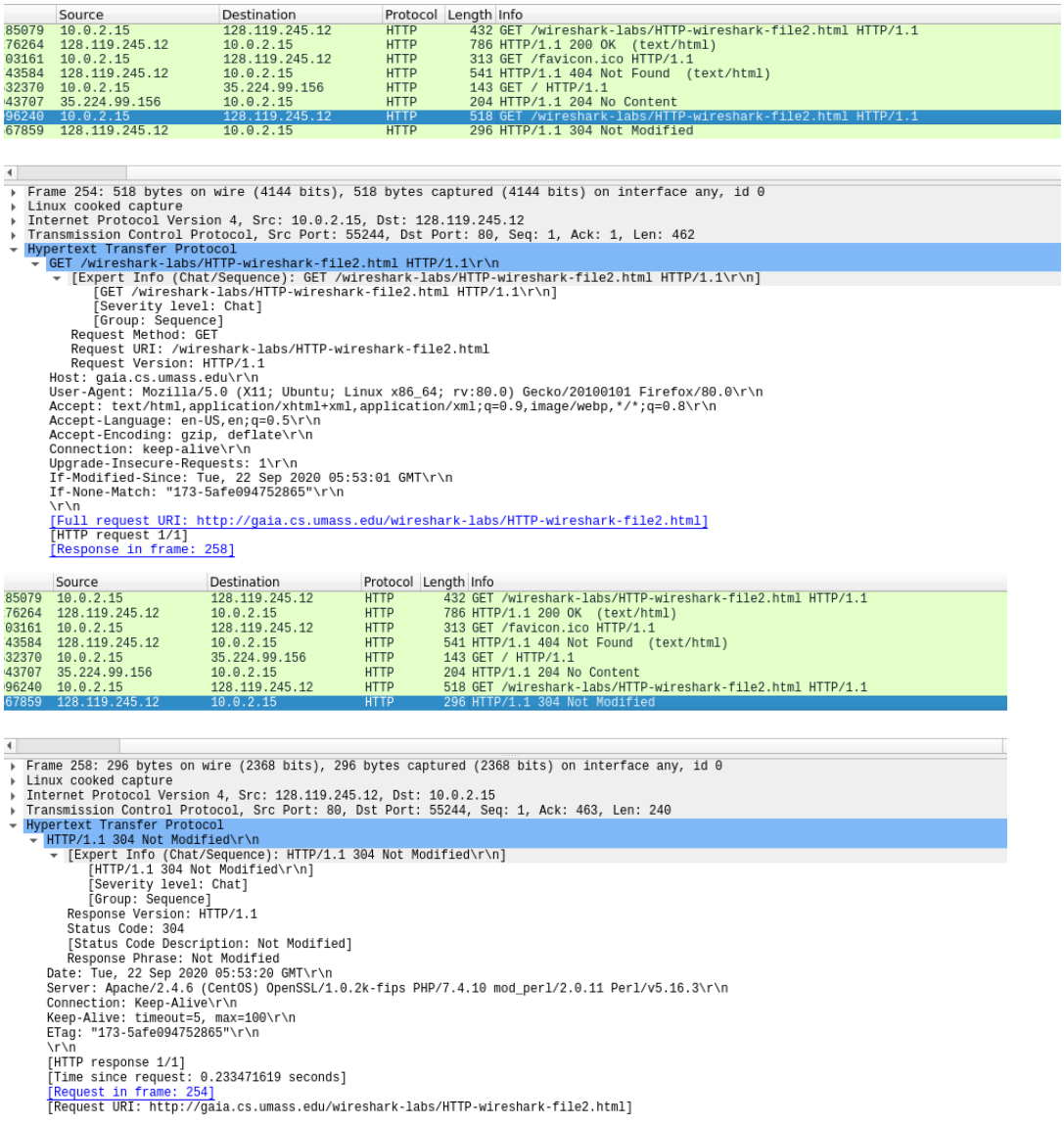
Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

No, there isn't any IF-MODIFIED-SINCE line in the first HTTP GET request from my browser to the server.

Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

The server explicitly returned the contents of the file. This is seen in Wireshark's LINE-BASED TEXT DATA field which shows what the server sent back to the browser.

Second HTTP GET Request



Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?

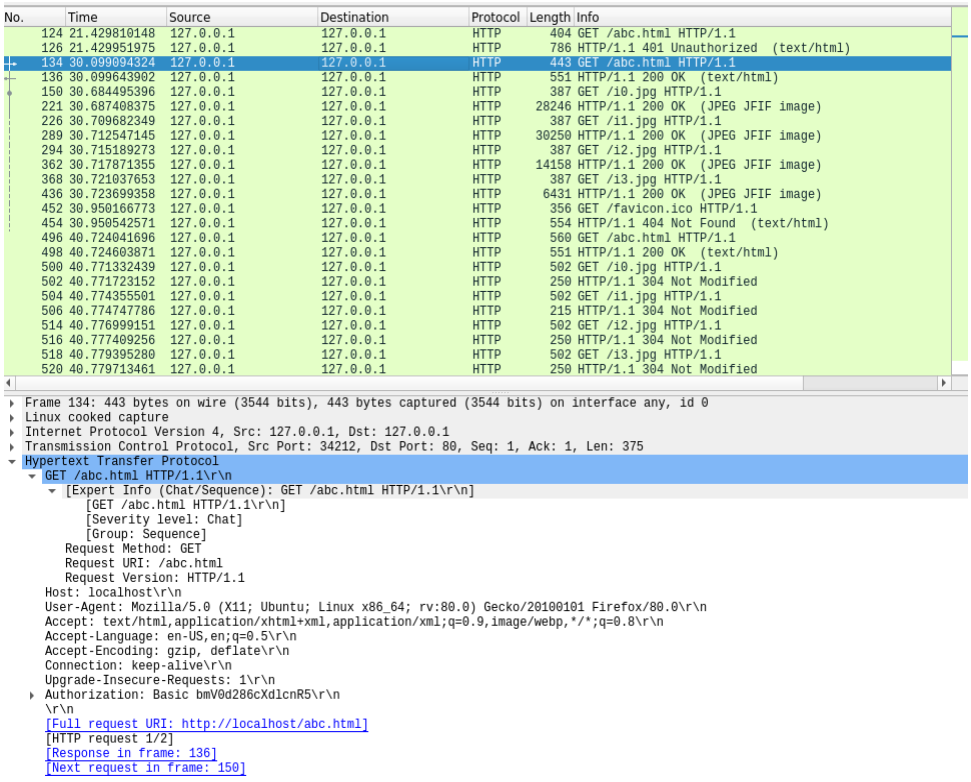
Yes, IF-MODIFIED-SINCE line is included in the second HTTP GET request from my browser to the server. The IF-MODIFIED-SINCE header holds the information of the date and time that was last accessed by me, Tue, 22 Sep 2020 05:53:01 GMT.

What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

The HTTP Status Code is "304: Not Modified". The server did not return the contents of the file because the browser simply retrieved the contents from its cache. Hence the LINE-BASED TEXT DATA is empty since the browser retrived the old file from its cached memory.

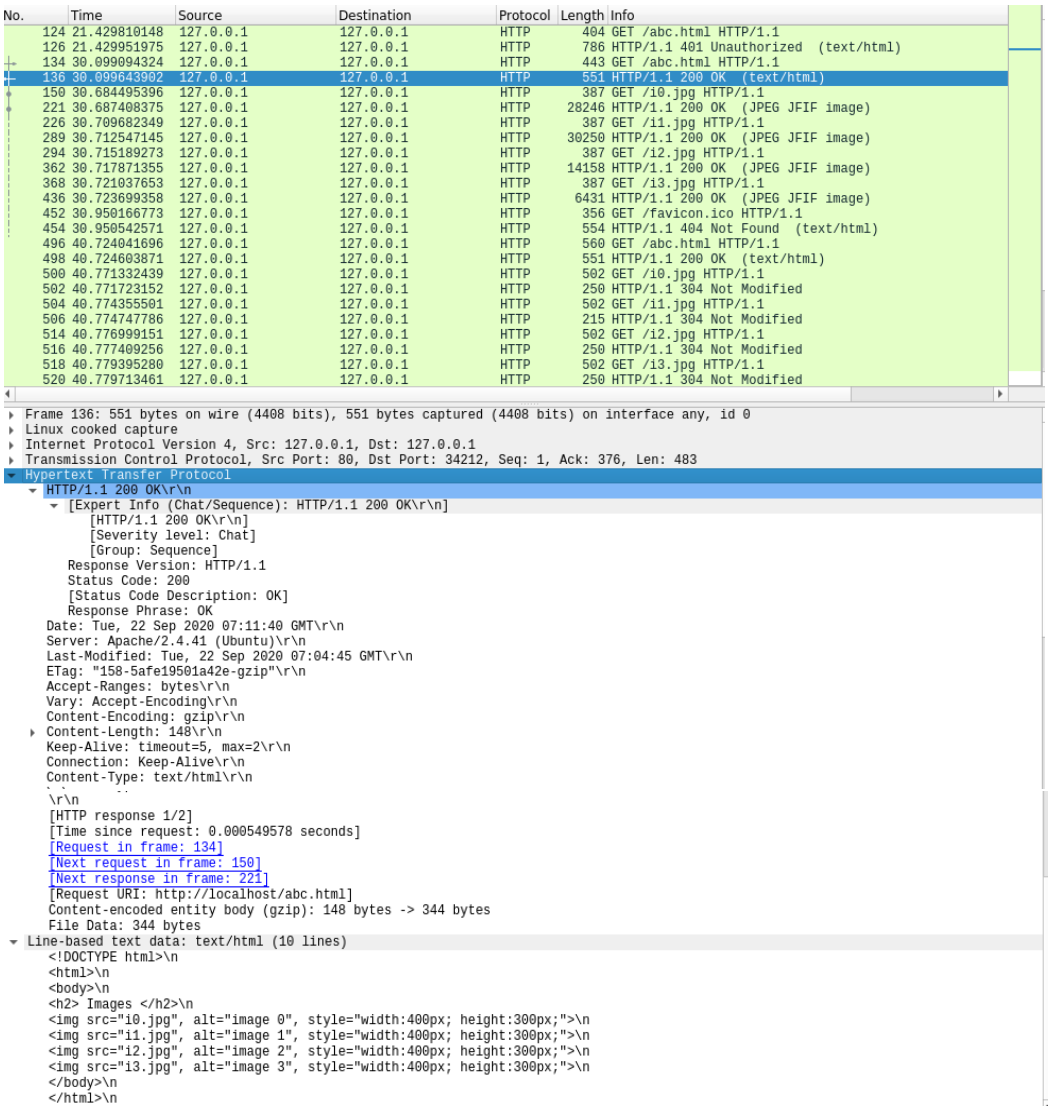
Above Task with 4 Images embedded on the server

First HTTP GET Request



No.	Time	Source	Destination	Protocol	Length	Info
124	21.429810148	127.0.0.1	127.0.0.1	HTTP	404	GET /abc.html HTTP/1.1
126	21.429951975	127.0.0.1	127.0.0.1	HTTP	786	HTTP/1.1 401 Unauthorized (text/html)
134	30.099094324	127.0.0.1	127.0.0.1	HTTP	443	GET /abc.html HTTP/1.1
136	30.099643902	127.0.0.1	127.0.0.1	HTTP	551	HTTP/1.1 200 OK (text/html)
150	30.684495396	127.0.0.1	127.0.0.1	HTTP	387	GET /i0.jpg HTTP/1.1
221	30.687408375	127.0.0.1	127.0.0.1	HTTP	28246	HTTP/1.1 200 OK (JPEG JFIF image)
226	30.709682349	127.0.0.1	127.0.0.1	HTTP	387	GET /i1.jpg HTTP/1.1
289	30.712547145	127.0.0.1	127.0.0.1	HTTP	30250	HTTP/1.1 200 OK (JPEG JFIF image)
294	30.715189273	127.0.0.1	127.0.0.1	HTTP	387	GET /i2.jpg HTTP/1.1
362	30.717871355	127.0.0.1	127.0.0.1	HTTP	14158	HTTP/1.1 200 OK (JPEG JFIF image)
368	30.721037653	127.0.0.1	127.0.0.1	HTTP	387	GET /i3.jpg HTTP/1.1
436	30.723699358	127.0.0.1	127.0.0.1	HTTP	6431	HTTP/1.1 200 OK (JPEG JFIF image)
452	30.950166773	127.0.0.1	127.0.0.1	HTTP	356	GET /favicon.ico HTTP/1.1
454	30.950542571	127.0.0.1	127.0.0.1	HTTP	554	HTTP/1.1 404 Not Found (text/html)
496	40.724041696	127.0.0.1	127.0.0.1	HTTP	560	GET /abc.html HTTP/1.1
498	40.724603871	127.0.0.1	127.0.0.1	HTTP	551	HTTP/1.1 200 OK (text/html)
500	40.771332439	127.0.0.1	127.0.0.1	HTTP	502	GET /i0.jpg HTTP/1.1
502	40.771723152	127.0.0.1	127.0.0.1	HTTP	250	HTTP/1.1 304 Not Modified
504	40.774355501	127.0.0.1	127.0.0.1	HTTP	502	GET /i1.jpg HTTP/1.1
506	40.774747786	127.0.0.1	127.0.0.1	HTTP	215	HTTP/1.1 304 Not Modified
514	40.776999151	127.0.0.1	127.0.0.1	HTTP	502	GET /i2.jpg HTTP/1.1
516	40.777409256	127.0.0.1	127.0.0.1	HTTP	250	HTTP/1.1 304 Not Modified
518	40.779395280	127.0.0.1	127.0.0.1	HTTP	502	GET /i3.jpg HTTP/1.1
520	40.779713461	127.0.0.1	127.0.0.1	HTTP	250	HTTP/1.1 304 Not Modified

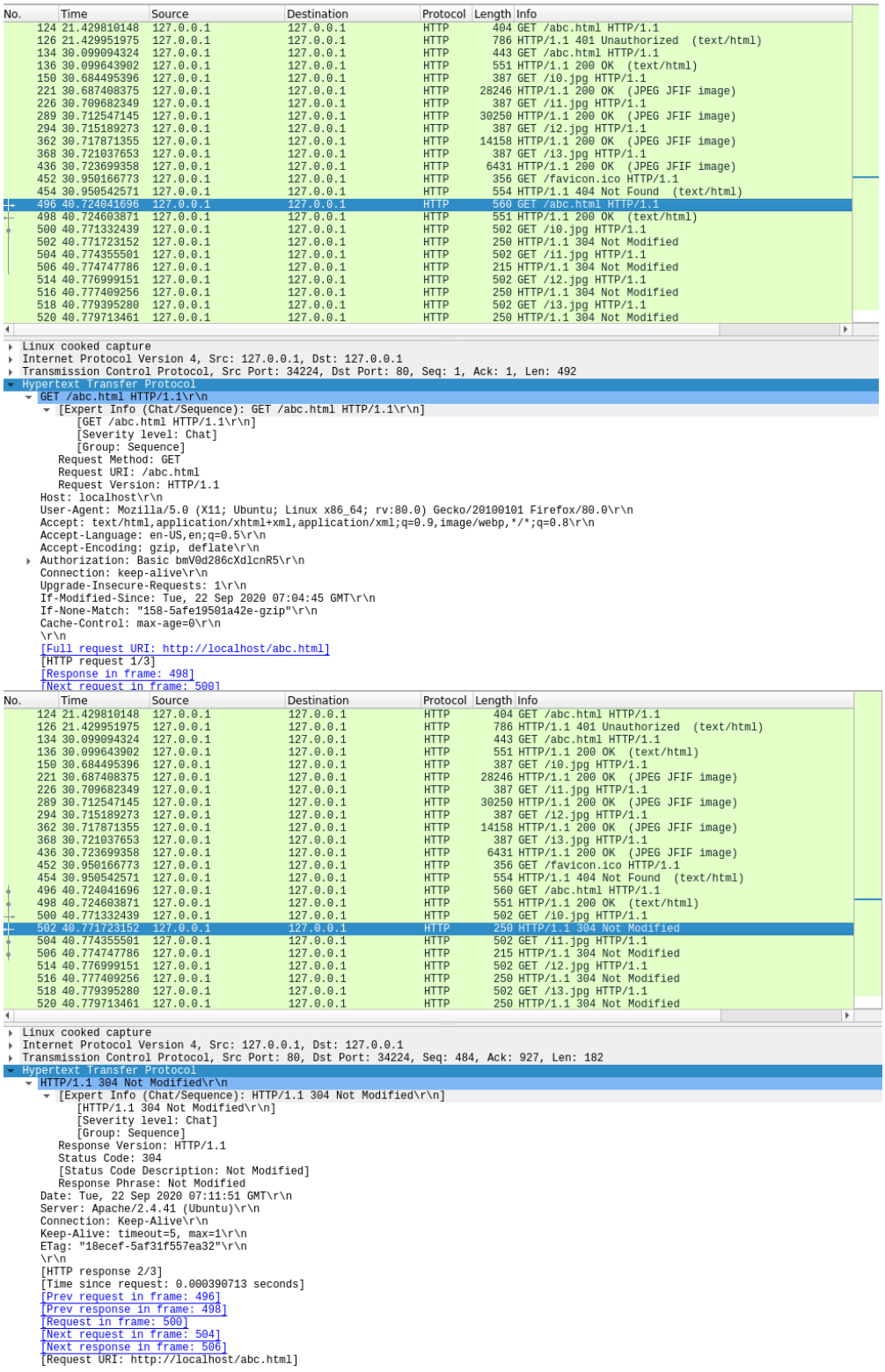
```
Frame 134: 443 bytes on wire (3544 bits), 443 bytes captured (3544 bits) on interface any, id 0
Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 34212, Dst Port: 80, Seq: 1, Ack: 1, Len: 375
Hypertext Transfer Protocol
  GET /abc.html HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /abc.html HTTP/1.1\r\n]
        [GET /abc.html HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
    Request Method: GET
    Request URI: /abc.html
    Request Version: HTTP/1.1
  Host: localhost\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Authorization: Basic bmV0d286cXdlcnR5\r\n
  \r\n
  [Full request URI: http://localhost/abc.html]
  [HTTP request 1/2]
  [Response in frame: 136]
  [Next request in frame: 150]
```



There isn't no IF-MODIFIED-SINCE line in the first HTTP GET request from my browser to the server.

The server explicitly returned the contents of the file. This is seen in Wireshark's LINE-BASED TEXT DATA field which shows what the server sent back to the browser. It mentions all the 4 images that were embedded in the page.

SECOND HTTP GET Request



```
No.     Time           Source          Destination     Protocol Length Info
    124 21.429810148   127.0.0.1       127.0.0.1       HTTP      404 GET /abc.html HTTP/1.1
    126 21.429951975   127.0.0.1       127.0.0.1       HTTP      786 HTTP/1.1 401 Unauthorized  (text/html)
    134 30.099094324   127.0.0.1       127.0.0.1       HTTP      443 GET /abc.html HTTP/1.1
    136 30.099643902   127.0.0.1       127.0.0.1       HTTP      551 HTTP/1.1 200 OK  (text/html)
    150 30.684495396   127.0.0.1       127.0.0.1       HTTP      387 GET /i0.jpg HTTP/1.1
    221 30.687408375   127.0.0.1       127.0.0.1       HTTP    28246 HTTP/1.1 200 OK  (JPEG JFIF image)
    226 30.709682349   127.0.0.1       127.0.0.1       HTTP      387 GET /i1.jpg HTTP/1.1
    289 30.712547145   127.0.0.1       127.0.0.1       HTTP    30250 HTTP/1.1 200 OK  (JPEG JFIF image)
    294 30.715189273   127.0.0.1       127.0.0.1       HTTP      387 GET /i2.jpg HTTP/1.1
    362 30.717871355   127.0.0.1       127.0.0.1       HTTP    14158 HTTP/1.1 200 OK  (JPEG JFIF image)
    368 30.721037653   127.0.0.1       127.0.0.1       HTTP      387 GET /i3.jpg HTTP/1.1
    436 30.723699358   127.0.0.1       127.0.0.1       HTTP     6431 HTTP/1.1 200 OK  (JPEG JFIF image)
    452 30.950166773   127.0.0.1       127.0.0.1       HTTP      356 GET /favicon.ico HTTP/1.1
    454 30.950542571   127.0.0.1       127.0.0.1       HTTP      554 HTTP/1.1 404 Not Found  (text/html)
    496 40.724041696   127.0.0.1       127.0.0.1       HTTP      560 GET /abc.html HTTP/1.1
    498 40.724603871   127.0.0.1       127.0.0.1       HTTP      551 HTTP/1.1 200 OK  (text/html)
    500 40.771332439   127.0.0.1       127.0.0.1       HTTP      502 GET /i0.jpg HTTP/1.1
    502 40.771723152   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified
    504 40.774355501   127.0.0.1       127.0.0.1       HTTP      502 GET /i1.jpg HTTP/1.1
    506 40.774747786   127.0.0.1       127.0.0.1       HTTP      215 HTTP/1.1 304 Not Modified
    514 40.776999151   127.0.0.1       127.0.0.1       HTTP      502 GET /i2.jpg HTTP/1.1
    516 40.777409256   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified
    518 40.779395280   127.0.0.1       127.0.0.1       HTTP      502 GET /i3.jpg HTTP/1.1
    520 40.779713461   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified

Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 34224, Dst Port: 80, Seq: 1, Ack: 1, Len: 492
Hypertext Transfer Protocol
  GET /abc.html HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /abc.html HTTP/1.1\r\n]
        [GET /abc.html HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /abc.html
      Request Version: HTTP/1.1
  Host: localhost\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:80.0) Gecko/20100101 Firefox/80.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Authorization: Basic bmV0d286cXdlcnR5\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  If-Modified-Since: Tue, 22 Sep 2020 07:04:45 GMT\r\n
  If-None-Match: "158-5afe19501a42e-gzip"\r\n
  Cache-Control: max-age=0\r\n
  \r\n
  [Full request URI: http://localhost/abc.html]
  [HTTP request 1/3]
  [Response in frame: 498]
  [Next request in frame: 500]

No.     Time           Source          Destination     Protocol Length Info
    124 21.429810148   127.0.0.1       127.0.0.1       HTTP      404 GET /abc.html HTTP/1.1
    126 21.429951975   127.0.0.1       127.0.0.1       HTTP      786 HTTP/1.1 401 Unauthorized  (text/html)
    134 30.099094324   127.0.0.1       127.0.0.1       HTTP      443 GET /abc.html HTTP/1.1
    136 30.099643902   127.0.0.1       127.0.0.1       HTTP      551 HTTP/1.1 200 OK  (text/html)
    150 30.684495396   127.0.0.1       127.0.0.1       HTTP      387 GET /i0.jpg HTTP/1.1
    221 30.687408375   127.0.0.1       127.0.0.1       HTTP    28246 HTTP/1.1 200 OK  (JPEG JFIF image)
    226 30.709682349   127.0.0.1       127.0.0.1       HTTP      387 GET /i1.jpg HTTP/1.1
    289 30.712547145   127.0.0.1       127.0.0.1       HTTP    30250 HTTP/1.1 200 OK  (JPEG JFIF image)
    294 30.715189273   127.0.0.1       127.0.0.1       HTTP      387 GET /i2.jpg HTTP/1.1
    362 30.717871355   127.0.0.1       127.0.0.1       HTTP    14158 HTTP/1.1 200 OK  (JPEG JFIF image)
    368 30.721037653   127.0.0.1       127.0.0.1       HTTP      387 GET /i3.jpg HTTP/1.1
    436 30.723699358   127.0.0.1       127.0.0.1       HTTP     6431 HTTP/1.1 200 OK  (JPEG JFIF image)
    452 30.950166773   127.0.0.1       127.0.0.1       HTTP      356 GET /favicon.ico HTTP/1.1
    454 30.950542571   127.0.0.1       127.0.0.1       HTTP      554 HTTP/1.1 404 Not Found  (text/html)
    496 40.724041696   127.0.0.1       127.0.0.1       HTTP      560 GET /abc.html HTTP/1.1
    498 40.724603871   127.0.0.1       127.0.0.1       HTTP      551 HTTP/1.1 200 OK  (text/html)
    500 40.771332439   127.0.0.1       127.0.0.1       HTTP      502 GET /i0.jpg HTTP/1.1
    502 40.771723152   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified
    504 40.774355501   127.0.0.1       127.0.0.1       HTTP      502 GET /i1.jpg HTTP/1.1
    506 40.774747786   127.0.0.1       127.0.0.1       HTTP      215 HTTP/1.1 304 Not Modified
    514 40.776999151   127.0.0.1       127.0.0.1       HTTP      502 GET /i2.jpg HTTP/1.1
    516 40.777409256   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified
    518 40.779395280   127.0.0.1       127.0.0.1       HTTP      502 GET /i3.jpg HTTP/1.1
    520 40.779713461   127.0.0.1       127.0.0.1       HTTP      250 HTTP/1.1 304 Not Modified

Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 34224, Seq: 484, Ack: 927, Len: 182
Hypertext Transfer Protocol
  HTTP/1.1 304 Not Modified\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
        [HTTP/1.1 304 Not Modified\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
  Date: Tue, 22 Sep 2020 07:11:51 GMT\r\n
  Server: Apache/2.4.41 (Ubuntu)\r\n
  Connection: Keep-Alive\r\n
  Keep-Alive: timeout=5, max=1\r\n
  ETag: "18ecef-5af31f557ea32"\r\n
  \r\n
  [HTTP response 2/3]
  [Time since request: 0.000390713 seconds]
  [Prev request in frame: 496]
  [Prev response in frame: 498]
  [Request in frame: 500]
  [Next request in frame: 504]
  [Next response in frame: 506]
  [Request URI: http://localhost/abc.html]
```

IF-MODIFIED-SINCE line is included in the second HTTP GET request from my browser to the server. The IF-MODIFIED-SINCE header holds the information of the date and time that was last accessed by me, Tue, 22 Sep 2020 07:11:51 GMT.

The HTTP Status Code is "304: Not Modified" for all 4 images requests. The server did not return the contents of the file because the browser simply retrieved the contents from its cache. Hence the LINE-BASED TEXT DATA is empty since the browser retrived the old image contents from its cached memory.

CONCLUSION:

- Analysed Basic Authentication and Cookies and behaviour of browser when the cookie is set from one embedded image
- Understood and worked out the decryption of the Auth-Basic encrypted string using base64 algorithm to retrieve username & password
- Understood various parameters set by the cookie in wireshark captures as per the password set-up
- Analysed behaviour of Conditional Get with If-Modified-Since when embedded objects are retrieved

Week #4
Implementation of a Local DNS Server

DNS (Domain Name System) is the Internet's phone book; it translates hostnames to IP addresses (and vice versa). This translation is through DNS resolution, which happens behind the scene.

The objectives of this lab are to understand:

- DNS and how it works
- Install and set up a DNS server
- Functionality and operations

Lab Setup

DNS Server: 10.2.22.184 User/Client: 10.2.22.195

Note: Use the default IP address provided by PESU LAN.

First Test:

Ping a computer such as www.flipkart.com. Please use Wireshark to show the DNS query triggered by your ping command and DNS response. Describe your observation. (Take a screenshot).

Part 1: Setting Up a Local DNS Server

Task 1: Configure the User Machine

On the client machine 10.2.22.195, we need to use 10.2.22.184 as the local DNS server. This is achieved by changing the resolver configuration file **(/etc/resolv.conf**) of the user machine, so the server 10.2.22.184 is added as the first nameserver entry in the file, i.e., this server will be used as the primary DNS server. Add the following entry to the **/etc/resolvconf/resolv.conf.d/head** file.

nameserver 10.2.22.184

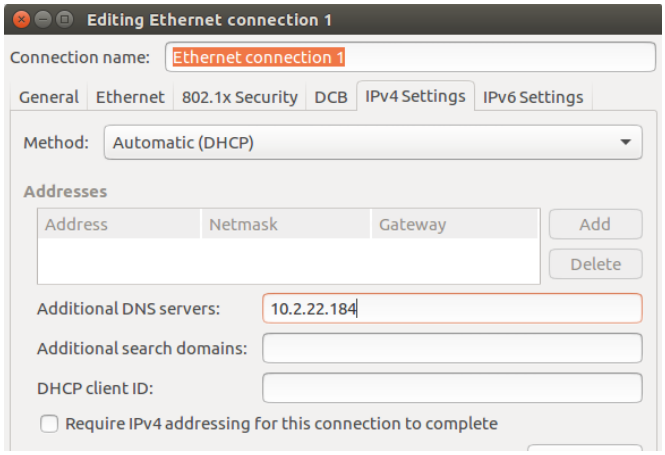
Run the following command for the change to take effect.

sudo resolvconf -u

The following screenshot shows how to set DNS server on the client machine.

```
isfcr@isfcr-H110M-H:~$ sudo nano /etc/resolvconf/resolv.conf.d/head
[sudo] password for isfcr:
isfcr@isfcr-H110M-H:~$ sudo cat /etc/resolvconf/resolv.conf.d/head
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.2.22.184
isfcr@isfcr-H110M-H:~$ sudo resolvconf -u
isfcr@isfcr-H110M-H:~$
```

Also, add 10.2.22.184 in 'Additional DNS servers' field in IPv4 settings of client machine.



Second Test:

Ping a computer such as www.flipkart.com. Please use Wireshark to show the DNS query triggered by your ping command and DNS response. Describe your observation. (Take a screenshot).

Task 2: Set Up a Local DNS Server

Note: If bind9 server is not already installed, install using the command

$ sudo apt-get update

$ sudo apt-get install bind9

Step 1: Configure the BIND9 Server.

BIND9 gets its configuration from a file called **/etc/bind/named.conf**. This file is the primary configuration file, and it usually contains several "include" entries. One of the included files is called **/etc/bind/named.conf.options**. This is where we typically set up the configuration options. Let us first set up an option related to DNS cache by adding a dump-file entry to the options block. The above option specifies where the cache content should be dumped to if BIND is asked to dump its cache.

```
isfcr@isfcr-H110M-H:~$ sudo nano /etc/bind/named.conf.options
[sudo] password for isfcr:
```

```
 GNU nano 2.5.3            File: /etc/bind/named.conf.options

options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        dump-file "/var/cache/bind/dump.db";
```

The above option specifies where the cache content should be dumped to if BIND is asked to dump its cache. If this option is not specified, BIND dumps the cache to a default file called **/var/cache/bind/named_dump.db**.

Step 2: Start DNS server

We start the DNS server using the command:

$ sudo service bind9 restart

```
isfcr@isfcr-H110M-H:~$ sudo service bind9 restart
isfcr@isfcr-H110M-H:~$
```

The two commands shown below are related to DNS cache. The first command dumps the content of the cache to the file specified above, and the second command clears the cache.

```
isfcr@isfcr-H110M-H:~$ sudo rndc dumpdb -cache
isfcr@isfcr-H110M-H:~$ sudo rndc flush
```

Step 3: Use the DNS server

Third Test:

Now, go back to your user machine (10.2.22.195), and ping a computer such as www.flipkart.com and describe your observation. Please use Wireshark to show the DNS query triggered by your ping command. Please also indicate when the DNS cache is used. (Take a screenshot).

Note: Compare the above three Wireshark DNS packet capture screenshots taken above.

Task 3: Host a Zone in the Local DNS server.

Assume that we own a domain, we will be responsible for providing the definitive answer regarding this domain. We will use our local DNS server as the authoritative nameserver for the domain. In this lab, we will set up an authoritative server for the **example.com** domain.

This domain name is reserved for use in documentation, and is not owned by anybody, so it is safe to use it.

Step 1: Create Zones

We had two zone entries in the DNS server by adding the following contents to **/etc/bind/named.conf** as shown in the below screenshot. The first zone is for forward lookup (from hostname to IP), and the second zone is for reverse lookup (from IP to hostname).

```
isfcr@isfcr-H110M-H:~$ sudo nano /etc/bind/named.conf
isfcr@isfcr-H110M-H:~$ sudo cat /etc/bind/named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";

zone "example.com" {
type master;
file "/etc/bind/example.com.db";
};

zone "22.2.10.in-addr.arpa" {
type master;
file "/etc/bind/10.2.22.db";
};
isfcr@isfcr-H110M-H:~$
```

Note: In above screenshot, 10.2.22.0 is the subnet mask of your IP address.

Step 2: Setup the forward lookup zone file

We create **example.com.db** zone file with the following contents in the **/etc/bind/** directory where the actual DNS resolution is stored.

```
$TTL 3D
@     IN    SOA   ns.example.com. admin.example.com. (
            2008111001
            8H
            2H
            4W
            1D)

@     IN    NS    ns.example.com.
@     IN    MX    10 mail.example.com.

www   IN    A     10.2.22.101
mail  IN    A     10.2.22.102
ns    IN    A     10.2.22.10
*.example.com.   IN    A 10.2.22.100
```

The symbol '@' is a special notation representing the origin specified in **named.conf** (the string after "**zone**"). Therefore, '@' here stands for **example.com**. This zone file contains 7 resource records (RRs), including a SOA (Start Of Authority) RR, a NS (Name Server) RR, a MX (Mail eXchanger) RR, and 4 A (host Address) RRs.

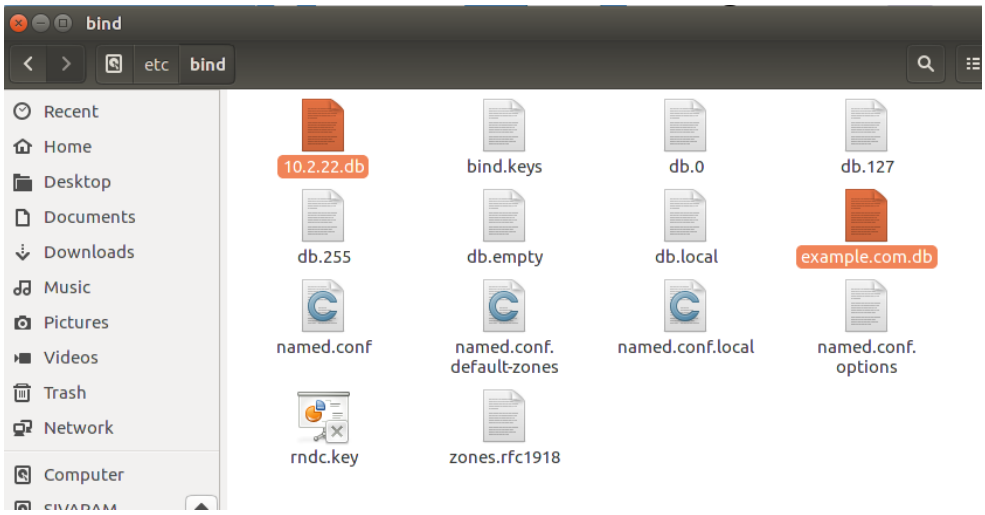
Step 3: Setup the reverse lookup zone file

We create a reverse DNS lookup file called **10.2.22.db** for the example.net domain to support DNS reverse lookup, i.e., from IP address to hostname in the **/etc/bind/** directory with the following contents.

```
$TTL 3D
@     IN    SOA   ns.example.com. admin.example.com. (
            2008111001
            8H
            2H
            4W
            1D)
@     IN    NS    ns.example.com.

101   IN    PTR   www.example.com.
102   IN    PTR   mail.example.com.
10    IN    PTR   ns.example.com.
```

Step 4: Copy the above files into **/etc/bind** location.

```
isfcr@isfcr-H110M-H:~$ sudo cp 10.2.22.db /etc/bind
isfcr@isfcr-H110M-H:~$ sudo cp example.com.db /etc/bind
```



Task 4: Restart the BIND server and test

Step 1: When all the changes are made, remember to restart the BIND server. Now we will restart the DNS server using the following command:

$ sudo service bind9 restart

```
isfcr@isfcr-H110M-H:~$ sudo service bind9 restart
isfcr@isfcr-H110M-H:~$
```

Step 2: Now, go back to the client machine and ask the local DNS server for the IP address of www.example.com using the dig command.

Dig stands for (Domain Information Groper) is a network administration command-line tool for querying DNS name servers. It is useful for verifying and troubleshooting DNS problems

and also to perform DNS lookups and displays the answers that are returned from the name server that were queried. dig is part of the BIND domain name server software suite.

```
isfcr@isfcr-H110M-H:~$ dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5668
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.2.22.101

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       10.2.22.10

;; Query time: 0 msec
;; SERVER: 10.2.22.184#53(10.2.22.184)
;; WHEN: Tue Jul 30 11:27:36 IST 2019
;; MSG SIZE  rcvd: 93
```

We can see that the ANSWER SECTION contains the DNS mapping. We can see that the IP address of www.example.com is now 10.2.22.101, which is what we have setup in the DNS server.

Step 3: Observe the results in Wireshark capture.

2	1.000080291	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171
3	8.029680511	::1	::1	UDP	65 42520 → 42520 Len=1
4	8.029707882	10.2.22.195	10.2.22.184	DNS	88 Standard query 0x1624 A www.example.com OPT
5	8.030388651	10.2.22.184	10.2.22.195	DNS	137 Standard query response 0x1624 A www.example.com A 10.2.22.101 NS ns.example.com A 10.2.22.10 OPT
6	9.120902499	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171
7	9.999525402	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171
8	10.999577685	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171
9	13.040903664	Giga-Byt_dc:e3:e9		ARP	62 Who has 10.2.22.195? Tell 10.2.22.184
10	13.040932978	Giga-Byt_76:0c:f5		ARP	44 10.2.22.195 is at e0:d5:5e:76:0c:f5
11	19.156379156	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171
12	20.000032310	Azurewav_56:b7:ed		ARP	62 Who has 10.2.22.161? Tell 10.2.22.171

```
▸ Frame 5: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.2.22.184, Dst: 10.2.22.195
▸ User Datagram Protocol, Src Port: 53, Dst Port: 37705
▾ Domain Name System (response)
    Transaction ID: 0x1624
  ▸ Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 2
  ▸ Queries
  ▾ Answers
    ▾ www.example.com: type A, class IN, addr 10.2.22.101
          Name: www.example.com
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 259200
          Data length: 4
          Address: 10.2.22.101
  ▾ Authoritative nameservers
    ▾ example.com: type NS, class IN, ns ns.example.com
          Name: example.com
          Type: NS (authoritative Name Server) (2)
          Class: IN (0x0001)
          Time to live: 259200
          Data length: 5
          Name Server: ns.example.com
  ▾ Additional records
    ▾ ns.example.com: type A, class IN, addr 10.2.22.10
          Name: ns.example.com
          Type: A (Host Address) (1)
          Class: IN (0x0001)
          Time to live: 259200
          Data length: 4
          Address: 10.2.22.10
    ▾ <Root>: type OPT
          Name: <Root>
          Type: OPT (41)
          UDP payload size: 4096
          Higher bits in extended RCODE: 0x00
          EDNS0 version: 0
        ▾ Z: 0x0000
            0... .... .... .... = DO bit: Cannot handle DNSSEC security RRs
            .000 0000 0000 0000 = Reserved: 0x0000
          Data length: 0
    [Request In: 4]
    [Time: 0.000680769 seconds]
```

To load and clear DNS cache, use the below commands.

```
isfcr@isfcr-H110M-H:~$ sudo rndc dumpdb -cache
isfcr@isfcr-H110M-H:~$ sudo rndc flush
```

Edmodo Requirements:

1) Three Wireshark packet capture screenshots for pinging (Packet list pane and Packet details pane) – **ping www.flipkart.com** command
2) **dig www.example.com** command (in Terminal)
3) Wireshark packet capture – **dig www.example.com** command (Packet list pane and Packet details pane)
4) Local DNS cache on server machine

Observation Notebook Requirements:

For **'ping www.flipkart.com',** answer the following questions

1) Locate the DNS query and response messages. Are then sent over UDP or TCP?

2) What is the destination port for the DNS query message? What is the source port of DNS response message?
3) To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?
4) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
5) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?
6) Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?
7) What is the destination port for the DNS query message? What is the source port of DNS response message?
8) To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
9) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
10) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

Computer Networks Laboratory – UE18CS305

PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D

Week #4 – Implementation of a Local DNS Server – 29 Sept 2020

OBJECTIVE:

- To understand DNS and how it works
- Install and set-up a DNS server & understand its functionalities and operations

SET-UP:

DNS Server – ubuntu1, IP Address - 10.0.2.12

```
ubuntu1@PES1201800046:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:73:bc:c2
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::e5cb:bb29:941d:676/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7579 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8823 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2095912 (2.0 MB)  TX bytes:686437 (686.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1478 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1478 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:127982 (127.9 KB)  TX bytes:127982 (127.9 KB)
```

DNS Client – ubuntu2, IP Address - 10.0.2.13

```
ubuntu2@PES1201800046:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:23:02:14
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2df0:7273:553:a69/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:706 errors:0 dropped:0 overruns:0 frame:0
          TX packets:551 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:86750 (86.7 KB)  TX bytes:55927 (55.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1020 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1020 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:81798 (81.7 KB)  TX bytes:81798 (81.7 KB)
```

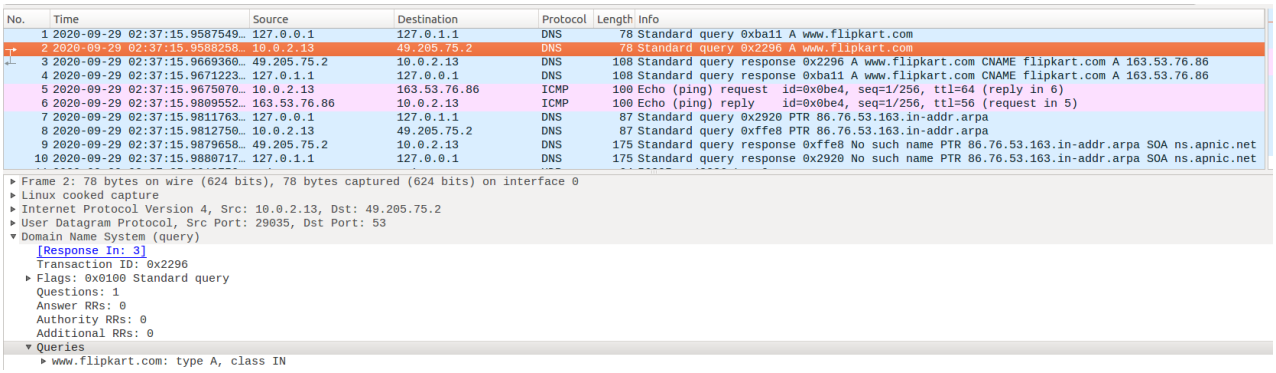
$ ping www.flipkart.com -c 1

```
ubuntu2@PES1201800046:~$ping www.flipkart.com -c1
PING flipkart.com (163.53.76.86) 56(84) bytes of data.
64 bytes from 163.53.76.86: icmp_seq=1 ttl=56 time=27.1 ms

--- flipkart.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 27.184/27.184/27.184/0.000 ms
```

OBSERVATIONS: Task 1 & 2 –

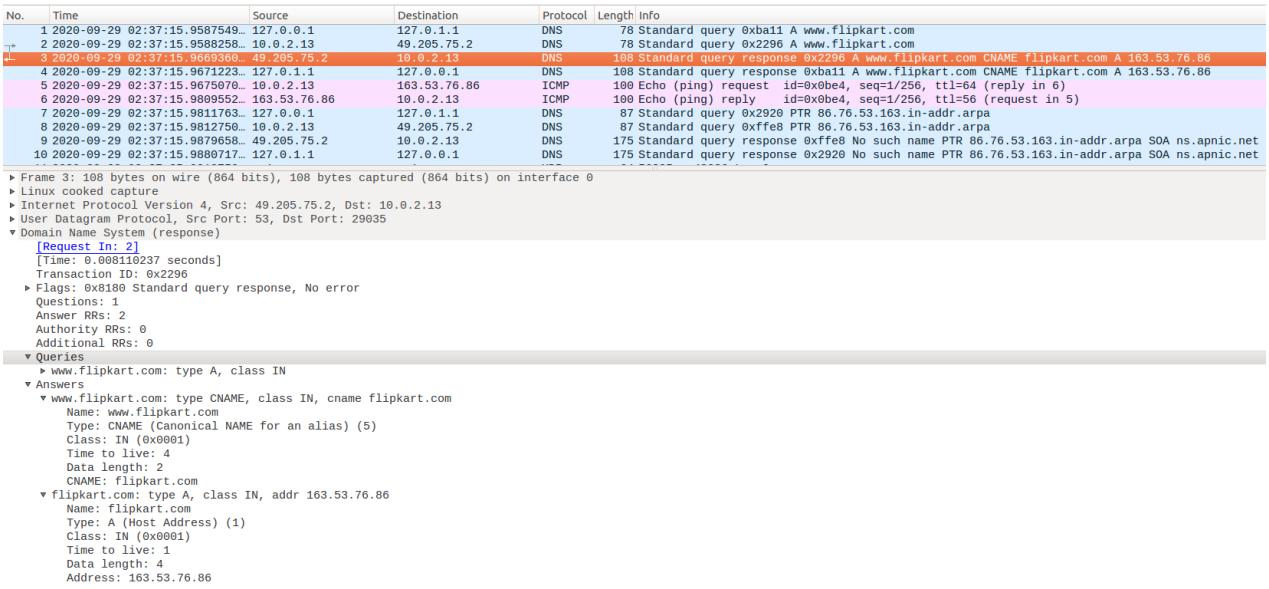
First Test:

DNS Query



Source IP Address	10.0.2.13
Destination IP Address	49.205.75.2
Protocol	DNS
Query	www.flipkart.com
Type	Standard Query Type A
Class	IN
Source Port	29035
Destination Port	53

DNS Response



Source IP Address	49.205.75.2
Destination IP Address	10.0.2.13
Protocol	DNS
Message	DNS Response
Source Port	53
Destination Port	29035

Answers:

www.flipkart.com

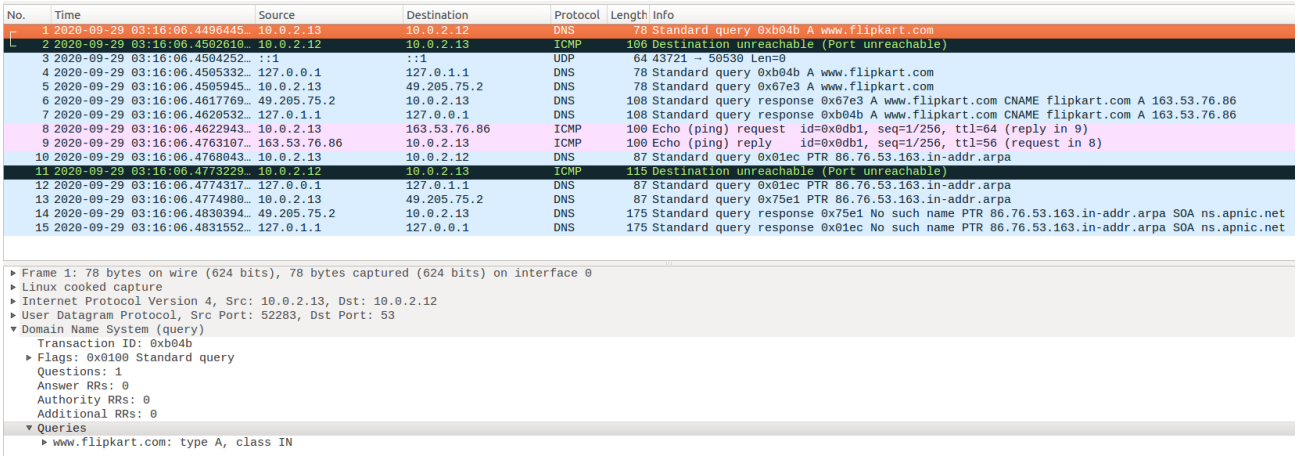
Name	www.flipkart.com
Type	CNAME (Canonical NAME for an alias) (5)
Class	IN (0x0001)
TTL	4
Data Length	2
CNAME	flipkart.com

flipkart.com

Name	www.flipkart.com
Type	A (Host Address) (1)
Class	IN (0x0001)
TTL	1
Data Length	4
Address	163.53.76.86

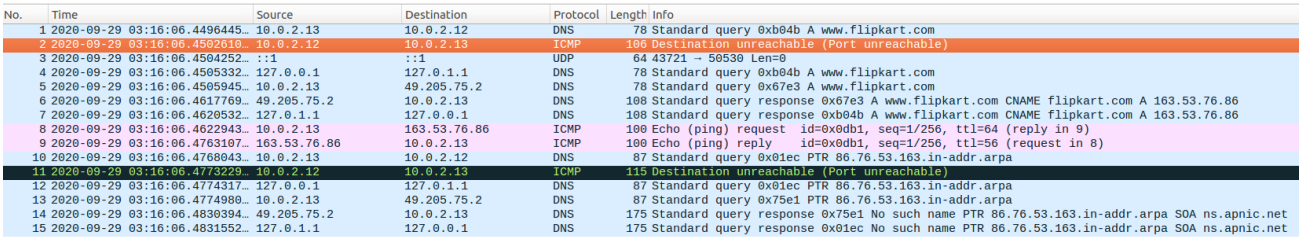
Second Test

DNS Query

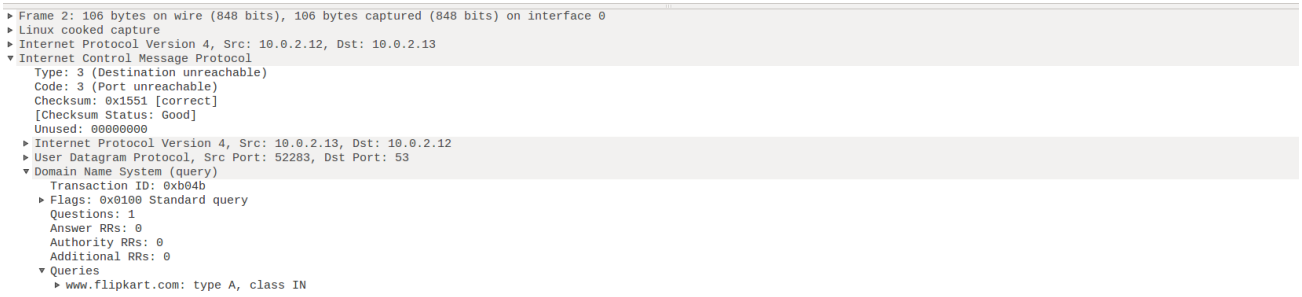


Source IP Address	10.0.2.13
Destination IP Address	10.0.2.12
Protocol	DNS
Query	www.flipkart.com
Type	Standard Query Type A
Class	Class IN
Source Port	52283
Destination Port	53

DNS Request



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-29 03:16:06.4496445…	10.0.2.13	10.0.2.12	DNS	78	Standard query 0xb04b A www.flipkart.com
2	2020-09-29 03:16:06.4562610…	10.0.2.12	10.0.2.13	ICMP	106	Destination unreachable (Port unreachable)
3	2020-09-29 03:16:06.4504252…	::1	::1	UDP	64	43721 → 50530 Len=0
4	2020-09-29 03:16:06.4505332…	127.0.0.1	127.0.1.1	DNS	78	Standard query 0xb04b A www.flipkart.com
5	2020-09-29 03:16:06.4505945…	10.0.2.13	49.205.75.2	DNS	78	Standard query 0x67e3 A www.flipkart.com
6	2020-09-29 03:16:06.4617769…	49.205.75.2	10.0.2.13	DNS	108	Standard query response 0x67e3 A www.flipkart.com CNAME flipkart.com A 163.53.76.86
7	2020-09-29 03:16:06.4620532…	127.0.1.1	127.0.0.1	DNS	108	Standard query response 0xb04b A www.flipkart.com CNAME flipkart.com A 163.53.76.86
8	2020-09-29 03:16:06.4622943…	10.0.2.13	163.53.76.86	ICMP	100	Echo (ping) request id=0x0db1, seq=1/256, ttl=64 (reply in 9)
9	2020-09-29 03:16:06.4763107…	163.53.76.86	10.0.2.13	ICMP	100	Echo (ping) reply id=0x0db1, seq=1/256, ttl=56 (request in 8)
10	2020-09-29 03:16:06.4768043…	10.0.2.13	10.0.2.12	DNS	87	Standard query 0x01ec PTR 86.76.53.163.in-addr.arpa
11	2020-09-29 03:16:06.4773229…	10.0.2.12	10.0.2.13	ICMP	115	Destination unreachable (Port unreachable)
12	2020-09-29 03:16:06.4774317…	127.0.0.1	127.0.1.1	DNS	87	Standard query 0x01ec PTR 86.76.53.163.in-addr.arpa
13	2020-09-29 03:16:06.4774980…	10.0.2.13	49.205.75.2	DNS	87	Standard query 0x75e1 PTR 86.76.53.163.in-addr.arpa
14	2020-09-29 03:16:06.4830394…	49.205.75.2	10.0.2.13	DNS	175	Standard query response 0x75e1 No such name PTR 86.76.53.163.in-addr.arpa SOA ns.apnic.net
15	2020-09-29 03:16:06.4831552…	127.0.1.1	127.0.0.1	DNS	175	Standard query response 0x01ec No such name PTR 86.76.53.163.in-addr.arpa SOA ns.apnic.net



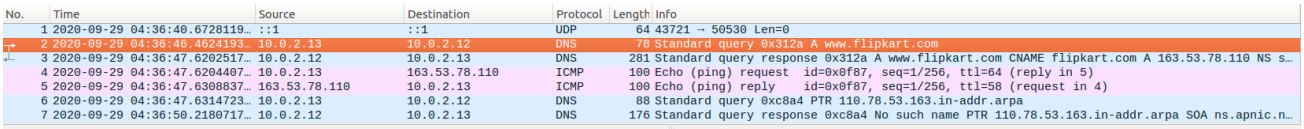
```
▸ Frame 2: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.12, Dst: 10.0.2.13
▾ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
    Checksum: 0x1551 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  ▸ Internet Protocol Version 4, Src: 10.0.2.13, Dst: 10.0.2.12
  ▸ User Datagram Protocol, Src Port: 52283, Dst Port: 53
  ▾ Domain Name System (query)
      Transaction ID: 0xb04b
    ▸ Flags: 0x0100 Standard query
      Questions: 1
      Answer RRs: 0
      Authority RRs: 0
      Additional RRs: 0
    ▾ Queries
      ▸ www.flipkart.com: type A, class IN
```

Source IP Address	10.0.2.12
Destination IP Address	10.0.2.13
Protocol	DNS / ICMP
Message	DNS Response (Destination Port unreachable)

The port is unreachable as the Local DNS server was not set up yet. Only the client machine (ubuntu2) has 10.0.2.12 as nameserver in its resolvconf head file. Hence, Answers section is also not present.

Third Test

DNS Query

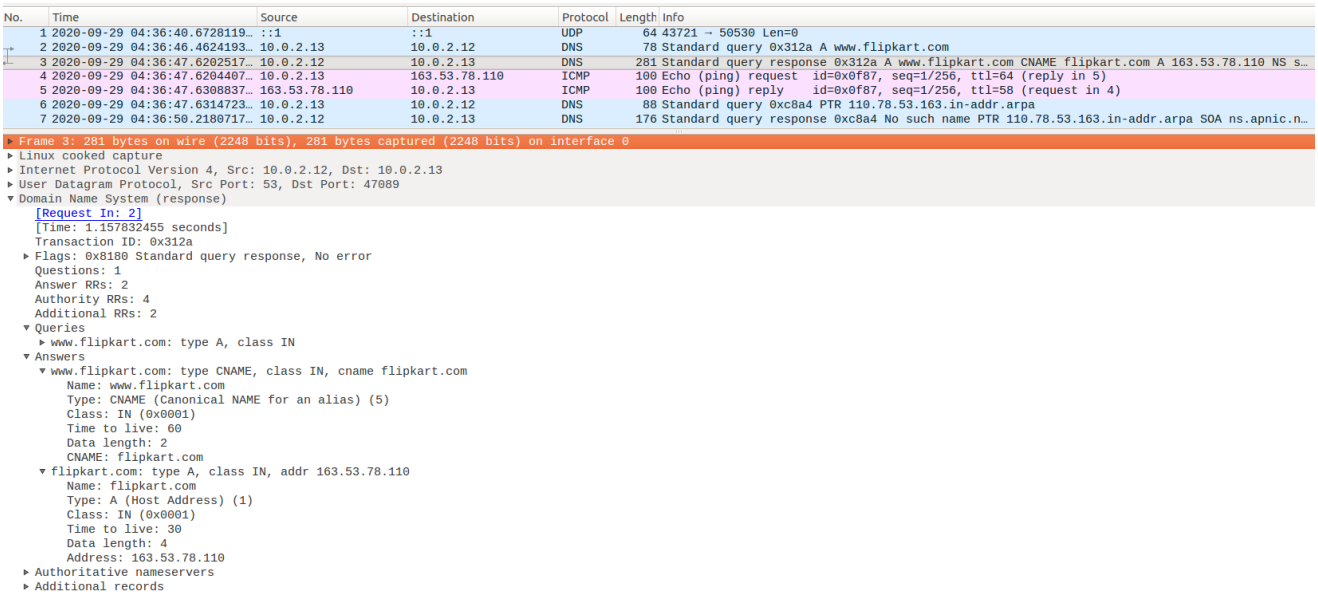


No.	Time	Source	Destination	Protocol	Length	Info
1	2020-09-29 04:36:40.6728119…	::1	::1	UDP	64	43721 → 50530 Len=0
2	2020-09-29 04:36:46.4624193…	10.0.2.13	10.0.2.12	DNS	78	Standard query 0x312a A www.flipkart.com
3	2020-09-29 04:36:47.6202517…	10.0.2.12	10.0.2.13	DNS	281	Standard query response 0x312a A www.flipkart.com CNAME flipkart.com A 163.53.78.110 NS s…
4	2020-09-29 04:36:47.6204407…	10.0.2.13	163.53.78.110	ICMP	100	Echo (ping) request id=0x0f87, seq=1/256, ttl=64 (reply in 5)
5	2020-09-29 04:36:47.6308837…	163.53.78.110	10.0.2.13	ICMP	100	Echo (ping) reply id=0x0f87, seq=1/256, ttl=58 (request in 4)
6	2020-09-29 04:36:47.6314723…	10.0.2.13	10.0.2.12	DNS	88	Standard query 0xc8a4 PTR 110.78.53.163.in-addr.arpa
7	2020-09-29 04:36:50.2180717…	10.0.2.12	10.0.2.13	DNS	176	Standard query response 0xc8a4 No such name PTR 110.78.53.163.in-addr.arpa SOA ns.apnic.n…

```
▸ Frame 2: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.13, Dst: 10.0.2.12
▸ User Datagram Protocol, Src Port: 47089, Dst Port: 53
▾ Domain Name System (query)
    [Response In: 3]
    Transaction ID: 0x312a
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ www.flipkart.com: type A, class IN
```

Source IP Address	10.0.2.13
Destination IP Address	10.0.2.12
Protocol	DNS
Query	www.flipkart.com
Type	Standard Query Type A
Class	Class IN
Source Port	47089
Destination Port	53

DNS Response



Source IP Address	10.0.2.12
Destination IP Address	10.0.2.13
Protocol	DNS
Message	DNS Response
Source Port	53
Destination Port	47089

Here, using bind9, the Local DNS server was successfully set up and the DNS for www.flipkart.com was also successfully resolved.

Answers:

www.flipkart.com

Name	www.flipkart.com
Type	CNAME (Canonical NAME for an alias) (5)
Class	IN (0x0001)
TTL	60
Data Length	2
CNAME	flipkart.com

flipkart.com

Name	www.flipkart.com
Type	A (Host Address) (1)
Class	IN (0x0001)
TTL	30
Data Length	4
Address	163.53.76.86

Local DNS cache

After executing, ping www.flipkart.com -c1, the Local DNS cache of flipkart.com record is seen in /var/cache/bind/dump.db.

The NS Authoritative Zone Server information & CNAME Type information is cached.

```
ubuntu1@PES1201800046:~$sudo cat /var/cache/bind/dump.db | grep "flipkart"
flipkart.com.           86370   NS      sdns14.ultradns.biz.
www.flipkart.com.       30      CNAME   flipkart.com.
ubuntu1@PES1201800046:~$
```

Comparing the above three Wireshark DNS packet capture screenshots taken above for test 1, 2, 3

- Test 1, 2, 3 for ping www.flipkart.com -c 1 had DNS Query Message of Standard Type A and Class IN, all had 53 as Destination Ports and Source IP Addresses 10.0.2.13 (Client Machine's IP Address)
- Test 2 had unreachable destination ports as Local DNS Server was not set up on server machine ubuntu 2
- Test 1 & 3 (Source IP Address – 10.0.2.12) for ping www.flipkart.com -c 1 had successful DNS Response Message with 53 as Source Port and Destination IP Address as 10.0.2.13 (Client's IP Address). Both had the following answers - CNAME Type (Data Length 2) and A Type (Data Length 4), Class IN. But the only difference was TTLs were higher in Test 3.

Task 3 & 4 –

$ dig www.example.com

```
ubuntu2@PES1201800046:~$dig www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49849
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.               IN      A

;; ANSWER SECTION:
www.example.com.        259200  IN      A       10.0.2.101

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.example.com.

;; ADDITIONAL SECTION:
ns.example.com.         259200  IN      A       10.0.2.10

;; Query time: 1 msec
;; SERVER: 10.0.2.12#53(10.0.2.12)
;; WHEN: Tue Sep 29 11:17:51 EDT 2020
;; MSG SIZE  rcvd: 93
```

Dig Command Output:

Header Section: dig command version is 9.10.3

Question Section: www.example.com is the input given to dig. It uses A record and IN Class

Answer Section: www.example.com. is the output given to dig as answer. It uses A record, IN Class & has 259200 TTL.

Authority Section: example.com is the DNS name server that has the authority to respond to this query. Here, ns.example.com is the name server for example.com

Additional Section: 10.0.2.10 is the IP Address of the Name Server ns.example.com which is listed in the Authority Section.

Stats Section:

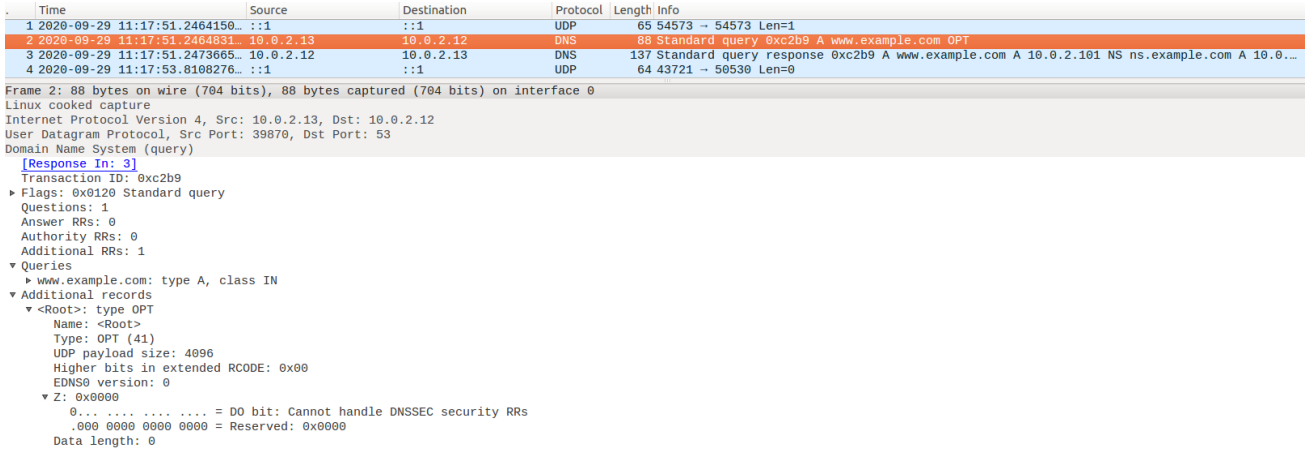
Query time: 1 msec

Server IP Address & Port: 10.0.2.12 (Port 53)

Time Stamp: Sept 29, 2020; 11:17:59 EDT

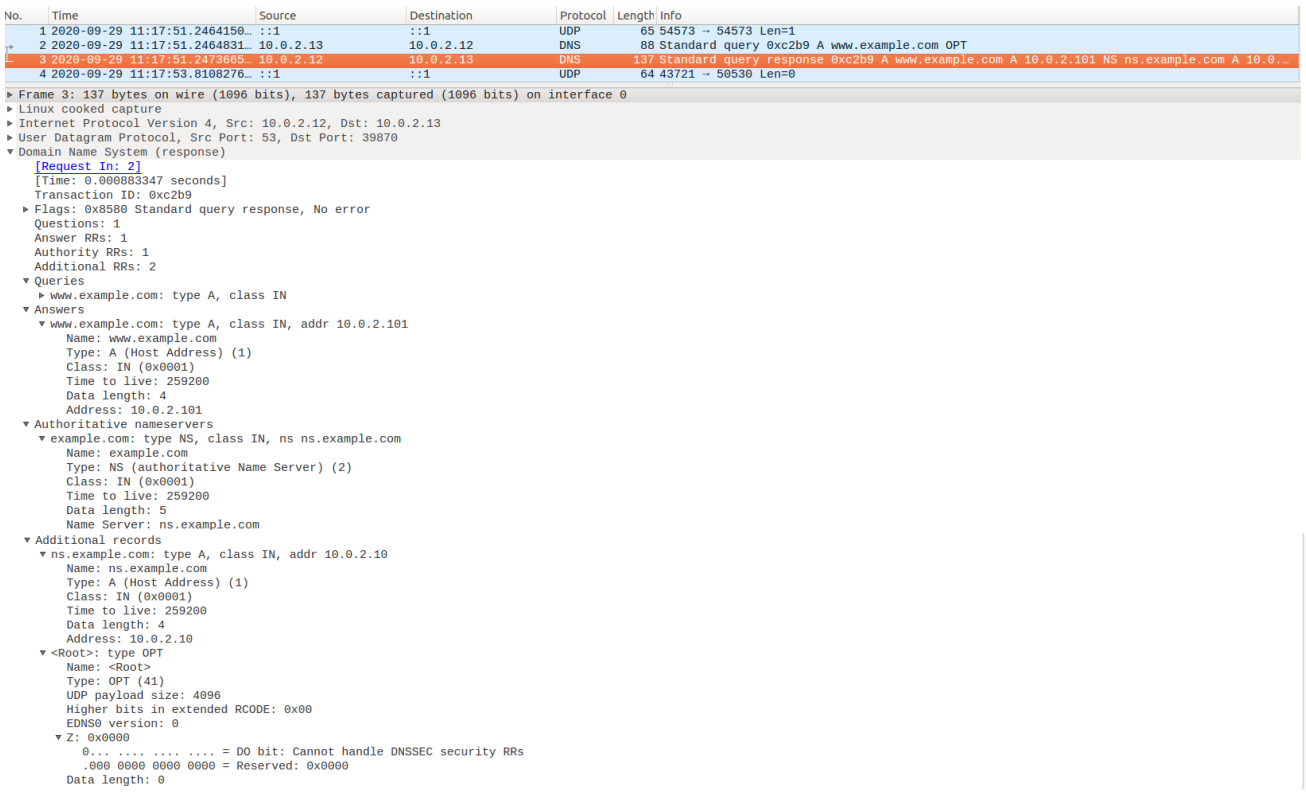
Message Size: rcvd: 93

DNS Query



Source IP Address	10.0.2.13
Destination IP Address	10.0.2.12
Protocol	DNS
Query	www.flipkart.com
Type	Standard Query Type A
Class	Class IN
Source Port	39870
Destination Port	53

DNS Response



Source IP Address	10.0.2.12
Destination IP Address	10.0.2.13
Protocol	DNS
Message	DNS Response
Source Port	53
Destination Port	3980

Here, using bind9, the Local DNS server was successfully set up and the DNS for www.example.com was also successfully resolved as seen by dig. Difference seen in Test 3 for flipkart.com & dig for example.com is that TTL is higher in dig example.com and Authoritative Name Servers in dig is taken from bind resolvconf dbs – example.com.db and 10.0.2.db that we modified.

Answers:

Name	www.example.com
Type	A (Host Address) (1)
Class	IN (0x0001)
TTL	252900
Data Length	4
Address	10.0.2.101

Authoritative nameservers

Name	example.com
Type	NS (Authoritative Name Server) (2)
Class	IN (0x0001)
TTL	252900
Data Length	5
Name Server	ns.example.com

Additional Records

Name	ns.example.com
Type	A (Host Address) (1)
Class	IN (0x0001)
TTL	252900
Data Length	4
Address	10.0.2.10

Local DNS cache for example.com records when grep is used is empty

```
ubuntu1@PES1201800046:~$sudo cat /var/cache/bind/dump.db | grep "example"
ubuntu1@PES1201800046:~$
```

CONCLUSION:

- DNS (Domain Name System) is the Internet's phone book, it translates hostnames to IP addresses and vice versa.
- Installed & set-up DNS server. Learnt about its functionalities & operations.

Observations for **$ ping www.flipkart.com -c 1**

1) Locate the DNS query and response messages. Are then sent over UDP or TCP?

A) DNS query and responses are attached as screenshots in the tasks/tests above.

The DNS query & response messages are sent as UDP.

2) What is the destination port for the DNS query message? What is the source port of DNS response message?

A) **Destination port for DNS query: 53**

Source port for DNS response: 53

3) To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?

A) DNS query message is sent to 10.0.2.12. When checked with ifconfig (screenshots are attached above), DNS server's IP address is also 10.0.2.12.

Yes, both the IP address sent in the DNS query and the IP address of DNS server are the same (10.0.2.12).

4) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

A) DNS Query message under "Queries section": www.flipkart.com : type A, class IN

The DNS query message is a type "A Standard Query" (where name is host name and value is IP address) and of "IN Class" (Class value of IN in resource records – 1 the Internet). No, the query does not contain any "Answers section".

5) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

A) DNS Response message, under "Answers section":

www.flipkart.com: type CNAME, class IN, cname flipkart.com
- Name: www.flipkart.com
- Type: CNAME (Canonical NAME for an alias) (5)
- CLASS: IN (0x0001)
- Time to live: 4
- Data length: 2
- CNAME: flipkart.com

www.flipkart.com: type CNAME, class IN, addr 163.53.76.86
- Name: www.flipkart.com
- Type: A (Host Address) (5)
- CLASS: IN (0x0001)
- Time to live: 1
- Data length: 4
- Address: 163.53.76.86

Two answers were provided in answers section of the DNS response message. Each of these answers contain information of the host, the type of address, class, the TTL, the data length and the IP address.

Type CNAME has name as the alias for its canonical name and value as the canonical name itself

Type A has name as the hostname and value as IP address

6) Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?

A) **Yes, the first TCP SYN / ICMP packet was sent to 163.53.76.86 (Destination IP address of the packet) which also corresponds to the first IP address provided in the DNS response message.**

Week #5

Simple Client-Server Application using Network Socket Programming

(No Instructor-Led Training Lab)

Objective:

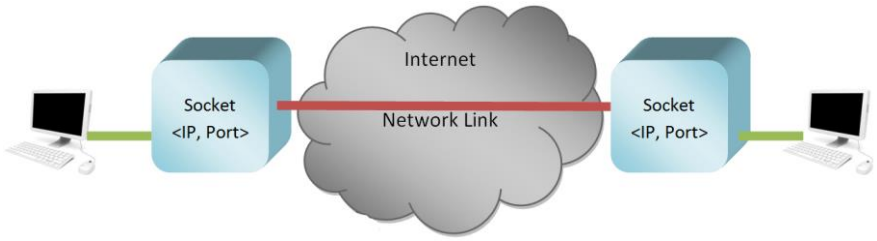
To develop a simple Client-Server application using TCP and UDP.

Pre requisites:

- Basic understanding of networking concepts and socket programming
- Knowledge of python

Sockets

Sockets are just the **endpoints of a two-way communication link** in a network. Socket helps in the communication of two processes/programs on a network (eg. Internet). The programs can communicate by reading/writing via their sockets. A socket comprises of: ***IP Address & Port number***



Task 1: (Mandatory for all students)

1. Create an application that will
    a. Convert lowercase letters to uppercase
        - e.g. [a…z] to [A…Z]
        - code will not change any special characters, e.g. &*!
    b. If the character is in uppercase, the program must not alter
2. Create Socket API both for client and server.
3. Must take the server address and port from the Command Line Interface (CLI).

Socket Programming with UDP

UDPClient.py

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

UDPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
message, clientAddress = serverSocket.recvfrom(2048)
modifiedMessage = message.upper()
serverSocket.sendto(modifiedMessage, clientAddress)
```

Socket Programming with TCP

TCPClient.py

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
```

```
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024)
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence)
connectionSocket.close()
```

Problems:

Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

1. Suppose you run TCPClient before you run TCPServer. What happens? Why?
2. Suppose you run UDPClient before you run UDPServer. What happens? Why?
3. What happens if you use different port numbers for the client and server sides?

Any one of the below tasks is mandatory for Week-5 completion and will be assigned by faculty members.

Task 2: Web Server

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

a) create a connection socket when contacted by a client (browser);
b) receive the HTTP request from this connection;

c) parse the request to determine the specific file being requested;
d) get the requested file from the server's file system;
e) create an HTTP response message consisting of the requested file preceded by header lines; and
f) send the response over the TCP connection to the requesting browser.

If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

For this assignment, the companion Web site provides the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web server.

Task 3: Multi-Threaded Web Proxy

In this assignment, you will develop a Web proxy. When your proxy receives an HTTP request for an object from a browser, it generates a new HTTP request for the same object and sends it to the origin server. When the proxy receives the corresponding HTTP response with the object from the origin server, it creates a new HTTP response, including the object, and sends it to the client. This proxy will be multi-threaded, so that it will be able to handle multiple requests at the same time.

For this assignment, the companion Web site provides the skeleton code for the proxy server. Your job is to complete the code, and then test it by having different browsers request Web objects via your proxy.

Task 4: Mail Client

The goal of this programming assignment is to create a simple mail client that sends email to any recipient. Your client will need to establish a TCP connection with a mail server (e.g., a Google mail server), dialogue with the mail server using the SMTP protocol, send an email message to a recipient (e.g., your friend) via the mail server, and finally close the TCP connection with the mail server.

For this assignment, the companion Web site provides the skeleton code for your client. Your job is to complete the code and test your client by sending email to different user accounts. You may also try sending through different servers (for example, through a Google mail server and through your university mail server).

Implementation Procedure:

The above-mentioned tasks can be implemented in several ways. They are,

1) Using 2 terminals
2) Using virtual machines
3) Connecting two machines using switch and cables (in University lab)

Any one method should be used for executing the above tasks.

Edmodo Submissions:

- Client program for datagram service (UDP)
- Server program for datagram service (UDP)
- Client program for reliable byte-stream service (TCP)
- Server program for reliable byte-stream service (TCP)
- Screenshots of your test cases (terminal)
- Wireshark screenshots for all the communications

Plagiarism Warning:

Make sure your terminal name is changed as per your SRN to identify plagiarized contents.

Computer Networks Laboratory – UE18CS304

PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D

Week #5 – Network Socket Programming – 7th October 2020

OBJECTIVE: To develop a simple Client-Server application using TCP and UDP

SET-UP:

Client Machine – ubuntu1 (IP Address – 10.0.2.12)

```
ubuntu1@PES1201800046:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:73:bc:c2
          inet addr:10.0.2.12  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::e5cb:bb29:941d:676/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:401 errors:0 dropped:0 overruns:0 frame:0
          TX packets:461 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:122691 (122.6 KB)  TX bytes:40412 (40.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:24447 (24.4 KB)  TX bytes:24447 (24.4 KB)

ubuntu1@PES1201800046:~$
```

Server Machine – ubuntu2 (IP Address 10.0.2.13)

```
ubuntu2@PES1201800046:~$ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:23:02:14
          inet addr:10.0.2.13  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::ff93:3e8e:49ca:f8db/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:92 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:54502 (54.5 KB)  TX bytes:15528 (15.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:21812 (21.8 KB)  TX bytes:21812 (21.8 KB)

ubuntu2@PES1201800046:~$
```

TASK 1:

1) To create an application that:
- Converts lowercase letters to uppercase letters, ex: [a … z] to [A … Z]
- But doesn't change any special characters [&*!]
- Also, doesn't alter the uppercase characters

2) To create Socket API for both Client & Server, that takes server address & port from CLI

CLI Output for TCP Connection

Client Machine

```
ubuntu1@PES1201800046:~$python3 TCPClient.py

Enter Server Name: 10.0.2.13
Enter Server Port: 12000

Enter a sentence: Hello, this is CN Lab Week #5!!!
Modified Sentence from Server:  HELLO, THIS IS CN LAB WEEK #5!!!
```

Server Machine

```
ubuntu2@PES1201800046:~$python3 TCPServer.py

The TCP Server is ready to receive

Connected with address 10.0.2.12
Recieved data from socket connection
Sent data to socket connection
```

Wireshark Captures for TCP Connection

SYN

No.	Time	Source	Destination	Protocol	Length	Info
2	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	76	54232 → 12000 [SYN] Seq=849913013 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSv…
3	2020-10-07 06:27:35.…	10.0.2.13	10.0.2.12	TCP	76	12000 → 54232 [SYN, ACK] Seq=3277403899 Ack=849913014 Win=28960 Len=0 MSS=…
4	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=0 TSval=114…
10	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	100	54232 → 12000 [PSH, ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=32 TSv…
11	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=0 TSval=114…
12	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	100	12000 → 54232 [PSH, ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=32 TSv…
13	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913046 Ack=3277403932 Win=29312 Len=0 TSval=114…
14	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [FIN, ACK] Seq=3277403932 Ack=849913046 Win=29056 Len=0 TSva…
15	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [FIN, ACK] Seq=849913046 Ack=3277403933 Win=29312 Len=0 TSva…
16	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403933 Ack=849913047 Win=29056 Len=0 TSval=114…

```
▶ Frame 2: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.0.2.12, Dst: 10.0.2.13
▼ Transmission Control Protocol, Src Port: 54232, Dst Port: 12000, Seq: 849913013, Len: 0
    Source Port: 54232
    Destination Port: 12000
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 849913013
    Acknowledgment number: 0
    Header Length: 40 bytes
  ▼ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ▶ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ··········S·]
    Window size value: 29200
    [Calculated window size: 29200]
    Checksum: 0x1847 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
```

SYN – ACK

No.	Time	Source	Destination	Protocol	Length	Info
2	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	76	54232 → 12000 [SYN] Seq=849913013 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSv…
3	2020-10-07 06:27:35.…	10.0.2.13	10.0.2.12	TCP	76	12000 → 54232 [SYN, ACK] Seq=3277403899 Ack=849913014 Win=28960 Len=0 MSS=…
4	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=0 TSval=114…
10	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	100	54232 → 12000 [PSH, ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=32 TSv…
11	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=0 TSval=114…
12	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	100	12000 → 54232 [PSH, ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=32 TSv…
13	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913046 Ack=3277403932 Win=29312 Len=0 TSval=114…
14	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [FIN, ACK] Seq=3277403932 Ack=849913046 Win=29056 Len=0 TSva…
15	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [FIN, ACK] Seq=849913046 Ack=3277403933 Win=29312 Len=0 TSva…
16	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403933 Ack=849913047 Win=29056 Len=0 TSval=114…

```
▶ Frame 3: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.0.2.13, Dst: 10.0.2.12
▼ Transmission Control Protocol, Src Port: 12000, Dst Port: 54232, Seq: 3277403899, Ack: 849913014, Len: 0
    Source Port: 12000
    Destination Port: 54232
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3277403899
    Acknowledgment number: 849913014
    Header Length: 40 bytes
  ▼ Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ▶ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A··S·]
    Window size value: 28960
    [Calculated window size: 28960]
    Checksum: 0xf772 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 2]
      [The RTT to ACK the segment was: 0.000539317 seconds]
      [iRTT: 0.000562051 seconds]
```

ACK

No.	Time	Source	Destination	Protocol	Length	Info
2	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	76	54232 → 12000 [SYN] Seq=849913013 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSv…
3	2020-10-07 06:27:35.…	10.0.2.13	10.0.2.12	TCP	76	12000 → 54232 [SYN, ACK] Seq=3277403899 Ack=849913014 Win=28960 Len=0 MSS=…
4	2020-10-07 06:27:35.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=0 TSval=114…
10	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	100	54232 → 12000 [PSH, ACK] Seq=849913014 Ack=3277403900 Win=29312 Len=32 TSv…
11	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=0 TSval=114…
12	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	100	12000 → 54232 [PSH, ACK] Seq=3277403900 Ack=849913046 Win=29056 Len=32 TSv…
13	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [ACK] Seq=849913046 Ack=3277403932 Win=29312 Len=0 TSval=114…
14	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [FIN, ACK] Seq=3277403932 Ack=849913046 Win=29056 Len=0 TSva…
15	2020-10-07 06:28:03.…	10.0.2.12	10.0.2.13	TCP	68	54232 → 12000 [FIN, ACK] Seq=849913046 Ack=3277403933 Win=29312 Len=0 TSva…
16	2020-10-07 06:28:03.…	10.0.2.13	10.0.2.12	TCP	68	12000 → 54232 [ACK] Seq=3277403933 Ack=849913047 Win=29056 Len=0 TSval=114…

```
▶ Frame 4: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.0.2.12, Dst: 10.0.2.13
▼ Transmission Control Protocol, Src Port: 54232, Dst Port: 12000, Seq: 849913014, Ack: 3277403900, Len: 0
    Source Port: 54232
    Destination Port: 12000
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 849913014
    Acknowledgment number: 3277403900
    Header Length: 32 bytes
  ▼ Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x183f [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 3]
      [The RTT to ACK the segment was: 0.000022734 seconds]
      [iRTT: 0.000562051 seconds]
```

TCP Connection Inferences:

TCP Connection is a 3-way handshake consists of 4 important processes – SYN, SYN-ACK, ACK, FIN.

In the first step Client (Source IP Address – 10.0.2.12) establishes a connection with the Server (Destination IP Address – 10.0.2.13) and port 12000 using SYN signal.

In the second step, the Server (Source IP Address – 10.0.2.13) from port 12000 responds to the Client (Destination IP Address – 10.0.2.12) with SYN-ACK signal-set.

In the final step, the Client (Source IP Address – 10.0.2.12) acknowledges the response to the client request with ACK signal.

Next the Client sends the data sentence to the Server, the Server then modifies the data to an upper case sentence and sends it back to the Client.

After the modified message is received on Client, TCP automatically terminates the connection between the Client and Server with FIN signal.

CLI Output for UDP Connection

Client Machine

```
ubuntu1@PES1201800046:~$python3 UDPClient.py

Enter Server Name: 10.0.2.13
Enter Server Port: 12000

Enter a sentence: Hello, this is CN Lab Week #5!!!
Modified Sentence from Server:  HELLO, THIS IS CN LAB WEEK #5!!!
```

Server Machine

```
ubuntu2@PES1201800046:~$python3 UDPServer.py

The UDP Server is ready to receive

Recieved data from socket connection with address 10.0.2.12
Sent data to socket connection
```

UDP Connection Inferences

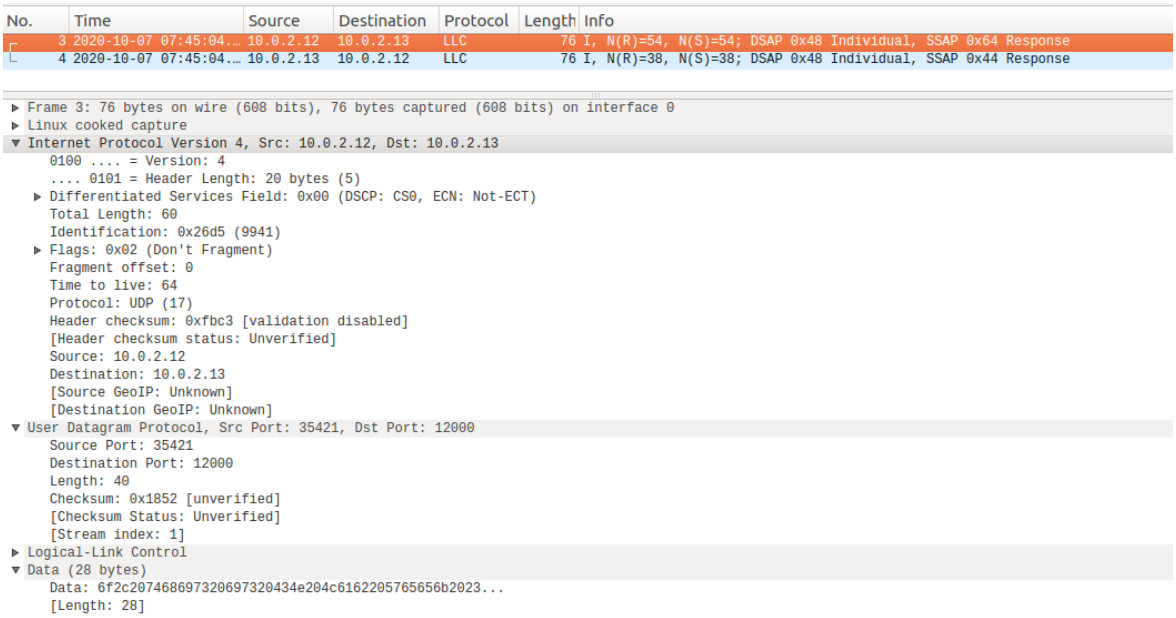
There is no handshaking before sending data in UDP Connection.

The sender, here the Client (Source IP Address – 10.0.2.12) explicitly attaches the IP destination address (10.0.2.13) and port number (12000) to each packet with the packet data and sends it to the receiver.

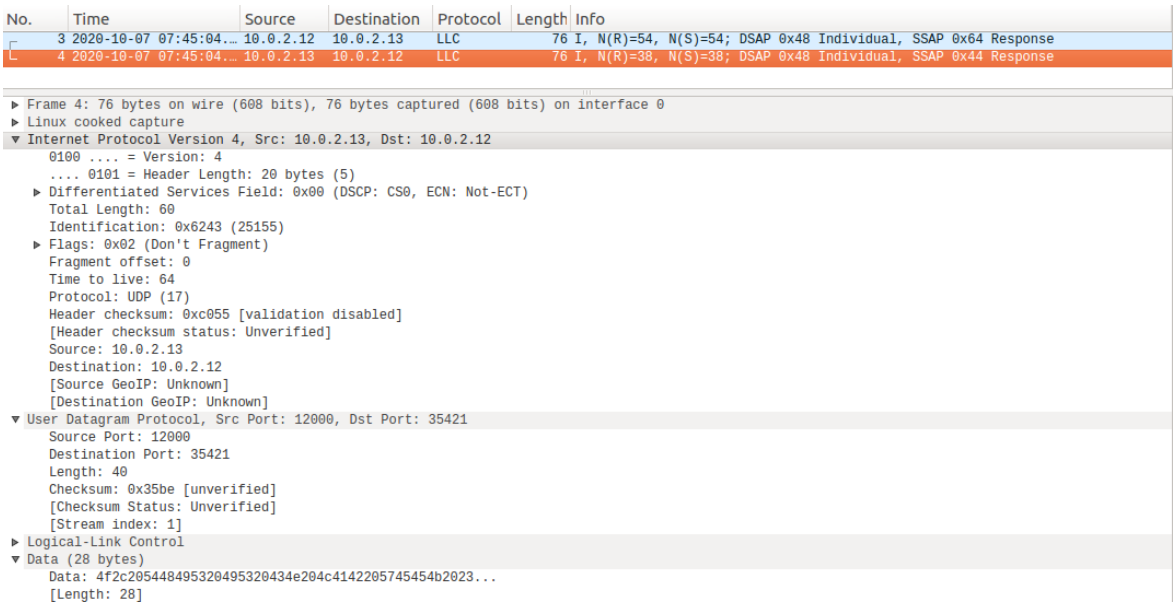
The receiver, here the Server (Source IP Address – 10.0.2.13) extracts sender IP address (10.0.2.12) and port number (12000) and data from received packet converts it to uppercase and sends it back to the sender.

Wireshark Captures for UDP Connection

UDP Request



UDP Response



1) Suppose you run TCPClient before you run TCPServer. What happens? Why?

If we run TCP Client first, the client tries to make a connection with a server process that doesn't exist. Hence Connection Refused Error is returned because the connection with the TCP server could not be made.

2) Suppose you run UDPClient before you run UDPServer. What happens? Why?

If we run UDPClient first but don't enter any sentence, the UDPClient doesn't try to establish a connection like the TCPClient with the server. So, no error is returned. When we enter a sentence afterwards, the UDPServer receives it, performs the uppercase modifications and returns the modified sentence back to the UDPClient to be displayed.

3) What happens if you use different port numbers for the client and server sides?

If we use different port numbers, the client will attempt to make a TCP connection with a non-existent process or wrong process and errors will be returned.

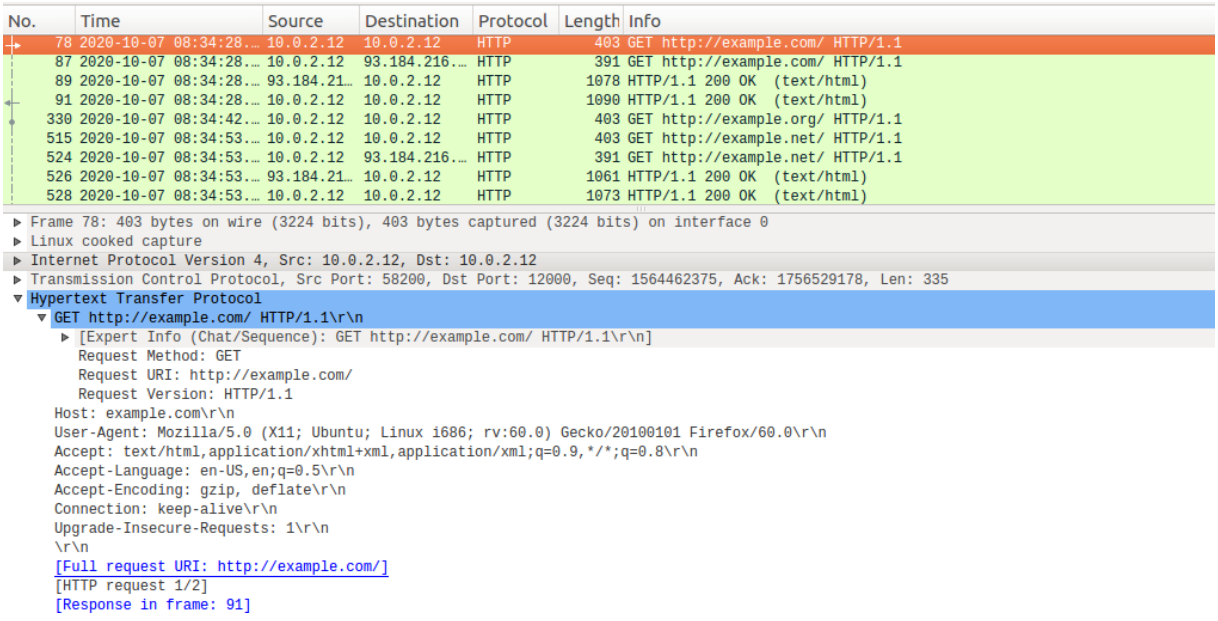
TASK 3: (Roll Number: 2)

To develop a multi-threaded Web Proxy to handle multiple HTTP requests at the same time. When the proxy receives a corresponding HTTP response with the object from the original server, it should create a new HTTP response with a new object and send it to the client.

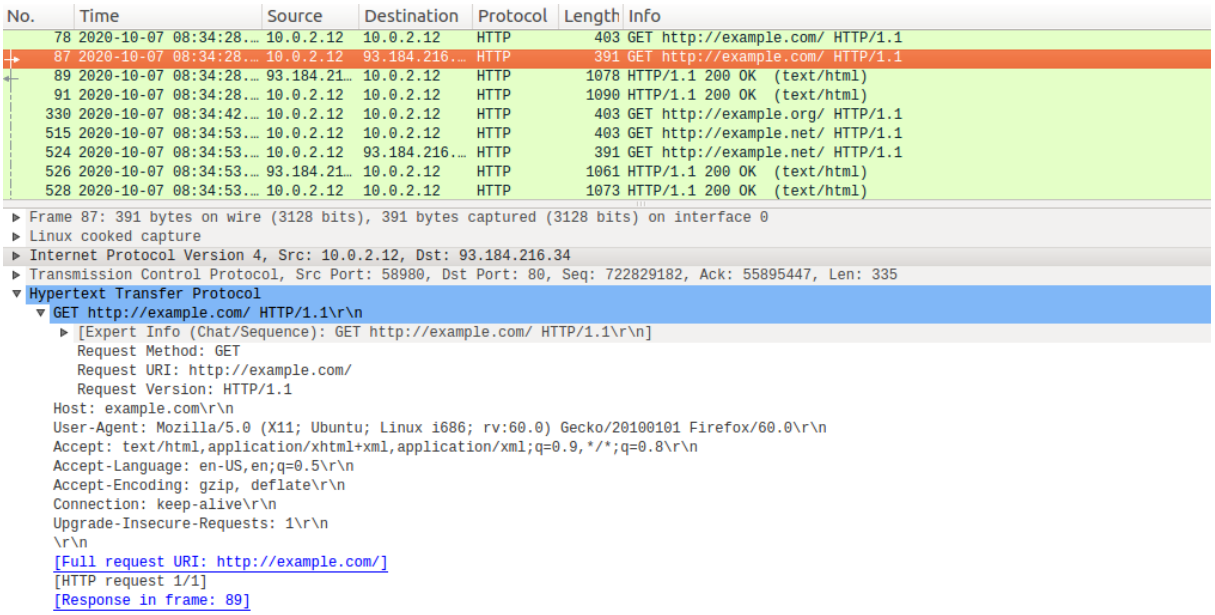
CLI Output for Proxy Server

```
ubuntu1@PES1201800046:~$python3 ProxyServer.py 12000

Proxy server Running on : 12000


Request          10.0.2.12       GET http://example.com/ HTTP/1.1
Response         10.0.2.12       HTTP/1.1 200 OK

Request          10.0.2.12       GET http://example.net/ HTTP/1.1
Response         10.0.2.12       HTTP/1.1 200 OK
```

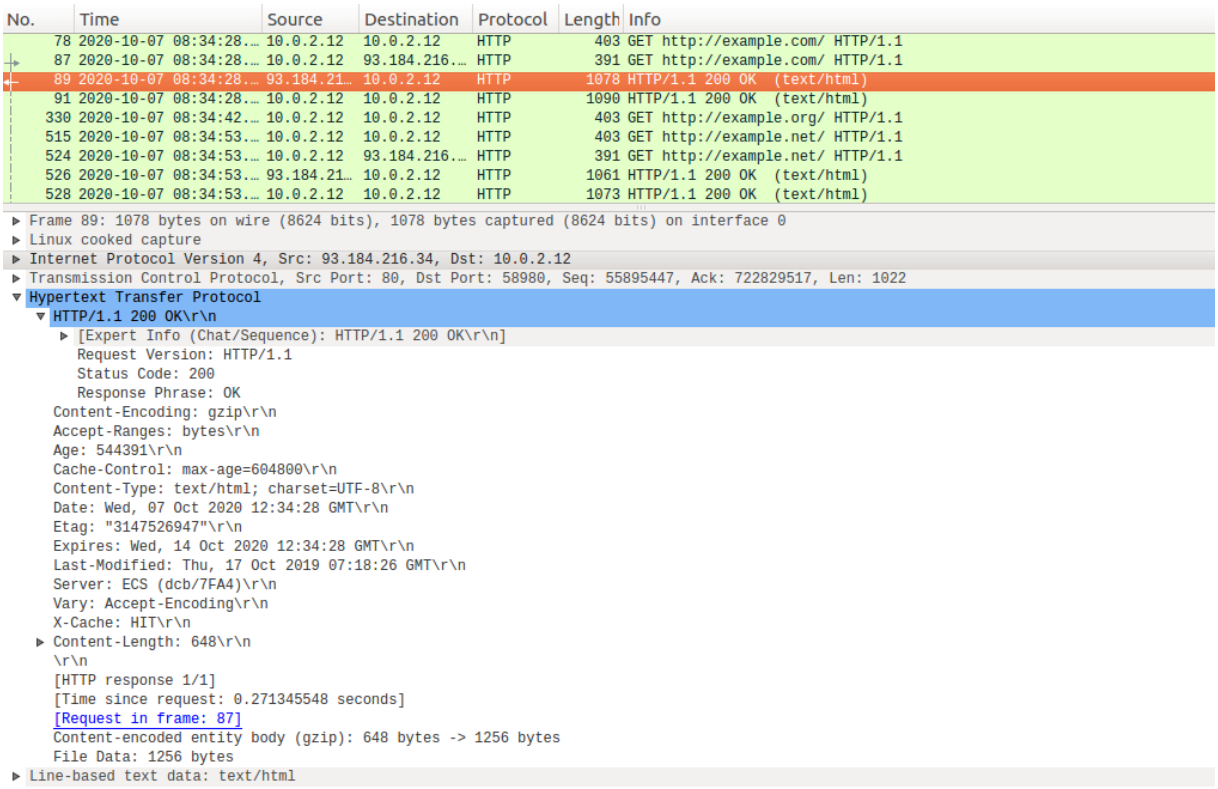
Client (Source IP Address – 10.0.2.12) Request to Proxy Server (Destination IP Address – 10.0.2.12, Port 12000)



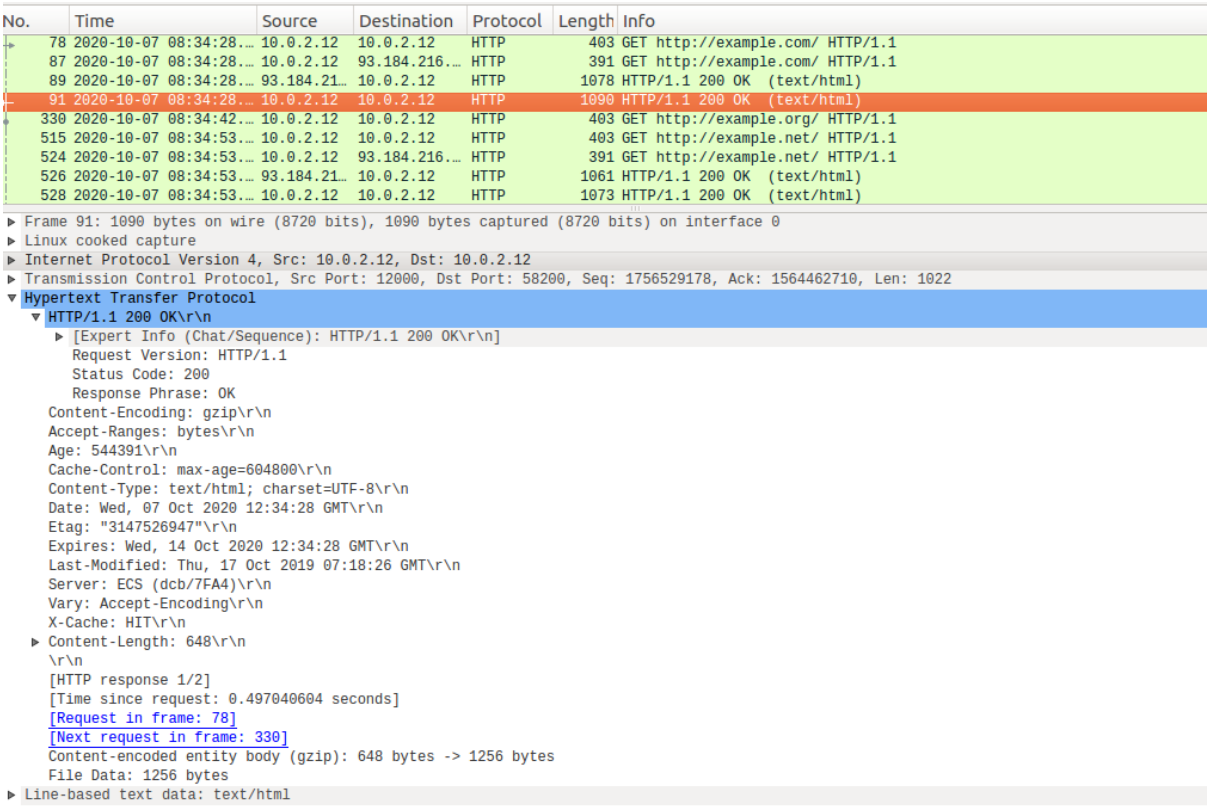
Proxy Server (Source IP Address – 10.0.2.12) Request to example.com Server (Destination IP Address – 93.184.21.34, Port 80)



example.com Server (Source IP Address – 93.184.216.34, Port 80) Response to Proxy Server (Destination IP Address – 10.0.2.12)



Proxy Server (Source IP Address – 10.0.2.12, Port 12000) Response to Client (Destination IP Address – 10.0.2.12)



CONCLUSION:

- Sockets comprise of IP Address and Port Number.
- Sockets are the end points of a two-way communication link of two processes/programs on a network. These programs communicate via reading and writing from their sockets.
- TCP is a reliable socket transport with 3-way (SYN, SYN-ACK, ACK) handshake connection which is byte-streamed oriented.
- UDP is an unreliable socket transport connection which is uses datagrams.
- Proxy server is a web cache that acts as both client and server which can reduce response time for client requests.

Week #6

Designing and Simulation of Network Topology using Cisco Packet Tracer

Objectives:

- To understand the purpose of Cisco Packet Tracer.
- To navigate, choose network and end devices and customize them.
- To interconnect devices and configure them using simple interface.
- To become familiar with building topologies in Packet Tracer.
- To simulate data interactions traveling through a network.

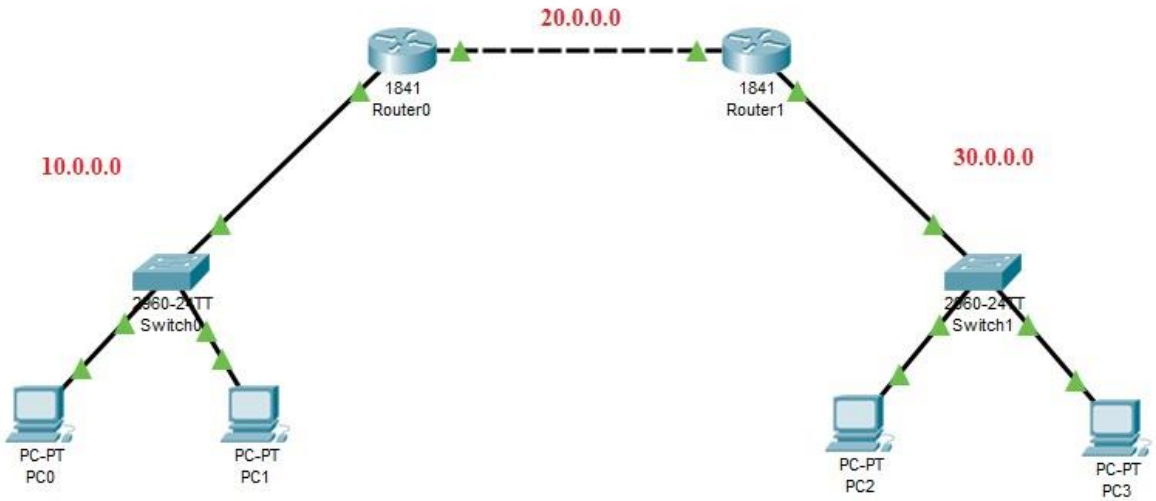
Prerequisites:

This lab assumes some understanding of the building blocks of communication networks and internet. At this point, we haven't discussed other protocols but you may use Packet Tracer in later labs to discuss those as well. Several types of devices and network connections can be used. For this experiment we will keep it simple by using end devices, switches, routers, and connections.

Task 1 (Demo)

Network Topology:

To replicate given scenario, create a topology in packet tracer, as shown in following image.



PC & Router Configuration Details:

PC0:

IP Address ---> 10.0.0.1

Gateway ---> 10.0.0.3

PC1:

IP Address ---> 10.0.0.2

Gateway ---> 10.0.0.3

Router 0:

FastEthernet0/0 ---> 10.0.0.3

FastEthernet0/1 ---> 20.0.0.1

Router 1:

FastEthernet0/0 ---> 20.0.0.2

FastEthernet0/1 ---> 30.0.0.1

PC2:

IP Address ---> 30.0.0.2

Gateway ---> 30.0.0.1

PC3:

IP Address ---> 30.0.0.3

Gateway ---> 30.0.0.1

Routing Table Entries:

Router	Network	Next Hop
Router 0	30.0.0.0	20.0.0.2
Router 1	10.0.0.0	20.0.0.1

Execution Procedure:

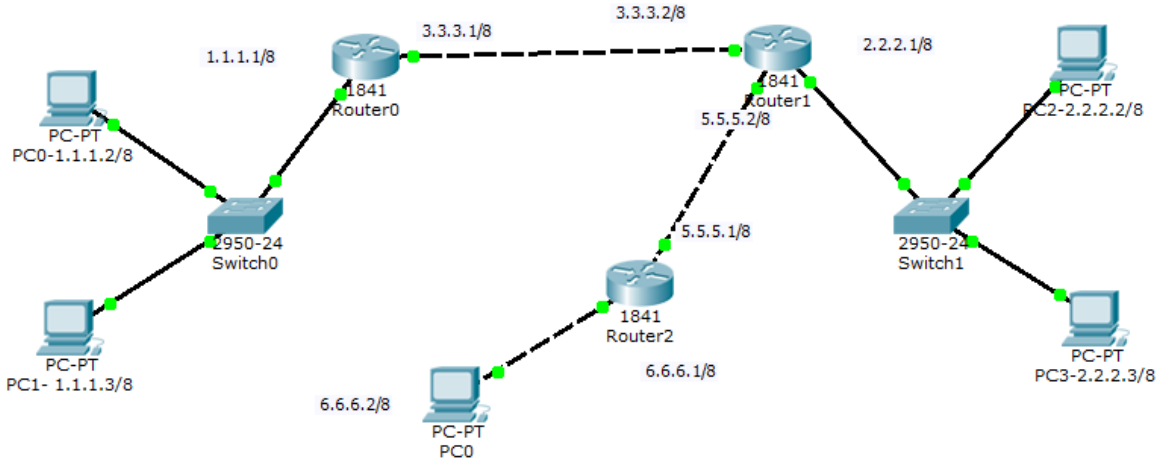
Task 1: Design a network topology with desktops, switches and routers similar to the network depicted in the above diagram.

Task 2: Configure the PCs and routers with the details provided above.

Task 3: Send a simple PDU from any PC on network 10.0.1.0 to any other PC on other network 10.0.3.0 and vice-versa.

Task 4: Simulate the network and observe the packet flow from one network to other.

Task 2 (Mandatory for Week-6)



Computer Systems Laboratory – UE18CS304

PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D – 27th October 2020

Week #6 – Designing & Simulation of Network Topology using Cisco Packet Tracer

Objectives:

- To understand the purpose of Cisco Packet Tracer.
- To navigate, choose network and end devices and customize them.
- To interconnect devices and configure them using simple interface.
- To become familiar with building topologies in Packet Tracer.
- To simulate data interactions traveling through a network.

Task 1

PC Configuration Details:

PC	IPv4 Address	Gateway
PC0	10.0.0.2	10.0.0.1
PC1	10.0.0.3	10.0.0.1
PC2	30.0.0.2	30.0.0.1
PC3	30.0.0.3	30.0.0.1

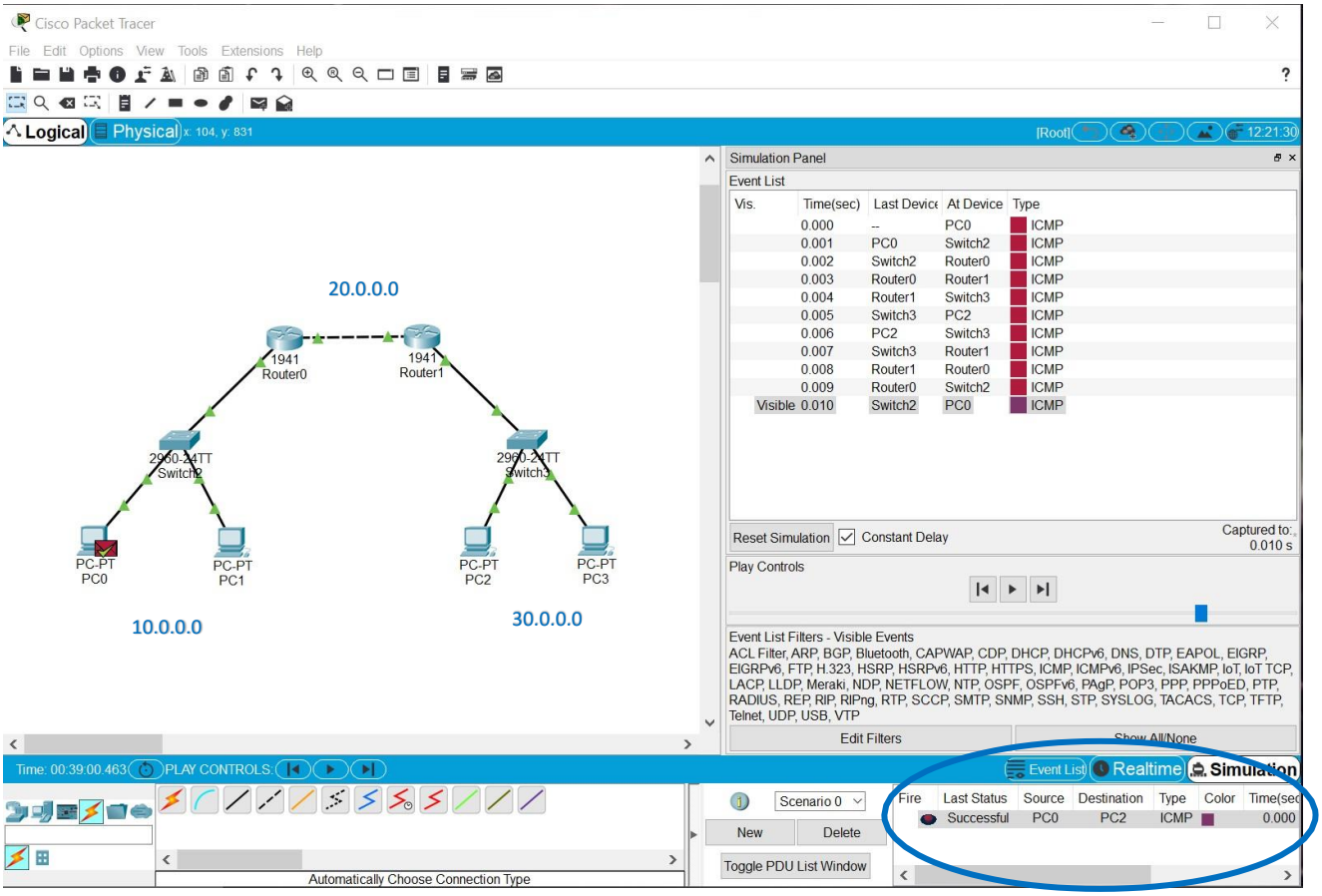
Router Configuration Details:

Router	FastEthernet0/0	FastEthernet0/1
Router0	10.0.0.1	20.0.0.1
Router1	30.0.0.1	20.0.0.2

Routing Table Entries:

Router	Network	Next Hop
Router0	30.0.0.0	20.0.0.2
Router1	10.0.0.0	20.0.0.1

Packet Transmission Simulation from PC0 to PC2



Task 2

PC Configuration Details:

PC	IPv4 Address	Gateway
PC0	1.1.1.2	1.1.1.1
PC1	1.1.1.3	1.1.1.1
PC2	2.2.2.2	2.2.2.1
PC3	2.2.2.3	2.2.2.1
PC4	6.6.6.2	6.6.6.1

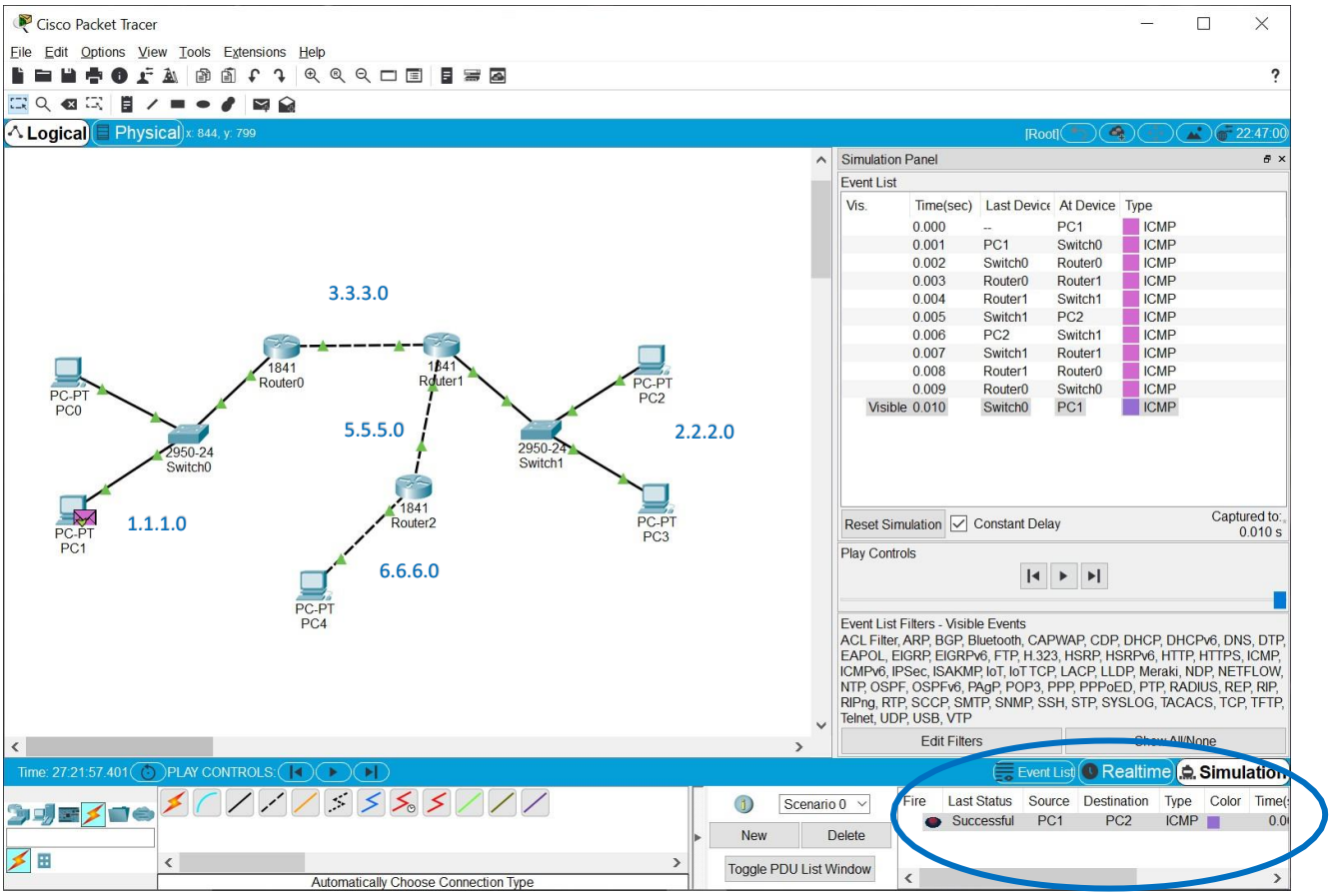
Router Configuration Details:

Router	FastEthernet0/0	FastEthernet0/1	Ethernet0/1/0
Router0	1.1.1.1	3.3.3.1	-
Router1	2.2.2.1	3.3.3.2	5.5.5.2
Router2	6.6.6.1	5.5.5.1	-

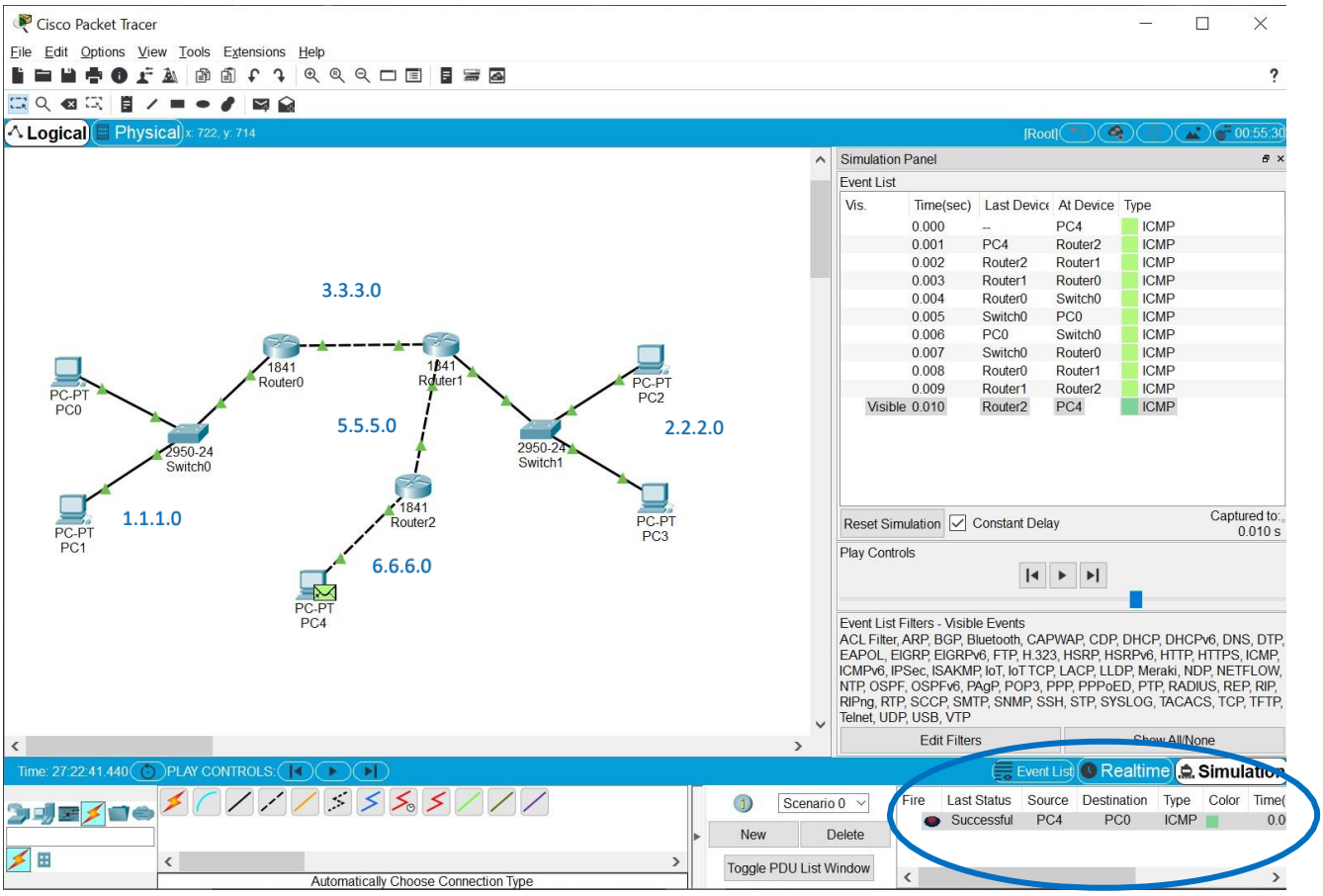
Routing Table Entries:

Router	Network	Next Hop
Router0	2.2.2.0	3.3.3.2
Router0	5.5.5.0	3.3.3.2
Router0	6.6.6.0	3.3.3.2
Router1	1.1.1.0	3.3.3.1
Router1	6.6.6.0	5.5.5.1
Router2	1.1.1.0	5.5.5.2
Router2	2.2.2.0	5.5.5.2
Router2	3.3.3.0	5.5.5.2

Packet Transmission Simulation from PC1 to PC2



Packet Transmission Simulation from PC4 to PC0



Conclusion:

- Cisco Packet Tracer helps us to learn about the principles of networking while implementing the protocols with Cisco routers, switches and other networking devices in the software.
- Understood how to build and configure different topologies along with routing table entries.
- Understood how data simulations & interactions travel through a network.

WEEK 7

EXCERCISE 1:
Using Cisco packet tracer understand the life of packet in internet.

Create the following topology in packet tracer.

```
            /---DNS
A – R1—R2
            \--- Web Server
```

Open the browser in A and access the webserver using sitename (not using IP Address). Traverse each packet (in simulation mode) and answer the following for each packet

Src IP, Dstn IP, Src Mac, Dstn MAC, pkt type (e.g. DNS, ARP, HTTP, TCP)

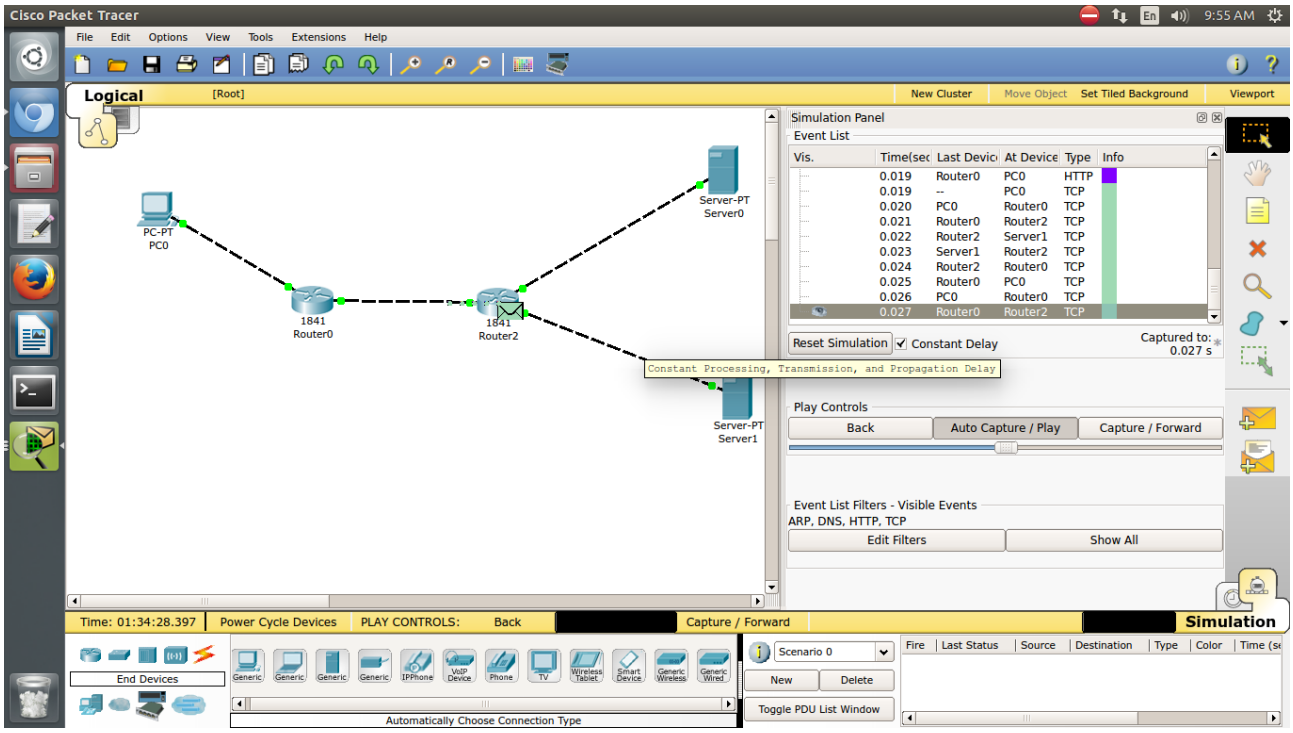
Observation: Does the number of packets traversed in the network change with second invocation of web request.

Experiment : Understanding the life of packet in internet.

Components Used : PC-Devices, DNS server, Web Server, Routers (everything on cisco packet tracer)

Topology Created :

```
-------        --------        --------        |-------------->DNS SERVER
|     |        |      |        |      |        |
|HOST |        |ROUTER|        |ROUTER|<-------|
| A   |<------>|  1   |<------>|  2   |
|     |        |      |        |      |<-------|
-------        ----|--         --------        |
                                                |-------------->WEB SERVER
```



CONFIGURATIONS :

HOST A : IP Address ---> 10.10.1.1
Gateway ------> 10.10.1.2
DNS Server ---> 192.168.1.2

ROUTER 1 : Incoming Interface IP --> 10.10.1.2 (Fast ethernet 0)
OUtgoing Interface IP --> 10.10.2.1 (Fast ethernet 1)

ROUTER 2 : Incoming Interface IP --> 10.10.2.2 (Fast ethernet 0)
OUtgoing Interface1 IP --> 192.168.1.1 (Fast ethernet 1)
Outgoing Interface2 IP --> 192.168.2.1 (External added interface)

DNS Server : IP Address ----> 192.168.1.2
Default Gateway : 192.168.1.1

WEB Server : IP Address ----> 192.168.2.2
Default Gateway : 192.168.2.1

ROUTING TABLE ENTRIES :

Router name	Network	Gateway
ROUTER 1	192.168.1.0	10.10.2.2
ROUTER 1	192.168.2.0	10.10.2.2
ROUTER 2	10.10.1.0	10.10.2.1

STEPS OF EXECUTION :

1. Firstly the topolgy was constructed and configured using the above details.
2. While configuing the DNS server (with the above information), a type-A record was also added :

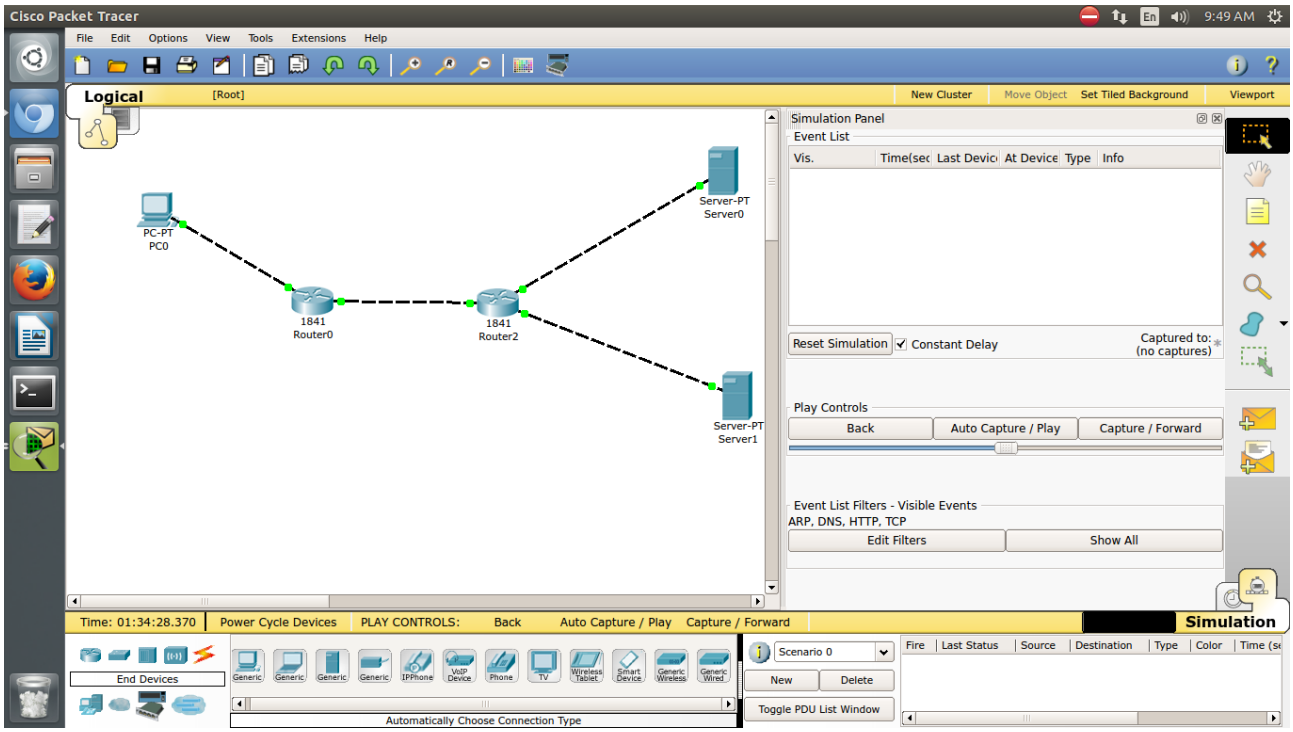
Record-type : Type-A
Name : google.com (NAME OF THE DOMAIN)
Address : IP address of web-server i.e. 192.168.2.2 (DOMAIN'S IP Address)

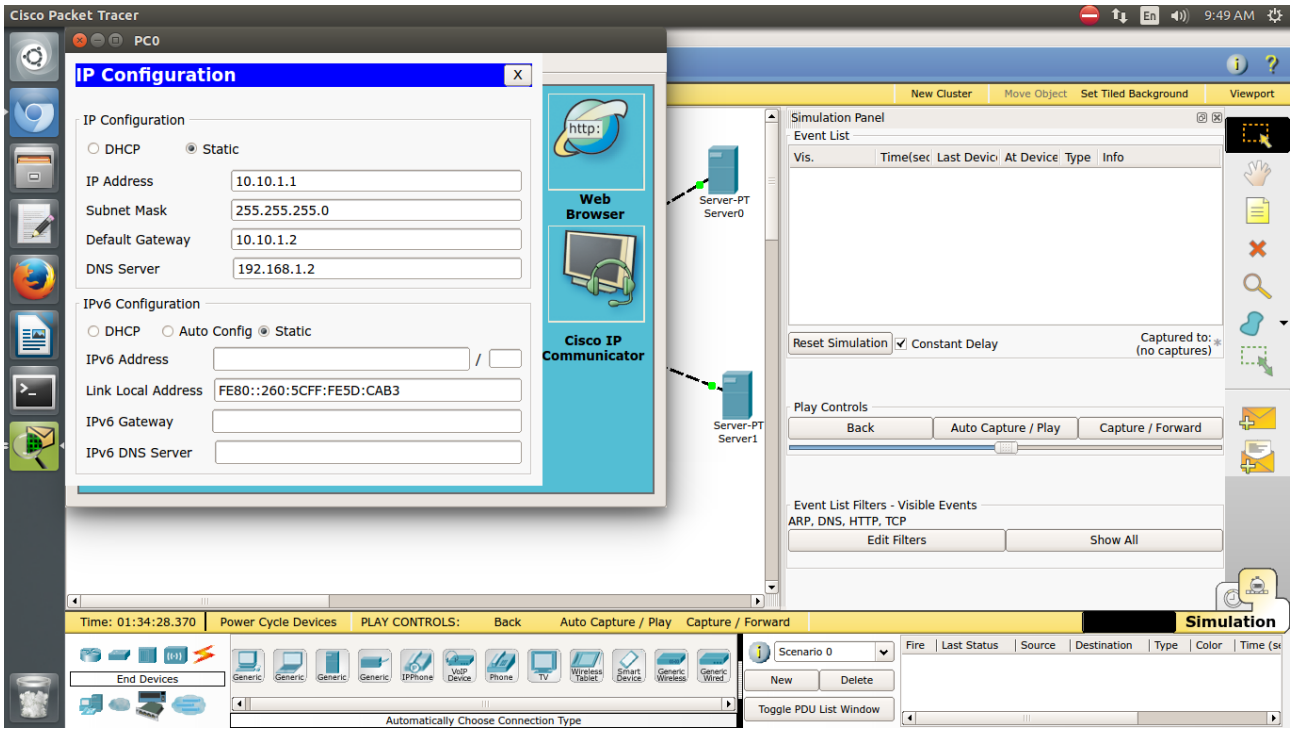
3. While configuring the Web Server (with the above information), the HTML page in the HTTP config information
is checked and we can add information over there to see the output over there.

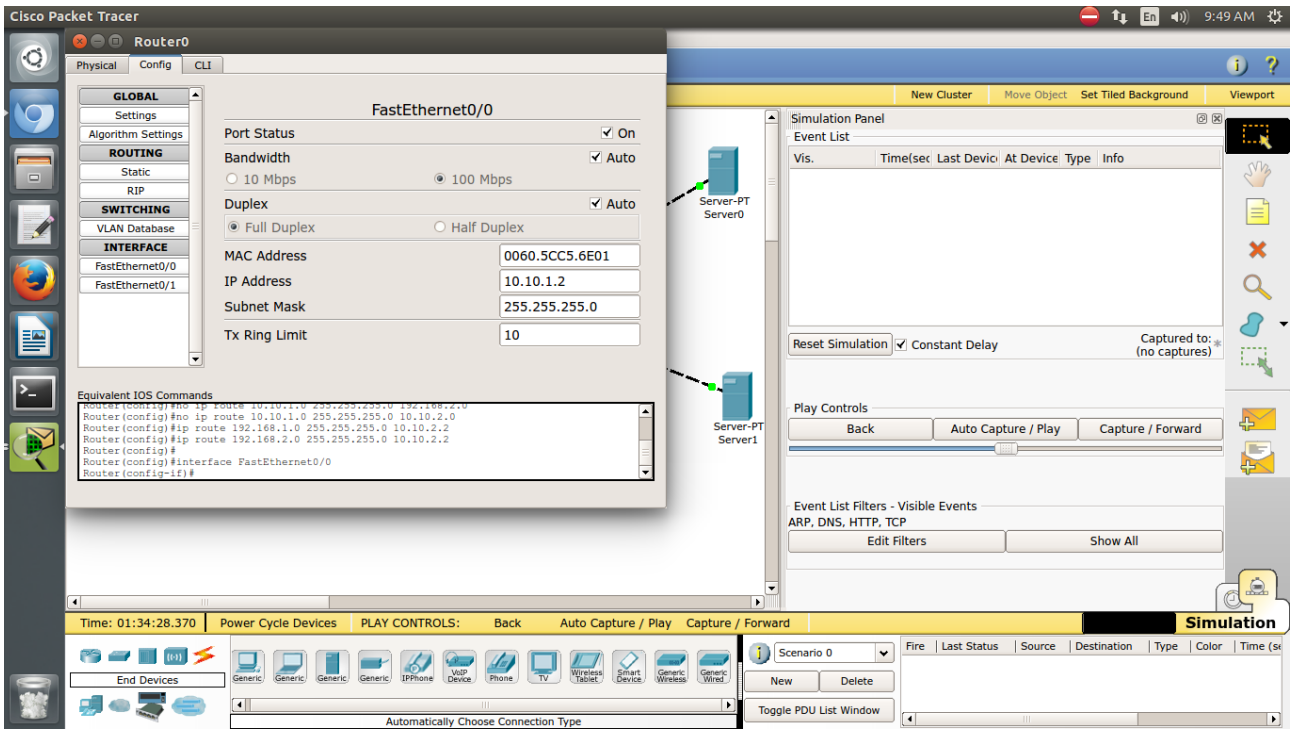
4. As the topology is created and all the devices are configured, we open the PC's Desktop on the cisco packet tracer
and type the name of the domain to be looked for as "google.com".

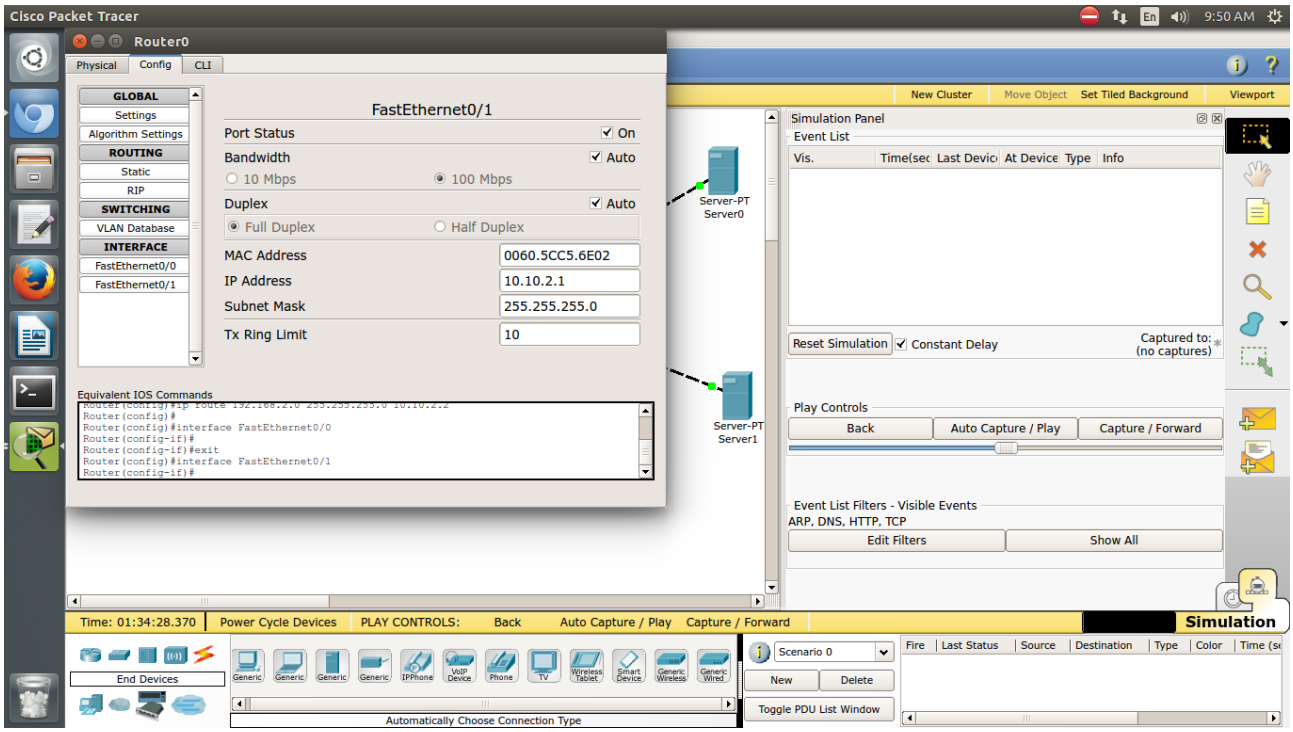
5. Now we open the packet tracer in th SIMULATION MODE and apply the filters on it fo capturing only the following
protocols :
a. Transmission Control Protocol
b. Address Resolution Protocol
c. Domain Name Service
d. Hyper Text Transfer Protocol

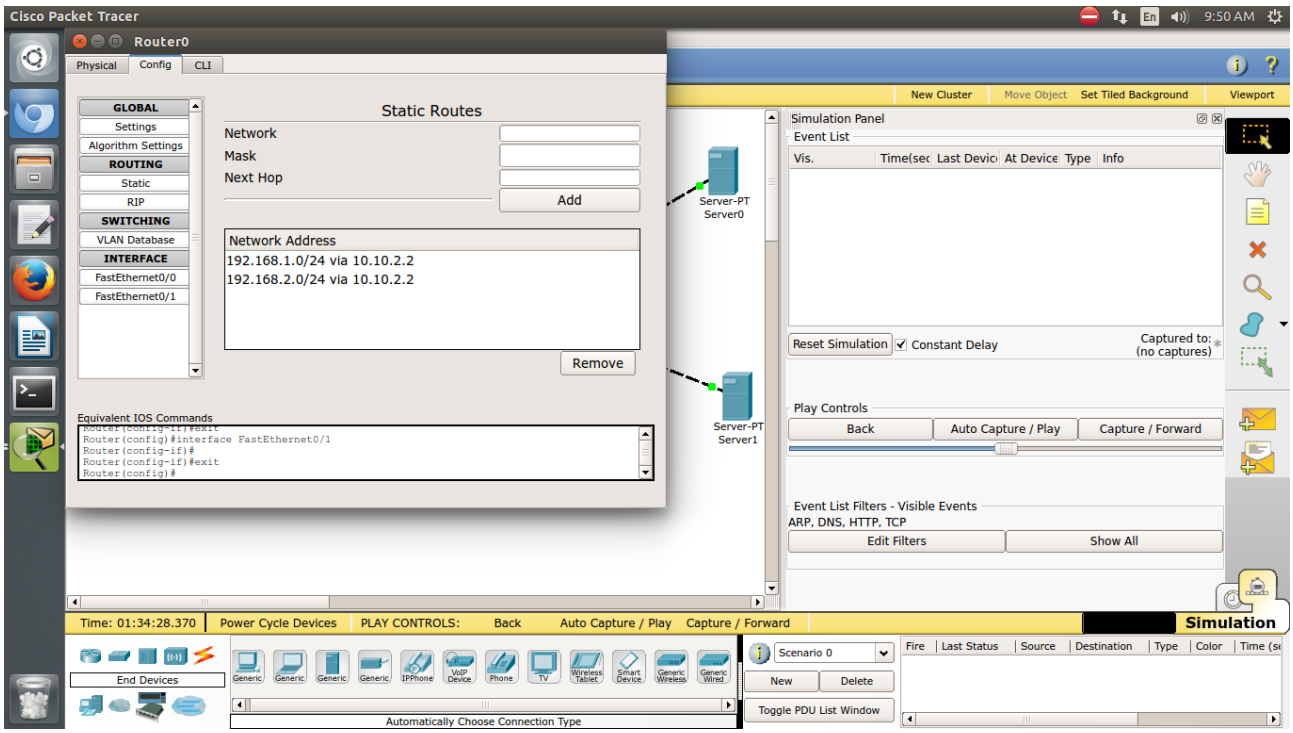
6. Now, on a proper configuration based topology, we achieve the web request from the web-server.

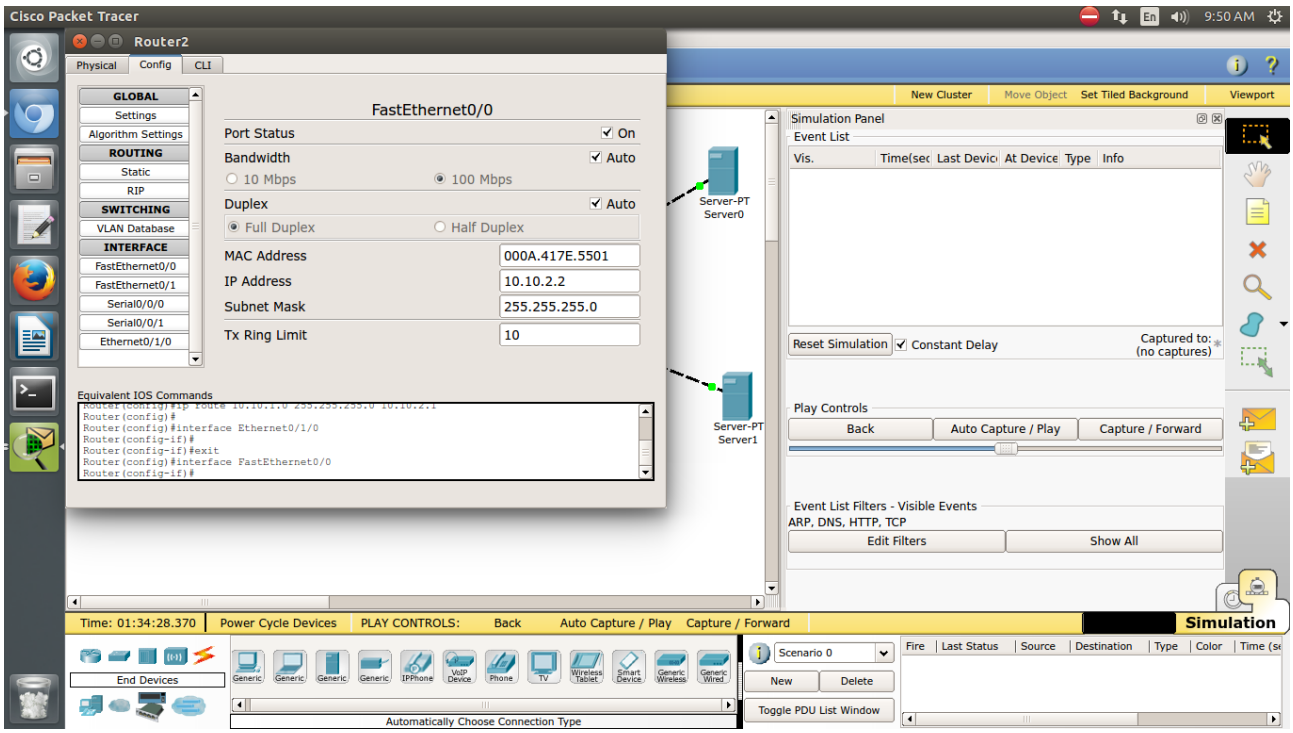


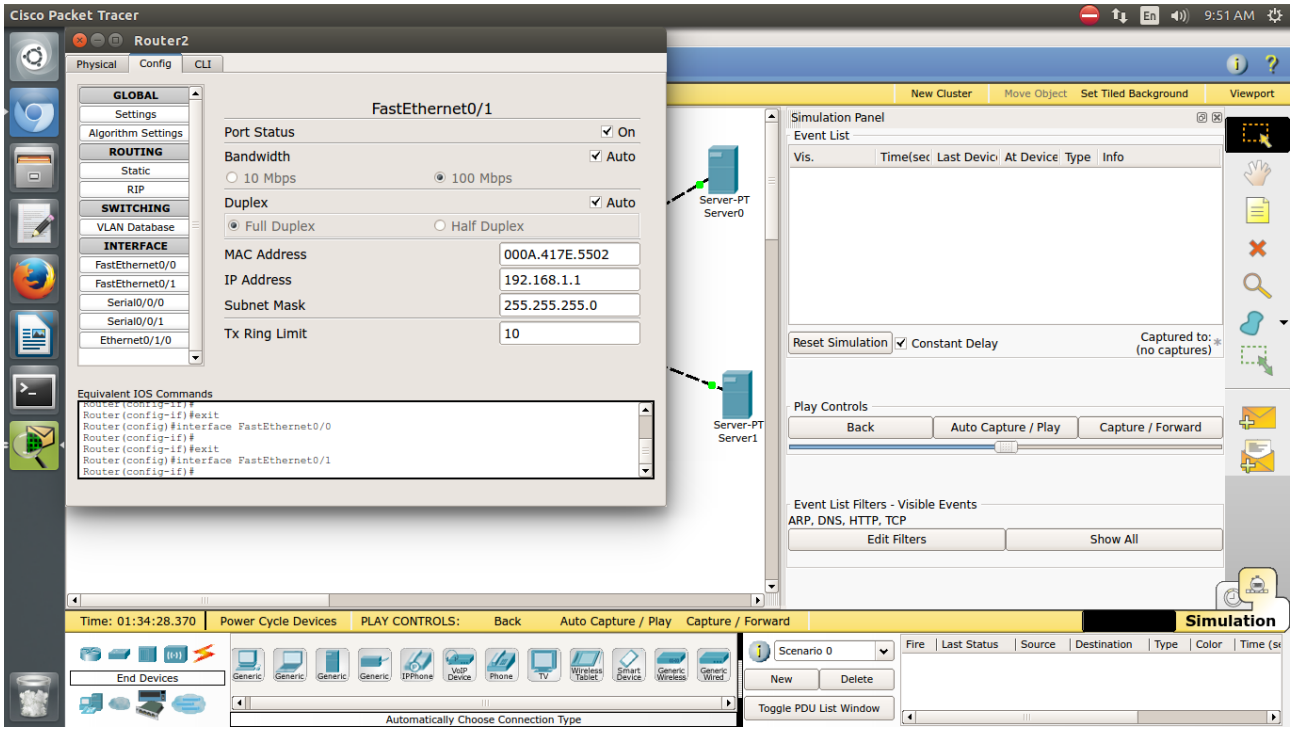


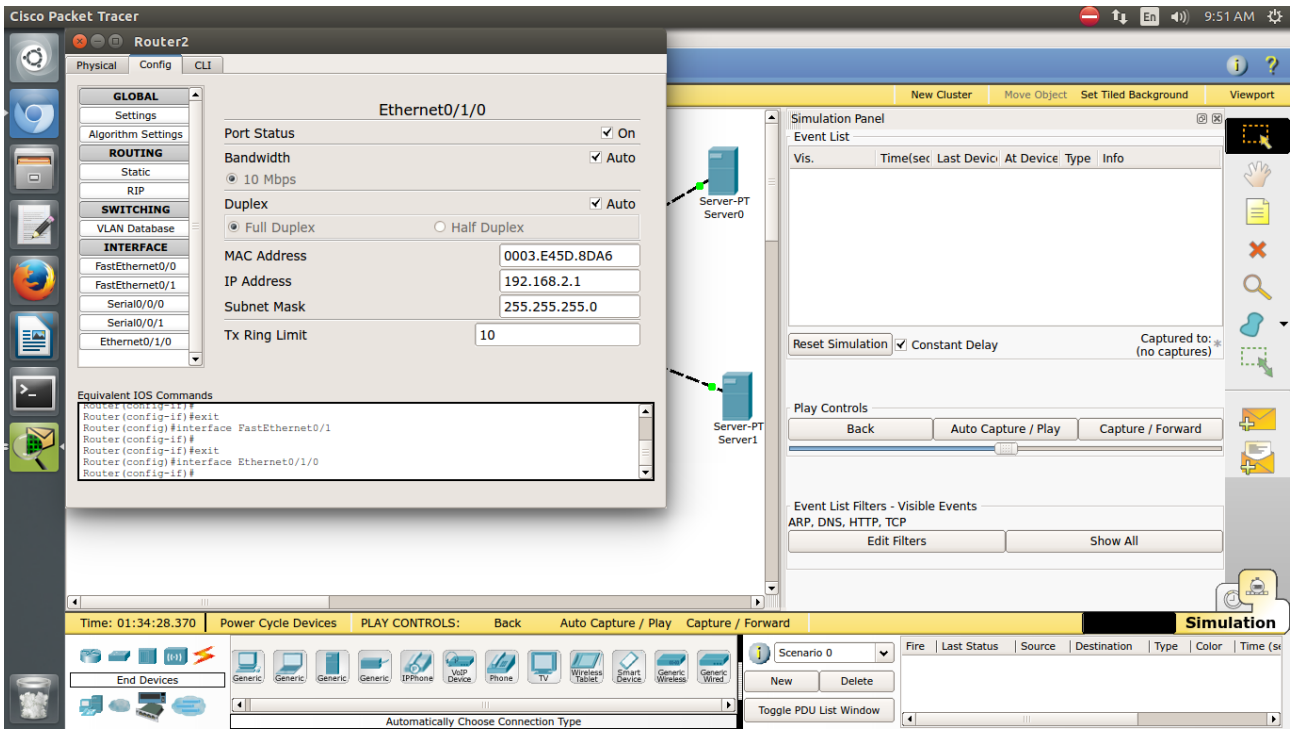

Cisco Packet Tracer
9:49 AM
Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
FastEthernet0/0
Port Status
On
Bandwidth
Auto
10 Mbps
100 Mbps
Duplex
Auto
Full Duplex
Half Duplex
MAC Address
0060.5CC5.6E01
IP Address
10.10.1.2
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#no ip route 10.10.1.0 255.255.255.0 10.10.2.0
Router(config)#ip route 192.168.1.0 255.255.255.0 10.10.2.2
Router(config)#ip route 192.168.2.0 255.255.255.0 10.10.2.2
Router(config)#
Router(config)#interface FastEthernet0/0
Router(config-if)#
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Server-PT
Server0
Server-PT
Server1
Reset Simulation
Constant Delay
Captured to:
(no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time

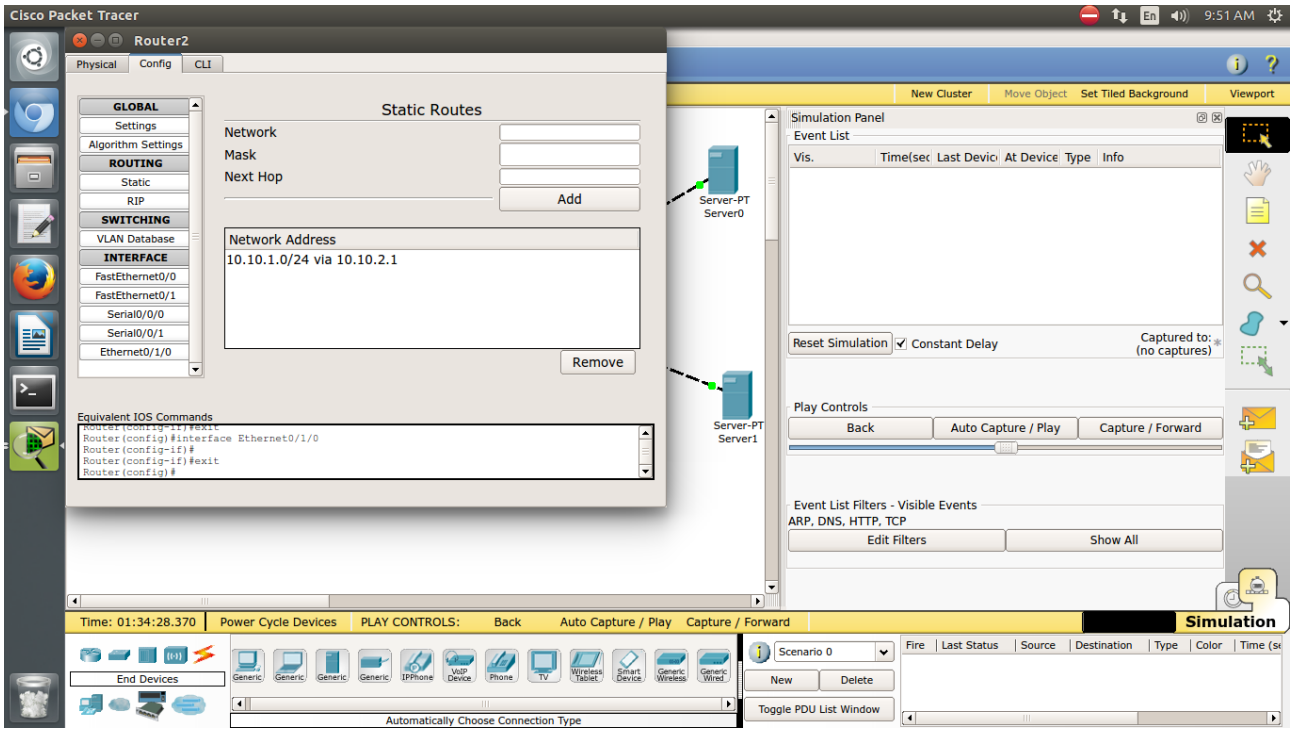

Cisco Packet Tracer
9:50 AM
Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
FastEthernet0/1
Port Status
On
Bandwidth
Auto
10 Mbps
100 Mbps
Duplex
Auto
Full Duplex
Half Duplex
MAC Address
0060.5CC5.6E02
IP Address
10.10.2.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Server-PT
Server0
Server-PT
Server1
Reset Simulation
Constant Delay
Captured to:
(no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time

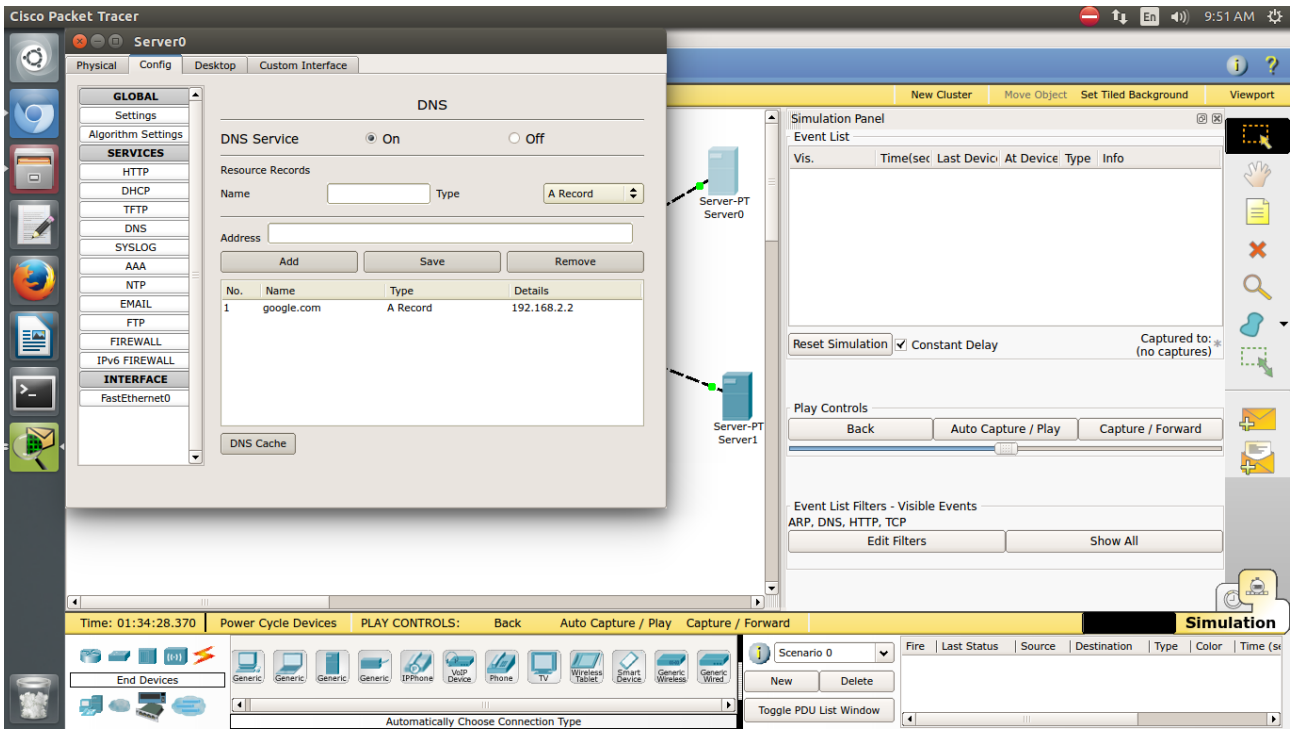

Cisco Packet Tracer
9:50 AM
Router0
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
Static Routes
Network
Mask
Next Hop
Add
Network Address
192.168.1.0/24 via 10.10.2.2
192.168.2.0/24 via 10.10.2.2
Remove
Equivalent IOS Commands
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#
Server-PT Server0
Server-PT Server1
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time (s

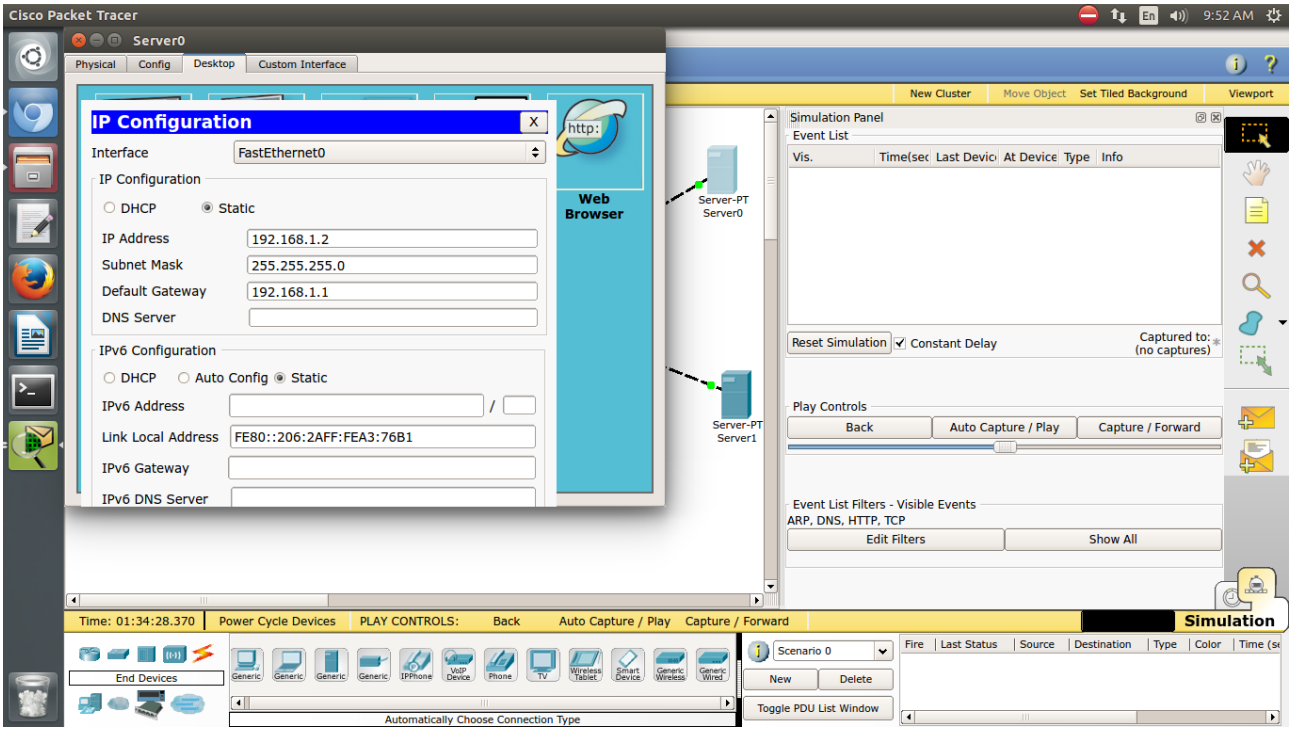

Cisco Packet Tracer
9:50 AM
Router2
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
Serial0/0/0
Serial0/0/1
Ethernet0/1/0
FastEthernet0/0
Port Status
On
Bandwidth
Auto
10 Mbps
100 Mbps
Duplex
Auto
Full Duplex
Half Duplex
MAC Address
000A.417E.5501
IP Address
10.10.2.2
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#
Router(config)#interface Ethernet0/1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Server-PT
Server0
Server-PT
Server1
Reset Simulation
Constant Delay
Captured to:
(no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time

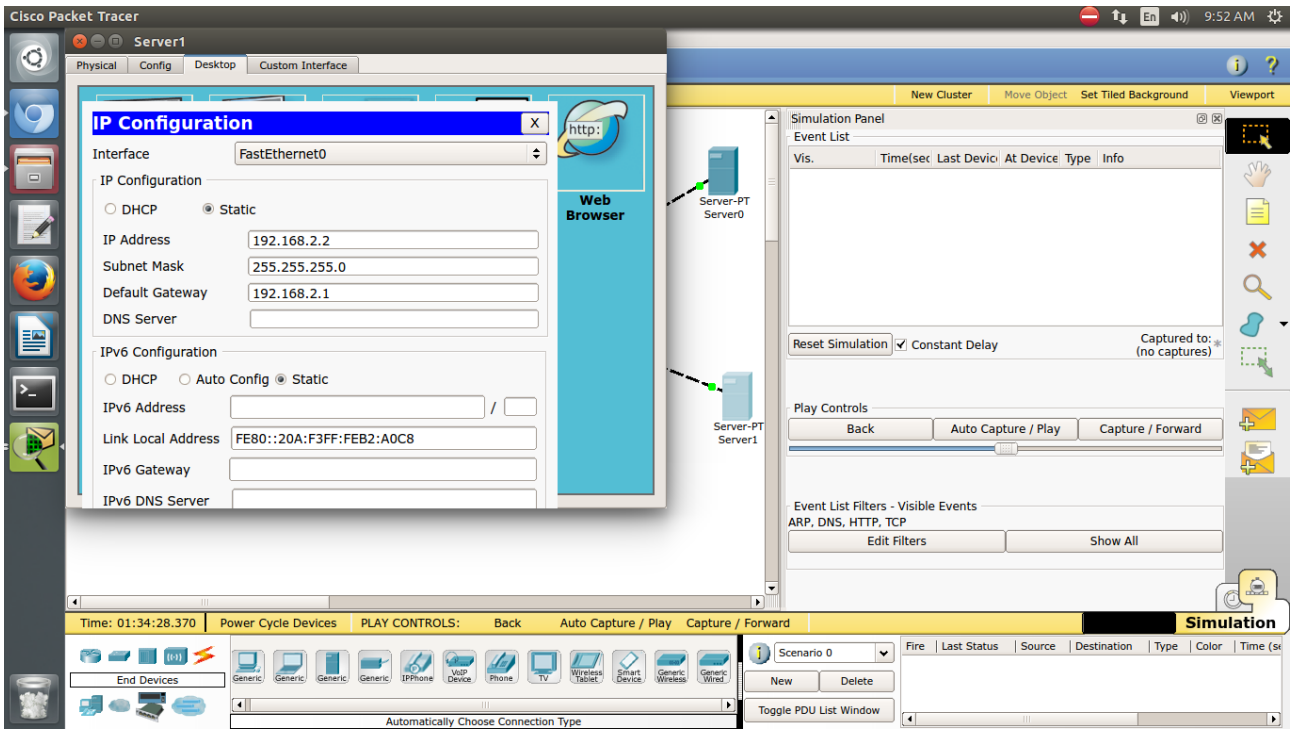

Cisco Packet Tracer
9:51 AM
Router2
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
Serial0/0/0
Serial0/0/1
Ethernet0/1/0
FastEthernet0/1
Port Status
On
Bandwidth
Auto
10 Mbps
100 Mbps
Duplex
Auto
Full Duplex
Half Duplex
MAC Address
000A.417E.5502
IP Address
192.168.1.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Server-PT
Server0
Server-PT
Server1
Reset Simulation
Constant Delay
Captured to:
(no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time

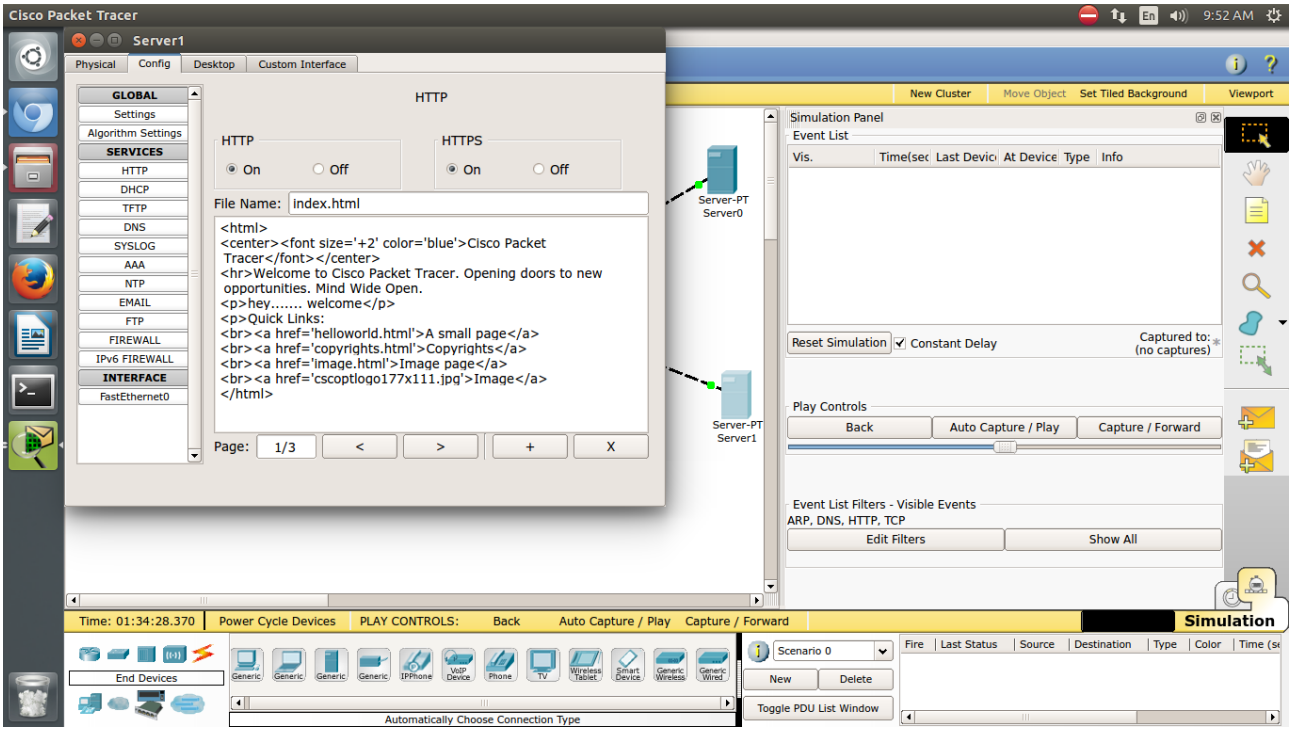

Cisco Packet Tracer
Router2
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
Serial0/0/0
Serial0/0/1
Ethernet0/1/0
Ethernet0/1/0
Port Status
On
Bandwidth
Auto
10 Mbps
Duplex
Auto
Full Duplex
Half Duplex
MAC Address
0003.E45D.8DA6
IP Address
192.168.2.1
Subnet Mask
255.255.255.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config-if)#exit
Router(config)#interface FastEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Ethernet0/1/0
Router(config-if)#
Server-PT Server0
Server-PT Server1
Simulation Panel
Event List
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window


Cisco Packet Tracer
Router2
Physical
Config
CLI
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
FastEthernet0/0
FastEthernet0/1
Serial0/0/0
Serial0/0/1
Ethernet0/1/0
Static Routes
Network
Mask
Next Hop
Add
Network Address
10.10.1.0/24 via 10.10.2.1
Remove
Equivalent IOS Commands
Router(config)#interface Ethernet0/1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#
Server-PT Server0
Server-PT Server1
Simulation Panel
Event List
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window


Cisco Packet Tracer
Server0
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
SERVICES
HTTP
DHCP
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
FIREWALL
IPv6 FIREWALL
INTERFACE
FastEthernet0
DNS
DNS Service
On
Off
Resource Records
Name
Type
A Record
Address
Add
Save
Remove
No.
Name
Type
Details
1
google.com
A Record
192.168.2.2
DNS Cache
Server-PT Server0
Server-PT Server1
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time


Cisco Packet Tracer
Server0
Physical
Config
Desktop
Custom Interface
IP Configuration
Interface
FastEthernet0
IP Configuration
DHCP
Static
IP Address
192.168.1.2
Subnet Mask
255.255.255.0
Default Gateway
192.168.1.1
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::206:2AFF:FEA3:76B1
IPv6 Gateway
IPv6 DNS Server
Web Browser
Server-PT Server0
Server-PT Server1
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time


Cisco Packet Tracer
Server1
Physical
Config
Desktop
Custom Interface
IP Configuration
Interface
FastEthernet0
IP Configuration
DHCP
Static
IP Address
192.168.2.2
Subnet Mask
255.255.255.0
Default Gateway
192.168.2.1
DNS Server
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::20A:F3FF:FEB2:A0C8
IPv6 Gateway
IPv6 DNS Server
Web Browser
Server-PT Server0
Server-PT Server1
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Vis.
Time(sec
Last Devic
At Device
Type
Info
Reset Simulation
Constant Delay
Captured to: (no captures)
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time

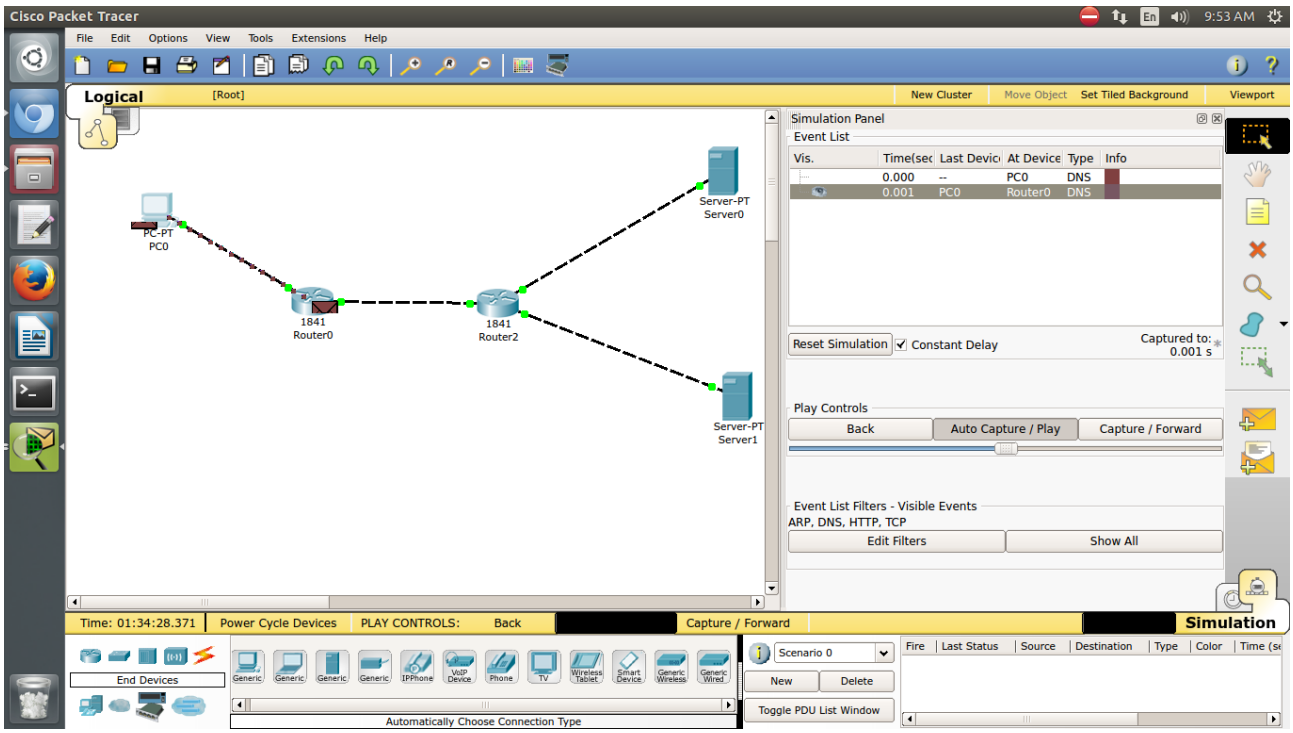

Cisco Packet Tracer
Server1
Physical
Config
Desktop
Custom Interface
GLOBAL
Settings
Algorithm Settings
SERVICES
HTTP
DHCP
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
FIREWALL
IPv6 FIREWALL
INTERFACE
FastEthernet0
HTTP
HTTP
On
Off
HTTPS
On
Off
File Name:
index.html
<html>
Cisco Packet Tracer
<hr>Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
hey..... welcome
Quick Links:

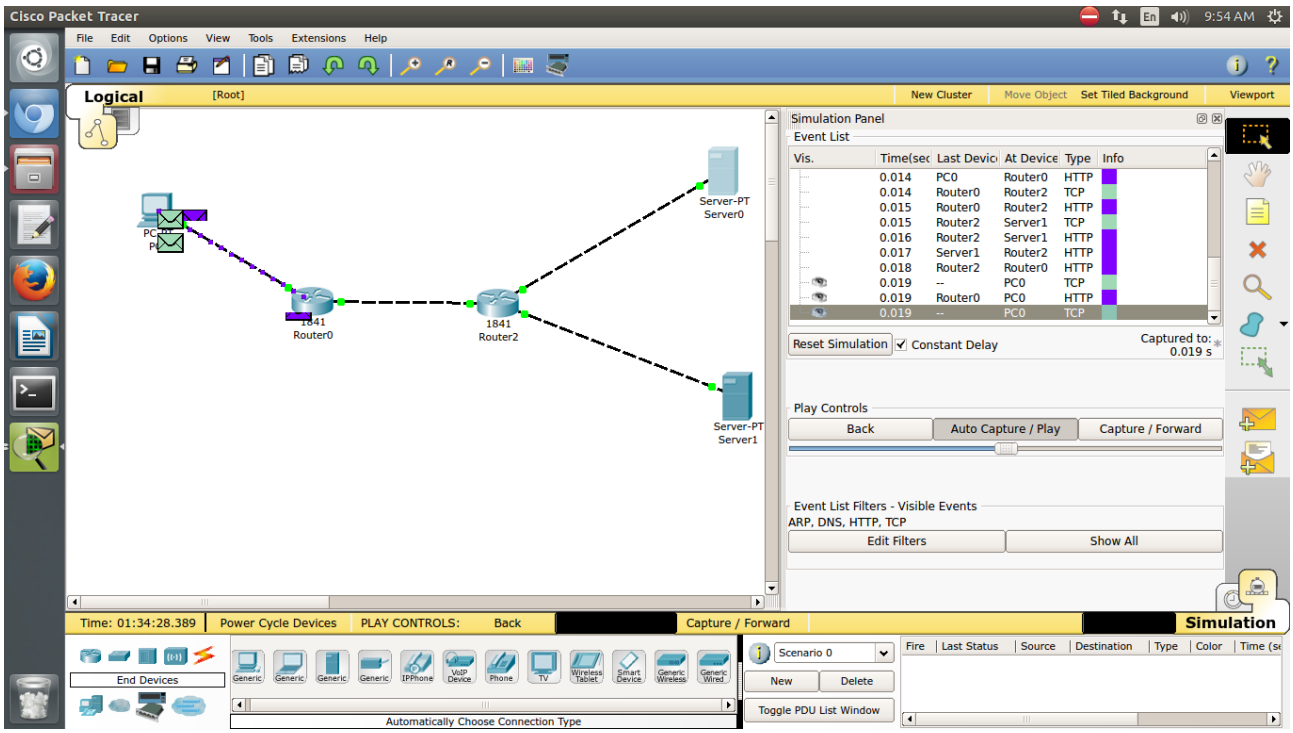
A small page

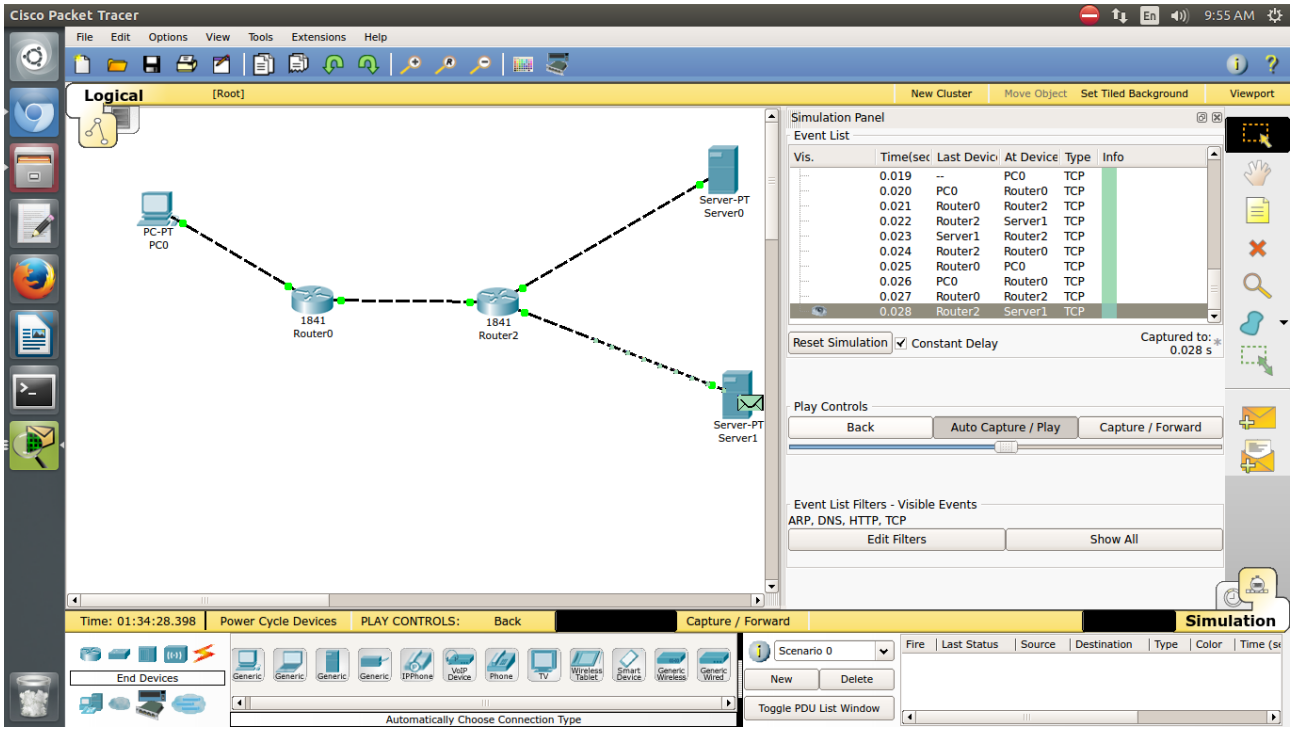
Copyrights

Image page

Image
</html>
Page:
1/3
Server-PT Server0
Server-PT Server1
New Cluster
Move Object
Set Tiled Background
Viewport
Simulation Panel
Event List
Reset Simulation
Constant Delay
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.370
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PC-PT PC0
1841 Router0
1841 Router2
Server-PT Server0
Server-PT Server1
Simulation Panel
Event List
Vis. Time(sec Last Devic At Device Type Info
0.000 -- PC0 DNS
0.001 PC0 Router0 DNS
Reset Simulation
Constant Delay
Captured to: 0.001 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.371
Power Cycle Devices
PLAY CONTROLS:
Back
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
1841 Router0
1841 Router2
Server-PT Server0
Server-PT Server1
Simulation Panel
Event List
Vis. Time(sec Last Devic At Device Type Info
0.014 PC0 Router0 HTTP
0.014 Router0 Router2 TCP
0.015 Router0 Router2 HTTP
0.015 Router2 Server1 TCP
0.016 Router2 Server1 HTTP
0.017 Server1 Router2 HTTP
0.018 Router2 Router0 HTTP
0.019 -- PC0 TCP
0.019 Router0 PC0 HTTP
0.019 -- PC0 TCP
Reset Simulation
Constant Delay
Captured to: 0.019 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters - Visible Events
ARP, DNS, HTTP, TCP
Edit Filters
Show All
Time: 01:34:28.389
Power Cycle Devices
PLAY CONTROLS:
Back
Capture / Forward
Simulation
End Devices
Automatically Choose Connection Type
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time



OBSERVATIONS :

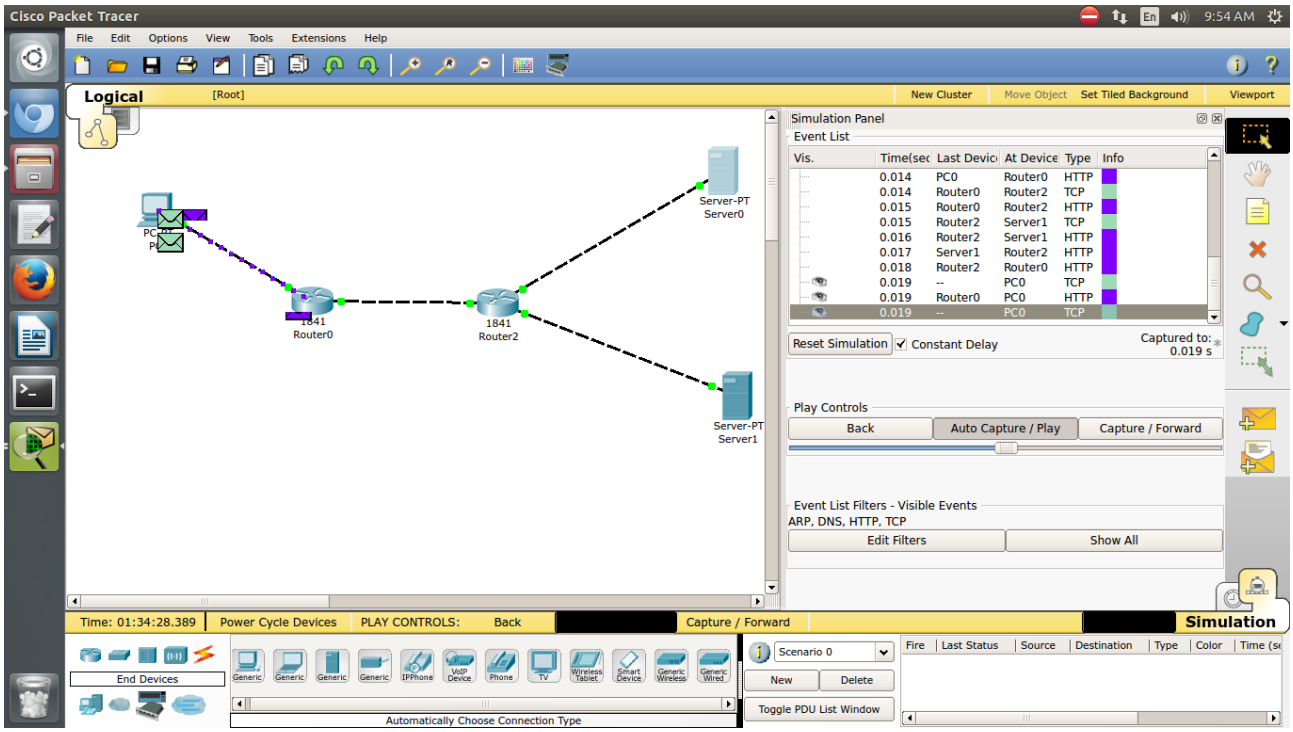
1. When the request for the domain "google.com" was made then because DNS serevr did not had the address in the cache the query took more
   time to resolve the page (access the page back to the client from the web server) than in the followed request.
   Time for 1st request : 0.328 seconds to capture (total time to fetch the page back.)
   Time for 2nd request : 0.019 seconds to capture (total time to fetch the page back.)

2. The reason for such a difference in time in the 2 requests being that DNS upon the first request of the web-server from the client
   cached the DNS-name and the IP address in its local DNS cache and on the subsequent request again doesn't need to search for the
   web-server again.

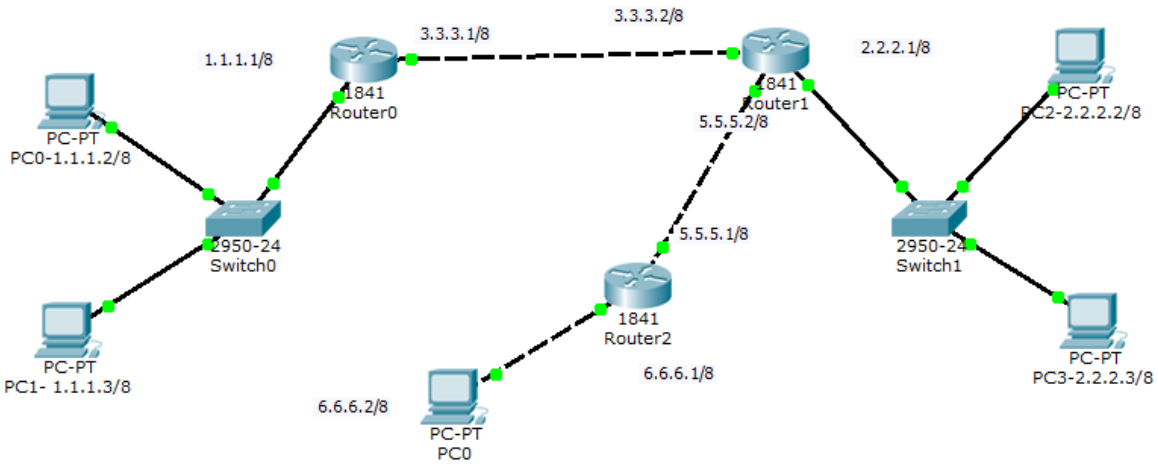
3. The ARP packets flowing were only seen in the first DNS request and not in the subsequent requset as because of the DNS cache.
   ALl other packets i.e. TCP, HTTP and DNS were seen in both the web-server requests.

4. The colour coding was observed in the simulation mode :
   ARP :
   HTTP : Purple
   DNS : Brown
   TCP : Green

5. The event list occuring is shown in the screenshots attached at the different places in the topology.The number of packets got
decreased by a fair amount as the ARP packets were not in the picture.



EXCERCISE 2 : Connect DNS Server and Web Sever for the blow toplogy and transmit the packets



Computer Systems Laboratory – UE18CS304

PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D – 27th October 2020

Week #7 – Life of a Packet in the Internet using Cisco Packet Tracer

Objectives:

To understand the life of a packet in the internet using Cisco Packet Tracer.

Host Configuration Details:

Host	IPv4 Address	Gateway	DNS Server
HostA	10.10.1.1	10.10.1.2	192.168.1.2

Router Configuration Details:

Router	FastEthernet0/0 Incoming Interface IP	FastEthernet0/1 Outgoing Interface IP	Ethernet0/1/0 Outgoing Interface IP
Router1	10.10.1.2	10.10.2.1	-
Router2	10.10.2.2	198.168.1.1	198.168.2.1

Server Configuration Details:

Server	IPv4 Address	Gateway
DNS Server	192.168.1.2	192.168.1.1
WEB Server	192.168.2.2	192.168.2.1

Routing Table Entries:

Router	Network	Next Hop
Router1	192.168.1.0	10.10.2.2
Router1	192.168.2.0	10.10.2.2
Router2	10.10.1.0	10.10.2.1

DNS Server Configuration:

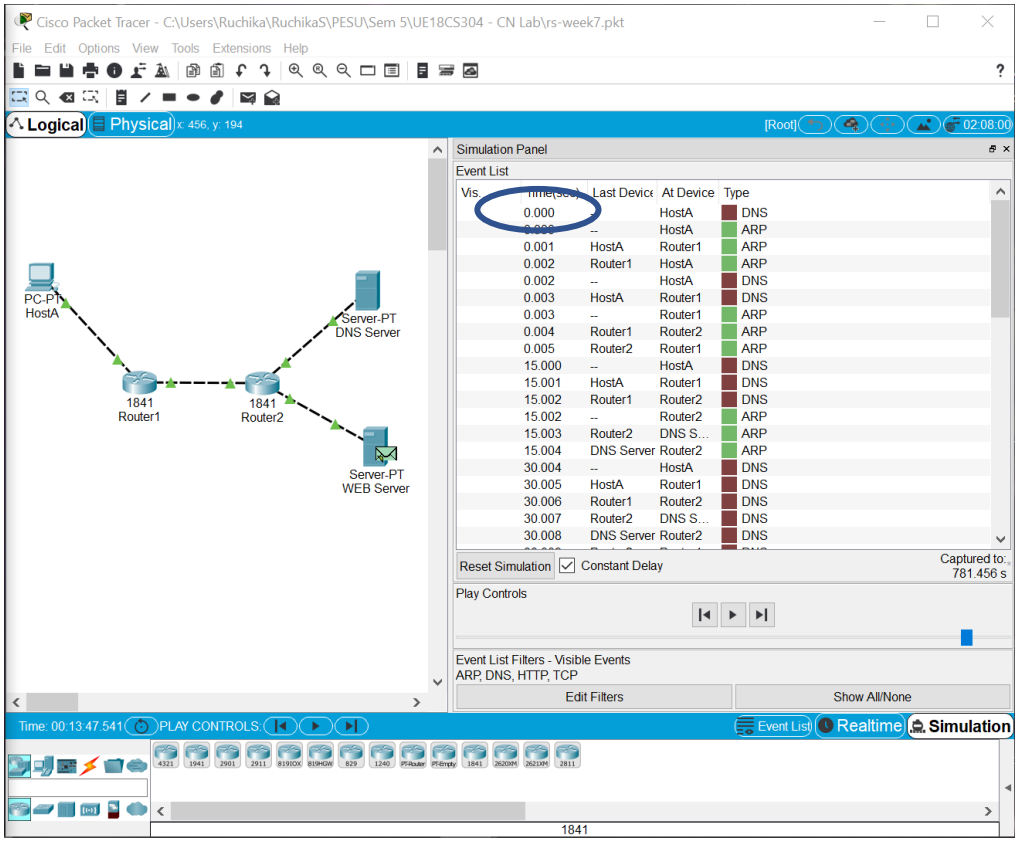
Record Type: Type-A Record

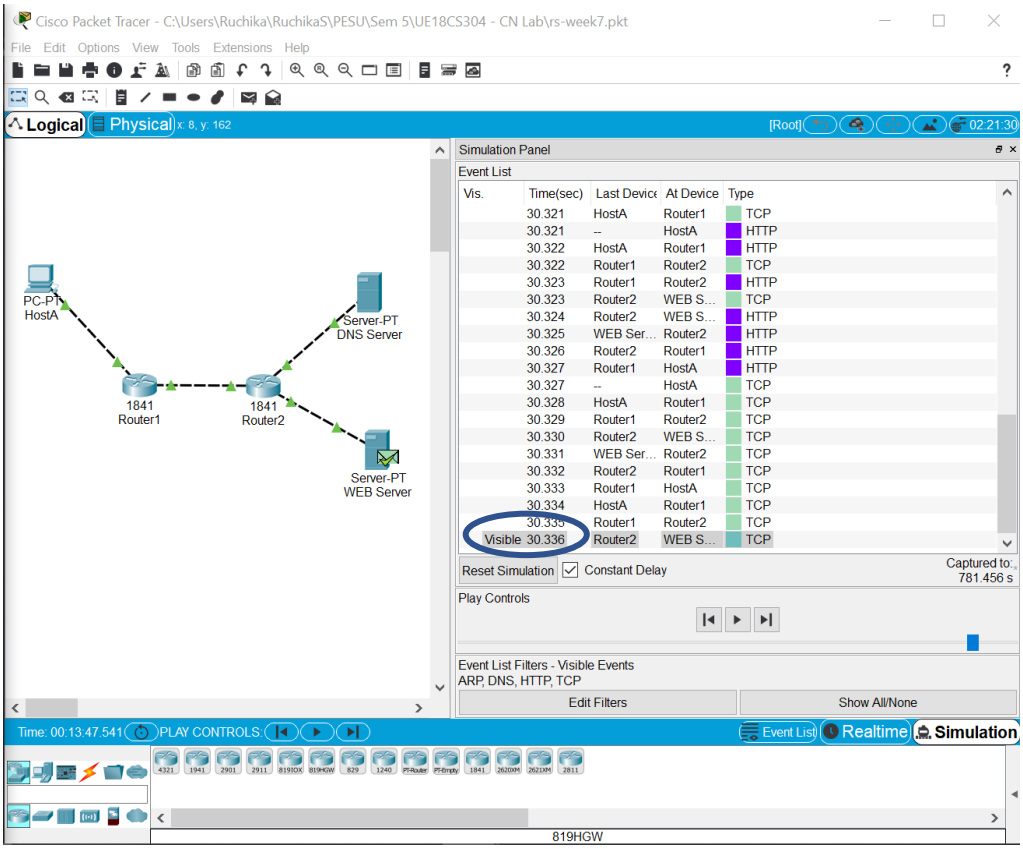
Name: google.com (Name of the domain)

Address: 192.168.2.2 (IP Address of the WEB server)

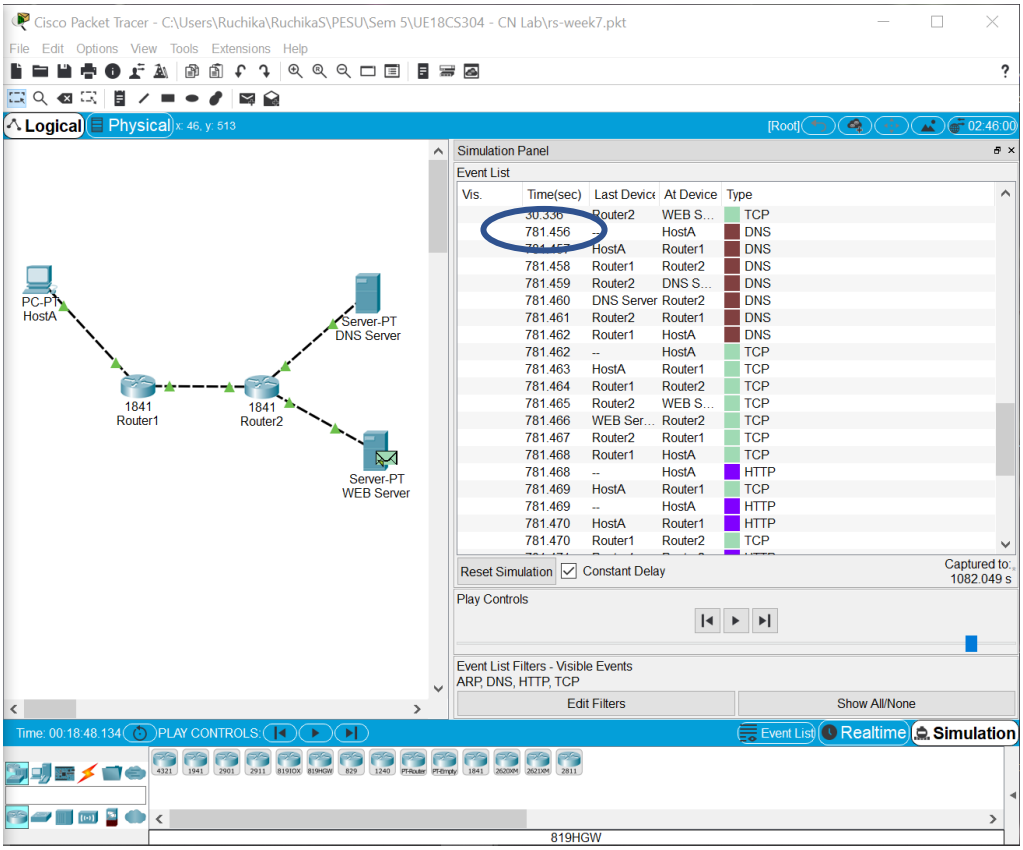
Filters Applied in the Simulation: TCP, ARP, DNS, HTTP

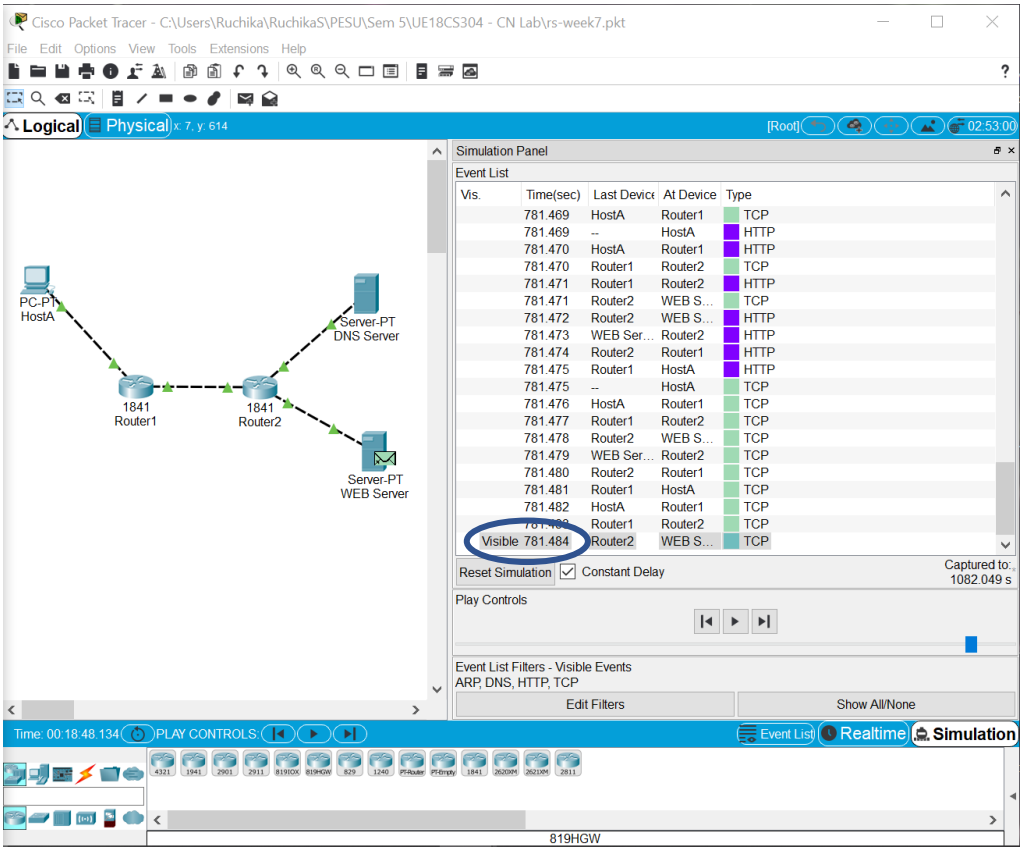
First Request - google.com



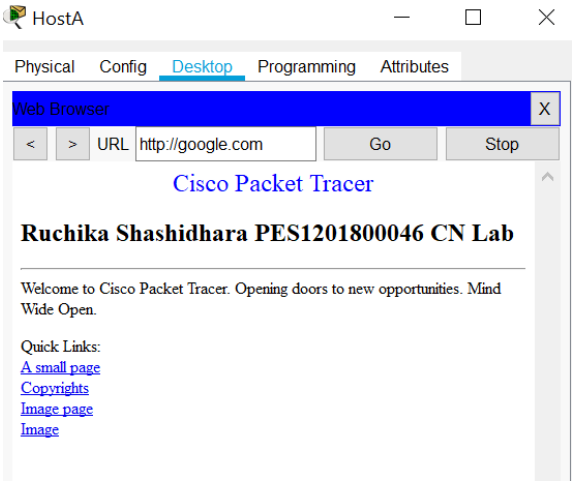


Second Request - google.com





Response - google.com



Observations:

- When the request for the domain **google.com** was made for the first time, since there was no DNS cache, the query took more time to resolve the page than in the followed request (while accessing the page back to the client from the web server). Upon the second request, the DNS Name & IP Address was retrieved from its local cache and the total time taken reduced. The number of packets also decreased by a fair amount in the second request, as ARP packets were not present.

  Time for 1st request: 30.336 – 0.000 = **30.366 seconds** to capture (total time to fetch the page back)

  Time for 2nd request: 781.484 – 781.456 = **0.028 seconds** to capture (total time to fetch the page back)

- The ARP packets flowing were only seen in the first DNS request and not in the subsequent request as the information was retrieved from the DNS cache itself. All other packets - TCP, HTTP and DNS were seen in both the web server requests.

  The colour coding was observed in the simulation mode:

  ARP: Dark Green

  HTTP: Purple

  DNS: Brown

  TCP: Green

Conclusion:

- Understood how to build and configure different topologies along with routing table entries.
- Understood how data simulations & interactions travel through a network.
- Understood how ARP, HTTP, DNS, TCP packets travel in a network with Web server, DNS server and routers similar to a network in the Internet.

Week 8

Understand the building blocks and usage of ClayNet Network Virtualization platform with reference to OSI Layer.

Objectives of the Lab:

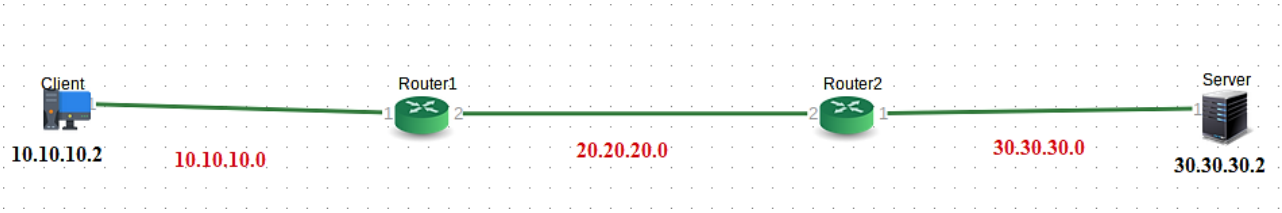
- Understand the building blocks of ClayNet.
- Build a simple client-server network using routers, switches, and network hosts.
- To learn the static IP routing behavior such as default and static routes and routing tables.
- Use common network utilities to verify LAN operation and analyze data traffic.

Prerequisites:

This lab assumes some understanding of the building blocks of communication networks and basic client-server architecture.

Topology 1:

Create a topology in ClayNet, as shown in following figure.



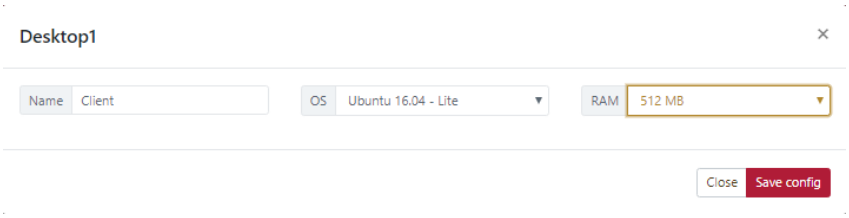
Important Instructions:

To access ClayNet, type **http://1.6.181.7:9000/** in browser**.** Login credentials will be provided by the faculty incharge.

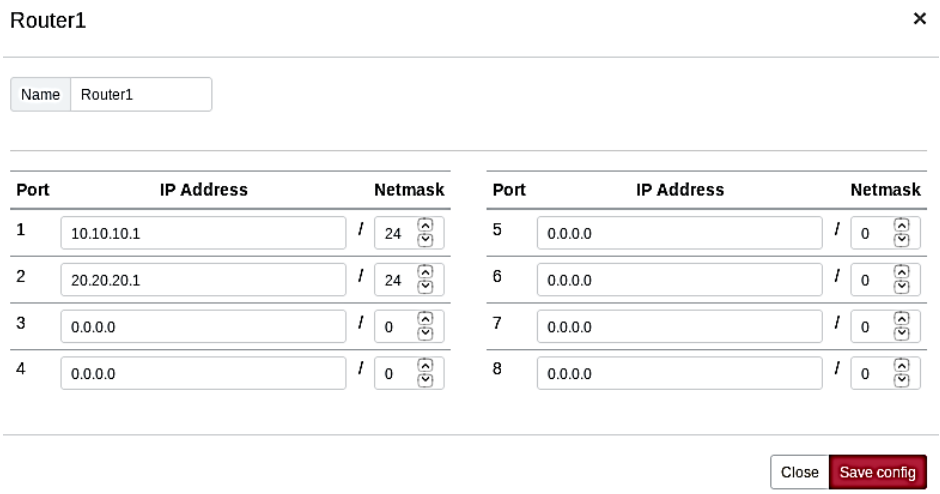
Execution Tasks:

Task 1: Understand the network and compute components available in ClayNet.

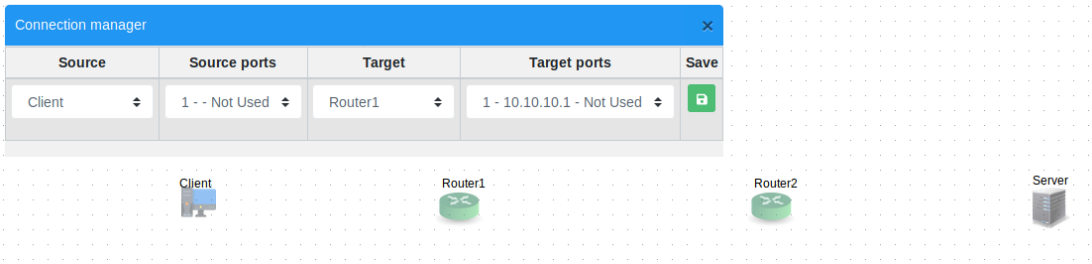
Task 2: Drag and drop the necessary components to create the given topology. Provide the names for compute, select OS (Ubuntu 16.04 – Lite or Ubuntu 16.04 – CLI) and RAM (512 MB) as shown below.



Task 3: Drag and drop the Routers and set the IP addresses for all the necessary router ports. (You can also set them later by right clicking on the router icon and selecting 'Device Configuration'.)



Task 4: Go to connection manager and select appropriate Source, Source ports, Target and Target ports and save the connection.

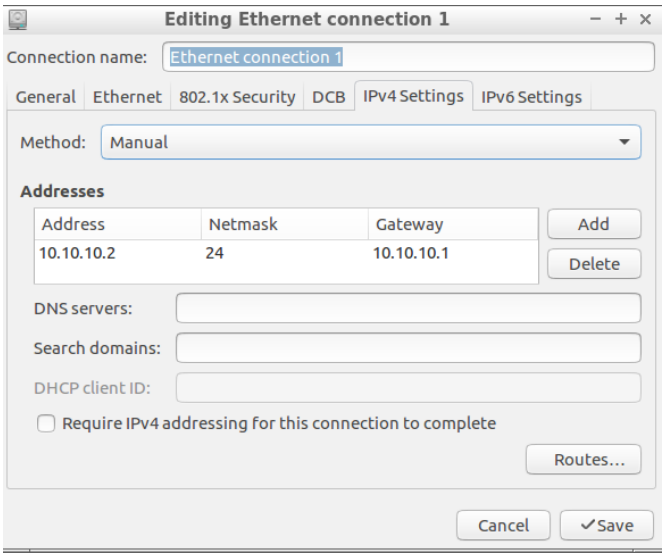


Task 5: To deploy the topology, save the topology first and deploy it by clicking 'Deploy' button available on the top. (Note: It will take few seconds or even minutes to deploy the topology for the first time).

Task 5: Go to 'Remote Desktop' by right clicking on client and server icons and set the IP addresses accordingly. Also add the gateway address. (Login: user - test, password - test)

Client:

IP Address ---> 10.10.10.2 Gateway ---> 10.10.10.1




```
test@Lubuntu-vm:~$ ping 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=0.862 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.363 ms
^C
--- 10.10.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.363/0.612/0.862/0.250 ms
test@Lubuntu-vm:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.014 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.011 ms
^C
--- 10.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.011/0.021/0.039/0.013 ms
test@Lubuntu-vm:~$
```

Server:

IP Address ---> 30.30.30.2 Gateway ---> 30.30.30.1

Task 6: From client, ping to server 30.30.30.2. Ping will not be successful and Router1 will reply with 'Destination host unreachable'.

```
test@Lubuntu-vm:~$ ping 30.30.30.2
PING 30.30.30.2 (30.30.30.2) 56(84) bytes of data.
From 10.10.10.1 icmp_seq=1 Destination Host Unreachable
From 10.10.10.1 icmp_seq=2 Destination Host Unreachable
From 10.10.10.1 icmp_seq=3 Destination Host Unreachable
^C
--- 30.30.30.2 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2049ms

test@Lubuntu-vm:~$
```

Task 7: Set up the following routing table entries for Routers 1 & 2.

Routers	Destination	Next hop gateway	Via
Router 1	30.30.30.0	20.20.20.2	Direct
Router 2	10.10.10.0	20.20.20.1	Direct

Steps to add the routing table entries:

Step 1: Login to Router1 by right clicking on Router icon and selecting ‘Console Access’. (Type ‘Enter’ key once to get into Login screen. Username - test, Password- test@12345)

Step 2: Display the routing table to view all static routes using the command.

```
show route summary -s active data
```

```
clayroot@ClayNet:~$ telnet 127.0.0.1  56075
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.



Login: Login: test
Password:

operational> show route summary -s active data

> IPv4 active routes

>> Destination : 10.10.10.0/24
   Gateway(s)  : { if-port-1
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 20.20.20.0/24
   Gateway(s)  : { if-port-2
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 127.0.0.0/8
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : R

>> Destination : 127.0.0.1/32
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : -

 Total number of IPv4 active routes displayed : 4
 No IPv6 active routes are available
 No MPLS active routes are available

operational>
```

Note in routing table of Router1 that there is no route to reach the destination network 30.30.30.0/24. Go to configure mode and start configuring the router for all the possible routes.

Step 3: Configure a static route in Router1 for destination 30.30.30.0/24 with next-hop gateway as 20.20.20.2, which is the IP address of Router2.

```
operational> configure
Entering configuration mode with exclusive access.
configure> create parameter-group ip-route to-n30
Info: Parameter group instance created.
configure> set enable yes
configure> set router data
configure> set destination 30.30.30.0/24
configure> set next-hop gateway 20.20.20.2
configure> save
Info: Parameter group ip-route "to-n30" saved
configure> exit
operational>
```

Step 4: Check routing table again and verify that the route is added.

```
operational> show route summary -s active data

> IPv4 active routes

>> Destination : 10.10.10.0/24
   Gateway(s)  : { if-port-1
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 20.20.20.0/24
   Gateway(s)  : { if-port-2
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 30.30.30.0/24
   Gateway(s)  : { if-port-2
                   20.20.20.2 }
   Source      : static
   Flags       : -

>> Destination : 127.0.0.0/8
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : R

>> Destination : 127.0.0.1/32
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : -

 Total number of IPv4 active routes displayed : 5
 No IPv6 active routes are available
 No MPLS active routes are available
```

Step 5: Repeat the steps 3 & 4 to configure a static route in Router2 for destination 10.10.10.0/24 with next-hop gateway as 20.20.20.1, which is the IP address of Router1.

```
operational> configure
Entering configuration mode with exclusive access.
configure> create parameter-group ip-route to-n10
Info: Parameter group instance created.
configure> set enable yes
configure> set router data
configure> set destination 10.10.10.0/24
configure> set next-hop gateway 20.20.20.1
configure> save
Info: Parameter group ip-route "to-n10" saved
configure> exit
operational>
```

```
operational> show route summary -s active data

> IPv4 active routes

>> Destination : 10.10.10.0/24
   Gateway(s)  : { if-port-2
                   20.20.20.1 }
   Source      : static
   Flags       : -

>> Destination : 20.20.20.0/24
   Gateway(s)  : { if-port-2
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 30.30.30.0/24
   Gateway(s)  : { if-port-1
                   0.0.0.0 }
   Source      : direct
   Flags       : -

>> Destination : 127.0.0.0/8
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : R

>> Destination : 127.0.0.1/32
   Gateway(s)  : { ^loopback-1
                   127.0.0.1 }
   Source      : direct
   Flags       : -

 Total number of IPv4 active routes displayed : 5
 No IPv6 active routes are available
 No MPLS active routes are available
```

Task 8: Now Ping will be successful as all the required routers are now configured. Observe the TTL getting decremented by 2 because two hops/routers are in between. Also keep the Wireshark ready for observation.

```
test@Lubuntu-vm:~$ ping 30.30.30.2
PING 30.30.30.2 (30.30.30.2) 56(84) bytes of data.
64 bytes from 30.30.30.2: icmp_seq=1 ttl=62 time=0.906 ms
64 bytes from 30.30.30.2: icmp_seq=2 ttl=62 time=0.781 ms
64 bytes from 30.30.30.2: icmp_seq=3 ttl=62 time=0.865 ms
64 bytes from 30.30.30.2: icmp_seq=4 ttl=62 time=1.08 ms
^C
--- 30.30.30.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.781/0.908/1.083/0.116 ms
test@Lubuntu-vm:~$
```

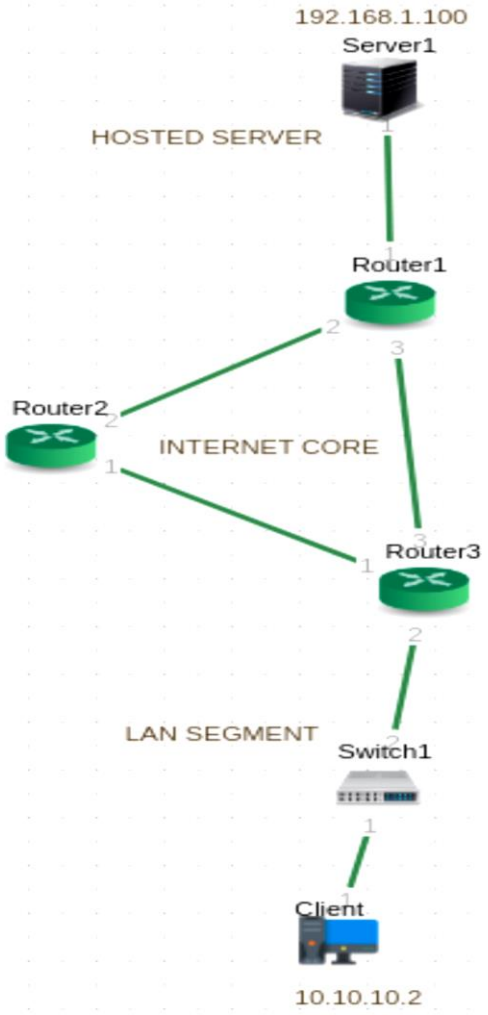
1	0.000000000	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=1/256, ttl=6
2	0.000951130	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=1/256, ttl=6
3	1.001056986	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=2/512, ttl=6
4	1.001749765	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=2/512, ttl=6
5	2.015331316	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=3/768, ttl=6
6	2.016067278	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=3/768, ttl=6
7	3.039417776	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=4/1024, ttl=
8	3.040217193	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=4/1024, ttl=
9	4.063367384	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=5/1280, ttl=
10	4.064747719	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=5/1280, ttl=
11	5.064965592	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=6/1536, ttl=
12	5.065842624	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=6/1536, ttl=
13	6.080629150	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=7/1792, ttl=
14	6.081360600	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=7/1792, ttl=
15	7.103340029	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=8/2048, ttl=
16	7.104075101	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=8/2048, ttl=
17	8.127362145	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=9/2304, ttl=
18	8.128168354	30.30.30.2	10.10.10.2	ICMP	100 Echo (ping) reply id=0x06e5, seq=9/2304, ttl=
19	9.151363544	10.10.10.2	30.30.30.2	ICMP	100 Echo (ping) request id=0x06e5, seq=10/2560, ttl=

Task 9: Also observe the output of **tracepath -n 30.30.30.2** command on Client.

Record these observations in your notebook. Upload the following screenshots in Edmodo.

1) Pinging
2) Wireshark capture

Topology 2: (Mandatory)



- Create and deploy the topology as show above.
- Appropriately configure static routing entries in all three routers, so that server is reachable from client desktop.

Observations Required:

- How many hops will client take to reach the server?
- Observe the RTT and justify your observation.
- While pinging, cut the link between Router1 and Router3. What will happen now?

Record these observations in your notebook. Upload the following screenshots in Edmodo.

1) Pinging
2) Wireshark capture

Computer Systems Laboratory – UE18CS304

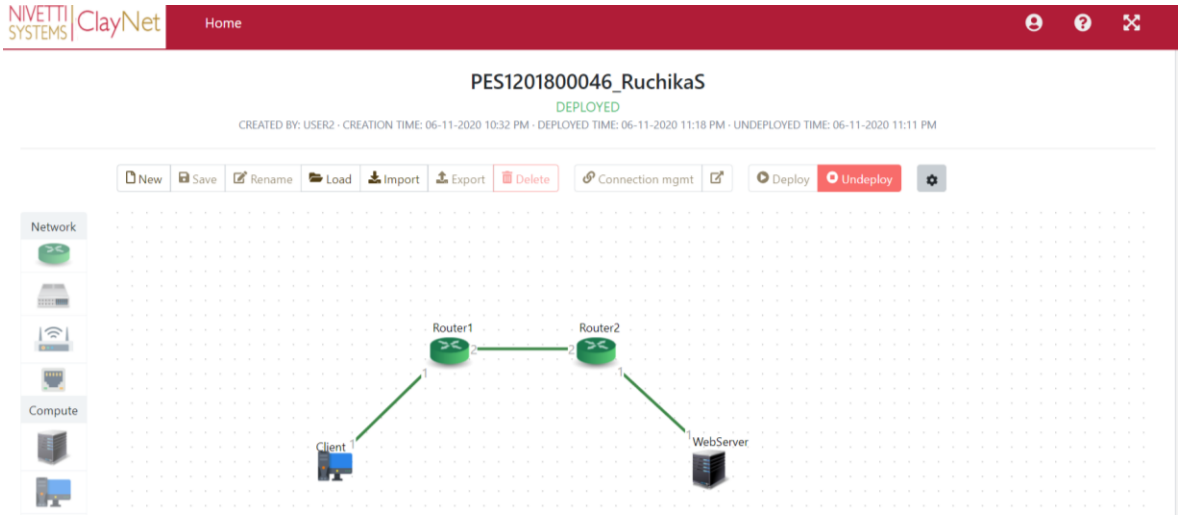
PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D

Week #8 – IPv4 Static Addressing and Routing in ClayNet – 7th November 2020

Objectives:

- To understand the building blocks of ClayNet
- To build a simple client-server network using routers, switches, and network hosts
- To learn the static IP routing behaviour such as default and static routes and routing tables

Topology



Host Configuration Details:

Host	IPv4 Address	Gateway
Client	10.10.10.2/24	10.10.10.1/24

Router Configuration Details:

Router	Port #1 IPv4 Address	Port #2 IPv4 Address
Router1	10.10.10.1/24	20.20.20.1/24
Router2	30.30.30.1/24	20.20.20.2/24

Server Configuration Details:

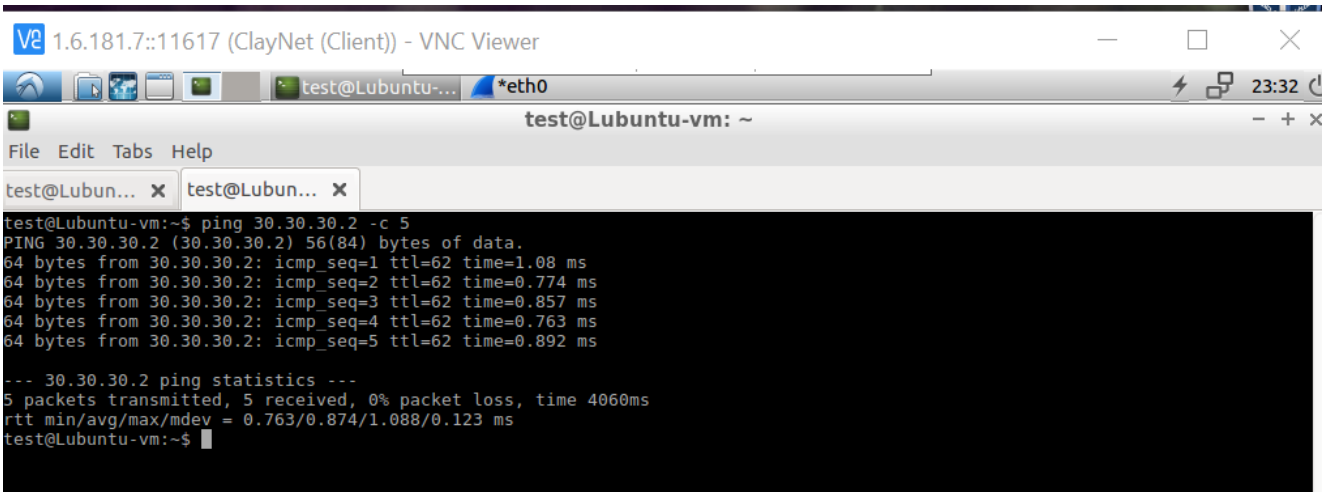
Server	IPv4 Address	Gateway
WebServer	30.30.30.2/24	30.30.30.1/24

Routing Table Entries Added:

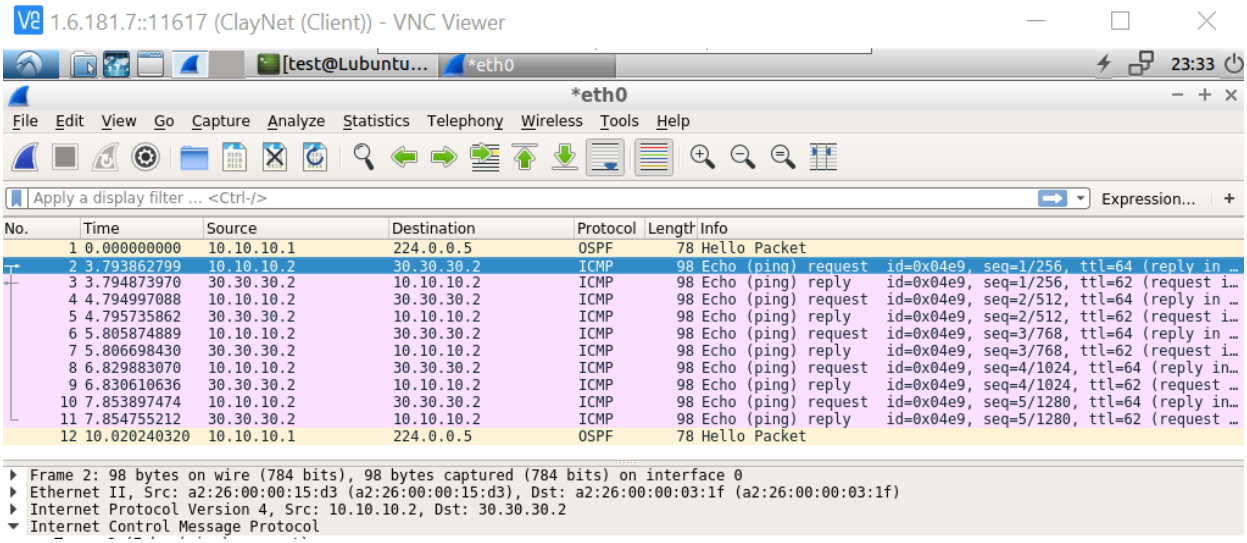
Router	Destination	Next Hop Gateway	Via
Router1	30.30.30.0/24	10.10.2.2/24	Direct
Router2	10.10.1.0/24	10.10.2.1/24	Direct

Observations on Client Machine:

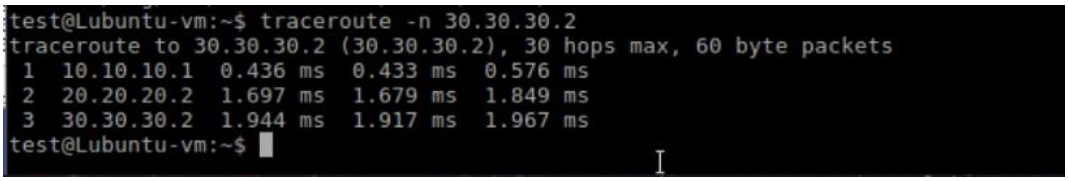
$ ping 30.30.30.2 -c 5



Wireshark Capture



$ traceroute -n 30.30.30.2



Conclusion:

- Built a simple Client-Server Network using Routers, Switches, & Network Hosts
- Learnt about static IPv4 routing behaviour with default & static routes & routing tables.
- With **tracepath -n 30.30.30.2** command on Client, we can see that it takes a total of 3 hops (Router1(10.10.10.1), Router2(20.20.20.1), WebServer(30.30.30.2)) to reach the Server.
- Round-trip-time (RTT) keeps getting decremented as 2 routers leads to 3 hops in between.

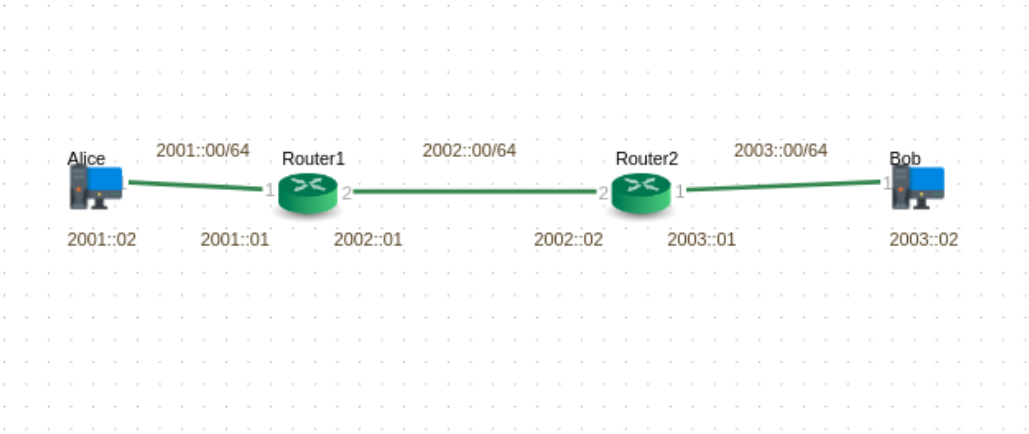
Week 9

IPv6 Configuration and Static Routing

Learning Objectives:

- Perform basic IPv6 configurations on a Desktop and Router.
- Distinguish between IPv4 and IPv6 addresses
- Configure IPv6 static routes in Router
- Observe traffic flow using IPv6 static routes.
- IPv6 neighbor cache entries
- Understanding IPv6 Link Local Address
- Working with ping6 and tracepath6

LAB Network Topology:



Steps :

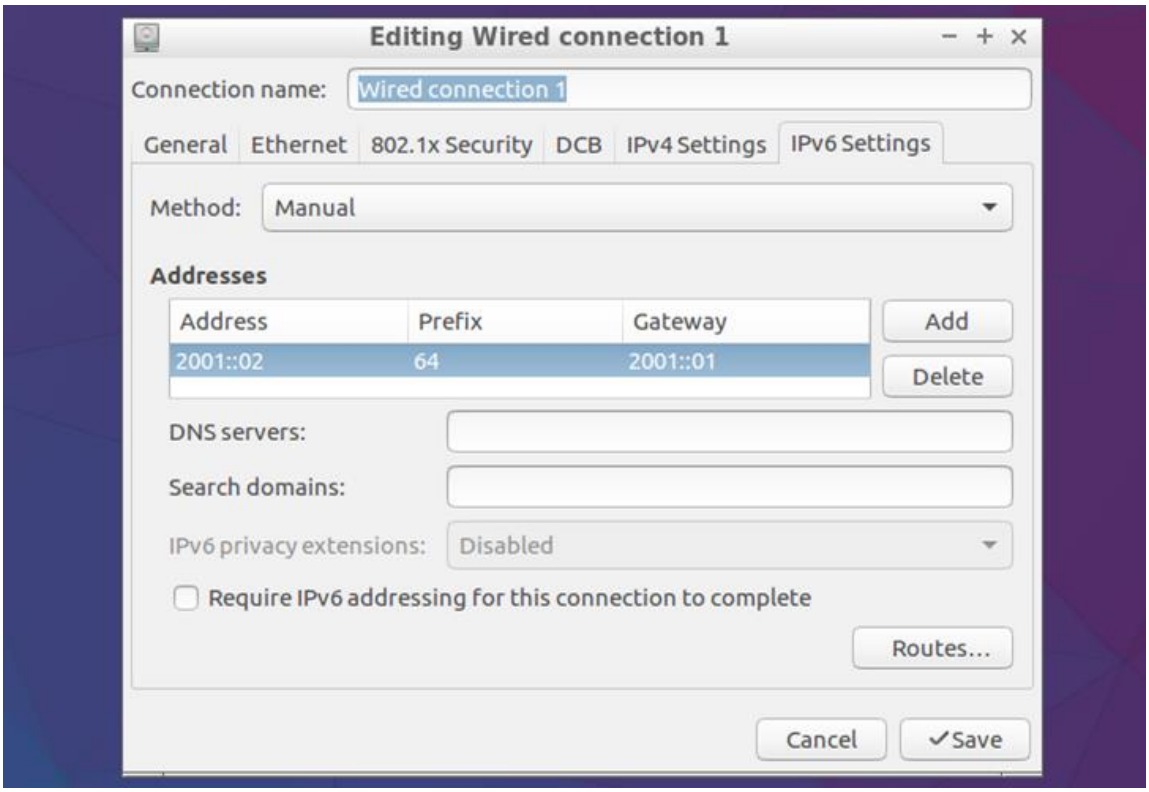
1. Create and deploy the given topology.

2. Configure the PC/Workstation IP address as mentioned in topology.

Alice
IPv6 address – 2001::02/64 , Gateway – 2001::01

Bob
IPv6 address – 2003::02/64 , Gateway – 2003::01

Example :



3. Enable IPv6 in Router-1

```
operational> configure
Entering configuration mode with exclusive access.
configure> modify parameter-group router data
Info: Parameter group instance loaded for modification.
configure> set ipv6 enable yes
configure> save
Info: Parameter group router "data" saved
configure>
```

```
Check IPv6 information in router details

operational> show router details data
```

4. Configure IPv6 interfaces in Router-1

* Configure IPv6 global address 2001::01/64 to interface if-port-1

```
operational> configure
Entering configuration mode with exclusive access.
configure> modify parameter-group interface if-port-1
Info: Parameter group instance loaded for modification.
configure> default ip ipv4
configure> enter ip ipv6
[ interface:"if-port-1" > ip > ipv6 ]
configure> show draft -e
[ interface:"if-port-1" > ip > ipv6 ]
 enable no
 address 0000:0000:0000:0000:0000:0000:0000:0000
 netmask 0000:0000:0000:0000:0000:0000:0000:0000
 peer-address 0000:0000:0000:0000:0000:0000:0000:0000
 peer-netmask 0000:0000:0000:0000:0000:0000:0000:0000
 link-local-address 0000:0000:0000:0000:0000:0000:0000:0000
 link-local-netmask 0000:0000:0000:0000:0000:0000:0000:0000
 preference 1
 metric 1
 ndp {
     cache-timeout 1200
     unsolicited-learning enable
 }
 vrrp {
     enable no
     virtual-router [+] {
     }
 }

configure> set enable yes
configure> set address 2001::01/64
configure> save
Info: Parameter group interface "if-port-1" saved
configure>
```

** Configure IPv6 global address 2002::01/64 to interface if-port-2*

```
configure> modify parameter-group interface if-port-2
Info: Parameter group instance loaded for modification.
configure> default ip ipv4
configure> set ip ipv6 enable yes
configure> set ip ipv6 address 2002::01/64
configure> save
Info: Parameter group interface "if-port-2" saved
configure> exit
```

** Verify Interface configurations*

```
operational> show interface all

 Interface name                  Status    Encaps-   IP address
                                           ulation
 ------------------------------------------------------------------------------
 if-port-1                       up        ethernet  2001::1/64
                                                     fe80::226:f7ff:fe00:6d/64
```

```
 if-port-2                       up        ethernet  2002::1/64
                                                     fe80::226:f7ff:fe00:6e/64
 if-port-3                       down      ethernet  -
 if-port-4                       down      ethernet  -
 if-port-5                       down      ethernet  -
 if-port-6                       down      ethernet  -
 if-port-7                       down      ethernet  -
 if-port-8                       down      ethernet  -
 management                      disabled  ethernet  10.0.0.12/24

 Total number of interfaces displayed : 9

operational>
```

Check IPv6 information in “show interface details” command output

```
operational> show interface details if-port-1 if-port-2
```

5. Configure IPv6 static routes in Router-1

** Configure a static route to reach 2003:00/64 network (Bob) with gateway as 2002::02(Router-2)*

```
operational> configure
Entering configuration mode with exclusive access.
configure> create parameter-group ip-route v6-route-2003-nw
Info: Parameter group instance created.
configure> show draft -e
[ ip-route:"v6-route-2003-nw" ]
*name "v6-route-2003-nw"
 enable no
 router ""
 destination 0.0.0.0
 netmask 0.0.0.0
 next-hop {
     router ""
     gateway 0.0.0.0
     label-switched-path ""
 }
 preference 30
 metric 2

configure> set enable yes
configure> set router data
configure> set destination 2003::/64
configure> set next-hop gateway 2002::02
configure> save
Info: Parameter group ip-route "v6-route-2003-nw" saved
configure>
configure>
```

6. Display IPv6 routing table in Router-1

The configured static route should appear in the IPv6 routing table

```
operational> show route summary -F ipv6 data
> IPv6 active routes
>> Destination : ::1/128
   Gateway(s)  : { ^loopback-16387
```

```
                    ::1 }
   Source       : direct
   Flags        : -
>> Destination  : 2001::/64
   Gateway(s)   : { if-port-1
                    :: }
   Source       : direct
   Flags        : -
>> Destination  : 2002::/64
   Gateway(s)   : { if-port-2
                    :: }
   Source       : direct
   Flags        : -
>> Destination  : 2003::/64
   Gateway(s)   : { if-port-2
                    2002::2 }
   Source       : static
   Flags        : -
>> Destination  : fe80::/64
   Gateway(s)   : { if-port-1
                    :: }
   Source       : direct
   Flags        : -
>> Destination  : fe80::/64
   Gateway(s)   : { if-port-2
                    :: }
   Source       : direct
   Flags        : -
 Total number of IPv6 active routes displayed : 6
 No IPv6 backup routes are available
operational>
```

7. Enable IPv6 in Router-2

```
operational> configure
Entering configuration mode with exclusive access.
configure> modify parameter-group router data
Info: Parameter group instance loaded for modification.
configure> set ipv6 enable yes
configure> save
Info: Parameter group router "data" saved
configure>
```

```
Check IPv6 information in router details

operational> show router details data
```

8. Configure IPv6 interfaces in Router-2

** Configure IPv6 global address 2003::01/64 to interface if-port-1*

```
configure> modify parameter-group interface if-port-1
Info: Parameter group instance loaded for modification.
configure> default ip ipv4
configure> set ip ipv6 enable yes
configure> set ip ipv6 address 2003::01/64
configure> save
Info: Parameter group interface "if-port-1" saved
configure> exit
```

** Configure IPv6 global address 2002::02/64 to interface if-port-2*

```
configure> modify parameter-group interface if-port-2
Info: Parameter group instance loaded for modification.
configure> default ip ipv4
configure> set ip ipv6 enable yes
configure> set ip ipv6 address 2002::02/64
configure> save
Info: Parameter group interface "if-port-2" saved
```

** Verify Interface configurations*

```
operational> show interface all
 Interface name                   Status    Encaps-   IP address
                                            ulation
 ------------------------------------------------------------------------------
 if-port-1                        up        ethernet  2003::1/64
                                                      fe80::226:f7ff:fe00:76/64
 if-port-2                        up        ethernet  2002::2/64
                                                      fe80::226:f7ff:fe00:77/64
 if-port-3                        down      ethernet  -
 if-port-4                        down      ethernet  -
 if-port-5                        down      ethernet  -
 if-port-6                        down      ethernet  -
 if-port-7                        down      ethernet  -
 if-port-8                        down      ethernet  -
 management                       disabled  ethernet  10.0.0.12/24
 Total number of interfaces displayed : 9
operational>
```

Check IPv6 information in “show interface details” command output

```
operational> show interface details if-port-1 if-port-2
```

9. Configure IPv6 static route in Router-2

** Configure a static route to reach 2001:00/64 network (Alice) with gateway as 2002::01(Router-1)*

```
operational> configure
Entering configuration mode with exclusive access.
configure> create parameter-group ip-route v6-route-2001-nw
Info: Parameter group instance created.
configure> show draft -e
[ ip-route:"v6-route-2001-nw" ]
*name "v6-route-2001-nw"
 enable no
 router ""
 destination 0.0.0.0
 netmask 0.0.0.0
 next-hop {
     router ""
     gateway 0.0.0.0
     label-switched-path ""
 }
 preference 30
 metric 2

configure> set enable yes
configure> set router data
configure> set destination 2001::/64
```

```
configure> set next-hop gateway 2002::01
configure> save
Info: Parameter group ip-route "v6-route-2001-nw" saved
configure> show draft -e
[ ip-route:"v6-route-2001-nw" ]
*name "v6-route-2001-nw"
 enable yes
 router "data"
 destination 2001:0000:0000:0000:0000:0000:0000:0000
 netmask ffff:ffff:ffff:ffff:0000:0000:0000:0000
 next-hop {
     router ""
     gateway 2002:0000:0000:0000:0000:0000:0000:0001
     label-switched-path ""
 }
 preference 30
 metric 2
configure>
```

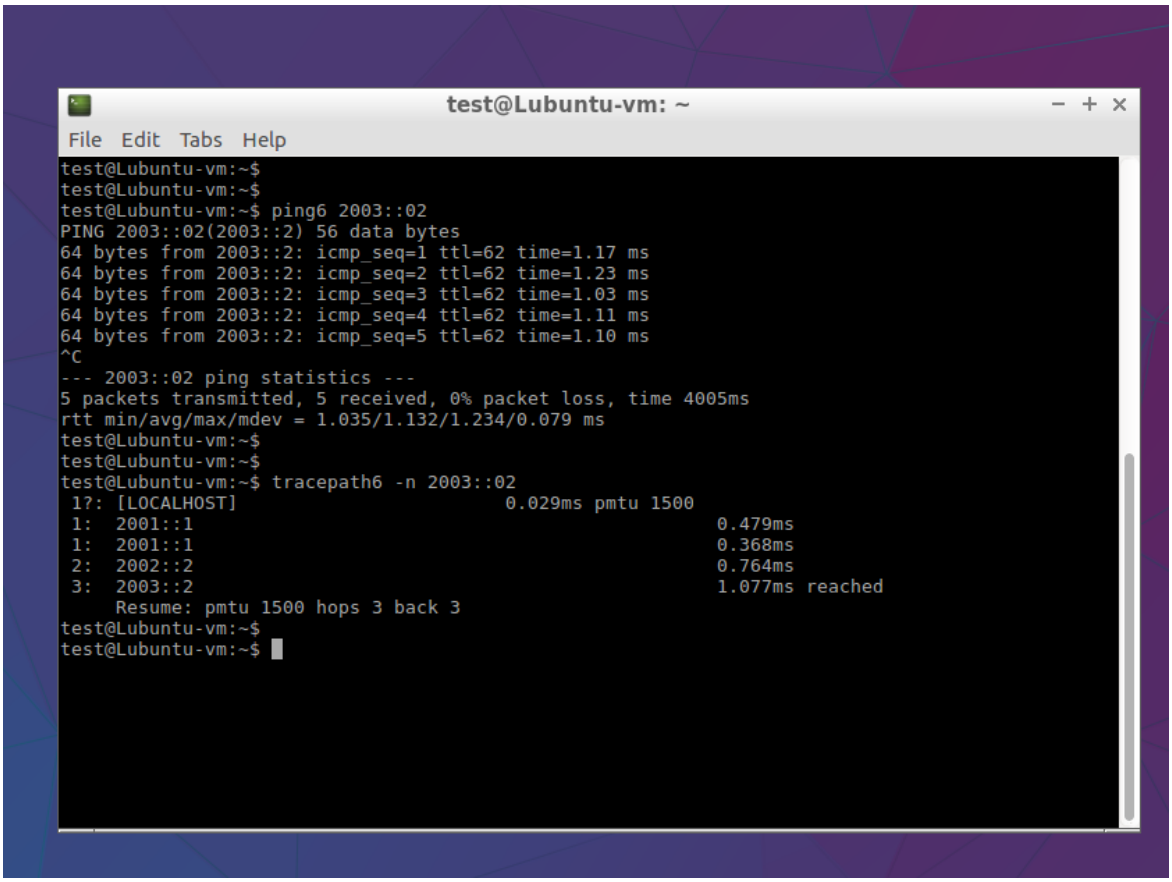
10. Display IPv6 routing table in Router-2

```
operational> show route summary -F ipv6 data
> IPv6 active routes
>> Destination : ::1/128
   Gateway(s)  : { ^loopback-16387
                   ::1 }
   Source      : direct
   Flags       : -
>> Destination : 2001::/64
   Gateway(s)  : { if-port-2
                   2002::1 }
   Source      : static
   Flags       : -
>> Destination : 2002::/64
   Gateway(s)  : { if-port-2
                   :: }
   Source      : direct
   Flags       : -
>> Destination : 2003::/64
   Gateway(s)  : { if-port-1
                   :: }
   Source      : direct
   Flags       : -
>> Destination : fe80::/64
   Gateway(s)  : { if-port-1
                   :: }
   Source      : direct
   Flags       : -
>> Destination : fe80::/64
   Gateway(s)  : { if-port-2
                   :: }
   Source      : direct
   Flags       : -
 Total number of IPv6 active routes displayed : 6
 No IPv6 backup routes are available
operational>
```

11. Verify traffic flow between Alice and Bob

* From Alice workstation ping Bob, observe the packet from and TTL in ping reply

* From Alice workstation run tracepath to Bob's IP. Observer the intermediate hops



12. Check IPv6 NDP table on Router-1

This is similar to ARP Table in IPv4.

```
operational> show ipv6 neighbour summary data

 Host address                                MAC address       Interface
 ------------------------------------------------------------------------------
 2002::1                                     00:26:f7:00:00:6e if-port-2
 2003::2                                     00:26:f7:00:09:3c if-port-1
 fe80::226:f7ff:fe00:6e                      00:26:f7:00:00:6e if-port-2
 fe80::5d97:cf2f:4a3:d8cb                    00:26:f7:00:09:3c if-port-1

 Total number of NDP entries displayed : 4

operational>
```

13. Verify auto-configured Link Local Address on IPv6 interfaces

All IPv6 enabled interfaces will have a link-local address. IPv6 link-local address is a unicast address that is configured automatically using the prefix FE80::/10 and port MAC in the modified EUI-64 format. The link-local address can also be manually configured.
Link-local addresses are used for a addressing on a single physical link. These addresses can be used to reach the neighboring nodes attached to the same link. Routers will not forward packets using link-local addresses. Two routers can have same link-local address and can still communicate over directly connected network. But, the global unicast address should be unique in a network as they are routable.

Login to Router-1 and check the auto-configured link local address.

For Example :

```
operational> show interface details if-port-1

> Interface : if-port-1
 General Information
 -------------------
 ID                     : 21
 Encapsulation          : ethernet
 MTU                    : 1500
 Base port type         : fast-ethernet
 Base port location     : { shelf-1 { active-controller base-slot } port-1 }
 State Information
 -----------------
 State                  : up
 Last state transition  : 15:19:44, Monday, March 18, 2019 IST
 Work flags             : -- -- ----- -----
 Ethernet information
 --------------------
 VLAN tagging           : disabled
 IP information
 --------------
 Router                 : data
 IPv6 information
 ----------------
 Address                : 2001::1
 Netmask                : ffff:ffff:ffff:ffff::
 Link local Address     : fe80::226:f7ff:fe00:6d <====== Combination of FE08 and port MAC
 Link local Netmask     : ffff:ffff:ffff:ffff::
 Scope Zone             : 33488917
 Preference             : 1
 Metric                 : 1
 TE information
 --------------
 Maximum Bandwidth                  : 10000 kbps
 Maximum Reservable Bandwidth       : 10000 kbps
 Update threshold percentage        : 10

operational>

operational> show fast-ethernet details { shelf-1 { active-controller base-slot } port-1 }

> Port : { shelf-1 { active-controller base-slot } port-1 }
 Port details
 ------------
 Name                :
 MAC address         : 00:26:f7:00:00:6d  <==============
 POST                : passed
 Media               : copper
 Loop back mode      : no-loopback
 State               : up
 Duplex mode         : half-duplex
 Speed               : ten-mbps
 Work flags          : ---- ----
operational>
```

14. Check the connectivity between Router-1 and Router-2 using Link Local Address

Login to Router-2 and get the link-local address of interface connected to Router-1.

Now, Login to Router-1 and ping the link-local address on Router-2 and observe the response. When pinging link-local address, the the name if out-going interface should be specified in the command. If no interface or wrong interface name is specified, ping will result in error or unsuccessful.

Example:

```
Router-1

operational> ping data:fe80::226:f7ff:fe00:77%if-port-2
PING fe80:0:1ff:16:226:f7ff:fe00:6e --> fe80::226:f7ff:fe00:77%33488918
16 bytes from fe80::226:f7ff:fe00:77%33488918: icmp_seq=0 hoplimit=64 time=0.496
ms
16 bytes from fe80::226:f7ff:fe00:77%33488918: icmp_seq=1 hoplimit=64 time=0.505
ms
16 bytes from fe80::226:f7ff:fe00:77%33488918: icmp_seq=2 hoplimit=64 time=0.470
ms
16 bytes from fe80::226:f7ff:fe00:77%33488918: icmp_seq=3 hoplimit=64 time=0.427
ms
16 bytes from fe80::226:f7ff:fe00:77%33488918: icmp_seq=4 hoplimit=64 time=0.475
ms

---- PING Statistics----
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.000/0.475/0.505/0.027 ms
operational>


operational> ping -c 5 data:fe80::226:f7ff:fe00:77
Error: No source address found for this destination
operational>

operational>  ping data:fe80::226:f7ff:fe00:77%if-port-1
PING fe80:0:1ff:15:226:f7ff:fe00:6d --> fe80::226:f7ff:fe00:77%33488917

---- PING Statistics----
8 packets transmitted, 0 packets received, 100.0% packet loss
operational>
```

Computer Systems Laboratory – UE18CS304

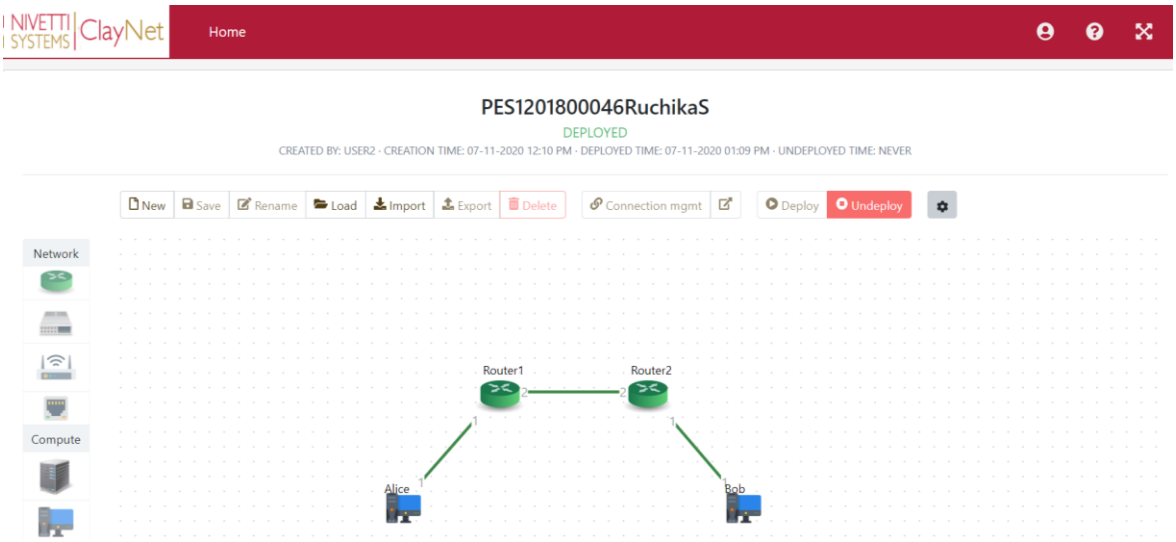
PES1201800046 – Ruchika Shashidhara – Sem 5 Sec D

Week #9 – IPv6 Static Addressing and Routing in ClayNet – 7th November 2020

Objectives:

- Perform basic IPv6 configurations on a Desktop and Router.
- Distinguish between IPv4 and IPv6 addresses
- Configure & observing the traffic flow using IPv6 static routes
- Understanding IPv6 Link Local Address and working with ping6 and tracepath6

Topology



Host Configuration Details:

Host	IPv6 Global Addr	Gateway	IPv6 Link Address
Alice	2001::02/64	2001::01/64	fe80::cf0d:6754:cf8d:a7bc /64
Bob	2003::02/64	2003::01/64	fe80::a58c:3e9d:d56a:3bc7 /64

Router Configuration Details:

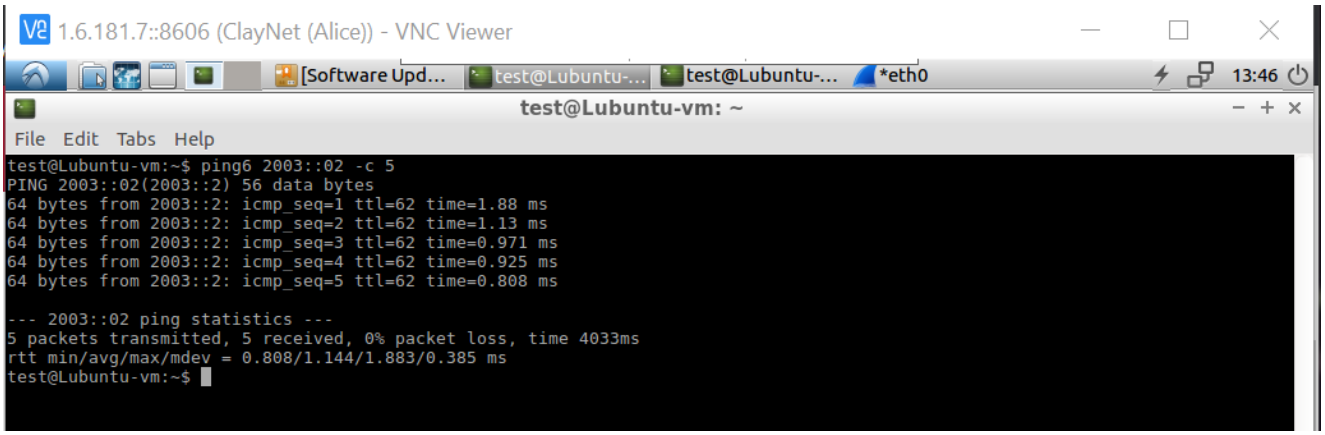
Router	Port	IPv6 Address	Link local Addr
Router1	Port 1	2001::01/64	fe80::cf08:6754:cf8d:a7bc /64
Router1	Port 2	2002::01/64	fe80::a026:ff:fe00:683/64
Router2	Port 1	2003::01/64	fe80::a58c:3e9d:d56a:3bc7/64
Router2	Port 2	2002::02/64	fe80::a026:ff:fe00:67a/64

Routing Table Entries:

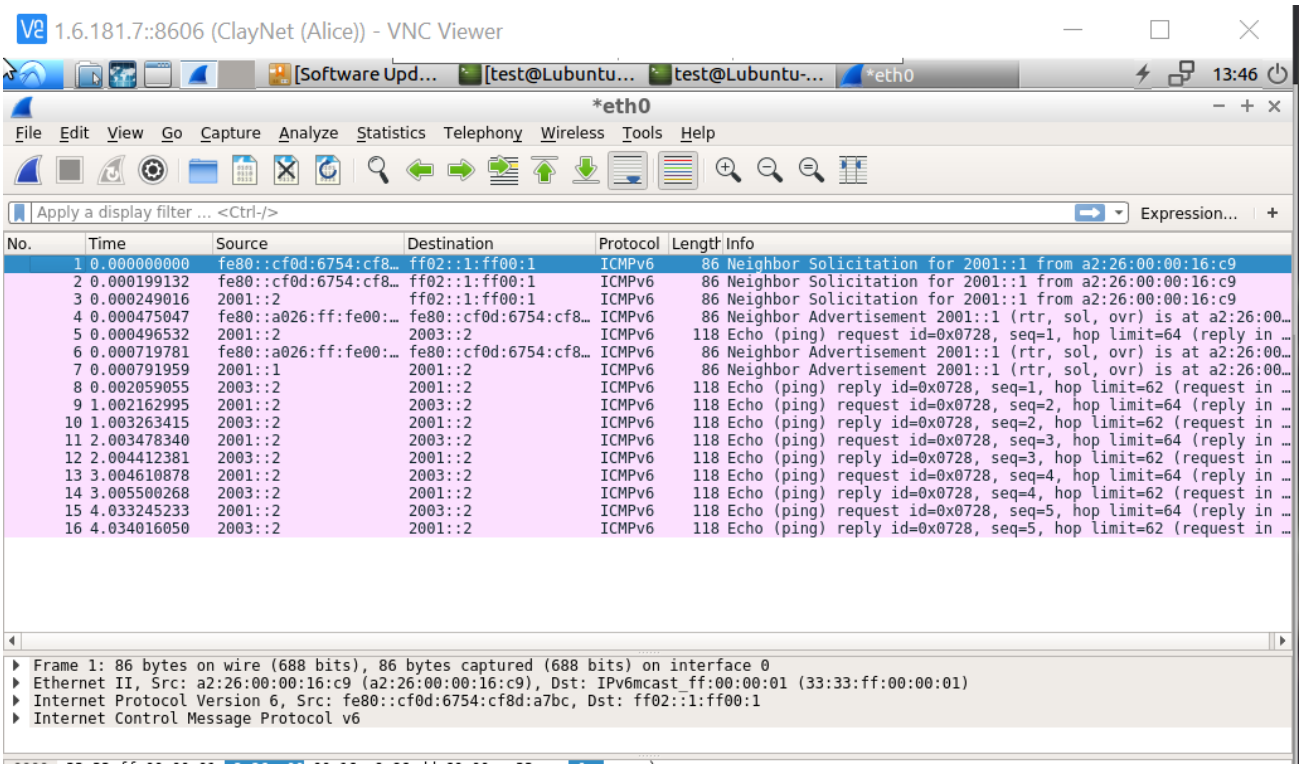
Router	Destination	Next Hop Gateway	Via
Router1	2003::/64	2002::02/64	Static
Router2	2001::/64	2002::01/64	Static

Observations from Alice's workstation (2001::02/64), pinging to Bob's workstation (2003::02/64):

$ ping 2003::02 -c 5



Wireshark Capture for ping 2003::02 -c 5



Protocol: ICMPv6

Total no. of Ping Requests: 5

Total no. of Ping Replies: 5

Packet Loss: 0%

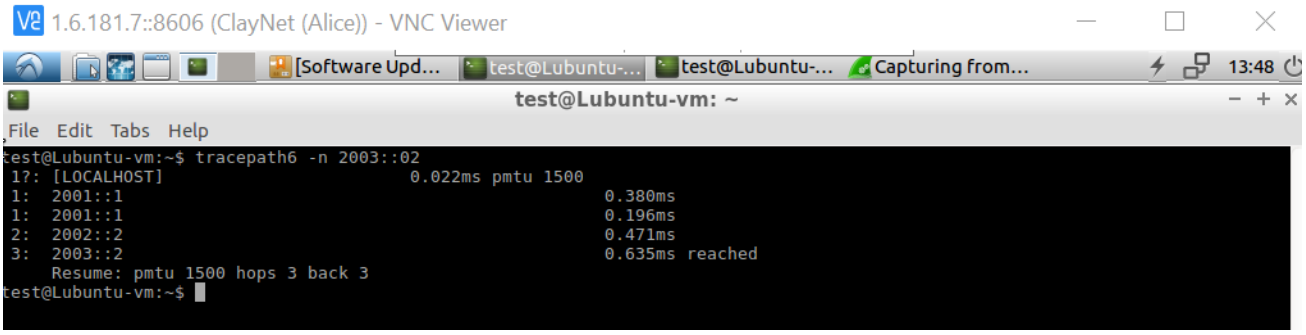
TTL :62

Min RTT: 0.808

Average RTT: 1.144

Observations of Traceroute from Alice's workstation (2001::02/64) to Bob's workstation (2003::02/64):

$ tracepath6 -n 2003::02



Traceroute from Alice's Workstation (2001::02/64) to Bob's Workstation (2003::02/64):

Total Number of Hops: 3 – from Router1 (2001::02/64), Router2 (2002::02/64) and then to Bob's workstation (2003::02/64)

Observations with IPv6 Link Local Address:

Verifying Link Local Address configurations on Router1 & Router2:

IPv6 NDP Table on Router1

```
operational> show ipv6 neighbour summary data

 Host address                              MAC address       Interface
 ------------------------------------------------------------------------------
 2001::2                                   a2:26:00:00:16:c9 if-port-1
 2002::2                                   a2:26:00:00:06:83 if-port-2
 fe80::a026:ff:fe00:683                    a2:26:00:00:06:83 if-port-2
 fe80::cf0d:6754:cf8d:a7bc                 a2:26:00:00:16:c9 if-port-1

 Total number of NDP entries displayed : 4
```

IPv6 NDP Table on Router2

```
operational> show ipv6 neighbour summary data

 Host address                              MAC address       Interface
 ------------------------------------------------------------------------------
 2002::1                                   a2:26:00:00:06:7a if-port-2
 2003::2                                   a2:26:00:00:16:cc if-port-1
 fe80::a026:ff:fe00:67a                    a2:26:00:00:06:7a if-port-2
 fe80::a58c:3e9d:d56a:3bc7                 a2:26:00:00:16:cc if-port-1

 Total number of NDP entries displayed : 4
```

Checking connectivity from Router1 & Router2 using Link Local Address from Router1

$ ping -c 5 data:fe80::a026:ff:fe0067a

```
operational> ping -c 5 data:fe80::a026:ff:fe00:67a%if-port-2
PING ::1 --> fe80::a026:ff:fe00:67a%33488916
16 bytes from fe80::a026:ff:fe00:67a: icmp_seq=0 hoplimit=64 time=0.084 ms
16 bytes from fe80::a026:ff:fe00:67a: icmp_seq=1 hoplimit=64 time=0.060 ms
16 bytes from fe80::a026:ff:fe00:67a: icmp_seq=2 hoplimit=64 time=0.061 ms
16 bytes from fe80::a026:ff:fe00:67a: icmp_seq=3 hoplimit=64 time=0.063 ms
16 bytes from fe80::a026:ff:fe00:67a: icmp_seq=4 hoplimit=64 time=0.060 ms

---- PING Statistics----
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.000/0.066/0.084/0.009 ms
```

Conclusion:

- Built a simple Network using Desktops, Routers with basic IPv6 Configurations
- Learnt about static IPv6 routing behaviour with default & static routes & routing tables.
- IPv4 is a numeric address separated by "." which uses ARP to map MAC Address and needs to be configured before network connection, whereas IPv6 is a hexadecimal address separated by ":" which uses NDP to map MAC Address and can be configured based on functionalities needed.
- With **ping 2003::02,** Round-trip-time (RTT) keeps getting decremented as 2 routers leads to 3 hops in between.
- With **tracepath -n 30.30.30.2** from Alice's Workstation, we can see that it takes a total of 3 hops from Router1 (2001::02/64), Router2 (2002::02/64) and then finally to reach Bob's workstation (2003::02/64)
- All IPv6 enabled interfaces have a link-local address which is configured automatically using the prefix fe80:: and port MAC in the modified EUI-64 format.
- The link-local address can also be manually configured for addressing on a single physical link. These addresses can be used to reach the neighbouring nodes attached to the same link but Routers cannot forward packets using link-local addresses. Hence, two routers can have same link-local address and can still communicate over directly connected network.
- When pinging the link-local address, the if-out-going interface is specified in the command & we can check the connectivity between Router1 and Router2 from any router.