Stripping IP Packets

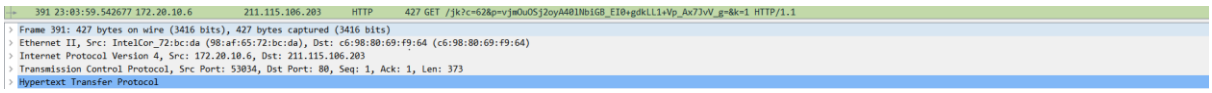
수학과 2016314786 김호진

[Name of applications]

(1) Applications running over TCP

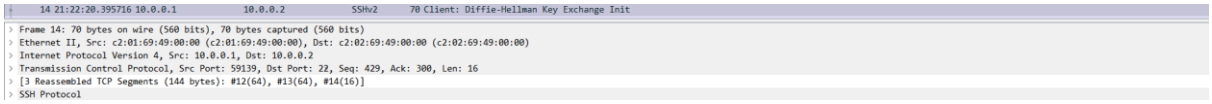
1. TLS (Transport Layer Security)

2. HTTP (Hypertext Transfer Protocol)

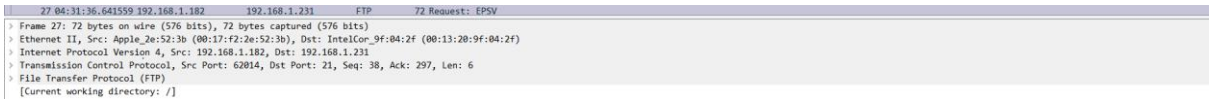


3. SMTP (Simple Mail Transfer Protocol)

4. SSH (Secure Shell)

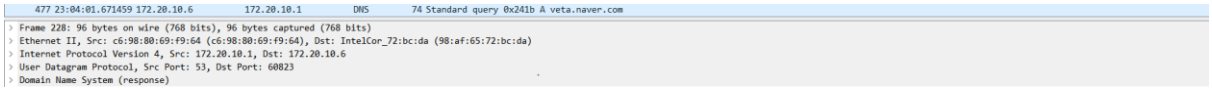


5. FTP (File Transfer Protocol)

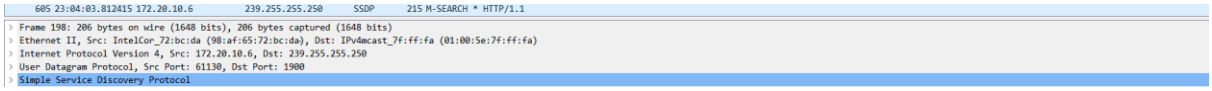


(2) Applications running over UDP

1. DNS (Domain Name System)



2. SSDP (Simple Service Discovery Protocol)



[Source code]

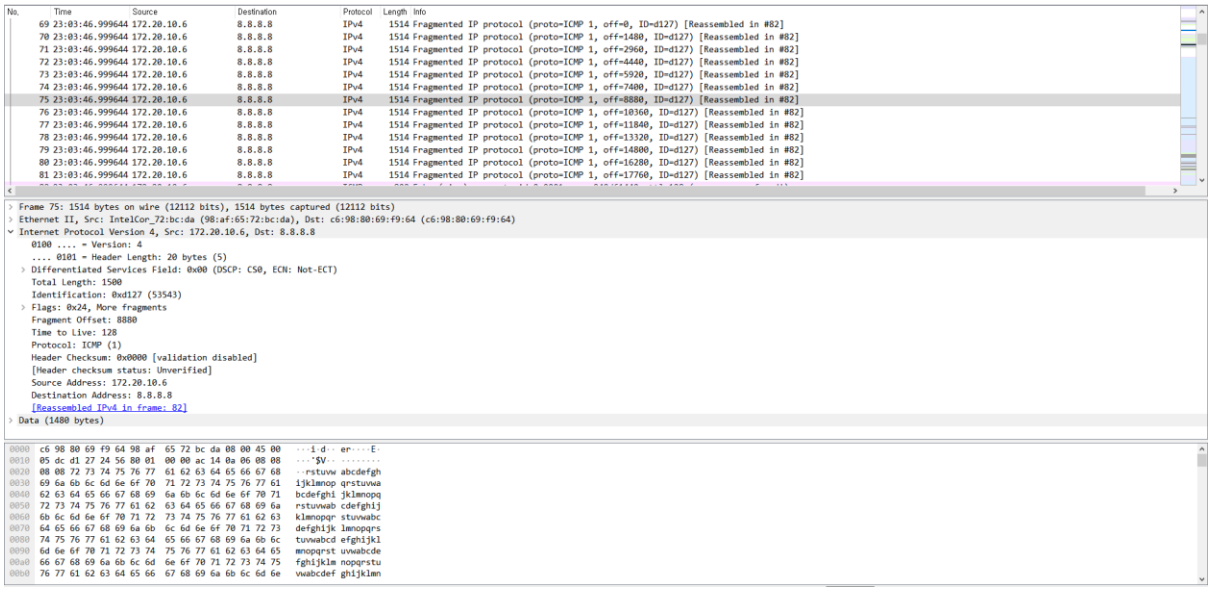
```c
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <time.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

// #include <WinSock2.h>
// #pragma comment(lib,"ws2_32")

typedef struct _Pcap_File_Header {
        unsigned int magic;
        unsigned short major;
        unsigned short minor;
        unsigned int timezone;
        unsigned timestamp;
        unsigned snap_len;
        unsigned linktype;
}PFHeader;

typedef struct _Packet_Header {
        unsigned int sec;
        unsigned int usec;
        unsigned int capture_len;
        unsigned int packet_len;
}PHeader;

typedef struct _Ethernet_Header {
        unsigned char dst_mac[6];
        unsigned char src_mac[6];
        unsigned short type;
}Ethernet_Header;

typedef struct _IP_Header {
        unsigned char header_len : 4;
        unsigned char version : 4;
        unsigned char service_type;
        unsigned short total_len;
        unsigned short identification;
        unsigned short fragmentation;
        unsigned char time_to_live;
        unsigned char protocol;
        unsigned short header_checksum;
        unsigned int src_addr;
        unsigned int dst_addr;
}IP_Header;

int parsePacket(FILE *fp);
void parseEthernet(char *buffer);
void viewMAC(unsigned char *mac);
```

- C Programming으로 작성했으며, Ubuntu 20.04 LTS 상에서 컴파일을 진행했습니다.
- Linux System이 아닌 Windows 상에서 컴파일을 진행할 경우, 주석처리 된 <Winsock2.h>를 추가해주시고 상단에 위치한 다섯 종류의 헤더들을 주석처리 해주시면 됩니다.
- 코드 부분을 더블 클릭하시면, 전체 소스코드를 확인하실 수 있습니다.

[Screenshots of output for sample packets and Verification]

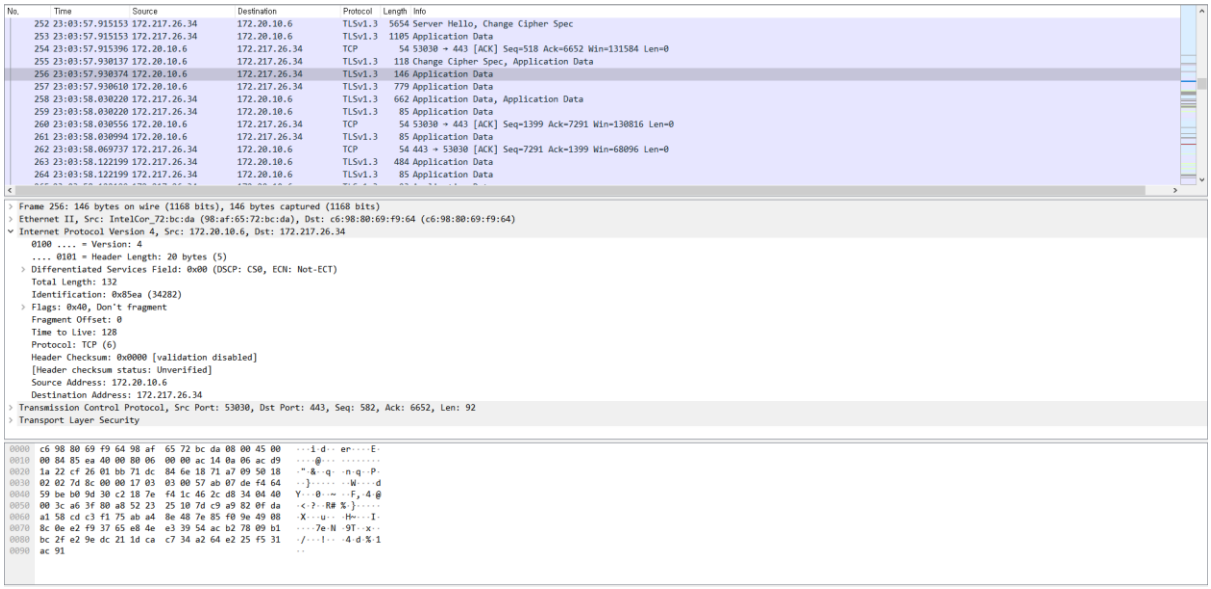
- **Frame 75**

```
================================================================================

[Frame 75] (Local time - 23:03:46.999644)
1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits).
Source MAC address: (98:af:65:72:bc:da) -> Destination MAC address: (c6:98:80:69:f9:64)
Source IP address: 172.20.10.6 -> Destination IP address: 8.8.8.8
IP Header Length: 20 bytes
IP Total Length: 1500
Identfication: 53543
Flags: More fragments
Fragment-offset: 1110 (8880)
Time to Live: 128
Protocol: ICMP (1)

================================================================================
```



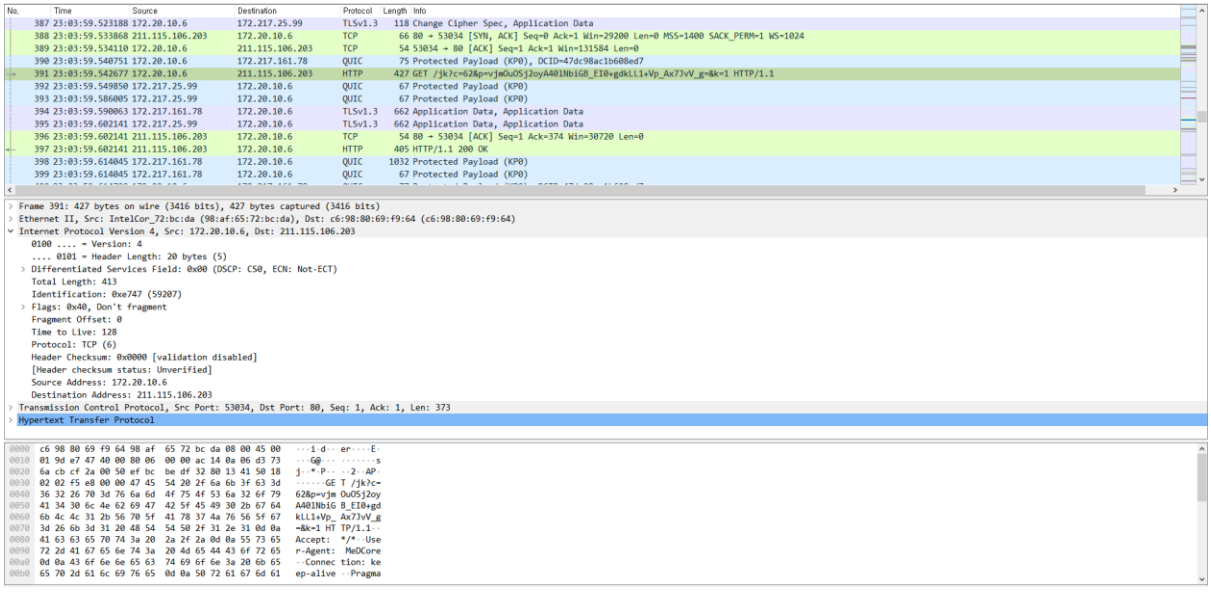
- **Frame 256**

```
================================================================================

[Frame 256] (Local time - 23:03:57.930374)
146 bytes on wire (1168 bits), 146 bytes captured (1168 bits).
Source MAC address: (98:af:65:72:bc:da) -> Destination MAC address: (c6:98:80:69:f9:64)
Source IP address: 172.20.10.6 -> Destination IP address: 172.217.26.34
IP Header Length: 20 bytes
IP Total Length: 132
Identfication: 34282
Flags: Don't fragment
Fragment-offset: 0 (0)
Time to Live: 128
Protocol: TCP (6)

================================================================================
```


No. Time Source Destination Protocol Length Info
252 23:03:57.915153 172.217.26.34 172.20.10.6 TLSv1.3 5654 Server Hello, Change Cipher Spec
253 23:03:57.915153 172.217.26.34 172.20.10.6 TLSv1.3 1105 Application Data
254 23:03:57.915396 172.20.10.6 172.217.26.34 TCP 54 53030 → 443 [ACK] Seq=518 Ack=6652 Win=131584 Len=0
255 23:03:57.930137 172.20.10.6 172.217.26.34 TLSv1.3 118 Change Cipher Spec, Application Data
256 23:03:57.930374 172.20.10.6 172.217.26.34 TLSv1.3 146 Application Data
257 23:03:57.930610 172.20.10.6 172.217.26.34 TLSv1.3 779 Application Data
258 23:03:58.030220 172.217.26.34 172.20.10.6 TLSv1.3 662 Application Data, Application Data
259 23:03:58.030220 172.217.26.34 172.20.10.6 TLSv1.3 85 Application Data
260 23:03:58.030556 172.20.10.6 172.217.26.34 TCP 54 53030 → 443 [ACK] Seq=1399 Ack=7291 Win=130816 Len=0
261 23:03:58.030994 172.20.10.6 172.217.26.34 TLSv1.3 85 Application Data
262 23:03:58.069737 172.217.26.34 172.20.10.6 TCP 54 443 → 53030 [ACK] Seq=7291 Ack=1399 Win=68096 Len=0
263 23:03:58.122199 172.217.26.34 172.20.10.6 TLSv1.3 484 Application Data
264 23:03:58.122199 172.217.26.34 172.20.10.6 TLSv1.3 85 Application Data
Frame 256: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits)
Ethernet II, Src: IntelCor_72:bc:da (98:af:65:72:bc:da), Dst: c6:98:80:69:f9:64 (c6:98:80:69:f9:64)
Internet Protocol Version 4, Src: 172.20.10.6, Dst: 172.217.26.34
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 132
Identification: 0x85ea (34282)
Flags: 0x40, Don't fragment
Fragment Offset: 0
Time to Live: 128
Protocol: TCP (6)
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.10.6
Destination Address: 172.217.26.34
Transmission Control Protocol, Src Port: 53030, Dst Port: 443, Seq: 582, Ack: 6652, Len: 92
Transport Layer Security
0000 c6 98 80 69 f9 64 98 af 65 72 bc da 08 00 45 00
0010 00 84 85 ea 40 00 80 06 00 00 ac 14 0a 06 ac d9
0020 1a 22 cf 26 01 bb 71 dc 84 6e 18 71 a7 09 50 18
0030 02 02 7d 8c 00 00 17 03 03 00 57 ab 07 de f4 64
0040 59 be b0 9d 30 c2 18 7e f4 1c 46 2c d8 34 04 40
0050 00 3c a6 3f 80 a8 52 23 25 10 7d c9 a9 82 0f da
0060 a1 58 cd c3 f1 75 ab a4 8e 48 7e 85 f0 9e 49 08
0070 8c 0e e2 f9 37 65 e8 4e e3 39 54 ac b2 78 09 b1
0080 bc 2f e2 9e dc 21 1d ca c7 34 a2 64 e2 25 f5 31
0090 ac 91

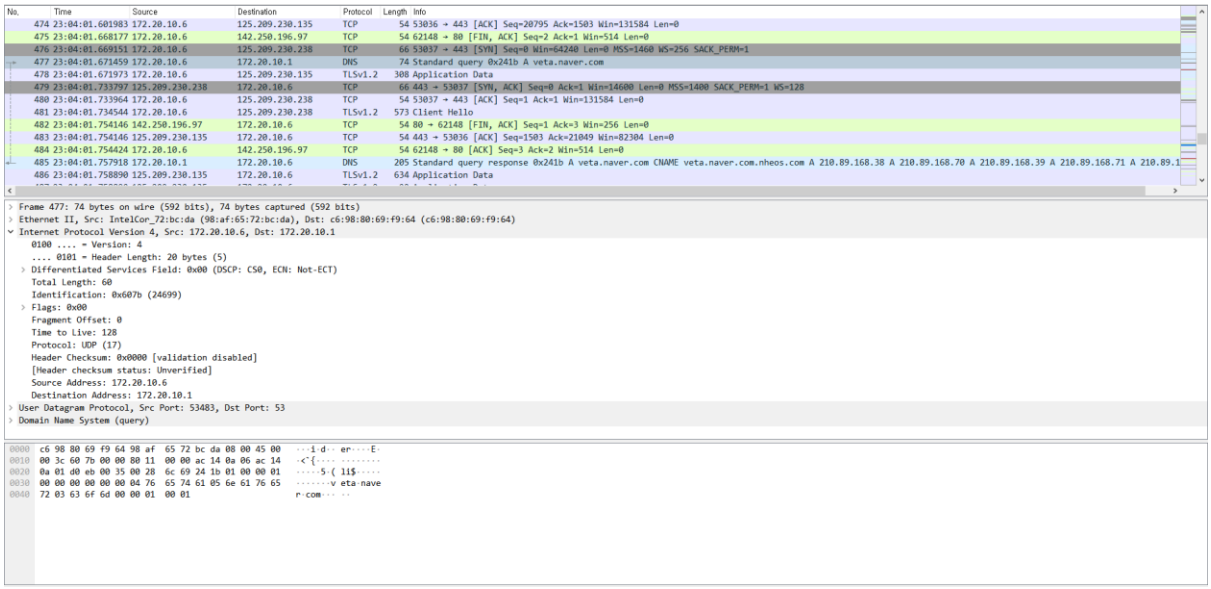
- **Frame 391**


[Frame 391] (Local time - 23:03:59.542677)
427 bytes on wire (3416 bits), 427 bytes captured (3416 bits).
Source MAC address: (98:af:65:72:bc:da) -> Destination MAC address: (c6:98:80:69:f9:64)
Source IP address: 172.20.10.6 -> Destination IP address: 211.115.106.203
IP Header Length: 20 bytes
IP Total Length: 413
Identfication: 59207
Flags: Don't fragment
Fragment-offset: 0 (0)
Time to Live: 128
Protocol: TCP (6)


No. Time Source Destination Protocol Length Info
387 23:03:59.523188 172.20.10.6 172.217.25.99 TLSv1.3 118 Change Cipher Spec, Application Data
388 23:03:59.533868 211.115.106.203 172.20.10.6 TCP 66 80 → 53034 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1400 SACK_PERM=1 WS=1024
389 23:03:59.534110 172.20.10.6 211.115.106.203 TCP 54 53034 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0
390 23:03:59.540751 172.20.10.6 172.217.161.78 QUIC 75 Protected Payload (KP0), DCID=47dc98ac1b608ed7
391 23:03:59.542677 172.20.10.6 211.115.106.203 HTTP 427 GET /jk?c=62&p=vjmOuOSj2oyA401NbiGB_EI0+gdkLL1+Vp_Ax7JvV_g=&k=1 HTTP/1.1
392 23:03:59.549850 172.217.25.99 172.20.10.6 QUIC 67 Protected Payload (KP0)
393 23:03:59.586005 172.217.25.99 172.20.10.6 QUIC 67 Protected Payload (KP0)
394 23:03:59.590063 172.217.161.78 172.20.10.6 TLSv1.3 662 Application Data, Application Data
395 23:03:59.602141 172.217.25.99 172.20.10.6 TLSv1.3 662 Application Data, Application Data
396 23:03:59.602141 211.115.106.203 172.20.10.6 TCP 54 80 → 53034 [ACK] Seq=1 Ack=374 Win=30720 Len=0
397 23:03:59.602141 211.115.106.203 172.20.10.6 HTTP 405 HTTP/1.1 200 OK
398 23:03:59.614045 172.217.161.78 172.20.10.6 QUIC 1032 Protected Payload (KP0)
399 23:03:59.614045 172.217.161.78 172.20.10.6 QUIC 67 Protected Payload (KP0)
Frame 391: 427 bytes on wire (3416 bits), 427 bytes captured (3416 bits)
Ethernet II, Src: IntelCor_72:bc:da (98:af:65:72:bc:da), Dst: c6:98:80:69:f9:64 (c6:98:80:69:f9:64)
Internet Protocol Version 4, Src: 172.20.10.6, Dst: 211.115.106.203
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 413
Identification: 0xe747 (59207)
Flags: 0x40, Don't fragment
Fragment Offset: 0
Time to Live: 128
Protocol: TCP (6)
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.10.6
Destination Address: 211.115.106.203
Transmission Control Protocol, Src Port: 53034, Dst Port: 80, Seq: 1, Ack: 1, Len: 373
Hypertext Transfer Protocol
0000 c6 98 80 69 f9 64 98 af 65 72 bc da 08 00 45 00
0010 01 9d e7 47 40 00 80 06 00 00 ac 14 0a 06 d3 73
0020 6a cb cf 2a 00 50 ef bc be df 32 80 13 41 50 18
0030 02 02 f5 e8 00 00 47 45 54 20 2f 6a 6b 3f 63 3d
0040 36 32 26 70 3d 76 6a 6d 4f 75 4f 53 6a 32 6f 79
0050 41 34 30 6c 4e 62 69 47 42 5f 45 49 30 2b 67 64
0060 6b 4c 4c 31 2b 56 70 5f 41 78 37 4a 76 56 5f 67
0070 3d 26 6b 3d 31 20 48 54 54 50 2f 31 2e 31 0d 0a
0080 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 55 73 65
0090 72 2d 41 67 65 6e 74 3a 20 4d 65 4d 43 6f 72 65
00a0 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65
00b0 65 70 2d 61 6c 69 76 65 0d 0a 50 72 61 67 6d 61

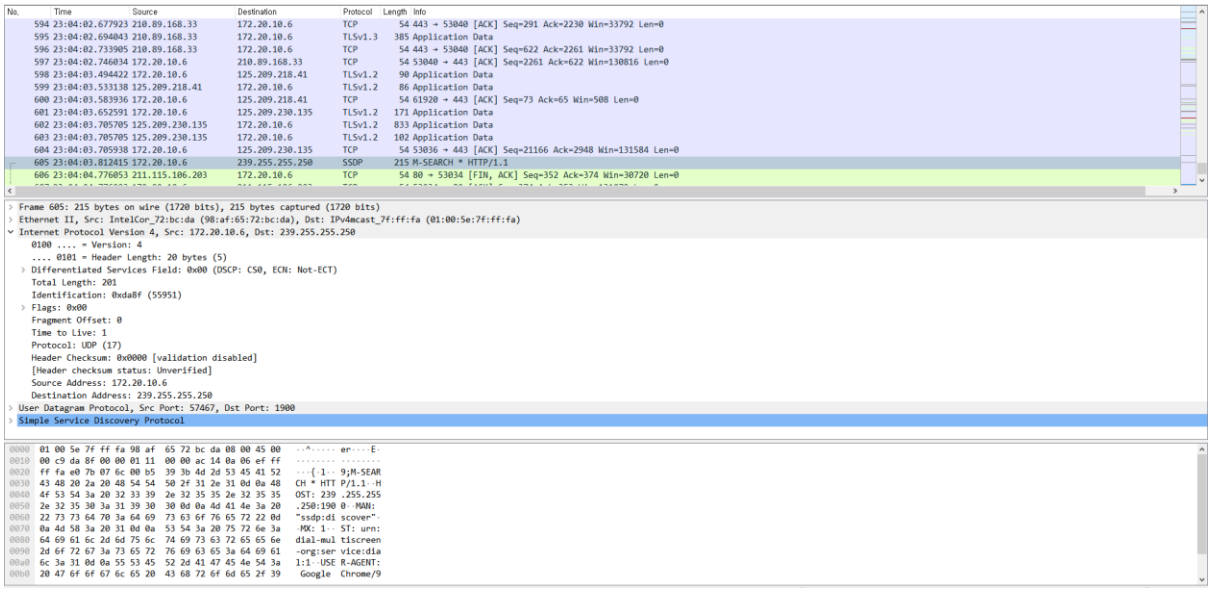
- **Frame 477**

```
================================================================================================

[Frame 477] (Local time - 23:04:01.671459)
74 bytes on wire (592 bits), 74 bytes captured (592 bits).
Source MAC address: (98:af:65:72:bc:da) -> Destination MAC address: (c6:98:80:69:f9:64)
Source IP address: 172.20.10.6 -> Destination IP address: 172.20.10.1
IP Header Length: 20 bytes
IP Total Length: 60
Identfication: 24699
Flags: DF & MF are not set
Fragment-offset: 0 (0)
Time to Live: 128
Protocol: UDP (17)

================================================================================================
```



- **Frame 605**

```
================================================================================================

[Frame 605] (Local time - 23:04:03.812415)
215 bytes on wire (1720 bits), 215 bytes captured (1720 bits).
Source MAC address: (98:af:65:72:bc:da) -> Destination MAC address: (01:00:5e:7f:ff:fa)
Source IP address: 172.20.10.6 -> Destination IP address: 239.255.255.250
IP Header Length: 20 bytes
IP Total Length: 201
Identfication: 55951
Flags: DF & MF are not set
Fragment-offset: 0 (0)
Time to Live: 1
Protocol: UDP (17)

================================================================================================
```



최대한 다양한 경우를 비교해 보기 위해 Protocol 종류, Application의 종류, Flags의 상태 등을 고려하여 위의 다섯 개의 패킷들을 샘플로 정했습니다. 소스코드의 실행 결과와 Wireshark 분석 결과를 비교해 보았고, 값이 모두 일치하는 것을 확인하였습니다.

[Discussion of unique experience]

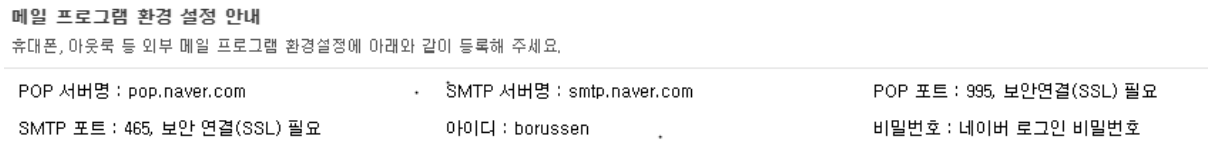
1. 과제를 진행하기 위해 많은 패킷들을 발생시켰지만, Fragmented packets을 발견하기 어려웠습니다. 그래서 명령 프롬프트를 통해 "ping 8.8.8.8 -l 20000" 명령어를 입력하여 인위적으로 Fragmented packets을 확인하였습니다. 저는 IP Fragmentation이 발생하지 않는 이유가 강의 시간 중 배웠던 내용과 연관이 있다고 생각했습니다. 지난 강의에서 Path MTU를 4계층(TCP, UDP)와 공유하면, IP를 fragment 할 필요가 없어져 Performance 측면에서 더욱 효율적이라는 내용을 배웠습니다. 이에 대해 추가적으로 찾아보니 TCP의 경우 PMTUD과정으로 최적의 MSS를 가지는 네트워크를 탐색하고 서버와 클라이언트가 서로 MSS를 교환하면서 Fragmentation의 발생을 방지하고 있으며, UDP의 경우에는 MSS를 알 수 없기 때문에 Minimum Reassembly Buffer Size보다 작은 크기의 Datagram으로 통신을 하여 Fragmentation을 피한다는 사실을 알 수 있었습니다. 결론적으로, 패킷 분석을 통해 IP Fragmentation이 Performance 측면에서 부정적인 영향을 주기 때문에 최대한 지양되고 있음을 확인할 수 있었습니다.

추가적으로, Fragmented Packet (Frame 75)의 Local time을 분석해면서 같은 Identification을 가지는 Fragmented Packets (Frame 69~81)은 순차적으로 전달되는 것이 아니라 동시에 전달된다는 것을 알 수 있었습니다. 또한, 이러한 Fragmented Packets은 각각 independent하기 때문에 Reassemble하기 위해 목적지를 명시해 주는 것(Reassembled in #82)을 확인할 수 있었습니다.

2. 패킷 분석을 하면서 가장 많이 볼 수 있는 Application 중 하나인 TLS (Transport Layer Security)에 대해 궁금한 점이 생겼습니다. TLS는 TLSv1.2와 TLSv1.3, 두 가지 버전이 사용되고 있는데 서로 간의 차이점을 알고 싶었습니다. 이에 대해 찾아본 결과, TLSv1.2의 경우 많은 웹 브라우저에서 사용되고 있지만 키 교환 알고리즘에 사용되는 RSA와 SRP, 인증 알고리즘에 사용되는 DSA 등과 같이 프로토콜의 특정 부분에 취약성이 존재하는 알고리즘이 포함되어 있는 문제가 있다는 것을 알게 되었습니다. TLSv1.3에서는 이러한 취약성이 존재하는 알고리즘을 제거하였고, 트래픽 분석을 방해하기 위해 데이터 교환에 대한 개인 정보를 추가하여 Handshake 과정을 암호화하였습니다. 간단히 정리하자면, TLSv1.3은 TLSv1.2에 비해 연결 속도를 향상시켰으며 보안강화 등 일부 성능에 있어서도 개선된 기능을 제공합니다. 그렇기 때문에 시간이 지날수록 점점 더 많은 웹 브라우저에서 TLSv1.3을 사용하게 될 것이라고 예상합니다.

3. TCP 기반의 다양한 Applications을 보고자 했지만, TLS(TLSv1.2, TLSv1.3)와 HTTP 이외의 것들을 찾아내기가 어려웠습니다. 그래서 외부에서 SMTP, SSHv2, FTP 패킷이 들어 있는 pcap file을 받아 분석해보았고, 이들이 어떤 상황에서 사용되는지를 알아봤습니다.

SMTP(Simple Mail Transfer Protocol)는 전자 우편을 송신하고 수신하는데 사용되는 TCP/IP 프로토콜로, 일반적으로 POP3 또는 IMAP(Internet Message Access Protocol)와 함께 사용되어 메시지를 서버 메일함에 저장하고 사용자를 위해 서버에서 주기적으로 메시지를 다운로드하는 역할을 합니다. 그런데 실제로는 SMTP는 TLS/SSL과 함께 사용된다고 합니다. 그 이유는 SMTP는 암호화를 제공하지 않기 때문에 해킹의 위험이 높기 때문입니다. 그래서 SMTP에 암호화를 제공하기 위해 TLS 또는 SSL이 함께 사용되고 있는 것입니다. 실제로 네이버 메일에서 POP/SMTP 설정을 들어가보니 다음을 확인할 수 있었습니다.



SSH(Secure SHell)은 네트워크 상의 다른 컴퓨터에 로그인하거나 원격 시스템에서 명령을 실행하고 다른 시스템으로 파일을 복사할 수 있도록 해주는 프로토콜을 말합니다. 기존의 Telnet을 대체하기 위해 설계되었으며, 강력한 인증 방법 및 안전하지 못한 네트워크에서 안전하게 통신을 할 수 있는 기능을 제공합니다. 마지막으로 FTP(File Transfer Protocol)은 서버와 클라이언트 사이의 파일 전송을 하기 위해 만들어진 TCP/IP 프로토콜입니다.

4. 패킷들을 분석하기 위한 프로그램을 만들기 위해 Pcap file Header, Packet Header, Ethernet Header 그리고 IPv4 Header의 구조를 차례로 확인해보고, 이를 프로그램 상에 구현하면서 명확하게 이해할 수 있게 되었습니다. Intel X86은 Little Endian을 사용하는 반면, 네트워크 바이트 순서는 Big Endian만을 사용하기 때문에 이를 고려하는데 약간의 어려움을 겪기는 했지만 패킷들을 분석해보면서 지금까지 배웠던 내용들을 확인해 보고 정리할 수 있는 기회가 되었습니다.