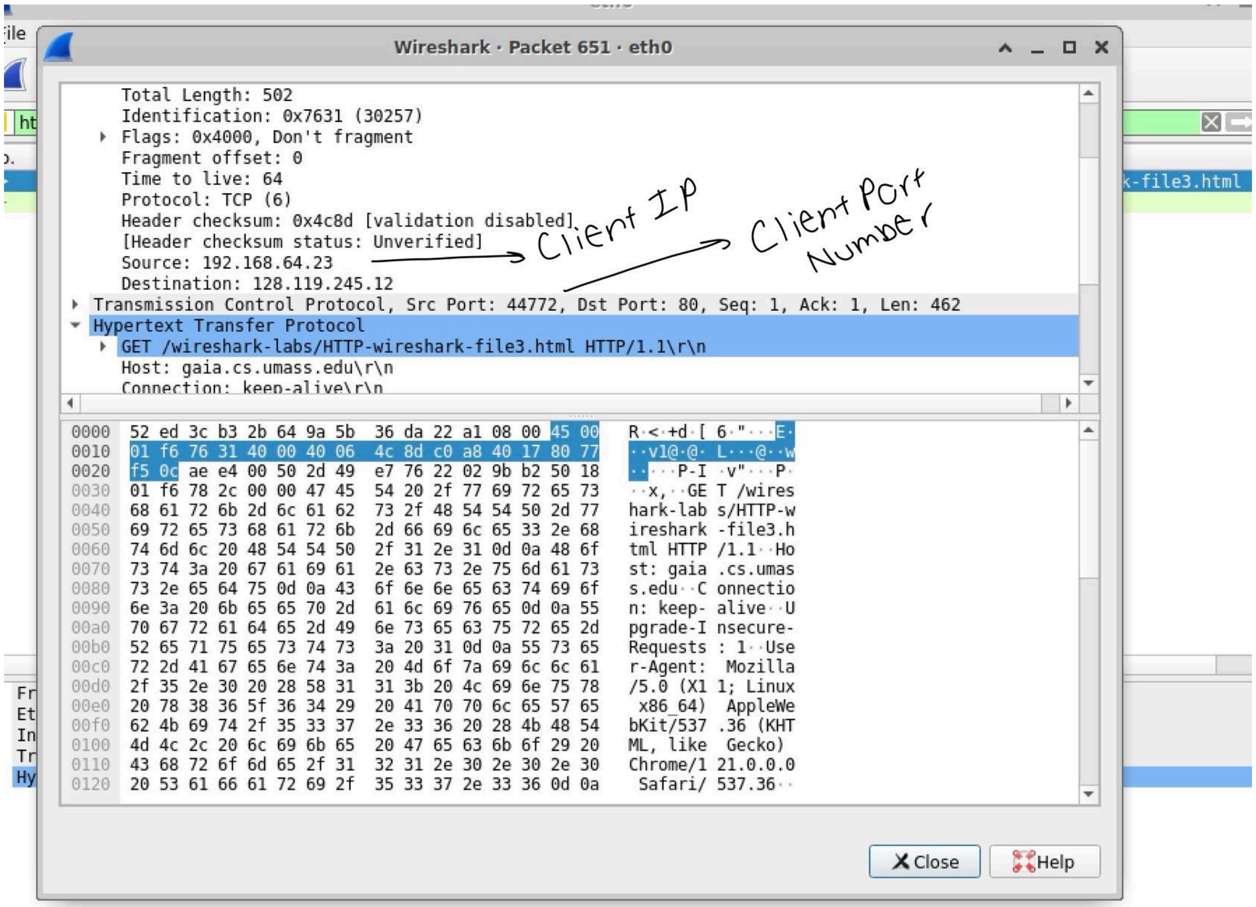
Pre-Lab

Table 1: Match the letter next to the Header with its Definition/Usage in "Answer Here"

	Header	**Match** ← →	**Answer Here**	**Definition/Usage**
A	"Accept-Encoding"		C	Carries credentials to authenticate the requestor.
B	"Accept"		D	Contains data from the server to be stored by the client.
C	"Authorization" X		F	Describes the compression formats that can be decoded.
D	"Cache-Control"		S	Details preferred content types for the response.
E	"Connection"		O	Directive to the client to store the cookie.
F	"Content-Encoding"		I	Directs how responses are to be stored by caches.
G	"Content-Length"		P	Information about the client software making the request.
H	"Content-Type" X		T	Numeric indication of the outcome of the server's attempt to fulfill the request.
I	"Cookie" X		K	Only send the response if the content has been modified since this date/time.
J	"Host"		E	Options for managing the current connection state.
K	"If-Modified-Since" X		L	Parameters for a persistent connection.
L	"Keep-Alive" X		A	Specifies the encoding used on the data.
M	"Last-Modified" X		R	Text description accompanying the status code, explaining the status.
N	"Server"		J	The domain name of the server and the TCP port number.
O	"Set-Cookie" X		G	The exact byte length of the HTTP body.
P	"User-Agent" X		H	The media type of the body of the request/response.
Q	"Request Version" X		N	The name of the server software responding to the request.
R	"Response Phrase"		Q	The protocol version used by the client in making the request.
S	"Request Method"		M	The time at which the content was last changed.
T	"Status Code" X		B	The type of action the client wants the server to take on a resource.

Q1.)



```
    Total Length: 502
    Identification: 0x7631 (30257)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x4c8d [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.64.23
    Destination: 128.119.245.12
▸ Transmission Control Protocol, Src Port: 44772, Dst Port: 80, Seq: 1, Ack: 1, Len: 462
▾ Hypertext Transfer Protocol
  ▸ GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    Connection: keep-alive\r\n
```

```
0000  52 ed 3c b3 2b 64 9a 5b  36 da 22 a1 08 00 45 00   R·<·+d·[ 6·"···E·
0010  01 f6 76 31 40 00 40 06  4c 8d c0 a8 40 17 80 77   ··v1@·@· L···@··w
0020  f5 0c ae e4 00 50 2d 49  e7 76 22 02 9b b2 50 18   ·····P-I ·v"···P·
0030  01 f6 78 2c 00 00 47 45  54 20 2f 77 69 72 65 73   ··x,··GE T /wires
0040  68 61 72 6b 2d 6c 61 62  73 2f 48 54 54 50 2d 77   hark-lab s/HTTP-w
0050  69 72 65 73 68 61 72 6b  2d 66 69 6c 65 33 2e 68   ireshark -file3.h
0060  74 6d 6c 20 48 54 54 50  2f 31 2e 31 0d 0a 48 6f   tml HTTP /1.1··Ho
0070  73 74 3a 20 67 61 69 61  2e 63 73 2e 75 6d 61 73   st: gaia .cs.umas
0080  73 2e 65 64 75 0d 0a 43  6f 6e 6e 65 63 74 69 6f   s.edu··C onnectio
0090  6e 3a 20 6b 65 65 70 2d  61 6c 69 76 65 0d 0a 55   n: keep- alive··U
00a0  70 67 72 61 64 65 2d 49  6e 73 65 63 75 72 65 2d   pgrade-I nsecure-
00b0  52 65 71 75 65 73 74 73  3a 20 31 0d 0a 55 73 65   Requests : 1··Use
00c0  72 2d 41 67 65 6e 74 3a  20 4d 6f 7a 69 6c 6c 61   r-Agent:  Mozilla
00d0  2f 35 2e 30 20 28 58 31  31 3b 20 4c 69 6e 75 78   /5.0 (X1 1; Linux
00e0  20 78 38 36 5f 36 34 29  20 41 70 70 6c 65 57 65    x86_64)  AppleWe
00f0  62 4b 69 74 2f 35 33 37  2e 33 36 20 28 4b 48 54   bKit/537 .36 (KHT
0100  4d 4c 2c 20 6c 69 6b 65  20 47 65 63 6b 6f 29 20   ML, like  Gecko)
0110  43 68 72 6f 6d 65 2f 31  32 31 2e 30 2e 30 2e 30   Chrome/1 21.0.0.0
0120  20 53 61 66 61 72 69 2f  35 33 37 2e 33 36 0d 0a    Safari/ 537.36··
```

What is the Client browser and OS Information:

- User-Agent: Mozilla/5.0 (x11; Linux x86_64) AppleWebKit/537.26 Chrome/1 21.0.0.0

What type of content is accepted by the Client?

- text/html,images,webp,png

What compressed data does the Client Accept?

- Accept-Encoding: gzip, deflate

B.)

What is the Server IP Address:

- 128.119.245.12

What port number is the Server listening on

-80

Are the Client and Server using the same version of HTTP?

- Yes the Server and Client are utilizing the same HTTP Version as the client request HTTP Message utilize version 1.1 for it's request and the Server responded in the status line with HTTP/1.1 200 OK

What is the code that indicates the result of the client's request ?

- The code 200 OK, Indicating the request succeeded and the information is returned in response

What is the status phrase that accompanies the above code?

  - OK, indicating that the server received and is sending the requested objects

What is the date and time of the server response?

-Date: Mon, 14 Oct 2024, 15:15:05 GTM

What is the server software information?

    - Server: Apacher/2.4.5 (CentOS) OpenSSL/1.0.2k-fips

What is the length of the request file:

    - Length: 4500 bytes

What response does the Server provide regarding the management of the network connection

  - Connection: Keep-Alive
  - With the configuration of the Keep-Alive: timeout= 5, max = 100
  - This means the server keeps the connection alive for 5 seconds and at max 100 requests can be made over the persistent connection

What type of content is being sent by the server?

    - text/html

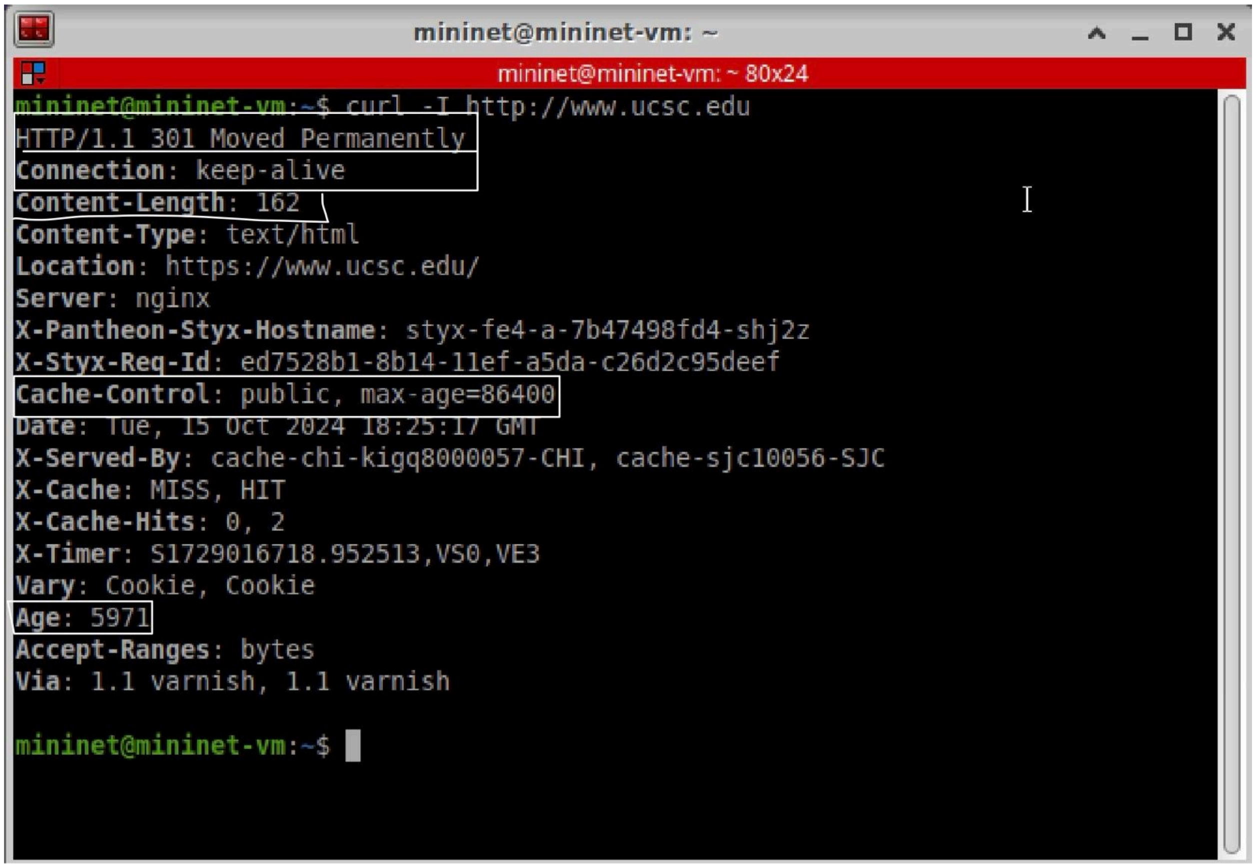
C.) Protocol Layers:

- How do application and transport laters interact differently in HTTP and HTTPS
- HTTP:
  - Application Layer
- HTTPS:
  - Utilizes the TLS Application for its network layer protocol, utilizing TLS requires the Browers / Client to verify the hostname with the server certification information
- If the bank transmitted a password over HTTP, would it be visible in a Wireshark Capture?
- How does this visibility differ transmitted when using HTTPS
  - The visibility differs by the Transport layer protocol TLS Encrypting the Data making it unreadable to the human view
- List all the proposals that make these two captures different. Focus on identifying protocols that contribute to security in HTTPS

Q2.)

a.) What is the full command you use

-curl -I http://www.ucsc.edu



```
mininet@mininet-vm:~$ curl -I http://www.ucsc.edu
HTTP/1.1 301 Moved Permanently
Connection: keep-alive
Content-Length: 162
Content-Type: text/html
Location: https://www.ucsc.edu/
Server: nginx
X-Pantheon-Styx-Hostname: styx-fe4-a-7b47498fd4-shj2z
X-Styx-Req-Id: ed7528b1-8b14-11ef-a5da-c26d2c95deef
Cache-Control: public, max-age=86400
Date: Tue, 15 Oct 2024 18:25:17 GMT
X-Served-By: cache-chi-kigq8000057-CHI, cache-sjc10056-SJC
X-Cache: MISS, HIT
X-Cache-Hits: 0, 2
X-Timer: S1729016718.952513,VS0,VE3
Vary: Cookie, Cookie
Age: 5971
Accept-Ranges: bytes
Via: 1.1 varnish, 1.1 varnish

mininet@mininet-vm:~$
```

b.) Does the server have the requested file? How do you know?

- The server does not contain the requested file as per the 301 Moved Permanently status message, which states the HTML file was moved and could not be retrieved from the requested URL, however, the link to the new URL is Location: https://www.ucsc.edu/

- How long has this response been cached? How is this information used?

    - The response has been cached for Age: 5971 seconds, the information is utilized with the Cache-Control: Max-Age = 86400 to ensure the user received a current version of the requested objects

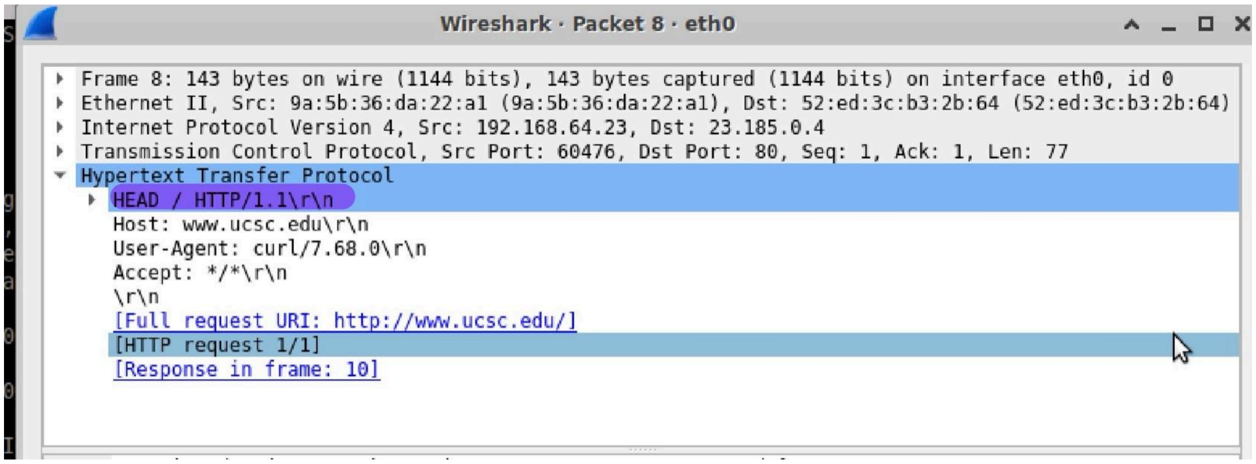
- What type of HTTP Connection is used? Explain what is means

    - The Connection: Keep-alive was utilized to maintain a connection between the client and server, to continuously send objects through the same connection without the process of reestablishing a connection everytime the user requests an object from the server

- What is the size(in bytes) of the base HTML paged?

    - The size is Content-Lenght: 162 bytes

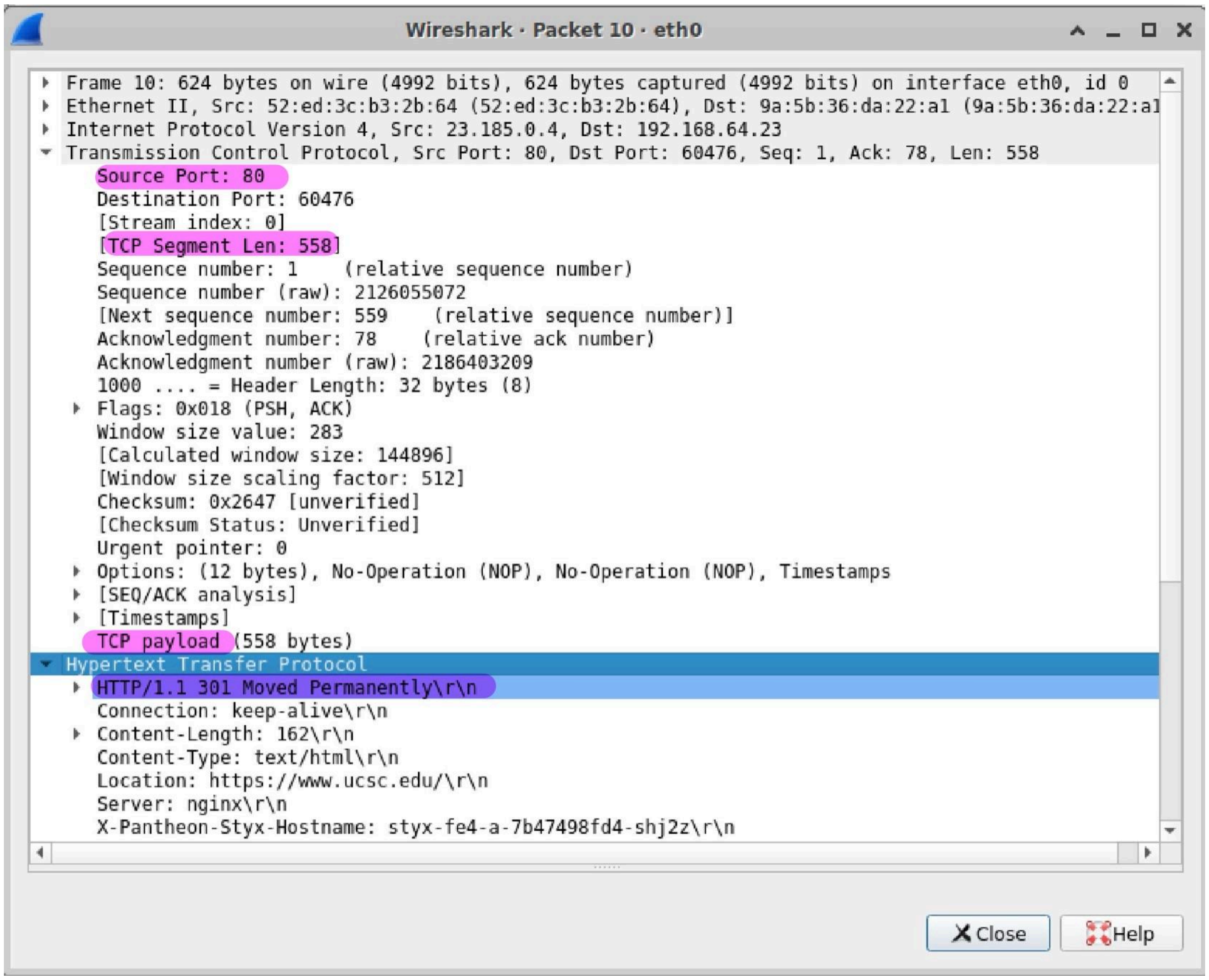
c.)



```
Wireshark · Packet 8 · eth0
▸ Frame 8: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits) on interface eth0, id 0
▸ Ethernet II, Src: 9a:5b:36:da:22:a1 (9a:5b:36:da:22:a1), Dst: 52:ed:3c:b3:2b:64 (52:ed:3c:b3:2b:64)
▸ Internet Protocol Version 4, Src: 192.168.64.23, Dst: 23.185.0.4
▸ Transmission Control Protocol, Src Port: 60476, Dst Port: 80, Seq: 1, Ack: 1, Len: 77
▾ Hypertext Transfer Protocol
  ▸ HEAD / HTTP/1.1\r\n
    Host: www.ucsc.edu\r\n
    User-Agent: curl/7.68.0\r\n
    Accept: */*\r\n
    \r\n
    [Full request URI: http://www.ucsc.edu/]
    [HTTP request 1/1]
    [Response in frame: 10]
```

-Which HTTP method is used for the client request? Explain the purpose of the HTTP Method you observed

- The client requested utilizing the HEAD method, the HEAD method is utilized to send the header information of the GET method without sending the object itself d.)


Wireshark · Packet 10 · eth0

```
▸ Frame 10: 624 bytes on wire (4992 bits), 624 bytes captured (4992 bits) on interface eth0, id 0
▸ Ethernet II, Src: 52:ed:3c:b3:2b:64 (52:ed:3c:b3:2b:64), Dst: 9a:5b:36:da:22:a1 (9a:5b:36:da:22:a1
▸ Internet Protocol Version 4, Src: 23.185.0.4, Dst: 192.168.64.23
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 60476, Seq: 1, Ack: 78, Len: 558
    Source Port: 80
    Destination Port: 60476
    [Stream index: 0]
    [TCP Segment Len: 558]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 2126055072
    [Next sequence number: 559    (relative sequence number)]
    Acknowledgment number: 78    (relative ack number)
    Acknowledgment number (raw): 2186403209
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 283
    [Calculated window size: 144896]
    [Window size scaling factor: 512]
    Checksum: 0x2647 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (558 bytes)
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 301 Moved Permanently\r\n
    Connection: keep-alive\r\n
  ▸ Content-Length: 162\r\n
    Content-Type: text/html\r\n
    Location: https://www.ucsc.edu/\r\n
    Server: nginx\r\n
    X-Pantheon-Styx-Hostname: styx-fe4-a-7b47498fd4-shj2z\r\n
```

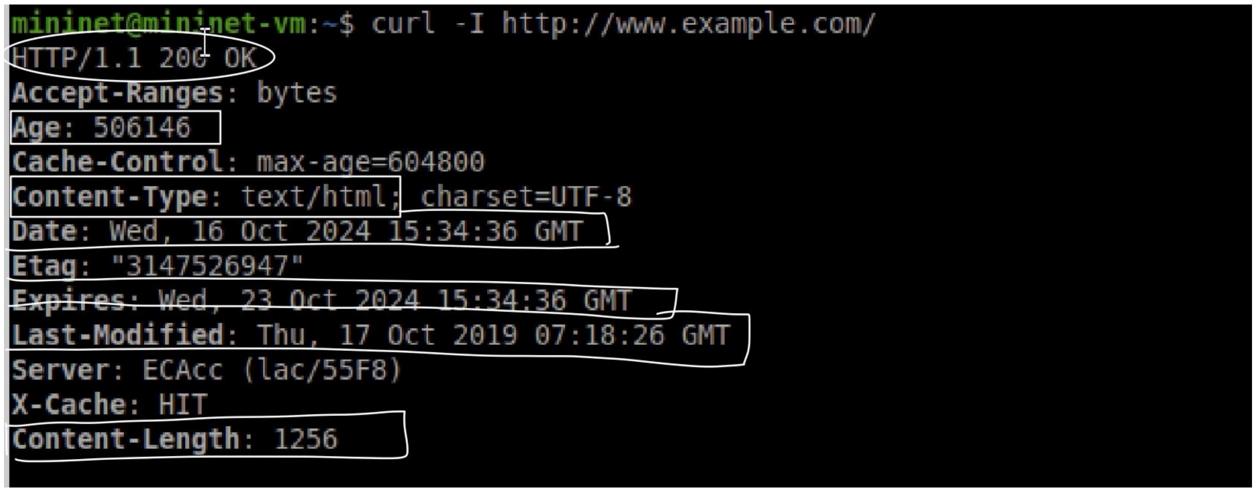
-What is the status code of the response message from the server? What does it mean?
- The Status Code 301 Moved Permanently status message, states the HTML file was moved and could not be retrieved from the requested url, however, the link to the new URL is Location: https://www.ucsc.edu/

- On what port number is the server listening for HTTP requests? Explain why this port is being used
- The server is utilizing port number 80, the default port for the server to listen for HTTP requests over the internet
-Which transport protocol was used for these requests?
- TCP is being utilzied

e.) Explain the concept of a TCP stream in the context of loading a webpage

- TCP Utilizes a byte stream for data transmission by treating the data (web page content) as a continuous flow of bytes, this allows for seamless transmission allowing for the browser to continually read data allowing for faster rendering of the web page and it's content

f.) Now use curl to load the response headers of http://www.example.com/. Describe the response displayed in the output terminal



```
mininet@mininet-vm:~$ curl -I http://www.example.com/
HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 506146
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Wed, 16 Oct 2024 15:34:36 GMT
Etag: "3147526947"
Expires: Wed, 23 Oct 2024 15:34:36 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (lac/55F8)
X-Cache: HIT
Content-Length: 1256
```

- The response from the host www.example.com is the 200 OK response code, which informs us that the server request succeeded and that the server is sending over the requested objects. The current age of the requested webpage is Age: 506146 seconds, the Cache-Control: has the mage as set to 604800 when the current age meets this value the original server will send an updated copy. The content type of the requested object is text/html. The server responded on Wed, October 16, 8:34 am. The last time the web page was modified on the server was on Thursday, October 17, 2019, almost 5 years ago. The length of the content of the request objects is 1256 bytes.

Q3.)

Describe the purpose of the purpose of the very first GET request, and the corresponding response message.

- The purpose of the very first GET Request is to request the base HTML file of the web page

Was it successful?

- Yes the request was successful as the server responded with 200 OK, stating the request was successful and is sending over the requested object

If successful, how many bytes of the requested file does the Server send?

- File Data: 942 Bytes

Copy the lines from the base HTML page that reference the two images

- <img src="http://gaia.cs.umass.edu/pearson.png" WIDTH="140" HEIGHT="82" > </p>\n
- <p align="left"><img src="http://kurose.cslash.net/8E_cover_small.jpg"\n

Enter each URL separately into your browser. What do you see?

- Pasting the URL in the browser, shows only the image itself

From the Wireshark trace, how many HTTP GET request messages did your browser send in total?

- In total, 3 HTTP Get request messages are shown, 1 for the base html file, and 2 more for the embedded image in the HTML file

One image is hosted on the same server as the Main HTML file. How can you prove which image is hosted on a different server?

- The image that is hosted on the same server as the HTML File both have the IP Address 192.168.64.23, while the image stored on a different server has a the different IP address of 178.79.137.164

Which Image is Hosted on a different server?

- The Image 8E_cover_small.jpg

Q4.)

a.) Why might your browser issue a Conditional get?

- The browser issues a conditional get as it sees there is already a cached version of the request object on the server, and wants to ensure it's sending the most updated version to the client

b.) Is your browser's first HTTP Get request to the server a Conditional Get? Explain your answer referring to the Wireshark trace.

- The first HTTP Get Request to the server is not the conditional get request, as this is the first request the client is initiating the request, and the server responded with a 200 OK, we know the second request is a conditional get as the response from the server is 304 Not modified

c.)

Describe how the second HTTP Get Message is different from the first one.

- The second HTTP Get Message is different as it contains the additional headers of Cache-Control: Max-Age = 0, If-None-Match: and If-Modified-Since: Thu, 17 Oct 2024

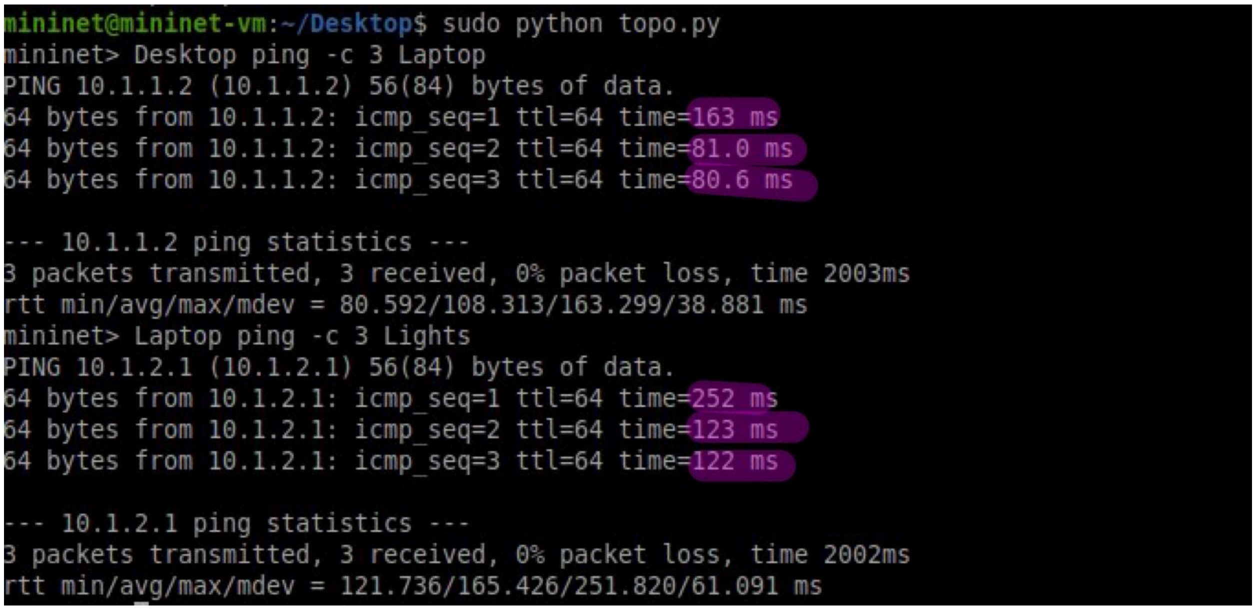
What is the "IF-Modified-since" header and its value, and why is it used?

- The IF-Modified-Since header is the header that stores the time the server information on the last modifies, it is used to compare time values to ensure the current object is up to date

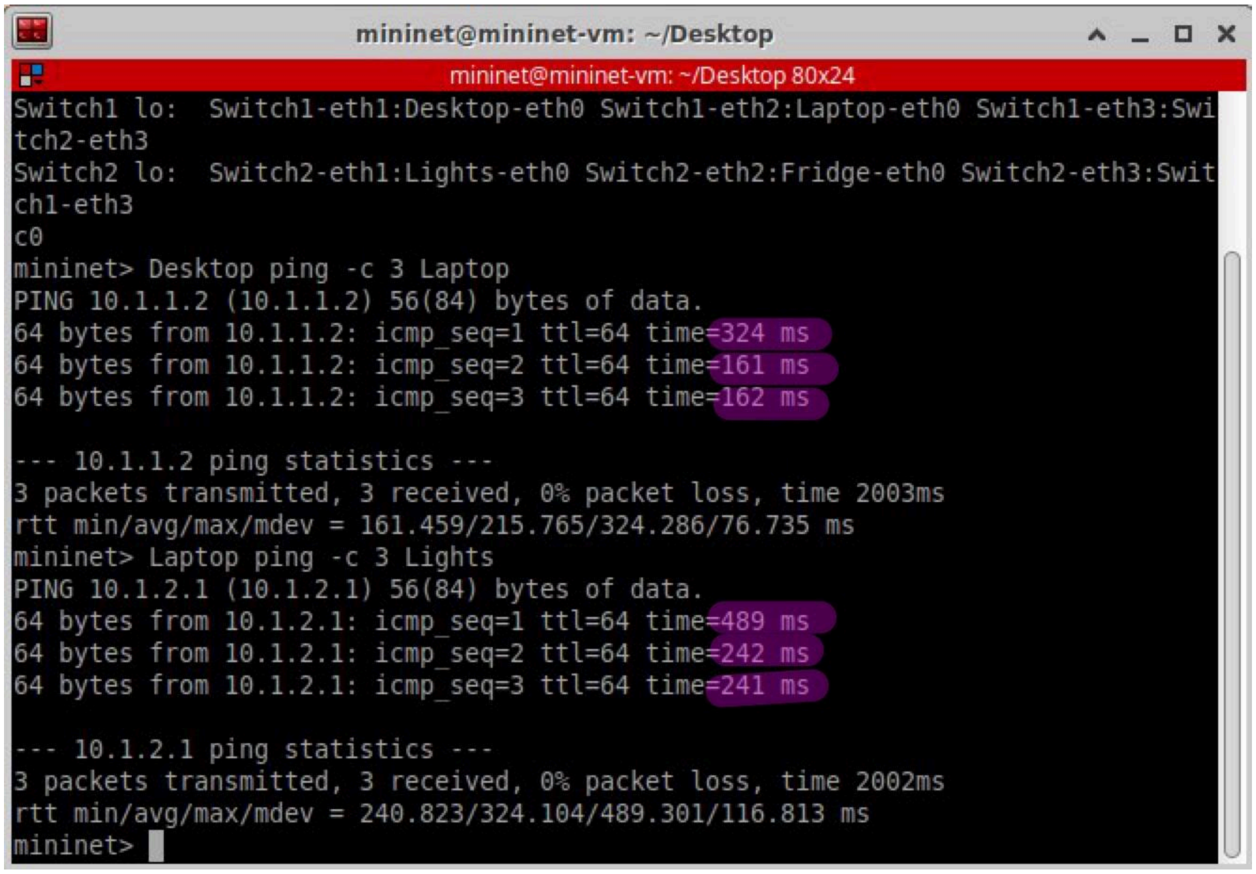
What is the date and its value, what does it mean?

- If-Modified-Since: Thur, 17 Oct 2024 05:59:02 GMT
- This is the time the value of the original request the server sent to the Client, the client is utilizing this information to compare with the file stored in the server to ensure it's the most up to date file

Q5.)



```
mininet@mininet-vm:~/Desktop$ sudo python topo.py
mininet> Desktop ping -c 3 Laptop
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=163 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=81.0 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=64 time=80.6 ms

--- 10.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 80.592/108.313/163.299/38.881 ms
mininet> Laptop ping -c 3 Lights
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
64 bytes from 10.1.2.1: icmp_seq=1 ttl=64 time=252 ms
64 bytes from 10.1.2.1: icmp_seq=2 ttl=64 time=123 ms
64 bytes from 10.1.2.1: icmp_seq=3 ttl=64 time=122 ms

--- 10.1.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 121.736/165.426/251.820/61.091 ms
```



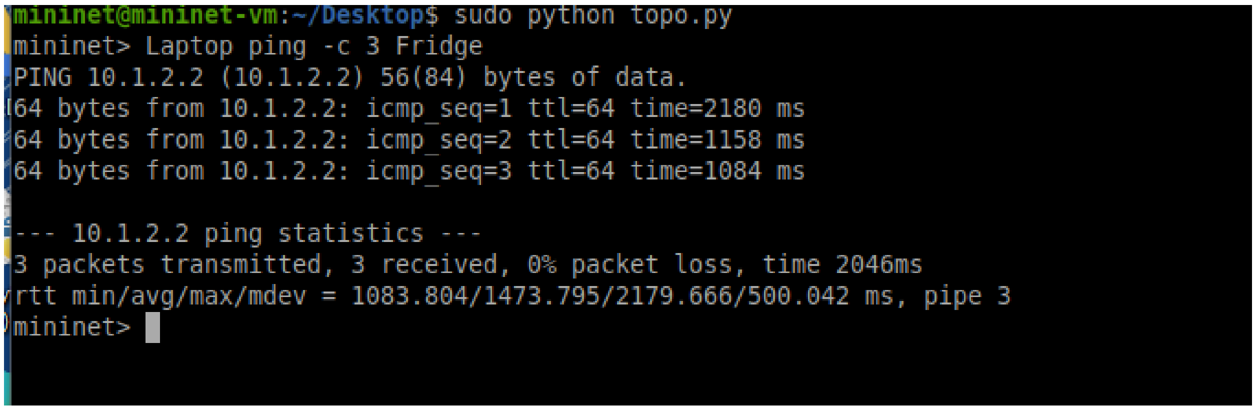
```
mininet@mininet-vm: ~/Desktop
mininet@mininet-vm: ~/Desktop 80x24
Switch1 lo:  Switch1-eth1:Desktop-eth0 Switch1-eth2:Laptop-eth0 Switch1-eth3:Swi
tch2-eth3
Switch2 lo:  Switch2-eth1:Lights-eth0 Switch2-eth2:Fridge-eth0 Switch2-eth3:Swit
ch1-eth3
c0
mininet> Desktop ping -c 3 Laptop
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=324 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=161 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=64 time=162 ms

--- 10.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 161.459/215.765/324.286/76.735 ms
mininet> Laptop ping -c 3 Lights
PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data.
64 bytes from 10.1.2.1: icmp_seq=1 ttl=64 time=489 ms
64 bytes from 10.1.2.1: icmp_seq=2 ttl=64 time=242 ms
64 bytes from 10.1.2.1: icmp_seq=3 ttl=64 time=241 ms

--- 10.1.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 240.823/324.104/489.301/116.813 ms
mininet> 
```

b.) Do you notice any changed in End-To-End Delay?Is this what you expected?Why or why not?

- Yes, we see an increase in the End-to-End delay from 163 ms to 324 ms, and 252 ms to 489 ms, this is what I expected as we increased the delay by a factor of 2 we see an increase in the end-to-end doubling in the time it takes.

Q6.)

a.)

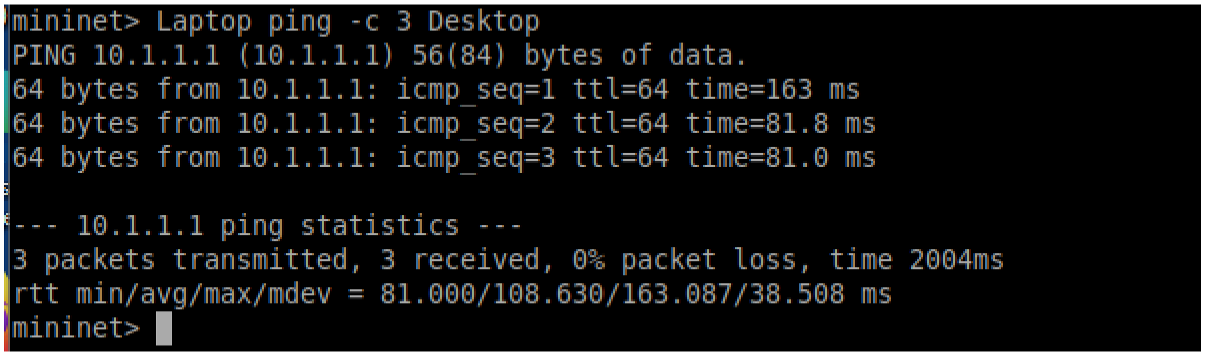


```
mininet@mininet-vm:~/Desktop$ sudo python topo.py
mininet> Laptop ping -c 3 Fridge
PING 10.1.2.2 (10.1.2.2) 56(84) bytes of data.
64 bytes from 10.1.2.2: icmp_seq=1 ttl=64 time=2180 ms
64 bytes from 10.1.2.2: icmp_seq=2 ttl=64 time=1158 ms
64 bytes from 10.1.2.2: icmp_seq=3 ttl=64 time=1084 ms

--- 10.1.2.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 1083.804/1473.795/2179.666/500.042 ms, pipe 3
mininet>
```

a.) What do you observe when you ping from the Laptop to the Fridge

- When we ping from the Laptop to Fridge we see a huge increase in delay due to the increased delay due to the ‘intercontinental connection’ increased transmission delay.

b.)



```
mininet> Laptop ping -c 3 Desktop
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=163 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=81.8 ms
64 bytes from 10.1.1.1: icmp_seq=3 ttl=64 time=81.0 ms

--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 81.000/108.630/163.087/38.508 ms
mininet>
```

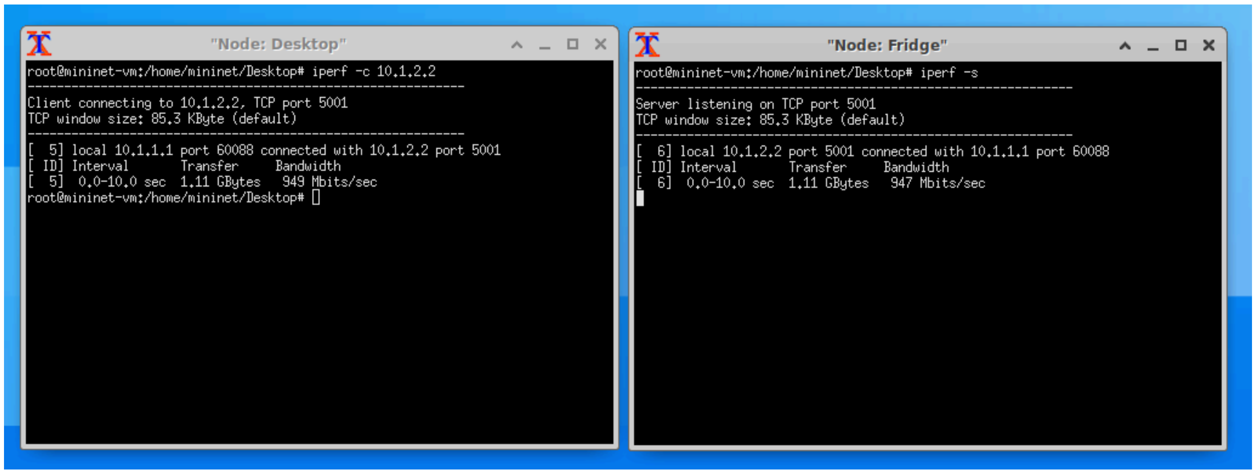
- Are the results expected?
    - Yes these results are expected as the link between Desktop and Laptop has not increased leaving us with the same results as previously

c.) Considering the End-to-End link delay, how does the delay between Switch 1 and Switch 2 impact the ping between Laptop-Fridge compared to Laptop-desktop? If there is a difference, what is the cause of the difference?

- The delay between Switch 1 to Switch 2 impacts the ping between the Laptop and Fridge as the ping has to travel between Switch 1 and Switch 2 increasing the overall delay compared to Laptop-Desktop. As Laptop-Desktop links weren’t increased there was no increase in delay.

Q7.)

a.) Research the iperf command and describe it in your own words.
-iperf is a tool that determines the bandwidth between 2 points in a network utilizing sockets by establishing a server and client to send traffic.
b.)



"Node: Desktop"

```
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.2.2
------------------------------------------------------------
Client connecting to 10.1.2.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 60088 connected with 10.1.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  1.11 GBytes   949 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# 
```

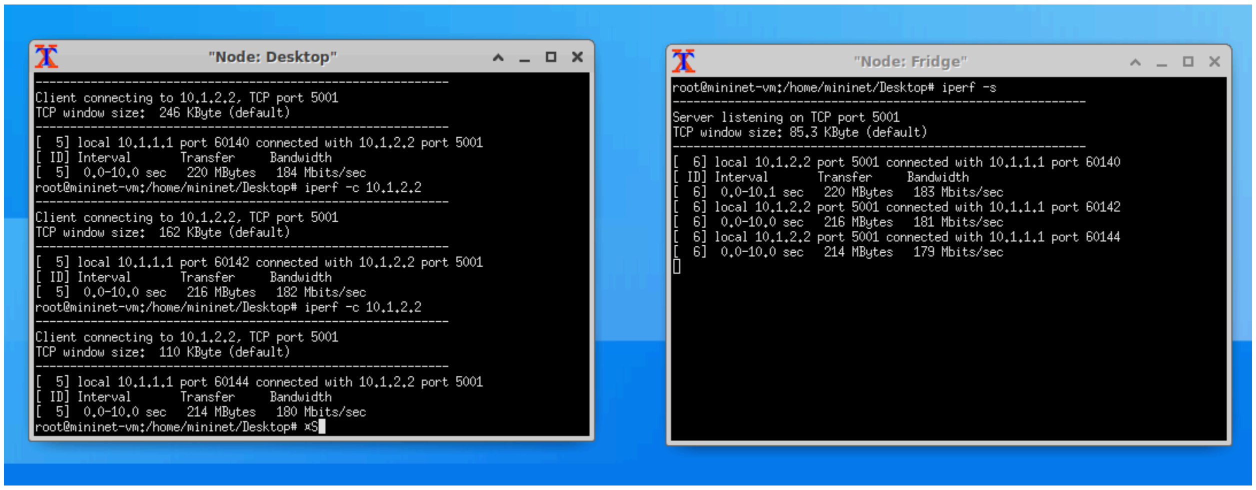
"Node: Fridge"

```
root@mininet-vm:/home/mininet/Desktop# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.1.2.2 port 5001 connected with 10.1.1.1 port 60088
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec  1.11 GBytes   947 Mbits/sec
```

What is the meaning of the rates displayed and are they what you expected?

- The meaning of the output shows in 10 seconds, the iperf protocol sent 1.11 Gigabytes through the link from the Desktop to the Fridge, and the bandwidth it utilized was 949 Mbits/secs or .949 Gigabytes, this is not what I expected as we expected the link to utilize the full 1 Gigabytes of bandwidth

c.)



"Node: Desktop"

```
------------------------------------------------------------
Client connecting to 10.1.2.2, TCP port 5001
TCP window size:  246 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 60140 connected with 10.1.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   220 MBytes   184 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.2.2
------------------------------------------------------------
Client connecting to 10.1.2.2, TCP port 5001
TCP window size:  162 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 60142 connected with 10.1.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   216 MBytes   182 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.2.2
------------------------------------------------------------
Client connecting to 10.1.2.2, TCP port 5001
TCP window size:  110 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 60144 connected with 10.1.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   214 MBytes   180 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# ¤S
```

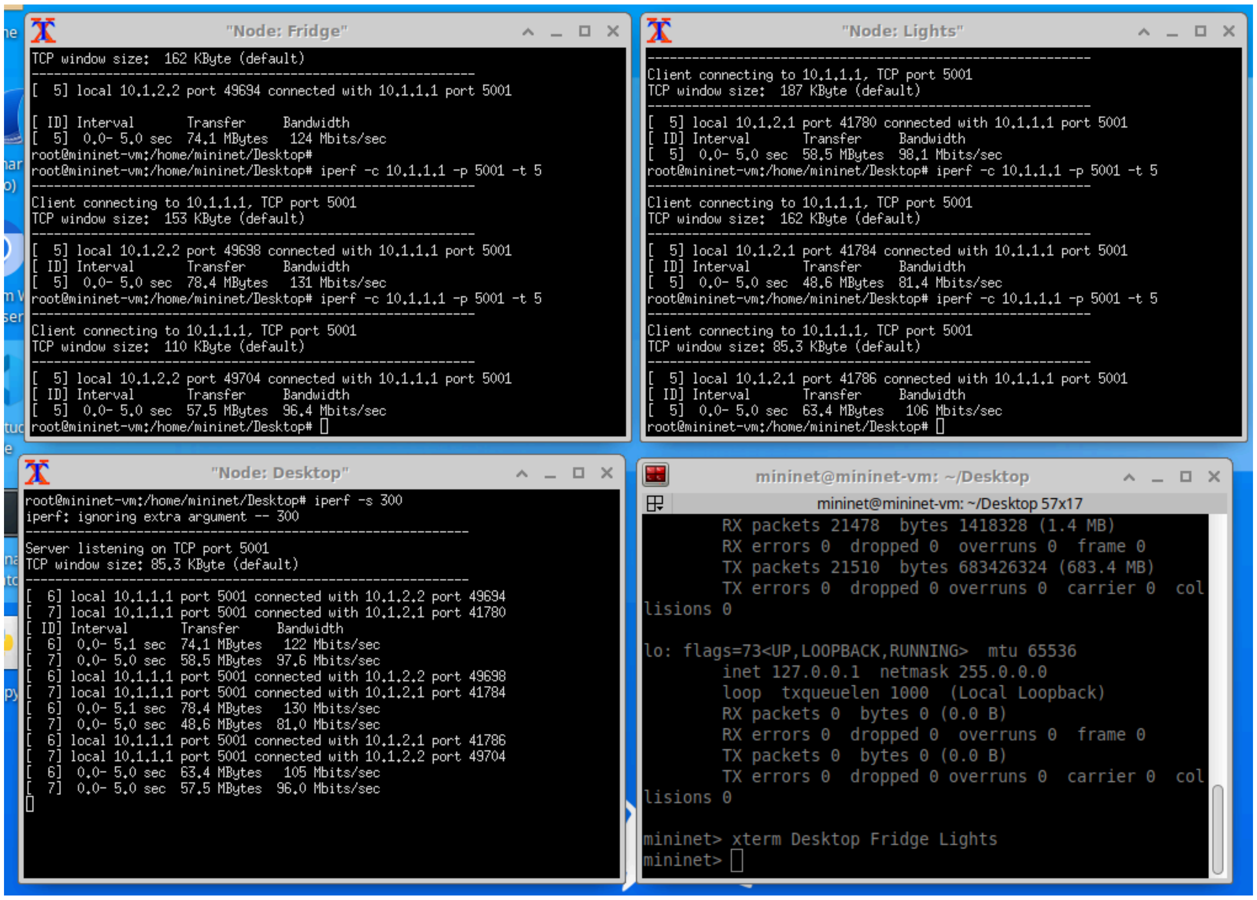
"Node: Fridge"

```
root@mininet-vm:/home/mininet/Desktop# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.1.2.2 port 5001 connected with 10.1.1.1 port 60140
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.1 sec   220 MBytes   183 Mbits/sec
[  6] local 10.1.2.2 port 5001 connected with 10.1.1.1 port 60142
[  6]  0.0-10.0 sec   216 MBytes   181 Mbits/sec
[  6] local 10.1.2.2 port 5001 connected with 10.1.1.1 port 60144
[  6]  0.0-10.0 sec   214 MBytes   179 Mbits/sec
```

Find the average throughput seen by the Fridge?Are your results what you expected?Which links have the greatest influence over the throughput observed? Explain carefully
- The average throughput of the fridge is 21.73 megabytes per second. Yes, the results are as expected; this is due to the average throughput being influenced by the smallest link as it creates a bottleneck. The switch-to-switch bandwidth is 1 gigabit, but the link from the desktop to switch 1 is 200 megabits per second, and the link from switch 2 to the fridge is also 200 megabits per second. Although the switch-to-switch router can transmit at its theoretical

bandwidth, this is not bigger bandwidth isn't transmitted at the refrigerator and desktop links, as they are capped at 200 Mbps

d.)



"Node: Fridge"

```
TCP window size:  162 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.2 port 49694 connected with 10.1.1.1 port 5001

[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  74.1 MBytes   124 Mbits/sec
root@mininet-vm:/home/mininet/Desktop#
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.1.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  153 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.2 port 49698 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  78.4 MBytes   131 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.1.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  110 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.2 port 49704 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  57.5 MBytes  96.4 Mbits/sec
root@mininet-vm:/home/mininet/Desktop#
```

"Node: Lights"

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  187 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.1 port 41780 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  58.5 MBytes  98.1 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.1.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  162 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.1 port 41784 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  48.6 MBytes  81.4 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.1.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.1.2.1 port 41786 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 5.0 sec  63.4 MBytes   106 Mbits/sec
root@mininet-vm:/home/mininet/Desktop#
```

"Node: Desktop"

```
root@mininet-vm:/home/mininet/Desktop# iperf -s 300
iperf: ignoring extra argument -- 300
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.1.1.1 port 5001 connected with 10.1.2.2 port 49694
[  7] local 10.1.1.1 port 5001 connected with 10.1.2.1 port 41780
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0- 5.1 sec  74.1 MBytes   122 Mbits/sec
[  7]  0.0- 5.0 sec  58.5 MBytes  97.6 Mbits/sec
[  6] local 10.1.1.1 port 5001 connected with 10.1.2.2 port 49698
[  7] local 10.1.1.1 port 5001 connected with 10.1.2.1 port 41784
[  6]  0.0- 5.1 sec  78.4 MBytes   130 Mbits/sec
[  7]  0.0- 5.0 sec  48.6 MBytes  81.0 Mbits/sec
[  6] local 10.1.1.1 port 5001 connected with 10.1.2.1 port 41786
[  7] local 10.1.1.1 port 5001 connected with 10.1.2.2 port 49704
[  6]  0.0- 5.0 sec  63.4 MBytes   105 Mbits/sec
[  7]  0.0- 5.0 sec  57.5 MBytes  96.0 Mbits/sec
```

mininet@mininet-vm: ~/Desktop

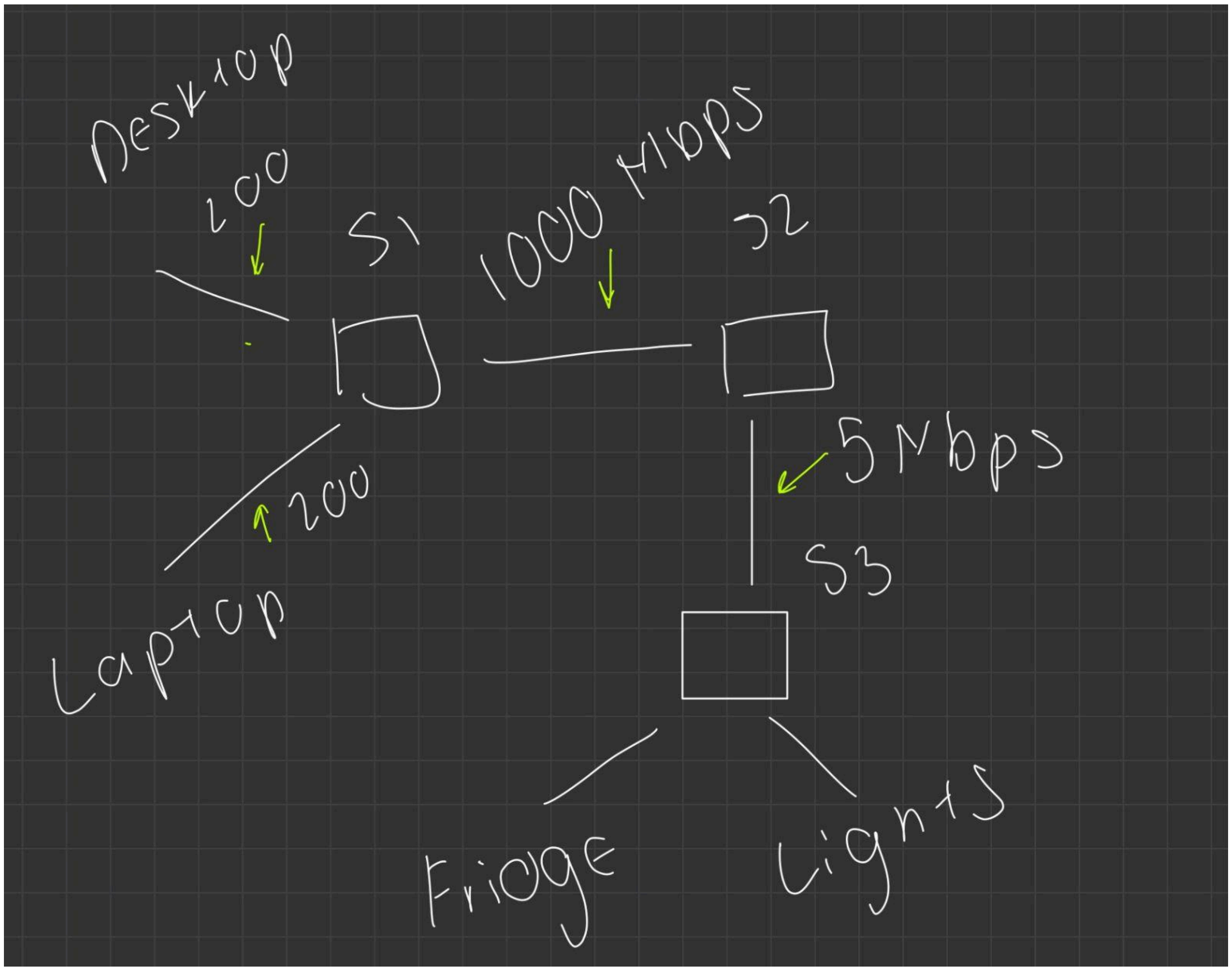
mininet@mininet-vm: ~/Desktop 57x17

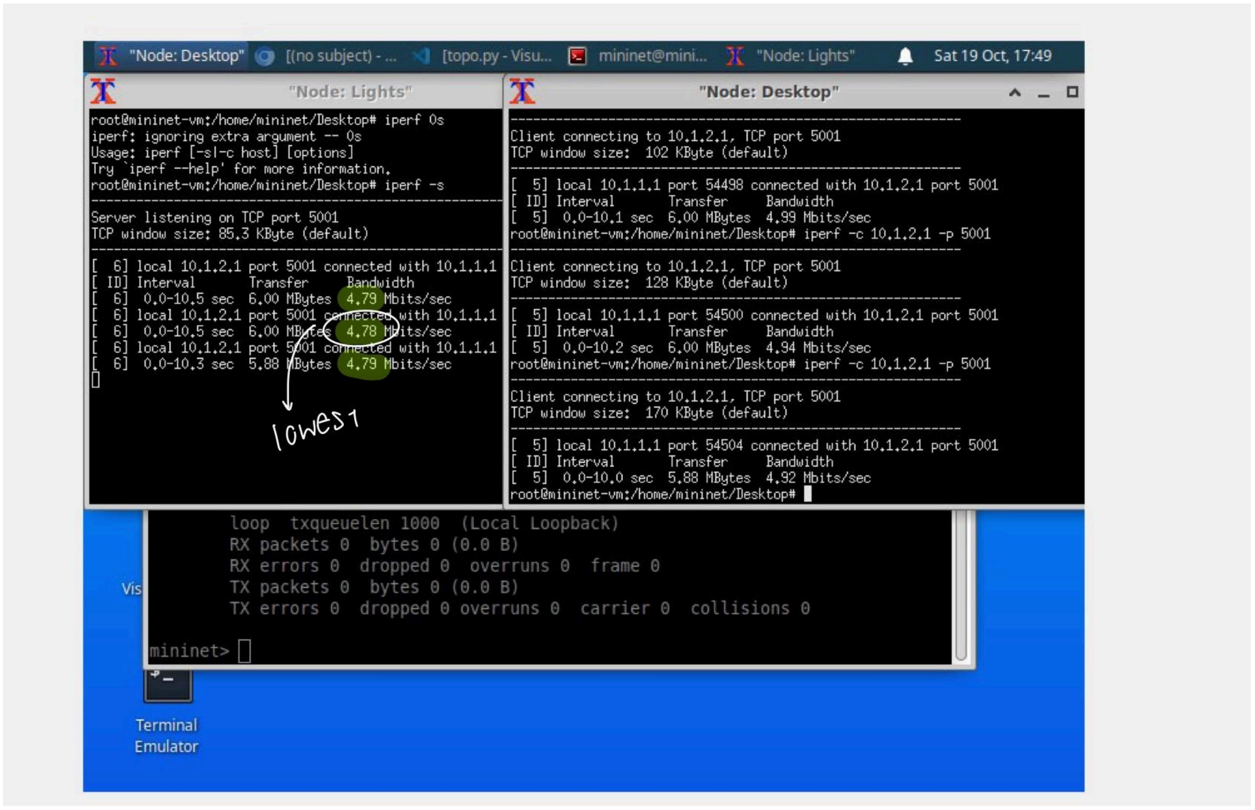
```
        RX packets 21478  bytes 1418328 (1.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21510  bytes 683426324 (683.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  col
lisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  col
lisions 0

mininet> xterm Desktop Fridge Lights
mininet>
```

Which links were shared by the two traffic flows, and what are their bandwidths? Explain your initial expectations in relation to your actual observations.

The link shared by the two traffic flows is the link from Switch 2 to Switch 1 and Switch 1 to Desktop, the bandwidths are 122 Mbit/secs, 97.6 Mbits/sec, 130 Mbits/sec, 81.0 Mbits/sec, 105 Mbits/sec, 96.0 Mbits/sec. My initial expectations would be for the shared link to be split down 50/50 for each to share 100 Mbits/sec, however, the results show that most of the bandwidth is being given to the the traffic that arrives first.

e.)


Desktop
200
S1
1000 Mbps
S2
5 Mbps
S3
200
Laptop
Fridge
Lights



"Node: Lights"

```
root@mininet-vm:/home/mininet/Desktop# iperf 0s
iperf: ignoring extra argument -- 0s
Usage: iperf [-s|-c host] [options]
Try `iperf --help' for more information.
root@mininet-vm:/home/mininet/Desktop# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.1.2.1 port 5001 connected with 10.1.1.1
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.5 sec  6.00 MBytes  4.79 Mbits/sec
[  6] local 10.1.2.1 port 5001 connected with 10.1.1.1
[  6]  0.0-10.5 sec  6.00 MBytes  4.78 Mbits/sec
[  6] local 10.1.2.1 port 5001 connected with 10.1.1.1
[  6]  0.0-10.3 sec  5.88 MBytes  4.79 Mbits/sec
```

"Node: Desktop"

```
------------------------------------------------------------
Client connecting to 10.1.2.1, TCP port 5001
TCP window size:  102 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 54498 connected with 10.1.2.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.1 sec  6.00 MBytes  4.99 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.2.1 -p 5001
------------------------------------------------------------
Client connecting to 10.1.2.1, TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 54500 connected with 10.1.2.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.2 sec  6.00 MBytes  4.94 Mbits/sec
root@mininet-vm:/home/mininet/Desktop# iperf -c 10.1.2.1 -p 5001
------------------------------------------------------------
Client connecting to 10.1.2.1, TCP port 5001
TCP window size:  170 KByte (default)
------------------------------------------------------------
[  5] local 10.1.1.1 port 54504 connected with 10.1.2.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  5.88 MBytes  4.92 Mbits/sec
root@mininet-vm:/home/mininet/Desktop#
```

```
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet>
```

What is the average throughput you observed? Discuss your results and how they have changed. Are your results what you expected? Why or Why not?

- The average throughput observed is 4.78 Mbits/sec. The results show a the throughput for all three traffic calls to Light is at a bandwidth of nearly 4.78, they have changed from previously being able to send from the theoretical bandwidth of 200 Mbits/sec to nearly 5 Mbits/sec, and the results are what I expected as the overall bandwidth of the network is always capped by the smaller link, meaning the 5Mbits would be the overall influence in the bandwidth of the network