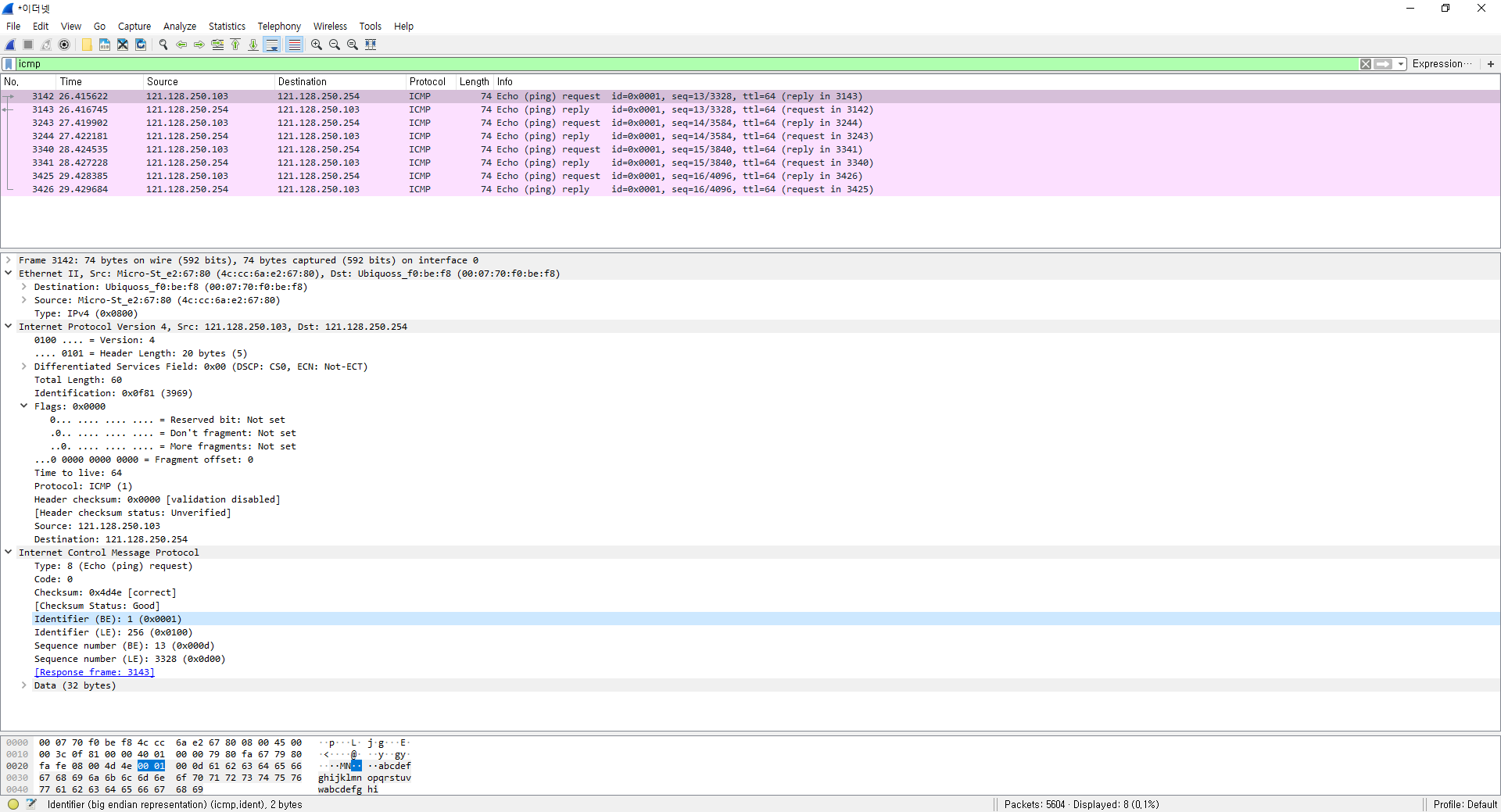
데이터통신 과제

2017320122 김정규

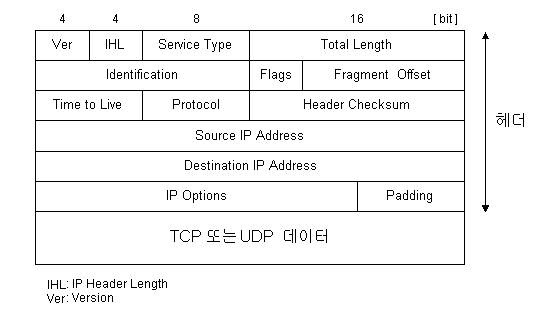
**#1 Ethernet 환경 (유선)에서 ping을 보낸 뒤의 response 받는 것까지 2개의 패킷에 대해 패킷 캡쳐 화면과 Ethernet (802.3) Header 분석**

<캡쳐화면>



<헤더 구조 분석>

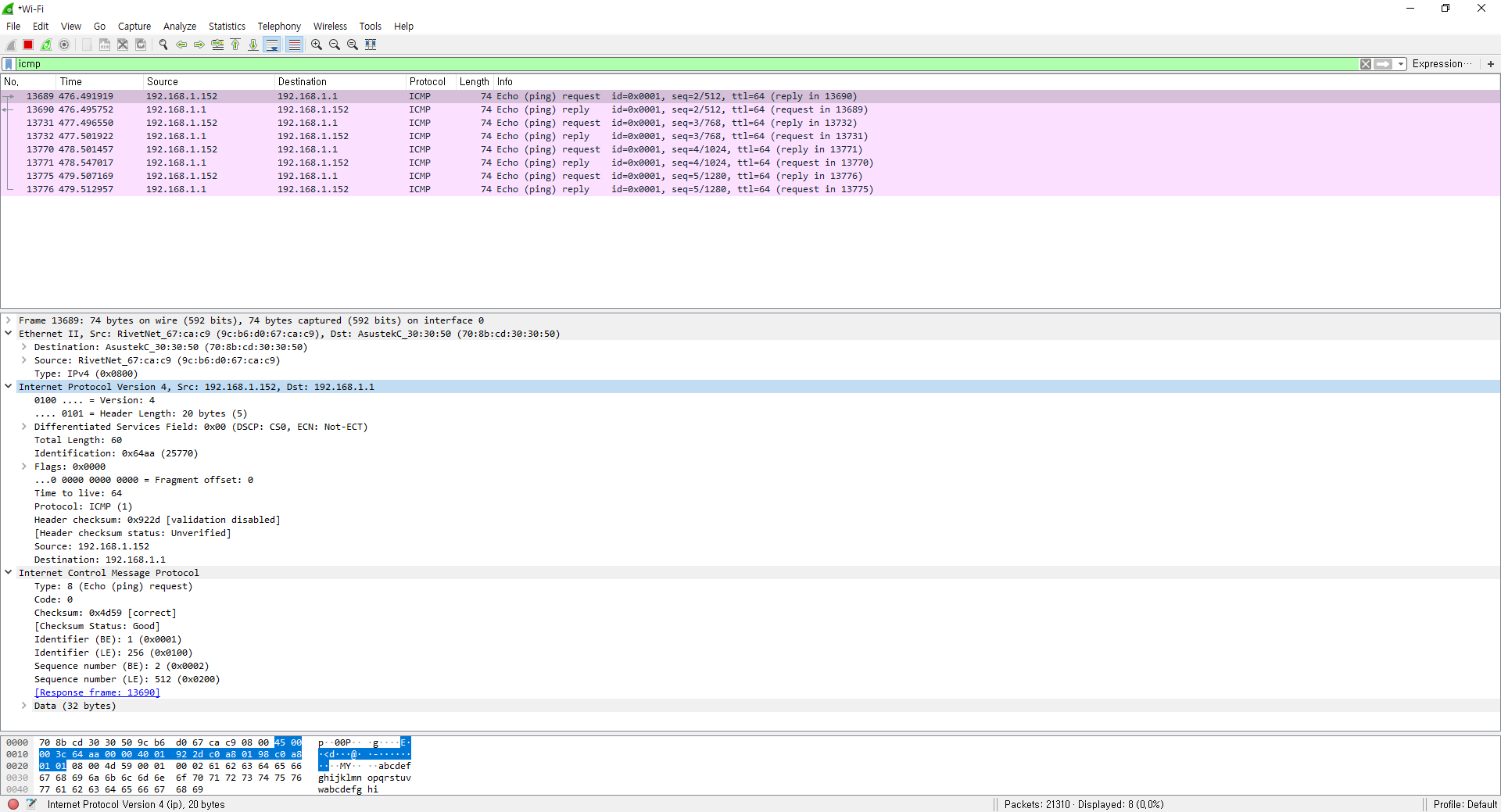
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **IHL** | | **TOS** | **Total Length** | |
| 0100 (0x4) | 0101 (0x5) | | 0000 0000 (0x00) | 0000 0000 0011 1100 (0x003c) | |
| **Identification** | | | | **flag** | **Fragment Offset** |
| 0000 1111 1000 0001 (0x0f81) | | | | 000 | 0 0000 0000 0000 |
| **TTL** | | **protocol** | | **Header Checksum** | |
| 0100 0000 (0x0040) | | 0000 0001 (0x01) | | 0000 0000 0000 0000 (0x0000) | |
| **Source Address** | | | | | |
| 79 80 fa 67 (121.128.250.103) | | | | | |
| **Destination Address** | | | | | |
| 79 80 fa fe (121.128.250.254) | | | | | |

상기의 표는 패킷 헤더의 구조를 설명하는 표이다. 이 표를 기준으로 헤더의 필드를 정리하였다.

좌측 상단 필드부터 **[Version]**은 IP의 버전을 나타낸다. 캡쳐에서 사용된 네트워크는 IPv4를 사용한다는 점을 확인할 수 있다. [IHL]은 IP에서 헤더가 차지하는 길이를 뜻한다. IP 패킷은 “헤더+데이터+옵션”으로 구성되기 때문에 헤더의 길이를 통해 데이터의 시작점 파악이 가능하다. 캡쳐된 패킷의 경우는 20바이트이며 이 패킷뿐만 아니라 대부분의 IHL 필드값은 5이다. **[TOS]**는 “Type Of Service”의 약자로 서피스 타입과 수준을 표시한다. 즉, 패킷이 얼마나 빠르게 처리 및 전달되어야 하는지를 의미한다. 캡쳐된 패킷은 0으로 일반처리용이다. **[Total Length]**는 패킷의 전체 길이를 바이트단위로 표시한다. 캡쳐된 패킷의 길이는 0011 1100로 60바이트이다. **[Identification]**은 하나의 패킷이 여러 개로 분할되었을 경우, 이들을 서로 구분하기 위해 부여하는 정보로, 일종의 일련번호이다. 캡쳐된 패킷은 “0x0f81”의 일련번호를 갖는다. **[flag]**는 총 세개의 비트로 이루어져 있으나 맨 앞 비트는 사용하지 않는다 두번째 비트는 분열의 유무, 세번째 비트는 이후 조각의 유무를 뜻한다. 캡쳐 내용에 따르면 필드값이 000이므로 분열하면서 뒤에 조각이 더 없음을 알 수 있다. **[Fragment Offset]**은 패킷이 분할 되기 전 데이터 시작점으로부터의 차이를 의미한다. 다시 말해 패킷 재조립 시 분할된 패킷 간의 순서에 대한 정보이다. 캡쳐된 패킷의 필드값이 0이므로 이 패킷이 첫번째 offset이라는 사실을 알 수 있다. **[TTL]**은 “Time To Live”의 약자로 IP 패킷의 수명이다. 최대값은 255이며 라우터를 통과할 때마다 1씩 감소하며 캡쳐된 패킷은 64의 TTL을 가진다. 앞으로 64개의 라우터를 통과할 수 있다는 의미로 해석된다. **[protocol]**의 필드값은 상위계층의 프로토콜이 무엇인지 표시한다. 나타날 수 있는 필드값은 0x06, 0x11, 0x01로 세 가지가 존재하는데 이들은 순서대로 각각 TCP, UDP, ICMP를 의미한다. 캡쳐화면에선 필드값이 0x01이므로 상위계층이 ICMP임을 알 수 있다. **[Header Checksum]**은 IP헤더의 체크섬을 저장한다. 헤더 부분에 대하여 에러 발생 시 이를 정정하기 위한 것이다. 그러나 이는 라우터를 지나갈 때마다 다시 계산되기 때문에 속도가 떨어진다. 이 때문에 의도적으로 계산이 빠지는 경우도 있는데 상기 캡쳐본에서는 체크섬을 계산하지 않았다. 원인은 패킷 전송 지점에서의 문제, 혹은 네트워크 장비의 문제일 수 있다. **[Source Address]**와 **[Destination Address]**는 각각 데이터가 전송된 컴퓨터의 IP 주소와 데이터의 목적지IP 주소를 의미한다. 캡쳐된 화면에서는 121.128.250.103가 출발지이고 121.128.250.254는 목적지이다.

**#2 Wifi (802.11) 환경 (무선)에서 ping을 보낸 뒤 wireshark로 잡은 패킷에 대해 response 받는 것까지 2개의 패킷을) 패킷 캡쳐 화면과 Layer2 Header 분석**

<캡쳐화면>



<헤더 구조 분석>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **IHL** | | **TOS** | **Total Length** | |
| 0100 (0x4) | 0101 (0x5) | | 0000 0000 (0x00) | 0000 0000 0011 1100 (0x003c) | |
| **Identification** | | | | **flag** | **Fragment Offset** |
| 0110 0100 1010 1010 (0x64aa) | | | | 000 | 0 0000 0000 0000 |
| **TTL** | | **protocol** | | **Header Checksum** | |
| 0100 0000 (0x0040) | | 0000 0001 (0x01) | | 1001 0010 0010 1101 (0x922d) | |
| **Source Address** | | | | | |
| c0 a8 01 98 (192.168.1.152) | | | | | |
| **Destination Address** | | | | | |
| c0 a8 01 01 (192.168.1.1) | | | | | |

패킷 헤더의 각 필드가 의미하는 자세한 내용은 1번 문제에서 모두 다루었으므로 2번부터는 필드값의 내용만 해석한다.

**Version)** IPv4 버전의 IP이다.

**IHL)** 패킷에서 헤더가 차지하는 길이는 20바이트이다.

**TOS)** 필드값이 0이므로 일반처리용이다.

**Total Length)** 패킷의 전체 길이는 60바이트이다.

**Identification)** 패킷의 일련번호는 0x64aa이다.

**flag)** 필드값이 000이므로 분열함과 동시에 이후의 조각이 없다.

**Fragment Offset)** 필드값이 0이므로 캡쳐된 패킷이 첫번쨰 offset이다.

**protocol)** 필드값이 0x01이므로 상위계층은 ICMP이다.

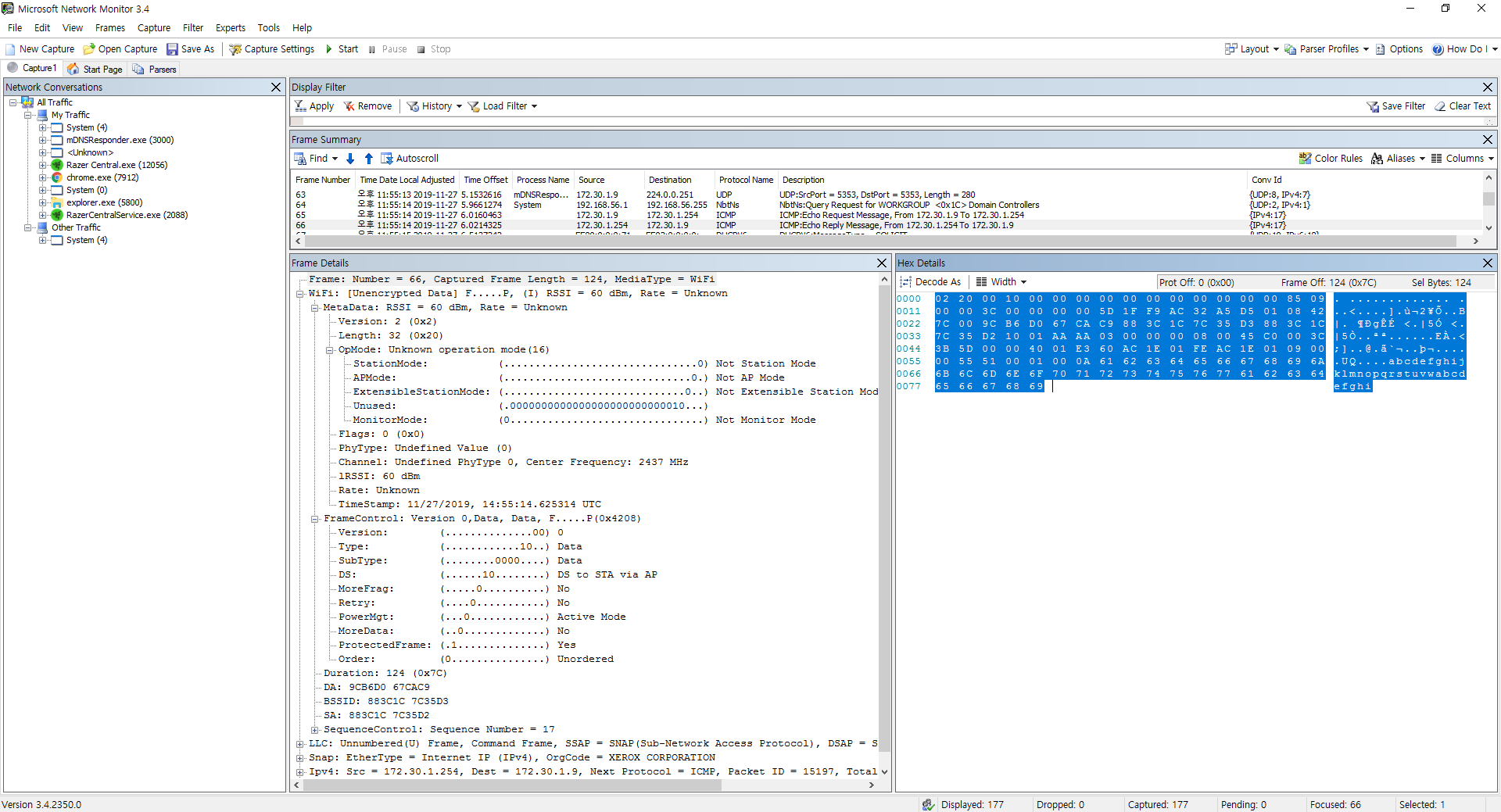
**Header Checksum)** 계산되지 않았다. 패킷 전송 지점 혹은 네트워크 장비의 문제로 추측된다.

**Source Address)** 데이터의 시작지점 주소(192.168.1.152)이다.

**Destination Address)** 데이터의 목적지 주소(192.168.1.1)이다.

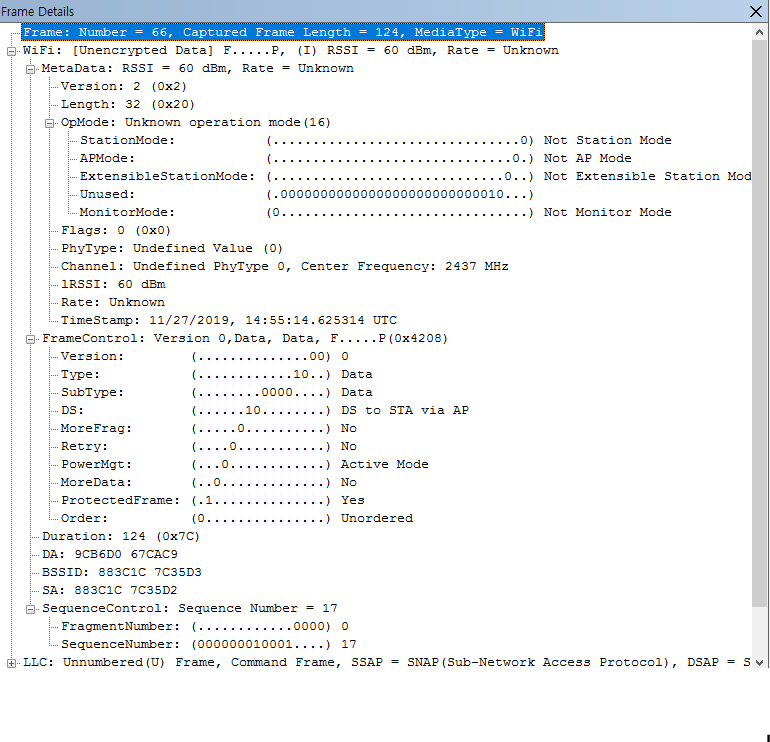
**#3 Wifi (802.11) 환경 (무선)에서 ping을 보낸 뒤 network monitor로 잡은 패킷에 대해 response 받는 것까지 2개의 패킷을 패킷 캡쳐 화면과 Layer2 Header 분석, wireshark로 잡았을 때의 패킷과, network monitor로 잡았을 때의 패킷이 다른 이유 설명**

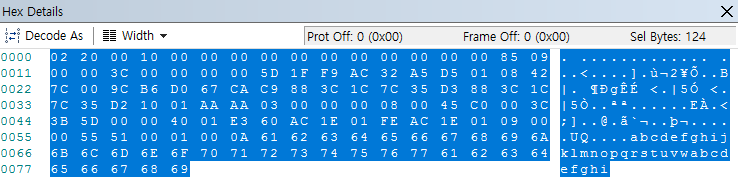
<캡쳐화면>

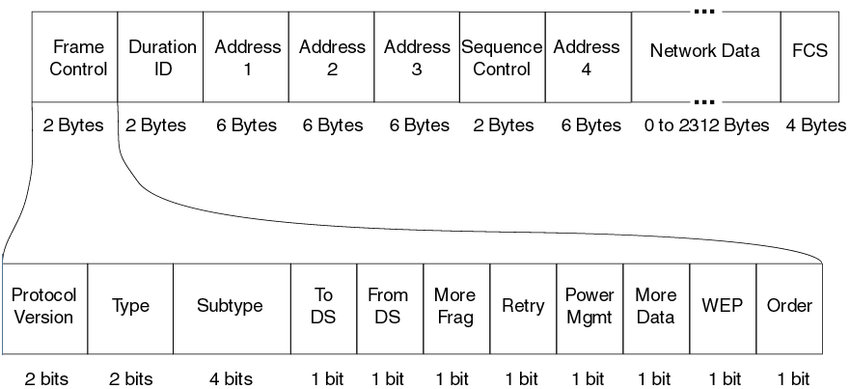


|  |  |
| --- | --- |
| Network Layer | |
| Data-Link  Layer | LLC |
| MAC |
| Physical Layer | |

Layer2는 Data Link Layer로 인접 노드 간 물리적인 회선을 Layer3가 사용 가능하도록 전송 에러가 없는 통신 채널로 변환시키는 계층이다. 이는 두가지 부계층으로 나뉘게 되는데 그것이 바로 MAC 부계층과 LLC 부계층이다. LLC 부계층은 여러 MAC 부계층과 Layer3 간의 접속을 담당하며 기본적으로 연결된 두 지점 간에 신뢰성 있는 패킷 링크로 전환한다. 반면 MAC 부계층은 공유 매체를 통해 신뢰성과 상관없이 패킷을 전송한다. 또한 Physical Layer의 토폴로지나 기타 특성들에 맞추는 제어를 담당한다.

<MAC 프레임 구조 캡쳐 화면>



<802.11 MAC 프레임 구조도>

상기 구조도를 통해 캡쳐된 MAC 프레임 필드를 분석한다.

**Version)** 필드값 00, 프로토콜의 버전을 의미한다. 현재 존재하는 버전은 하나뿐이다.

**Type)** 필드값 10, 프레임의 유형을 나타낸다. 종류는 00, 01, 10으로 각각 관리/제어/데이터프레임을 의미한다. 캡쳐 화면은 데이터프레임을 나타내고 있다.

**SubType)** 필드값 0000, 프레임 유형의 세부 유형을 나타낸다.

**DS)** 필드값 10, “Distribution System”의 약자로 AP 연결 및 무선망 확장을 기능하는 네트워크다. DS에 전송되는 방향에 따라 ToDS와 FromDS로 나뉘며 이를 비트로 나타낸다. 캡쳐 된 필드값 10은 STA에서 AP로 향하는 데이터 프레임이다.

**MoreFrag)** 필드값 0, 동일 MSDU내에서 fragment되었는지에 대한 여부를 뜻한다.

**Retry)** 필드값 0, 이전 프레임이 재전송된 것인지에 대한 여부이다. (중복 수신 방지)

**PowerMgt)** 필드값 0, 전력절감 모드의 활성화 여부를 뜻한다.

**MoreData)** 필드값 0, 추가 데이터 여부를 나타낸다. (관리/데이터프레임에서 사용)

**ProtectedFrame)** 필드값 1, 프레임 바디의 암호화 여부를 의미한다.

**Order)** 필드값 0, HT Control Field 존재 여부를 나타낸다.

**Duration)** **7c -** 전송된 프레임을 무선 노드가 읽고 난 뒤, Destination Address가 자신의 MAC Address와 다를 때 이 필드에 따라 대기하게 된다.

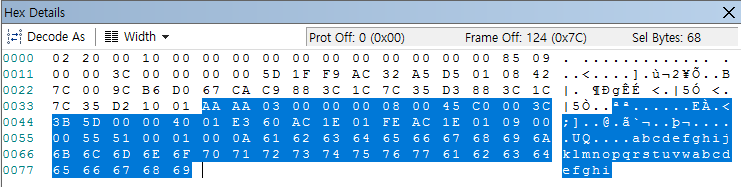
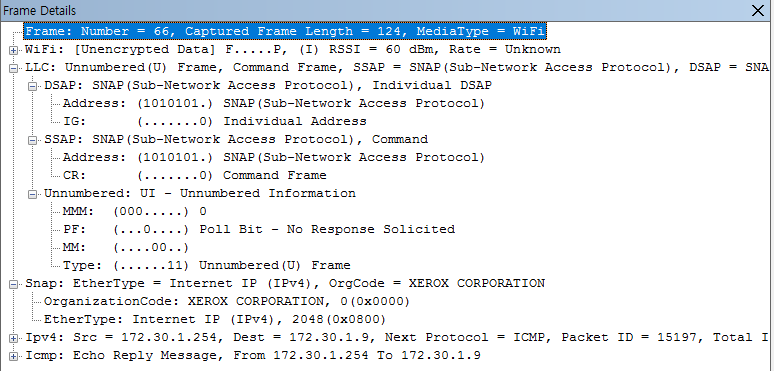
**Address Field)**

일반적으로 세가지 Address Field를 사용하는데, 이는 다음과 같다.

* DA (Destination Address) 9C B6 D0 67 CA C9
* BSSID (Basic Service Set Identifier) 88 3C 1C 7C 35 D3
* SA (Source Address) 88 3C 1C 7C 35 D2

**Sequence Control) 70 00** – 순서가 뒤바뀌거나 중복되는 패킷이 발생치 않도록 하는 필드이다. 총 16bit로 구성되었으며 매 MSDU마다 순서를 부여해주는 Sequence Number이 12bit, 분할된 매 프레임마다 차례대로 순서를 부여하는 Fragment Number이 4bit를 차지한다.

<LLC/SNAP 프레임 캡쳐화면>



<LLC 프레임 구조도>



LLC프레임은 일종의 MAC 부계층의 데이터와 같이 이용된다. 상기 구조도에서는 LLC 부계층만 다루지만 실제로는 캡쳐 화면에 나타나듯 두개의 헤더가 연결되어있다. LLC헤더와 OUI,, 이더넷, 그리고 SNAP 헤더가 함께 결합된 상태이다.

**DSAP (Destination Service Access Point) AA** – AA는 SNAP을 의미한다.

**SSAP (Source Service Access Point) AA** – AA는 SNAP을 의미한다.

**Unnumberd 03** – 제어필드

**Organization Code 00 00 00** – 제조사가 부여한 고유 번호로 조직이나 벤더를 의미하는데 실제로 값이 따로 부여되는 경우는 얼마 없다.

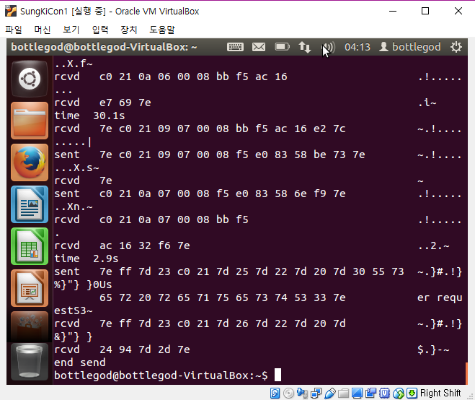
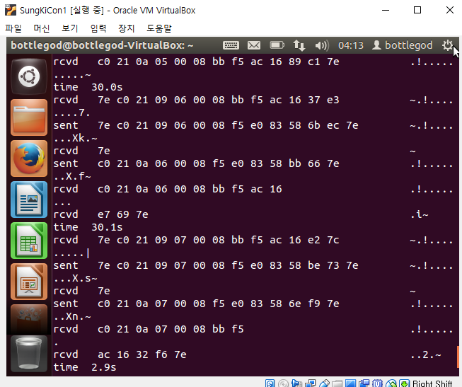
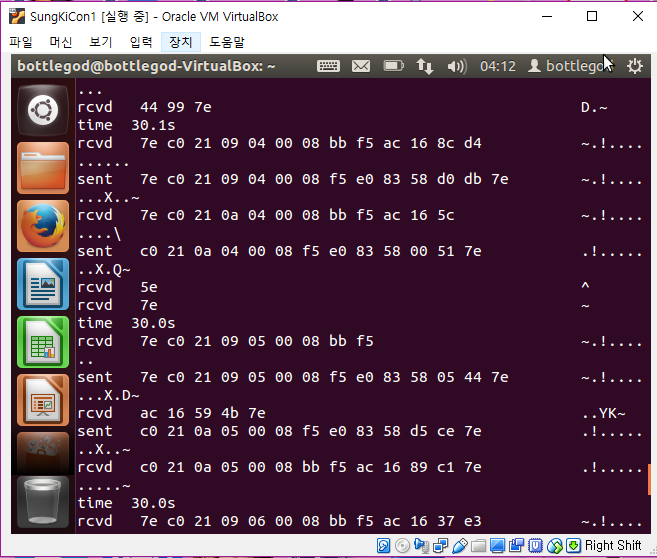
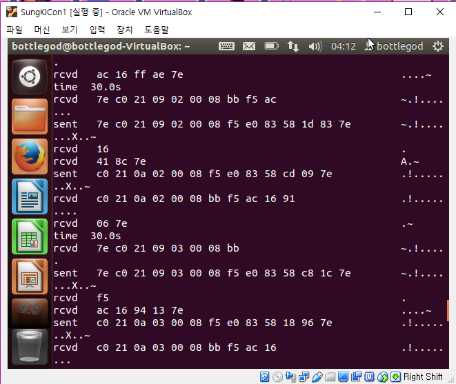
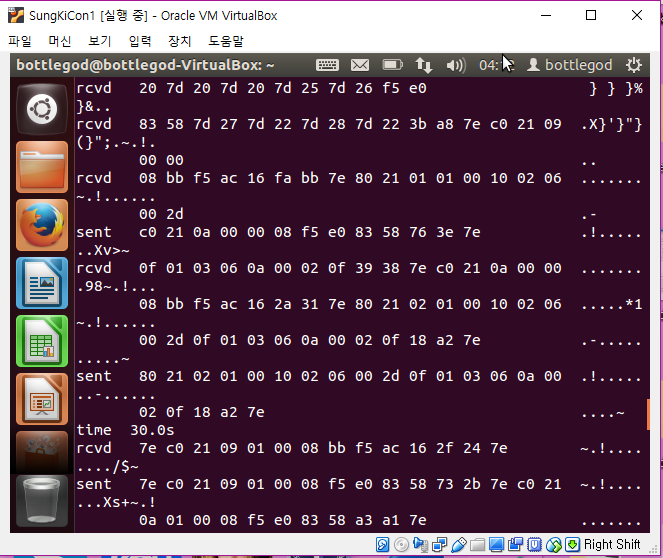
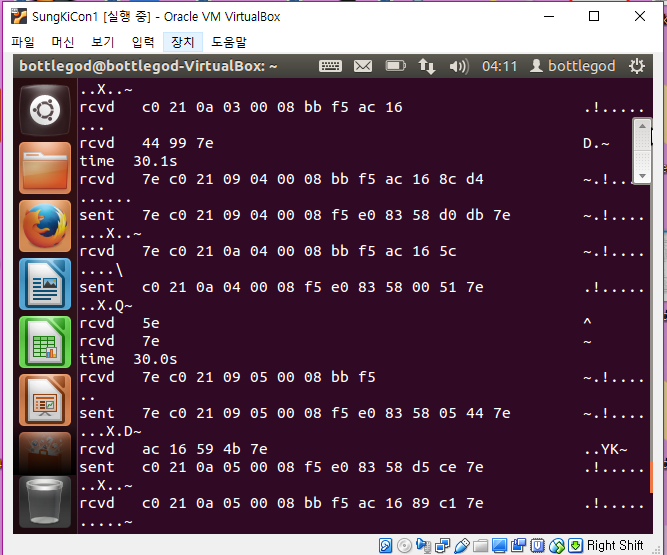
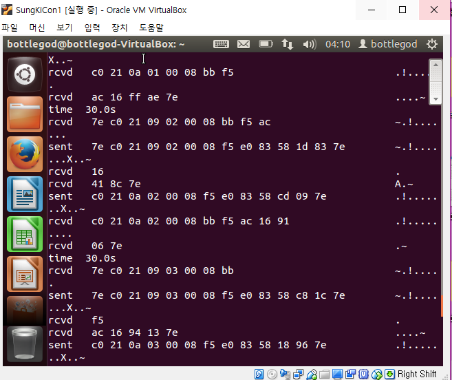
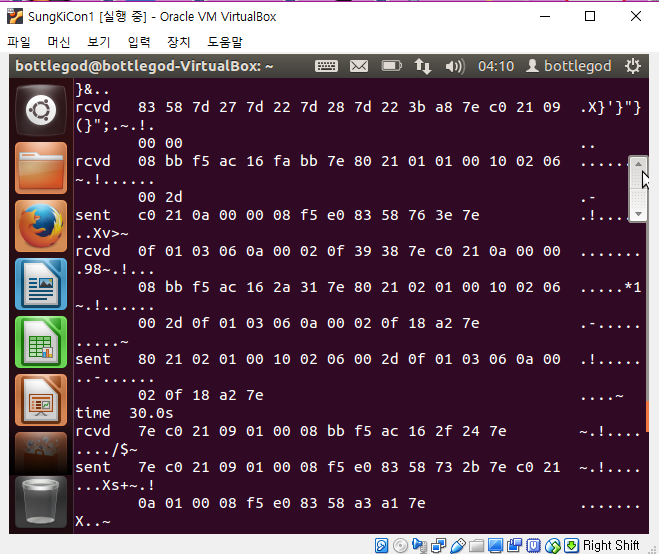
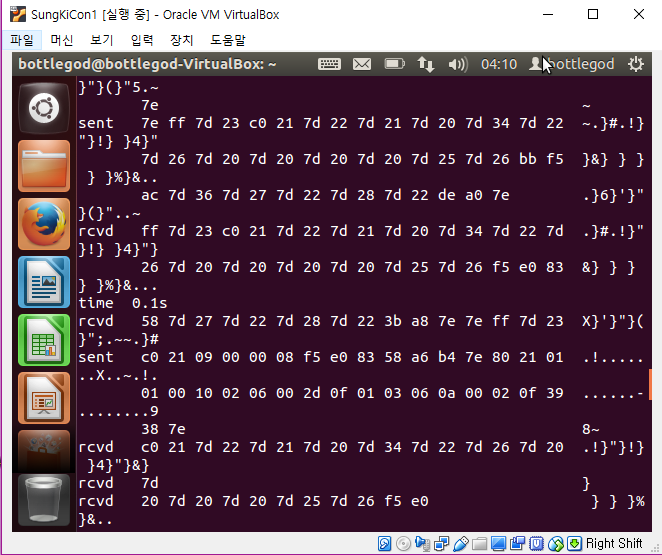
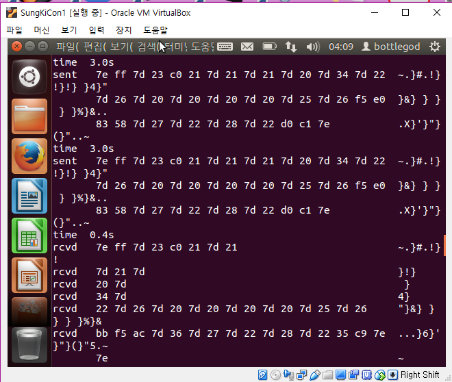
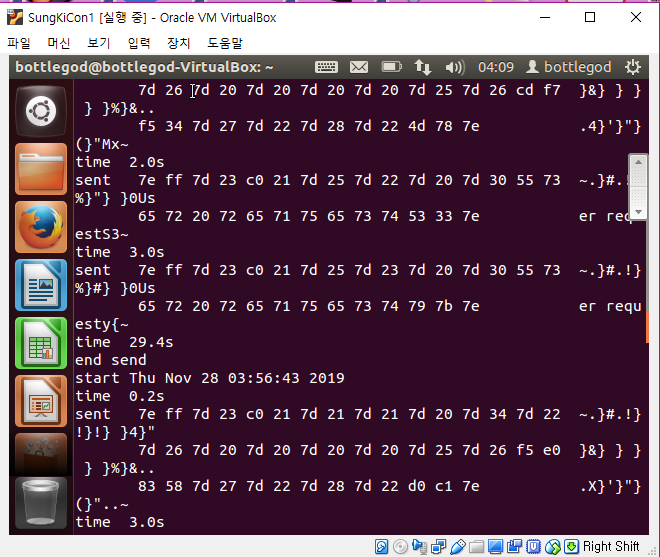
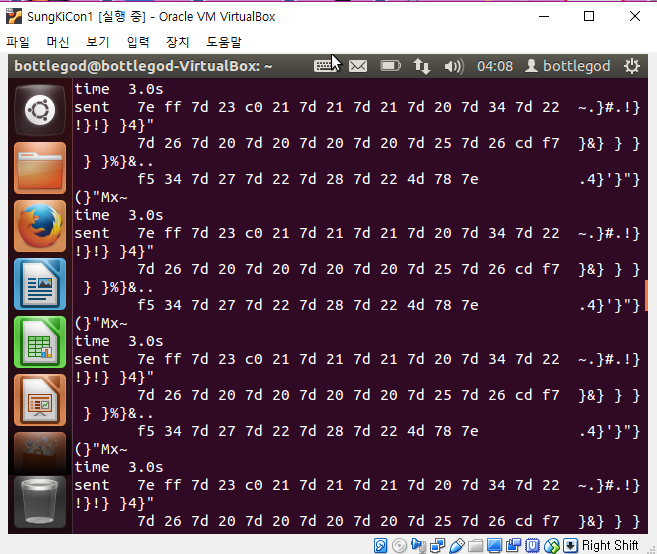
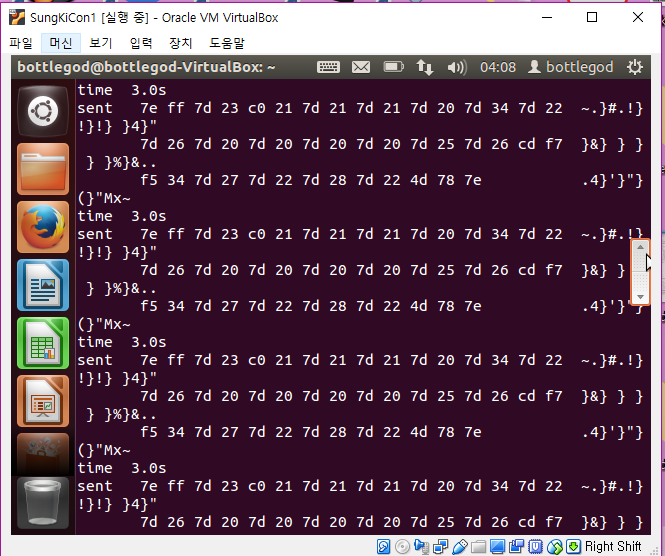
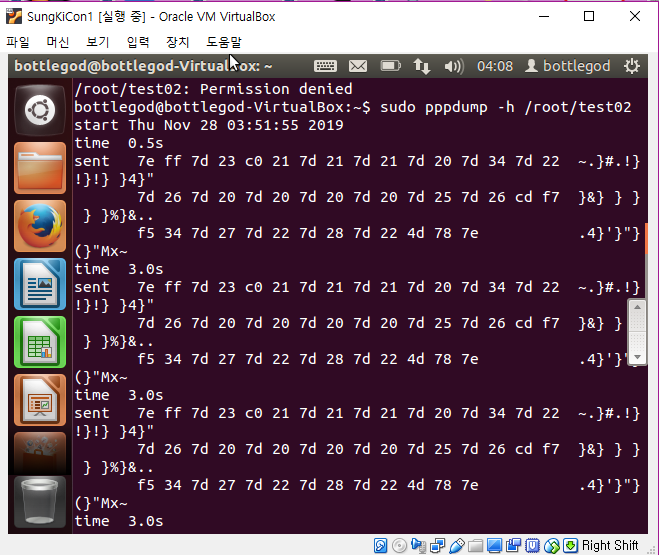
**Type 08 00** – IP를 가리킨다(이더넷 타입과 동일).

**Wireshark로 캡쳐한 패킷과 Network Monitor로 캡쳐한 패킷이 다른 이유 :**

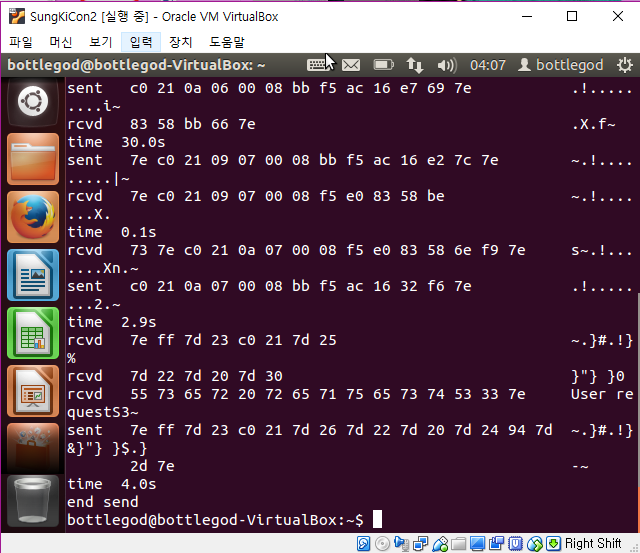
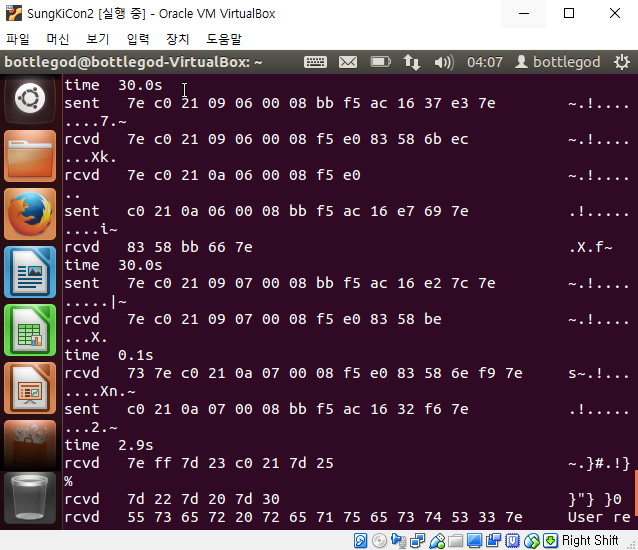
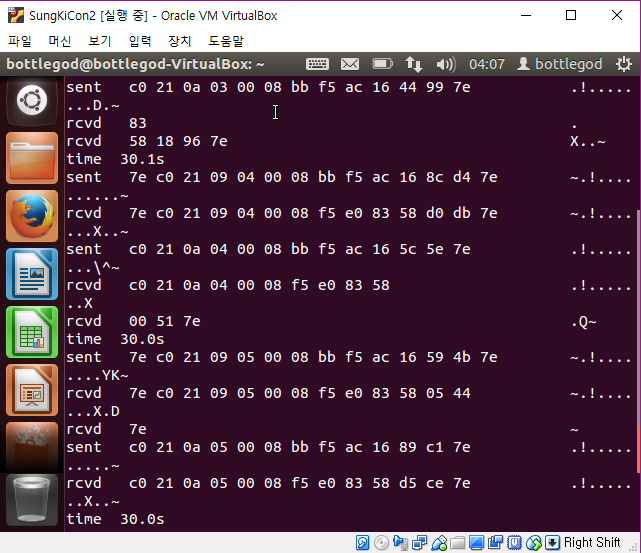
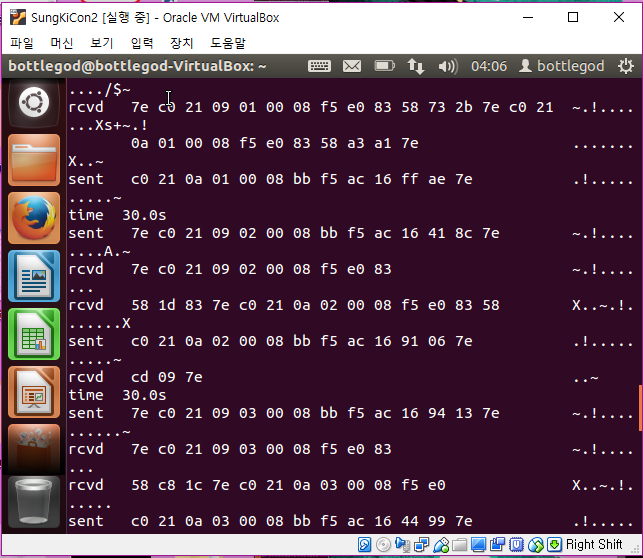
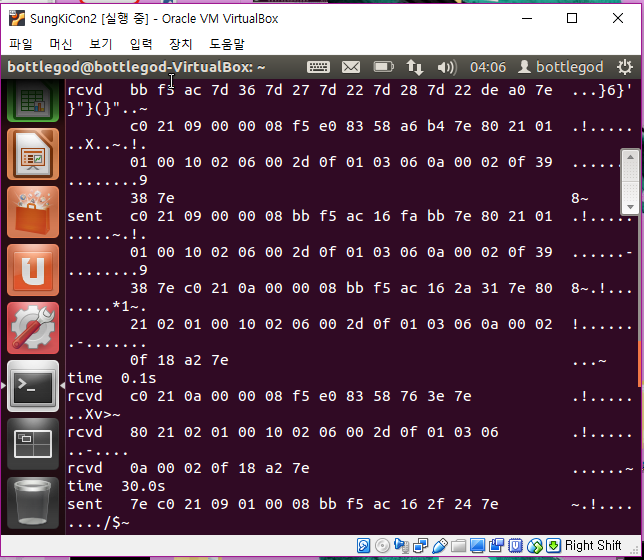
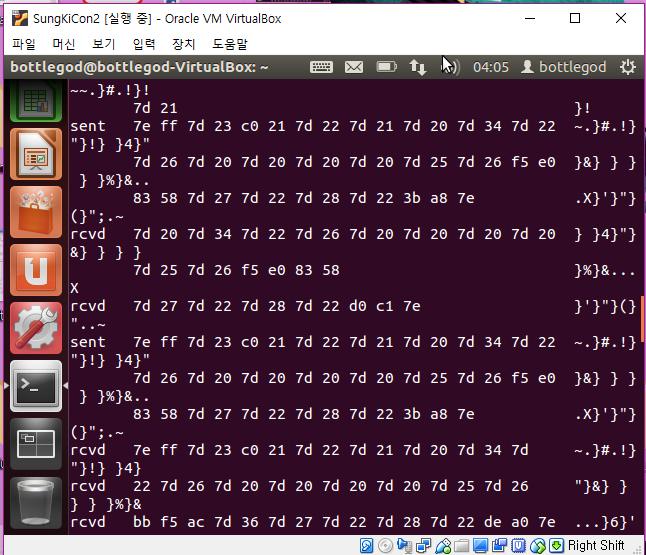
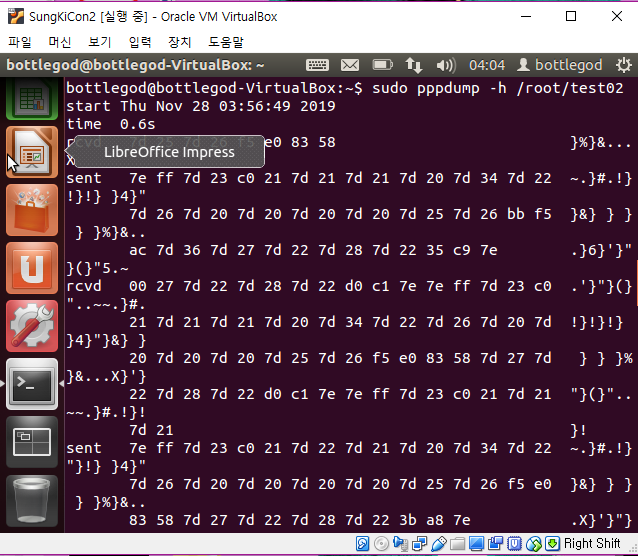
Monitor Mode는 NIC(Network Interface Controller)가 탑재된 장치가 무선 네트워크로부터 전송 받은 모든 트래픽에 대해서 모니터링이 가능케 만든다. 이를 통에 액세스 지점과의 연결에 상관없이 패킷을 캡쳐하는 것이 가능하다. 또 Promiscuous Mod라는 것이 존재하는데 Monitor Mod가 무선 네트워크에서만 동작하는 반면 이 모드는 유선 네트워크이든 무선 네트워크이든 개의치 않고 둘 다 적용이 가능하다. 어쨌거나 무선 카드는 전송 받은 패킷을 별다른 제약 없이 Monitoring하는 것이 가능하므로 Monitoring Mod는 곧 무선 네트워크 한정 Promiscuous Mod라 할 수 있다. 무선 네트워크 드라이브를 사용하면서 802.11 프레임을 전송 및 수신이 가능하다. 이 때 무선 도구를 이용하기 위해 Monitor Mod를 활성화 한 상태에 있어야 한다.

네트워크 트래픽에는 크게 관리/제어 패킷과 데이터 패킷이 존재한다. 제대로 된 패킷을 분석하기 위해선 당연히 관리/제어 패킷과 데이터 패킷 모두가 필요하다. 그러나 Wireshark에는 모든 패킷을 캡쳐하는 Monitor Mod 따위의 설정을 해주지 않았다. 따라서 Wireshark는 관리/제어 패킷은 무시한 채 데이터 패킷만 캡쳐하게 된다. 이 때문에 Wireshark에서 패킷 분석을 제대로 하지 못 한 것이다.

**#4 pppd 명령어를 사용하여 ppp configuration 과정을 pppd connection debug 화면을 기반으로 설명**

****

<VM1 캡쳐 화면>



<VM2 캡쳐 화면>

**PPP란?**

PPP (The Point-to-Point Protocol) 란 서로 다른 업체의 원격 액세스 소프트웨어들이 시리얼라인 상으로 서로 연결하여 TCP/IP 프로토콜로 통신할 수 있도록 만들기 위해 제정된 표준 규약이다(1994년에 RFC 1661로 정의된 인터넷 표준 프로토콜). 이를 쉽게 풀어쓰면, (1)서로 다른 업체의 원격 액세스 소프트웨어(유저의 클라이언트와 서버 라우터 상의 routing process 등)가 (2)시리얼라인 상으로 서로 연결하여(인터넷 전용선이 아닌 모뎀과 전화선 등으로 연결하여) (3)TCP/IP 프로토콜로 통신할 수 있도록 제정된 표준 규약이라는 말이다.

**Configuration 과정**

우선 상기 VM1과 VM2의 캡쳐 화면을 분석해보자

1. VM1에서 Configure Req를 송신하면 VM2에서 Configuration Ack로 응답한다.
2. VM1에서 Configure Echo를 송신하면 VM2에서 Echo Reply로 응답한다.
3. VM1에서 IPCP Configure Req를 송신하면 VM2에서 IPCP Configuration Ack로 응답한다.
4. 위 과정이 끝나고 VM1에서 송신을 Terminate하여 Req를 보내면 VM2도 이를 Ack하여 Terminate된다.

모든 PPP 프로토콜 중에서 가장 중요한 단일 프로토콜은 PPP Link Control Protocol (LCP)이다. LCP는 PPP의 "보스"라고 할 수 있다. 이는 통신의 원할한 수행과 다른 프로토콜의 행동을 “감독”하는 역할을 맡는다.

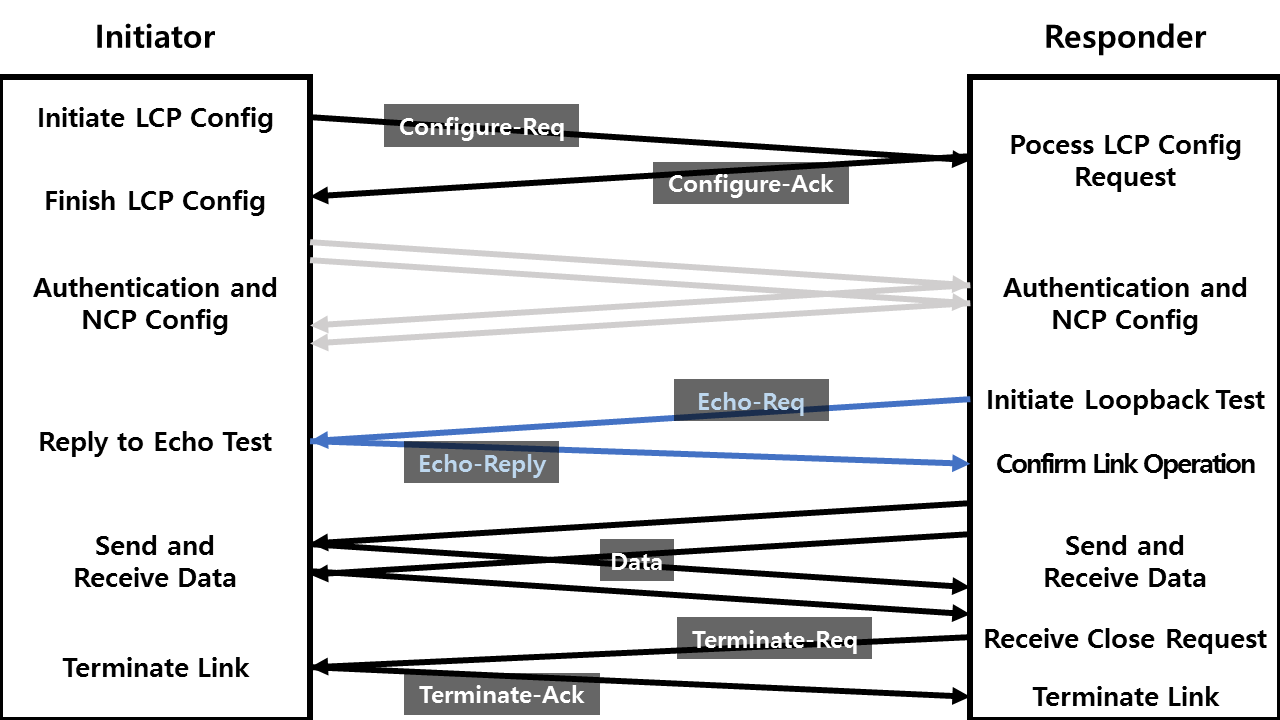
다시 말해, PPP는 링크에 관한 것이고 LCP는 그러한 링크를 제어하는 ​​것에 관한 것이다. PPP 링크의 수행은 일종의 유기체처럼 다양한 "Life Stage"를 진행하는 것으로 생각할 수 있다. 이 "Link Life"는 세 가지 주요 단계로 이루어져 있으며 여기서 LCP는 각 단계에서 중요한 역할을 맡는다.

**1)** 링크 구성 : 링크의 매개 변수를 설정하고 협상하는 프로세스

**2)** 링크 유지 관리 : 열린 링크를 관리하는 프로세스

**3)** 링크 종료 : 기존 링크가 더 이상 필요하지 않거나 기본 물리 계층 연결이 닫힐 때 기존 링크를 닫는 프로세스

이들 각각의 기능은 PPP 링크의 "Life Phases"들 중 하나에 해당한다. 링크 구성은 링크의 초기 링크 설정 단계에서 수행되는데, 이러한 링크 유지 보수는 링크가 열려있는 동안 발생하며 물론 링크 종료 단계 에서 링크 종료가 발생한다. 이하의 그림은 LCP 링크의 간단히 설명하며, PPP 연결의 서로 다른 Life Phases 동안 LCP 수행 중 나타나는 다양한 메시지들 교환을 볼 수 있다.

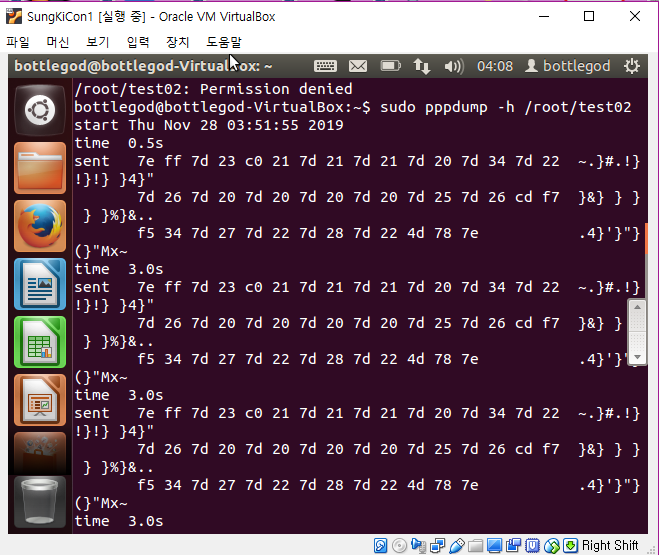


<PPP 링크 제어 프로토콜 메시지 교환>

위 그림을 다시 설명하자면, PPP 연결의 여러 단계에서 LCP가 수행하는 여러 메시지들의 교환을 간략히 나타낸다. 여기서 링크의 구성은 Configure-Request와 Configure-Ack의 교환으로 표시된다. 그리고 복수의 NCP를 인증하고 구성하기 위해 다른 PPP 프로토콜을 사용하여 후속 교환을 진행한 후, 링크는 링크 열기 단계로 돌입하게 된다. 이 예시에서 Echo-Request와 Echo-Reply 메시지는 사전에 링크를 테스트하기 위해 사용되며 두 장치에서 데이터를 송수신한다. 이 때 만약 잘못된 코드 필드가 존재한다면 데이터 메시지가 거부 된 것으로 표시된다(Code-Reject). 이 과정이 모두 수행되면 Terminate-Request를 사용하여 링크가 종료된다.

**#5 ppp 실행 후, pppdump 패킷 분석 (LCP, IPCP, IP Frame이 왜 해당 값을 갖는지)**

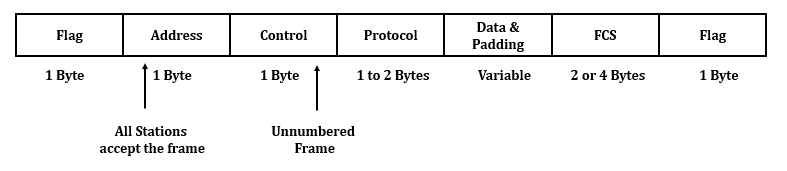
**5-1 ppp**

****

<캡쳐화면>

(빨간 사각형 = ppp)

(파란 사각형 = LCP)



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Flag** | **Address** | **Control** | **Protocol** | **Data&Padding** | **FCS** | **Flag** |
| 7e | ff | 7d 23 | C0 21 |  | 4d 78 | 7e |

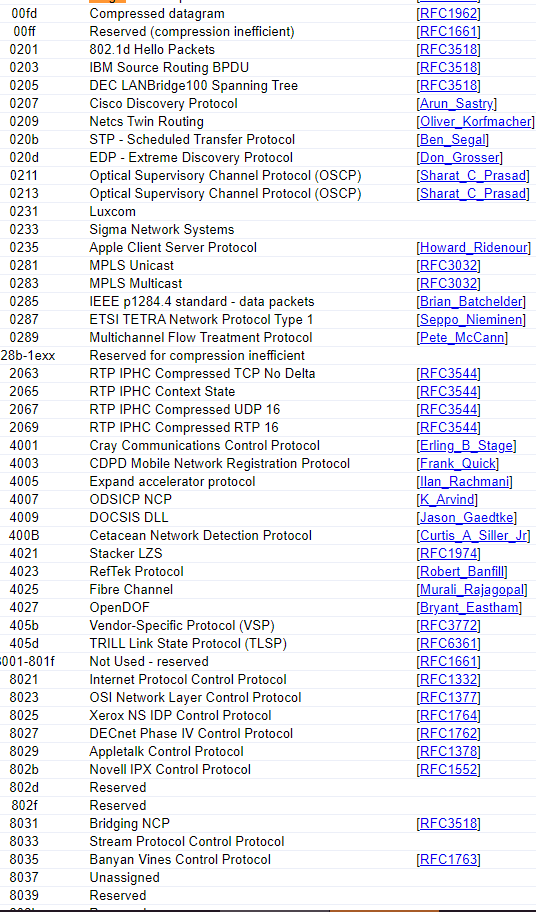
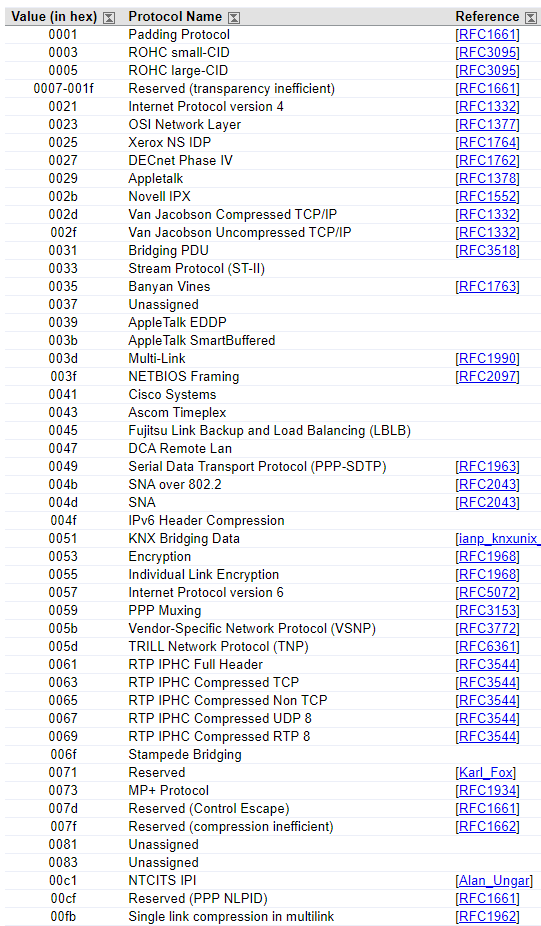
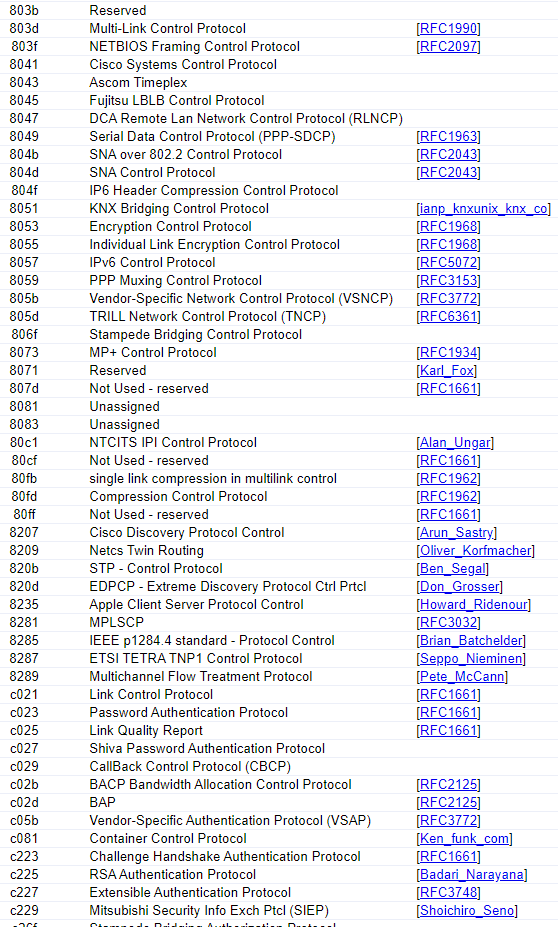
<PPP 일반 프레임에 따른 정리>

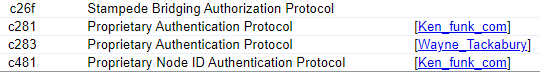
**Flag)** 프레임의 시작과 끝을 뜻한다. 여기서는 항상 7e의 값을 지닌다.

**Address)** 이 필드의 실질적인 의미는 존재하지 않는다. 왜냐하면 이 보고서는 ppp에서 두 장치의 직접적인 연결에 대해서만 다루었기 때문이다. 그러므로 값은 언제나 ff이다. 다만 HDLC에서는 목적지의 주소로서 기능한다.

**Control)** 이진 시퀀스를 포함 하는 단일 바이트로, 순서와 상관 없이 사용자 데이터의 전송을 요구한다. HDLC에서는 다양한 제어 기능의 역할을 하나 ppp에서는 00000011로 고정된다.

**Protocol)** 프레임의 정보 필드에 캡슐화 된 프로토콜을 식별하는 2바이트짜리 값이다.이하의 그림들은 프로토콜 필드값의 종류이다.

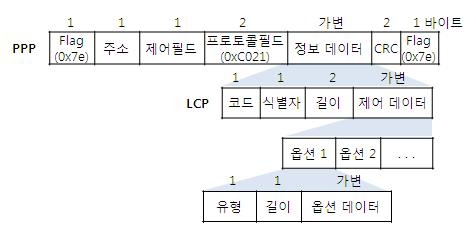


<출처 : IANA>

**Data & Padding)** 실질적으로 서로 교환하는 데이터이다. 종종 Padding을 위해 더미데이터를 추가하기도 한다.

**FCS)** 전송 오류로부터 보호가 이루어지도록 계산된 체크섬이다. 이는 위에 설명한 Address, Control, Protocol, Data & Padding 필드를 통해 계산한다.

**5-2 CP**



<LCP 프레임 구조>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **ID** | **Length** | | **비고** |
| 7D 21 | 7D 21 | 7D 20 7D 34 | | Config-Req |
| **Type** | **Length** | **Option Data** | | **비고** |
| 7D 22 | 7D 27 | 7D 20 7D 20 7D 20 7D 20 | | ACCM |
| **Type** | **Length** | **Magic Number** | | **비고** |
| 7D 25 | 7D 26 | BE 3A 85 DA | | Magic Data |
| **Type** | **Length** | **Type** | **Length** | **비고** |
| 7D 27 | 7D 22 | 7D 28 | 7D 22 | Protocol Field Compression |

<프레임 구조에 따른 정리>

**Code)** 제어 프레임에 존재하는 단일 바이트 값으로 제어 메시지 유형을 나타낸다. 캡쳐본에 따르면 Config-Req를 의미한다.

**ID)** 들어온 Request를 Reply와 일치하도록 만드는 필드이나. 회신할 때 ID의 필드값이 회신의 ID가 된다.

**Length)** 제어 프레임의 길이를 지정한다. 데이터 필드의 길이가 유동적이기 때문이다. Length 필드는 Code, ID, 데이터 필드를 포함한 모든 필드를 포괄한다.

|  |  |  |
| --- | --- | --- |
| **Code** | **Option Type** | **Meaning** |
| 0 | Reserved | Being Reserved |
| 1 | Max. Rec. Unit | 뒤의 패킷 크기가 극단적 |
| 2 | ACCM | 동기/비동기 전환 |
| 3 | Auth. Protocol | 인증 프로토콜 협상 |
| 4 | Quality Protocol | 링크 품질 모니터링 |
| 5 | Magic Number | 상대가 loopback일 경우 프레임 확인 |
| 7 | Prot. Field Compression | 프로토콜 값 압축 |
| 8 | Address and Ctrl. Field Compression | Address와 제어필드 값 압축 |
| F | Compound Frame |  |

<LCP Option>

(첫 비트에 옵션의 타입과 길이를 나타냄)

|  |  |  |
| --- | --- | --- |
| **Code** | **Option Type** | **Meaning** |
| 1 | IP Addresses |  |
| 2 | IP Compression Protocol | 프로토콜 압축 방법 협상 |
| 3 | IP Address | 각 종단점의 IP Address 협상 |
| 4 | Mobile IP |  |

<IPCP Option>

**결론**

Point-to-Point의 송수신 패킷은 PPP 기본 프레임에 따라 여러 프로토콜과 옵션 타입에 적합한 데이터를 읽어낸다. 또한 LCP/IPCP 패킷도 송수신 된다는 사실도 앞서 설명한 바와 같이 장치들 양 끝 지점의 Request/Reply 신호가 오가는 사이에서 확인할 수 있었다.