CSE351: Computer Network
Homework 1

Student Name: Nguyen Minh Duc
Student ID: 20202026

1 Execution Environment

In this homework, my execution environment is a `ThinkStation` running Windows 11 Pro for Workstations 64-bit. The `Wireshark` version is 4.0.10 downloaded from Wireshark's homepage. The summary of the system information can be found in Table 1.

Hardware	ThinkStation
Operating system	Windows 11 Pro
Wireshark version	4.0.10

Table 1: Execution Environment

2 Traffic Analysis

In this homework, we are going to practice packet sniffing and analyze the network traffic data with 8 different applications using `Wireshark`.

2.1 Web

Let's start with the first-ever website, created by Tim Berners-Lee from CERN. The European Organization for Nuclear Research still hosts this webpage at http://info.cern.ch/hypertext/WWW/TheProject.html. Let's try to analyze the network traffic when accessing to this page using `Wireshark`.

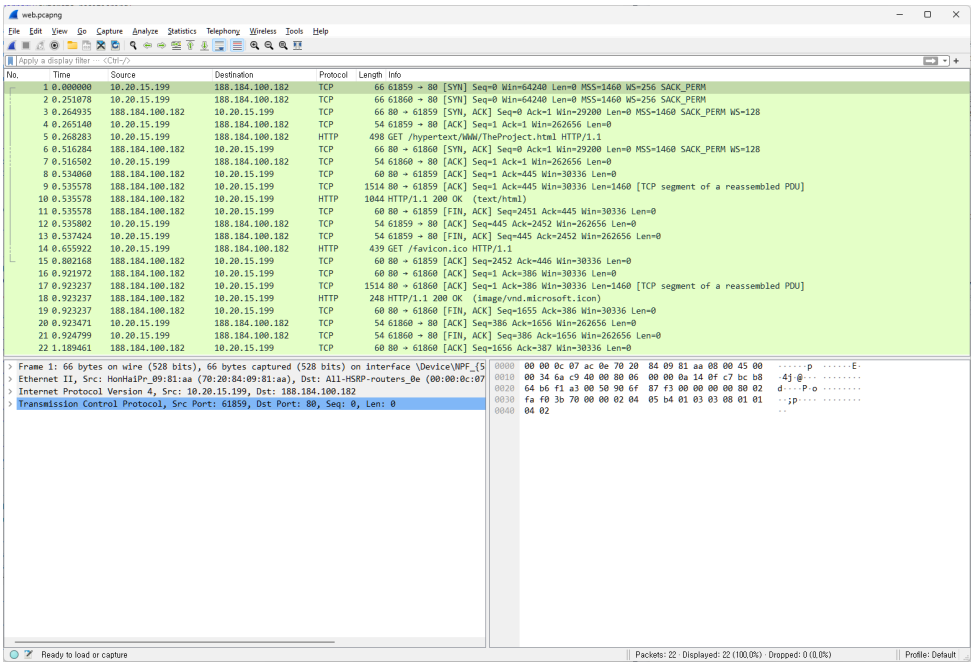


Figure 1: Web Traffic

There are a total of 22 packets sent and received from our hosts while trying to access the webpage. The traffic was created by opening up a browser (in this case it is Chrome, and I will use Chrome throughout this Homework), and entering the aforementioned URL.

In the beginning, there are 2 connection establishment requests numbered 1 and 2. The corresponding streams for the two initialization are $[1, 3, 4, 5, 8, 9, 10, 11, 12, 13, 15]$ and $[2, 6, 7, 14, 16, 17, 18, 19, 20, 21]$, respectively.

In the first stream, the client and the server establish the connection with ports 61859 and 80 respectively, resulting in packets 1 and 3. The client then sends packet 4 as an acknowledgment back to the server, while making a GET request to the server using HTTP/1.1. The server then acknowledges the client's request, sending back a `ACK` response and 2 packets (9 and 10), each containing the partitioned data requested from the user. Please check Figure 2 for more details. After sending all of the data, the server initiates the connection close to the server with packet 11, which is acknowledged by the client with packet 13, leading to the final packets of the stream, i.e. 15.

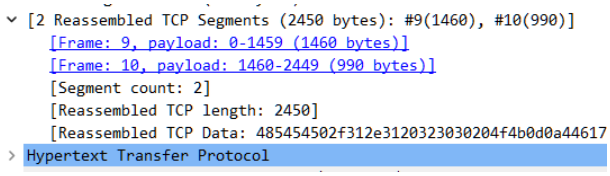


Figure 2: Two segments' length from the server's responses

There is another stream starting from packet 2 that has the same procedure as the above stream. The only difference is the requested data, in which the first stream is the web HTML document, and the second stream is the favicon. The total time taken in the network traffic is 1.189 seconds.

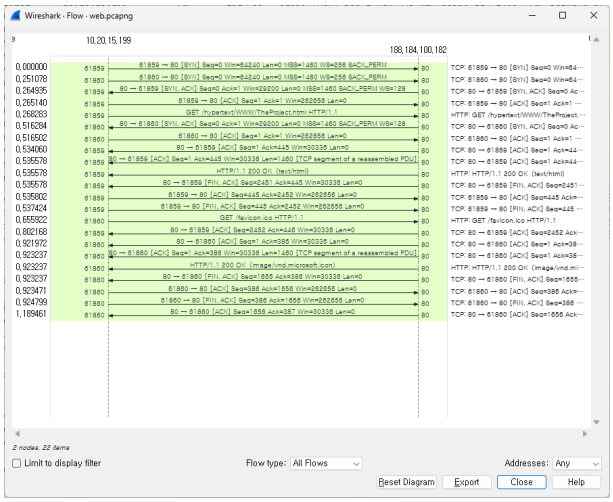


Figure 3: Flow chart of requesting the first website

The requesting and response formats are similar to what we have learned in class. The packets are encapsulated by different layers, namely Frame (link layer), Datagram(network layer), Segment(transport layer), and Message(application layer). Please refer to Figure 4 for more details. The structure of the response packets is also similar to the request packets.

In Figure 5, we can see the contents of packet 5 (request from client) and packets 9 and 10 (responses from server). The requested host is `info.cern.ch`, the HTTP version is 1.1, the connection is persistent, the client is Chrome, the request prefers text/HTML, XML, images, etc.., and the accepted language is Vietnamese. The server responded with an OK 200 code on Friday, October 27, 2023, at 05:40 GMT, the server is Apache, the content's length is 2217 bytes, it is of the type HTML, and this is a conditional GET (based on the Last-Modified field). The payload is the HTML of the website.

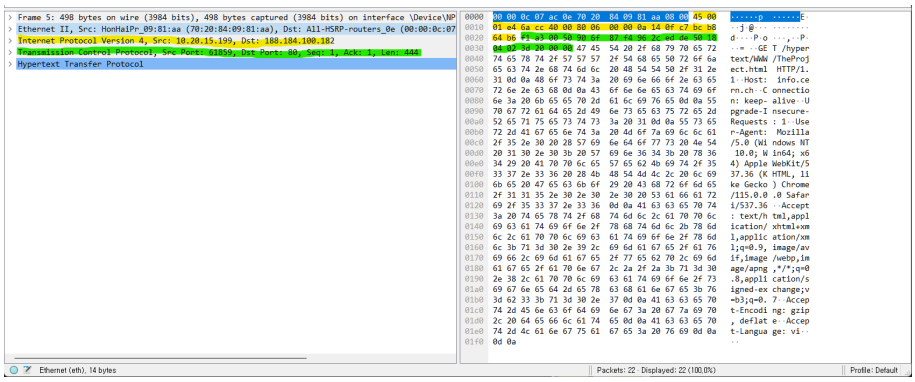


Figure 4: Example of a request format (Packet 5)

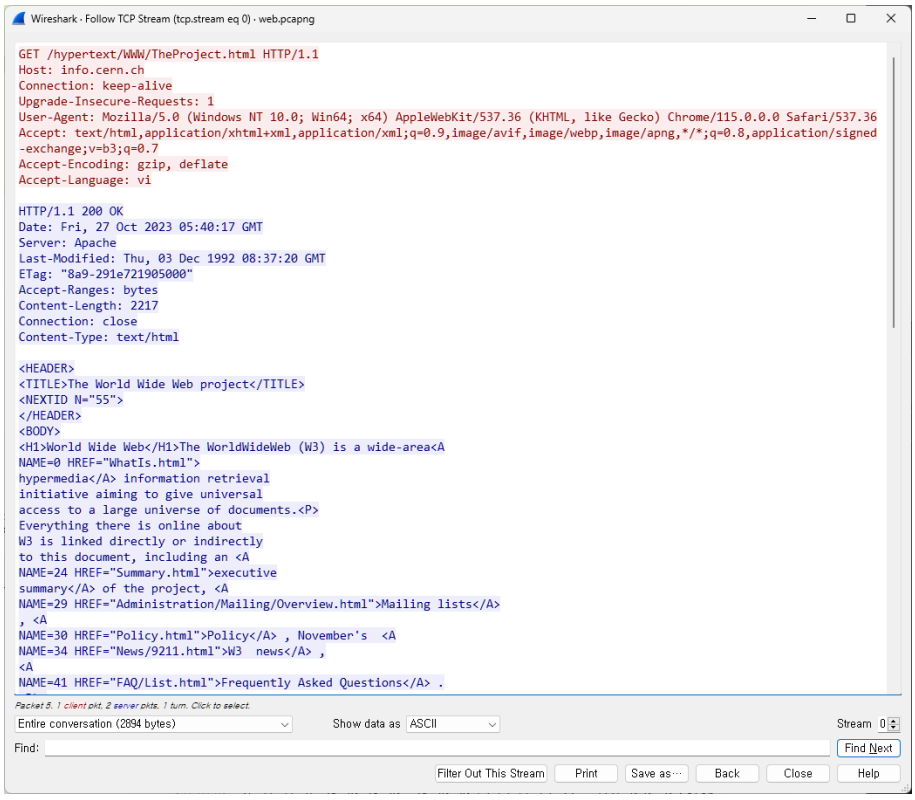


Figure 5: Conversation between server and client

2.2 E-mail

In this section, we will analyze the traffic when sending the email by using Thunderbird. First, I logged in to my Gmail account account and pressed `New Message` to draft an email. Thunderbird uses port number 587 to send the data, hence, we need to filter the packets in Wireshark by inserting the command `port 587` in the Capture Filter. The resulting packets of sending the email with a title of "123" and the body of "123123123123" are in Figure 7.

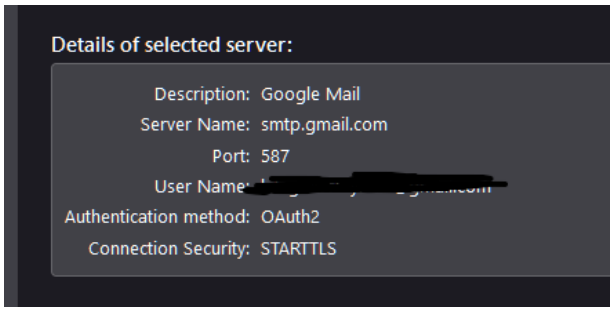


Figure 6: Thunderbird server information

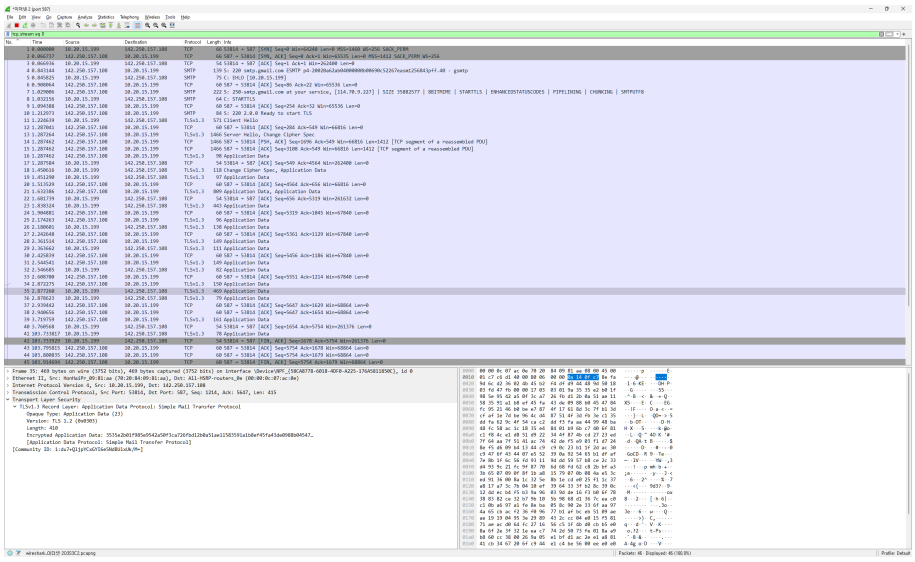


Figure 7: Email packets

As we can see, it first tries to establish the connection with the first packet and gets acknowledgment from the server in the second packet. Next, the server attempts to communicate with the client with SMTP protocol and the client continues the conversation by identifying its IP address with the message `EHLO [10.20.15.199]`. After that, the client sends a `STARTTLS` message, which will decrypt all of the messages exchanged in this conversation. Next, the client sends the email information including the sender, the recipient, the body, etc. to the server, and the server responds with exit codes. The conversation continues until packet 42, where the client stops the email sending and initiates the connection closing, to which the server responds with packet 45, ending the whole process. Please refer to Figure 8 for more details.

Figure 8: Email Conversation

The graph flow of the entire process can be found in Figure 9. The total time taken is 3.76 seconds.

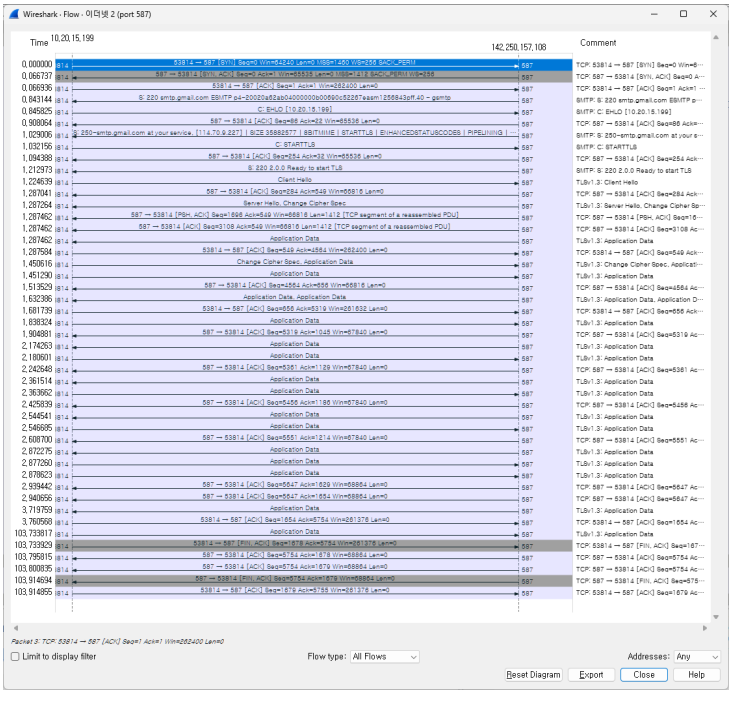


Figure 9: Email Flow

2.3 DNS

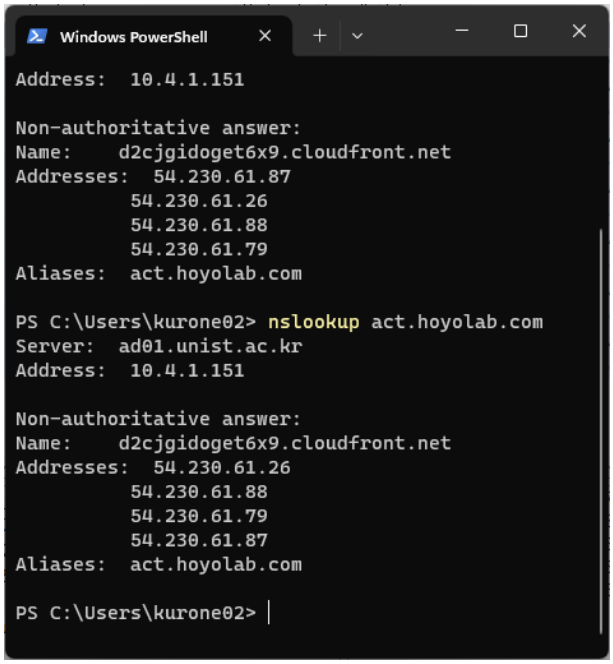


Figure 10: nslookup command

In this section, we are going to analyze the traffic created by querying the DNS. With a simple `nslookup` command, we can successfully query a domain name for its IP address. Please refer to Figure 10 for more details. I use the domain "act.hoyolab.com" for this section.

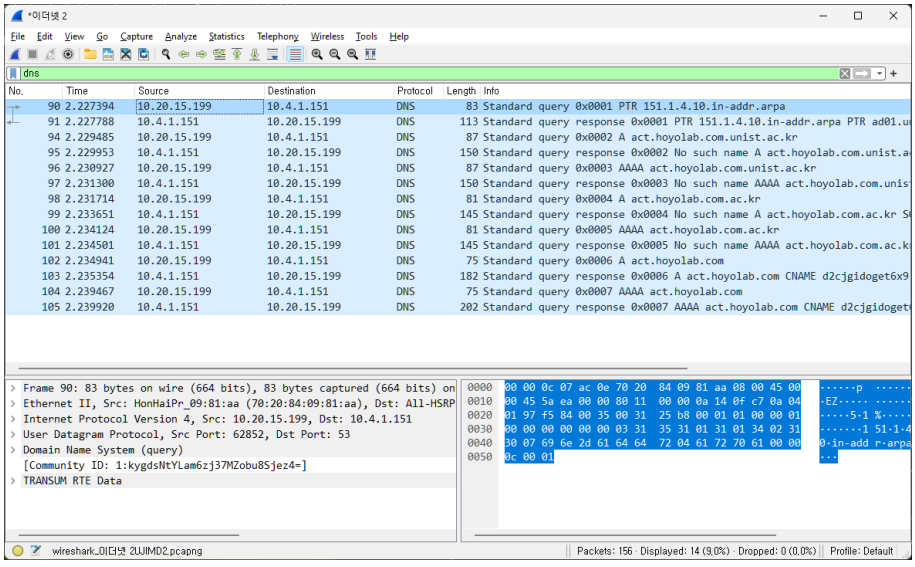


Figure 11: DNS packets

By filtering all other protocols that are not DNS, Wireshark displays 14 packets, which are 7 back-and-forth request and response pairs between the client and the local DNS server `10.4.1.151` to resolve the IP address. Here, note that all of them use UDP protocol (we can verify that by looking at the information in the bottom left corner of Figure 11). The destination port is 53, which is similar to the DNS port number that we learned in class. The query and answer pairs look as follows:

The local DNS server attempts to resolve the domain by requesting 5 different domains before finally getting the correct one. Note that there is an AAAA record type, which refers to the IPv6 type. In the 6th attempt, the server successfully obtains 5 RRs, which contain a CNAME type and 4 A types

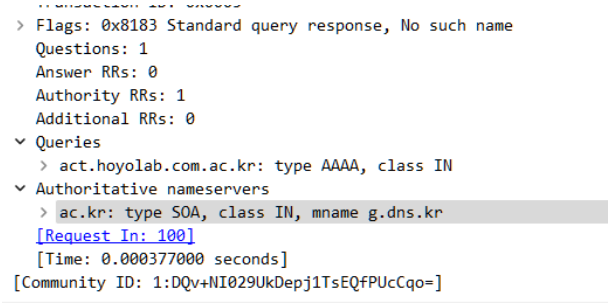


Figure 12: Query-Answer pair of packets 100 and 101

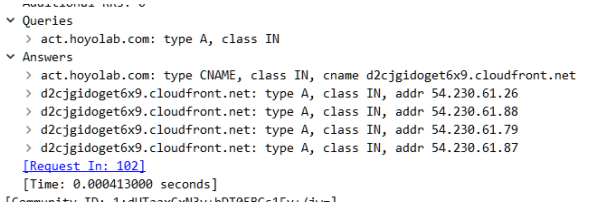


Figure 13: The 6-th attempt

(Figure 13, and in the last query, it returns an authoritative nameserver. The results contain the domain "cloudfront.net" indicating that Hoyolab is using CloudFront services.

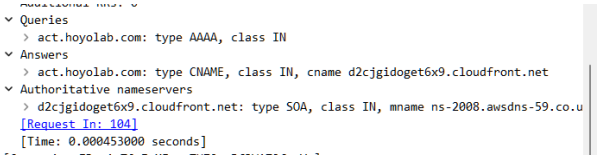


Figure 14: The final request

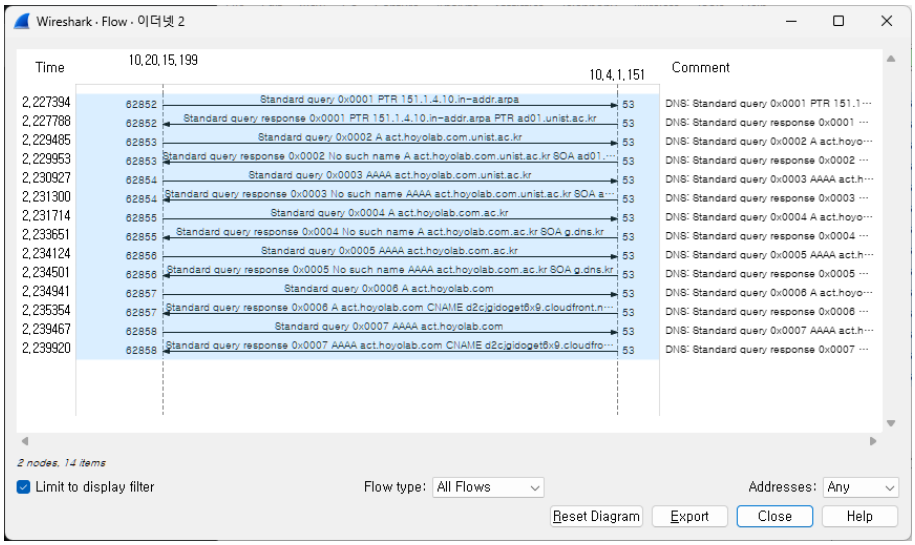


Figure 15: DNS Flow Graph

2.4 P2P File Sharing

In packets that has BT – DHT protocol, it is a packet for sharing peer list in dictionary format. After getting a peer list in p2p file sharing, especially in Bittorrent, it will make some TCP connections with peers in the peer list. After making a TCP connection, It will share chunks(the piece of file that I want

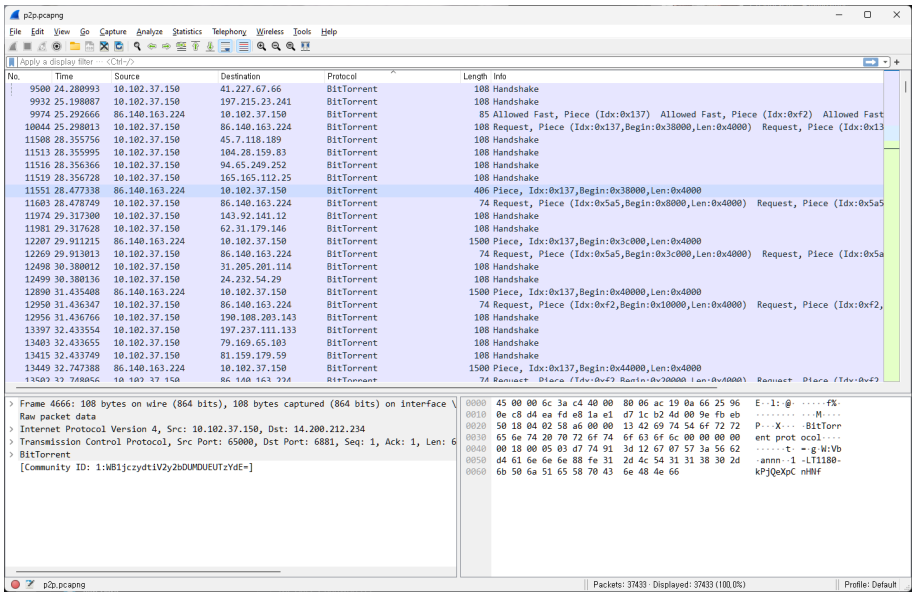


Figure 16: BitTorrent Traffic with BT-BHT protocol

to download) in TCP connections. It follows the tit-for-tat method and gets chunks from a peer who has the rarest chunks that I don't have.

2.5 Video Streaming

In this section, we will investigate the network traffic of YouTube, a popular video streaming platform. To create the traffic, we just need to open a browser and type www.youtube.com in the URL bar. Next, we can select a video to see the traffic in Wireshark. In particular, I choose this video: A masterpiece (click to watch, highly recommended). Before analyzing, we need to filter out all traffic that does not involve YouTube by entering `host www.youtube.com` in the Capture Filter. Please refer to Figure 20 for more details about the traffic. As we can see, the traffic consists of multiple UDP protocols, which is the appropriate protocol for video streaming since we need fast communication and loss can be tolerated. As we learned in class, `DASH` divides the video into multiple chunks in the time domain with various bit rates. We can observe in the traffic that the client is requesting chunks of the video and the server follows and sends the appropriate chunk to the user. Note that if the chunks are too large, for example from packets 9 to 20, we have to break that chunk into smaller pieces to send over the UDP protocol. Here, both the requests and the responses are encrypted, hence, we don't know the message details (Figure 18). Also, the communication port number to YouTube is 443, and the IP address of YouTube in my case is 142.250.199.110.

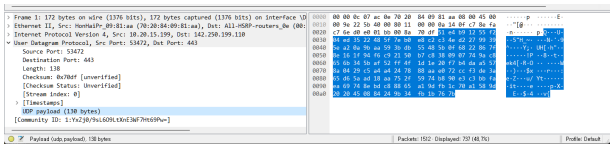


Figure 18: The first UDP request

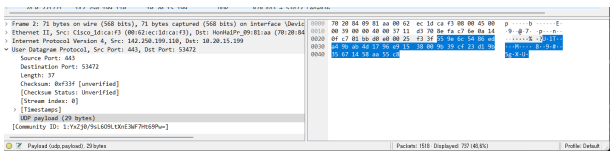


Figure 19: The first UDP response

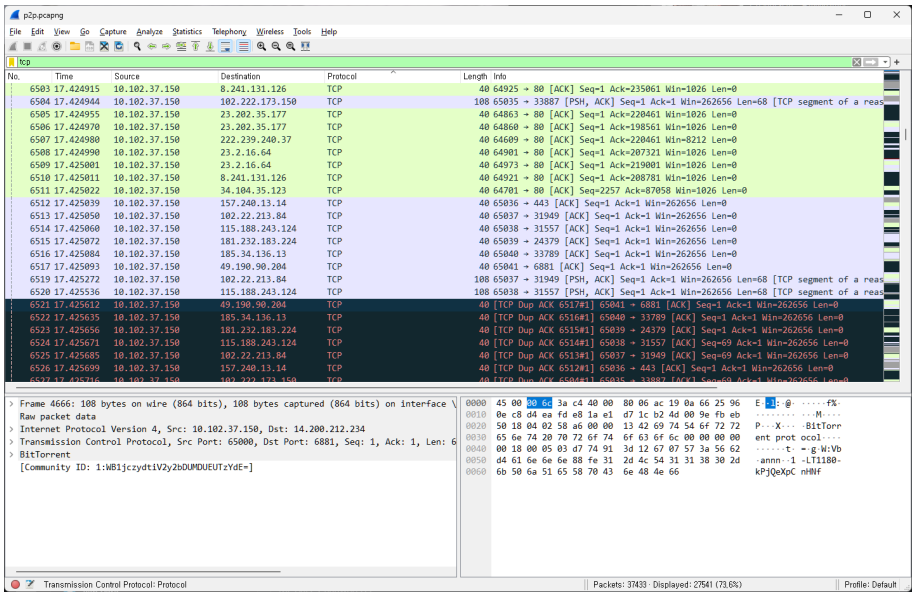


Figure 17: TCP connection with all peers in the list

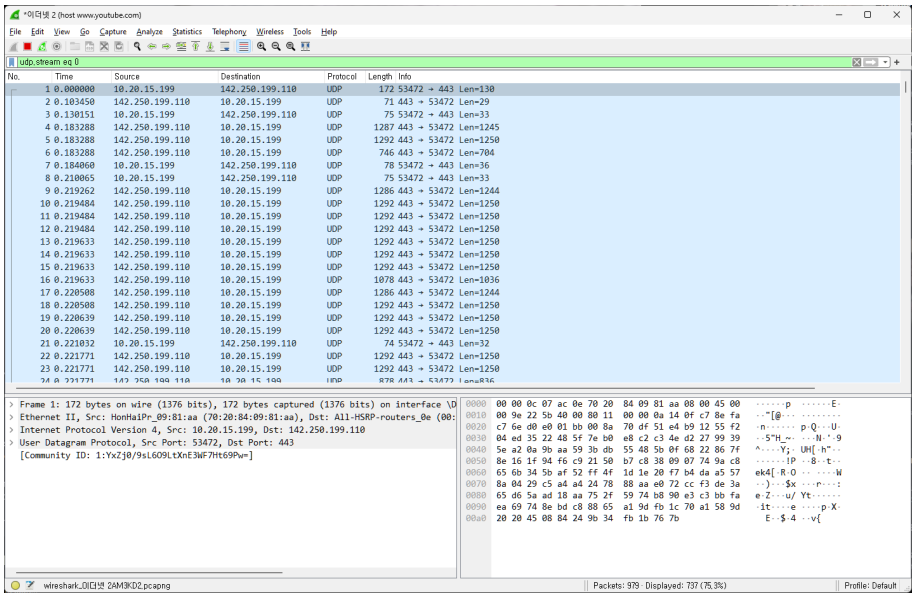


Figure 20: Video Traffic

2.6 Social Networking

In this section, we will analyze the network traffic when accessing Facebook. Same as the previous section, we will have to filter out all traffic that is not Facebook by using the command `host www.facebook.com`. Here, we can see that Facebook has an IP address of 31.13.76.35 in my case. Surprisingly, we only see the UDP protocol in the case of Facebook. Maybe it is because Facebook wants to ensure a smooth user experience by using the UDP protocol, which is lightweight, simple, and fast (since most of the contents are images and videos, loss can be tolerated). Most of the contents seem to be encrypted or images, so we cannot see the communication between the client and Facebook. Also, it seems that Facebook divides the requested data into smaller chunks if the data is too big (packets 13 to 24).

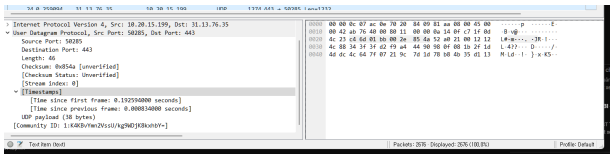


Figure 21: Client's request in packet 12

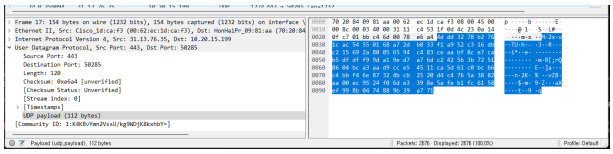


Figure 22: Facebook's response in packet 17

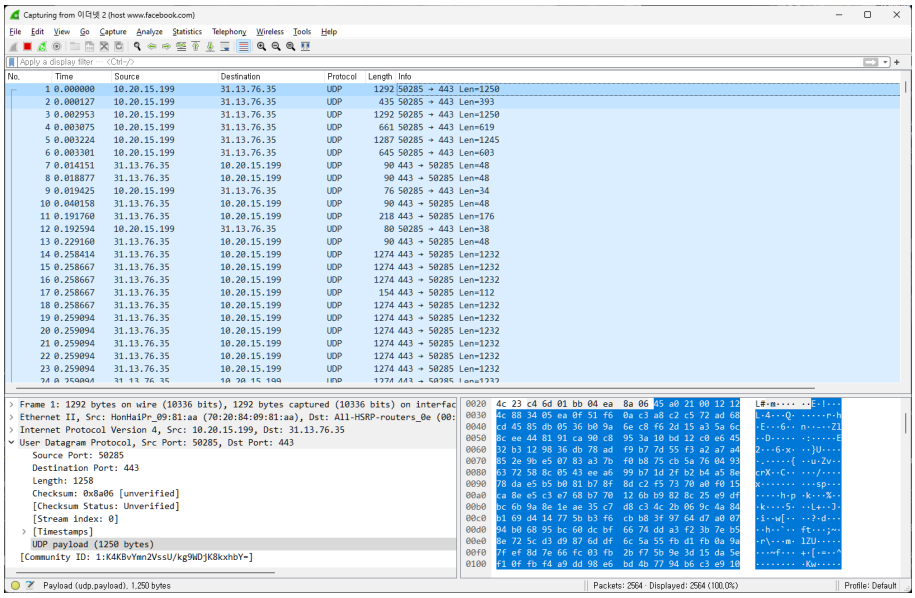


Figure 23: Facebook Traffic

2.7 SSH Login

In this section, let's analyze the traffic of SSH login. Here, I'm trying to access with the command `ssh -p [port number hidden due to security] duc@haiv.unist.ac.kr`, hence, we need the Capture Filter `host haiv.unist.ac.kr and port [port number hidden due to security]`. Please refer to Figure 25 for the traffic. Here, the SSH connection uses mostly TCP and WebSocket protocol. Unfortunately, the data is, again, encrypted for security, so we cannot see the messages. When initialized, it creates a lot of WebSocket instances, which might infers that they uses TCP in WebSocket (Figure **??**. When I type `ls` in the terminal of SSH, there are packets sent and received, which might contain the file name of the home directory (sadly, I cannot share the SSH screen).

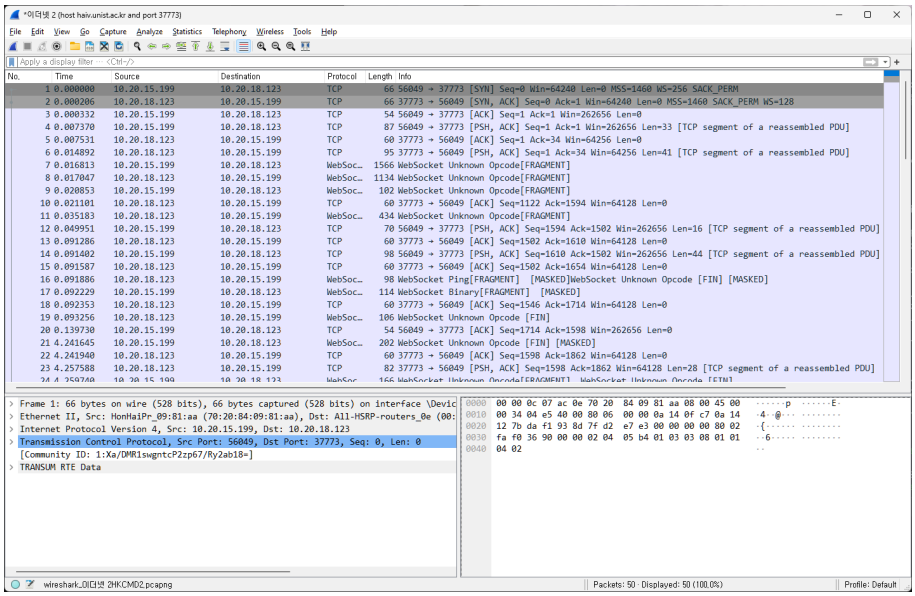


Figure 24: SSH Traffic

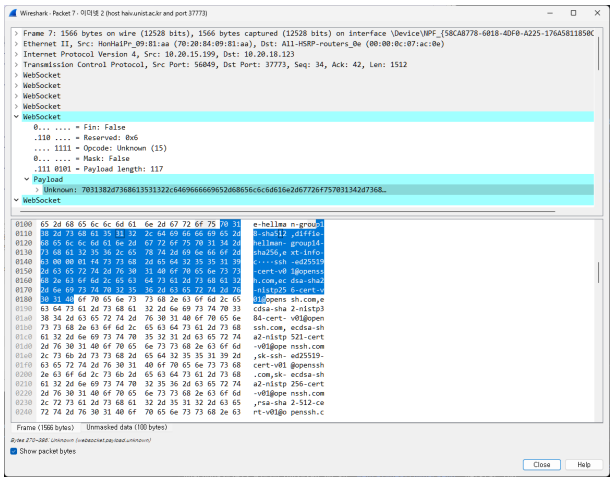
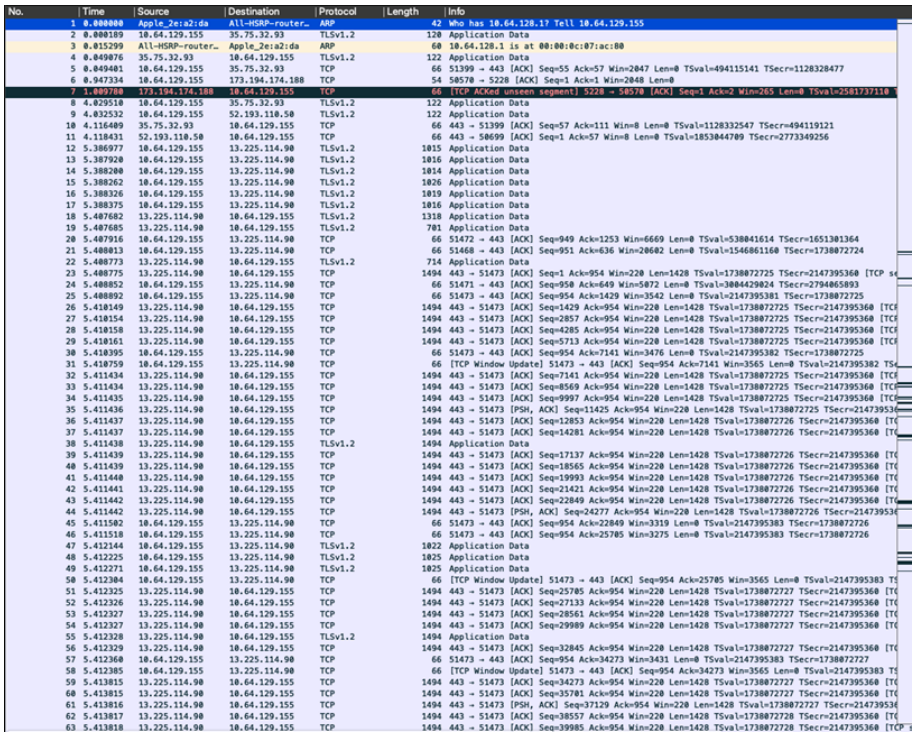


Figure 25: Request packet 7

2.8 Network games

In this section, we will analyze the traffic of a network game. My chosen game is `teen-titans-go-ninja-run` at cartoonnetworkhq.com. Surprisingly, the game uses TCP for communications while we learned that games usually use UDP for interactivity. Maybe it is because it needs to update the state of characters with more reliability, also it would not take much time because the information needed to run this game is not too heavy that TCP can easily handle.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Apple_2e:a2:da	All-HSRP-router…	ARP	42	Who has 10.64.128.17 Tell 10.64.129.155
2	0.000189	10.64.129.155	35.75.32.93	TLSv1.2	120	Application Data
3	0.015299	All-HSRP-router…	Apple_2e:a2:da	ARP	60	10.64.128.1 is at 00:00:0c:07:ac:80
4	0.049076	35.75.32.93	10.64.129.155	TLSv1.2	122	Application Data
5	0.049401	10.64.129.155	35.75.32.93	TCP	66	51399 → 443 [ACK] Seq=55 Ack=57 Win=2047 Len=0 TSval=494115141 TSecr=1128328477
6	0.947334	10.64.129.155	173.194.174.188	TCP	54	50570 → 5228 [ACK] Seq=1 Ack=1 Win=2048 Len=0
7	1.009780	173.194.174.188	10.64.129.155	TCP	66	[TCP ACKed unseen segment] 5228 → 50570 [ACK] Seq=1 Ack=2 Win=265 Len=0 TSval=2581737110 T
8	4.029510	10.64.129.155	35.75.32.93	TLSv1.2	122	Application Data
9	4.032532	10.64.129.155	52.193.110.50	TLSv1.2	122	Application Data
10	4.116409	35.75.32.93	10.64.129.155	TCP	66	443 → 51399 [ACK] Seq=57 Ack=111 Win=8 Len=0 TSval=1128332547 TSecr=494119121
11	4.118431	52.193.110.50	10.64.129.155	TCP	66	443 → 50699 [ACK] Seq=1 Ack=57 Win=8 Len=0 TSval=1853044709 TSecr=2773349256
12	5.386977	10.64.129.155	13.225.114.90	TLSv1.2	1015	Application Data
13	5.387920	10.64.129.155	13.225.114.90	TLSv1.2	1016	Application Data
14	5.388200	10.64.129.155	13.225.114.90	TLSv1.2	1014	Application Data
15	5.388262	10.64.129.155	13.225.114.90	TLSv1.2	1026	Application Data
16	5.388326	10.64.129.155	13.225.114.90	TLSv1.2	1019	Application Data
17	5.388375	10.64.129.155	13.225.114.90	TLSv1.2	1016	Application Data
18	5.407682	13.225.114.90	10.64.129.155	TLSv1.2	1318	Application Data
19	5.407685	13.225.114.90	10.64.129.155	TLSv1.2	701	Application Data
20	5.407916	10.64.129.155	13.225.114.90	TCP	66	51472 → 443 [ACK] Seq=949 Ack=1253 Win=6669 Len=0 TSval=538041614 TSecr=1651301364
21	5.408013	10.64.129.155	13.225.114.90	TCP	66	51468 → 443 [ACK] Seq=951 Ack=636 Win=20602 Len=0 TSval=1546861160 TSecr=1730072724
22	5.408773	13.225.114.90	10.64.129.155	TLSv1.2	714	Application Data
23	5.408775	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=1 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP se
24	5.408852	10.64.129.155	13.225.114.90	TCP	66	51471 → 443 [ACK] Seq=950 Ack=649 Win=5072 Len=0 TSval=3004429024 TSecr=2794065893
25	5.408892	10.64.129.155	13.225.114.90	TCP	66	51473 → 443 [ACK] Seq=954 Ack=1429 Win=3542 Len=0 TSval=2147395381 TSecr=1730072725
26	5.410149	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=1429 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
27	5.410154	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=2857 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
28	5.410158	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=4285 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
29	5.410161	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=5713 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
30	5.410395	10.64.129.155	13.225.114.90	TCP	66	51473 → 443 [ACK] Seq=954 Ack=7141 Win=3476 Len=0 TSval=2147395382 TSecr=1730072725
31	5.410759	10.64.129.155	13.225.114.90	TCP	66	[TCP Window Update] 51473 → 443 [ACK] Seq=954 Ack=7141 Win=3565 Len=0 TSval=2147395382 TSe
32	5.411434	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=7141 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
33	5.411434	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=8569 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
34	5.411435	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=9997 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=2147395360 [TCP
35	5.411436	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [PSH, ACK] Seq=11425 Ack=954 Win=220 Len=1428 TSval=1730072725 TSecr=214739536
36	5.411437	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=12853 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
37	5.411437	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=14281 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
38	5.411438	13.225.114.90	10.64.129.155	TLSv1.2	1494	Application Data
39	5.411439	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=17137 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
40	5.411439	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=18565 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
41	5.411440	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=19993 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
42	5.411441	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=21421 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
43	5.411442	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=22849 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=2147395360 [TC
44	5.411442	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [PSH, ACK] Seq=24277 Ack=954 Win=220 Len=1428 TSval=1730072726 TSecr=214739536
45	5.411502	10.64.129.155	13.225.114.90	TCP	66	51473 → 443 [ACK] Seq=954 Ack=22849 Win=3319 Len=0 TSval=2147395383 TSecr=1730072726
46	5.411518	10.64.129.155	13.225.114.90	TCP	66	51473 → 443 [ACK] Seq=954 Ack=25705 Win=3275 Len=0 TSval=2147395383 TSecr=1730072726
47	5.412144	10.64.129.155	13.225.114.90	TLSv1.2	1022	Application Data
48	5.412225	10.64.129.155	13.225.114.90	TLSv1.2	1025	Application Data
49	5.412271	10.64.129.155	13.225.114.90	TLSv1.2	1025	Application Data
50	5.412304	10.64.129.155	13.225.114.90	TCP	66	[TCP Window Update] 51473 → 443 [ACK] Seq=954 Ack=25705 Win=3565 Len=0 TSval=2147395383 TS
51	5.412325	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=25705 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
52	5.412326	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=27133 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
53	5.412327	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=28561 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
54	5.412327	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=29989 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
55	5.412328	13.225.114.90	10.64.129.155	TLSv1.2	1494	Application Data
56	5.412329	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=32845 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
57	5.412360	10.64.129.155	13.225.114.90	TCP	66	51473 → 443 [ACK] Seq=954 Ack=34273 Win=3431 Len=0 TSval=2147395383 TSecr=1730072727
58	5.412385	10.64.129.155	13.225.114.90	TCP	66	[TCP Window Update] 51473 → 443 [ACK] Seq=954 Ack=34273 Win=3565 Len=0 TSval=2147395383 TS
59	5.413815	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=34273 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
60	5.413815	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=35701 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=2147395360 [TC
61	5.413816	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [PSH, ACK] Seq=37129 Ack=954 Win=220 Len=1428 TSval=1730072727 TSecr=214739536
62	5.413817	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=38557 Ack=954 Win=220 Len=1428 TSval=1730072728 TSecr=2147395360 [TC
63	5.413818	13.225.114.90	10.64.129.155	TCP	1494	443 → 51473 [ACK] Seq=39985 Ack=954 Win=220 Len=1428 TSval=1730072728 TSecr=2147395360 [TCP

Figure 26: Game Traffic