Stripping TCP and UDP Packets

수학과 2016314786 김호진

[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")

int TCP = 0;
int UDP = 0;

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;
```

- C Programming으로 작성했으며, Ubuntu 20.04 LTS 상에서 컴파일을 진행했습니다.
- Linux System이 아닌 Windows에서 컴파일을 진행할 경우, 주석처리 된 두 종류의 헤더를 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]

최대한 다양한 경우를 비교해 보기 위해 Transport layer의 Protocol 종류 (TCP or UDP), Application type의 다양성, Control field의 상태, Payload의 유무, TCP Option, fragmented packet 등을 고려하여 다음 여섯 개의 패킷들을 샘플로 정했습니다. 소스코드의 실행 결과와 Wireshark 분석 결과를 비교해 보았고, 과제에서 요구하는 값들이 모두 일치하는 것을 확인하였습니다.

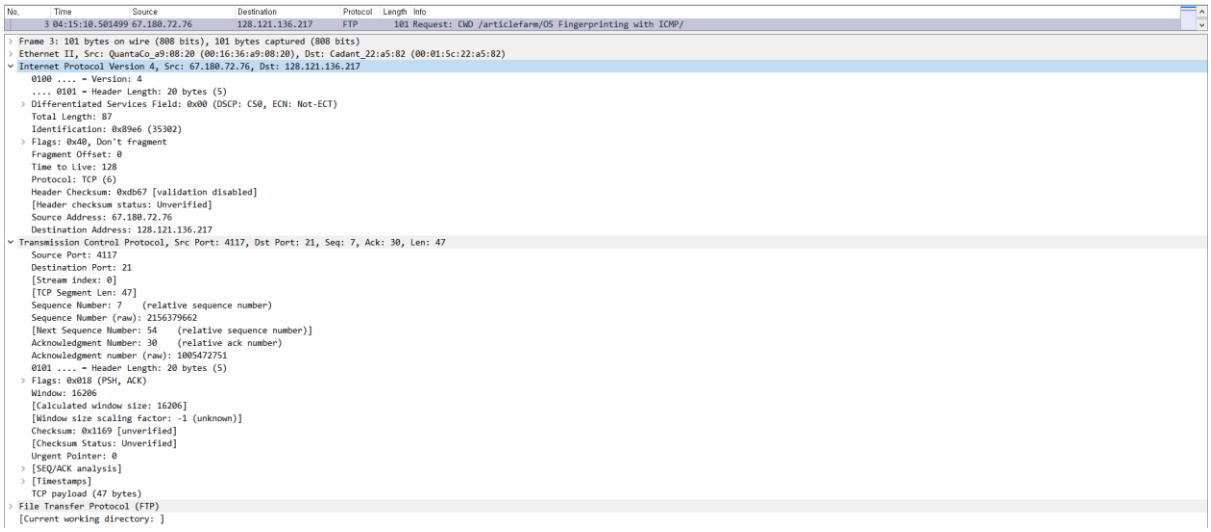
- **Frame 3 (TCP / Application type - FTP)**

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

[Frame 3] Local time - 04:15:10.501499
101 bytes on wire (808 bits), 101 bytes captured (808 bits)
Total Length: 87 bytes / IP Header Length: 20 bytes (5)

Protocol: TCP (6)
Source Port: 4117
Destination Port: 21
Sequence Number (raw): 2156379662
(Starting sequence number: 2156379662 / Ending Sequence number: 2156379708)
Acknowledgment Number (raw): 1005472751
Flags: PSH ACK
Window: 16206
Urgent Pointer: 0
TCP payload (47 bytes)
Application type: FTP

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



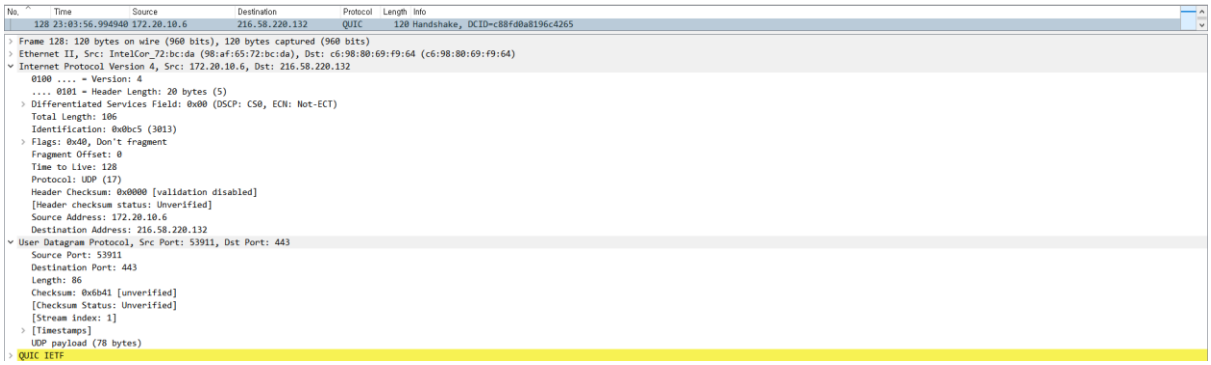
- **Frame 128 (UDP / Application type - QUIC)**

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

[Frame 128] Local time - 23:03:56.994940
120 bytes on wire (960 bits), 120 bytes captured (960 bits)
Total Length: 106 bytes / IP Header Length: 20 bytes (5)

Protocol: UDP (17)
Source Port: 53911
Destination Port: 443
UDP payload (78 bytes)
Application type: QUIC

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



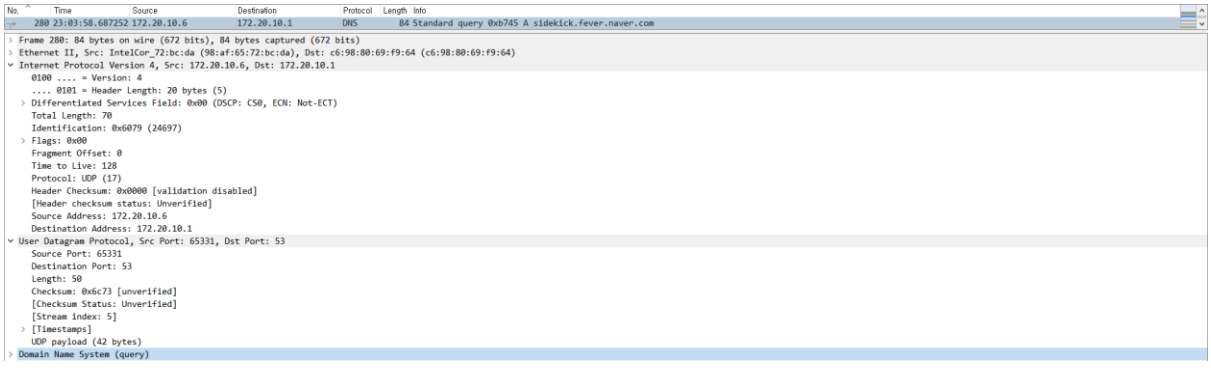
- **Frame 280 (UDP / Application type – DNS)**

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

[Frame 280] Local time - 23:03:58.687252
84 bytes on wire (672 bits), 84 bytes captured (672 bits)
Total Length: 70 bytes / IP Header Length: 20 bytes (5)

Protocol: UDP (17)
Source Port: 65331
Destination Port: 53
UDP payload (42 bytes)
Application type: DNS

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



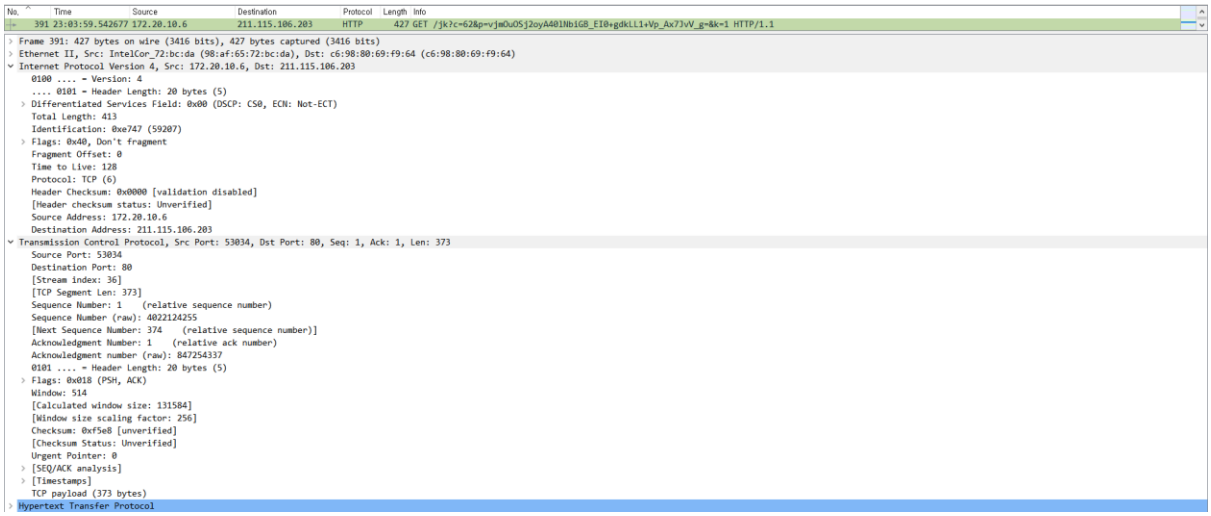
- **Frame 391 (TCP / Application type - HTTP)**

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

[Frame 391] Local time - 23:03:59.542677
427 bytes on wire (3416 bits), 427 bytes captured (3416 bits)
Total Length: 413 bytes / IP Header Length: 20 bytes (5)

Protocol: TCP (6)
Source Port: 53034
Destination Port: 80
Sequence Number (raw): 4022124255
(Starting sequence number: 4022124255 / Ending Sequence number: 4022124627)
Acknowledgment Number (raw): 847254337
Flags: PSH ACK
Window: 514
Urgent Pointer: 0
TCP payload (373 bytes)
Application type: HTTP

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



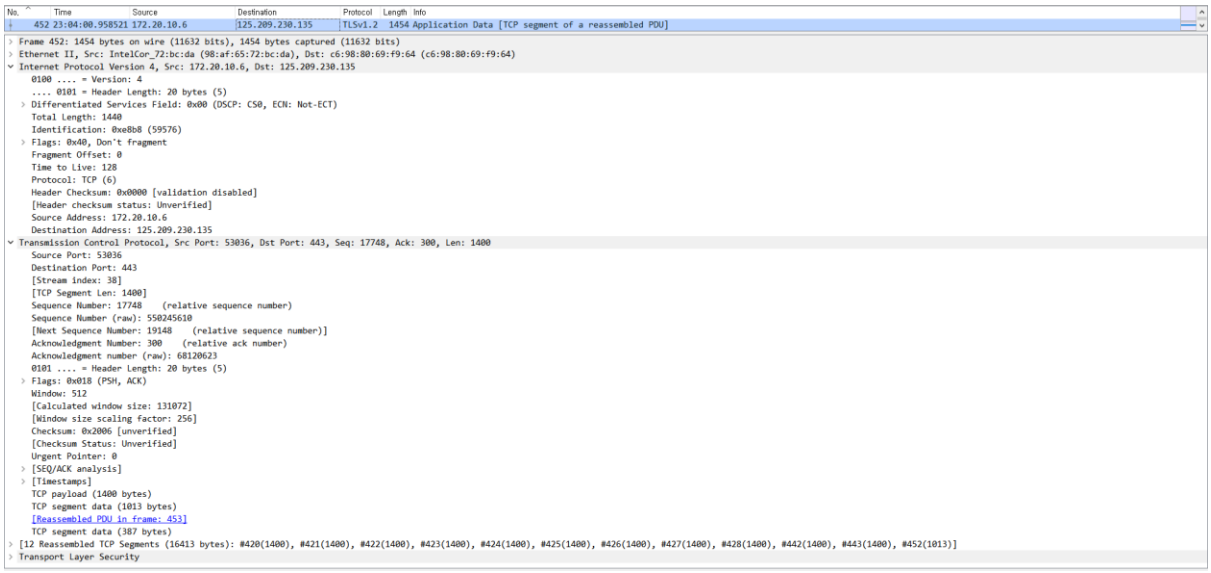
- **Frame 452 (TCP / Application type - HTTPS)**

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

[Frame 452] Local time - 23:04:00.958521
1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
Total Length: 1440 bytes / IP Header Length: 20 bytes (5)

Protocol: TCP (6)
Source Port: 53036
Destination Port: 443
Sequence Number (raw): 550245610
(Starting sequence number: 550245610 / Ending Sequence number: 550247009)
Acknowledgment Number (raw): 68120623
Flags: PSH ACK
Window: 512
Urgent Pointer: 0
TCP payload (1400 bytes)
Application type: HTTPS

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



* 와이어샤크의 분석 결과를 살펴보면, Fragmented Segments 중 하나임을 알 수 있습니다.

```
[12 Reassembled TCP Segments (16413 bytes): #420(1400), #421(1400), #422(1400), #423(1400), #424(1400), #425(1400), #426(1400), #427(1400), #428(1400), #442(1400), #443(1400), #452(1013)]
```

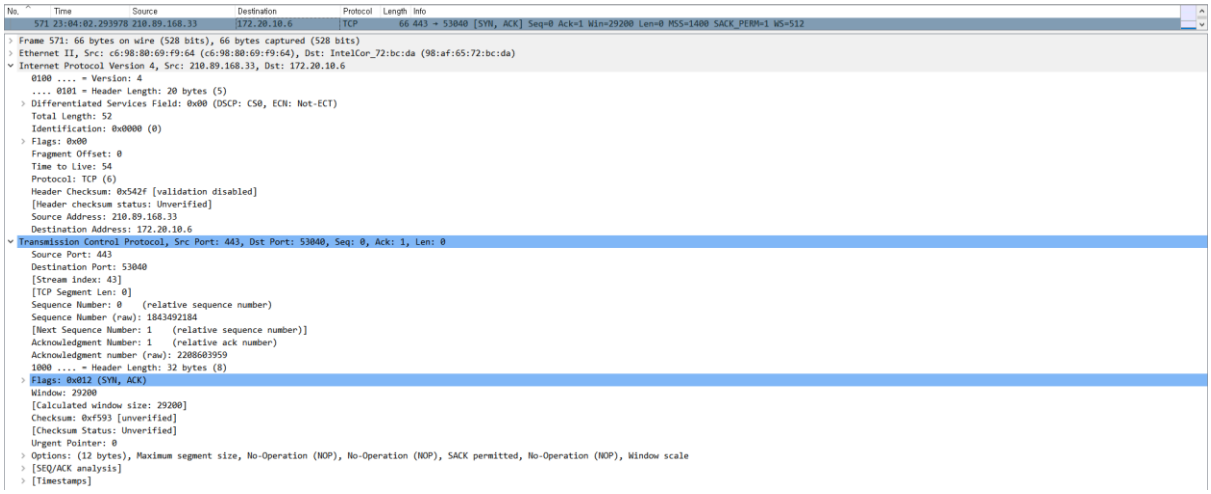
- **Frame 571 (TCP / Application type - HTTPS)**

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

[Frame 571] Local time - 23:04:02.293978
66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Total Length: 52 bytes / IP Header Length: 20 bytes (5)

Protocol: TCP (6)
Source Port: 443
Destination Port: 53040
Sequence Number (raw): 1843492184
Acknowledgment Number (raw): 2208603959
Flags: ACK SYN
Window: 29200
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size (MSS), No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale (WSCALE)
TCP payload (0 bytes)
Application type: HTTPS

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



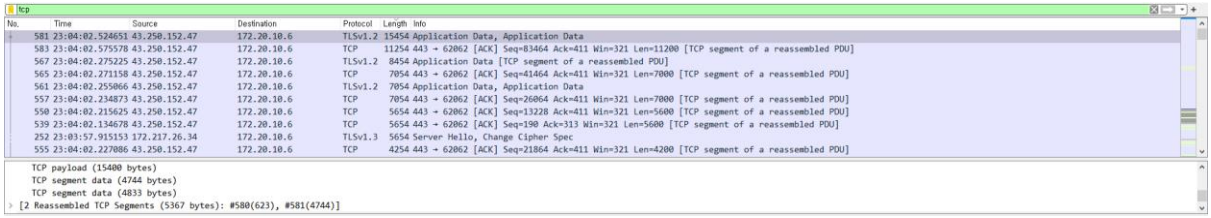
* TCP Options (12bytes): Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale이 제대로 출력되었음을 확인할 수 있습니다.

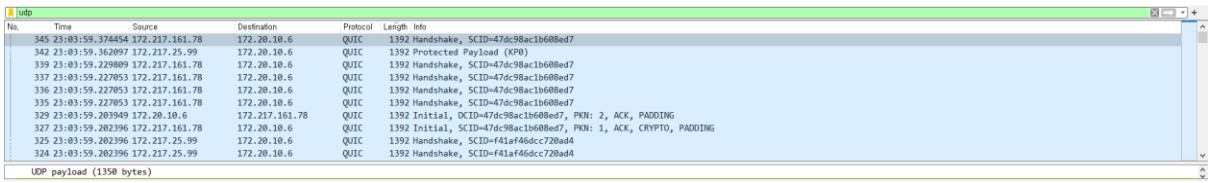
- **The greatest payload sizes among segments in TCP and UDP, respectively**

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

The greatest payload sizes among segments in TCP :15400 bytes
The greatest payload sizes among segments in UDP :1350 bytes

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





[Discussion of unique experience]

1. 이번 과제에서는 Applications에 대한 제출 조건이 따로 없어서 TCP와 UDP에서 동작하는 각각의 Applications을 간략하게 정리해보았습니다.

(1) Applications running over TCP

- TLS (Transport Layer Security)

  : 클라이언트/서버 응용 프로그램이 네트워크로 통신하는 과정에서 도청, 간섭, 위조를 방지하기 위해 설계된 암호 규약입니다.

- HTTP (Hypertext Transfer Protocol)

  : 클라이언트와 서버 사이에 이루어지는 TCP/IP 기반 요청/응답 프로토콜입니다.

- HTTPS (Hypertext Transfer Protocol over Secure socket layer)

  : HTTP의 보안이 강화된 버전으로, 소켓 통신에서 일반 텍스트를 이용하는 대신 SSL이나 TLS 프로토콜을 통해 세션 데이터를 암호화하여 데이터의 적절한 보호를 보장합니다.

- TELNET

  : 인터넷이나 로컬 영역 네트워크 연결에 쓰이는 네트워크 프로토콜입니다.

- SSH (Secure Shell)

  : 원격지 호스트 컴퓨터에 접속하기 위해 사용되는 인터넷 프로토콜로, SSH는 암호화 기법을 사용하기 때문에 통신이 노출된다고 하더라도 이해할 수 없는 암호화된 문자로 보입니다.

- FTP (File Transfer Protocol)

  : TCP/IP 프로토콜을 가지고 서버와 클라이언트 사이의 파일 전송을 하기 위한 프로토콜입니다.

- SMTP (Simple Mail Transfer Protocol)

  : 인터넷에서 이메일을 보내기 위해 사용되는 프로토콜로, 메일 서버 간의 송수신만 아니라, 메일 클라이언트에서 메일 서버로 메일을 보낼 때에도 사용됩니다.

- POP3 (Post Office Protocol 3)

  : 원격 서버로부터 TCP/TP 연결을 통해 이메일을 가져오는데 사용됩니다. 윈도우 라이브 핫 메일, G메일 같은 대부분의 웹 메일에서 지원합니다.

- IMAP4

  : POP와 같은 역할을 수행하지만, IMAP는 온라인과 오프라인 모드를 모두 지원하므로 POP3를 사용할 때와 달리 이메일 메시지를 서버에 남겨 두었다가 추후에 지울 수 있습니다.

-BGP (Boarder Gateway Protocol)

: 인터넷에서 목적지까지 경유하는 AS 중 라우팅 및 자율 시스템의 순서를 전송하기 위해 설계된 경로 지정 알고리즘으로, 표준화된 외부 게이트웨이 프로토콜의 하나입니다.

연결 지향형인 TCP는 3-way handshaking 과정을 통해 연결 후 통신을 시작하며, 흐름제어와 혼잡제어를 지원하고 데이터의 순서를 보장합니다. 그렇기 때문에 UDP에 비해 속도가 느리다는 단점에도 불구하고 데이터 오류가 발생하지 않아야 하는 웹 HTTP 통신, 이메일, 파일전송 등에서 TCP를 사용합니다.

(2) Applications running over UDP

- DNS (Domain Name System)

: 호스트의 도메인 이름을 네트워크 주소로 바꾸거나 그 반대의 변환을 수행합니다.

- SSDP (Simple Service Discovery Protocol)

: 네트워크 서비스나 정보를 찾기 위해 사용하는 네트워크 프로토콜입니다.

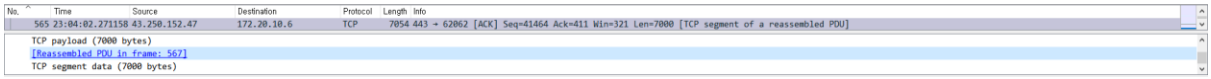
- QUIC

: 범용 목적의 전송 계층 프로토콜로써, 구글에 의하여 개발되었습니다.

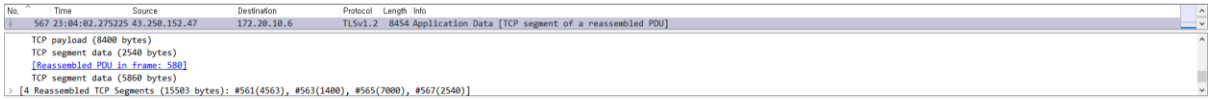
이 밖에도 IPTV, 음성 인터넷 프로토콜(VoIP), TFTP, 많은 온라인 게임 등의 Network Application에서 UDP를 사용합니다.

2. HTTP/3에서 TCP가 아닌 UDP를 선택한 이유가 궁금했기 때문에, 이번 과제를 하면서 UDP를 기반으로 하는 Application layer protocol, QUIC에 대한 내용을 추가적으로 찾아보았습니다. QUIC은 구글에서 2013년에 공식 발표한 프로토콜로써 전달 속도의 개선과 더불어 클라이언트와 서버의 연결 수를 최소화하고, 대역폭(Bandwidth)을 예상하여 패킷 혼잡(Congestion)을 피할 수 있는 것이 주요한 특징입니다. 기존의 TCP의 경우, 성능보다는 기능에 초점을 두었기 때문에 현대의 멀티미디어 컨텐츠를 디바이스에 빠르게 전달해야 하는 상황에서 이러한 TCP의 한계를 극복하고 최적화하는 것이 많은 기업들의 도전 과제였습니다. QUIC은 UDP의 빠른 성능을 이용해 해당 문제를 어느 정도 해결할 수 있었습니다. 또한 UDP의 단순한 헤더 구조에서 알 수 있듯이, UDP는 커스터마이징이 용이하기 때문에 개발자가 TCP가 가지고 있던 기능을 필요에 의해 구현할 수 있는 장점도 가지고 있습니다. 이러한 장점들에 기반해 HTTP/3가 UDP를 사용함으로써 기존 프로토콜에 비해 연결 설정 시 Latency가 감소하고, 패킷 손실 감지에 걸리는 시간이 단축되었으며 Multiplexing을 지원할 수 있게 되었습니다. HTTP/3가 기존 HTTP에 비해 단기간에 나왔고, 매우 급격한 변화가 있었기 때문에 많은 우려가 존재하기는 하지만 기존의 TCP 기반 HTTP가 가지는 한계점을 돌파하기 위한 좋은 시도임은 분명해 보입니다.

3. 이번 과제를 하면서, 가장 예상 외였던 결과는 greatest payload sizes among segments in TCP가 매우 큰 값을 가지는 것이었습니다. 일반적으로 Ethernet에서 MTU(Maximum Transmission Unit) 값은 1500 bytes이기 때문에 IPv4와 TCP의 Header에 Option이 존재하지 않는다는 가정 하에서 IP Header의 크기(20 bytes)와 TCP Header의 크기(20 bytes)를 각각 뺀 1460 bytes가 MSS(Maximum Segment Size)로 설정됩니다. 그렇기 때문에 greatest payload sizes among segments in TCP의 값 역시 1460 bytes 또는 그 이하가 나올 것이라고 생각했습니다. 하지만, 과제를 수행하며 확인해보니 대다수의 패킷들이 1460 bytes 이하의 Payload를 가지긴 했지만 위에 첨부된 사진에서 볼 수 있듯 1460 bytes를 넘어서는 Payload를 가지는 몇몇 패킷들도 존재했습니다. 이러한 결과가 발생한 이유에 대해 알아보고자 했지만, 정확한 답을 찾아내지 못했습니다. 다만, 매우 큰 값의 Payload를 가지는 대다수의 Packet들의 Info에 'TCP segment of a reassembled PDU'라고 명시되어 있고, 'Reassembled PDU in frame'으로 명시된 패킷들이 1460 bytes 이하의 값을 가지는 것을 보았을 때 Reassemble하는 과정을 거쳐 Segment를 전송할 수 있는 크기로 만들어주는 것이 아닌가 추측해보았습니다.



No.565 (7000 bytes) and something else reassembled PDU in No.567 (8400 bytes).



No.567 (5860 bytes) and something else reassembled PDU in No.580 (1400 bytes).