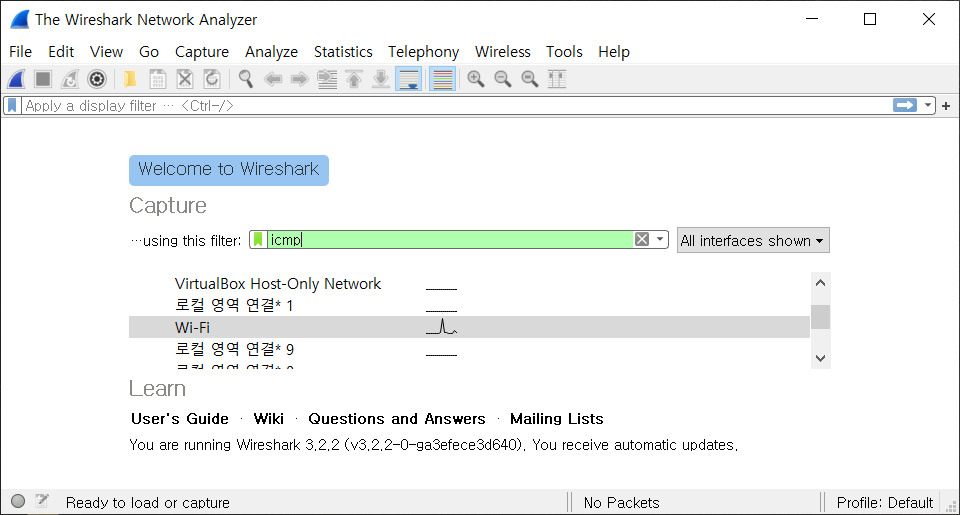
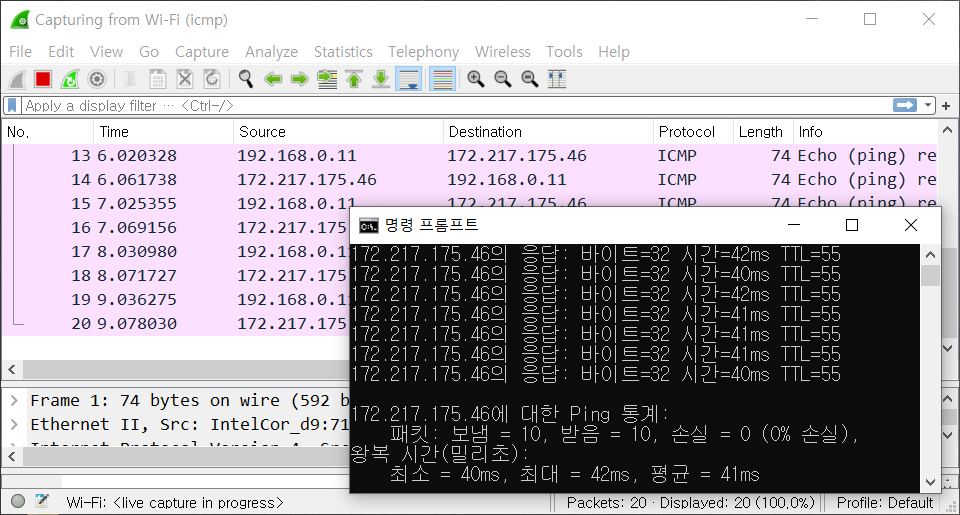
**인터넷 프로토콜 HW2**

12161756 윤성호

**< 과제 진행 환경 >**

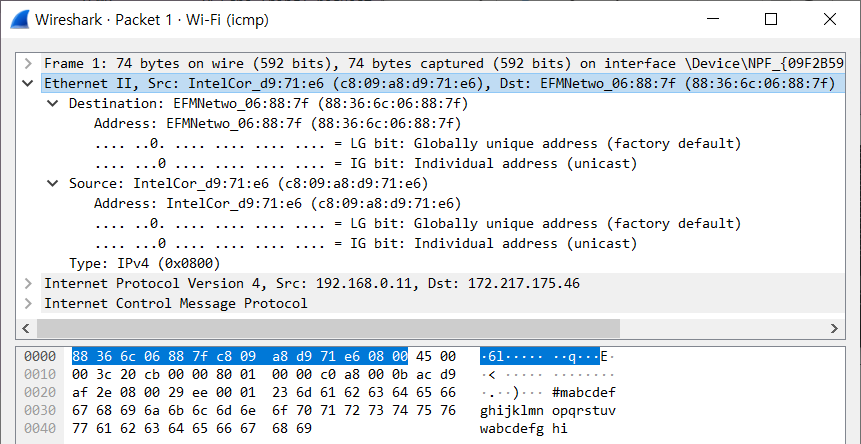


**= Wireshark(v\_3.2.2)를 실행한다. 현재 이용 중인 인터페이스(Wi-Fi)를 한 번 클릭 후 filter칸에 icmp를 입력하고 다시 Wi-Fi를 더블클릭 하여 Capture를 시작한다.**



**= [ping -n 10 google.com] 명령어를 실행하자 20개의 패킷이 Wireshark capture 창에 기록되는 것을 확인할 수 있다.**

1. 첫번째 Echo Request의 MAC frame 값들을 분석하여라

****

= Capture 화면에서 첫 번째 Echo Request를 더블 클릭하자 위와 같은 창이 나타났다.

= 위 캡쳐에서 파란색으로 블록이 잡힌 부분이 MAC frame 값들이다.

**1) Destination address (88:36:6c:06:88:7f)**

: 패킷의 가장 앞 부분에 있으며 6bytes를 차지한다. 주소 첫 1byte 부분(88)을 bit로 표기하면 10001000이다. 여기서 7번째와 8번째 bit가 0이므로 globally unique & individual address이다.

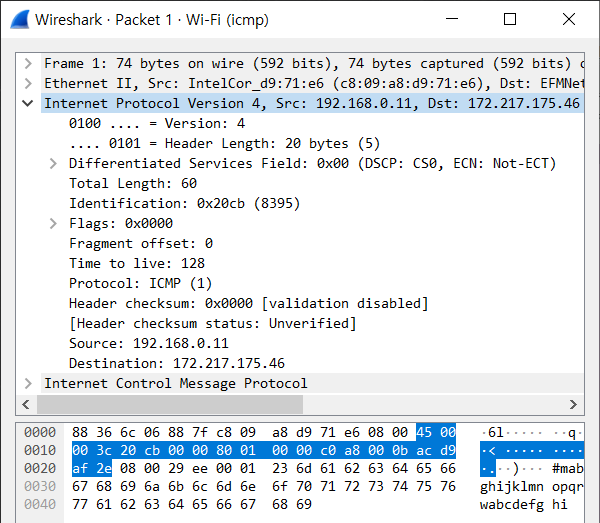
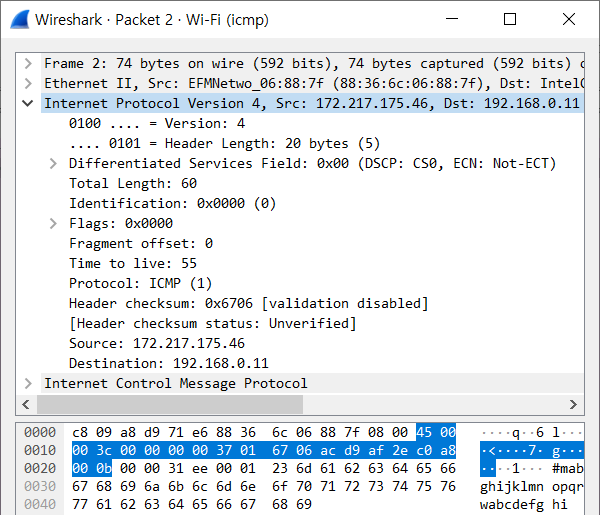
**2) Source address (c8:09:a8:d9:71:ef)**

: 6bytes를 차지하며 마찬가지로 첫 1byte 부분(c8)을 bit로 표기시 11001000이므로 globally unique & individual address이다.

**3) Type (0x0800[IPv4])**

: 2bytes를 차지하며, 0x0800값은 뒤에 이어질 Payload에 실린 Protocol이 IPv4임을 나타낸다.

2. IP 데이터그램을 주고 받을 때 값의 변화가 있는 필드는 무엇인가? 어떻게 값이 바뀌고 있는가? (IP헤더의 내용)

**** 

= 위 캡처에서 왼쪽 것이 request, 오른쪽 것이 reply이며 블록 잡힌 부분이 IP헤더 부분이다.

**= 변화 O : 식별자, TTL, 헤더 체크섬, 송신측 IP 주소, 수신측 IP 주소**

= 변화 X : 버전, 헤더길이, 서비스 타입, 전체 길이, 플래그, 단편화 옵셋, 프로토콜

**1) 식별자 (0x20cb -> 0x0000 -> 0x20cc -> 0x0000 -> 0x20cd -> 0x0000 ……)**

: request(0x20XX)와 reply(0x0000)이 반복되었다. request만 정렬해서 보면 0x20cb에서 1씩 증가하였고 모든 reply는 0x0000으로 동일하였다. 최초 request의 송신 측에서는 패킷이 분할될 일이 없기 때문에 request값이 각기 다른 것으로 해석된다.

**2) TTL (request : 0x80[128] / reply : 0x37[55])**

: request측은 현재 이용 중인 컴퓨터로, Windows 운영체제를 이용하고 있기에 TTL이 128로 나타났다. reply값을 통해 google.com에서는 최초에 TTL이 64인 ip datagram을 보냈고 9개의 라우터를 거쳐 현재 컴퓨터에 도착한 것으로 보인다.

**3) 헤더 체크섬 (request : 0x0000 / reply : 0x6706)**

: request(0x0000)와 reply(0x6706)이 반복되어 나타났다. [validation disabled]라고 표기된 것으로 보아 검증 기능이 비활성화된 것으로 보인다.

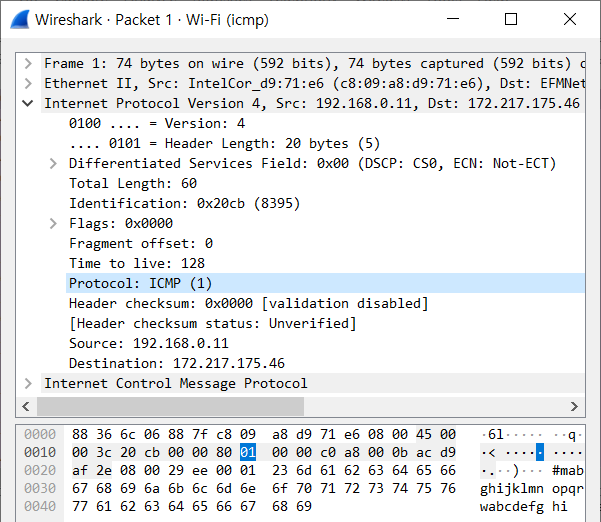
**4) 송신측 IP 주소 (request : 0xc0a8000b[192.168.0.11] / reply : 0xacd9af2e[172.217.175.46)**

: request에서는 내 IP가 표기됐고 reply에서는 google.com의 IP가 표기됐다.

**5) 수신측 IP 주소 (request : 0xacd9af2e[172.217.175.46 / reply : 0xc0a8000b[192.168.0.11])**

: request에서는 google.com의 IP가, reply에서는 내 IP가 표기됐다.

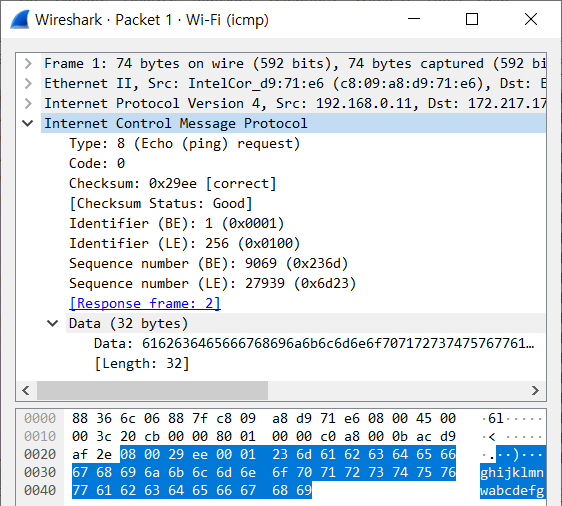
3. IP 헤더의 프로토콜 필드는 IP 데이터그램이 전달하는 데이터가 어떤 상위 계층의 데이터인지를 나타낸다. ICMP는 프로토콜 필드의 값이 무엇인가?

****

= 이 실습에서 IP 데이터그램은 ICMP 데이터를 전달하기 때문에 IP 헤더의 프로토콜 필드에는 ICMP를 나타내는 값이 표기된다.

= 따라서 **0x01**(1)이 ICMP를 나타내는 프로토콜 필드 값임을 알 수 있다.

4. ICMP 패킷의 포맷을 Wireshark 화면에 나와있는 값을 이용하여 그려 보라. 수업시간에 배운바와 같이 패킷 포맷을 그린다.

= 첫 번째 request 패킷을 토대로 그렸으며 **한 칸을 1byte**로 표현했다.

5. 여러 개의 ICMP Echo Request/Reply 패킷들이 교환될 때 값이 틀려지는 부분이 있다면 어떤 것 인지 해석해 보아라.

**= 변화 O : 메시지 타입, 체크섬, Sequence number(메시지 의존 부분)**

= 변화 X : 코드, Identifier(메시지 의존 부분), 메시지 정보

**1) 메시지 타입 (request : 0x08 / reply : 0x00)**

: 메시지 타입은 ICMP 메시지의 종류를 기술한다. request를 나타내는 8과 reply를 나타내는 0이 패킷 각 메시지의 종류에 맞게 표기되었다.

**2) 체크섬 (request : 0x29ee~0x29e5 / reply : 0x31ee~0x31e5)**

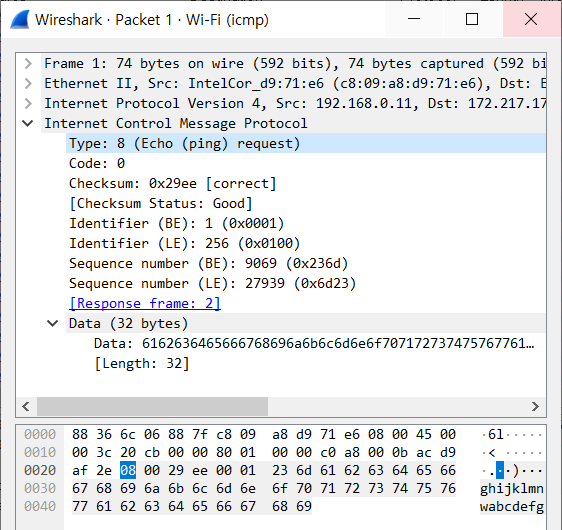
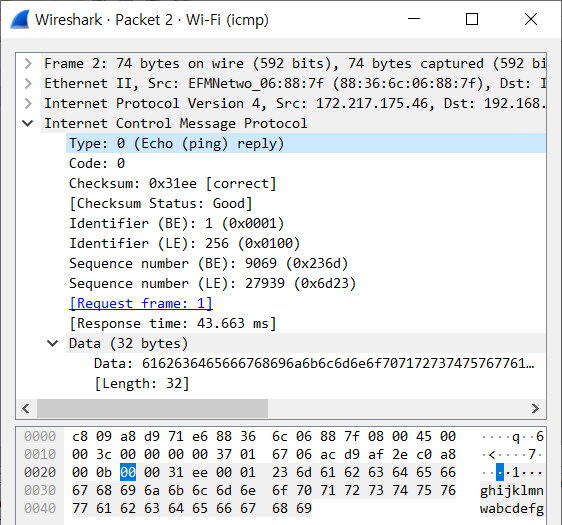
: 체크섬은 ICMP 메시지에 대해 체크섬을 계산한 값이 삽입된다. 첫 번째 request에서는 0x29ee이였고 다음 request부터는 1씩 감소한 값이 표기되었다. reply에서는 0x31ee에서 1씩 감소한 값이 표기되었다. 20개의 패킷 모두 각각의 패킷 값들이 다르기 때문에 체크섬값 또한 모두 다르다.

**3) Sequence number (0x236d -> 0x236d -> 0x236e -> 0x236e -> 0x236f -> 0x236f ……)**

: 메시지 의존 부분 중 뒤쪽 4bytes 값인 Sequence number은 request-reply 한 쌍씩 서로 같은 값을 가졌다.

6. ICMP echo request의 type와 code값은 무엇인가?

7. ICMP echo reply의 type와 code값은 무엇인가?

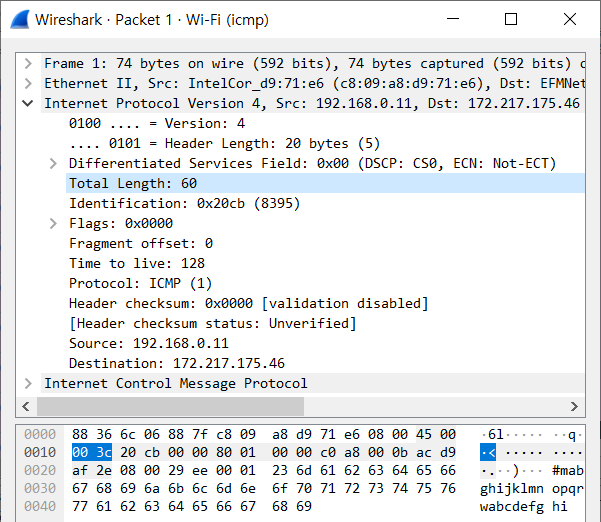
 

= 왼쪽이 request 캡쳐 이미지이다. **ICMP echo request type = 8, code = 0**임을 알 수 있다.

= 오른쪽은 reply 캡쳐 이미지이다. **ICMP echo reply type = 0, code = 0**임을 알 수 있다.

8. ICMP 패킷의 전체 길이는 얼마인가?

= ICMP 패킷은 IP 패킷의 header 뒤에 오는 data에 위치한다. 따라서 IP 패킷의 필드 중 Total Length값에서 Header Length값을 빼면 구할 수 있다.



= ICMP 패킷의 전체 길이는 60 – 20 = **40 bytes**임을 알 수 있다.