Lab 2: HTTP and Throughput

(100 points)

Platform: This entire lab is to be run on the Mininet VM. The lab computers are available for all of these problems.

Submission:

Important Announcement about Gradescope submissions:

- When submissions are made on Gradescope, **the page/region for each problem MUST be marked.**
- We will not grade submissions that are not marked. If you do not know how to do this, ask for help.
- If a submission does not have the region/page marked, you will be asked to resubmit.
- The recorded date for the assignment will be the resubmission date, and late days will be counted.

One PDF file for this assignment is to be submitted directly on Gradescope.
Naming convention of the file: [YourCruzID].pdf.

For all questions that require a screenshot:

- Make sure that a **date timestamp** is visible next to your results.
- No credit will be given for screenshots without a timestamp.

References: Chapters 1 and 2, Computer Networking: A Top Down Approach

- Section 1.4: Delay, Loss and Throughput in Packet-Switched Networks
- Section 2.2: The Web and HTTP
- RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1

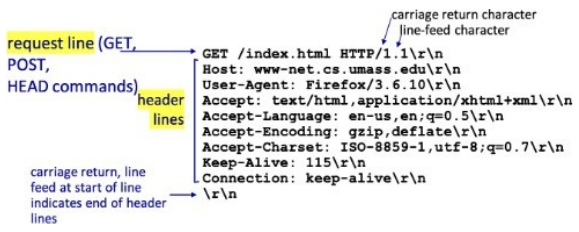
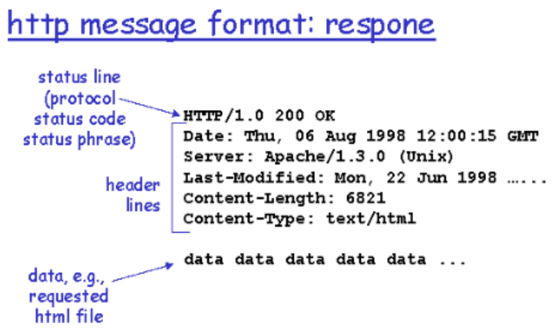
Introduction to HTTP

Before beginning this lab, it is very helpful to read Section 2.2 of the text.[1]

The HTTP protocol is the *lingua franca* of the Web. HTTP specifies methods and headers so that web clients can issue requests for web objects and receive responses from web servers. HTTP methods provide a set of actions to be performed on web resources. You will observe several of these methods in this assignment. HTTP headers are extra helpful information included in an HTTP message sent by either the client or the server. Headers are specified as name:value pairs. You can search the vast number of HTTP headers online.

Having gotten our feet wet with the Wireshark packet sniffer in Lab1, we're now ready to use Wireshark to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic request/response interaction between client and server, HTTP message formats, retrieving HTML files with multiple embedded objects, and Conditional HTTP requests. Additionally, in the last problem, we will take a look at link transmission rates and bottleneck links.

Please review class notes and the textbook to review the format of the messages exchanged. A sample client request and server response are shown below:

HTTP Request Message	HTTP Response Message
	

carriage return character
line-feed character
request line (GET, POST, HEAD commands)
header lines
carriage return, line feed at start of line indicates end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

http message format: respone
status line (protocol status code status phrase)
header lines
data, e.g., requested html file

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

Part A - HTTP: Hypertext Transfer Protocol

1. [35 pts] The Basic HTTP request/response messages

a. **[10 points]** In the following problem, match the HTTP headers to their respective definitions and usages. This exercise will help deepen your understanding of how each header contributes to the HTTP communication process, as detailed in RFC 9110

Table 1: Match the Header (on the left) with its Definition/Usage (on the right) by filling in the "Answer Here" column **(No Need To Memorize Header Definitions)**

	Header	Match ← →	Answer Here	Definition/Usage
A	"Accept-Encoding"		C	Carries credentials to authenticate the requestor.
B	"Accept"		O	Contains data from the server to be stored by the client.
C	"Authorization"		A	Describes the compression formats that can be decoded.
D	"Cache-Control"		B	Details preferred content types for the response.
E	"Connection"		I	Directive to the client to store the cookie.
F	"Content-Encoding"		D	Directs how responses are to be stored by caches.
G	"Content-Length"		P	Information about the client software making the request.
H	"Content-Type"		T	Numeric indication of the outcome of the server's attempt to fulfill the request.
I	"Cookie"		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"		L	Parameters for a persistent connection.
L	"Keep-Alive"		F	Specifies the encoding used on the data.
M	"Last-Modified"		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"		G	The exact byte length of the HTTP body.
P	"User-Agent"		H	The media type of the body of the request/response.
Q	"Request Version"		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"		S	The type of action the client wants the server to take on a resource.

b. **[10 points]** Now let's explore HTTP by downloading a very simple HTML file - one that is very short and contains no embedded objects.

Do the following:

- First, open up the web browser in the Mininet VM.
- Clear the cache in the browser[2] (go to the Chromium browser setting, search Clear browsing data, select everything for all time, and clear data).
- Start up Wireshark. Set the display filter so that only HTTP messages will be displayed. (We're only interested in the HTTP protocol here and don't want to see the clutter of all captured packets).
  Then, start a packet capture using the interface **eth0**.
- Go back to the browser and enter:
  http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html
  Are you familiar with the very important document displayed by your browser?!
- Stop the Wireshark packet capture, ignore the GET /favicon.ico packet.

Answer the questions in the following tables:

Table 2: In Wireshark, use the Client Request Packet to answer the following. All answers are visible in the packet.

Network and Transport Layer (TCP/IP)	
Questions:	**Answer** (Field Detail from Wireshark)
Example: *What is the destination port number?*	*Example Solution (from Wireshark):* *Destination port: http (80)*
What is the Client IP Address?	10.0.2.15
What is the Client port number?	38456
Application Layer (HTTP)	
Request Line (Initial Line of HTTP Request) Questions:	**Answer** (Elements of the Request Line from Wireshark)
What is the Method used for the request?	get
What is the URI requested by the client?	/wireshark-labs/HTTP-wireshark-file3.html
What HTTP version is used by the Client?	1.1
HTTP Headers (Following the Request Line) Questions:	**Answer** (Header Name:Value Pair from Wireshark)
What is the hostname?	Gaia.cs.umass.edu
What is the Client connection management preference?	keep-alive
What is the Client browser and OS information?	Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like

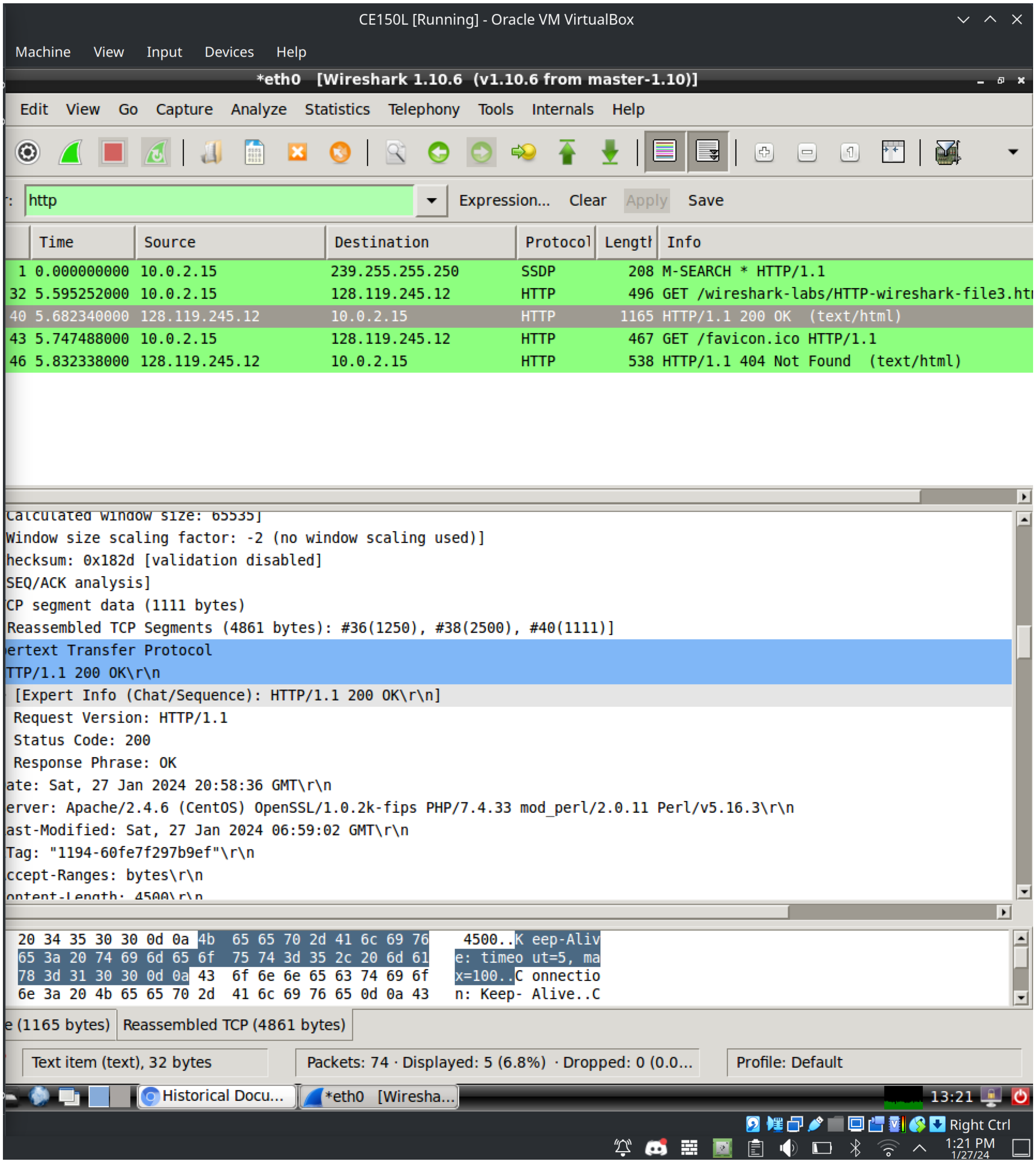
	Gecko) Ubuntu Chromium/63.0.3239.84 Chrome/63.0.3239.84 Safari/537.36
What types of content are accepted by the Client?	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
What compressed data does the Client accept?	gzip, deflate

Include a **<u>timestamped screenshot</u>**, and circle the client IP Address and port number.

c. **[10 points] Table 3:** In Wireshark, use the Server Response Packet to answer the following. All answers are visible in the packet.

Network and Transport Layer (TCP/IP)	
Questions:	**Answer** (Field Detail from Wireshark)
What is the Server IP address?	128.119.245.12
What port number is the Server listening on?	80
Application Layer (HTTP)	
Status Line (Initial Line of HTTP Response) Questions:	**Answer** (Elements of the Status Line from Wireshark)
What HTTP version is used by the Server?	1.1
What is the code that indicates the result of the Client's request?	200
What is the status phrase that accompanies the above code?	OK
HTTP Headers (Following the Status Line) Questions:	**Answer** (Header Name:Value pair from Wireshark)
What is the date and time of the server response?	Sat, 27 Jan 2024 20:58:36 GMT
What is the server software information?	Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3
What is the length of the requested file?	Content length: 4500
What response does the Server provide regarding the management of the network connection?	Keep alive
What type of content is being sent by the Server?	text/html; charset=UTF-8

Include a **<u>timestamped screenshot</u>** and circle the date.


CE150L [Running] - Oracle VM VirtualBox
Machine View Input Devices Help
*eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: http

```
Time         Source          Destination      Protocol Length Info
 1 0.000000000 10.0.2.15       239.255.255.250  SSDP      208 M-SEARCH * HTTP/1.1
32 5.595252000 10.0.2.15       128.119.245.12   HTTP      496 GET /wireshark-labs/HTTP-wireshark-file3.ht
40 5.682340000 128.119.245.12  10.0.2.15        HTTP     1165 HTTP/1.1 200 OK  (text/html)
43 5.747488000 10.0.2.15       128.119.245.12   HTTP      467 GET /favicon.ico HTTP/1.1
46 5.832338000 128.119.245.12  10.0.2.15        HTTP      538 HTTP/1.1 404 Not Found  (text/html)
```

```
Window size scaling factor: -2 (no window scaling used)]
hecksum: 0x182d [validation disabled]
SEQ/ACK analysis]
CP segment data (1111 bytes)
Reassembled TCP Segments (4861 bytes): #36(1250), #38(2500), #40(1111)]
ertext Transfer Protocol
TTP/1.1 200 OK\r\n
 [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
 Request Version: HTTP/1.1
 Status Code: 200
 Response Phrase: OK
ate: Sat, 27 Jan 2024 20:58:36 GMT\r\n
erver: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
ast-Modified: Sat, 27 Jan 2024 06:59:02 GMT\r\n
Tag: "1194-60fe7f297b9ef"\r\n
ccept-Ranges: bytes\r\n
```

```
 20 34 35 30 30 0d 0a 4b  65 65 70 2d 41 6c 69 76    4500..K eep-Aliv
 65 3a 20 74 69 6d 65 6f  75 74 3d 35 2c 20 6d 61   e: timeo ut=5, ma
 78 3d 31 30 30 0d 0a 43  6f 6e 6e 65 63 74 69 6f   x=100..C onnectio
 6e 3a 20 4b 65 65 70 2d  41 6c 69 76 65 0d 0a 43   n: Keep- Alive..C
```

e (1165 bytes) | Reassembled TCP (4861 bytes)

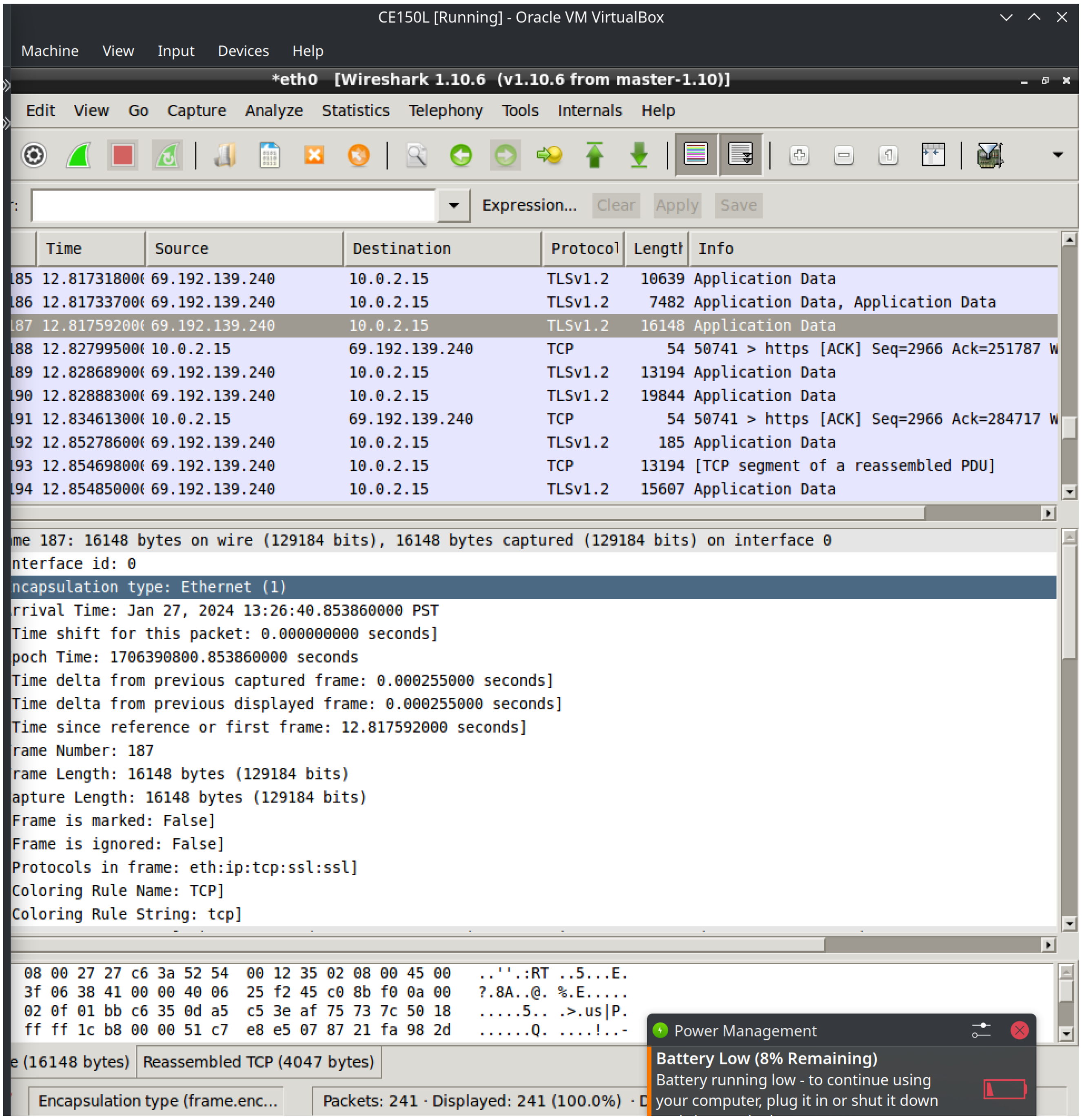
Text item (text), 32 bytes | Packets: 74 · Displayed: 5 (6.8%) · Dropped: 0 (0.0... | Profile: Default

d. **[5 points] Protocol Layers:**
For the following problem, here's a reference to HTTPS:
https://datatracker.ietf.org/doc/html/rfc2818

Now do a new Wireshark packet capture for "https://www.wellsfargo.com/" without using the **http** filter.
In the Wireshark captures of both "https://www.wellsfargo.com/" and "http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html", examine the different layers (e.g., application, transport) of network protocols.

- How do the **application and transport layers** interact **differently** in HTTP and HTTPS? One big thing is that https communicates over port 443 while http operates at port 80, second is https uses http it does not operate it's own protocol rather it just secures http
- If the bank transmitted a password over HTTP, would it be visible in a Wireshark capture? If a password was shared through http you would be able to capture It. How does this visibility differ if transmitted when using HTTPS?
- Https scrambles everything sent through it which makes it very difficult to unscramble the password without knowing the pattern that the local browser and website made.
- List all the protocols that make these two captures **different**. Focus on identifying protocols that contribute to security in the HTTPS capture.
- Https uses tls to secure it, it has http/2

Include a **timestamped screenshot** and circle at least one of the protocols listed.


CE150L [Running] - Oracle VM VirtualBox
Machine View Input Devices Help
*eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

```
   Time         Source          Destination     Protocol Length Info
185 12.81731800( 69.192.139.240  10.0.2.15       TLSv1.2   10639 Application Data
186 12.81733700( 69.192.139.240  10.0.2.15       TLSv1.2    7482 Application Data, Application Data
187 12.81759200( 69.192.139.240  10.0.2.15       TLSv1.2   16148 Application Data
188 12.82799500( 10.0.2.15       69.192.139.240  TCP          54 50741 > https [ACK] Seq=2966 Ack=251787 W
189 12.82868900( 69.192.139.240  10.0.2.15       TLSv1.2   13194 Application Data
190 12.82888300( 69.192.139.240  10.0.2.15       TLSv1.2   19844 Application Data
191 12.83461300( 10.0.2.15       69.192.139.240  TCP          54 50741 > https [ACK] Seq=2966 Ack=284717 W
192 12.85278600( 69.192.139.240  10.0.2.15       TLSv1.2     185 Application Data
193 12.85469800( 69.192.139.240  10.0.2.15       TCP       13194 [TCP segment of a reassembled PDU]
194 12.85485000( 69.192.139.240  10.0.2.15       TLSv1.2   15607 Application Data
```

```
me 187: 16148 bytes on wire (129184 bits), 16148 bytes captured (129184 bits) on interface 0
nterface id: 0
ncapsulation type: Ethernet (1)
rrival Time: Jan 27, 2024 13:26:40.853860000 PST
Time shift for this packet: 0.000000000 seconds]
poch Time: 1706390800.853860000 seconds
Time delta from previous captured frame: 0.000255000 seconds]
Time delta from previous displayed frame: 0.000255000 seconds]
Time since reference or first frame: 12.817592000 seconds]
rame Number: 187
rame Length: 16148 bytes (129184 bits)
apture Length: 16148 bytes (129184 bits)
Frame is marked: False]
Frame is ignored: False]
Protocols in frame: eth:ip:tcp:ssl:ssl]
Coloring Rule Name: TCP]
Coloring Rule String: tcp]
```

```
08 00 27 27 c6 3a 52 54  00 12 35 02 08 00 45 00   ..''.:RT ..5...E.
3f 06 38 41 00 00 40 06  25 f2 45 c0 8b f0 0a 00   ?.8A..@. %.E.....
02 0f 01 bb c6 35 0d a5  c5 3e af 75 73 7c 50 18   .....5.. .>.us|P.
ff ff 1c b8 00 00 51 c7  e8 e5 07 87 21 fa 98 2d   ......Q. ....!..-
```

e (16148 bytes) Reassembled TCP (4047 bytes)

Encapsulation type (frame.enc... Packets: 241 · Displayed: 241 (100.0%) · D

Power Management
Battery Low (8% Remaining)
Battery running low - to continue using your computer, plug it in or shut it down

2. [12 pts] Experiment with the wget command

Start by reading up on the wget command. Determine which command option to use with wget so that the **server returns only the HTTP headers and not the actual requested file.** Use the option to load this URL: www.httpforever.com

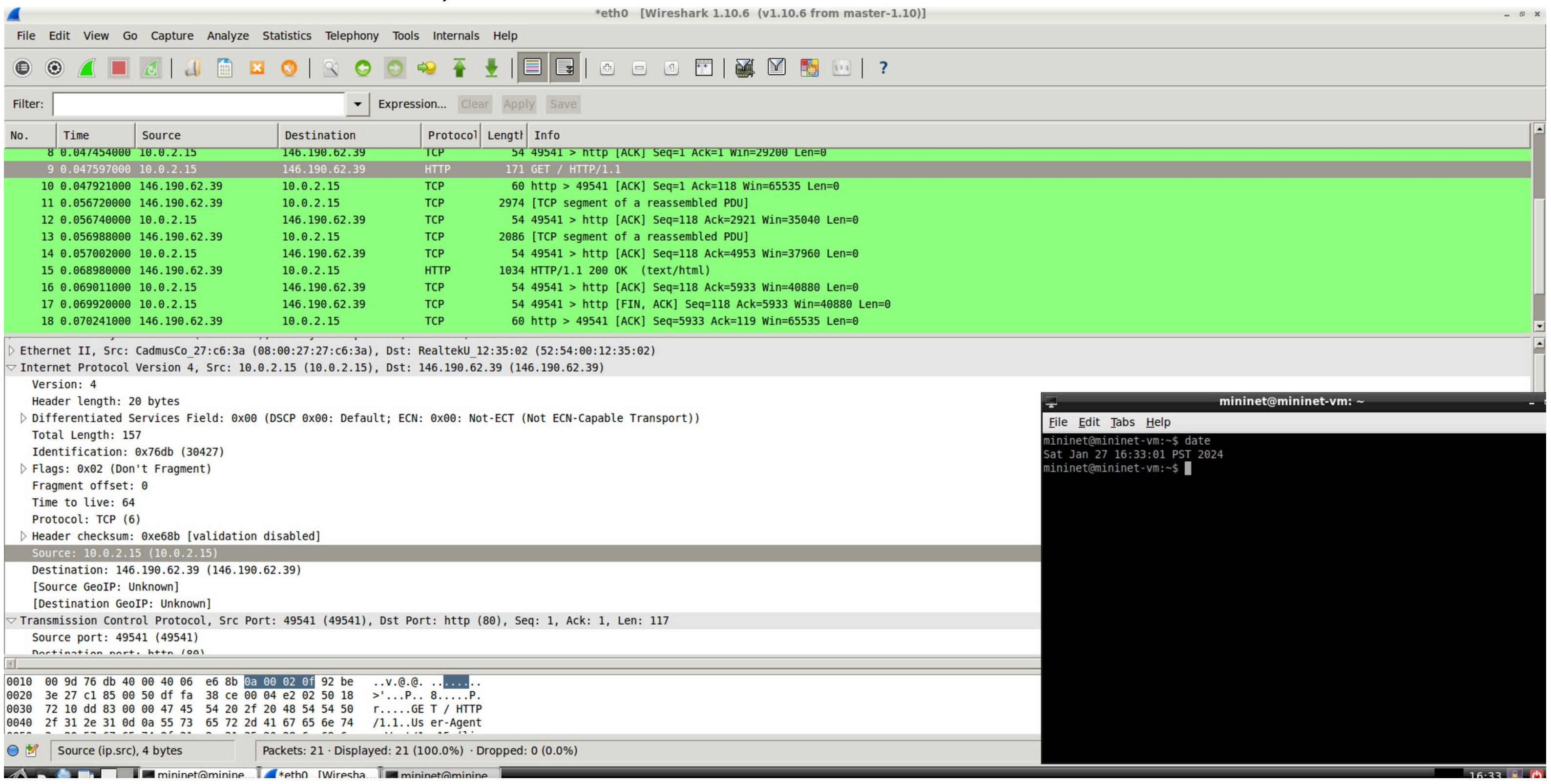
Run Wireshark to capture the packets exchanged during the wget execution.

a. What is the full command you used?
b. wget --server-response www.httpforever.com

Use the wget terminal output to answer the following questions:

c. Server Response
   - Does the server have the requested file? yes How do you know? Next to OK it says text
   - When was the web page last modified?Last-Modified: Wed, 22 Mar 2023 14:54:48 GMT\r\n How is this information used? Most likely if you cached parts of the website it can check if it was edited and if you need to pull all the info again
   - What type of *HTTP connection* is used? Keep ALive Explain what it means. It uses one tcp connection to send and receive info
   - What is the size (in bytes) of the base HTML page?56

Include a **single timestamped screenshot** and circle each question above (there should be 4 items circled).


*eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
mininet@mininet-vm: ~
File Edit Tabs Help
mininet@minine... *eth0 [Wiresha... mininet@minine... 16:33

```
No.  Time         Source          Destination     Protocol Length Info
   8 0.047454000  10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
   9 0.047597000  10.0.2.15       146.190.62.39   HTTP      171 GET / HTTP/1.1
  10 0.047921000  146.190.62.39   10.0.2.15       TCP        60 http > 49541 [ACK] Seq=1 Ack=118 Win=65535 Len=0
  11 0.056720000  146.190.62.39   10.0.2.15       TCP      2974 [TCP segment of a reassembled PDU]
  12 0.056740000  10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=2921 Win=35040 Len=0
  13 0.056988000  146.190.62.39   10.0.2.15       TCP      2086 [TCP segment of a reassembled PDU]
  14 0.057002000  10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=4953 Win=37960 Len=0
  15 0.068980000  146.190.62.39   10.0.2.15       HTTP     1034 HTTP/1.1 200 OK  (text/html)
  16 0.069011000  10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=5933 Win=40880 Len=0
  17 0.069920000  10.0.2.15       146.190.62.39   TCP        54 49541 > http [FIN, ACK] Seq=118 Ack=5933 Win=40880 Len=0
  18 0.070241000  146.190.62.39   10.0.2.15       TCP        60 http > 49541 [ACK] Seq=5933 Ack=119 Win=65535 Len=0

▷ Ethernet II, Src: CadmusCo_27:c6:3a (08:00:27:27:c6:3a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▽ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 146.190.62.39 (146.190.62.39)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 157
    Identification: 0x76db (30427)
  ▷ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▷ Header checksum: 0xe68b [validation disabled]
    Source: 10.0.2.15 (10.0.2.15)
    Destination: 146.190.62.39 (146.190.62.39)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ Transmission Control Protocol, Src Port: 49541 (49541), Dst Port: http (80), Seq: 1, Ack: 1, Len: 117
    Source port: 49541 (49541)

0010  00 9d 76 db 40 00 40 06  e6 8b 0a 00 02 0f 92 be   ..v.@.@. ........
0020  3e 27 c1 85 00 50 df fa  38 ce 00 04 e2 02 50 18   >'...P.. 8.....P.
0030  72 10 dd 83 00 00 47 45  54 20 2f 20 48 54 54 50   r.....GE T / HTTP
0040  2f 31 2e 31 0d 0a 55 73  65 72 2d 41 67 65 6e 74   /1.1..Us er-Agent

Source (ip.src), 4 bytes    Packets: 21 · Displayed: 21 (100.0%) · Dropped: 0 (0.0%)
```

```
mininet@mininet-vm:~$ date
Sat Jan 27 16:33:01 PST 2024
mininet@mininet-vm:~$
```

Now look at your Wireshark capture:

Apply a filter in Wireshark and find the HTTP packet from your client to the web server.

Right-click on that packet and select **"Follow TCP Stream".** You should be able to see all the packets transferred on that TCP connection.

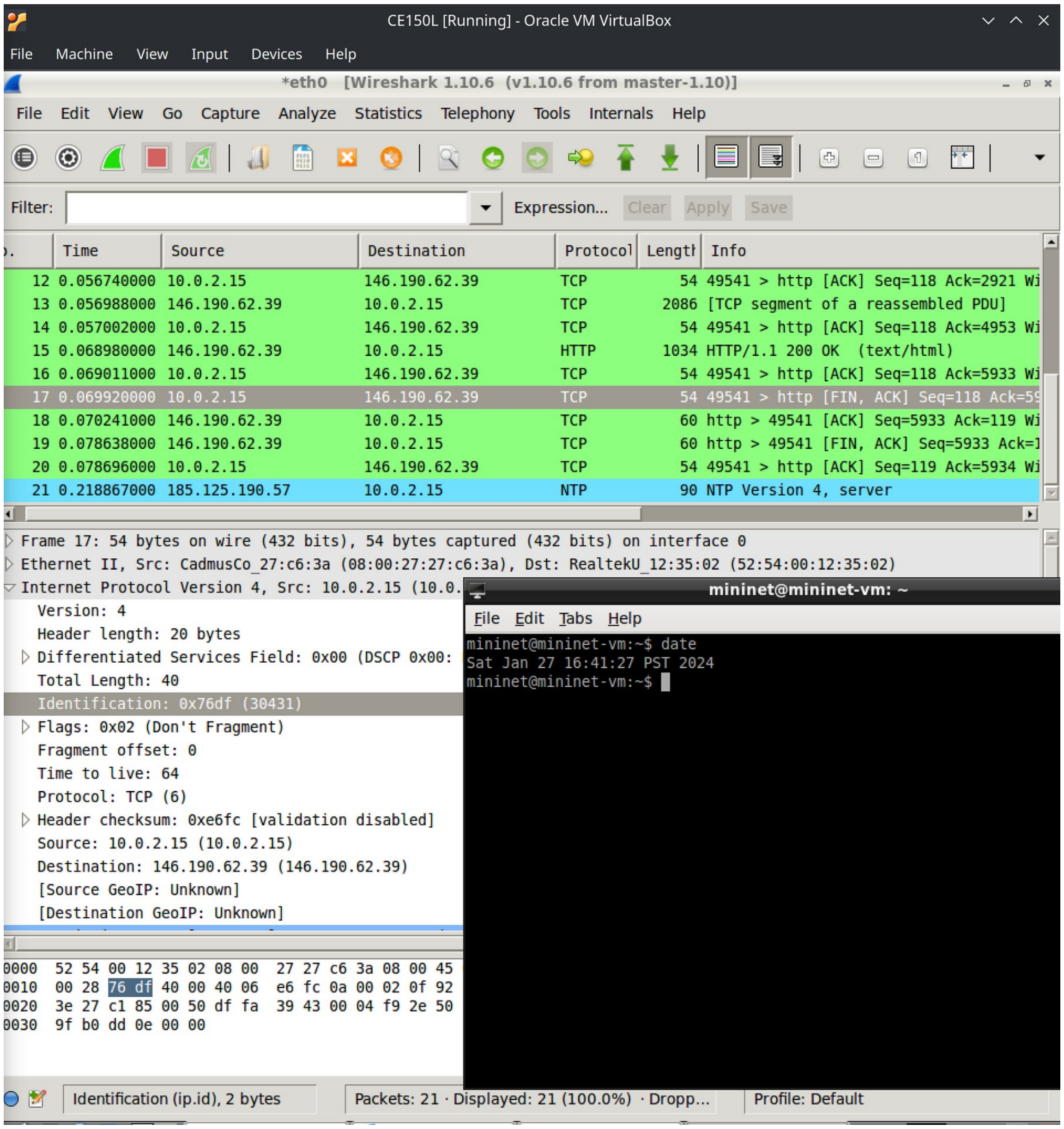
d. Client Request:
   - According to Wireshark, which HTTP method is used for the client request? G E T Is this what you expected? Y e s b e c a u s e I m o n l y r e q u e s t i n g h e a d e r s Why or why not?
   - Explain the HTTP method you observe.
   - Get, It request the info from the server

e. Server:
   - What is the status code of the response message from the server? What does it mean?200 it means OK
   - On what port number is the server listening for HTTP requests? Explain why this port is being used.80 becase it is an http request
   - Which Transport layer protocol is used for these requests?
   - tcp

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

g. The TCP stream allows a two way communication between client and server that can allow data and images to travel safely between the two

Include a **timestamped screenshot** of the Follow TCP Stream with circles around fields that were used in the above questions/answers.


CE150L [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
*eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
 .  Time        Source          Destination     Protocol Length Info
12 0.056740000 10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=2921 Wi
13 0.056988000 146.190.62.39   10.0.2.15       TCP      2086 [TCP segment of a reassembled PDU]
14 0.057002000 10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=4953 Wi
15 0.068980000 146.190.62.39   10.0.2.15       HTTP     1034 HTTP/1.1 200 OK  (text/html)
16 0.069011000 10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=118 Ack=5933 Wi
17 0.069920000 10.0.2.15       146.190.62.39   TCP        54 49541 > http [FIN, ACK] Seq=118 Ack=59
18 0.070241000 146.190.62.39   10.0.2.15       TCP        60 http > 49541 [ACK] Seq=5933 Ack=119 Wi
19 0.078638000 146.190.62.39   10.0.2.15       TCP        60 http > 49541 [FIN, ACK] Seq=5933 Ack=1
20 0.078696000 10.0.2.15       146.190.62.39   TCP        54 49541 > http [ACK] Seq=119 Ack=5934 Wi
21 0.218867000 185.125.190.57  10.0.2.15       NTP        90 NTP Version 4, server
```

```
▷ Frame 17: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_27:c6:3a (08:00:27:27:c6:3a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▽ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00:
    Total Length: 40
    Identification: 0x76df (30431)
  ▷ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▷ Header checksum: 0xe6fc [validation disabled]
    Source: 10.0.2.15 (10.0.2.15)
    Destination: 146.190.62.39 (146.190.62.39)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
```

```
0000  52 54 00 12 35 02 08 00  27 27 c6 3a 08 00 45
0010  00 28 76 df 40 00 40 06  e6 fc 0a 00 02 0f 92
0020  3e 27 c1 85 00 50 df fa  39 43 00 04 f9 2e 50
0030  9f b0 dd 0e 00 00
```

```
mininet@mininet-vm:~$ date
Sat Jan 27 16:41:27 PST 2024
mininet@mininet-vm:~$ 
```

Identification (ip.id), 2 bytes · Packets: 21 · Displayed: 21 (100.0%) · Dropp... · Profile: Default

3. [16 pts] Looking at Embedded Objects

Let’s look at what happens when your browser downloads a file with references to several embedded objects, i.e., a file that includes references to other objects, accessible by hyperlink, that are stored on a server(s).

Do the following:

- Enter http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html in your browser to display a short HTML file with two embedded images.
- Start a Wireshark capture.
- These two images are referenced in the base HTML file. The images themselves are not contained in the HTML file; instead, the URLs for the images are referenced using HTML. As discussed in the textbook (and class), your browser retrieves these objects from the indicated websites.
- Stop the Wireshark packet capture and filter for **http** packets.
- ***Ignore the /favicon.ico request/response in Wireshark***.

a. [10 points] Table 4:

Find the response packet associated with the **first** GET request in Wireshark.	
Questions:	**Answers:**
Describe the purpose of the very first GET request and the corresponding response message.	The Purpose of the first get and response was to get the html text to load the page, that being said images do not get sent in this
Was it successful? Explain how you know.	Yes, because if you go to the html data in the captured package you can see the entire web page data
If successful, how many bytes of the requested file does the Server send?	942
Copy the lines from the base **HTML** page that reference the two images (Locate the raw data in this packet and look for the <img> tags).	Reference 1:<p align="left"><img src="http://kurose.cslash.net/8E_cover_small.jpg"\n Reference 2:<img src="http://gaia.cs.umass.edu/pearson.png" WIDTH="140" HEIGHT="82" > </p>\n
Enter each URL separately into your browser, what do you see?	URL 1: image of the book URL 2: the pearson logo
From the Wireshark trace, how many HTTP GET request messages did your browser send in total? ***(Ignore /favicon.ico in Wireshark***)	3 in totoal

<u>**Embedded images:**</u> Using Wireshark, identify the two HTTP client requests for the embedded images and their corresponding server responses.	
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 8e cover small jpg is hosted on a different site as you can see that the destination address if differentt
Which image is hosted on a **different** server?	The 8e cover small jpg is hosted on a different site

b. **[6 points]** Draw a timing diagram to show the sequence of interactions between the client and server when loading "http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html".
Your diagram should include:

- The client (your web browser), one server hosting the HTML file, and the other server that hosts one of the embedded images.
- The initial GET request for the HTML file and the server's response.
- Subsequent GET requests for each embedded image and the server response.
- Use arrows to indicate the direction of each request and response.
- There should be a clear order of interactions, labeling each step to show the progression from the initial request to the complete rendering of the web page with all embedded images.

4. [7 pts] HTTP Conditional GET

Recall from Section 2.2.5 that most web browsers perform object caching and thus often perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty[2].

Now do the following (follow instructions closely):

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up Wireshark and listen on the **eth0** interface.

1st HTTP GET:

Enter this 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.

2nd HTTP GET:

Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)

- Stop the Wireshark packet capture and filter to display **HTTP** packets.

***(Ignore the /favicon.ico request/response in Wireshark*.)

Questions:

a. Why might your browser issue a Conditional GET?
b. If it is not sure if it has updated since it cached the data,

1st HTTP GET and server response:

c. Is the first HTTP GET request from your browser to the server a Conditional GET? Explain your answer referencing Wireshark. No because its the first time it is caching the data as we cleared it earlier (there is no previous get request). However the second should be.

2nd HTTP GET and server response:

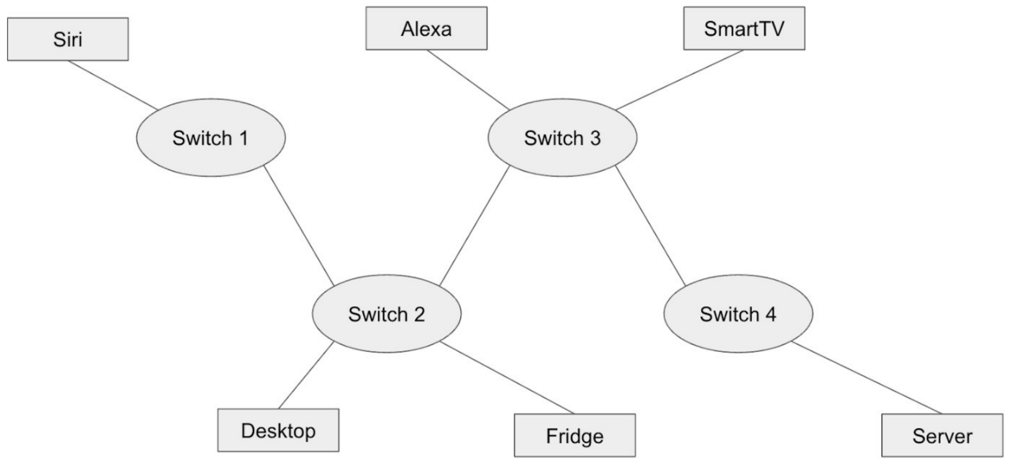
d. Now inspect the contents of the second HTTP GET request from your browser to the server.
   - Describe how the second HTTP GET message is different from the first one.
   - It responds with a 304 message meaning nothin has been modified
   - What is the "IF-MODIFIED-SINCE:" header and its value, and why is it used?
   - It compares your cached data with its the If-modified-since data and makes the decision on whether to use cached or get new data
   - What is the date and its value, what does it mean? It means the date of the get request

Part B - Measuring Performance in Mininet

In this section, we continue our study of network performance and focus on throughput. We make a distinction between transmission rate (often observed at the sender) and throughput. Throughput is the rate data is delivered to the receiver and is specifically observed at the Application Layer.

Note: Remember that rates are measured in bits per second and when the numbers are too large we use notation such as Gpbs (gigabits per second) and Mbps (megabits per second). Typically these rates are stated in bits (not bytes). G = 10^9 and M = 10^6

Figure1:



5. [30 pts] Exploring Transmission Rate

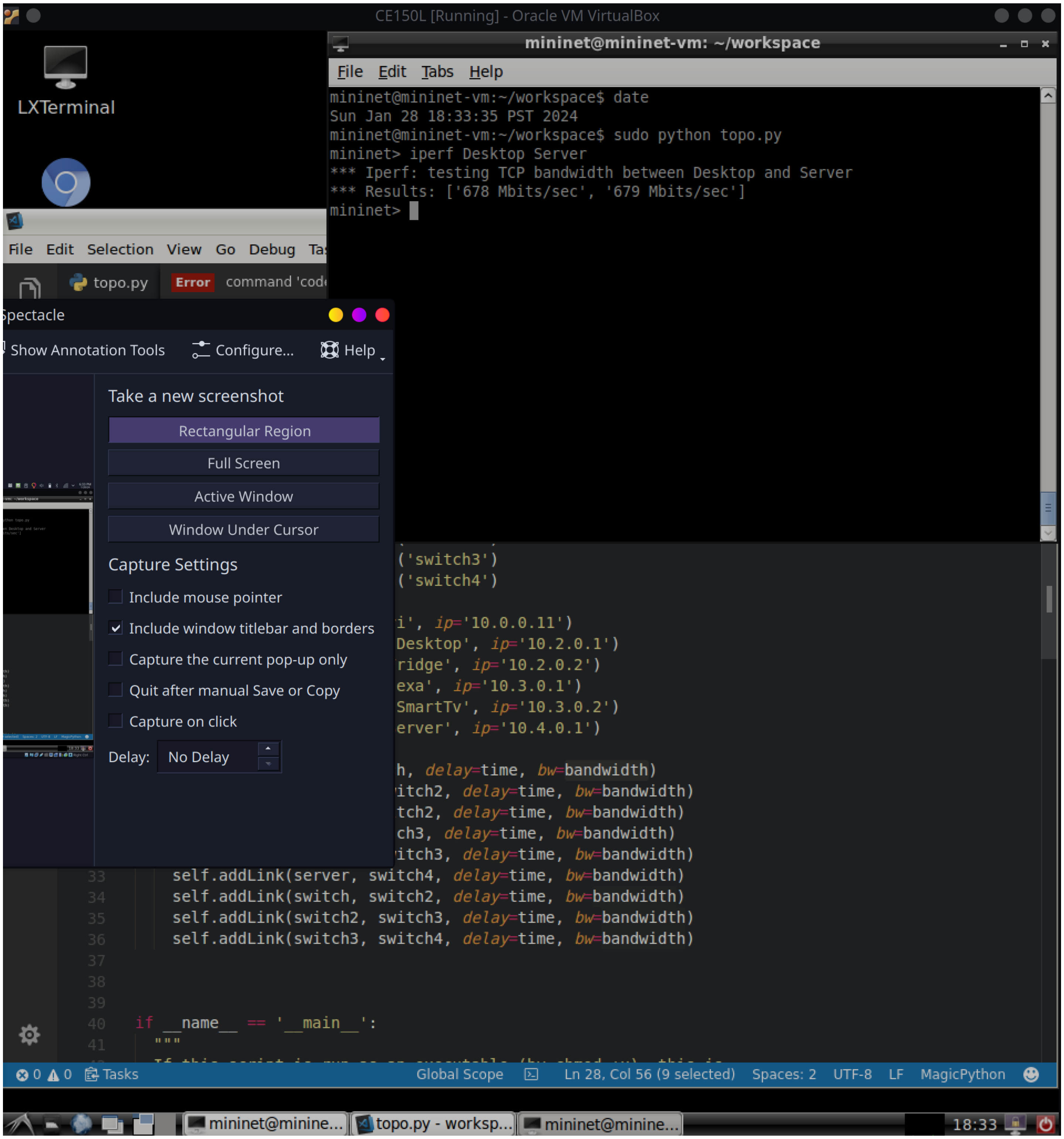
a. Research the **iperf** tool and describe it in your own words.

Using **iperf** on your <u>**topology from Lab1 (Figure 1)**</u>:

b. <u>Restricted bandwidth:</u> The bandwidth of the links can be altered by adding a `bw=` parameter to addLink (which takes its argument as a number representing the bandwidth in Mbps). Set the bandwidth of all links to **1 Gbps**.

   Run the `iperf` command between the Desktop and Server in the Mininet command line and screenshot the output. Explain the results. What is the meaning of the rates displayed and are they what you expected?

Include a **<u>timestamped screenshot</u>** of the results.



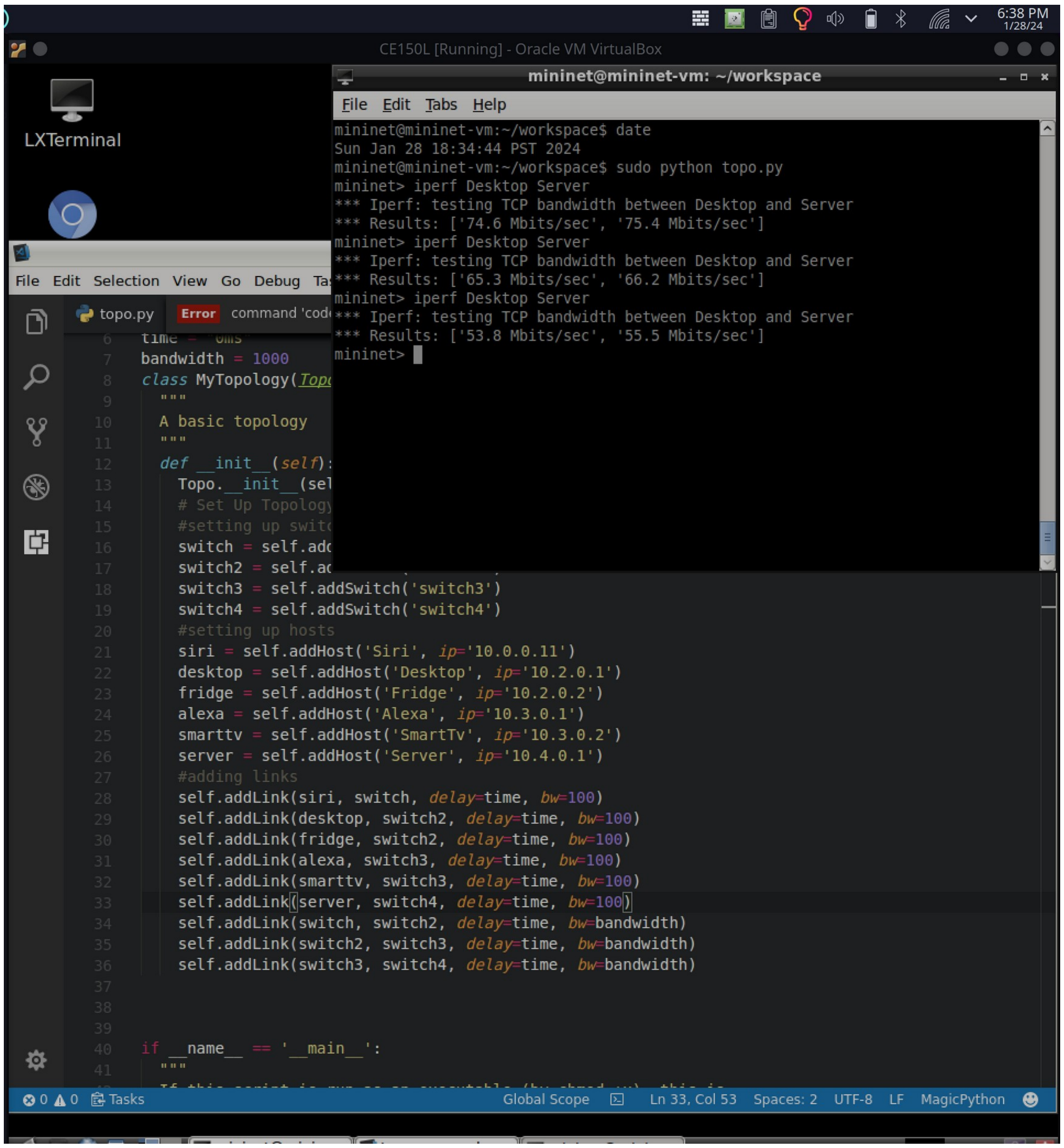
```
mininet@mininet-vm:~/workspace$ date
Sun Jan 28 18:33:35 PST 2024
mininet@mininet-vm:~/workspace$ sudo python topo.py
mininet> iperf Desktop Server
*** Iperf: testing TCP bandwidth between Desktop and Server
*** Results: ['678 Mbits/sec', '679 Mbits/sec']
mininet> 
```

```
('switch3')
('switch4')

i', ip='10.0.0.11')
Desktop', ip='10.2.0.1')
ridge', ip='10.2.0.2')
exa', ip='10.3.0.1')
SmartTv', ip='10.3.0.2')
erver', ip='10.4.0.1')

h, delay=time, bw=bandwidth)
itch2, delay=time, bw=bandwidth)
tch2, delay=time, bw=bandwidth)
ch3, delay=time, bw=bandwidth)
itch3, delay=time, bw=bandwidth)
        self.addLink(server, switch4, delay=time, bw=bandwidth)
        self.addLink(switch, switch2, delay=time, bw=bandwidth)
        self.addLink(switch2, switch3, delay=time, bw=bandwidth)
        self.addLink(switch3, switch4, delay=time, bw=bandwidth)



    if __name__ == '__main__':
        """
```

CE150L [Running] - Oracle VM VirtualBox
mininet@mininet-vm: ~/workspace
LXTerminal
File Edit Selection View Go Debug
topo.py
Error command 'cod
Spectacle
Show Annotation Tools
Configure...
Help
Take a new screenshot
Rectangular Region
Full Screen
Active Window
Window Under Cursor
Capture Settings
Include mouse pointer
Include window titlebar and borders
Capture the current pop-up only
Quit after manual Save or Copy
Capture on click
Delay: No Delay
Global Scope
Ln 28, Col 56 (9 selected) Spaces: 2 UTF-8 LF MagicPython
mininet@minine... topo.py - worksp... mininet@minine...
18:33

c. <u>Changing link bandwidth</u>: Change the bandwidth of edge (host-to-switch) links to **100 Mbps**. Run **iperf** 3 times between the Server and Desktop and find the average throughput seen by the Desktop. Are your results what you expected? Explain.

Average is 65.6, yes I expected this, we limited bandwith between all devices to routers as 100 mbps so there is no way for it to exceed said bandwith.



6:38 PM
1/28/24
CE150L [Running] - Oracle VM VirtualBox
LXTerminal
mininet@mininet-vm: ~/workspace
File Edit Tabs Help
File Edit Selection View Go Debug Ta
topo.py
Error command 'cod
0 0 Tasks
Global Scope Ln 33, Col 53 Spaces: 2 UTF-8 LF MagicPython

```
mininet@mininet-vm:~/workspace$ date
Sun Jan 28 18:34:44 PST 2024
mininet@mininet-vm:~/workspace$ sudo python topo.py
mininet> iperf Desktop Server
*** Iperf: testing TCP bandwidth between Desktop and Server
*** Results: ['74.6 Mbits/sec', '75.4 Mbits/sec']
mininet> iperf Desktop Server
*** Iperf: testing TCP bandwidth between Desktop and Server
*** Results: ['65.3 Mbits/sec', '66.2 Mbits/sec']
mininet> iperf Desktop Server
*** Iperf: testing TCP bandwidth between Desktop and Server
*** Results: ['53.8 Mbits/sec', '55.5 Mbits/sec']
mininet>
```

```
time = '0ms'
bandwidth = 1000
class MyTopology(Topo
  """
  A basic topology
  """
  def __init__(self):
    Topo.__init__(sel
    # Set Up Topology
    #setting up switc
    switch = self.add
    switch2 = self.ad
    switch3 = self.addSwitch('switch3')
    switch4 = self.addSwitch('switch4')
    #setting up hosts
    siri = self.addHost('Siri', ip='10.0.0.11')
    desktop = self.addHost('Desktop', ip='10.2.0.1')
    fridge = self.addHost('Fridge', ip='10.2.0.2')
    alexa = self.addHost('Alexa', ip='10.3.0.1')
    smarttv = self.addHost('SmartTv', ip='10.3.0.2')
    server = self.addHost('Server', ip='10.4.0.1')
    #adding links
    self.addLink(siri, switch, delay=time, bw=100)
    self.addLink(desktop, switch2, delay=time, bw=100)
    self.addLink(fridge, switch2, delay=time, bw=100)
    self.addLink(alexa, switch3, delay=time, bw=100)
    self.addLink(smarttv, switch3, delay=time, bw=100)
    self.addLink(server, switch4, delay=time, bw=100)
    self.addLink(switch, switch2, delay=time, bw=bandwidth)
    self.addLink(switch2, switch3, delay=time, bw=bandwidth)
    self.addLink(switch3, switch4, delay=time, bw=bandwidth)



if __name__ == '__main__':
  """
  If this script is run as an executable (by chmod +x), this is
```

Include a **<u>timestamped screenshot</u>** of the results.

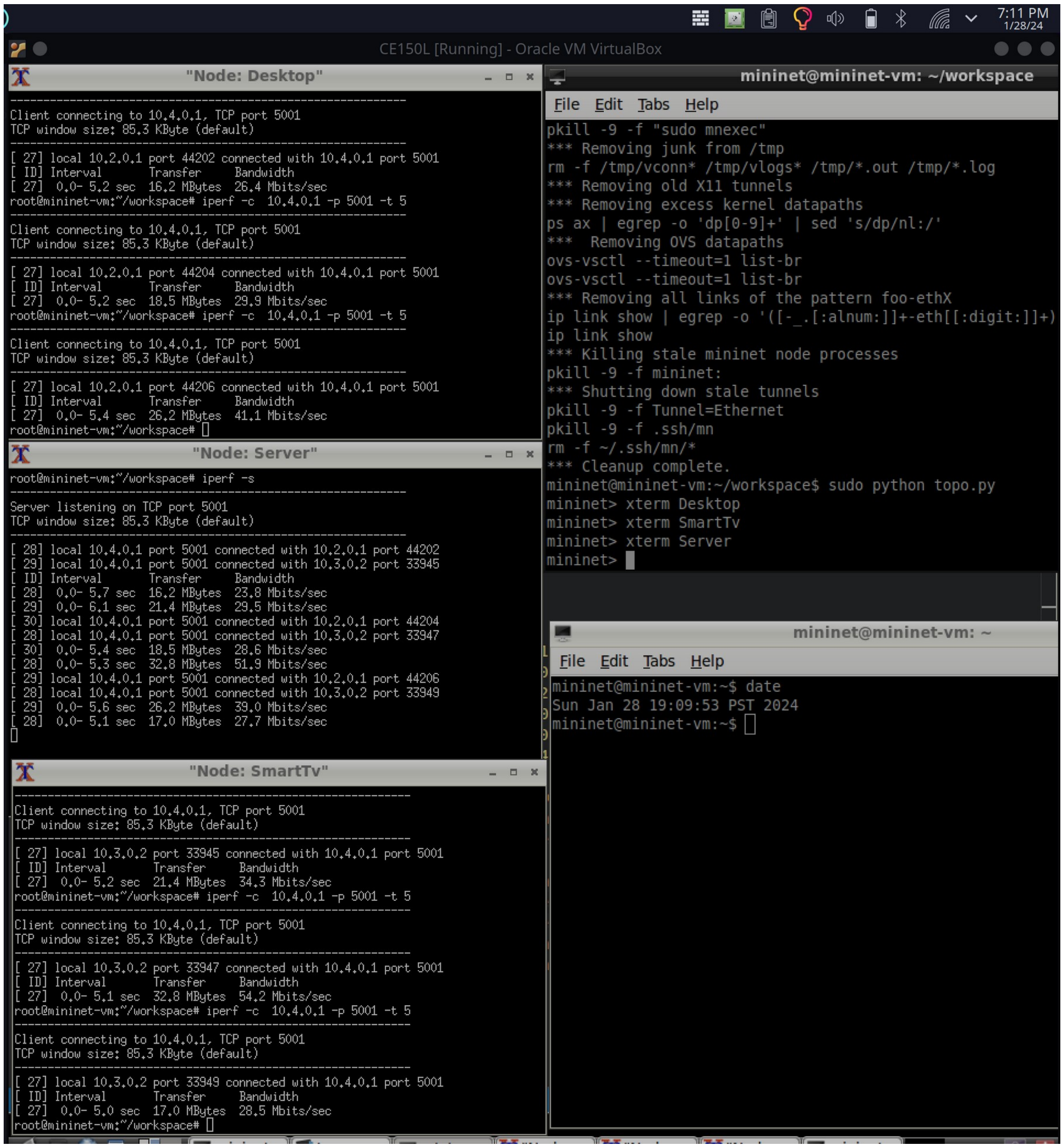
d. <u>Cross-traffic:</u> We will now generate simultaneous traffic between Desktop-Server and SmartTV-Server. Configure "Server" as an iperf server using `iperf -s`. Then generate iperf traffic from Desktop and SmartTv for 5 seconds as clients using:

```
iperf -c <server IP> -p <server port> -t <seconds>
```

Be sure to run both `iperf -c` commands simultaneously. Repeat 3 times and average the results.

Which links were shared by the two traffic flows? Explain what you expected to happen. Discuss your observations after running iperf and if they differ from your initial expectation.

Include a **<u>timestamped screenshot</u>** of the results (this should have 4 terminal windows, 1 mininet, and 3 Xterm - Server, Desktop, SmartTv).



```
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44202 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.2 sec  16.2 MBytes  26.4 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44204 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.2 sec  18.5 MBytes  29.9 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44206 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.4 sec  26.2 MBytes  41.1 Mbits/sec
root@mininet-vm:~/workspace# 
```

```
root@mininet-vm:~/workspace# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 28] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44202
[ 29] local 10.4.0.1 port 5001 connected with 10.3.0.2 port 33945
[ ID] Interval       Transfer     Bandwidth
[ 28]  0.0- 5.7 sec  16.2 MBytes  23.8 Mbits/sec
[ 29]  0.0- 6.1 sec  21.4 MBytes  29.5 Mbits/sec
[ 30] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44204
[ 28] local 10.4.0.1 port 5001 connected with 10.3.0.2 port 33947
[ 30]  0.0- 5.4 sec  18.5 MBytes  28.6 Mbits/sec
[ 28]  0.0- 5.3 sec  32.8 MBytes  51.9 Mbits/sec
[ 29] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44206
[ 28] local 10.4.0.1 port 5001 connected with 10.3.0.2 port 33949
[ 29]  0.0- 5.6 sec  26.2 MBytes  39.0 Mbits/sec
[ 28]  0.0- 5.1 sec  17.0 MBytes  27.7 Mbits/sec
```

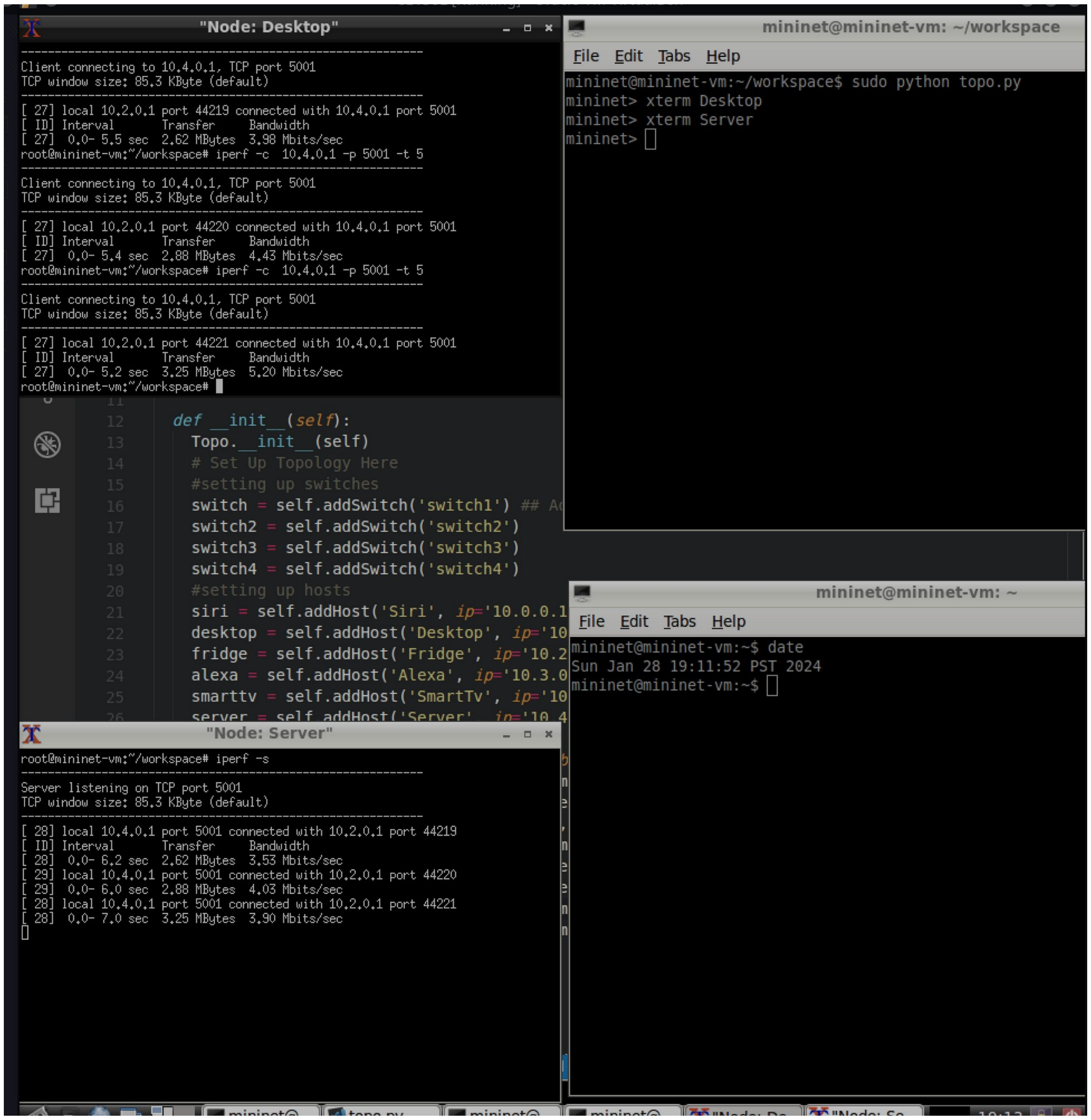
```
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.3.0.2 port 33945 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.2 sec  21.4 MBytes  34.3 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.3.0.2 port 33947 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.1 sec  32.8 MBytes  54.2 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.3.0.2 port 33949 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.0 sec  17.0 MBytes  28.5 Mbits/sec
root@mininet-vm:~/workspace# 
```

```
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet-vm:~/workspace$ sudo python topo.py
mininet> xterm Desktop
mininet> xterm SmartTv
mininet> xterm Server
mininet> 
```

```
mininet@mininet-vm:~$ date
Sun Jan 28 19:09:53 PST 2024
mininet@mininet-vm:~$ 
```

e. <u>The bottleneck link:</u> In this problem, we will observe the network with a bottleneck link. Change the bandwidth between Switch2-Switch3 to **5 Mbps**. Now run **iperf** between Server and Desktop 3 times and take the average of the results.

Discuss your results and how they have changed. Are your results what you expected? Why or why not? Explain.



"Node: Desktop"

```
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44219 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.5 sec  2.62 MBytes  3.98 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44220 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.4 sec  2.88 MBytes  4.43 Mbits/sec
root@mininet-vm:~/workspace# iperf -c  10.4.0.1 -p 5001 -t 5
------------------------------------------------------------
Client connecting to 10.4.0.1, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 27] local 10.2.0.1 port 44221 connected with 10.4.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0- 5.2 sec  3.25 MBytes  5.20 Mbits/sec
root@mininet-vm:~/workspace# 
```

```python
    def __init__(self):
        Topo.__init__(self)
        # Set Up Topology Here
        #setting up switches
        switch = self.addSwitch('switch1') ## Ad
        switch2 = self.addSwitch('switch2')
        switch3 = self.addSwitch('switch3')
        switch4 = self.addSwitch('switch4')
        #setting up hosts
        siri = self.addHost('Siri', ip='10.0.0.1
        desktop = self.addHost('Desktop', ip='10
        fridge = self.addHost('Fridge', ip='10.2
        alexa = self.addHost('Alexa', ip='10.3.0
        smarttv = self.addHost('SmartTv', ip='10
        server = self.addHost('Server', ip='10.4
```

"Node: Server"

```
root@mininet-vm:~/workspace# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 28] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44219
[ ID] Interval       Transfer     Bandwidth
[ 28]  0.0- 6.2 sec  2.62 MBytes  3.53 Mbits/sec
[ 29] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44220
[ 29]  0.0- 6.0 sec  2.88 MBytes  4.03 Mbits/sec
[ 28] local 10.4.0.1 port 5001 connected with 10.2.0.1 port 44221
[ 28]  0.0- 7.0 sec  3.25 MBytes  3.90 Mbits/sec
```

mininet@mininet-vm: ~/workspace

```
mininet@mininet-vm:~/workspace$ sudo python topo.py
mininet> xterm Desktop
mininet> xterm Server
mininet> 
```

mininet@mininet-vm: ~

```
mininet@mininet-vm:~$ date
Sun Jan 28 19:11:52 PST 2024
mininet@mininet-vm:~$ 
```

Average is 4.5mbps and yes this is expected as even though the bandwith between the switch and devices are 100 mbps limiting the switches still slows it down.

Take a **timestamped screenshot** and circle the lowest value for the bandwidth observed.

f. Make a table comparing the output in 5(b), 5(c), 5(d), and 5(e). Discuss your results.

Problem	Bandwidth (s2-s3)	Bandwidth (host-switch)	Source of Data (Desktop/SmartTv)	Destination (Server)	Throughput	Rate at Sender
5(b)	1 Gbps	1 Gbps	Desktop	Server	678	679

5(c)	1 Gbps	100 Mbps	Desktop	Server	64.57	65.7
5(d)	1 Gbps	100 Mbps	Desktop	Server	32.47	30.47
5(d)	1 Gbps	100 Mbps	SmartTv	Server	39	36.37
5(e)	5 Mbps	100 Mbps	Desktop	Server	4.54	3.82

Screenshots Table:

Timestamped Screenshot Check-off: (just for you to keep track of your progress)

Problem #	Screenshot
1b, c, d	
2b, e	
5b, c, d, e	

Footnotes:

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8h ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.*

[2] See https://www.howtogeek.com/304218/how-to-clear-your-history-in-any-browser/ for instructions on clearing your browser cache.

[3] *Hint:* ideally, you should see an If-Modified-Since header since you've just downloaded this page a few seconds ago. However, depending on the browser you're using, and the format of the server's earlier response to your initial GET, your browser may not include an If-Modified-Since even if the document has been downloaded and cached. The Chrome browser is pretty good at regularly using If-Modified-Since. But Safari and Firefox are much more finicky about when to use If-Modified-Since. Life isn't always as easy in practice as it is in theory!