

REPORT

컴퓨터네트워크 Assignment #1

Computer Network Assignment #1

Packet Capture Using Wireshark



학과	컴퓨터학과
학번	2015410056
이름	김지윤
제출 일자	2018/05/13
담당 교수	민성기 교수님

Assignment #1: Packet Capture Using Wireshark

Understand how packets are actually generated and transmitted from the end host to the end host

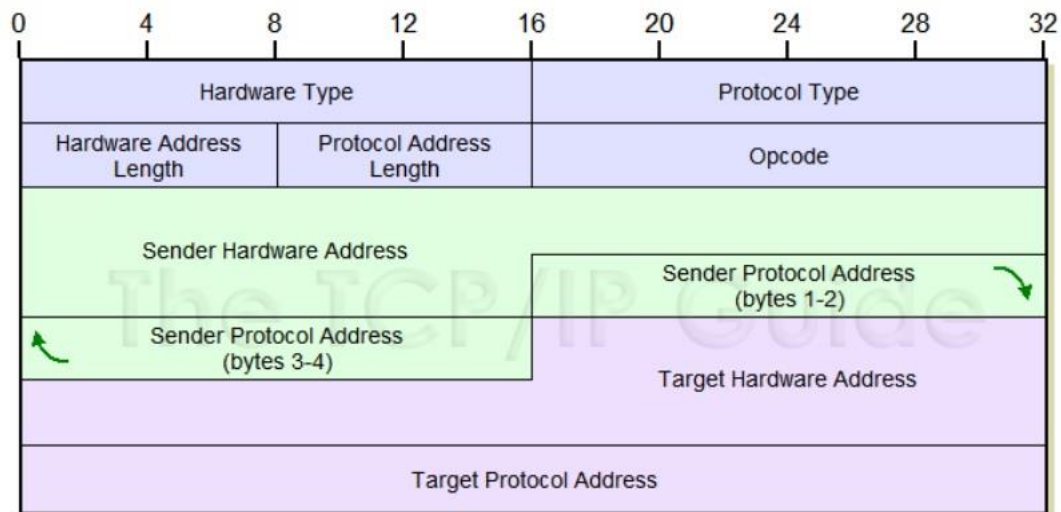
■ ARP Packet Capture & Analysis

a. ARP (Address Resolution Protocol)

패킷을 host에 전달하기 위해서는 host의 물리 주소(Ethernet의 Identifier = MAC Address)가 필요하다. 하지만 패킷이 가지는 것은 Destination IP Address (IP의 Identifier) 밖에 없기 때문에 Destination의 물리 주소를 얻기 위해 ARP(Address Resolution Protocol)을 이용한다.

Destination까지 1-hop-away인 router에서 (Destination의 subnet router에서) 패킷이 필요한 물리 주소를 얻기 위해 Destination IP Address를 담은 request 메시지를 broadcast로 보내고 Destination IP Address에 해당하는 host는 자신의 물리주소를 담아 reply 메시지를 보내 자신의 물리 주소(MAC Address)를 알린다.

b. ARP Packet Format



- ✓ Hardware Type: 물리 주소의 종류를 명시한다. (Ethernet일 경우 0x0001)
- ✓ Protocol Type: 상위 layer의 프로토콜 종류를 명시한다. (보통의 경우 IP 0x0800)
- ✓ Hardware Address Length: 물리 주소의 길이를 8bit로 나타내며 byte단위이다.
- ✓ Protocol Address Length: 프로토콜 주소의 길이를 8bit로 나타내며 byte단위이다.

- ✓ Opcode: 메시지 타입을 나타낸다. (1: request / 2: reply)
- ✓ Sender / Target Hardware Address: sender와 target의 물리 주소 값을 나타낸다.
- ✓ Sender / Target Protocol Address: sender와 target의 프로토콜 주소 값을 나타낸다.

c. ARP Packet Capture & Analysis

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	IntelCor_dc:f5:78	Broadcast	ARP	42	Who has 192.168.0.1 is at
2	0.002043	AsustekC_c2:4c:c0	IntelCor_dc:f5:78	ARP	42	192.168.1.1 is at
7	0.055105	IntelCor_dc:f5:78	Broadcast	ARP	42	Who has 192.168.0.1 is at
8	0.057664	AsustekC_c2:4c:c0	IntelCor_dc:f5:78	ARP	42	192.168.1.1 is at
11	0.078588	IntelCor_dc:f5:78	Broadcast	ARP	42	Who has 192.168.0.1 is at

Wireshark를 이용해 ARP packet들을 캡처 했고 그 중 위의 두 패킷(No.1, No.2)을 분석할 것이다.

1) Packet No.1 (Request)

```

Wireshark · Packet 25 · arp.pcapng

> Frame 25: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface
  > Ethernet II, Src: EfmNetwo_76:81:36 (88:36:6c:76:81:36), Dst: IntelCor_dc:f5:78 (e8:b1:fc:dc:f5:78)
    > Destination: IntelCor_dc:f5:78 (e8:b1:fc:dc:f5:78)
    > Source: EfmNetwo_76:81:36 (88:36:6c:76:81:36)
      Type: ARP (0x0806)
    > Address Resolution Protocol (request)
      Hardware type: Ethernet (1)
      Protocol type: IPv4 (0x0800)
      Hardware size: 6
      Protocol size: 4
      Opcode: request (1)
      Sender MAC address: EfmNetwo_76:81:36 (88:36:6c:76:81:36)
      Sender IP address: 192.168.0.1
      Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
      Target IP address: 192.168.0.5

0000  e8 b1 fc dc f5 78 88 36 6c 76 81 36 08 06 00 01  ....x.6.l.v.6....
0010  08 00 06 04 00 01 88 36 6c 76 81 36 c0 a8 00 01  ....6.l.v.6....
0020  00 00 00 00 00 00 c0 a8 00 05  ....
  
```

ARP 패킷은 Ethernet패킷의 payload로서 담기며 **Ethernet의 type필드의 값은 0x0806으로 상위 layer의 protocol이 ARP라는 것을 명시하고 있다.** (빨간색 네모)

위의 packet은 IP Address를 이용해 Etherent 주소를 알아내려는 **request** 코드 패킷이다. 따라서 hardware type의 값은 Ethernet을 나타내는 0x0001, protocol type 값은 IP

를 나타내는 0x0800이 된다.

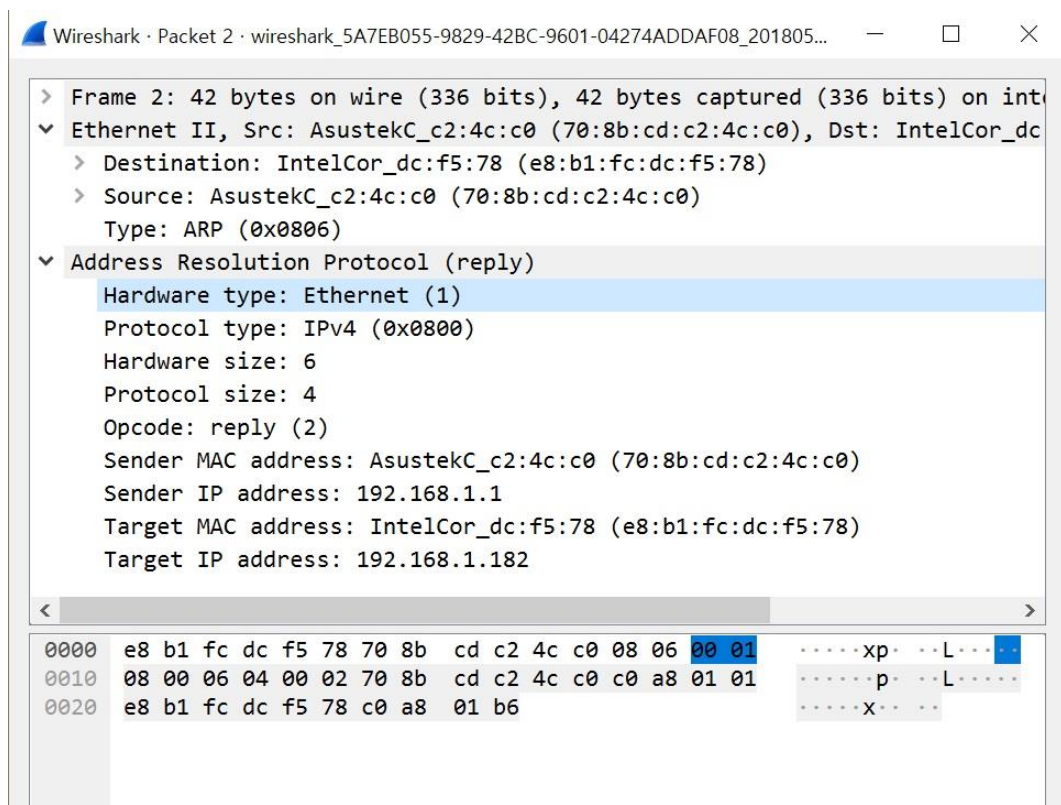
IP Address의 길이는 32bits = 4bytes / Ethernet (MAC) Address의 길이는 48bits = 6bytes이다. 뒤이어 나오는 두 필드 Hardware size, Protocol size에 해당하는 값이 차례로 들어가게 된다. (byte 단위이므로 6, 4가 차례로 들어간다) 그리고 뒤의 Opcode의 값 1은 패킷이 하드웨어 주소를 요청하는 request 패킷이라는 것을 명시한다.

sender와 target의 MAC Address(물리 주소)와 IP Address가 차례로 나오는데 그 값은 아래와 같다.

- ✓ Sender MAC Address: **e8:b1:fc:dc:f5:78**
- ✓ Sender IP Address: **192.168.1.182**
- ✓ Target MAC Address: **00:00:00:00:00:00** (**broadcast**일 경우의 물리 주소)
- ✓ Target IP Address: **192.168.1.1**

이 패킷은 Destination 주소를 가지고 Destination의 MAC address를 요청하는 request 패킷으로 Target MAC Address 값을 broadcast로 하여 Destination의 subnet에 존재하는 모든 host에 메시지를 보낸다. 그리고 Destination IP주소를 가지는 host만이 이 request에 대한 응답을 보내는데 그 패킷이 아래 나오는 Packet No.2이다.

2) Packet No.2 (Reply)



Wireshark · Packet 2 · wireshark_5A7EB055-9829-42BC-9601-04274ADDAF08_201805...

> Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface
▼ Ethernet II, Src: AsustekC_c2:4c:c0 (70:8b:cd:c2:4c:c0), Dst: IntelCor_dc
 > Destination: IntelCor_dc:f5:78 (e8:b1:fc:dc:f5:78)
 > Source: AsustekC_c2:4c:c0 (70:8b:cd:c2:4c:c0)
 Type: ARP (0x0806)
▼ Address Resolution Protocol (reply)
 Hardware type: Ethernet (1)
 Protocol type: IPv4 (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: reply (2)
 Sender MAC address: AsustekC_c2:4c:c0 (70:8b:cd:c2:4c:c0)
 Sender IP address: 192.168.1.1
 Target MAC address: IntelCor_dc:f5:78 (e8:b1:fc:dc:f5:78)
 Target IP address: 192.168.1.182

0000 e8 b1 fc dc f5 78 70 8b cd c2 4c c0 08 06 00 01xp..L...
0010 08 00 06 04 00 02 70 8b cd c2 4c c0 00 a8 01 01p..L...
0020 e8 b1 fc dc f5 78 c0 a8 01 b6x..

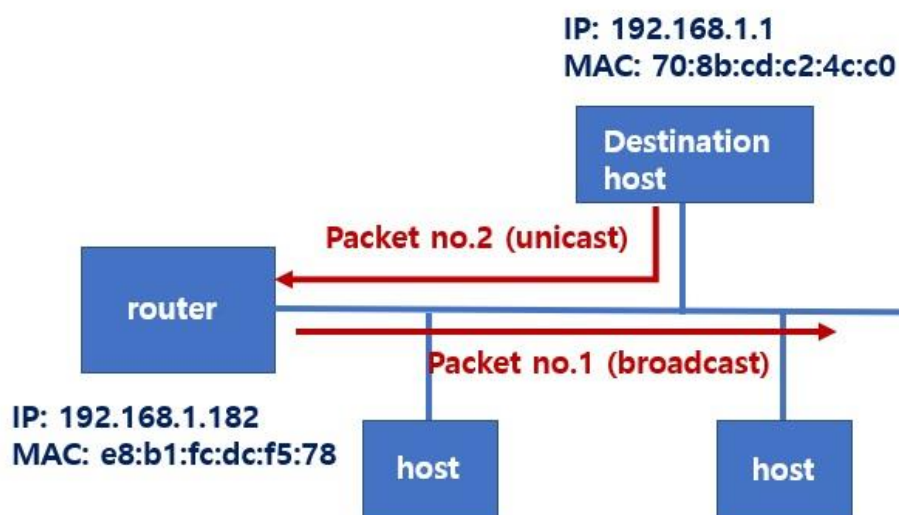
Packet No.2에 대한 reply packet이다. 하드웨어 주소와 프로토콜 주소는 Packet No.1과 같고 Opcode 값은 reply를 나타내는 2이다.

Sender와 Target의 MAC Address, IP Address 값을 차례로 보면,

- ✓ Sender MAC Address: **70:8b:cd:c2:4c:c0** (Packet No.1이 요청했던 MAC Address)
- ✓ Sender IP Address: **192.168.1.1** (Packet No.1의 Target IP Address)
- ✓ Target MAC Address: **e8:b1:fc:dc:f5:78** (Packet No.1의 Sender MAC Address)
- ✓ Target IP Address: **192.168.1.182** (Packet No.1의 Sender IP Address)

보이는 바와 같이 Sender와 Target의 MAC IP Address 값이 request packet의 것과 뒤바뀌어 있고 (Packet No.1의) Target MAC Address자리 (이 패킷의 Sender MAC Address 필드)는 이 패킷을 보내는 host의 MAC Address로 채워져 있다.

Packet No.2는 Packet No.1이 요청했던 정보 **MAC Address**를 담아 Packet No.1의 출처로 패킷을 보내는 것이다.

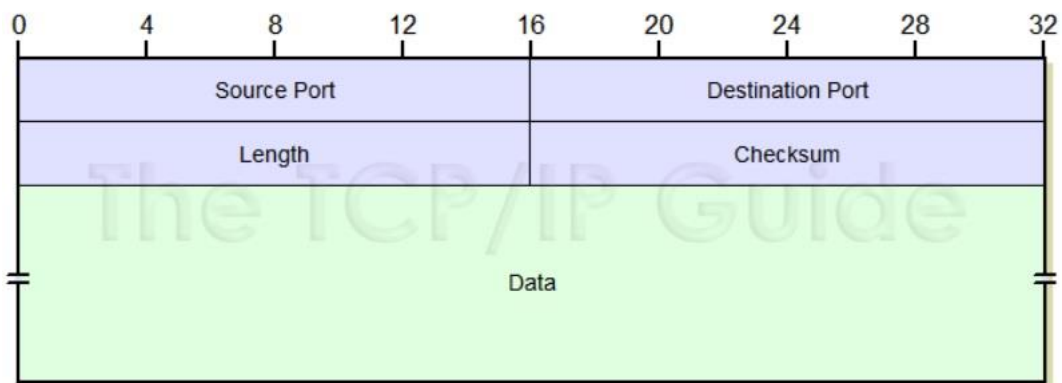


■ UDP Packet Capture & Analysis

a. UDP (User Datagram Protocol)

UDP는 port 번호를 이용해 IP 패킷을 그에 맞는 프로세스와 연결시켜주는 Transport layer의 protocol이다. TCP와 다르게 datagram-oriented이며 그 구조가 매우 간단하고 reliability를 제공하지 않는다. (**UDP의 Identifier = Port number**)

b. UDP Packet Format



UDP Packet의 구조는 아주 간단하다. payload부분을 제외하면 port 번호가 UDP가 나타내는 정보의 전부이다.

- ✓ Source Port: 패킷을 보내는 쪽의 프로세스를 나타내는 port 번호이다.
- ✓ Destination Port: 패킷을 받아야 하는 쪽의 port 번호이다.
- ✓ Length: UDP datagram 전체의 길이를 나타낸다. (header의 길이 + payload의 길이)
- ✓ Checksum: UDP의 checksum은 그 자신을 담고 있는 IP header의 일부까지 포함한다.
- ✓ Data: UDP 패킷의 payload 부분

c. UDP Packet Capture & Analysis

1) Server / Client Programming

UDP socket program in winsock을 돌려보고, 교환된 패킷을 분석한다.

(코드 출처: <https://www.binarytides.com/udp-socket-programming-in-winsock/>)

위 프로그램은 localhost를 이용하고 있어, localhost packet을 캡처하기 위해 RawCap을 활용하였다.

아래의 두 이미지는 차례로 Server, Client 코드를 각각 돌린 실행 화면과 WireShark의 packet 캡처 화면이다.

C:\WINDOWS\system32\cmd.exe

Initialising Winsock...Initialised.
Socket created.
Bind done
Waiting for data...Received packet from 127.0.0.1:55031
Data: hello
Waiting for data...Received packet from 127.0.0.1:55031
Data: how are you
Waiting for data...Received packet from 127.0.0.1:55031
Data: my name is
Waiting for data...Received packet from 127.0.0.1:55031
Data: jiyeon kim
Waiting for data...

<server>

C:\WINDOWS\system32\cmd.exe

Initialising Winsock...Initialised.
Enter message : hello
hello
Enter message : how are you
how are you
Enter message : my name is
my name is
Enter message : jiyeon kim
jiyeon kim
Enter message :
_

<client>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	33	55031 → 8888 Len=5
2	0.001997	127.0.0.1	127.0.0.1	UDP	33	8888 → 55031 Len=5
3	2.184319	127.0.0.1	127.0.0.1	UDP	39	55031 → 8888 Len=11

위의 패킷들 중 input 'hello'에 대한 패킷 두 개(실행화면의 노란 네모 / Wireshark의 빨간 네모 부분)를 분석할 것이다. (Packet No.1, Packet No.2)

① Packet No.1

Wireshark · Packet 1 · loopback.pcap

> Frame 1: 33 bytes on wire (264 bits), 33 bytes captured (264 bits)
Raw packet data

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 33

Identification: 0x2708 (9992)

> Flags: 0x0000

Time to live: 128

Protocol: UDP (17)

Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]

Source: 127.0.0.1

Destination: 127.0.0.1

> User Datagram Protocol, Src Port: 55031, Dst Port: 8888

Source Port: 55031

Destination Port: 8888

Length: 13

Checksum: 0xc44f [unverified]
[Checksum Status: Unverified]

[Stream index: 0]

> Data (5 bytes)

Data: 68656c6c6f

[Length: 5]

0000 45 00 00 21 27 08 00 00 80 11 00 00 7f 00 00 01 E..!'.

0010 7f 00 00 01 d6 f7 22 b8 00 0d c4 4f 68 65 6c 6c". ...hell

0020 6f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

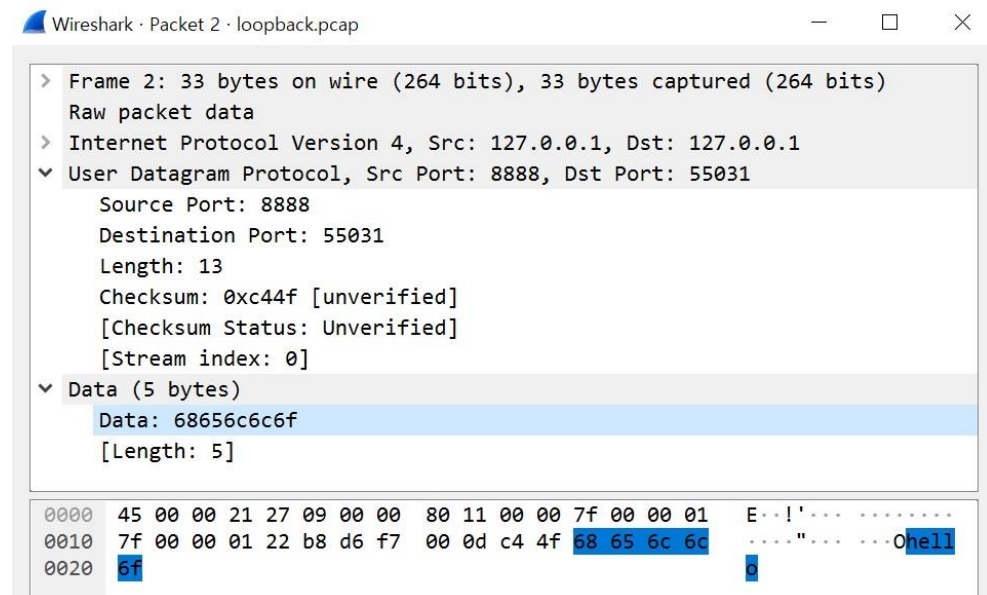
UDP packet은 payload로서 DNS packet을 가지고, IP packer의 payload로서 담긴다.
IP packet의 protocol필드의 값은 17으로 상위 layer의 protocol이 UDP라는 것을 명시하고 있다. (빨간색 네모)

UDP의 identifier는 port number이며 port 번호를 통해 패킷을 프로세스와 연결시키는 것이 UDP의 가장 큰 역할이다.

- ✓ Source port: 55031
- ✓ Destination port: 8888
- ✓ Length: UDP header(8 bytes) + payload의 바이트 수이며 이 패킷의 경우 payload는 length 5의 char string으로 그 크기가 5byte이다. 따라서 length 값은 $8+5 = 13$ 이 된다.
- ✓ Checksum: UDP datagram과 IP header의 일부를 포함한 checksum이다.

이 패킷은 client에서 서버로 보내는 패킷으로 source port 55031은 UDP client 프로그램 (프로세스)을 의미하며 destination port 8888은 UDP server 프로그램 (프로세스)을 의미한다.

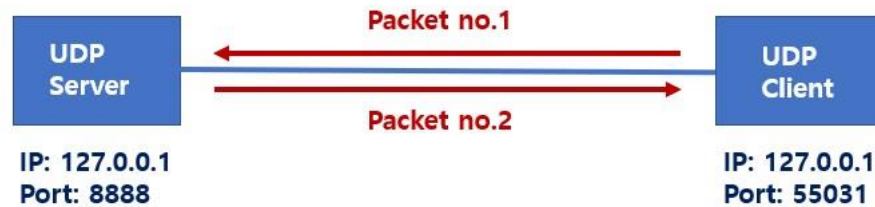
② Packet No.2



- ✓ Source port: 8888 (No.1의 destination port)
- ✓ Destination port: 55031 (No.1의 source port)

이 패킷의 source IP Address와 destination IP Address가 각각 Packet No.1의 destination IP Address, source IP Address 일치하고 port 번호가 위와 같이 뒤바뀐다.

어 있는 것으로 이 패킷은 Packet No.1에 대한 server의 응답이라는 것을 알 수 있다.



2) DNS Query Using 'nslookup'

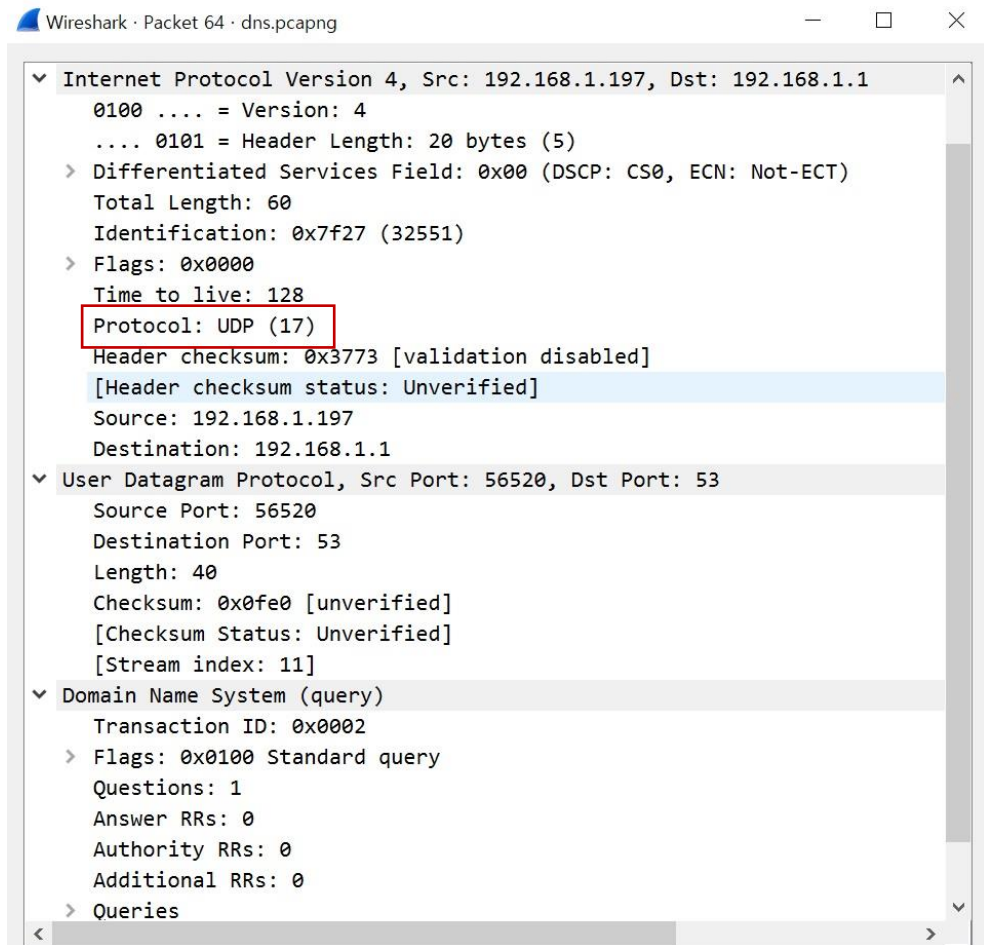
cmd에서 nslookup 명령어를 이용하여 DNS 패킷 교환에 사용되는 UDP패킷을 알아 본다.

dns.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
47	6.043208	192.168.1.197	192.168.1.1	DNS	84	Standard query 0x0001 PTR 1.
48	6.044721	192.168.1.1	192.168.1.197	DNS	113	Standard query response 0x0001 PTR 1.
64	11.820125	192.168.1.197	192.168.1.1	DNS	74	Standard query 0x0002 A www
65	11.823651	192.168.1.1	192.168.1.197	DNS	90	Standard query response 0x0002 A www
66	11.826965	192.168.1.197	192.168.1.1	DNS	74	Standard query 0x0003 AAAA w
67	11.854263	192.168.1.1	192.168.1.197	DNS	102	Standard query response 0x0003 AAAA w

Wireshark를 이용해 온 패킷들을 캡처 했고, 그 중 Packet No.64와 No.65를 분석할 것이다.

① Packet No.64



UDP packet은 payload로서 DNS packet을 가지고, IP packer의 payload로서 담긴다. **IP packet의 protocol필드의 값은 17으로 상위 layer의 protocol이 UDP라는 것을 명시하고 있다.** (빨간색 네모)

UDP의 메인 역할은 port 번호를 통해 패킷을 전달되어야 할 프로세스를 명시해 주는 것이다.

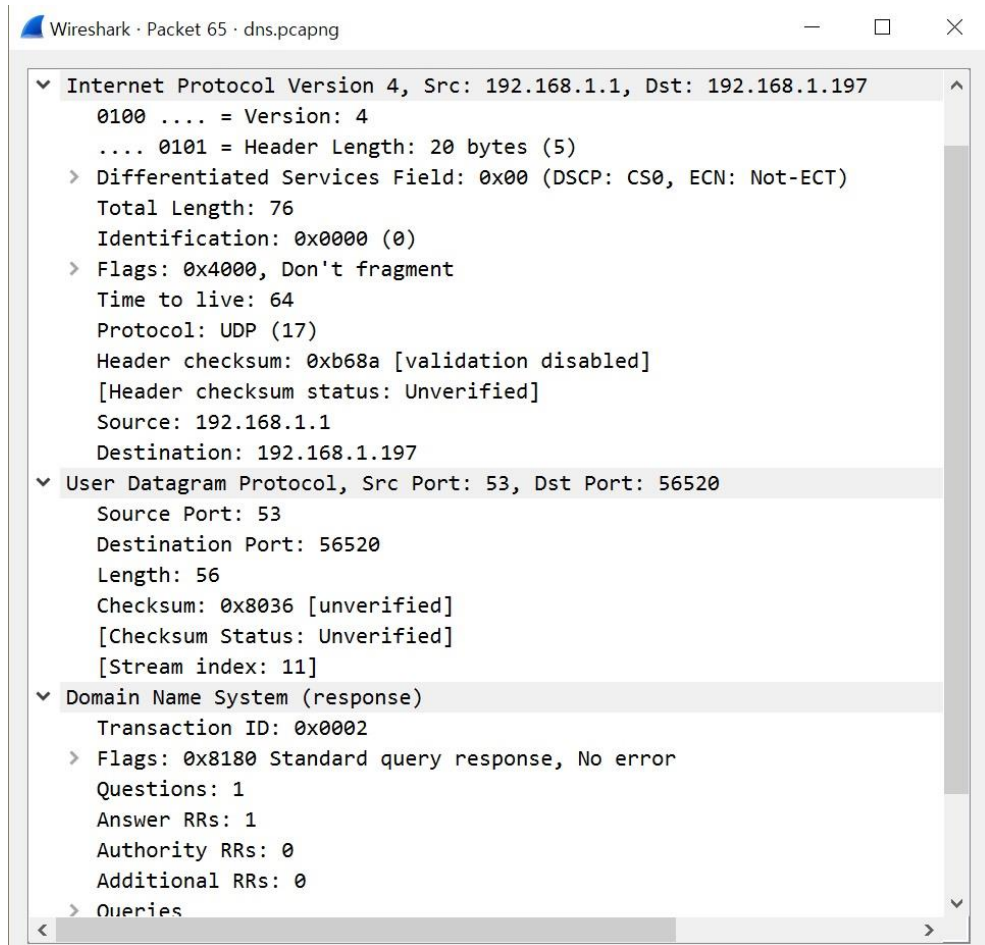
- ✓ Source port: 56520
- ✓ Destination port: 53
- ✓ Length: 40

Payload길이와 header길이를 포함한 값이다. UDP header의 길이는 보통 8 byte이므로 payload (DNS packet)의 길이는 32이다.

- ✓ Checksum: UDP datagram과 IP header 필드의 일부에 대한 checksum이다.

이 패킷은 nslookup www.google.com에 대한 패킷으로 이에 대한 응답이 Packet No.65이다.

② Packet No.65

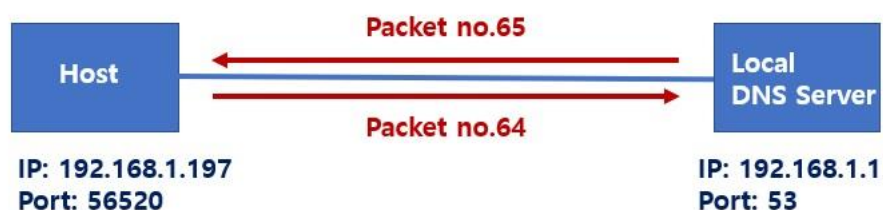


위 Packet No.64와 Source, Destination IP Address를 비교해보면 위 패킷에 대한 응답 패킷이라는 것을 알 수 있다.

UDP packet header부분 필드를 살펴보면

- ✓ Source port: 53 (No.64의 destination port)
- ✓ Destination port: 56520 (No.64의 source port)

Packet No.64와 source / destination port 번호가 서로 뒤바뀐 것을 알 수 있으며 이는 port 번호가 통신하는 프로세스 중 패킷에 알맞은 것을 명시하는 역할이라는 것을 의미한다.



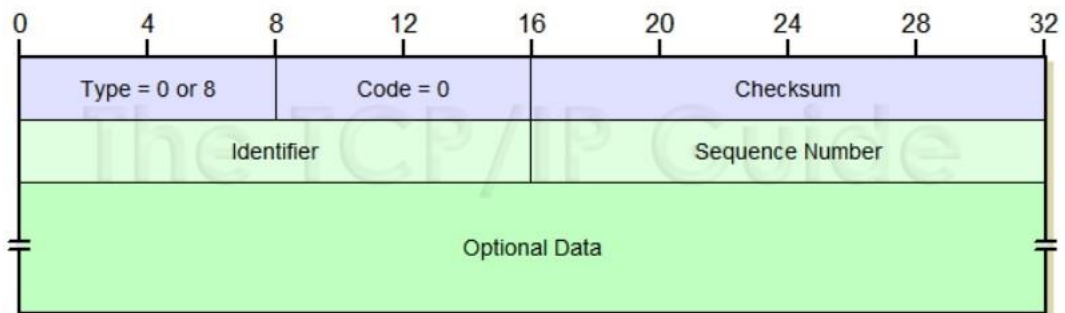
■ ICMP Packet Capture & Analysis

a. ICMP (Internet Control Message Protocol)

ICMP 패킷은 IP의 보조역할을 하는 network layer의 protocol이다.

ICMP 메시지는 크게 Error Message와 Query 메시지로 나눌 수 있으며 이번 과제에선 query 메시지를 분석할 것이다.

b. ICMP Packet Format



Informational message일 때의 ICMP packet 구조이다.

- ✓ Type: ICMP 메시지가 error메시지인지 query 메시지인지는 type값이 따라 갈린다. (echo request는 8, echo reply는 0)
- ✓ Code: Type에 대한 세부 설명이 들어가지만 echo request / reply는 이 필드가 필요 없다.
- ✓ Checksum: ICMP header에 대한 checksum이다.
- ✓ Identifier echo request와 echo reply를 match시키는 식별자이다.
- ✓ Sequence Number 같은 echo request와 echo reply가
- ✓ Optional Data 추가적으로 전달되어야 할 데이터가 여기에 담긴다. 이 필드의 크기는 정해져 있지 않다

c. ICMP Packet Capture & Analysis

```

=====
IPv4 경로 테이블
=====
활성 경로:
네트워크 대상      네트워크 마스크      게이트웨이      인터페이스      메트릭
0.0.0.0            0.0.0.0            192.168.1.1      192.168.1.182      40
127.0.0.0          255.0.0.0          연결됨           127.0.0.1          331
127.0.0.1          255.255.255.255    연결됨           127.0.0.1          331
127.255.255.255    255.255.255.255    연결됨           127.0.0.1          331
192.168.1.0         255.255.255.0      연결됨           192.168.1.182      296
192.168.1.182       255.255.255.255    연결됨           192.168.1.182      296
192.168.1.255       255.255.255.255    연결됨           192.168.1.182      296
224.0.0.0          240.0.0.0          연결됨           127.0.0.1          331
224.0.0.0          240.0.0.0          연결됨           192.168.1.182      296
255.255.255.255    255.255.255.255    연결됨           127.0.0.1          331
255.255.255.255    255.255.255.255    연결됨           192.168.1.182      296
=====
영구 경로:
없음

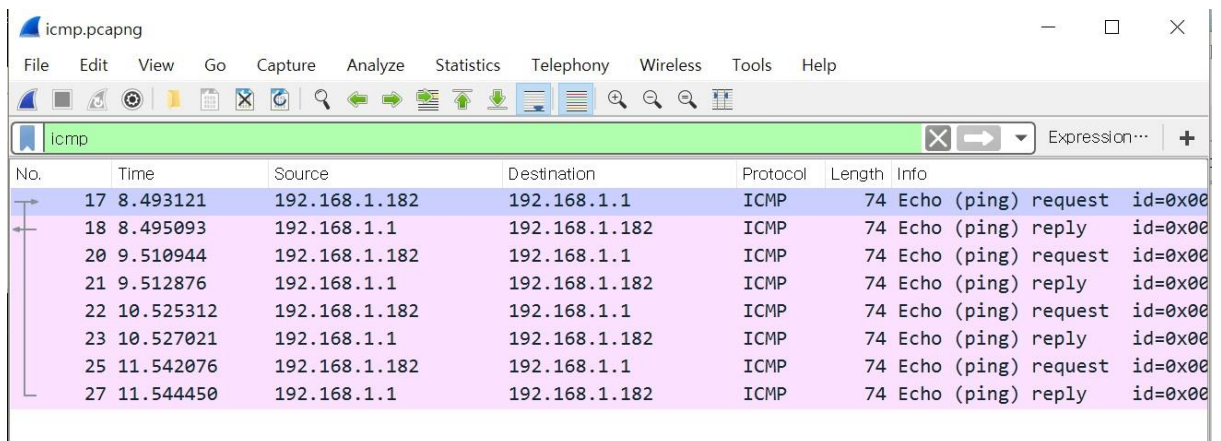
C:\Users\User>ping 192.168.1.1

Ping 192.168.1.1 32바이트 데이터 사용:
192.168.1.1의 응답: 바이트=32 시간=2ms TTL=64
192.168.1.1의 응답: 바이트=32 시간=2ms TTL=64
192.168.1.1의 응답: 바이트=32 시간=1ms TTL=64
192.168.1.1의 응답: 바이트=32 시간=2ms TTL=64

192.168.1.1에 대한 Ping 통계:
패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
왕복 시간(밀리초):
최소 = 1ms, 최대 = 2ms, 평균 = 1ms

```

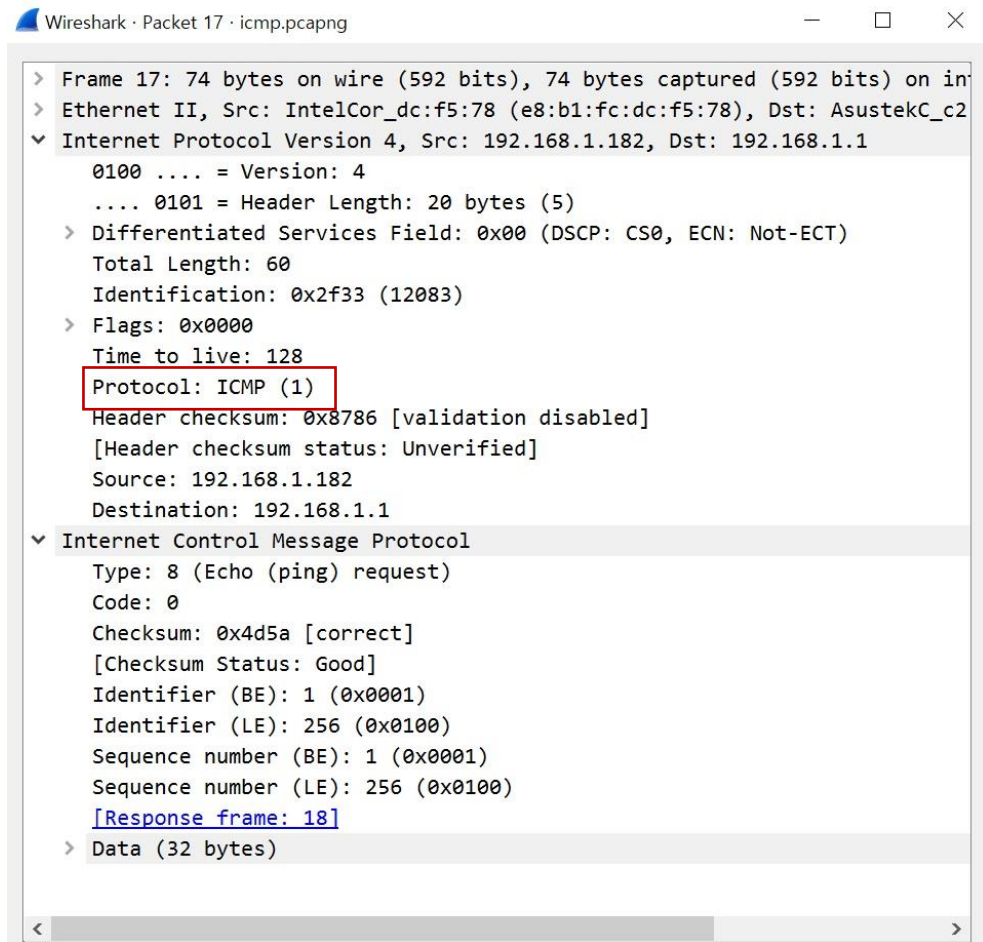
cmd에서 default gateway (192.168.1.1)에 대해 ping을 실행시키고 Wireshark를 이용해 아래와 같이 그 과정에서 교환된 ICMP Packet들을 캡처했다.



Ping을 실행했을 때 총 4번의 응답이 왔고 wireshark에선 총 8개의 ICMP packet이 캡처되었는데 차례대로 두 개씩 echo request, echo reply로 쌍을 이룬다. 각 한 쌍은 하나의 응답에 매치된다.

이 중 맨 위의 한 쌍을 살펴보자면 아래와 같다.

1) Packet No.17 (Echo request)



ICMP는 IP packet의 payload로서 담긴다. IP packet의 protocol필드의 값은 1로 payload의 protocol이 ICMP라는 것을 명시하고 있다. (빨간색 네모)

ICMP packet의 필드를 살펴보자면 우선 Type의 값은 8로 이 메시지가 Echo request query 메시지라는 것을 나타내며 echo request이기 때문에 code에 대한 값은 따로 필요하지 않아 이 필드는 0으로 채워져있다.

다음으로 살펴볼 것이 Identifier와 Sequence number인데 각 값은 아래와 같다.

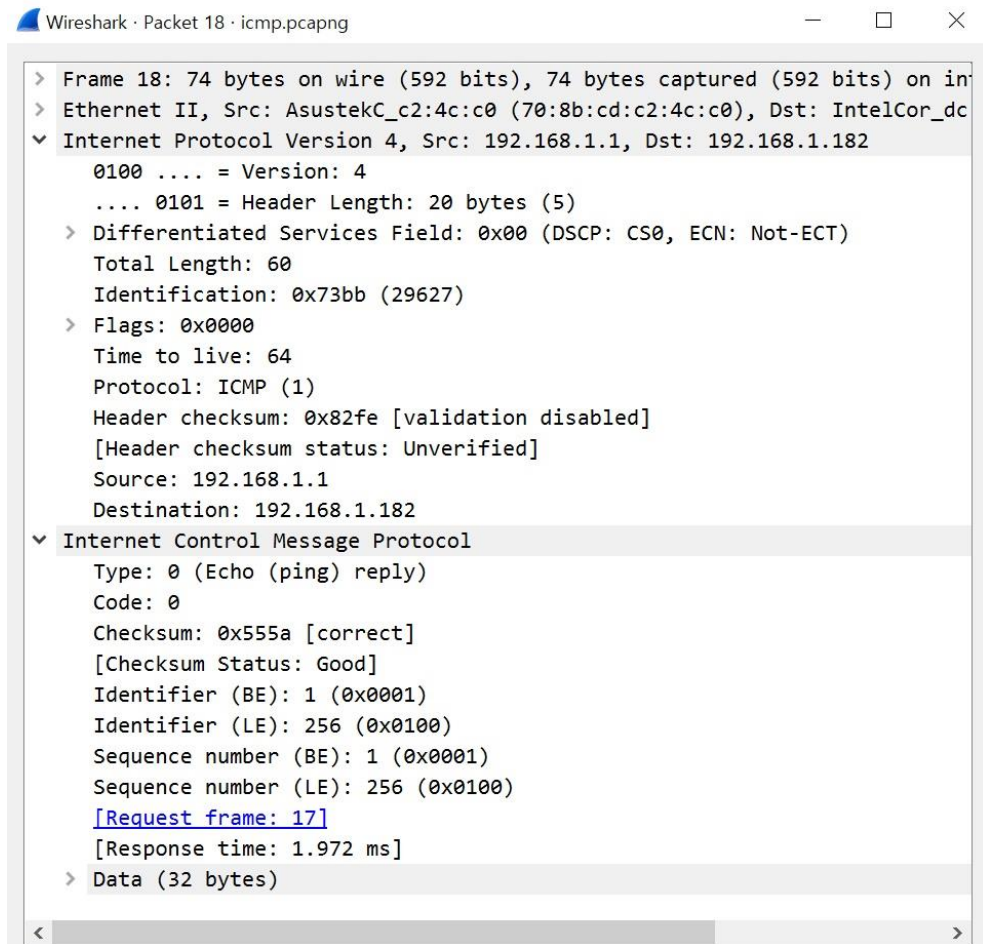
✓ Identifier: 1

✓ Sequence number: 1

Identifier는 request에 대한 reply 패킷을 구분하기 위하여 사용되고, sequence number는 같은 identifier에 대해 (반복되는 같은 작업에 대해) 부여되는 일련번호와 같은데, 이 두 필드 값은 모두 짝지어진 echo reply의 것과 같아야 한다.

이제 Packet No.18을 살펴보자

2) Packet No.18 (Echo reply)



Packet No.17에 바로 뒤이어 도착한 ICMP packet이다. Type 필드의 값은 0으로 해당 ICMP packet은 echo reply 메시지다.

Identifier와 Sequence number 필드의 값은 아래와 같다.

✓ Identifier: 1 (= Packet No.17's Identifier)

✓ Sequence number: 1 (= Packet No.17's Sequence number)

Identifier와 sequence number가 모두 packet No.17의 것과 일치하며 이는 이 packet이 packet No.17 echo request에 대응하는 echo reply 메시지라는 것을 나타낸다.

Wireshark에 캡처된 8개의 ICMP packet 중 나머지 3쌍의 identifier와 sequence number를 살펴보면 identifier는 모두 1로 같은 값을 나타내고 sequence number는 2, 3, 4로 1씩 늘어난다.