

Computer Network Homework

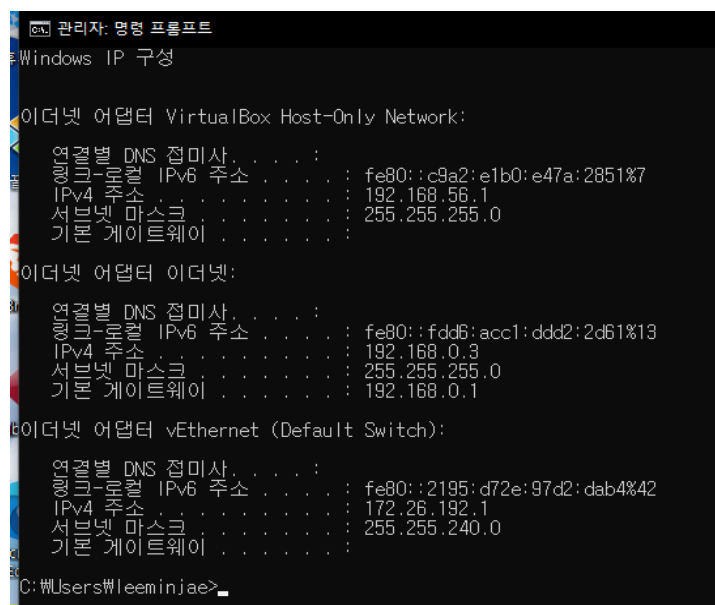
20161190 이민재

My homework is analyze different seven network applications by running Wireshark. WireShark is a program that allows to analyze packet. I downloaded the Stable Release (3.4.9 October6, 2121) version. My execution environment is as follows.

- OS : Windows 10 Pro (x64bit)
- Processor : Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
- Ram : 16.0GB
- SSD : 128GB
- HDD : 2000GB
- G-Card : NVIDIA GeForce GTX 660

I test 7 network applications Web, E-mail, DNS, network games, video conferencing, video streaming and social networking. While analyzing, there were some network applications that used similar protocols, but I tried to analyze the differences as much as possible. When some questions arise, I searched to find out why this result occurred. I also enter some command in cmd for more accurate analysis

Before starting the analysis, using the ipconfig command at the cmd terminal, my ip address(192.168.0.3) was checked.



```
관리자: 명령 프롬프트
Windows IP 구성

이더넷 어댑터 VirtualBox Host-Only Network:

연결별 DNS 접미사 . . . . . :
링크-로컬 IPv6 주소 . . . . . : fe80::c9a2:e1b0:e47a:2851%7
IPv4 주소 . . . . . : 192.168.56.1
서브넷 마스크 . . . . . : 255.255.255.0
기본 게이트웨이 . . . . . :

이더넷 어댑터 이더넷:

연결별 DNS 접미사 . . . . . :
링크-로컬 IPv6 주소 . . . . . : fe80::fdd6:acc1:ddd2:2db1%13
IPv4 주소 . . . . . : 192.168.0.3
서브넷 마스크 . . . . . : 255.255.255.0
기본 게이트웨이 . . . . . : 192.168.0.1

이더넷 어댑터 vEthernet (Default Switch):

연결별 DNS 접미사 . . . . . :
링크-로컬 IPv6 주소 . . . . . : fe80::2195:d72e:97d2:dab4%42
IPv4 주소 . . . . . : 172.26.192.1
서브넷 마스크 . . . . . : 255.255.240.0
기본 게이트웨이 . . . . . :

C:\Users\leeminjae>
```

★ Web

I did a web analysis. In Wireshark, too many packets are displayed. Because I only wanted to get information about web, I used capture filter "http". (There are another method) And access google and surfing. The result is as follows. Through the Info part, it can be seen that it is a packet related to the chrome web.

Top

middle

Bottom

No.	Time	Source	Destination	Protocol	Length	Info
76984	71.976478	192.168.0.3	34.104.35.123	HTTP	328	HEAD /edged1/chromewebstore/L2Nocm9tZV91eHR1bnNpb24vYmxvYnMvMjg0QUFYSnN4MFUtaEQwMDZqVGRkVkFmZWw1.0.6.0_aemomkdnpcapdnfajjbbcbdbj1jbpmpj.crx HTTP/1.1
76994	72.020073	34.104.35.123	192.168.0.3	HTTP	609	HTTP/1.1 200 OK
76998	72.039986	192.168.0.3	34.104.35.123	HTTP	379	GET /edged1/chromewebstore/L2Nocm9tZV91eHR1bnNpb24vYmxvYnMvMjg0QUFYSnN4MFUtaEQwMDZqVGRkVkFmZWw1.0.6.0_aemomkdnpcapdnfajjbbcbdbj1jbpmpj.crx HTTP/1.1
77013	72.082638	34.104.35.123	192.168.0.3	HTTP	90	HTTP/1.1 200 OK (application/x-chrome-extension)
78957	80.291306	192.168.0.3	34.64.4.16	HTTP	328	HEAD /update-delta/gcmjkmgd1gnkkccocmoeimainajmmjnii/9.30.0/9.29.4/d2e216bc74
78959	80.300450	34.64.4.16	192.168.0.3	HTTP	762	HTTP/1.1 200 OK
78964	80.338902	192.168.0.3	34.64.4.16	HTTP	400	GET /update-delta/gcmjkmgd1gnkkccocmoeimainajmmjnii/9.30.0/9.29.4/d2e216bc74
78969	80.348114	34.64.4.16	192.168.0.3	HTTP	499	HTTP/1.1 206 Partial Content
79031	82.302573	192.168.0.3	34.64.4.16	HTTP	403	GET /update-delta/gcmjkmgd1gnkkccocmoeimainajmmjnii/9.30.0/9.29.4/d2e216bc74
79033	82.312149	34.64.4.16	192.168.0.3	HTTP	871	HTTP/1.1 206 Partial Content

Frame 76984: 328 bytes on wire (2624 bits), 328 bytes captured (2624 bits) on interface \Device\NPF_{7BEC0FFC-8D98-437B-B6F0-62E4EB9A2CE2}, id 0
Ethernet II, Src: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82), Dst: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44)
Internet Protocol Version 4, Src: 192.168.0.3, Dst: 34.104.35.123
Transmission Control Protocol, Src Port: 13320, Dst Port: 80, Seq: 1, Ack: 1, Len: 274
Hypertext Transfer Protocol
HEAD /edged1/chromewebstore/L2Nocm9tZV91eHR1bnNpb24vYmxvYnMvMjg0QUFYSnN4MFUtaEQwMDZqVGRkVkFmZWw1.0.6.0_aemomkdnpcapdnfajjbbcbdbj1jbpmpj.crx HTTP/1.1
Connection: Keep-Alive\r\n
Accept: */*\r\n
Accept-Encoding: identity\r\n
User-Agent: Microsoft BITS/7.8\r\n

0030 02 02 e8 71 00 00 48 45 41 44 20 2f 65 64 67 65 ...q...HE AD /edge
0040 64 6c 2f 63 68 72 6f 6d 65 77 65 62 73 74 6f 72 d1/chrom ewebstor
0050 65 2f 4c 32 4e 6f 63 6d 39 74 5a 56 39 6c 65 48 e/L2Nocm 9tZV91eh

Text item (text), 150 byte(s) | Packets: 114613 - Displayed: 120 (0.1%) | Profile: Default

- Top part

At the first line. Source (My ip address 192.168.0.3) send to Destination (address is 34.104.35.123) a packet(Length is 328 and use HTTP protocol). So (192.168.0.3) request to (34.104.35.123).

At the second line, (34.104.35.123) response to (192.168.0.3). The content is 200 OK that mean request succeeded, requested object later in this message.

- Middle part (if you want more detail using double click, my case is HTTP)

The first line show information about Frame, bytes and so on. The second line show data link Layer(my case is Ethernet) The third line show about network layer. This include protocol version, total length, source address, destination address and so on.

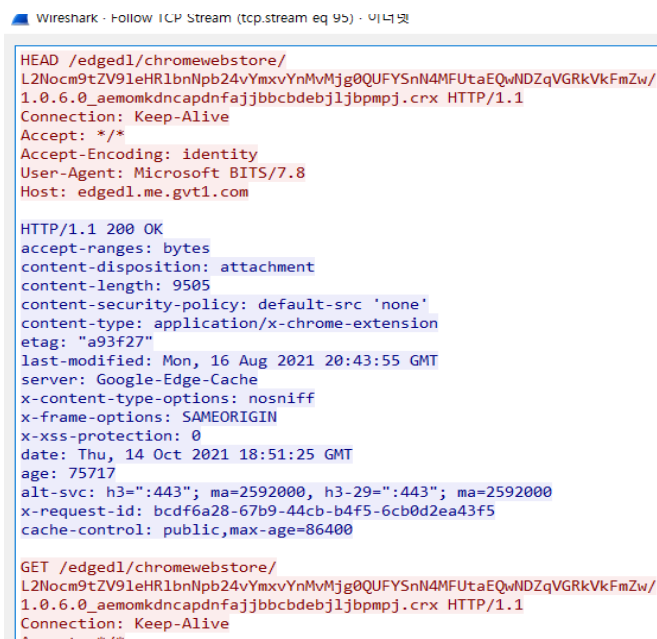
The forth line show transport layer(TCP protocol) that has source port, destination port, sequence number, ack number, headerlength, code bits and so on. As we learned HTTP uses TCP and port number is 80. Client initiates TCP connection to server and port 80.(HTTP's basic port num is 80)

And at the fifth line, I can see more detail Hypertext Transfer Protocol by double click. As we know HTTP request message is human-readable format. After looking at the other shapes,

HTTP request message's general format is request line, header lines, body. It also has \r\n that carriage return character, line-feed character. It was the same as learned in class. More detail, I can see web page consists of base HTML-file which includes several referenced objects. Each object is addressable by a URL too.

-Bottom part

The numbers at the bottom are change of request and response message contents to hexadecimal.



```
Wireshark - Follow TCP Stream (tcp.stream eq 95) - 이너넷

HEAD /edged1/chromewebstore/
L2Nocm9tZV91eHRlbnNpb24vYmxvYnMvMjg0QUFYSnN4MFUtaEQwNDZqVGRkVkFmZw/
1.0.6.0_aemomkdncapdnfajjbbcbdebjljbmpj.crx HTTP/1.1
Connection: Keep-Alive
Accept: */*
Accept-Encoding: identity
User-Agent: Microsoft BITS/7.8
Host: edged1.me.gvt1.com

HTTP/1.1 200 OK
accept-ranges: bytes
content-disposition: attachment
content-length: 9505
content-security-policy: default-src 'none'
content-type: application/x-chrome-extension
etag: "a93f27"
last-modified: Mon, 16 Aug 2021 20:43:55 GMT
server: Google-Edge-Cache
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-xss-protection: 0
date: Thu, 14 Oct 2021 18:51:25 GMT
age: 75717
alt-svc: h3=":443"; ma=2592000, h3-29=":443"; ma=2592000
x-request-id: bcdf6a28-67b9-44cb-b4f5-6cb0d2ea43f5
cache-control: public,max-age=86400

GET /edged1/chromewebstore/
L2Nocm9tZV91eHRlbnNpb24vYmxvYnMvMjg0QUFYSnN4MFUtaEQwNDZqVGRkVkFmZw/
1.0.6.0_aemomkdncapdnfajjbbcbdebjljbmpj.crx HTTP/1.1
Connection: Keep-Alive
Accept: */*
```

I want to see more deep. So using follow -> tcp Stream I capture this TCP screen. Red letter is data that client send to server ,the blue letter is data that server send to client.

Server send information that HTTP version is 1.1, response is 200 OK, accept-ranges is bytes, content-length is 9505, last-modified is MON, 16 Aug 2021 20:43:55 GMT... and so on.

-Summary

As learned in class, http is made of human-readable ASCII. The format was also expressed as request line, header lines, and body. Carriage return character and line-feed character could also be checked for each line. And HTTP used TCP.

And most of them used persistent connection (HTTP 1.1)(one RTT for all the referenced objects) more efficient than Non-persistent connection(2 RTT per object).

★ E-Mail

I did an e-mail analysis through “Outlook”. To do this, at the cmd I used the command “ping outlook.office.com” to find out the address of Outlook. The Outlook address was 52.98.51.146.

```
C:\Users\leeminjae>ping outlook.office.com
Ping 1CN-efz.ms-acdc.office.com [52.98.51.146] 32바이트 데이터 사용:
```

I heard that the mail service uses the SMTP protocol in class, but it was not easy to find it. Upon investigation, the actual mail client uses the TLS protocol. This is because SMTP does not provide encryption, which poses a high risk of hacking.

-TLS's handshaking process

The image shows a Wireshark packet capture of a TLS handshake. The packet list on the left shows several packets, with packet 5445 (Server Hello) selected. The packet details pane on the right shows the structure of the TLSv1.2 record, including the Handshake Protocol: Multiple Handshake Messages. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
5439	135.527646	192.168.0.3	52.109.44.44	TLSv1.2	248	Client Hello
5445	135.537584	52.109.44.44	192.168.0.3	TLSv1.2	113	Server Hello, Certificate, Certificate Status, Server Key Exchange, Se
5447	135.540992	192.168.0.3	52.109.44.44	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5448	135.550209	52.109.44.44	192.168.0.3	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
5449	135.550858	192.168.0.3	52.109.44.44	TLSv1.2	394	Application Data
5463	135.559745	192.168.0.3	52.109.44.44	TLSv1.2	1494	Application Data [TCP segment of a reassembled PDU]
5464	135.559745	192.168.0.3	52.109.44.44	TLSv1.2	1362	Application Data
5473	135.639238	52.109.44.44	192.168.0.3	TLSv1.2	868	Application Data

Frame 5445: 113 bytes on wire (904 bits), 113 bytes captured (904 bits) on interface \Device\NPF_{7BEC0FFC-8D98-437B-B6F0-62E4EB9A2CE2}, id 0
> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)
> Internet Protocol Version 4, Src: 52.109.44.44, Dst: 192.168.0.3
> Transmission Control Protocol, Src Port: 443, Dst Port: 9102, Seq: 5841, Ack: 195, Len: 59
> [5 Reassembled TCP Segments (5899 bytes): #5440(1460), #5441(1460), #5443(1460), #5444(1460), #5445(59)]
v Transport Layer Security
v TLSv1.2 Record Layer: Handshake Protocol: Multiple Handshake Messages
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 5894
> Handshake Protocol: Server Hello
> Handshake Protocol: Certificate
> Handshake Protocol: Certificate Status
> Handshake Protocol: Server Key Exchange

0020 00 03 01 bb 23 8e d2 c4 fe f9 19 e5 64 8b 50 18 --.#... ..d.P.
0030 40 29 1e bd 00 00 1c 5d d2 6d 5b 15 5f 6d e8 c2 @).....] -m[.m..
0040 e3 bb da 8e 7a 54 c9 73 9e c6 02 62 bb 5a c9 6czT.s ...b.Z.l
0050 1c 34 e7 1a 31 dc ea a5 c2 1e 3b 8f 17 3d 4e 09 -4..1... ;..=N.

Frame (113 bytes) Reassembled TCP (5899 bytes)
Transmission Control Protocol (tcp), 20 byte(s) | Packets: 43054 · Displayed: 4524 (10.5%) | Profile: Default

1. Client hello : My ip request version info, cipher suites, random, compression method to server.
2. Server hello : Server response chipper suites, certificate status, server key exchange to my ip and server hello, end.
3. Client key exchange : after this process, they(server and client) decide the TLS version, cipher suite and end identification each other.
4. Client chipper spec : Send encryption and hash algorithm to server. Notification that encryption will proceed in the manner negotiated so far.
5. Change Cipher Spec : Send algorithm to client and close handshake process .
6. Encrypted communication : Encrypted messages are delivered in an application data

packet. After the message delivery is completed, the server and the client are disconnected after a 4-way handshake process by transmitting a pin flag.

-Summary

Before exchanging data, the TLS protocol certificate each other, and encryption decryption key exchange. Data is exchanged using encryption algorithms. As a result of the investigation after the communication, the SMTP protocol was slightly used, mainly the TLS protocol used. Looking at the TLS protocol version, TLSv1.2 and TLSv1.3 were used. It is said that TLSv1.2 version was old and the vulnerabilities came out, so TLSv1.3 appeared.

★ DNS(Domain Name System)

When we use the Internet, we use domain names that are easy to memorize instead of IP addresses that are difficult to memorize. However, an IP address is actually needed. DNS implements to replace domain names with IP addresses used on the actual network.

```
C:\Users\leeeminjae>nslookup
기본 서버: kns.kornet.net
Address: 168.126.63.1

> www.google.com
서버: kns.kornet.net
Address: 168.126.63.1

권한 없는 응답:
이름: www.google.com
Addresses: 2404:6800:4004:821::2004
172.217.25.100
```

At the terminal, I used the “nslookup” command and typed the Google address that domain name we know. Therefore, it informed me of Google's IP address of 172.217.25.100. And I caught the DNS packets on the wire shark.

No.	Time	Source	Destination	Protocol	Length	Info
7	5.258968	192.168.0.3	168.126.63.1	DNS	85	Standard query 0x0001 PTR 1.63.126
8	5.268739	168.126.63.1	192.168.0.3	DNS	361	Standard query response 0x0001 PTR
19	8.753671	192.168.0.3	168.126.63.1	DNS	74	Standard query 0x0002 A www.google
20	8.763241	168.126.63.1	192.168.0.3	DNS	338	Standard query response 0x0002 A w
21	8.765691	192.168.0.3	168.126.63.1	DNS	74	Standard query 0x0003 AAAA www.goo
22	8.775492	168.126.63.1	192.168.0.3	DNS	350	Standard query response 0x0003 AAA
606	23.796855	192.168.0.3	168.126.63.1	DNS	80	Standard query 0x6cec A cc-api-dat
607	23.822429	168.126.63.1	192.168.0.3	DNS	428	Standard query response 0x6cec A c
653	41.353726	192.168.0.3	168.126.63.1	DNS	75	Standard query 0x7c18 A notifu ad

> Frame 20: 338 bytes on wire (2704 bits), 338 bytes captured (2704 bits) on interface \Device\NPF_{7BEC0FFC-8D98}	
> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)	
> Internet Protocol Version 4, Src: 168.126.63.1, Dst: 192.168.0.3	
> User Datagram Protocol, Src Port: 53, Dst Port: 53550	
▼ Domain Name System (response)	
Transaction ID: 0x0002	
> Flags: 0x8180 Standard query response, No error	
Questions: 1	
Answer RRs: 1	
Authority RRs: 4	
Additional RRs: 8	
> Queries	
> Answers	
> Authoritative nameservers	
> Additional records	
[Request In: 19]	

0030	00 01 00 04 00 08 03 77 77 77 06 67 6f 6f 67 6c	...w ww.googl
0040	65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 01	e.com.....
0050	00 00 00 86 00 04 ac d9 19 64 c0 10 00 02 00 01d.....

-DNS analysis

I taught DNS use both UDP and TCP. However, in my case, most of them were using UDP. Research has shown that UDP is used for general DNS inquiries, and TCP is used for sending packets exceeding 512 bytes with Zone transfer. I used UDP because I had a simple inquiry. Port number was using number 53 as learned in class.

-Transaction ID : This field identifies whether the query sent by the client and the received response match, and uses numbers in the range of 16 bits.

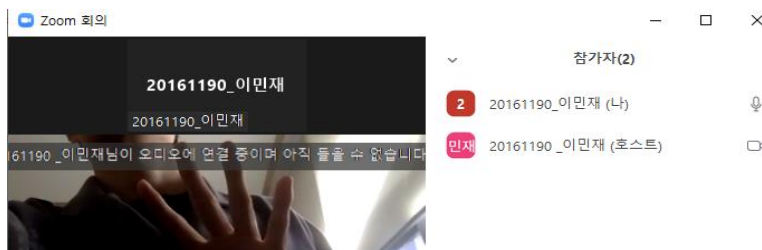
-Flags : This field consists of numerous fields that define the characteristics of the query and has eight flags. QR, Opcode, AA, TC, RD, RA, Z, RCODE

-Queries : Name, type, and class are displayed in this field. I learned about four types in class. My case was type A, name is hostname and value is IP address. And class is IN(General case) mean Internet class.

-Remainder case : Answers, Authoritative nameservers, Additional records are all same format. Name, Type, Class field is same as Queries. And also has TTI, RDLenght, Rdata.

★ Video Conferencing(Zoom)

To analyze video conference, I used the zoom used in our course. I held a zoom meeting on my MacBook and turned on the camera. And I accessed the Zoom meeting that I had already opened by MacBook on my desktop. The process captured packets through wired sharks.



In class, I learned that TCP guarantees high reliability but slow, and UDP does not guarantee 100% reliability but faster than TCP. Therefore, it was speculated that UDP would be used for Zoom in the voice and video areas, where fast data delivery is more important than 100% reliability. And I checked if my thoughts were right.

-UDP Analysis

3.80.20.200	192.168.0.3	TLSv1.2	1514 Server Hello
3.80.20.200	192.168.0.3	TCP	1514 443 → 13684 [ACK] Seq=1461 Ack=518 Win=28672
192.168.0.3	3.80.20.200	TCP	54 13684 → 443 [ACK] Seq=518 Ack=2921 Win=13132
3.80.20.200	192.168.0.3	TLSv1.2	1230 Certificate [TCP segment of a reassembled PD
3.80.20.200	192.168.0.3	TLSv1.2	725 Certificate Status, Server Key Exchange, Ser
192.168.0.3	3.80.20.200	TCP	54 13684 → 443 [ACK] Seq=518 Ack=4768 Win=13132
192.168.0.3	3.80.20.200	TLSv1.2	147 Client Key Exchange, Change Cipher Spec, Enc
3.235.82.212	192.168.0.3	TCP	60 443 → 13685 [ACK] Seq=1 Ack=518 Win=28672 Le
3.235.82.212	192.168.0.3	TLSv1.2	1514 Server Hello
3.235.82.212	192.168.0.3	TCP	1514 443 → 13685 [ACK] Seq=1461 Ack=518 Win=28672
192.168.0.3	3.235.82.212	TCP	54 13685 → 443 [ACK] Seq=518 Ack=2921 Win=13132
3.235.82.212	192.168.0.3	TLSv1.2	1514 Certificate [TCP segment of a reassembled PD
3.235.82.212	192.168.0.3	TLSv1.2	470 Certificate Status, Server Key Exchange, Ser
192.168.0.3	3.235.82.212	TCP	54 13685 → 443 [ACK] Seq=518 Ack=4797 Win=13132
192.168.0.3	3.235.82.212	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Enc
3.80.20.200	192.168.0.3	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Mess
192.168.0.3	3.80.20.200	TLSv1.2	1169 Application Data
3.235.82.212	192.168.0.3	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Mess
192.168.0.3	3.235.82.212	TCP	1514 13685 → 443 [ACK] Seq=644 Ack=4848 Win=13132
192.168.0.3	3.235.82.212	TLSv1.2	193 Application Data

When accessing Zoom for the first time, packet exchange was performed with IP adress 3.800.20200 and 3.235.82.212. Packets were exchanged between TCP and the TLSv12 protocol. As I saw earlier, client and server handshake through TLSv.12. Client hello, Server hello, Client key exchange, Client chipper spec, Change cipher spec... And they exchanged encrypted Application Data.

2319	27.776858	134.224.228.21	192.168.0.3	UDP	292 8801 → 58574 Len=250
2320	27.776925	192.168.0.3	134.224.228.21	UDP	1078 58573 → 8801 Len=1036
2321	27.785884	134.224.228.21	192.168.0.3	UDP	1076 8801 → 58573 Len=1034
2322	27.785948	134.224.228.21	192.168.0.3	UDP	1076 8801 → 58573 Len=1034
2323	27.796002	134.224.228.21	192.168.0.3	UDP	1075 8801 → 58573 Len=1033
2324	27.796064	134.224.228.21	192.168.0.3	UDP	1075 8801 → 58573 Len=1033
2325	27.796137	192.168.0.3	134.224.228.21	UDP	1078 58573 → 8801 Len=1036
2326	27.796164	192.168.0.3	134.224.228.21	UDP	1078 58573 → 8801 Len=1036
2327	27.796179	192.168.0.3	134.224.228.21	UDP	1078 58573 → 8801 Len=1036
2328	27.797130	134.224.228.21	192.168.0.3	UDP	299 8801 → 58574 Len=257

> Frame 2323: 1075 bytes on wire (8600 bits), 1075 bytes captured (8600 bits) on interface \Device\NPF_{...}

> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)

> Internet Protocol Version 4, Src: 134.224.228.21, Dst: 192.168.0.3

> User Datagram Protocol, Src Port: 8801, Dst Port: 58573

Source Port: 8801

Destination Port: 58573

Length: 1041

Checksum: 0xf56f [unverified]

[Checksum Status: Unverified]

[Stream index: 18]

> [Timestamps]

UDP payload (1033 bytes)

0000 70 85 c2 6e 4e 82 70 5d cc cd 36 44 08 00 45 00 p...N.p] ..6D..E.

0010 04 25 0c 38 40 00 22 11 1c ef 86 e0 e4 15 c0 a8 ~%8@.".....

0020 00 03 22 61 e4 cd 04 11 f5 6f 05 01 23 00 06 ae .."a...o...#...

Then I accessed the meeting I left open on my MacBook. Likewise, the server and client went through a handshake process using the TLSv.12 protocol. After that, (192.168.0.3)client and (134.228.21)server exchanged numerous UDP protocols.

If you look at the source port, it is 8801. Port num 1024~49151 is a registered port and 8801 is already registered by zoom. Destination port is 58573 that mean me. Port num 49152 ~ 65535 are dynamically assigned ports. It is dynamically assigned and can change every time when someone access it.

-Summary

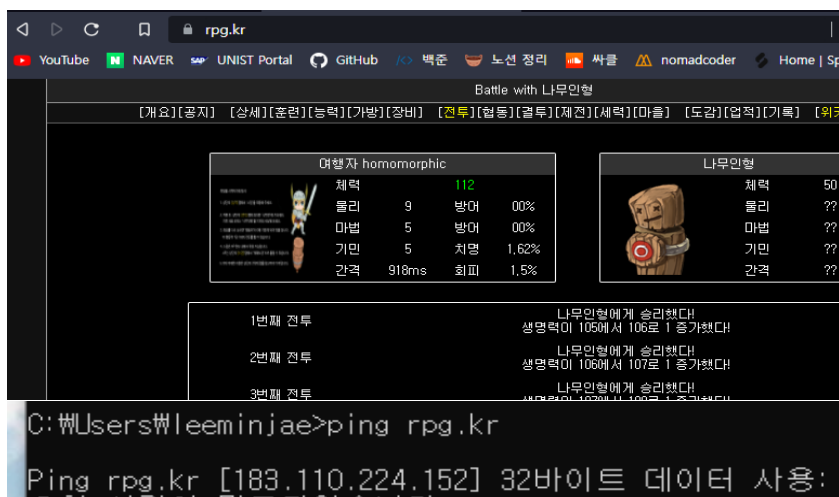
Of course, video streaming does not use UDP in all, When communicate at login or initiall accessing, use TLS and TCP protocol.

As learned in course, after accessing the video, it was communicated through UDP. Because UDP is used, the video may be cut off for a while and information may be lost, but because real-time delivery is important, an immediate video came out again.

I think when designing a network, it is important to select well TCP or UDP depending on the situation.

★ Network Games

I analyzed the RPG.kr network game. It is a nurturing rpg game that allows you to catch monsters or duel with other people. The IP address(183.110.224.152) was found through the “ping” command in cmd, and packets were collected through filtering in wireshark. Based on what I learned in class, I guessed that network games would use TCP.



-TCP Analysis

571	168.480648	192.168.0.3	183.110.224.152	TCP	66	1948 → 443 [SYN] Seq=0 Win=64240 Len=0
573	168.490441	183.110.224.152	192.168.0.3	TCP	66	443 → 1948 [SYN, ACK] Seq=0 Ack=1 Win=2
574	168.490482	192.168.0.3	183.110.224.152	TCP	54	1948 → 443 [ACK] Seq=1 Ack=1 Win=131328
575	168.490703	192.168.0.3	183.110.224.152	TLSv1.2	575	Client Hello
576	168.498909	183.110.224.152	192.168.0.3	TCP	60	443 → 1948 [ACK] Seq=1 Ack=522 Win=3072
577	168.499133	183.110.224.152	192.168.0.3	TLSv1.2	210	Server Hello, Change Cipher Spec, Encry
578	168.499306	192.168.0.3	183.110.224.152	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake
579	168.499419	192.168.0.3	183.110.224.152	TLSv1.2	853	Application Data
580	168.507937	183.110.224.152	192.168.0.3	TCP	60	443 → 1948 [ACK] Seq=157 Ack=1372 Win=3
581	168.513071	183.110.224.152	192.168.0.3	TCP	1514	443 → 1948 [ACK] Seq=157 Ack=1372 Win=3
582	168.513071	183.110.224.152	192.168.0.3	TLSv1.2	1171	Application Data
583	168.513071	183.110.224.152	192.168.0.3	TLSv1.2	88	Application Data

> Frame 581: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{7BEC0FFC-8D98-...}	
> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)	
> Internet Protocol Version 4, Src: 183.110.224.152, Dst: 192.168.0.3	
▼ Transmission Control Protocol, Src Port: 443, Dst Port: 1948, Seq: 157, Ack: 1372, Len: 1460	
Source Port: 443	
Destination Port: 1948	
[Stream index: 28]	
[TCP Segment Len: 1460]	
Sequence Number: 157 (relative sequence number)	

0020	00 03 01 bb 07 9c 52 7a a4 a9 a3 55 d6 a5 50 10Rz...U..P..
------	---	------------------

As a result of checking through the wire shark, TCP was actually used in this network game. As in the previous analysis, the client and server go through a handshake operation with TLSv.12.

There are six TCP flags, which are initially synchronized through the SYN flag. During the next game, most of them send a packet related to the response to the ACK flag. And the connection termination request is made through the FIN flag and the connection is disconnected to the RST flag.

-Summary

Network games use TCP. If UDP is used, game user information or item capabilities, etc. could be lost. I confirmed that TCP is being used because reliability is particularly important in games.

As a result of further research, real-time streaming game may also consider using UDP in games. Can't we use both TCP for important information and UDP if we don't? However, it is said that using UDP and TCP together is too complicated and can causes certain losses. So these days some games, it is said that methods of implementing TCP characteristics with UDP(reliable UDP) may also be used.

★ Video streaming (Youtube)

I used Youtube to analyze video streaming. I connected to Wire Shark and listened to music on YouTube. Then, very many QUIC packets were caught. I searched thinking this protocol might be related to YouTube. As a result of checking in DNS, the packets used in the ip address related to youtube were QUIC.

-QUIC analysis

8	0.993375	192.168.0.3	3.237.55.202	TCP	54 [TCP ACKed unseen segment]
9	4.733285	192.168.0.3	58.123.102.50	QUIC	1392 Initial, DCID=e2ee64491670a88c
10	4.733620	192.168.0.3	58.123.102.50	QUIC	119 0-RTT, DCID=e2ee64491670a88c
11	4.733991	192.168.0.3	58.123.102.50	QUIC	1090 0-RTT, DCID=e2ee64491670a88c
12	4.745427	58.123.102.50	192.168.0.3	QUIC	1392 Handshake, SCID=e2ee64491670a88c
13	4.745524	58.123.102.50	192.168.0.3	QUIC	1392 Handshake, SCID=e2ee64491670a88c
14	4.745585	58.123.102.50	192.168.0.3	QUIC	1392 Handshake, SCID=e2ee64491670a88c
15	4.745711	58.123.102.50	192.168.0.3	QUIC	1317 Protected Payload (KP0)
16	4.746069	192.168.0.3	58.123.102.50	QUIC	83 Handshake, DCID=e2ee64491670a88c
17	4.746693	192.168.0.3	58.123.102.50	QUIC	1215 Protected Payload (KP0)

>	Internet Protocol Version 4, Src: 58.123.102.50, Dst: 192.168.0.3
>	User Datagram Protocol, Src Port: 443, Dst Port: 59156
>	QUIC IETF
>	QUIC Connection information
	[Packet Length: 1350]
	1... = Header Form: Long Header (1)
	.1.. = Fixed Bit: True
	..10 = Packet Type: Handshake (2)
	Version: 1 (0x00000001)
	Destination Connection ID Length: 0
	Source Connection ID Length: 8
	Source Connection ID: e2ee64491670a88c
	Length: 1333
>	[Expert Info (Warning/Decryption): Failed to create decryption context: Secrets are not available]

0030	08 e2 ee 64 49 16 70 a8 8c 45 35 78 3b 8a 20 42	..dI.p..E5x;- B
0040	30 d5 9f a8 39 e4 dd 7a ad d8 50 2f f0 1c f6 48	0...9...z ..P/...H
0050	d9 ef a6 d6 18 70 21 f6 38 e5 3d 5d 50 22 ae 56p!.. 8-=lP"-V

Looking at the QUIC IETF section, there are various information that we have looked at above. In particular, there is a phrase called “Failed to create content: Secrets are not available: and I wondered if encryption was used.

Searching at it, QUIC was a protocol encapsulated with UDP and was security equivalent to TLS. It was also a protocol created by Google to reduce latency that occurs due to the nature of TCP. Since QUIC operates on UDP, the theoretically RTT value is 0.(In reality, it is not zero.) Therefore, it is said that the speed is fast because client hello, server hello, certificate, cipher spec... process is omitted.

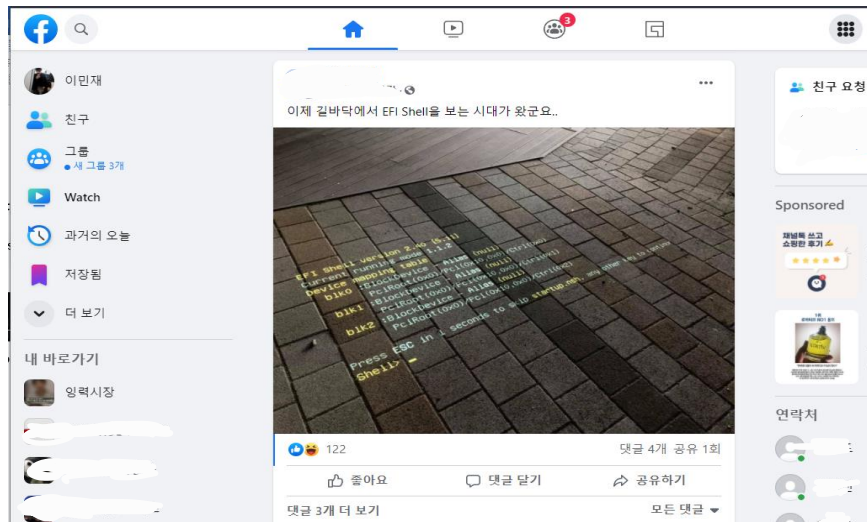
To confirm this, I compared speeds using an Explorer browser using TCP. I compared connecting to YouTube using Chrome vs YouTube using Explorer. As a result, it was shown that QUIC is 6 to 10 times faster than TCP.

-Summary

In network class, I learned that UDP has advantages in fast-deafers because it is not affected by TCP's cohesion control. So, I learned that even if I throw away a little bit of the reality of data, it is used for streaming services where speed is important. In fact, YouTube delivered information at a high speed using the QUIC protocol operating on UDP.

★ Social Networking (Facebook)

I selected Facebook, one of the social networks, as the last analysis. Because Facebook seemed to use various protocols because there were many types of data such as photos, videos, text, and messages.



```
C:\Users\leeminjae>ping www.facebook.com

Ping star-mini.c10r.facebook.com [157.240.215.35]
157.240.215.35의 응답: 바이트=32 시간=9ms TTL=49
```

As always, I checked Facebook's ip using the “ping” command and started analyzing it.

-Social network analysis

53	1.192854	157.240.215.35	192.168.0.3	UDP	67 443 → 57019 Len=25
54	1.260960	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232
55	1.261005	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232
56	1.261065	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232
57	1.261170	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232
58	1.261180	192.168.0.3	157.240.215.35	UDP	83 57019 → 443 Len=41
59	1.261276	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232
60	1.261718	157.240.215.35	192.168.0.3	UDP	1274 443 → 57019 Len=1232

> Frame 56: 1274 bytes on wire (10192 bits), 1274 bytes captured (10192 bits) on interface \Device\NPF_{7044 57.493267 31.13.76.8} 192.168.0.3					
> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)					
> Internet Protocol Version 4, Src: 157.240.215.35, Dst: 192.168.0.3					
▼ User Datagram Protocol, Src Port: 443, Dst Port: 57019					
7844	57.493267	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7845	57.493373	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7846	57.493487	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7847	57.493583	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7848	57.493664	192.168.0.3	31.13.76.8	QUIC	77 Protected Payload (KP0)
7849	57.493685	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7850	57.493790	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7851	57.493896	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7852	57.494000	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7853	57.494112	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7854	57.494204	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)
7855	57.494322	31.13.76.8	192.168.0.3	QUIC	1274 Protected Payload (KP0)

> Frame 7847: 1274 bytes on wire (10192 bits), 1274 bytes captured (10192 bits) on interface \Device\NPF_{7044 57.493267 31.13.76.8} 192.168.0.3					
> Ethernet II, Src: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44), Dst: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82)					
> Internet Protocol Version 4, Src: 31.13.76.8, Dst: 192.168.0.3					
▼ User Datagram Protocol, Src Port: 443, Dst Port: 60888					

5616	38.696607	157.240.215.9	192.168.0.3	TLSv1.2	94 Application Data
5617	38.696607	157.240.215.9	192.168.0.3	TLSv1.2	128 Application Data
5618	38.696676	192.168.0.3	157.240.215.9	TCP	54 6790 → 443 [ACK] Seq=54526
5619	38.697037	157.240.215.9	192.168.0.3	TCP	60 443 → 6790 [ACK] Seq=12524
5620	38.697627	192.168.0.3	157.240.215.9	TLSv1.2	88 Application Data
5621	38.698113	192.168.0.3	157.240.215.9	TLSv1.2	88 Application Data
5622	38.698557	157.240.215.9	192.168.0.3	TLSv1.2	94 Application Data
5623	38.698557	157.240.215.9	192.168.0.3	TLSv1.2	128 Application Data
5624	38.698676	192.168.0.3	157.240.215.9	TCP	54 6790 → 443 [ACK] Seq=54526

Frame 5618: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{78EC0FFC-Ethernet II, Src: ASRockIn_6e:4e:82 (70:85:c2:6e:4e:82), Dst: EFMNetwo_cd:36:44 (70:5d:cc:cd:36:44)
 Internet Protocol Version 4, Src: 192.168.0.3, Dst: 157.240.215.9
 Transmission Control Protocol, Src Port: 6790, Dst Port: 443, Seq: 54526, Ack: 12524, Len: 0

On Facebook, different protocols were used depending on the situation. UDP was used to scroll down and read posts. It seemed to be because I had to quickly bring in new posts in real time.

While scrolling down, the QUIC protocol was used when the video was played. As QUIC was used in video streaming, it seems to perform particularly well in videos. The characteristic of QUIC is fast and support encryption so it is a next-generation protocol. According to the survey, after Facebook introduced QUIC, which greatly reduced video request errors and buffering.

Also, I tried chatting with my friend on Facebook. In this case, TCP and TLSv1.2 were used. This is because in the case of chatting, high reliability is more important than speed.

-Summary

The above three protocols were mainly used, and various protocols such as DNS, TLSv1.3, ARP, ICMPv6, and IGMPv2 were also used. Facebook uses different protocols depending on the situation because there are various data forms.