Assignment 1: Raw Sockets

CS 331: Computer Networks

Team 2:
Shubham Agrawal - 22110249
Vraj Shah - 22110292

Supervisor:
Prof. Sameer G. Kulkarni

Indian Institute of Technology Gandhinagar

Saturday 1st February, 2025

Contents

1 Introduction

This report outlines the work for CS 331: Computer Networks Assignment #1. The task required the implementation of a packet sniffer and the examination of packet capture (PCAP) files. The subsequent sections detail the computed metrics, responses to specific PCAP queries, and the process of real-time packet capturing.

2 Packet Sniffer Implementation

The packet sniffer was implemented in Python using socket programming. It was designed to capture packets from the specified network interface and perform real-time analysis of the network packets.

3 Extension of Packet Sniffer Implementation

The same python program was extended, which utilized the original sniffer to capture packets and process them to generate various required matrices as mentioned in part 1 of the Assignment and then subsequently answers the pcap related queries as required in part 2 of the Assignment.

4 Part 1: Metrics and Plots

4.1 Total Data and Packet Statistics

The program analyzed the replayed `2.pcap` file [2 mod 9 = 2] to get the following inferences:

- **Total Data Transferred:** `364642055 bytes`
- **Total Packets Transferred:** `805997 packets`
- **Minimum Packet Size:** `42 bytes`
- **Maximum Packet Size:** `1514 bytes`
- **Average Packet Size:** `452.4112 bytes`

The following figure shows a histogram showing the distribution of packet sizes:

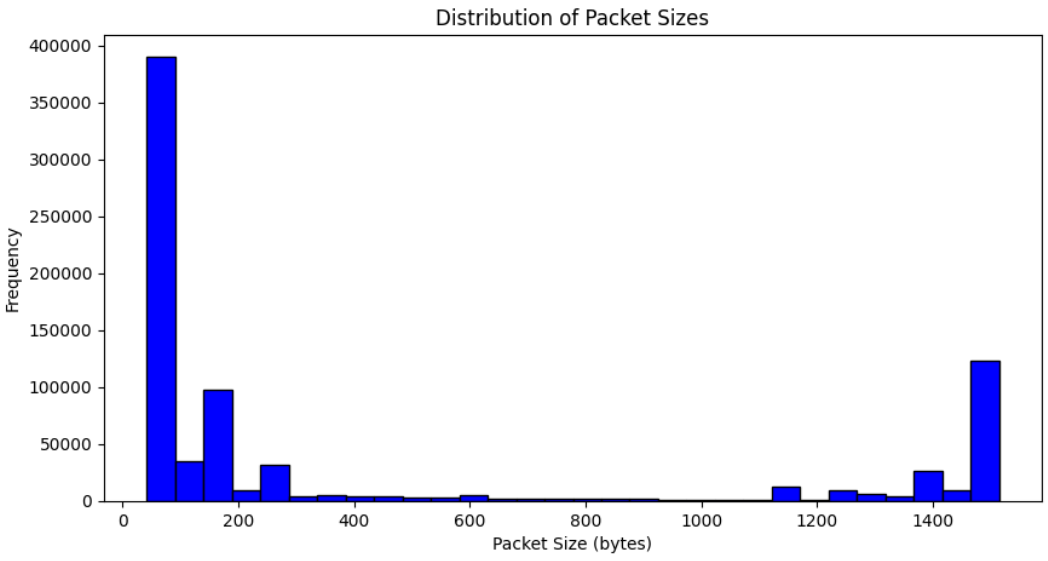
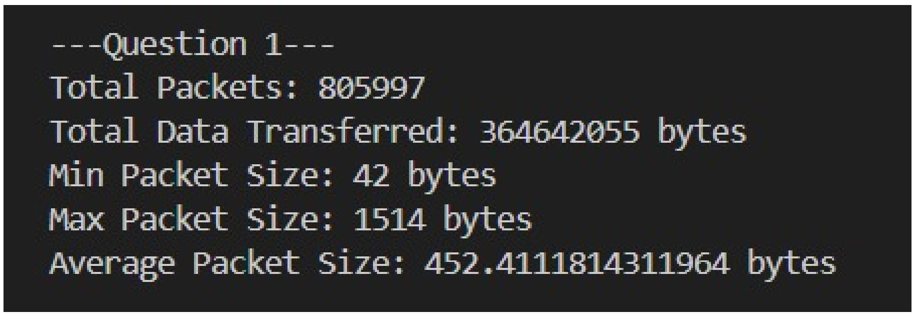


Figure 1: Histogram of packet size vs frequency

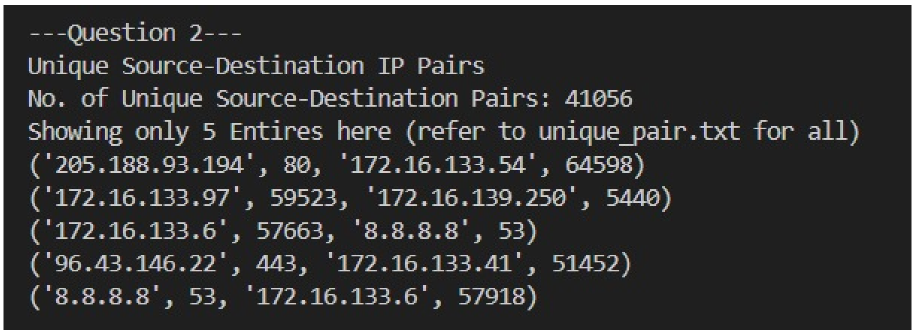


```
---Question 1---
Total Packets: 805997
Total Data Transferred: 364642055 bytes
Min Packet Size: 42 bytes
Max Packet Size: 1514 bytes
Average Packet Size: 452.4111814311964 bytes
```

Figure 2: Result: Part-1 Q1

4.2 Unique Source-Destination Pairs

The unique source-destination IP pairs were identified and stored in a text file named `unique_pairs.txt`, included in the GitHub repository. Only IP packets having TCP or UDP layers were considered. The stored contents are in the form `source-ip:port` $\rightarrow$ `destination-ip:port`.



```
---Question 2---
Unique Source-Destination IP Pairs
No. of Unique Source-Destination Pairs: 41056
Showing only 5 Entires here (refer to unique_pair.txt for all)
('205.188.93.194', 80, '172.16.133.54', 64598)
('172.16.133.97', 59523, '172.16.139.250', 5440)
('172.16.133.6', 57663, '8.8.8.8', 53)
('96.43.146.22', 443, '172.16.133.41', 51452)
('8.8.8.8', 53, '172.16.133.6', 57918)
```

Figure 3: Result: Part-1 Q2

4.3 Source IP and total flow

Unique source IP's and their corresponding total flows were stored in a text file named `unique_src_ip.txt`, included in the GitHub repository. Only IP packets were considered. The stored contents are in the form `source-ip : total flow count`.

4.4 Destination IP and total flow

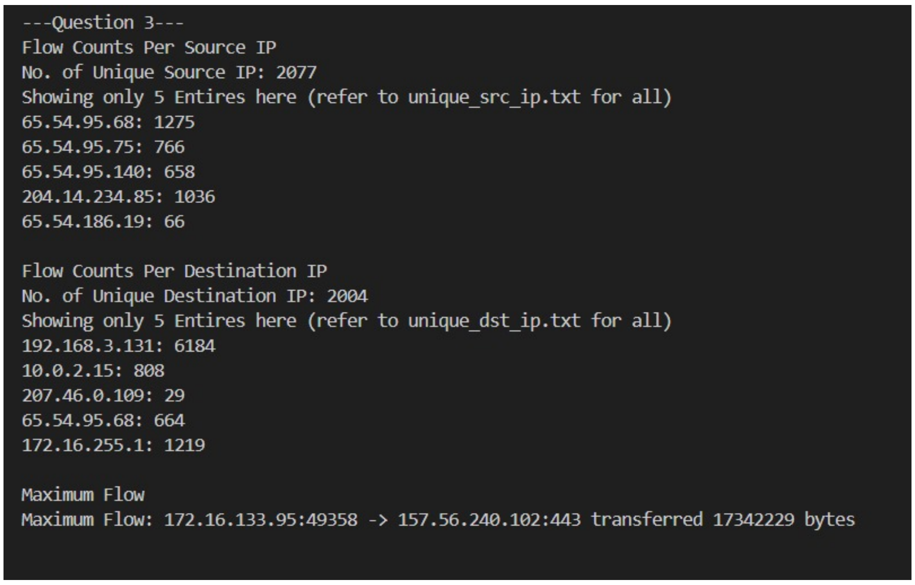
Unique destination IP's and their corresponding total flows were stored in a text file named `unique_dst_ip.txt`, included in the GitHub repository. Only IP packets were considered. The stored contents are in the form `destination-ip : total flow count`.

4.5 Maximum Flow Analysis

To identify the source-destination pair (source IP:port and destination IP:port) that transferred the most data, a dictionary, `flow_data`, was maintained where the key represents the source-destination pair and the value stores the cumulative data transfer for that pair.

After updating the dictionary with the size of each transferred packet for each source-destination pair, we identified the source-destination pair with the highest total transfer.

> **Maximum Flow**: The source IP:port and destination IP:port pair that transferred the most data is `172.16.133.95:49358` $\rightarrow$ `157.56.240.102:443`, with a total of `17342229` bytes transferred.



```
---Question 3---
Flow Counts Per Source IP
No. of Unique Source IP: 2077
Showing only 5 Entires here (refer to unique_src_ip.txt for all)
65.54.95.68: 1275
65.54.95.75: 766
65.54.95.140: 658
204.14.234.85: 1036
65.54.186.19: 66

Flow Counts Per Destination IP
No. of Unique Destination IP: 2004
Showing only 5 Entires here (refer to unique_dst_ip.txt for all)
192.168.3.131: 6184
10.0.2.15: 808
207.46.0.109: 29
65.54.95.68: 664
172.16.255.1: 1219

Maximum Flow
Maximum Flow: 172.16.133.95:49358 -> 157.56.240.102:443 transferred 17342229 bytes
```

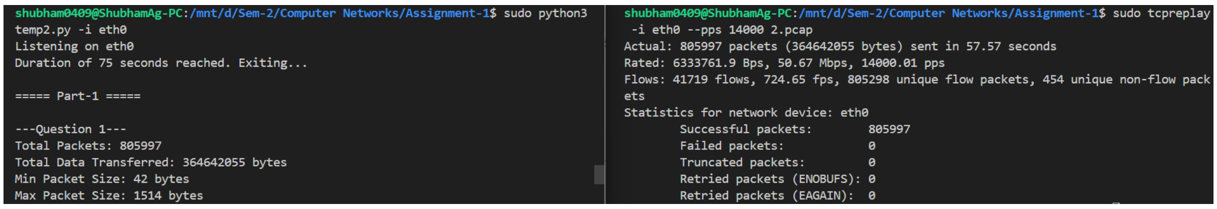
Figure 4: Result: Part-1 Q3

4.6 Top Speed Capture without Data Loss

The performance of the program was evaluated by capturing the content without any loss of data, under the following two conditions:

1. Running both tcpreplay and the sniffer program on the same machine (VM)
2. Running tcpreplay on one machine and the sniffer program on a different machine.

In both cases, the performance was measured in terms of the number of packets per second (pps) and the bandwidth in megabits per second (mbps) to determine the maximum speed at which the program could successfully capture all the data without the loss of any bytes/packets. The results of these tests are shown in the table below:



```
shubham0409@ShubhamAg-PC:/mnt/d/Sem-2/Computer Networks/Assignment-1$ sudo python3
temp2.py -i eth0
Listening on eth0
Duration of 75 seconds reached. Exiting...

===== Part-1 =====

---Question 1---
Total Packets: 805997
Total Data Transferred: 364642055 bytes
Min Packet Size: 42 bytes
Max Packet Size: 1514 bytes
```

```
shubham0409@ShubhamAg-PC:/mnt/d/Sem-2/Computer Networks/Assignment-1$ sudo tcpreplay
 -i eth0 --pps 14000 2.pcap
Actual: 805997 packets (364642055 bytes) sent in 57.57 seconds
Rated: 6333761.9 Bps, 50.67 Mbps, 14000.01 pps
Flows: 41719 flows, 724.65 fps, 805298 unique flow packets, 454 unique non-flow pack
ets
Statistics for network device: eth0
        Successful packets:        805997
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
```

Figure 5: Top Speed Same Machine

Scenario	PPS	Mbps
Same Machine	14000	50.67
Different Machines	-	-

Table 1: Top Speed Results

*The top speed varied every time, this is an approximation pertaining to the last few runs.

5 Part 2: Capture the Flag, PCAP specific questions

In this section, we address the following questions based on the TCP packet analysis.

Q1. *There is a message in TCP packets which contains my IP address. Find that IP address.*

Answer: The IP address was identified by searching for the string "My ip address = " in the payload of the TCP packets. The IP address was identified as `10.1.2.200`.

Q2. *Find the number of packets with that IP address.*

Answer: After identifying the IP address, the number of packets containing this IP address, either as a source or as a destination was counted. The total number of packets with this IP address was found out to be `80`.

Q3. *I have sent the name of the laptop in one of the TCP packets.*

1. **Find the name of the laptop.**

   Answer: The name of the laptop was found in the payload of the TCP packet by searching for the string "The name of laptop =". The laptop name was found to be `lenovo`.

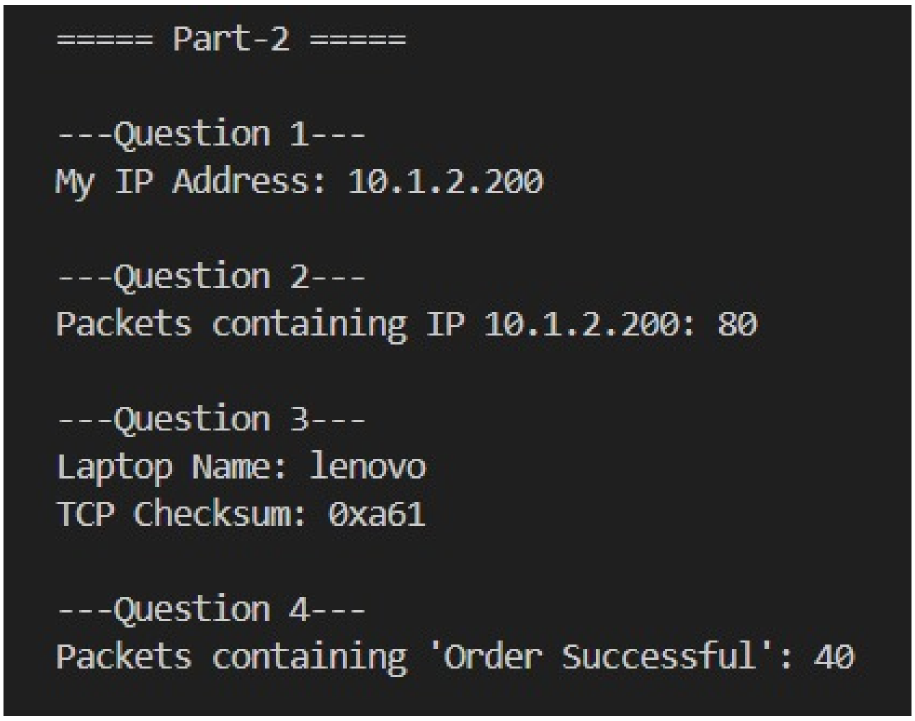
2. **Find the TCP checksum of that packet.**

   Answer: The TCP checksum for the packet containing the laptop name was calculated and found to be `0x0a61`.

Q4. *Find the number of packets which contain the message "Order Successfull".*

Answer: By searching for the string "Order Successfull" in the payload of the TCP packets, the total number of packets containing the message was determined to be `40` packets.

***The network and its contents were analyzed using `Wireshark` while replaying using `tcpreplay` to identify the exact string that needed to be searched for in the Python code.** This approach allowed us to inspect the raw packet data and extract the relevant string patterns, ensuring the correct matches were identified during the analysis.



```
===== Part-2 =====

---Question 1---
My IP Address: 10.1.2.200

---Question 2---
Packets containing IP 10.1.2.200: 80

---Question 3---
Laptop Name: lenovo
TCP Checksum: 0xa61

---Question 4---
Packets containing 'Order Successful': 40
```

Figure 6: Result: Part-2

6 Part 3: Real-Time Packet Capture

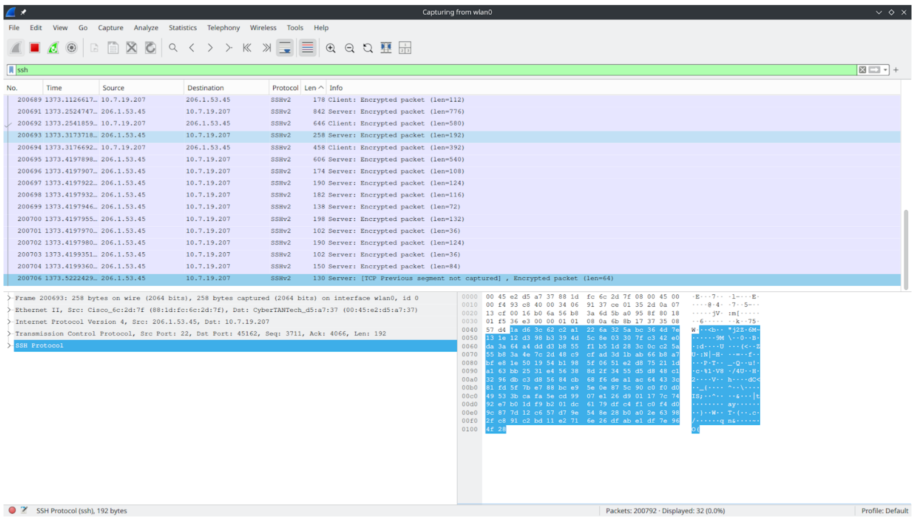
In this section, we capture and analyze network packets using the Wireshark tool to gain insights into various application layer protocols and network performance.

6.1 Wireshark Packet Capture

1(a). *List at least 5 different application layer protocols that we have not discussed so far in the classroom and describe in 1-2 sentences the operation/usage of the protocol and its layer of operation, along with the associated RFC number if any.*

Answer: Some unique application-layer protocols observed are:

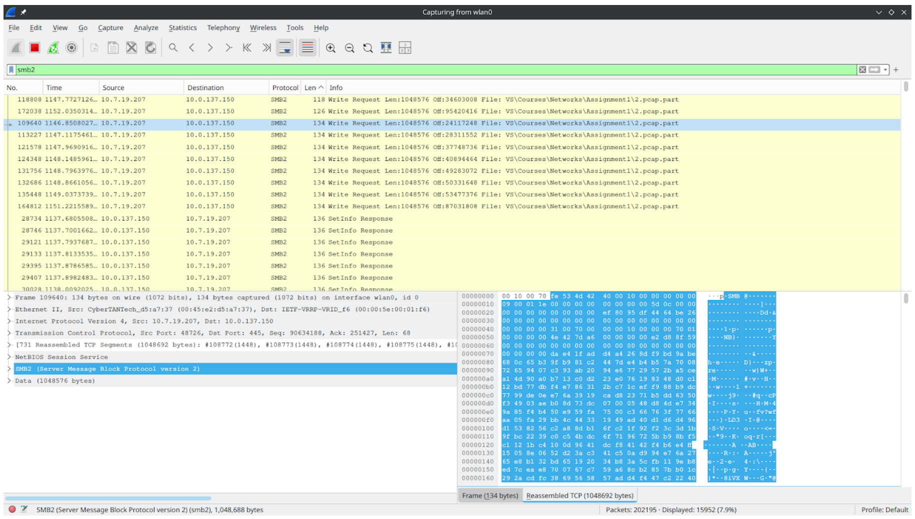
1. **SSHv2:** The Secure Shell (SSH) version 2 is a cryptographic network protocol used for secure data communication. It is primarily used for remote login to computer systems and for secure file transfer. SSHv2 provides strong authentication and encrypted data communication, replacing older protocols like Telnet and rlogin. **Associated Document:** *RFC 4251.*


Capturing from wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ssh

No.    Time          Source        Destination   Protocol Len Info
200689 1373.1126617… 10.7.19.207   206.1.53.45   SSHv2    178 Client: Encrypted packet (len=112)
200691 1373.2524747… 206.1.53.45   10.7.19.207   SSHv2    842 Server: Encrypted packet (len=776)
200692 1373.2541859… 10.7.19.207   206.1.53.45   SSHv2    646 Client: Encrypted packet (len=580)
200693 1373.3173718… 206.1.53.45   10.7.19.207   SSHv2    258 Server: Encrypted packet (len=192)
200694 1373.3176692… 10.7.19.207   206.1.53.45   SSHv2    458 Client: Encrypted packet (len=392)
200695 1373.4197898… 206.1.53.45   10.7.19.207   SSHv2    606 Server: Encrypted packet (len=540)
200696 1373.4197907… 206.1.53.45   10.7.19.207   SSHv2    174 Server: Encrypted packet (len=108)
200697 1373.4197922… 206.1.53.45   10.7.19.207   SSHv2    190 Server: Encrypted packet (len=124)
200698 1373.4197932… 206.1.53.45   10.7.19.207   SSHv2    182 Server: Encrypted packet (len=116)
200699 1373.4197946… 206.1.53.45   10.7.19.207   SSHv2    138 Server: Encrypted packet (len=72)
200700 1373.4197955… 206.1.53.45   10.7.19.207   SSHv2    198 Server: Encrypted packet (len=132)
200701 1373.4197970… 206.1.53.45   10.7.19.207   SSHv2    102 Server: Encrypted packet (len=36)
200702 1373.4197980… 206.1.53.45   10.7.19.207   SSHv2    190 Server: Encrypted packet (len=124)
200703 1373.4199351… 206.1.53.45   10.7.19.207   SSHv2    102 Server: Encrypted packet (len=36)
200704 1373.4199360… 206.1.53.45   10.7.19.207   SSHv2    150 Server: Encrypted packet (len=84)
200706 1373.5222429… 206.1.53.45   10.7.19.207   SSHv2    130 Server: [TCP Previous segment not captured] , Encrypted packet (len=64)

> Frame 200693: 258 bytes on wire (2064 bits), 258 bytes captured (2064 bits) on interface wlan0, id 0
> Ethernet II, Src: Cisco_6c:2d:7f (88:1d:fc:6c:2d:7f), Dst: CyberTANTech_d5:a7:37 (00:45:e2:d5:a7:37)
> Internet Protocol Version 4, Src: 206.1.53.45, Dst: 10.7.19.207
> Transmission Control Protocol, Src Port: 22, Dst Port: 45162, Seq: 3711, Ack: 4066, Len: 192
> SSH Protocol

0000  00 45 e2 d5 a7 37 88 1d  fc 6c 2d 7f 08 00 45 00   ·E···7·· ·l-···E·
0010  00 f4 93 c8 40 00 34 06  91 37 ce 01 35 2d 0a 07   ····@·4· ·7··5-··
0020  13 cf 00 16 b0 6a 56 b8  3a 6d 5b a0 95 8f 80 18   ·····jV· :m[·····
0030  01 f5 36 e3 00 00 01 01  08 0a 6b 8b 17 37 35 08   ··6····· ··k··75·
0040  57 d4 1a d6 3c 62 c2 a1  22 6a 32 5a bc 36 4d 7e   W···<b·· "j2Z·6M~
0050  13 1e 12 d3 98 b3 39 4d  5c 8e 03 30 7f c3 42 e0   ······9M \··0··B·
0060  da 3a 64 a4 dd d3 b8 55  f1 b5 1d 28 3c 0c c2 5a   ·:d····U ···(<··Z
0070  55 b8 3a 4e 7c 2d 48 c9  cf ad 3d 1b ab 66 b8 a7   U·:N|-H· ··=··f··
0080  bf e8 1e 50 19 54 b1 98  5f 06 51 e2 d8 75 21 1d   ···P·T·· _·Q··u!·
0090  a1 63 bb 25 31 e4 56 38  8d 2f 34 55 d5 d8 48 c1   ·c·%1·V8 ·/4U··H·
00a0  32 96 db c3 d8 56 84 cb  68 f6 de a1 ac 64 43 3c   2····V·· h····dC<
00b0  81 fd 5f 7b e7 88 bc e9  5e 0e 87 5c 90 c0 f0 d0   ··_{···· ^··\····
00c0  49 53 3b ca fa 5e cd 99  07 e1 26 d9 01 17 7c 74   IS;··^·· ··&···|t
00d0  92 e7 b0 1d f9 b2 01 dc  61 79 df c4 f1 c0 f4 d0   ········ ay······
00e0  9c 87 7d 12 c6 57 d7 9e  54 8e 28 b0 a0 2e 63 98   ··}··W·· T·(··.c·
00f0  2f c8 91 c2 bd 11 e2 71  6e 26 df ab e1 df 7e 96   /······q n&····~·
0100  4f 28                                              O(

SSH Protocol (ssh), 192 bytes        Packets: 200792 · Displayed: 32 (0.0%)        Profile: Default
```

Figure 7: SSHv2

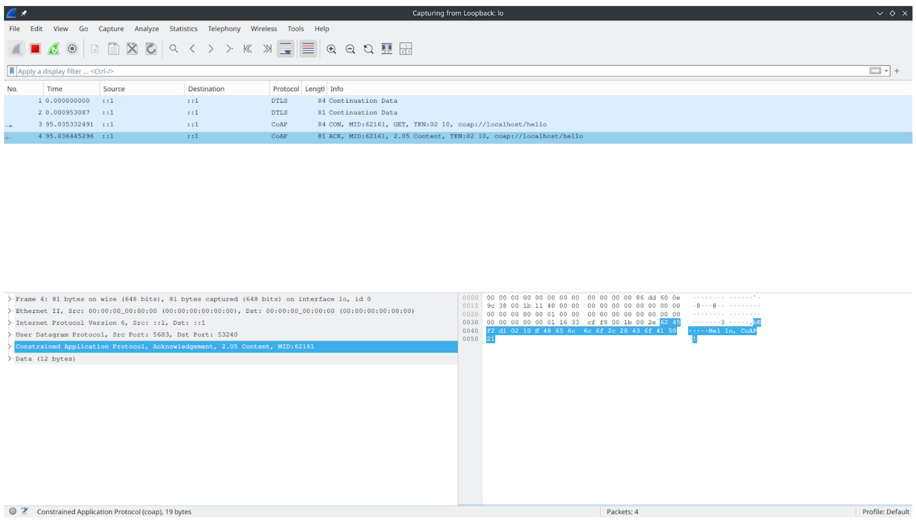
2. **SMB2:** Server Message Block version 2 (SMB2) is a network file sharing protocol that allows applications to read and write to files and request services from server programs in a computer network. It is an improved version of the SMB protocol, providing faster file sharing, and enhanced security and reliability. **Associated Document:** *Microsoft Document titled '[MS-SMB2]'*.


Capturing from wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
smb2

No.    Time          Source        Destination   Protocol Len Info
118808 1147.7727126… 10.7.19.207   10.0.137.150  SMB2     118 Write Request Len:1048576 Off:34603008 File: VS\Courses\Networks\Assignment1\2.pcap.part
172038 1152.0350314… 10.7.19.207   10.0.137.150  SMB2     126 Write Request Len:1048576 Off:95420416 File: VS\Courses\Networks\Assignment1\2.pcap.part
109640 1146.8508027… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:24117248 File: VS\Courses\Networks\Assignment1\2.pcap.part
113227 1147.1175461… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:28311552 File: VS\Courses\Networks\Assignment1\2.pcap.part
121578 1147.9690916… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:37748736 File: VS\Courses\Networks\Assignment1\2.pcap.part
124348 1148.1485961… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:40894464 File: VS\Courses\Networks\Assignment1\2.pcap.part
131756 1148.7963976… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:49283072 File: VS\Courses\Networks\Assignment1\2.pcap.part
132686 1148.8661056… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:50331648 File: VS\Courses\Networks\Assignment1\2.pcap.part
135448 1149.0373739… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:53477376 File: VS\Courses\Networks\Assignment1\2.pcap.part
164812 1151.2215589… 10.7.19.207   10.0.137.150  SMB2     134 Write Request Len:1048576 Off:87031808 File: VS\Courses\Networks\Assignment1\2.pcap.part
 28734 1137.6805508… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 28746 1137.7001662… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 29121 1137.7937687… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 29133 1137.8133535… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 29395 1137.8786585… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 29407 1137.8982483… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response
 30028 1138.0092025… 10.0.137.150  10.7.19.207   SMB2     136 SetInfo Response

> Frame 109640: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface wlan0, id 0
> Ethernet II, Src: CyberTANTech_d5:a7:37 (00:45:e2:d5:a7:37), Dst: IETF-VRRP-VRID_f6 (00:00:5e:00:01:f6)
> Internet Protocol Version 4, Src: 10.7.19.207, Dst: 10.0.137.150
> Transmission Control Protocol, Src Port: 48726, Dst Port: 445, Seq: 90634188, Ack: 251427, Len: 68
> [731 Reassembled TCP Segments (1048692 bytes): #108772(1448), #108773(1448), #108774(1448), #108775(1448), #1
> NetBIOS Session Service
> SMB2 (Server Message Block Protocol version 2)
> Data (1048576 bytes)

00000000  00 10 00 70 fe 53 4d 42  40 00 10 00 00 00 00 00   ···p·SMB @·······
00000010  09 00 01 1e 00 00 00 00  00 00 00 00 5d 0c 00 00   ········ ····]···
00000020  00 00 00 00 00 00 00 00  ef 80 95 df 44 64 be 26   ········ ····Dd·&
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00000040  00 00 00 00 31 00 70 00  00 00 10 00 00 00 70 01   ····1·p· ······p·
00000050  00 00 00 00 4e 42 7d a6  00 00 00 00 e2 d8 8f 59   ····NB}· ·······Y
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00000070  00 00 00 00 da e4 1f ad  d4 a4 26 8d f9 bd 9a be   ········ ··&·····
00000080  68 0c 65 b3 9f b9 81 c2  44 7d e4 b4 b5 7a 70 08   h·e····· D}···zp·
00000090  72 65 94 07 c3 93 ab 20  94 e6 77 29 57 2b a5 ce   re······ ··w)W+··
000000a0  a1 4d 90 a0 b7 13 c0 d2  23 e0 76 19 83 48 d0 c1   ·M······ #·v··H··
000000b0  12 bd 77 db f4 e7 86 31  2b c7 1c ef f9 88 b9 dc   ··w····1 +·······
000000c0  77 99 de 0e e7 6a 39 19  ca d8 23 71 b5 dd 63 50   w····j9· ··#q··cP
000000d0  f3 49 03 ae b0 8d 73 dc  07 00 05 48 d8 4d e7 34   ·I····s· ···H·M·4
000000e0  9a 85 f4 b4 50 e9 59 fa  75 00 c3 66 76 3f 77 66   ····P·Y· u··fv?wf
000000f0  aa 05 fa 29 bb 4c 44 33  19 49 ad 40 d1 d6 d4 96   ···)·LD3 ·I·@····
00000100  d1 53 82 56 c2 a8 8d b1  6f c2 1f 92 f2 3c 3d 1b   ·S·V···· o····<=·
00000110  9f bc 22 39 c0 c5 4b dc  6f 71 96 72 5b b9 8b f5   ··"9··K· oq·r[···
00000120  c1 12 1b c4 10 0d 96 41  dc f8 41 42 f4 b6 e4 ff   ·······A ··AB····
00000130  15 05 8e 06 52 d2 3a c3  41 c5 0a d9 94 e7 6a 27   ····R·:· A·····j'
00000140  65 e8 b1 32 bd 65 19 20  34 b8 3a 5c fb 11 9e b8   e··2·e·  4·:\····
00000150  ed 7c ea e8 70 07 67 c7  59 a6 8c b2 85 7b b0 1c   ·|··p·g· Y····{··
00000160  29 2a cd fc 38 69 56 58  57 ad d4 f4 47 c2 22 40   )*··8iVX W···G·"@

Frame (134 bytes)   Reassembled TCP (1048692 bytes)

SMB2 (Server Message Block Protocol version 2) (smb2), 1,048,688 bytes        Packets: 202195 · Displayed: 15952 (7.9%)        Profile: Default
```

Figure 8: SMB2

3. **CoAP:** The Constrained Application Protocol (CoAP) is a lightweight web transfer protocol designed for constrained devices and low-power networks. It is used in the Internet of Things (IoT) to enable simple communication between devices over UDP. **Associated Document:** *RFC 7252*.


Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time          Source  Destination  Protocol  Lengtl  Info
  1  0.000000000   ::1     ::1          DTLS      84      Continuation Data
  2  0.000953087   ::1     ::1          DTLS      81      Continuation Data
  3  95.035332491  ::1     ::1          CoAP      84      CON, MID:62161, GET, TKN:02 10, coap://localhost/hello
  4  95.036445296  ::1     ::1          CoAP      81      ACK, MID:62161, 2.05 Content, TKN:02 10, coap://localhost/hello
```

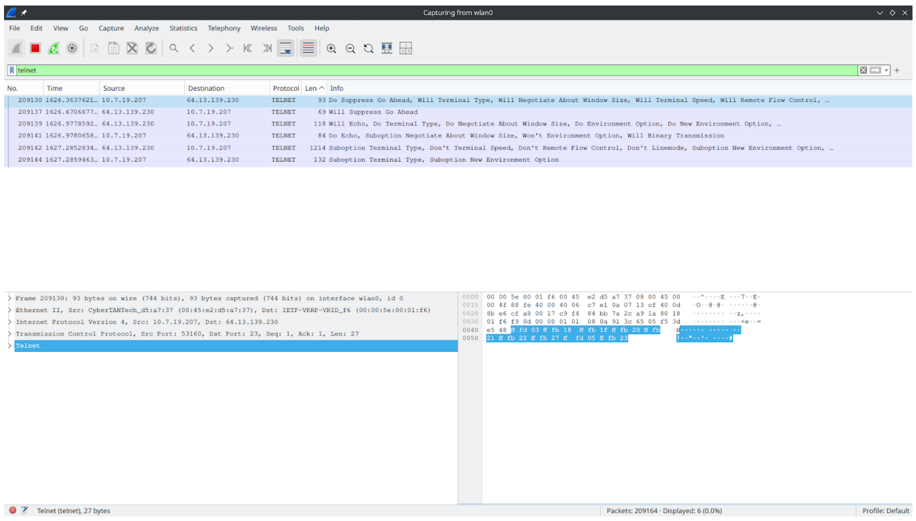
```
> Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> User Datagram Protocol, Src Port: 5683, Dst Port: 53240
> Constrained Application Protocol, Acknowledgement, 2.05 Content, MID:62161
> Data (12 bytes)
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0e   ········ ······`·
0010  9c 38 00 1b 11 40 00 00  00 00 00 00 00 00 00 00   ·8···@·· ········
0020  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 01 16 33  cf f8 00 1b 00 2e 62 45   ·······3 ······.bE
0040  f2 d1 02 10 ff 48 65 6c  6c 6f 2c 20 43 6f 41 50   ·····Hel lo, CoAP
0050  21                                                 !
```

```
Constrained Application Protocol (coap), 19 bytes          Packets: 4          Profile: Default
```

Figure 9: CoAP

4. **Telnet:** Telnet is a network protocol used to provide a command-line interface for communication with remote devices over a TCP/IP network. While historically used for remote server management, it is now considered insecure due to lack of encryption and has been largely replaced by SSH. **Associated Document:** *RFC 854*.


Capturing from wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
telnet

```
No.     Time           Source         Destination    Protocol  Len  Info
209130  1626.3637621…  10.7.19.207    64.13.139.230  TELNET      93 Do Suppress Go Ahead, Will Terminal Type, Will Negotiate About Window Size, Will Terminal Speed, Will Remote Flow Control, …
209137  1626.6706677…  64.13.139.230  10.7.19.207    TELNET      69 Will Suppress Go Ahead
209139  1626.9778592…  64.13.139.230  10.7.19.207    TELNET     118 Will Echo, Do Terminal Type, Do Negotiate About Window Size, Do Environment Option, Do New Environment Option, …
209141  1626.9780658…  10.7.19.207    64.13.139.230  TELNET      84 Do Echo, Suboption Negotiate About Window Size, Won't Environment Option, Will Binary Transmission
209142  1627.2852834…  64.13.139.230  10.7.19.207    TELNET    1214 Suboption Terminal Type, Don't Terminal Speed, Don't Remote Flow Control, Don't Linemode, Suboption New Environment Option, …
209144  1627.2859463…  10.7.19.207    64.13.139.230  TELNET     132 Suboption Terminal Type, Suboption New Environment Option
```

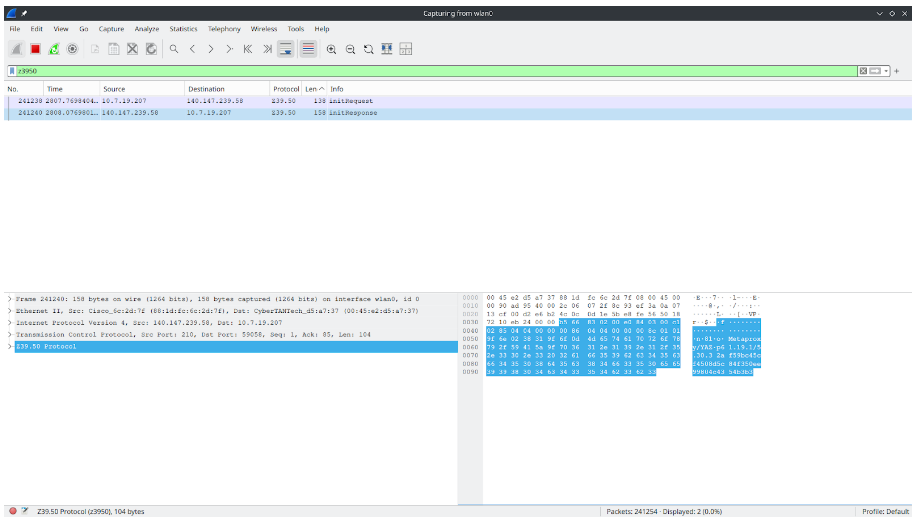
```
> Frame 209130: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface wlan0, id 0
> Ethernet II, Src: CyberTANTech_d5:a7:37 (00:45:e2:d5:a7:37), Dst: IETF-VRRP-VRID_f6 (00:00:5e:00:01:f6)
> Internet Protocol Version 4, Src: 10.7.19.207, Dst: 64.13.139.230
> Transmission Control Protocol, Src Port: 53160, Dst Port: 23, Seq: 1, Ack: 1, Len: 27
> Telnet
```

```
0000  00 00 5e 00 01 f6 00 45  e2 d5 a7 37 08 00 45 00   ··^····E ···7··E·
0010  00 4f 88 fe 40 00 40 06  c7 e1 0a 07 13 cf 40 0d   ·O··@·@· ······@·
0020  8b e6 cf a8 00 17 c9 f4  84 bb 7a 2c a9 1a 80 18   ········ ··z,····
0030  01 f6 f9 0d 00 00 01 01  08 0a 91 3c 65 05 f5 3d   ········ ···<e··=
0040  e5 48 ff fd 03 ff fb 18  ff fb 1f ff fb 20 ff fb   ·H······ ······ ··
0050  21 ff fb 22 ff fb 27 ff  fd 05 ff fb 23            !··"··'· ····#
```

```
Telnet (telnet), 27 bytes          Packets: 209164 · Displayed: 6 (0.0%)          Profile: Default
```

Figure 10: TELNET

5. **Z39.50:** Z39.50 is an information retrieval protocol designed for searching and retrieving bibliographic and related information from remote databases. It is widely used in libraries and academic institutions for searching catalog records. **Associated Document:** *RFC 1729*.


Capturing from wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
z3950

```
No.     Time            Source          Destination     Protocol Len Info
 241238 2807.7698404… 10.7.19.207     140.147.239.58  Z39.50   138 initRequest
 241240 2808.0769801… 140.147.239.58  10.7.19.207     Z39.50   158 initResponse
```

```
> Frame 241240: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface wlan0, id 0
> Ethernet II, Src: Cisco_6c:2d:7f (88:1d:fc:6c:2d:7f), Dst: CyberTANTech_d5:a7:37 (00:45:e2:d5:a7:37)
> Internet Protocol Version 4, Src: 140.147.239.58, Dst: 10.7.19.207
> Transmission Control Protocol, Src Port: 210, Dst Port: 59058, Seq: 1, Ack: 85, Len: 104
> Z39.50 Protocol
```

```
0000  00 45 e2 d5 a7 37 88 1d  fc 6c 2d 7f 08 00 45 00   ·E···7·· ·l-···E·
0010  00 90 ad 95 40 00 2c 06  07 2f 8c 93 ef 3a 0a 07   ····@·,· ·/···:··
0020  13 cf 00 d2 e6 b2 4c 0c  0d 1e 5b e8 fe 56 50 18   ······L· ··[··VP·
0030  72 10 eb 24 00 00 b5 66  83 02 00 e0 84 03 00 c1   r··$···f ········
0040  02 85 04 04 00 00 00 86  04 04 00 00 00 8c 01 01   ········ ········
0050  9f 6e 02 38 31 9f 6f 0d  4d 65 74 61 70 72 6f 78   ·n·81·o· Metaprox
0060  79 2f 59 41 5a 9f 70 36  31 2e 31 39 2e 31 2f 35   y/YAZ·p6 1.19.1/5
0070  2e 33 30 2e 33 20 32 61  66 35 39 62 63 34 35 63   .30.3 2a f59bc45c
0080  66 34 35 30 38 64 35 63  38 34 66 33 35 30 65 65   f4508d5c 84f350ee
0090  39 39 38 30 34 63 34 33  35 34 62 33 62 33         99804c43 54b3b3
```

```
Z39.50 Protocol (z3950), 104 bytes        Packets: 241254 · Displayed: 2 (0.0%)        Profile: Default
```

Figure 11: Z39.50

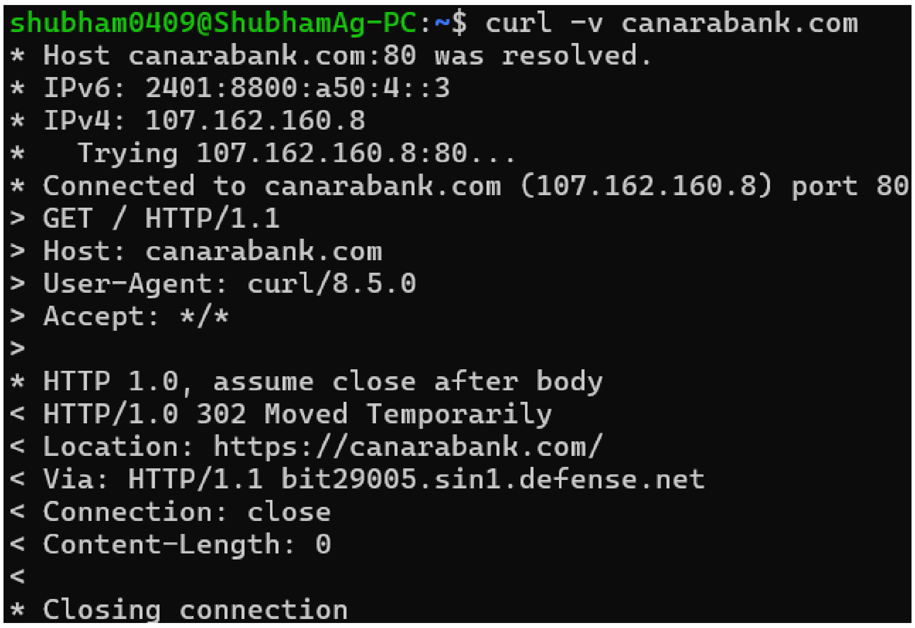
6.2 Website Packet Analysis

2(a). *Identify the request line with the version of the application layer protocol and the IP address for the websites* `canarabank.com`*,* `github.com`*, and* `netflix.com`*. Also, identify whether the connection(s) is/are persistent or not.*

Answer: The following information was gathered using the `curl -v` command to observe the request headers and connection details for each website listed.

1. **canarabank.com:**

   - Request Line: `GET / HTTP/1.1`
   - Application layer protocol version: HTTP/1.1
   - IP Address: 107.162.160.8
   - Persistent Connections: No

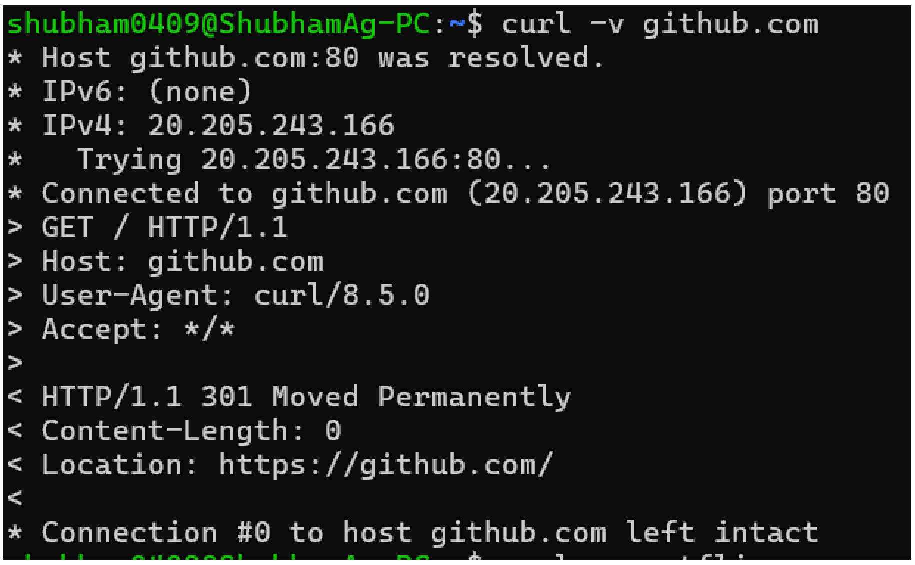


```
shubham0409@ShubhamAg-PC:~$ curl -v canarabank.com
* Host canarabank.com:80 was resolved.
* IPv6: 2401:8800:a50:4::3
* IPv4: 107.162.160.8
*   Trying 107.162.160.8:80...
* Connected to canarabank.com (107.162.160.8) port 80
> GET / HTTP/1.1
> Host: canarabank.com
> User-Agent: curl/8.5.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 302 Moved Temporarily
< Location: https://canarabank.com/
< Via: HTTP/1.1 bit29005.sin1.defense.net
< Connection: close
< Content-Length: 0
<
* Closing connection
```

Figure 12: Canarabank

2. **github.com:**

   - Request Line: `GET / HTTP/1.1`
   - Application layer protocol version: HTTP/1.1
   - IP Address: 20.205.243.166
   - Persistent Connections: Yes



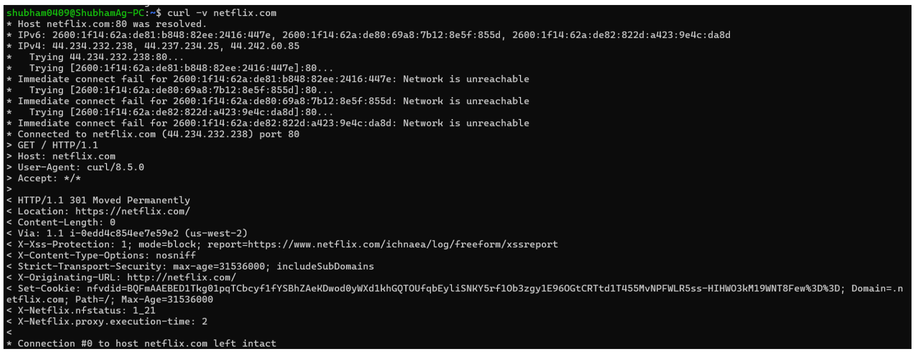
```
shubham0409@ShubhamAg-PC:~$ curl -v github.com
* Host github.com:80 was resolved.
* IPv6: (none)
* IPv4: 20.205.243.166
*   Trying 20.205.243.166:80...
* Connected to github.com (20.205.243.166) port 80
> GET / HTTP/1.1
> Host: github.com
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Content-Length: 0
< Location: https://github.com/
<
* Connection #0 to host github.com left intact
```

Figure 13: Github

3. **netflix.com:**

   - Request Line: `GET / HTTP/1.1`

- Application layer protocol version: HTTP/1.1
- IP Address: 44.234.232.238, 44.237.234.25, 44.242.60.85
- Persistent Connections: Yes



```
shubham0409@ShubhamAg-PC:~$ curl -v netflix.com
* Host netflix.com:80 was resolved.
* IPv6: 2600:1f14:62a:de81:b848:82ee:2416:447e, 2600:1f14:62a:de80:69a8:7b12:8e5f:855d, 2600:1f14:62a:de82:822d:a423:9e4c:da8d
* IPv4: 44.234.232.238, 44.237.234.25, 44.242.60.85
*   Trying 44.234.232.238:80...
*   Trying [2600:1f14:62a:de81:b848:82ee:2416:447e]:80...
* Immediate connect fail for 2600:1f14:62a:de81:b848:82ee:2416:447e: Network is unreachable
*   Trying [2600:1f14:62a:de80:69a8:7b12:8e5f:855d]:80...
* Immediate connect fail for 2600:1f14:62a:de80:69a8:7b12:8e5f:855d: Network is unreachable
*   Trying [2600:1f14:62a:de82:822d:a423:9e4c:da8d]:80...
* Immediate connect fail for 2600:1f14:62a:de82:822d:a423:9e4c:da8d: Network is unreachable
* Connected to netflix.com (44.234.232.238) port 80
> GET / HTTP/1.1
> Host: netflix.com
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Location: https://netflix.com/
< Content-Length: 0
< Via: 1.1 i-0edd4c854ee7e59e2 (us-west-2)
< X-Xss-Protection: 1; mode=block; report=https://www.netflix.com/ichnaea/log/freeform/xssreport
< X-Content-Type-Options: nosniff
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< X-Originating-URL: http://netflix.com/
< Set-Cookie: nfvdid=BQFmAAEBED1Tkg01pqTCbcyf1fYSBhZAeKDwod0yWXd1khGQTOUfqbEyliSNKY5rf1Ob3zgy1E96OGtCRTtd1T455MvNPFWLR5ss-HIHWO3kM19WNT8Few%3D%3D; Domain=.n
etflix.com; Path=/; Max-Age=31536000
< X-Netflix.nfstatus: 1_21
< X-Netflix.proxy.execution-time: 2
<
* Connection #0 to host netflix.com left intact
```

Figure 14: Netflix

*[If connection closed, then not persistent connection. If connection left intact, then persistent connection. Refer the screenshots attached]

2(b). *For any one of the websites, list any three header field names and corresponding values in the request and response message. Also, list any three HTTP error codes obtained while loading one of the pages with a brief description.*

1. **Website:** `https://github.com`
2. **Request Headers:**
   - **Origin:** `https://github.com/` Indicates the URL of the origin page.
   - **accept-language:** `en-GB,en-US;q=0.9,en;q=0.8` Indicates the preferred languages for the response
   - **Sec-CH-UA-Platform:** `"Windows"` Shows that the platform is Windows.
3. **Response Headers:**
   - **Accept-Ranges:** `bytes` Indicates that the server supports range requests, allowing clients to request specific byte ranges of a resource.
   - **Content-Type:** `text/css` Specifies that the content of the response is CSS (Cascading Style Sheets) code.

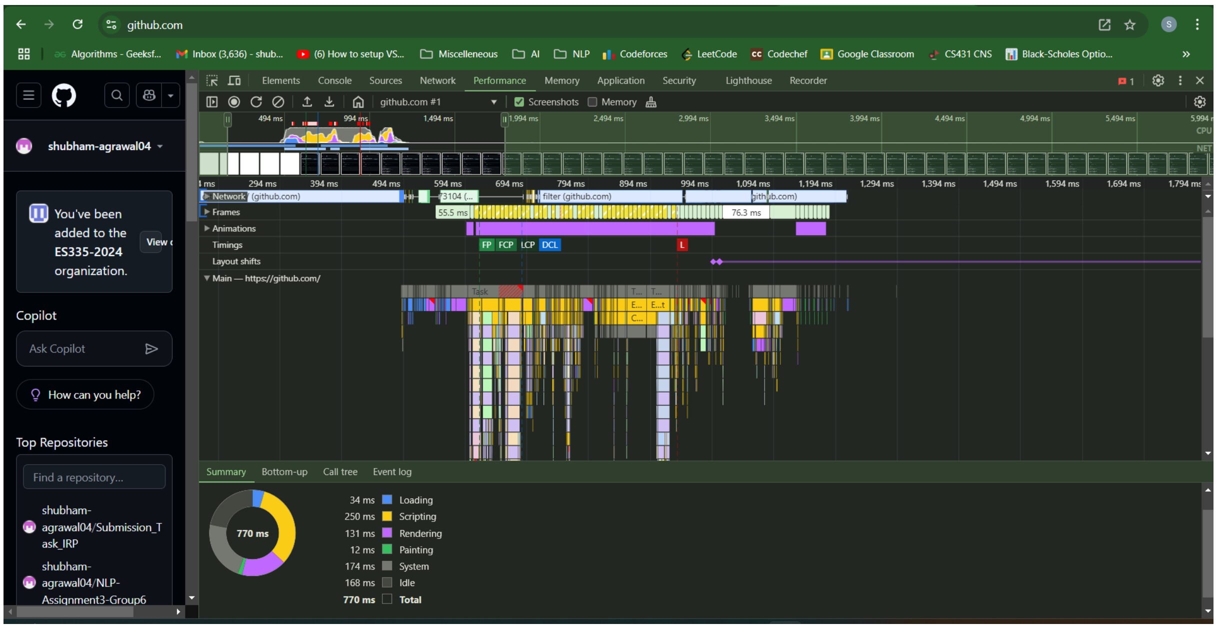
- **Content-Length:** 21799 Specifies the size of the response body in bytes, in this case, 21,799 bytes.

4. **HTTP Error Codes:**

   - **404 Not Found:** Indicates that the requested resource could not be found on the server.
   - **403 Forbidden:** The server understands the request but refuses to authorize it.
   - **429 Too Many Requests:** The user has sent too many requests in a given amount of time.

2(c). *Capture the Performance metrics that your browser records when a page is loaded and also report the list of cookies used and the associated flags in the request and response headers. Please report the browser name and screenshot of the performance metrics reported for any one of the page loads.*

1. **Webpage:** `https://github.com`
2. **Browser Name:** Google Chrome
3. **Performance Metrics:**



```
34 ms  Loading
250 ms  Scripting
131 ms  Rendering
12 ms  Painting
174 ms  System
168 ms  Idle
770 ms  Total
```

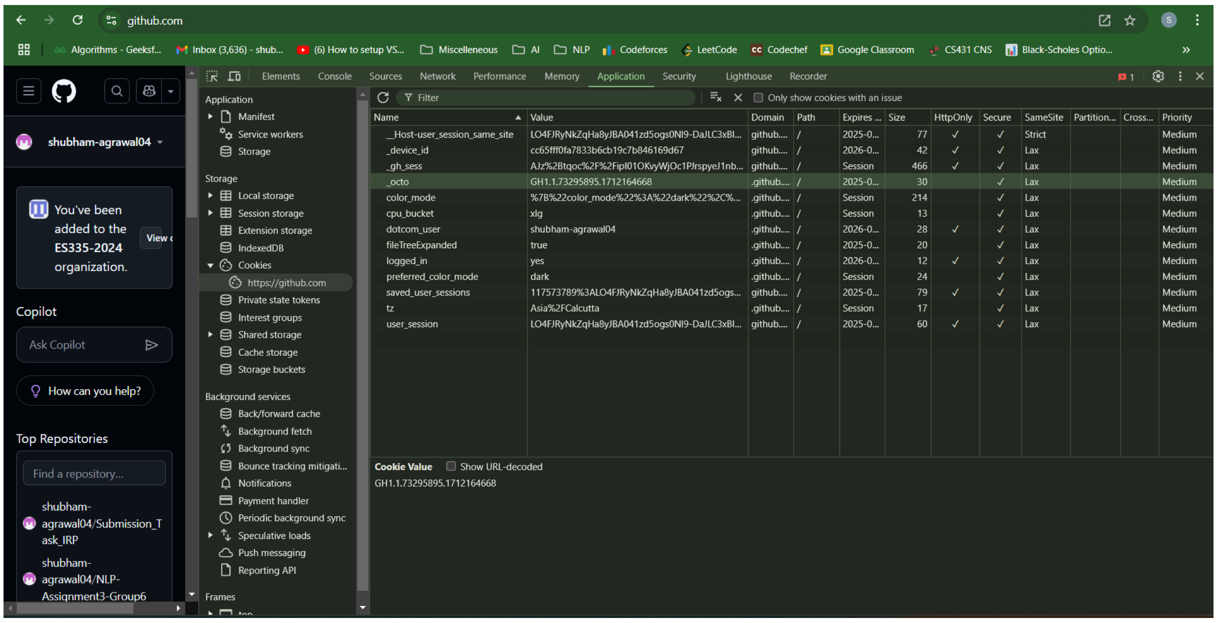
Figure 15: Performance Metrics

4. **Cookies and Flags:**

Key Cookies:

- `_gh_sess` (Session Cookie)
- `logged_in=yes`
- `preferred_color_mode=dark`
- `cpu_bucket=xlg`
- `tz=Asia/Calcutta`
- `dotcom_user=shubham-agrawal04`

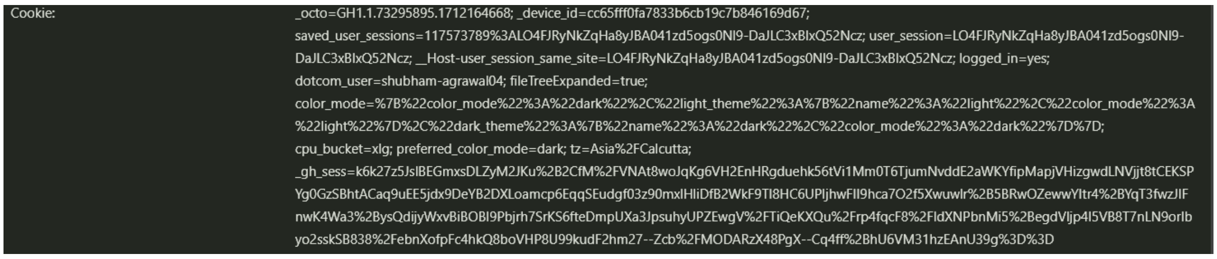
Flags: HTTPOnly and Secure are applied to the session cookie `_gh_sess`



Name	Value	Domain	Path	Expires ...	Size	HttpOnly	Secure	SameSite	Partition...	Cross...	Priority
__Host-user_session_same_site	LO4FJRyNkZqHa8yJBA041zd5ogs0NI9-DaJLC3xBl...	github....	/	2025-0...	77	✓	✓	Strict			Medium
_device_id	cc65fff0fa7833b6cb19c7b846169d67	github....	/	2026-0...	42	✓	✓	Lax			Medium
_gh_sess	AJz%2Btqoc%2F%2FipI01OKvyWjOc1PJrspyeJ1nb...	github....	/	Session	466	✓	✓	Lax			Medium
_octo	GH1.1.73295895.1712164668	.github....	/	2025-0...	30		✓	Lax			Medium
color_mode	%7B%22color_mode%22%3A%22dark%22%2C%...	.github....	/	Session	214		✓	Lax			Medium
cpu_bucket	xlg	.github....	/	Session	13		✓	Lax			Medium
dotcom_user	shubham-agrawal04	.github....	/	2026-0...	28	✓	✓	Lax			Medium
fileTreeExpanded	true	.github....	/	2025-0...	20		✓	Lax			Medium
logged_in	yes	.github....	/	2026-0...	12	✓	✓	Lax			Medium
preferred_color_mode	dark	.github....	/	Session	24		✓	Lax			Medium
saved_user_sessions	117573789%3ALO4FJRyNkZqHa8yJBA041zd5ogs...	github....	/	2025-0...	79	✓	✓	Lax			Medium
tz	Asia%2FCalcutta	.github....	/	Session	17		✓	Lax			Medium
user_session	LO4FJRyNkZqHa8yJBA041zd5ogs0NI9-DaJLC3xBl...	github....	/	2025-0...	60	✓	✓	Lax			Medium

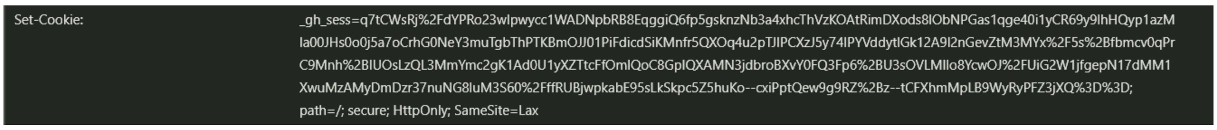
Cookie Value
GH1.1.73295895.1712164668

Figure 16: Cookies



```
Cookie:    _octo=GH1.1.73295895.1712164668; _device_id=cc65fff0fa7833b6cb19c7b846169d67;
           saved_user_sessions=117573789%3ALO4FJRyNkZqHa8yJBA041zd5ogs0NI9-DaJLC3xBlxQ52Ncz; user_session=LO4FJRyNkZqHa8yJBA041zd5ogs0NI9-
           DaJLC3xBlxQ52Ncz; __Host-user_session_same_site=LO4FJRyNkZqHa8yJBA041zd5ogs0NI9-DaJLC3xBlxQ52Ncz; logged_in=yes;
           dotcom_user=shubham-agrawal04; fileTreeExpanded=true;
           color_mode=%7B%22color_mode%22%3A%22dark%22%2C%22light_theme%22%3A%7B%22name%22%3A%22light%22%2C%22color_mode%22%3A
           %22light%22%7D%2C%22dark_theme%22%3A%7B%22name%22%3A%22dark%22%2C%22color_mode%22%3A%22dark%22%7D%7D;
           cpu_bucket=xlg; preferred_color_mode=dark; tz=Asia%2FCalcutta;
           _gh_sess=k6k27z5JslBEGmxsDLZyM2JKu%2B2CfM%2FVNAt8woJqKg6VH2EnHRgduehk56tVi1Mm0T6TjumNvddE2aWKYfipMapjVHizgwdLNVjjt8tCEKSP
           Yg0GzSBhtACaq9uEE5jdx9DeYB2DXLoamcp6EqqSEudgf03z90mxlHliDfB2WkF9Tl8HC6UPljhwFIl9hca7O2f5Xwuwlr%2B5BRwOZewwYltr4%2BYqT3fwzJIF
           nwK4Wa3%2BysQdijyWxvBiBOBI9Pbjrh7SrKS6fteDmpUXa3JpsuhyUPZEwgV%2FTiQeKXQu%2Frp4fqcF8%2FldXNPbnMi5%2BegdVljp4I5VB8T7nLN9orlb
           yo2sskSB838%2FebnXofpFc4hkQ8boVHP8U99kudF2hm27--Zcb%2FMODARzX48PgX--Cq4ff%2BhU6VM31hzEAnU39g%3D%3D
```

Figure 17: Request-header Cookies



```
Set-Cookie:        _gh_sess=q7tCWsRj%2FdYPRo23wlpwycc1WADNpbRB8EqggiQ6fp5gsknzNb3a4xhcThVzKOAtRimDXods8lObNPGas1qge40i1yCR69y9lhHQyp1azM
                   la00JHs0o0j5a7oCrhG0NeY3muTgbThPTKBmOJJ01PiFdicdSiKMnfr5QXOq4u2pTJIPCXzJ5y74IPYVddytlGk12A9l2nGevZtM3MYx%2F5s%2Bfbmcv0qPr
                   C9Mnh%2BIUOsLzQL3MmYmc2gK1Ad0U1yXZTtcFfOmlQoC8GplQXAMN3jdbroBXvY0FQ3Fp6%2BU3sOVLMllo8YcwOJ%2FUiG2W1jfgepN17dMM1
                   XwuMzAMyDmDzr37nuNG8luM3S60%2FffRUBjwpkabE95sLkSkpc5Z5huKo--cxiPptQew9g9RZ%2Bz--tCFXhmMpLB9WyRyPFZ3jXQ%3D%3D;
                   path=/; secure; HttpOnly; SameSite=Lax
```

Figure 18: Response-header Cookies

7 Files and Code Location

All files and code related to this assignment can be found in the following GitHub repository:

`https://github.com/shubham-agrawal04/CN_Assignment-1`

8 Replicating the Results

To replicate the results of this assignment for Part-1 and Part-2, follow these steps:

8.1 Clone the repository

Clone the repository using the following command:

```
git clone https://github.com/shubham-agrawal04/CN_Assignment-1.git
cd repo-directory
```

8.2 Install dependencies

Ensure that you have Python installed on your system as well as the necessary Python libraries.

8.3 Install PCAP File

Ensure that you have 2.pcap file installed on your device. Move the file into the repo-directory.

Download here: `https://drive.google.com/drive/folders/1n84jGddZ38fDjy9jKH3qw3J_H0SaKThu`

8.4 Set up your environment

- Configure the Ethernet connection between two machines correctly.
- It is recommended to use the `eth0` port of your WSL instance in Windows for both `tcpreplay` and the packet sniffing script.
- Ensure that your Wi-Fi internet connection is turned off to avoid interference with the Ethernet-based communication.

8.5 Run the packet sniffing script

On the machine where you are sniffing packets, execute the following command:

```
sudo python3 temp2.py -i [interface/port-name]
```

8.6 Run tcpreplay

On the machine where tcpreplay is set up, use the following command to replay the packets:

```
sudo tcpreplay -i [interface/port-name] --pps=[pps-speed]  2.pcap
```

(Any pps less than equal to 14000 will idealy give correct results for same machine run)

(May need to try for multiple pps for getting optimal results for different machine run)

8.7 Check results

Wait for the script to complete executing and voila! You have the results.