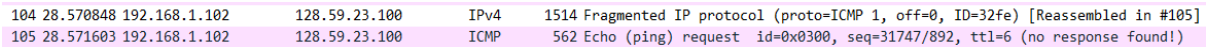
Packet 분석을 통한 Data Fragmentation의 이해

Task 1: 주어진 데이터에서 fragmentation 살펴보기

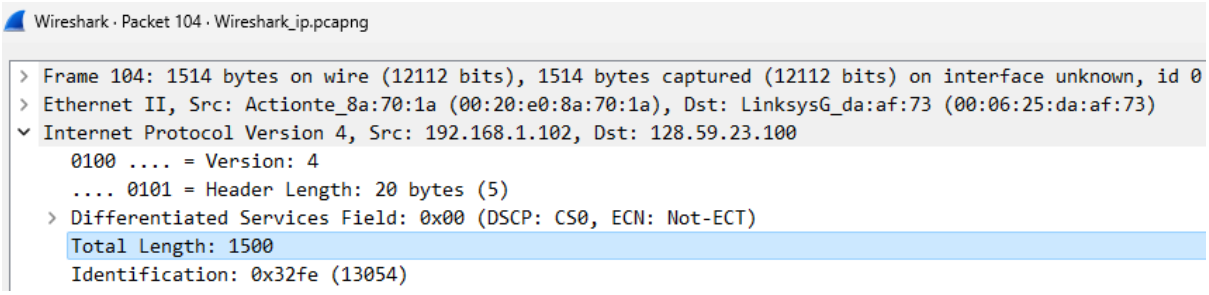
A)

“Wireshark_ip.pcapng” 파일을 살펴보니 104번과 105번 패킷을 발견할 수 있었다.

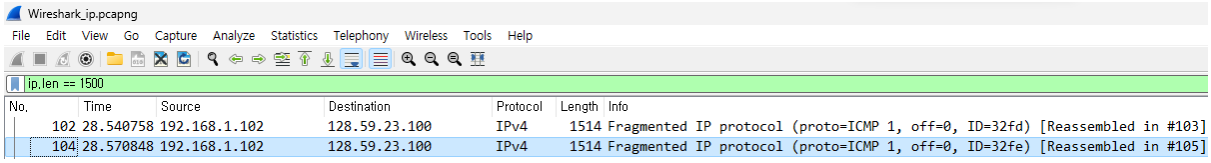


104번 패킷과 105번 패킷

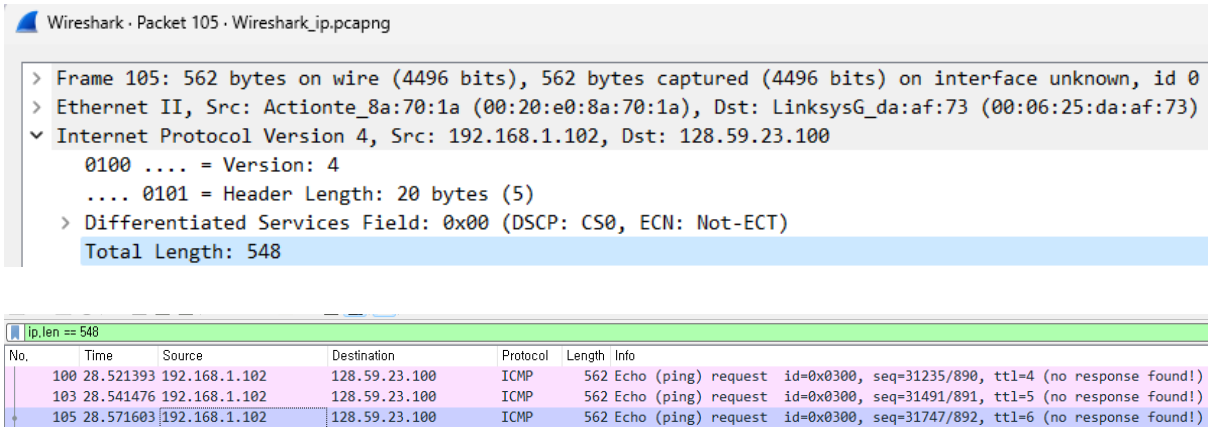
104번 패킷의 경우 ‘Total Length: 1500' 부분을 통해 IP datagram의 크기가 1500바이트임을 알 수 있었다.



정확한 확인을 위해 ‘ip.len == 1500’ 명령어를 통해 필터링 기능을 사용하여 크기가 1500바이트인 IP datagram을 가진 패킷만 출력해보았다. 102번 패킷 아래 104번 패킷이 위치한 것을 확인할 수 있었다.



105번 패킷의 경우 'Total Length: 548' 부분을 통해 IP datagram의 크기가 548바이트임을 알 수 있었다. 마찬가지로 'ip.len == 548' 명령어로 필터링을 시도해본 결과, 105번 패킷을 확인할 수 있었다.



B)

아래 사진들은 104번(좌측)과 105번 패킷(우측)의 일부이다. 아래 사진들을 통해 104번 패킷은 첫 번째 조각으로, 105번 패킷은 두 번째 조각으로 구성되어 있다는 것을 알 수 있었다.

원본 패킷이 두 조각을 나누어졌음을 확인한 후, 두 패킷을 합쳐 원래 패킷의 데이터 크기를 구할 수 있었다. 각 패킷 데이터 크기의 계산 과정은 아래와 같다.

- 아래 표에서 두 패킷의 IP Header 길이가 20 byte 인 이유

—> 사진 상으로는 동일하게 보이지만, 104번과 105번 패킷의 일부이다. 두 패킷 모두 Version 4의 IP Protocol을 사용하고 있으니, IP 헤더의 길이는 모두 20 바이트임을 알 수 있다.

```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
```

```
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.59.23.100
```

	104번	**105번**
Total Length	1500	548
Ip Header 길이	20	20
조각의 data 크기	1480 (= 1500 - 20)	528 (= 548 - 20)

따라서, fragmentation 전 원본 패킷의 데이터 크기는 1480 + 528 = 2008 바이트이다.

C)

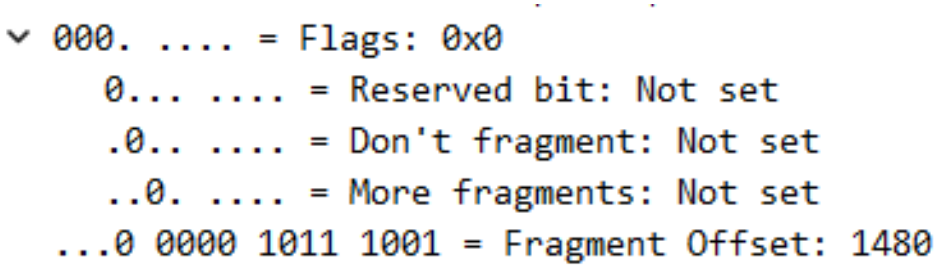
아래 사진은 104번 패킷의 IP datagram Header 부분이다. 나는 헤더 속 Flags 필드 부분에서 이 패킷이 fragmentation된 조각이라는 것을 짐작할 수 있었다. 104번 패킷의 IP datagram header를 보면, More fragments 비트가 1로 설정되어 있다. 이 비트는 fragmentation되었음을 나타내는 Flag 필드 중 하나로, 이 값이 1로 설정되어 있으면 이 데이터그램이 여러 개의 패킷으로 나뉘어져 전송된 것을 의미한다.

추가적으로 Don’t fragment 필드의 비트가 0으로 설정되어 있으니, fragmentation이 허용된 상태였음도 확인할 수 있었다.

```
▾ 001. .... = Flags: 0x1, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
  ...0 0000 0000 0000 = Fragment Offset: 0
```

아래 사진은 105번 패킷의 IP datagram Header 부분이다. 105번 패킷의 IP datagram header를 살펴보면, Fragment Offset 필드의 값이 1480으로 설정되어 있음을 확인할 수 있다. 이것은 이 데이터그램이 원본 데이터에서 1480바이트 이후부터 나눠진 조각임을 나타내며, 이를 통해 104번처럼 105번 패킷도 fragmentation된 상태의 패킷임을 알 수 있었다.

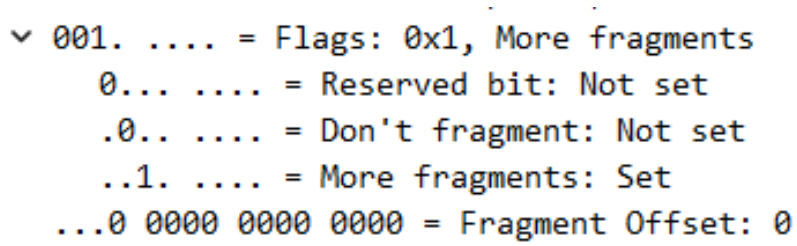
또 104번 패킷의 경우와 마찬가지로 Don't fragment 필드의 비트가 0으로 설정되어 있어 fragmentation이 허용된 상태였음도 확인할 수 있었다.



D)

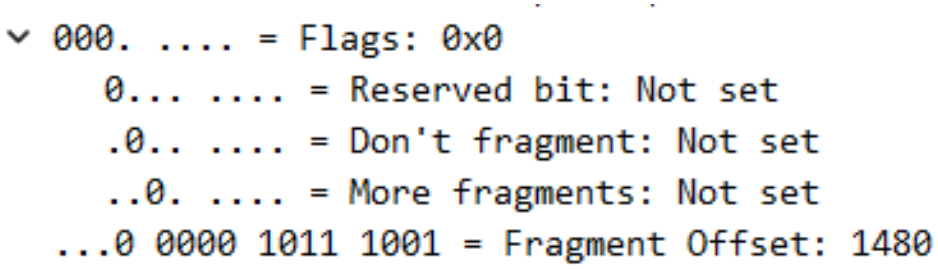
104번과 105번 중 104번 패킷이 첫 번째 fragmentation된 조각이다. 단편의 순서는 IP 데이터그램의 Fragment Offset 필드를 통해 확인할 수 있다. Fragment Offset 필드는 원본 데이터에서 IP 데이터그램이 시작하는 위치를 나타내는 필드로, 원본 데이터의 시작부터 IP 데이터그램이 시작하는 위치까지의 바이트 수를 의미한다.

아래는 104번 패킷의 IP datagram header 사진이다. 104번 패킷의 IP datagram header의 Fragment Offset 필드를 살펴보면 값이 0으로 설정되어 있는데, 이 값을 통해 이 데이터그램(즉, 104번 패킷)이 원본 데이터의 첫 번째 단편임을 알 수 있다.



E)

104번과 105번 외에 추가적인 단편은 존재하지 않으며, 이는 마지막 단편인 105번 패킷을 통해 알 수 있다. 아래 사진은 105번 패킷의 IP datagram 헤더 부분이다. 패킷의 IP datagram Header에서 More fragments 비트가 0으로 설정되어 있으므로, 더 이상의 단편이 없다는 것을 알 수 있다. 따라서 105번을 마지막으로 더 이상의 추가적인 단편은 존재하지 않는다.



F)

단편화(fragmentation)는 전송하려는 패킷의 크기가 네트워크 링크의 MTU(Maximum Transmission Unit)보다 큰 경우 발생한다. MTU는 링크에서 전송 가능한 최대 패킷 크기를 의미하며, 이보다 크기가 큰 패킷이 전송되면 네트워크 장비에서 패킷을 단편화하여 전송한다. 따라서, 단편화는 패킷의 크기와 MTU의 차이로 인해 발생하며, 이를 통해 전송 속도를 조절할 수 있다.

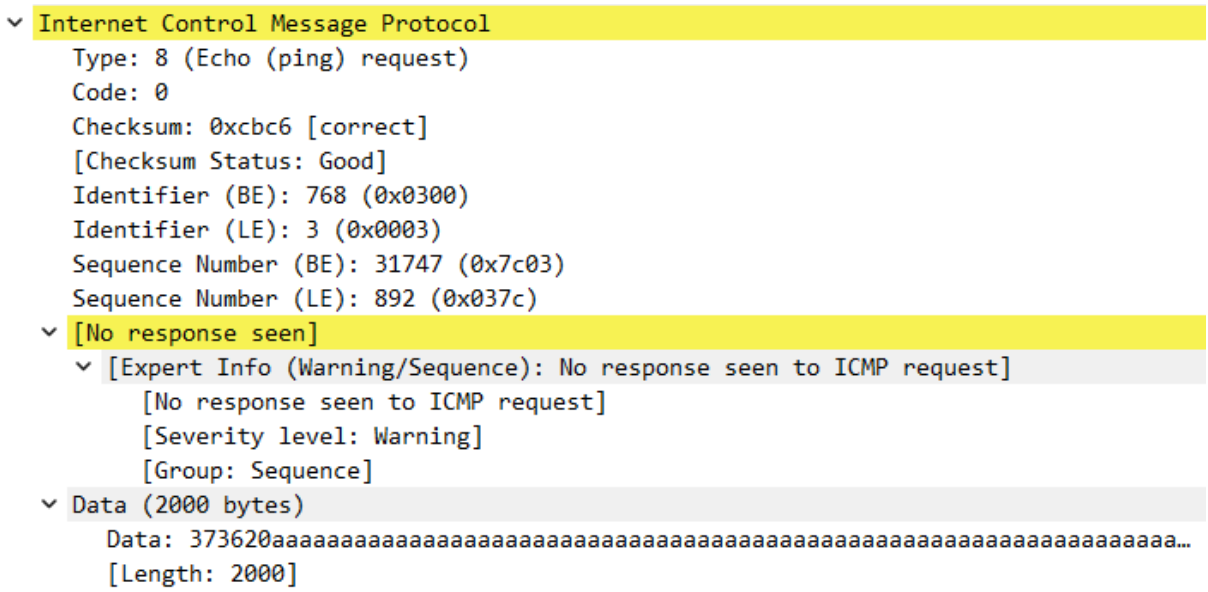
G)

여전히 IPv4 프로토콜을 사용하고 MTU가 1500바이트로 같다면, 데이터 크기가 3500바이트가 되는 경우의 변화는 아래와 같다.

	첫 번째 단편	두 번째 단편	세 번째 단편
IP 헤더	20 바이트	20 바이트	20 바이트
데이터	1480 바이트	1480 바이트	540 바이트

MTU가 1500바이트이니, 3500바이트의 데이터를 단편화할 경우 총 3개의 단편으로 나누어지게 된다. 세 단편 모두 IP 헤더는 20 바이트로 동일하니, 첫 번째 단편의 경우 최대로 가져갈 수 있는 데이터 크기인 1480바이트를 데이터 크기로 가지게 된다. 두 번째 단편도 마찬가지로 1480바이트를 데이터 크기로 가진다. 세 번째 단편의 경우 남은 540바이트를 데이터 크기로 가져가게 된다.

H)



105번 패킷의 일부

105번 패킷에서의 ICMP 메시지 타입은 8(Echo Request)으로, 이는 특정 호스트와의 통신을 진단하고자 할 때 사용하는 ping 명령어에서도 이용된다. Echo Request 메시지는 목적지 호스트로 전송되어 해당 호스트가 이를 수신하면 Echo Reply 메시지를 송신하는데, 이를 통해 네트워크의 연결 상태를 확인할 수 있다.

105번 패킷의 IP 헤더의 Time to Live (TTL) 값은 6으로, 이 값은 해당 패킷이 네트워크에서 라우터를 통과할 때마다 1씩 감소한다. TTL 값은 라우팅 루프와 같은 문제를 방지하기 위해 사용되며, 만약 6번째 포워딩을 거친 105번 패킷의 TTL 값이 0이 되면 105번 패킷은 폐기된다. 라우터는 TTL 값이 0이 된 105번 패킷을 폐기하고, ICMP (Internet Control Message Protocol) "Time Exceeded" 메시지를 송신측으로 보내어 해당 패킷이 폐기되었음을 알린다. 결론적으로 105번 패킷은 6개 이하의 라우터를 통과하여 목적지에 도달하여야 한다. 한편 해당 패킷에 대한 응답이 없으므로, Expert Info에서 해당 패킷에 대한 응답이 없음을 경고하고 있다.