

## 자체 평가표

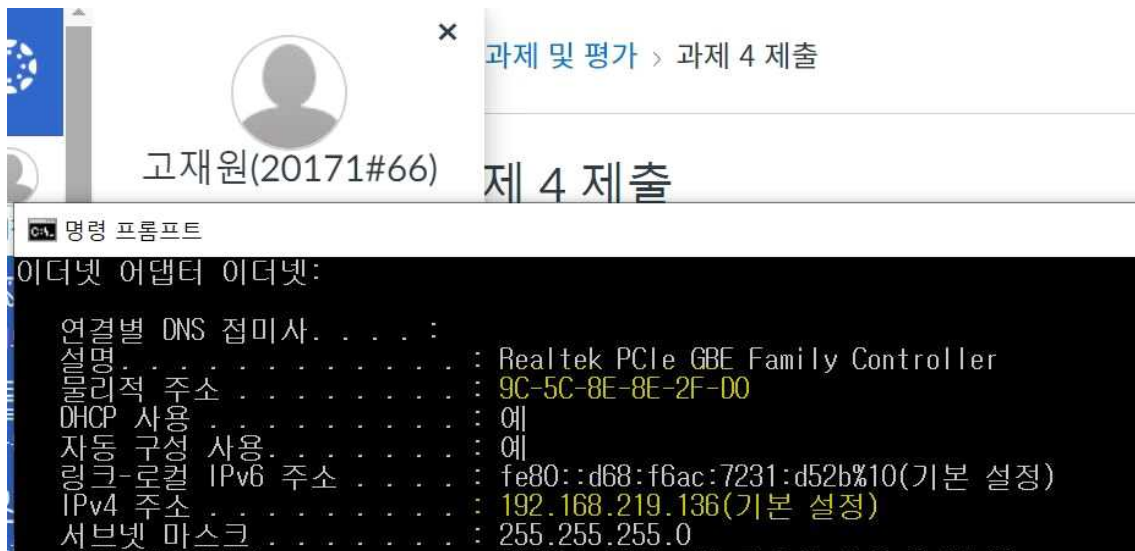
평가 항목	학생 자체 평가	평가 (빈 칸)
<p>LAN 내부 통신 ::</p> <ul style="list-style-type: none"> <li>- 실험 시나리오 분명하게 설명/표시</li> <li>- IP 수준이 아닌 MAC 수준에서 전달과정을 분석 및 확인하였는가?</li> <li>* 실제 전달되는 패킷의 + Dest IP 주소?</li> <li>+ Dest MAC 주소?</li> <li>- 파악된 MAC 주소에 따라 전송 과정 설명</li> <li>- ARP 동작 과정을 분석할 필요 없음. (각 노드의 MAC 주소 결과만 명기하면 됨)</li> </ul>	<p>1. 모든 과정을 실험 시나리오 &gt; 테스트 과정 &gt; 분석 &amp; 해석 순으로 작성하였습니다.</p> <p>2. 실제 전달을 MAC 수준(Ethernet header)에서 확인하였으며, 패킷의 Dest MAC, IP주소 확인 또한 ‘분석 &amp; 결과’에 명시하였습니다.</p> <p>3. 또한 MAC 주소에 따라 패킷의 전송 과정을 설명하였으며, 필요 없는 부분은 표기 후 생략하였습니다.</p>	
<p>LAN 외부 통신 ::</p> <ul style="list-style-type: none"> <li>- 실험 시나리오 분명하게 설명/표시</li> <li>- IP 수준이 아닌 MAC 수준에서 전달과정을 분석 및 확인하였는가?</li> <li>* 실제 전달되는 패킷의 + Dest IP 주소?</li> <li>+ Dest MAC 주소?</li> <li>- 파악된 MAC 주소에 따라 전송 과정 설명</li> </ul>	<p>1. 모든 과정을 실험 시나리오 &gt; 테스트 과정 &gt; 분석 &amp; 해석 순으로 작성하였습니다.</p> <p>2. 실제 전달을 MAC 수준(Ethernet header)에서 확인하였으며, 패킷의 Dest MAC, IP주소 확인 또한 ‘분석 &amp; 결과’에 명시하였습니다.</p> <p>3. 또한 MAC 주소에 따라 패킷의 전송 과정을 설명하였으며, 필요 없는 부분은 표기 후 생략하였습니다.</p>	
<p>TTL 만료 실험 ::</p> <ul style="list-style-type: none"> <li>- TTL 만료로 버려지는 트래픽 생성 방법</li> <li>- 수신자는 이를 어떻게 아는가?</li> <li>* 단순 응답 없음이 아니라</li> <li>* TTL 만료를 아는 방법을</li> <li>* 패킷 분석으로 파악</li> <li>- MAC 수준은 필요 없음</li> <li>- IP/ICMP 패킷 내용을 분석</li> </ul>	<p>1. 모든 과정을 실험 시나리오 &gt; 테스트 과정 &gt; 분석 &amp; 해석 순으로 작성하였습니다.</p> <p>2. TTL 만료로 버려지는 트래픽이 생기는 시나리오를 명시, 실험하였습니다.</p> <p>3. IP/ICMP수준에서의 패킷 분석을 통해 TTL 만료를 아는 방법을 분석 &amp; 해석에 명시하였습니다.</p>	
총평	<p>과제4를 하면서 추상적으로만 알던 개념이 구체적으로 이해되었습니다.</p> <p>그동안 컴퓨터 통신을 가르쳐 주셔서 감사합니다.</p>	

# WireShark를 이용한 네트워크에서의 IP패킷 전달 분석

20171666 고재원

시작하기에 앞서...

- 작성자의 PC MAC, IP주소 정보 :



- 작성자의 핸드폰 MAC, IP주소 정보 :

MAC 주소 : D8:55:75:09:06:3C  
IP 주소 : fe80::da55:75ff:fe09:63c  
192.168.219.120

## 1) LAN 내부에서의 통신

### 1-1) 실험 시나리오

- 현재 작성자의 PC(L2:Ethernet)와 휴대폰(L2:Wi-Fi)은 공유기를 통해 하나의 인터넷 주소를 공유 중.
- 따라서 같은 LAN내에서, 작성자의 PC에서 휴대폰의 사설 IP주소로 반복적으로 ping을 쏘아 통신을 하며, 이때의 상황을 wireshark을 이용하여 포착.

### 1-2) 테스트 과정

- IP주소 확인을 통해 같은 LAN 내부에 있음을 확인.
  - 서브넷 마스크(255.255.255.0)를 이용하여 각 단말기의 네트워크 주소가 192.168.219.0임을 파악, 같은 네트워크 주소를 가지므로 작성자의 PC와 휴대폰은 같은 LAN에 속함을 확인.

- PC와 핸드폰의 통신 ( ping )

```
C:\WINDOWS\system32>ping 192.168.219.120 -t

Ping 192.168.219.120 32바이트 데이터 사용:
192.168.219.120의 응답: 바이트=32 시간=210ms TTL=64
192.168.219.120의 응답: 바이트=32 시간=159ms TTL=64
192.168.219.120의 응답: 바이트=32 시간=219ms TTL=64
192.168.219.120의 응답: 바이트=32 시간=219ms TTL=64
192.168.219.120의 응답: 바이트=32 시간=226ms TTL=64
```

- 이 상황을 wireshark로 포착 및 필터링 (ping이 ICMP를 사용하므로)

icmp						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.219.136	192.168.219.120	ICMP	74	Echo (ping) request
2	0.186662	192.168.219.120	192.168.219.136	ICMP	74	Echo (ping) reply
4	1.031795	192.168.219.136	192.168.219.120	ICMP	74	Echo (ping) request
5	1.207479	192.168.219.120	192.168.219.136	ICMP	74	Echo (ping) reply
8	2.062248	192.168.219.136	192.168.219.120	ICMP	74	Echo (ping) request
11	2.231316	192.168.219.120	192.168.219.136	ICMP	74	Echo (ping) reply

### 1-3) 분석 & 결과

- 패킷 분석 : 캡처 된 패킷의 구성은 다음과 같다.



- Ethernet 헤더 분석 : MAC 주소 ( Source & Destination )
  - Dst MAC주소가 핸드폰의 MAC주소와 일치함을 알 수 있다.

```
Wireshark · Packet 4 · 이더넷

> Frame 4: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interf
v Ethernet II, Src: ASUSTekC_8e:2f:d0 (9c:5c:8e:8e:2f:d0), Dst: SamsungE_09:06
  > Destination: SamsungE_09:06:3c (d8:55:75:09:06:3c)
  v Source: ASUSTekC_8e:2f:d0 (9c:5c:8e:8e:2f:d0)
    Address: ASUSTekC_8e:2f:d0 (9c:5c:8e:8e:2f:d0)
      .... ..0. .... .. = LG bit: Globally unique address (factory
      .... ..0. .... .. = IG bit: Individual address (unicast)

0000  d8 55 75 09 06 3c 9c 5c 8e 8e 2f d0 08 00 45 00  ·Uu·<·\ ··/···E·
0010  00 3c 02 44 00 00 80 01 00 00 c0 a8 db 88 c0 a8  ·<·D···· ······
0020  db 78 08 00 3b 48 00 01 12 13 61 62 63 64 65 66  ·x·;H· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69                    wabcdefg hi
```

- IPv4 헤더 분석 : IP 주소 ( Source & Destination )
  - dst IP, src IP주소 또한 각각 휴대폰, PC의 IP주소와 일치함을 확인.

```
Wireshark · Packet 4 · 이더넷

Time to Live: 128
Protocol: ICMP (1)
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.219.136
Destination Address: 192.168.219.120
> Internet Control Message Protocol

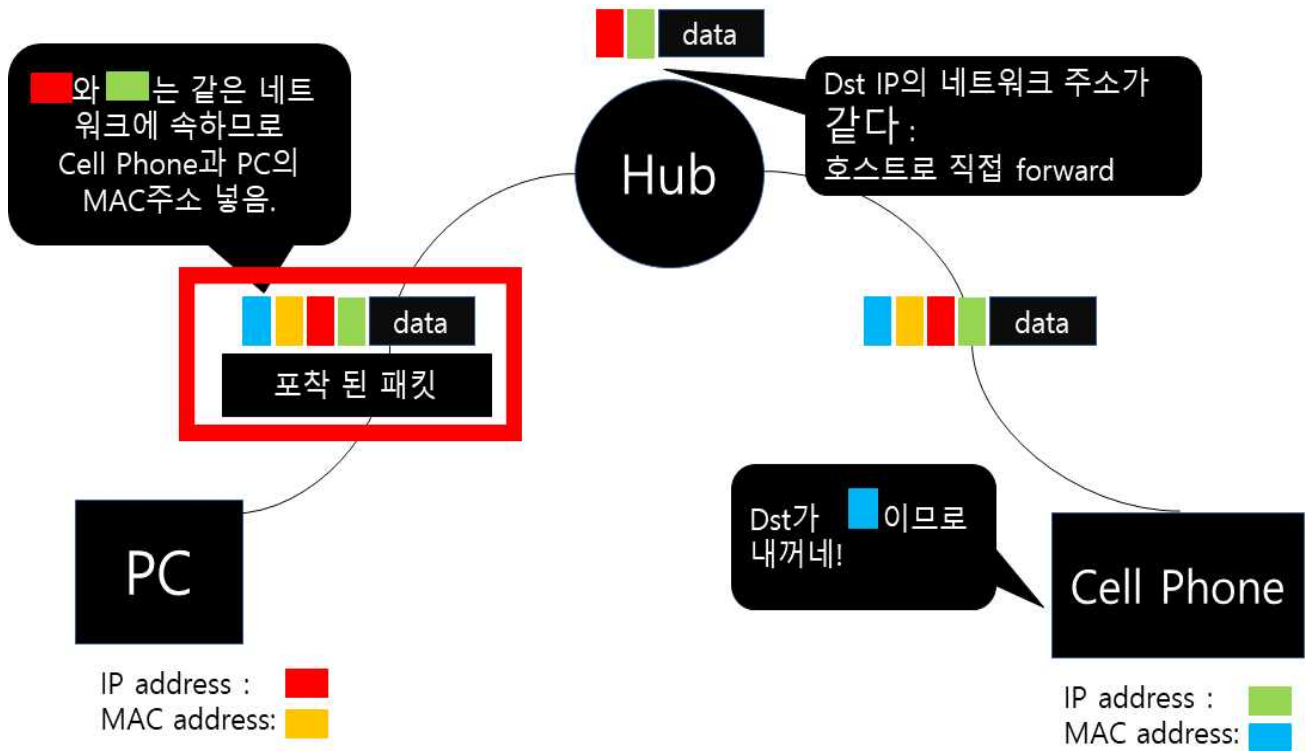
0000  d8 55 75 09 06 3c 9c 5c 8e 8e 2f d0 08 00 45 00  ·Uu·<·\ ··/···E·
0010  00 3c 02 44 00 00 80 01 00 00 c0 a8 db 88 c0 a8  ·<·D···· ······
0020  db 78 08 00 3b 48 00 01 12 13 61 62 63 64 65 66  ·x·;H· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69                    wabcdefg hi
```

■ 패킷 분석에 따른 전송 과정 [ ARP, DHCP 과정 생략 ]

※ 교재의 그림과 다르게 wireshark상에 포착된 패킷에서는 dest MAC address가 src MAC address보다 앞에 위치하여 이와 같게 표현하였습니다.

즉 헤더 표현 : [dst MAC | src MAC | src IP | dst IP ]

[ PC -> Cell phone, 패킷을 헤더+데이터로만 표현 ]



## 2) LAN 외부에서의 통신

### 2-1) 실험 시나리오

- 작성자의 PC(L2: Ethernet)와 Google의 홈페이지를 운영하는 서버는 서로 다른 network에 속함.
- 작성자의 PC에서 Google의 홈페이지에 핑을 쏘아 통신을 하며, 이때의 상황을 wireshark을 이용하여 포착 및 분석.

### 2-2) 테스트 과정

- Google 서버로의 통신(Ping test)

```
Ping www.google.com [172.217.174.196] 32바이트 데이터 사용:
172.217.174.196의 응답: 바이트=32 시간=79ms TTL=106
172.217.174.196의 응답: 바이트=32 시간=81ms TTL=106
172.217.174.196의 응답: 바이트=32 시간=80ms TTL=106
172.217.174.196의 응답: 바이트=32 시간=79ms TTL=106
172.217.174.196의 응답: 바이트=32 시간=81ms TTL=106
172.217.174.196의 응답: 바이트=32 시간=80ms TTL=106
```

- 이때의 상황을 wireshark로 포착 및 필터링

\*이더넷

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Info
48	8.157581	192.168.219.136	172.217.174.196	ICMP	74	Echo (ping) request
50	8.237022	172.217.174.196	192.168.219.136	ICMP	74	Echo (ping) reply
52	9.188481	192.168.219.136	172.217.174.196	ICMP	74	Echo (ping) request
53	9.269448	172.217.174.196	192.168.219.136	ICMP	74	Echo (ping) reply
54	10.216439	192.168.219.136	172.217.174.196	ICMP	74	Echo (ping) request
56	10.297055	172.217.174.196	192.168.219.136	ICMP	74	Echo (ping) reply

## 2-3) 분석 & 결과

- 패킷 분석 : 캡처 된 패킷의 구성은 다음과 같다.



- Ethernet 헤더 분석 : MAC 주소 ( Source & Destination )
  - LAN 내부에서의 통신과 달리 Destination의 MAC주소가 허브의 MAC 주소이다.

```
Wireshark - Packet 121 - 이더넷
> Frame 121: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{4CF4C...
> Ethernet II, Src: ASUSTekC_8e:2f:d0 (9c:5c:8e:8e:2f:d0), Dst: Mercury_1d:bb:8d (08:5d:dd:1d:bb:8d)
> Internet Protocol Version 4, Src: 192.168.219.136, Dst: 142.250.207.68
> Internet Control Message Protocol

0000  08 5d dd 1d bb 8d 9c 5c 8e 8e 2f d0 08 00 45 00  .].....\ ..../...E.
0010  00 3c e3 be 00 00 80 01 00 00 c0 a8 db 88 8e fa  .<.....
0020  cf 44 08 00 3a e2 00 01 12 79 61 62 63 64 65 66  .D...:... .yabcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69                    wabcdefg hi
```

- 작성자의 허브 IP, MAC주소

```
C:\WINDOWS\system32>arp -a

인터페이스: 192.168.219.136 --- 0xa
인터넷 주소      물리적 주소      유형
192.168.219.1     08-5d-dd-1d-bb-8d  동적
```



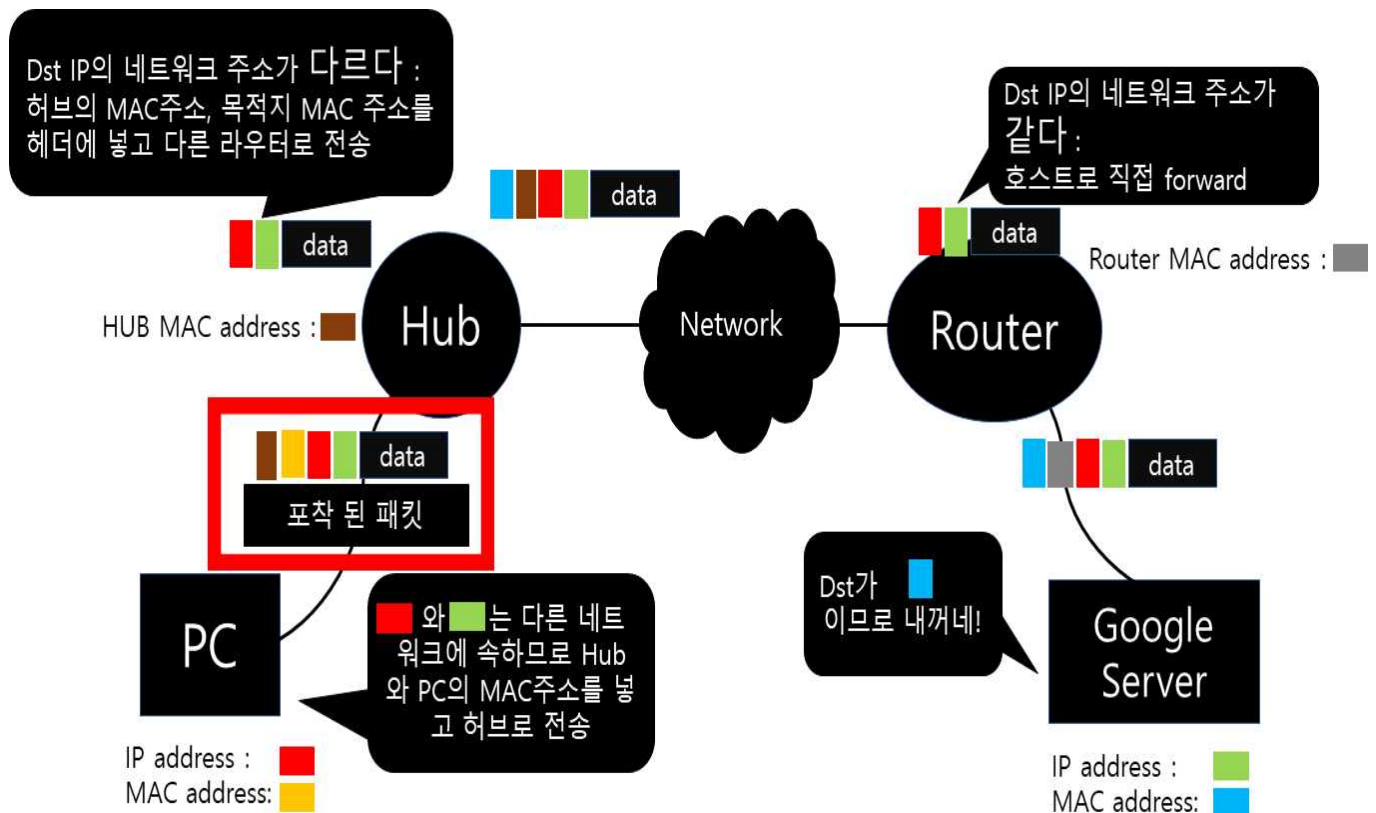
■ IPv4 헤더 분석 : IP주소 ( Source & Destination )

- 하지만 DST IP주소는 허브의 주소가 아닌 구글 서버의 주소이므로 MAC 수준에서 분석을 해야 한다.

Time to Live: 128		
Protocol: ICMP (1)		
Header Checksum: 0x0000 [validation disabled]		
[Header checksum status: Unverified]		
Source Address: 192.168.219.136		
Destination Address: 142.250.204.132		
> Internet Control Message Protocol		
<		
0000	08 5d dd 1d bb 8d 9c 5c 8e 8e 2f d0 08 00 45 00	·]·····\··/·····E·
0010	00 3c 0e 82 00 00 80 01 00 00 c0 a8 db 88 8e fa	·<················
0020	cc 84 08 00 3a c3 00 01 12 98 61 62 63 64 65 66	·····:········abcdef
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67 68 69	wabcdefg hi

■ 패킷 분석에 따른 전송 과정 [ DNS, ARP, DHCP 과정 생략 ]

[ PC -> Google Server ]





### 3) TTL 만료 실험

#### 3-1) 실험 시나리오

- 실험2의 상황과 같이 작성자의 PC(L2:ethernet)에서 외부 network에 있는 구글 서버에 ping을 쏘는 상황.
- 이때, ping의 -i 옵션을 이용하여 TTL을 임의의 값으로 설정하여 TTL만료가 발생하게 한다. (실험에서는 10으로 설정)
- 이때의 상황을 wireshark로 포착 후 캡처 된 패킷을 통해 어떻게 PC에서 TTL 만료를 알아내는지 분석

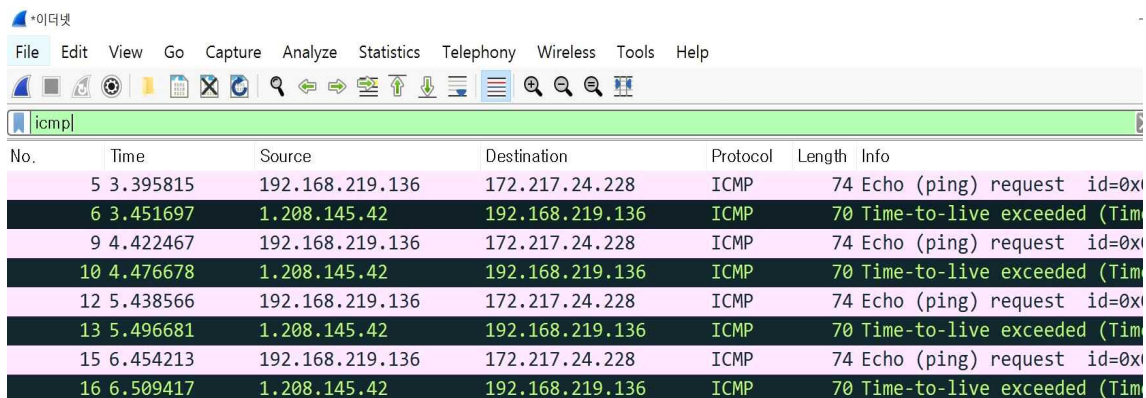
#### 3-2) 테스트 과정

- 작성자의 PC에서 TTL = 10으로 설정 후 구글 서버에게 핑을 쏜.

```
C:\WINDOWS\system32>ping -i 10 www.google.net

Ping www.google.com [172.217.24.228] 32바이트 데이터 사용:
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
```

- 이때의 상황을 wireshark로 캡처 및 필터링.



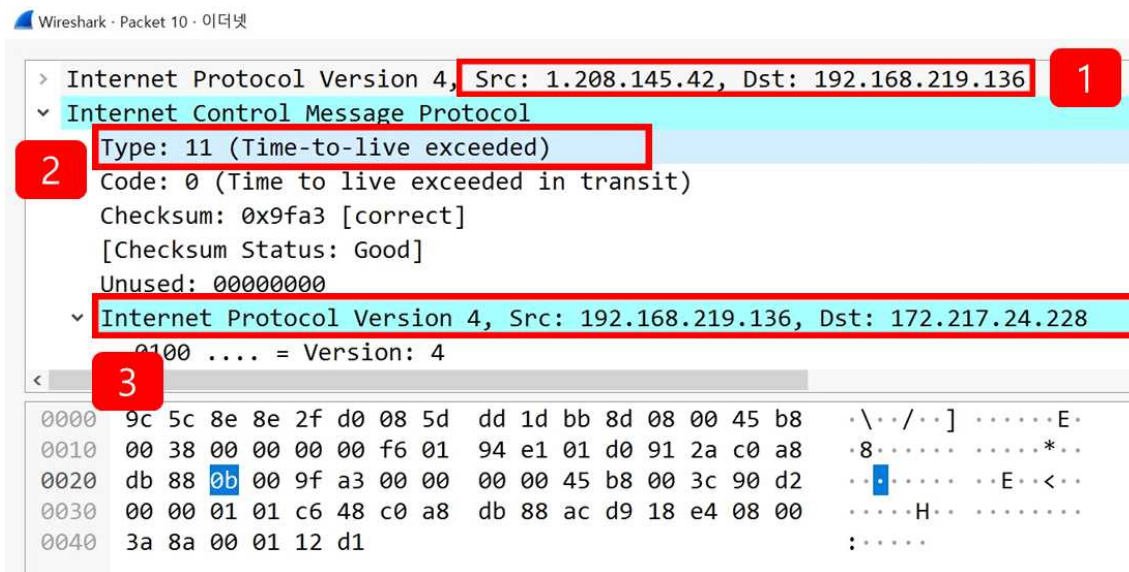
No.	Time	Source	Destination	Protocol	Length	Info
5	3.395815	192.168.219.136	172.217.24.228	ICMP	74	Echo (ping) request id=0x1
6	3.451697	1.208.145.42	192.168.219.136	ICMP	70	Time-to-live exceeded (Tim
9	4.422467	192.168.219.136	172.217.24.228	ICMP	74	Echo (ping) request id=0x1
10	4.476678	1.208.145.42	192.168.219.136	ICMP	70	Time-to-live exceeded (Tim
12	5.438566	192.168.219.136	172.217.24.228	ICMP	74	Echo (ping) request id=0x1
13	5.496681	1.208.145.42	192.168.219.136	ICMP	70	Time-to-live exceeded (Tim
15	6.454213	192.168.219.136	172.217.24.228	ICMP	74	Echo (ping) request id=0x1
16	6.509417	1.208.145.42	192.168.219.136	ICMP	70	Time-to-live exceeded (Tim

### 3-3) 분석 & 결과

- 패킷 분석 : 캡처 된 패킷의 구성은 다음과 같다.



#### ■ IPv4, ICMP 헤더 분석



1) 해당 패킷의 헤더에서 ICMP 헤더 부분을 보면 1.208.145.42.의 IP주소를 가진 라우터에서 작성자의 PC에게 패킷을 보냈음을 알 수 있다.

2) 해당 부분을 보면 그림에서와 같이 type 11의 에러가 일어났음을 저장하고 있다. 이를 통해 앞서 보낸 패킷에 TTL 만료 에러가 발생했음을 알 수 있다.

3) 하지만 1), 2)만으로는 작성자의 PC에서 ‘어떤 패킷’이 TTL만료가 일어났는지 확인이 불가능하다. 따라서 TTL만료가 발생한 라우터에서 추가 정보를 헤더에 저장하여 작성자의 PC에 전송하였음을 확인 할 수 있다.

(다음 페이지 상단 그림 참고)

Wireshark · Packet 10 · 이더넷

Unused: 00000000

Internet Protocol Version 4 Src: 192.168.219.136, Dst: 172.217.24.228

0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 > Differentiated Services Field: 0xb8 (DSCP: EF PHB, ECN: Not-ECT)  
 Total Length: 60  
 Identification: 0x90d2 (37074)  
 > Flags: 0x00  
 ...0 0000 0000 0000 = Fragment Offset: 0

TTL만료가 발생한 패킷의 목적지, 전송지 IP 주소

TTL만료가 발생한 패킷의 식별 값

0000	9c 5c 8e 8e 2f d0 08 5d dd 1d bb 8d 08 00 45 b8	·\·/·] ·····E·
0010	00 38 00 00 00 00 f6 01 94 e1 01 d0 91 2a c0 a8	·8····· ·····*·
0020	db 88 0b 00 9f a3 00 00 00 00 45 b8 00 3c 90 d2	····· ···E·<·
0030	00 00 01 01 c6 48 c0 a8 db 88 ac d9 18 e4 08 00	·····H· ·····
0040	3a 8a 00 01 12 d1	:····

또한 위의 식별자가 앞서 보낸 ping에 의해 만들어진 패킷의 식별자와 일치함을 알 수 있다. (하단 그림 참고) ( 좌: 보낸 패킷 우: 받은 패킷)

Wireshark · Packet 5 · 이더넷

0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 > Differentiated Services Field: 0xb8 (DSCP: EF PHB, ECN: Not-ECT)  
 Total Length: 60  
 Identification: 0x90d1 (37073)  
 > Flags: 0x00

Wireshark · Packet 6 · 이더넷

Internet Protocol Version 4, Src: 192.168.219.136, Dst: 172.217.24.228

0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes (5)  
 > Differentiated Services Field: 0xb8 (DSCP: EF PHB, ECN: Not-ECT)  
 Total Length: 60  
 Identification: 0x90d1 (37073)  
 > Flags: 0x00

따라서 1), 2), 3)의 정보를 통해 TTL만료 시 송신자가 이를 파악할 수 있다.

마지막으로,

이를 PC에서 확인하였다는 것을 앞서 보인 ping 테스트 결과로 알 수 있다.

```
C:\WINDOWS\system32>ping -i 10 www.google.net

Ping www.google.com [172.217.24.228] 32바이트 데이터 사용:
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
1.208.145.42의 응답: 전송하는 동안 TTL이 만료되었습니다.
```