<Network HW1. Stripping IP Packet> 2015310884 철학과 박소현

1. Source Code

[linux에서 실행하기]

sudo apt-get install libpcap-dev //pcap library를 사용하므로 설치

g++ packetParser.cpp -o packetParser -lpcap

./packetParser

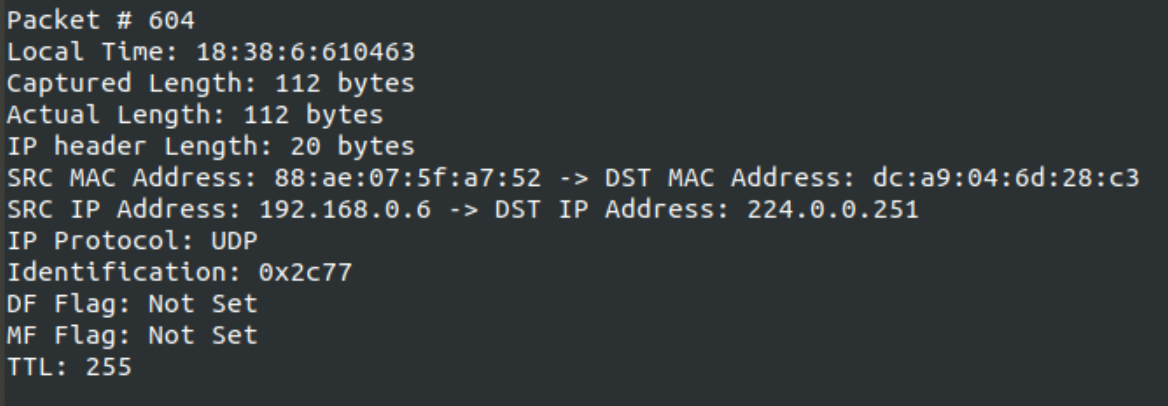
|  |
| --- |
| packetParser.cpp |
| #include <string>  #include <iostream>  #include <pcap.h>  #include <arpa/inet.h>  using namespace std;  //hours, minutes and seconds  void ConvertReadableTime(int n, int usec)  {  n = n % (24 \* 3600);  int hour = n / 3600;    n %= 3600;  int minutes = n / 60 ;    n %= 60;  int seconds = n;    cout << hour << ":" << minutes << ":" << seconds << ":";  printf("%06d\n", usec);  }  void parsing(pcap\_t \*pcap)  {  struct pcap\_pkthdr \*header;  const u\_char \*data;    u\_int packetCount = 0;  while (int returnValue = pcap\_next\_ex(pcap, &header, &data) >= 0)  {  printf("Packet # %i\n", ++packetCount); //Show the packet number    // loop through the packet and print it as hexidecimal representations of octets  //for (int i=0; i < header->caplen; i++)  //{  // if ( (i % 16) == 0)  // printf("\n");  // printf("%.2x ", data[i]); //Print each octet as hex (x), make sure there is always two characters (.2).  //}  //printf("\n");  //pcap\_pkthdr  printf("Local Time: ");  ConvertReadableTime((int)header->ts.tv\_sec, (int)header->ts.tv\_usec);    printf("Captured Length: %d bytes\n", header->caplen); //Show the size in bytes of the packet  printf("Actual Length: %d bytes\n", header->len);    //IP header Length  printf("IP header Length: %d bytes\n", (data[14]&15) \* 4);  //MAC Address  printf("SRC MAC Address: ");  for(int i=6; i<=11; i++){  printf("%.2x", data[i]);    if(i != 11) printf(":");  else printf(" -> ");  }    printf("DST MAC Address: ");  for(int i=0; i<=5; i++){  printf("%.2x", data[i]);    if(i != 5) printf(":");  else printf("\n");  }    //IP Address  printf("SRC IP Address: ");  for(int i=26; i<=29; i++){  printf("%d", data[i]);    if(i != 29) printf(".");  else printf(" -> ");  }    printf("DST IP Address: ");  for(int i=30; i<=33; i++){  printf("%d", data[i]);    if(i != 33) printf(".");  else printf("\n");  }    //IP Protocol  printf("IP Protocol: ");  int ipProtocol = data[23];    switch (ipProtocol) {  case 1: printf("ICMP\n"); break;  case 6: printf("TCP\n"); break;  case 17: printf("UDP\n"); break;    default: printf("%d\n", ipProtocol); break;  }    //Identification  printf("Identification: 0x");  for(int i=18; i<=19; i++)  printf("%.2x", data[i]);  printf("\n");    //Flags DF/MF  printf("DF Flag: ");  if((data[20]&64) != 0) printf("Set\n");  else printf("Not Set\n");  printf("MF Flag: ");  if((data[20]&32) != 0) printf("Set\n");  else printf("Not Set\n");    //TTL  printf("TTL: %d\n", data[22]);    printf("\n\n");  }  }  int main(int argc, char \*argv[])  {  string file;  cout << "Input file name with path (example./home/sohyun/Network/packetCapture.pcap):\n";  cin >> file;    char errbuff[PCAP\_ERRBUF\_SIZE];  pcap\_t \*pcap = pcap\_open\_offline(file.c\_str(), errbuff);    parsing(pcap);    return 0;  } |

2. Screenshots of output for three sample packets

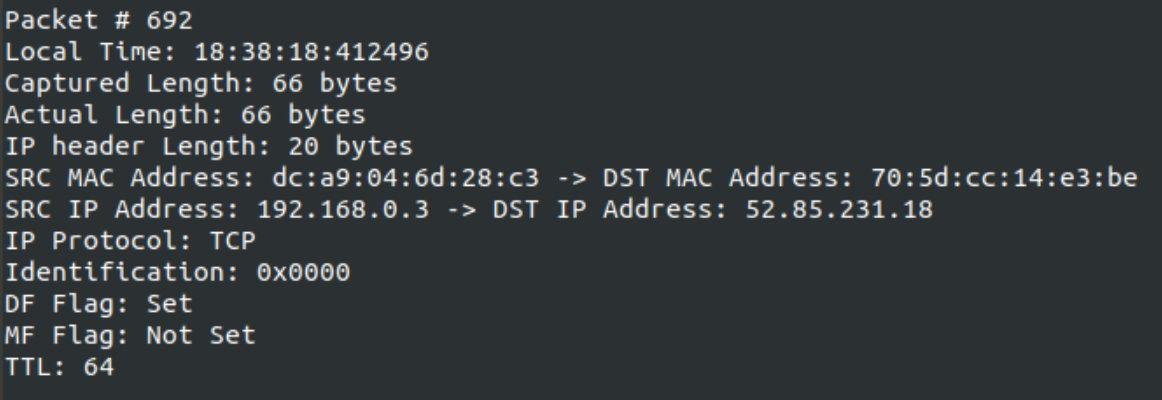
[1]



[2]

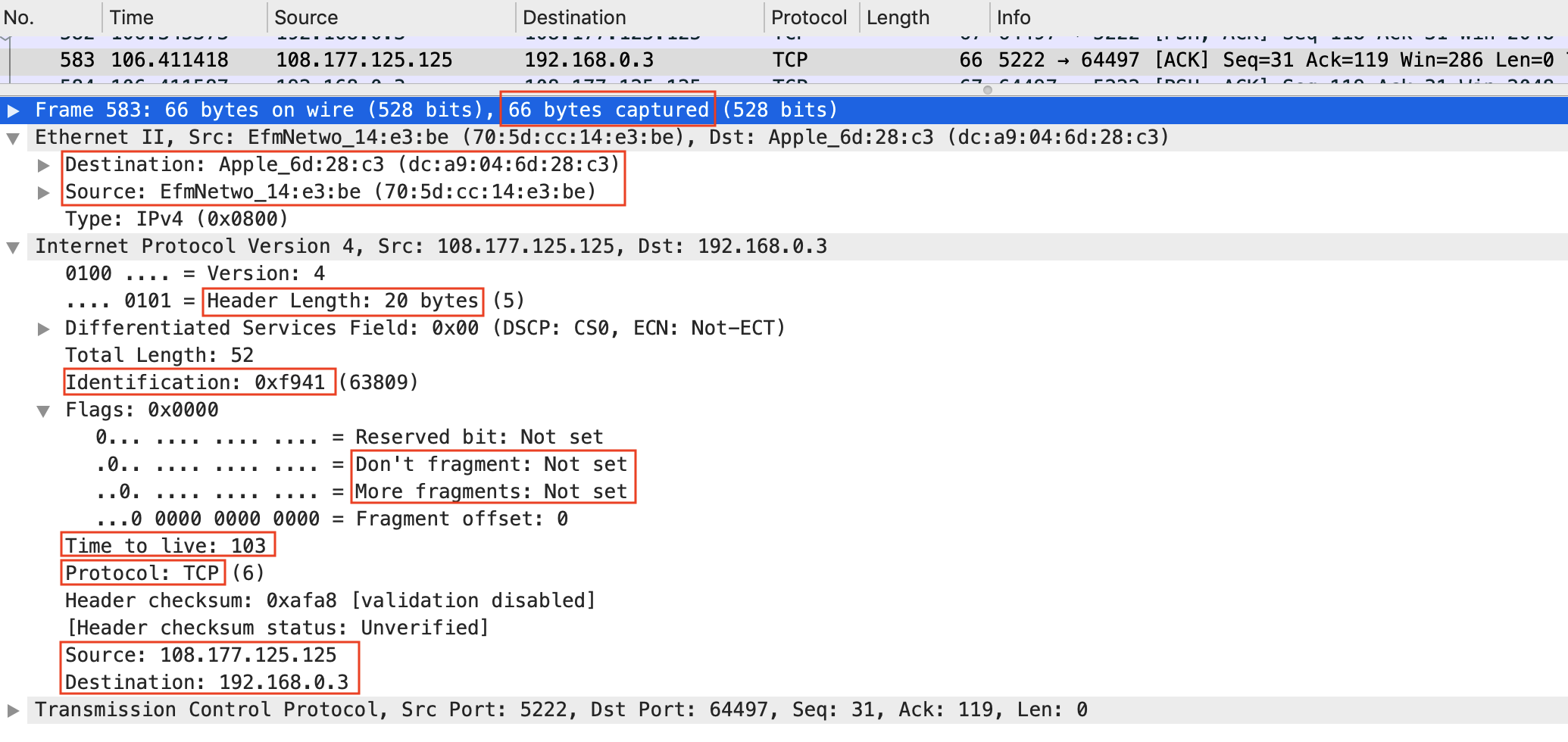


[3]

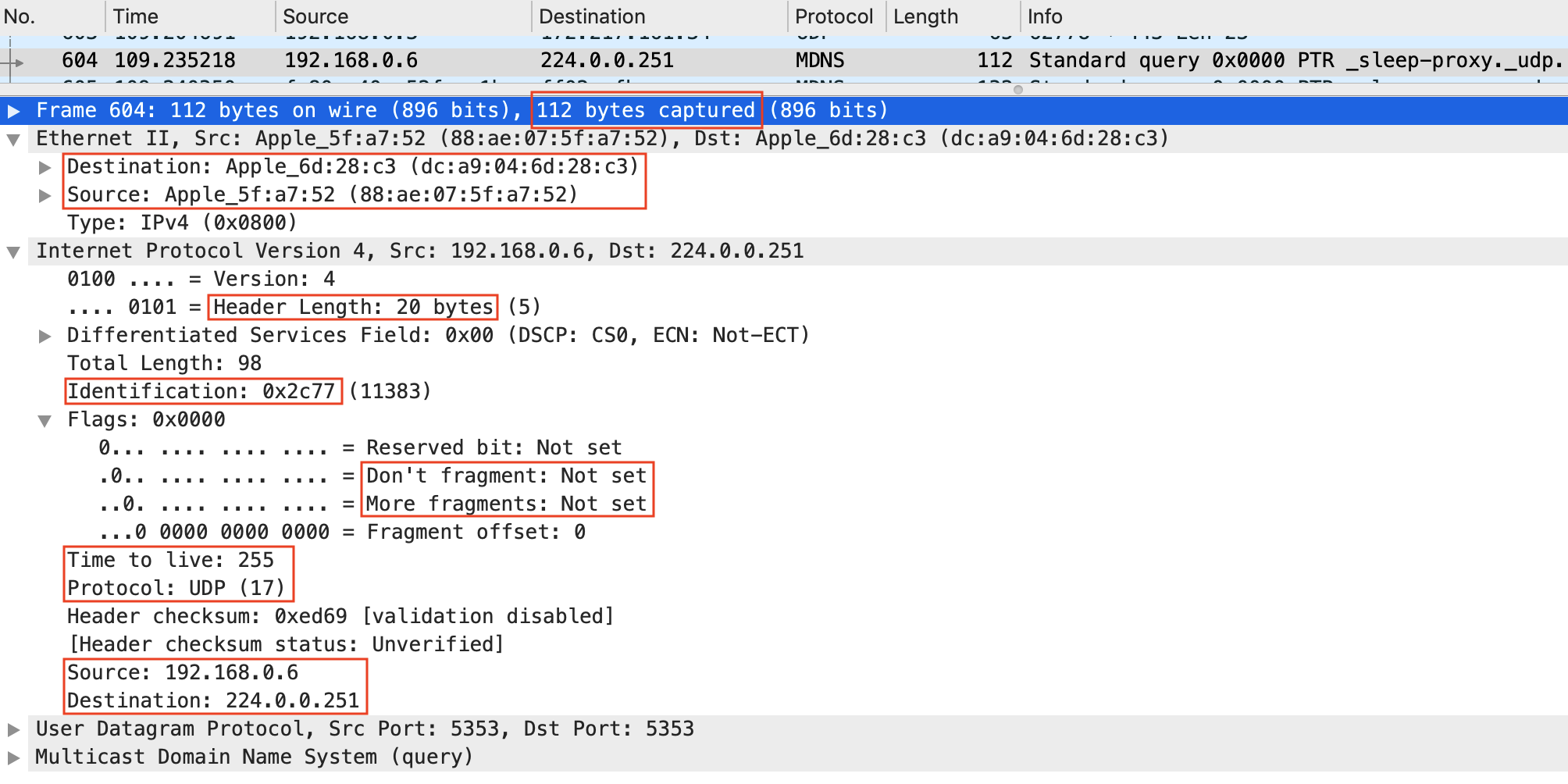


3. Verification with Wireshark

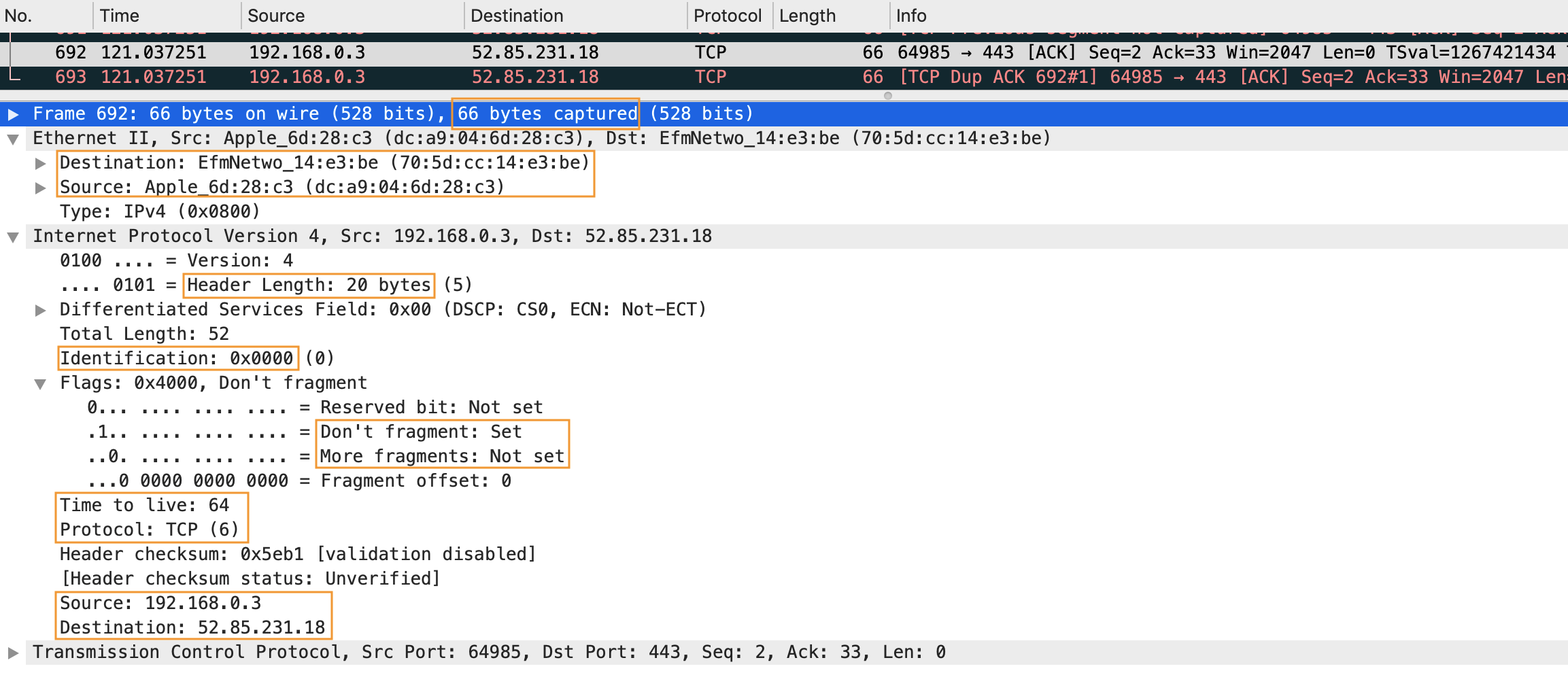
[1]



[2]



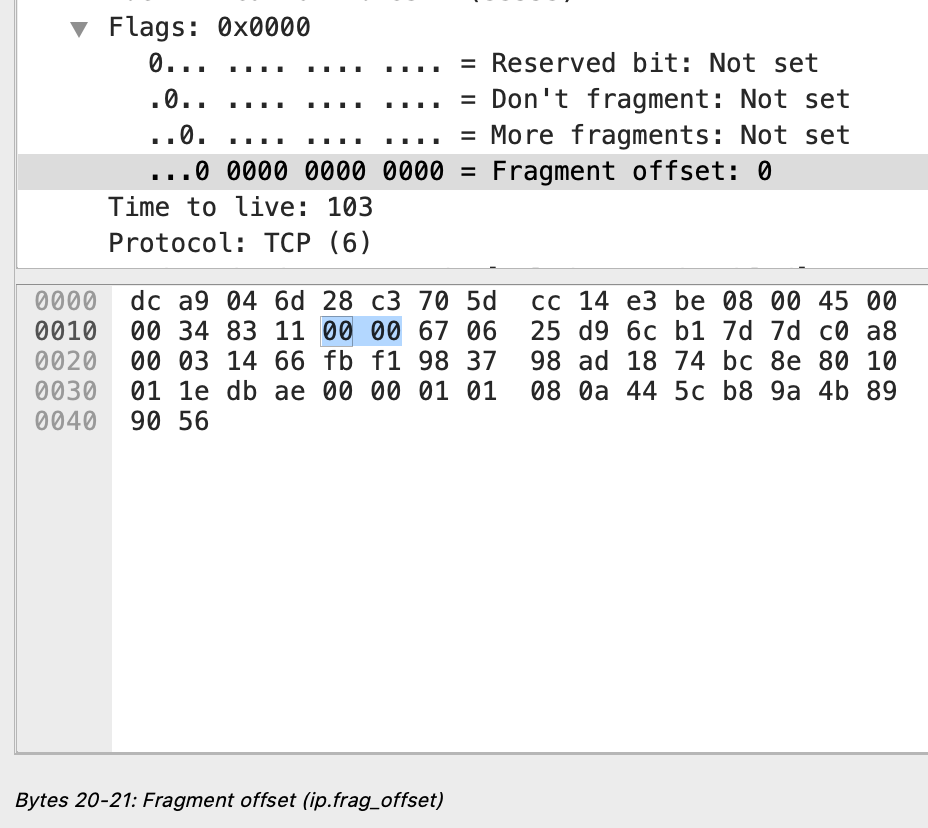
[3]



4. Discussion of unique experience

이번 과제를 통해서 IP header에 대한 이해도를 높일 수 있었고, Wireshark와 pcap library를 사용해 볼 수 있는 기회가 되었다.

특히 DF와 MF를 구할 때 어려움이 있었는데, Wireshark로 직접 확인해보았더니,



DF/MF를 알 수 있는 Flags 가 분리되어 있지 않고, Fragment offset 안에 포함되어 있었기에 비트 연산자 & 를 사용하여 Flags 부분을 분리해내야 됐었다.

pcap library를 사용한 덕분에 pcap\_next\_ex() 함수를 사용하여 pcap 파일을 깔끔하게 읽는 방법을 알게 되었고, octet 데이터 처리 방법도 익숙해진 듯 하다.