Wireshark 설치하고 시작하기

Wireshark은 network상의 packet을 포착(Capture)하고 분석(Analyze)해서 보여주는(Display) 무료로 사용할 수 있는 도구(Tool)이다. 이 중 포착 기능은 직접 구현되어 있지 않고 기존의 다른 도구에 의존하는데 Unix에서는 libpcap, Windows에서는 WinPcap이라고 하는 드라이버를 사용한다. 이 밖의 다른 종류의 Packet capture 도구를 사용할 수도 있다.

여기서 중요한 것은 해당 packet들을 capture하는 기능은 완전히 수동적(passive)이며 투명(transparent)하다는 것이다. 수동적이라는 것은 관찰을 위해서 뭔가 지시를 하거나 packet을 생성하지 않는 것을 말하며, 투명하다는 것은 관찰 대상이 되는 packet들을 보내고 받는 입장에서 전혀 영향 없는 것을 말한다.

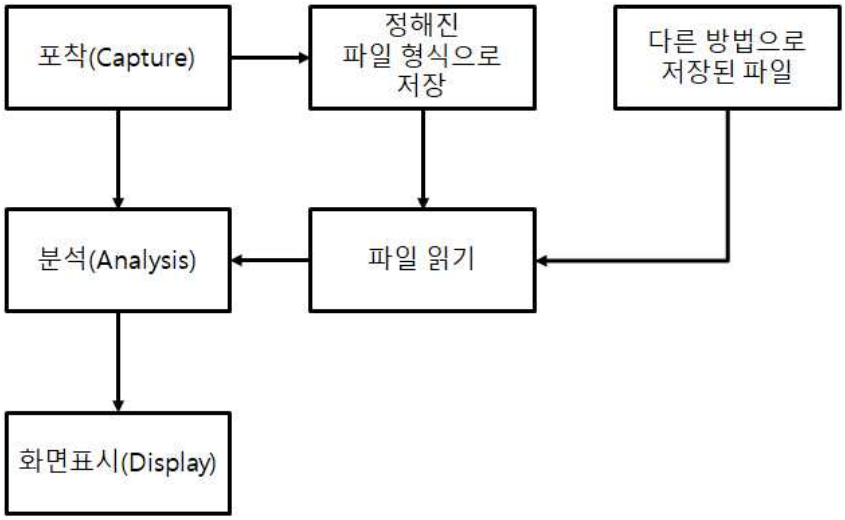
Wireshark은 1998년에 University of Missouri-Kansas City의 대학원생 Gerald Combs가 Ethereal이라는 이름으로 개발하여 시작되었는데, 저작권문제로 해당 이름을 사용하지 못하게 되어 2006년부터 Wireshark으로 이름을 바꾸어 통용되어 왔다. Wireshark은 다양한 packet 분석 도구들 중 두각을 나타내어 지금에 이르러서는 가장 많이 사용되는 packet analyzer가 되었다.

Wireshark은 다양한 용도로 사용되는데, 네트워크의 문제를 발견하기 위해서, 보안 문제를 시험하기 위해서, 프로토콜 개발 시에 디버그를 위해서, 혹은 우리 책의 의도처럼 네트워크의 다양한 기능을 배우기 위해서 사용될 수 있다.

Wireshark은 Linux, macOS, BSD, Solaris 등 Unix기반 OS와 Windows에서 사용이 가능하다.

Wireshark의 주요 기능은 다음과 같다.
- Ethernet LAN card나 WLAN card 등의 Network interface에서 실시간으로 packet을 포착한다.
- 포착한 결과를 파일에 저장한다.
- 포착한 결과 파일을 읽어서 내용을 보여준다. 다른 capture 프로그램 (tcpdump/WinDump 등) 에서 capture하여 저장한 결과도 읽어 들인다.
- 포착한 packet의 내용을 들여다보고 분석한다.
- 분석한 결과를 일목요연하게 보여준다.

아래와 같은 그림으로 정리할 수 있다.



이 중 분석과 화면표시 기능이 Wireshark의 핵심인데, 다음과 같은 주요 기능들을 포함하고 있다.

1. Filter

먼저, 사용자가 정의한 바대로 포착한 packet들을 Filter할 수 있다. Filter는 Display filter와 Capture Filter의 두 가지 기능이 있는데, Display filter는 저장된 packet들 중 조건을 만족하는 packet들만을 화면에 표시해주는 기능이고, Capture filter는 해당 조건을 만족하는 packet들만을 처음부터 저장해서 분석하고 보여주는 기능이다.

2. Colorize

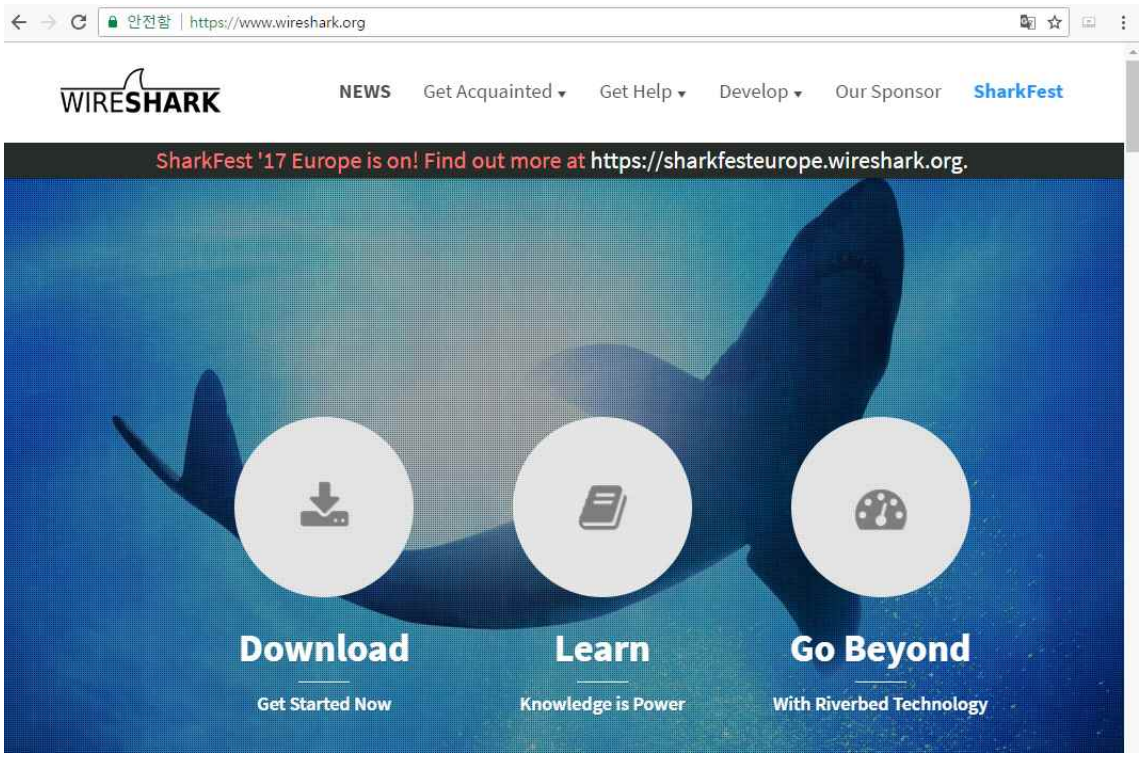
또 다른 유용한 기능은 packet들을 종류에 따라 색으로 구분하여 보여주는 기능이다. 해당 기능도 구분하고자 하는 packet의 종류와 색을 사용자가 정의할 수 있다.

3. Statistics

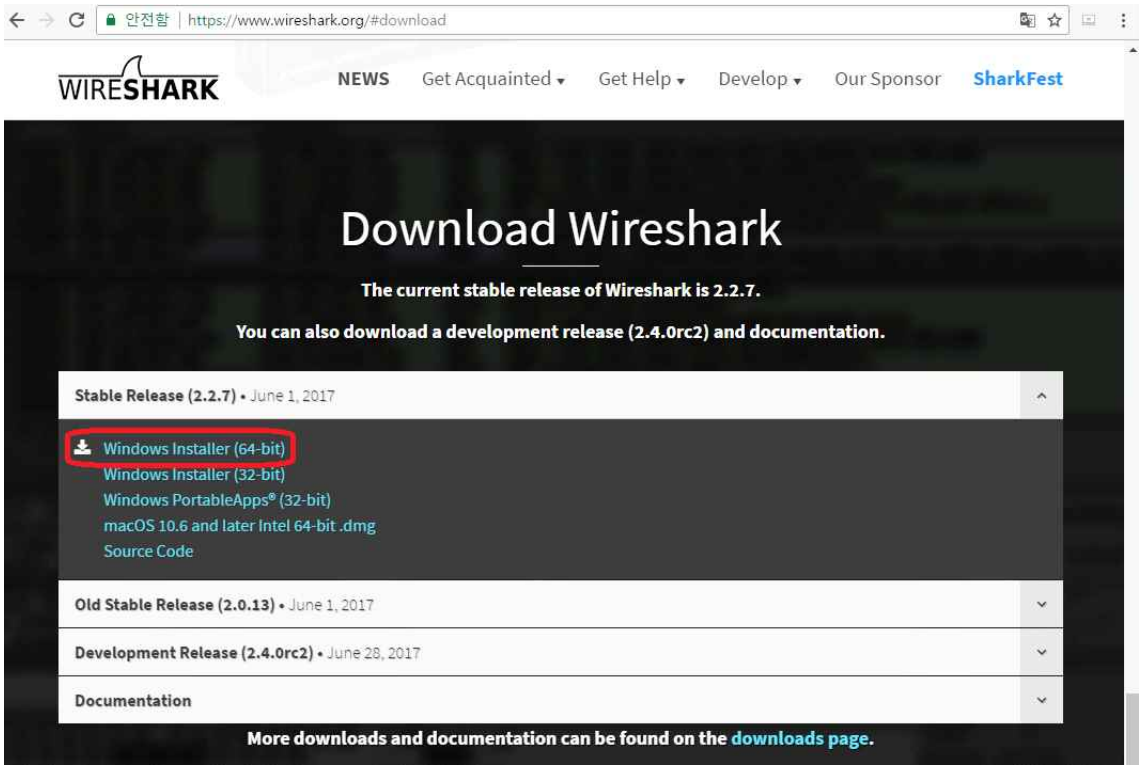
Packet들을 구분하여 해당 카테고리의 다양한 통계를 제공한다.

Activity: Wireshark 설치하고 시작하기

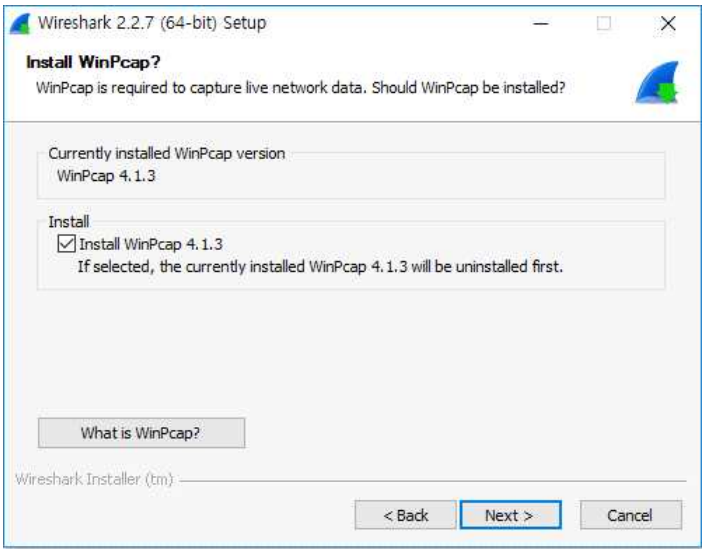
Wireshark을 설치하고 실행시켜 본다. Wireshark은 freeware로 아래 사이트에서 download 후 설치할 수 있다.



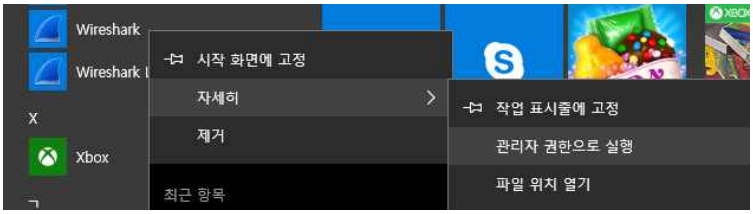
위에서 Download를 선택하면 아래 화면이 나온다. 현재 사용하고 있는 운영체제를 감지해서 이에 맞는 버전이 제시된다.



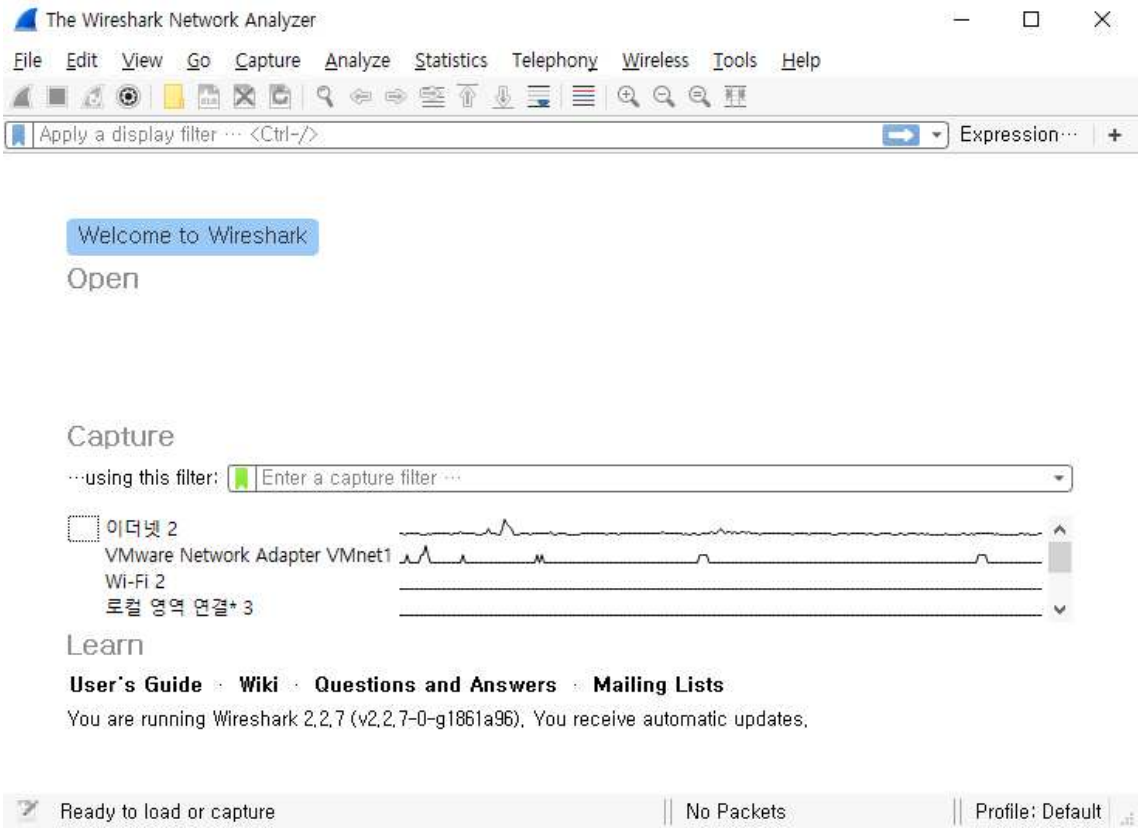
Download후에 설치를 진행하면 WinPcap이라는 프로그램을 설치할 것인지를 묻는다. WinPcap이 이미 설치된 컴퓨터가 아니면 꼭 설치가 필요한 프로그램이다. 사실 Wireshark은 capture된 frame들을 분석하여 정보를 보기 쉽게 제공해주는 일종의 사용자 인터페이스 프로그램에 가깝다. 실제로 frame들을 capture하는 일은 운영체제와 실제 유무선 네트워크 인터페이스 사이에 존재하는 WinPcap이라는 driver의 역할이다.



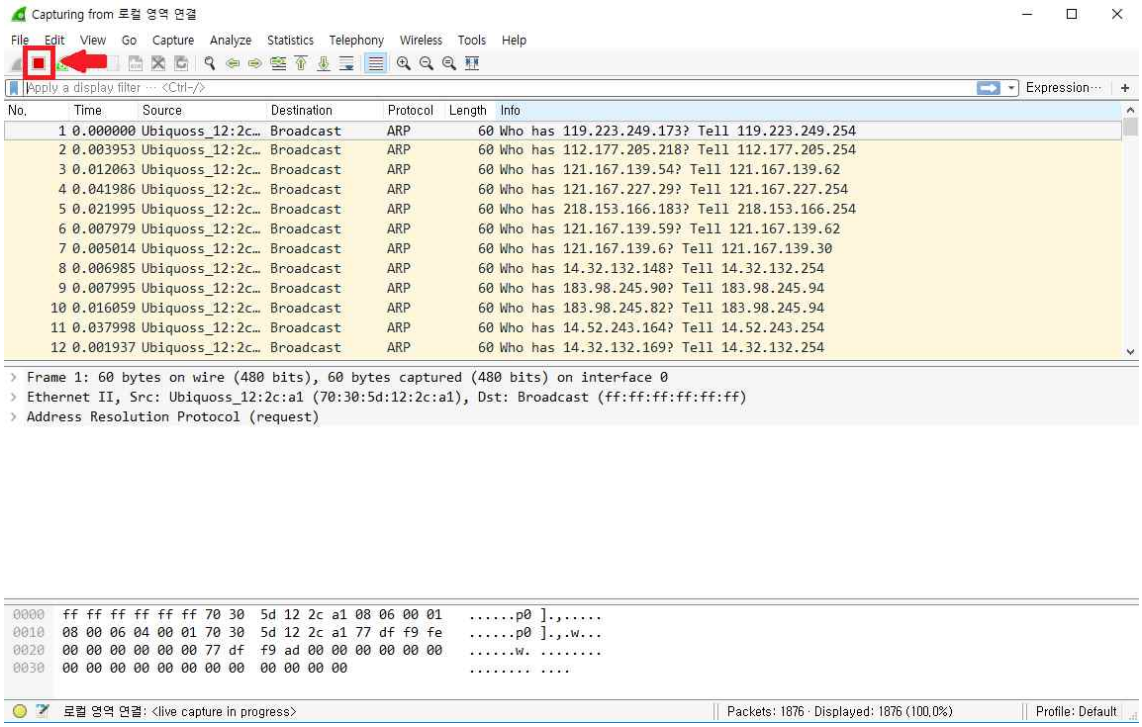
설치가 끝나면 실행해 보도록 하자. Wireshark과 Wireshark Legacy 두 가지 모두 선택 가능한데 우리는 Wireshark을 선택한다. 이 때 주의할 점은 Wireshark을 관리자 권한으로 실행해야 한다는 것이다. WinPcap의 frame capture를 일반 사용자에게 허용하지 않는 운영체제가 많아서인데, 관리자 권한으로 실행하면 이 문제가 해결된다.



Wireshark을 실행하면 아래와 같은 화면이 뜬다.

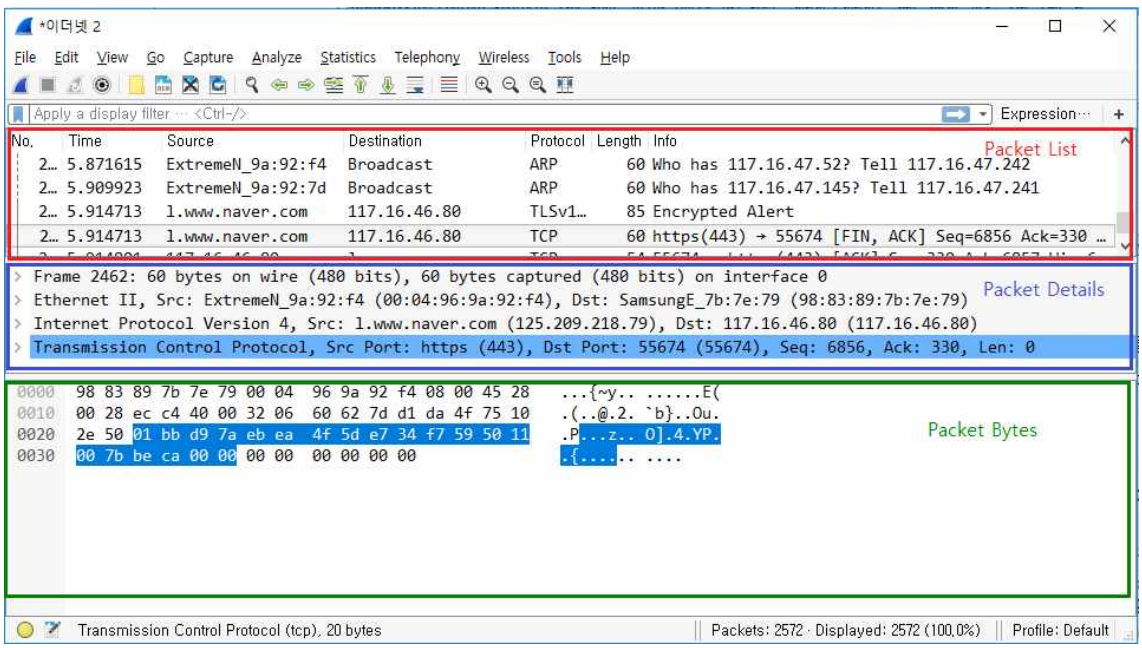


위에서 보는 바와 같이, 하나의 컴퓨터에 여러 개의 Interface가 존재하는 것을 알 수 있다. 실제 사용하고 있는 interface를 알 수 있는 간단한 방법은 각 interface 옆의 그래프를 확인하는 것이다. 위에서는 “이더넷 2”가 가장 활발히 사용되고 있다, 어떤 interface에서 프레임을 capture하여 분석할지 결정이 되었으면 간단히 해당 interface를 더블클릭하는 방법으로 시작할 수 있다. 여기서는 “이더넷 2”를 더블클릭해서 capture를 시작해보자.



적당히 시간을 보내서 capture가 진행되면, 위 그림에 표시한, 왼쪽 상단 메뉴 중 빨간 색 사각형 버튼을 눌러서 capture를 중단한다.

여기서 Capture의 결과를 살펴보도록 하자. Wireshark에는 세 개의 window가 있는데, 각각 Packet List window, Packet Details window, Packet Bytes window이라 부르며 그 모양과 위치는 아래와 같다.



Packet list window는 Wireshark이 capture한 packet들을 시간 순서대로 나열해서 보여준

다. (시간 순서 이외의 다른 방법으로 정렬도 가능하다.) Packet list window 가장 왼쪽의 column인 Number는 capture된 순서를 의미한다. Time은 packet이 capture된 시간 (capture 시작 시간에서 부터의 상대적인 시간), Source와 Destination은 packet의 출발지와 목적지 주소, Protocol은 packet이 포함하는 주요 프로토콜, Length는 byte 단위의 길이, Info는 해당 packet이 내포하고 있는 주요 정보이다. 예를 들어 위 그림의 Packet list window에서 가장 위에 보이는 packet은 ARP(Address Resolution Protocol) 기능을 수행하는 packet이며 “Info" 칼럼에 명시된 “Who has 117.16.47.52? Tell 117.16.47.242”라는 것은 packet이 수행하려고 하는 구체적인 기능을 나타내는데, 여기서는 “나는 117.16.47.242 주소를 가지고 있으니, 117.16.47.52 주소를 가지고 있는 사람은 답장해라”라는 요청이다. 이러한 내용을 알려주려면, 정확하게 프로토콜을 이해하고 해당 내용이 어디에 있는지 파악하고 분석하는 것이 필요한데 이것이 Wireshark의 핵심 기능이다.

Packet Details window는 Packet list window에서 선택된 packet의 자세한 사항을 알려준다. 위 그림에서는 위에서 네 번째 packet이 선택되었는데, 해당 packet은 TCP 프로토콜의 packet이며, 이를 packet details window에서 자세히 확인할 수 있다. 선택된 packet의 일련번호는 2462번이며, Ethernet II, Internet Protocol version 4, Transmission Control Protocol packet이다. 해당 packet의 대표 protocol을 TCP로 알려준 것을 기억하자.

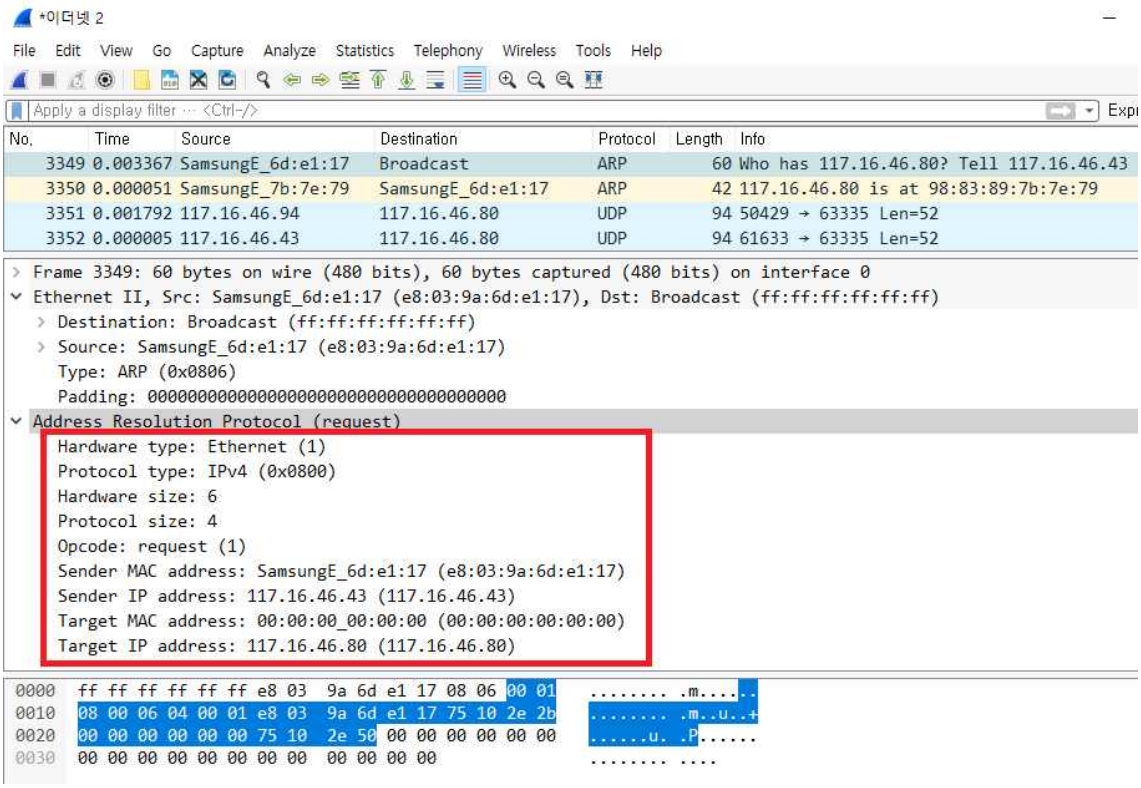
마지막 window는 Packet Bytes window이며, Packet details window에서 선택된 부분에 해당하는 Hexadecimal data를 보여준다. 위 그림에서는 TCP protocol의 header부분이 선택되어 이를 보여준다.

여기서 Wireshark에서 파악한 내용을 바탕으로 프로토콜이 무엇인지 ARP를 중심으로 살펴보도록 하자.

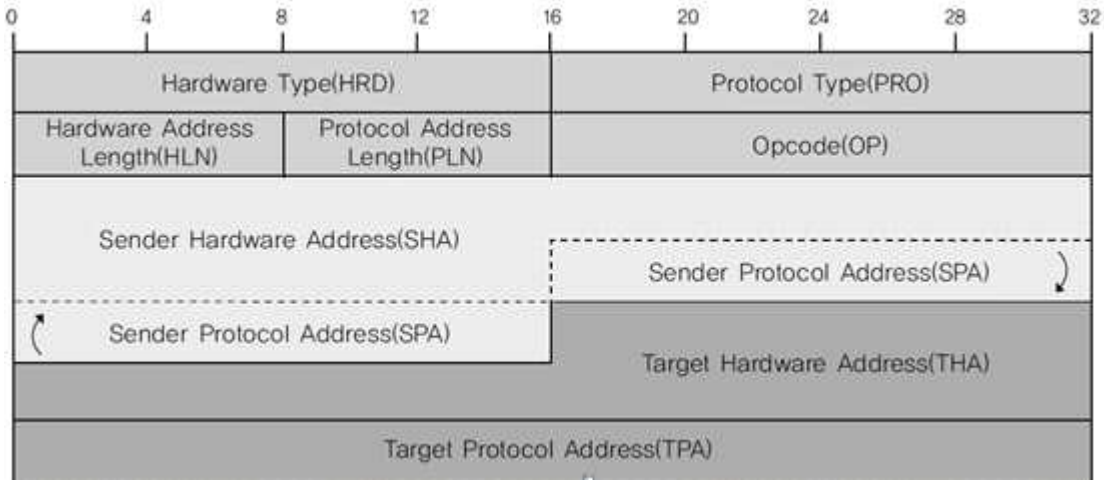
프로토콜이란 데이터를 주고받는 상호 간에 미리 약속된 규약이다. 프로토콜에는 여러 가지 요소가 포함되어 있다. 주고받는 데이터의 구조(structure), 해당 구조 속에 포함된 각 부분의 의미(meaning), 데이터를 주고받는 시간, 절차와 속도 등을 포함한다. 이는 마치 대화를 하는 사람 간에 언어와 에티켓이라는 약속이 미리 정해져 있는 것과 비견된다.

Address resolution protocol은 성공적인 데이터 교환을 위해 존재하는 수많은 프로토콜 중 하나이다. 먼저 ARP가 사용하는 데이터의 구조를 살펴보자. 이를 위해 Wireshark으로 적당히 capture한 후 Packet list window에서 ARP packet을 찾아서 선택해 준다. ARP packet은 가장 흔히 관찰할 수 있는 packet이므로 힘들이지 않고 찾을 수 있을 것이다.

Activity: ARP packet 관찰



위 그림에서는 3349번과 3350번의 두 개의 packet이 ARP packet이다. 3349번 packet을 선택하서 클릭하면 Packet detail window에 해당 packet의 자세한 내용이 보이는데, 이 중 "Address Resolution Protocol (request)" 부분을 click하면 가장 아래의 Packet byte window에 파란색으로 ARP 부분이 표시된다. Hexadecimal notation으로는 두 자리 숫자가 1byte에 해당하므로 파란색으로 선택된 부분은 총 28byte이다. ARP packet의 내부를 각각 click해 보면 Hardware type 2byte, Protocol type 2byte... 등으로 구성되어 있다는 것도 알 수 있다. 이를 바탕으로 ARP packet을 재구성해서 그려보면 아래와 같다.



위 그림에서 네모로 표시한 각 부분을 Field라고 부른다. ARP는 Hardware type부터 Destination Protocol Address까지 9개의 field를 가지고 있으며 모든 field는 고정된 길이를 가지고 있고, 전체 packet의 길이는 따라서 26byte이다. 이 26byte 길이는 고정되어 있는 것은 아니며 Hardware address length와 Protocol address length의 값이 바뀌면 이에 따라 바뀌게 된다.

위의 field 중 Hardware size(혹은 Hardware length), Protocol size(혹은 Protocol length)라고 표현된 부분은 Hardware address length와 Protocol address length의 간략한 표현이다.

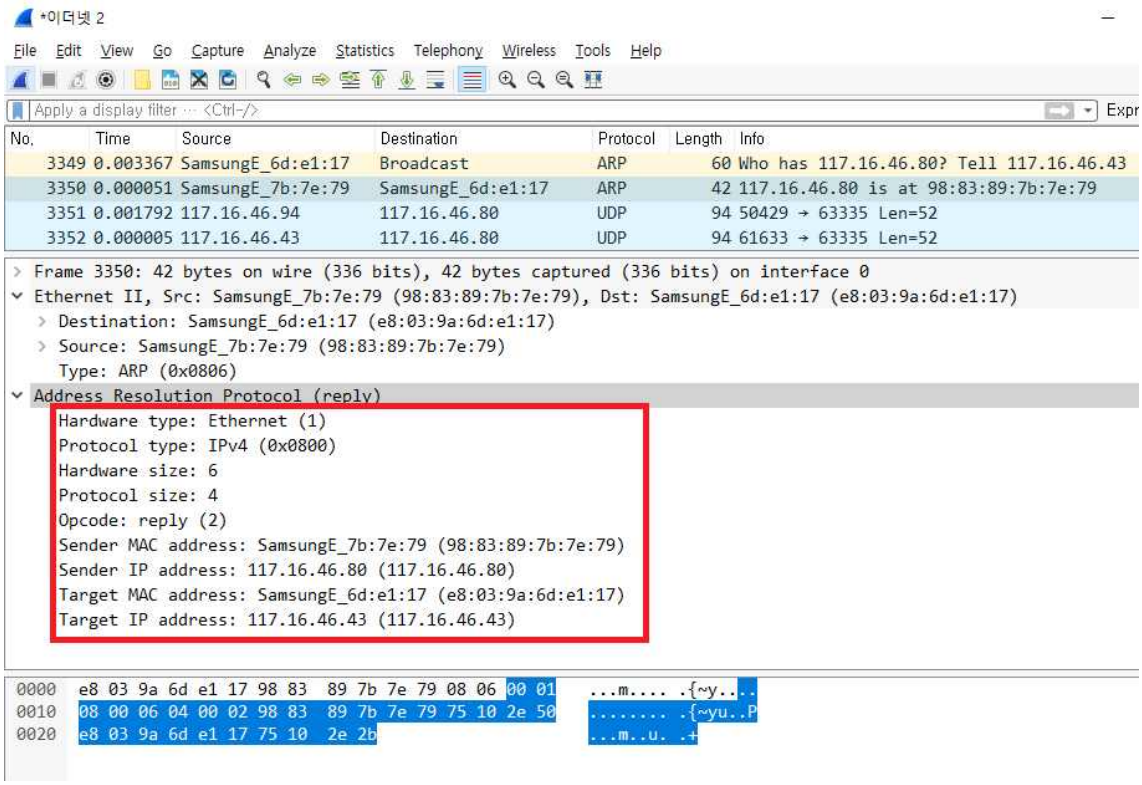
해당 packet의 operation code(Opcode)가 request로 명시되어 있는 것은 Wireshark이 해당 field의 값을 읽어서 분석하여 알려준 결과이다.

해당 packet은 무엇을 request하고 있는가? 그것은 Target MAC address가 00:00:00_00:00:00으로 표시된 것에서 힌트를 얻을 수 있다.

```
Address Resolution Protocol (request)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: request (1)
   Sender MAC address: SamsungE_6d:e1:17 (e8:03:9a:6d:e1:17)
   Sender IP address: 117.16.46.43 (117.16.46.43)
   Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
   Target IP address: 117.16.46.80 (117.16.46.80)
```

해당 packet은 이런 메시지를 담고 있다. “나의 MAC address는 e8:03:9a:6d:e1:17이고, IP address는 117.16.46.43이다. 나는 IP address가 117.16.46.80인 장비의 MAC address를 알고 싶다. 해당 IP address를 가진 장비는 대답하라.”

IP address가 117.16.46.80인 장비가 이 ARP request를 받으면 자신의 MAC address를, 이번엔 Reply (Opcode 2)임을 명시해서 답장하게 된다. 이렇게 Request와 Reply 간단한 데이터 교환이 ARP의 규약이다. 해당 Reply packet은 바로 다음에 등장한다.



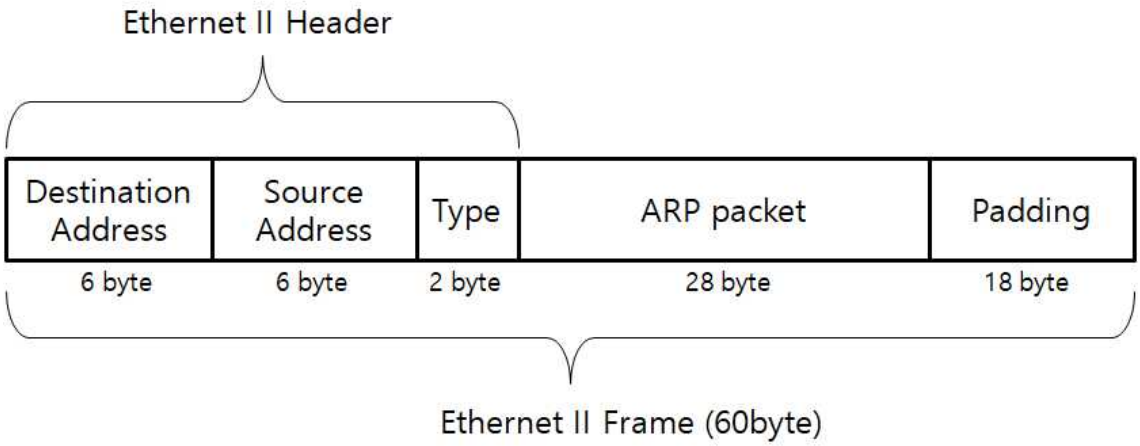
위 그림의 3350번 packet이 Reply packet이다. 여기서는 “Sender MAC address” field에 Request에서 요청한 MAC address를 명시하여 보내준다.

Layered architecture

위 ARP의 예에서 본 packet의 전체 구조는 다음과 같다.

```
> Frame 3349: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
v Ethernet II, Src: SamsungE_6d:e1:17 (e8:03:9a:6d:e1:17), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: SamsungE_6d:e1:17 (e8:03:9a:6d:e1:17)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: SamsungE_6d:e1:17 (e8:03:9a:6d:e1:17)
    Sender IP address: 117.16.46.43 (117.16.46.43)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 117.16.46.80 (117.16.46.80)
```

이를 그림으로 나타내면 아래와 같다.



Ethernet frame중 header를 제외한 부분, 즉 ARP packet과 Padding 부분을 Ethernet payload라고 한다. Payload라고 하는 것은 과금 대상 적재물이란 것인데, 트럭이 짐칸에 실은 짐을 의미한다. 즉 여기서 Ethernet frame 전체는 짐을 실은 트럭으로, 짐은 ARP packet과 padding으로 볼 수 있다. 다만 Padding은 짐칸이 비어 있을 때 채워 넣는 가상의 데이터이므로 과금 대상이라고 하기에는 무리가 있다. Payload가 40byte보다 작은 경우 payload와 padding을 합쳐서 40byte가 되도록 만들어 넣어 준다. 이렇게 Ethernet frame의 최소 길이가, Wireshark에서는 관찰되지 않는 CRC 4byte를 포함하여, 64byte가 되도록 만들어 준다. 이렇게 Ethernet frame의 최소 길이가 정의되는 이유는 Bus 구조에서 가장 멀리 떨어진 노드 두 개가 동시에 각각 frame의 전송을 시작했을 때 양쪽 모두 충돌을 감지할 수 있게 하기 위해서이다. 충돌이란 신호가 중첩되어 의도한 데이터를 추출하는 것이 불가능한

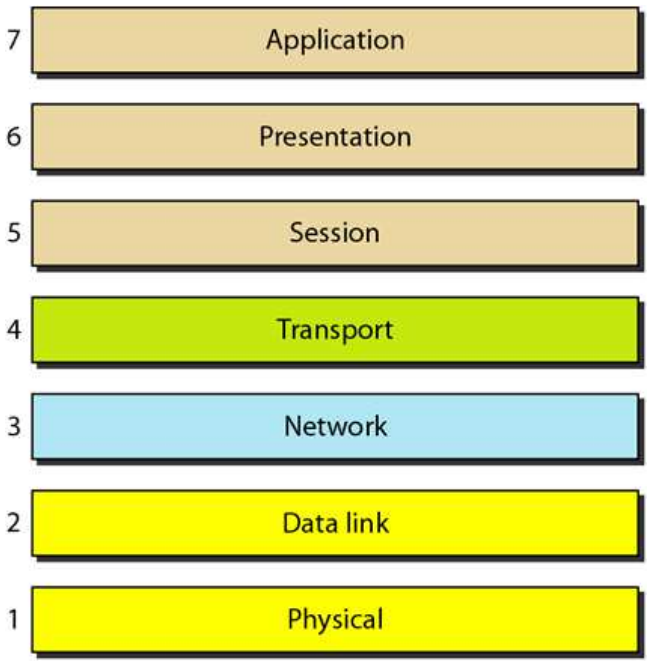
상황을 의미한다.

* 경우에 따라 payload가 40byte가 채 되지 않는데도 padding이 없는 Ethernet frame이 관찰된다. 주로 ARP의 request나 reply packet에서 관찰되는데 이는 사실 NIC에서 padding을 제거하고 OS로 올려보내기 때문에 그렇게 보이는 것뿐이다.

* Ethernet frame은 Padding 뒤에 CRC(Cyclic Redundancy Check)라고 하는 4byte field를 가지고 있다. 이 field가 Wireshark에서는 관찰되지 않는데 그 이유는 앞서 NIC구조 부분에서 설명했듯이, Wireshark이 CRC가 제거된 이후에(수신 시) 혹은 CRC를 NIC이 삽입하기 전에(송신 시) packet을 capture하기 때문이다.

이렇게 ARP packet은 Ethernet frame에 실어 날라진다. 다시 말하면 ARP는 Ethernet의 서비스에 의존한다. 우리가 택배를 보낼 때 택배회사의 서비스에 의존하고, 택배 회사는 트럭 운전사의 서비스에 의존하고, 트럭 운전사는 트럭 자체의 서비스에 의존하게 되는데 이러한 시스템을 layer (계층) 구조를 가진 시스템이라고 한다.

UN(United Nations) 산하의 표준화 단체인 ISO(International Standard Organization)는 1970년대 말 네트워크 시스템의 기능을 아래와 같이 7개의 계층(layer)로 구분한 OSI(Open Systems Interconnect) 구조를 제안하였다.

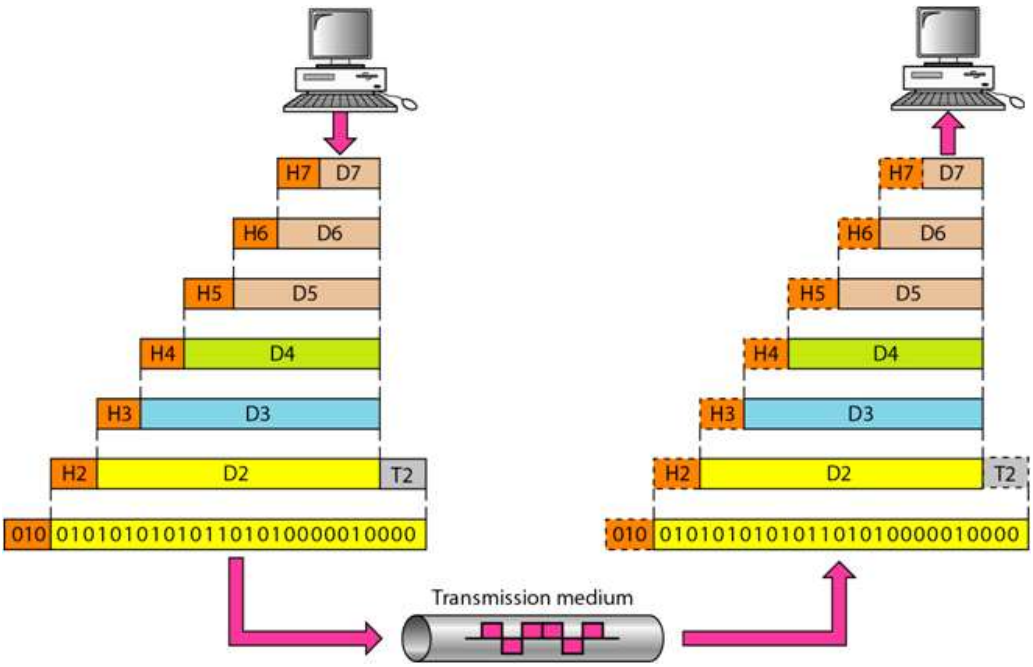


해당 구조에 따르면 7계층 Application이 사용자에 대한 서비스를 담당하고, 6계층이 7계층이 요청하는 서비스를 처리한다. 이런 식으로 상위 계층이 하위 계층에게 서비스를 요청하는 방식이 반복되며, 최하위 1계층은 전송 매체에 전자기 신호 (Electromagnetic signal) 혹은 광학 신호(Optical signal)를 실어서 최종적으로 서비스를 처리하게 된다. 여기서 상위와 하위

는 상대적인 개념이며, 계층을 표현하는 수의 크기가 큰 계층이 상대적으로 상위 계층이다. 즉 4계층이 3계층의 상위 계층이다.

* 서로 인접한 계층 간에만 상·하위의 구별이 의미가 있다. 5계층이 3계층보다 상위 계층이긴 하지만 직접적인 인터페이스가 없으므로 상/하위를 구분하는 의미가 없다.

서비스를 요청받아서 처리하는 과정에서, 이에 필요한 정보를 추가로 실어서 보낼 필요가 생긴다. 예를 들어 4계층인 Transport 계층은 멀리 떨어진 상대방 컴퓨터의 "Process"에 데이터를 나누어 보내고, 이를 받은 상대방 컴퓨터의 Transport 계층은 나누어져 순서 없이 들어온 데이터의 조각들을 순서에 맞게 재조합해서 상위 계층으로 보내는 역할을 맡는데, 이를 위해 3계층인 Network 계층에게 이를 상대방 컴퓨터까지 전달해 달라는 요청을 한다. 이런 서비스를 요청받은 3계층은 해당 컴퓨터까지의 경로를 찾아서 실제로 전달하는 역할을 수행한다. 이렇게 경로를 찾아서 전달하기 위해 컴퓨터의 주소 정보가 필요하다. 3계층은 고유의 주소 체계를 가지고 있어서 4계층이 맡긴 데이터에 3계층 주소 등의 정보를 추가로 덧붙여서 보낸다. 이 덧붙인 정보를 헤더(Header), 특히 3계층 헤더(Layer 3 header)라고 한다. 이러한 관계를 그림으로 살펴보면 아래와 같다.



상위 계층의 서비스를 요청받으면 해당 서비스를 수행하는데 필요한 정보를 헤더에 덧붙여 새로운 데이터를 만든다. 이 과정을 Encapsulation이라고 한다. 거꾸로, 하위 계층에서 데이터를 전달 받으면 해당 계층이 해야 하는 서비스를 다 한 후 더 이상 필요 없어진 헤더를 없애는 과정을 거치는데 이 과정을 Decapsulation이라고 한다. 위 그림에서 H2 등으로 표현된 것이 해당 계층의 Header이다. T2는 2계층에서만 사용되는 Trailer를 의미한다. Header와 비슷한 역할을 하지만 payload의 뒤에 붙는다는 차이가 있다.