PCAP

https://youtu.be/lq5YgaeIoDU

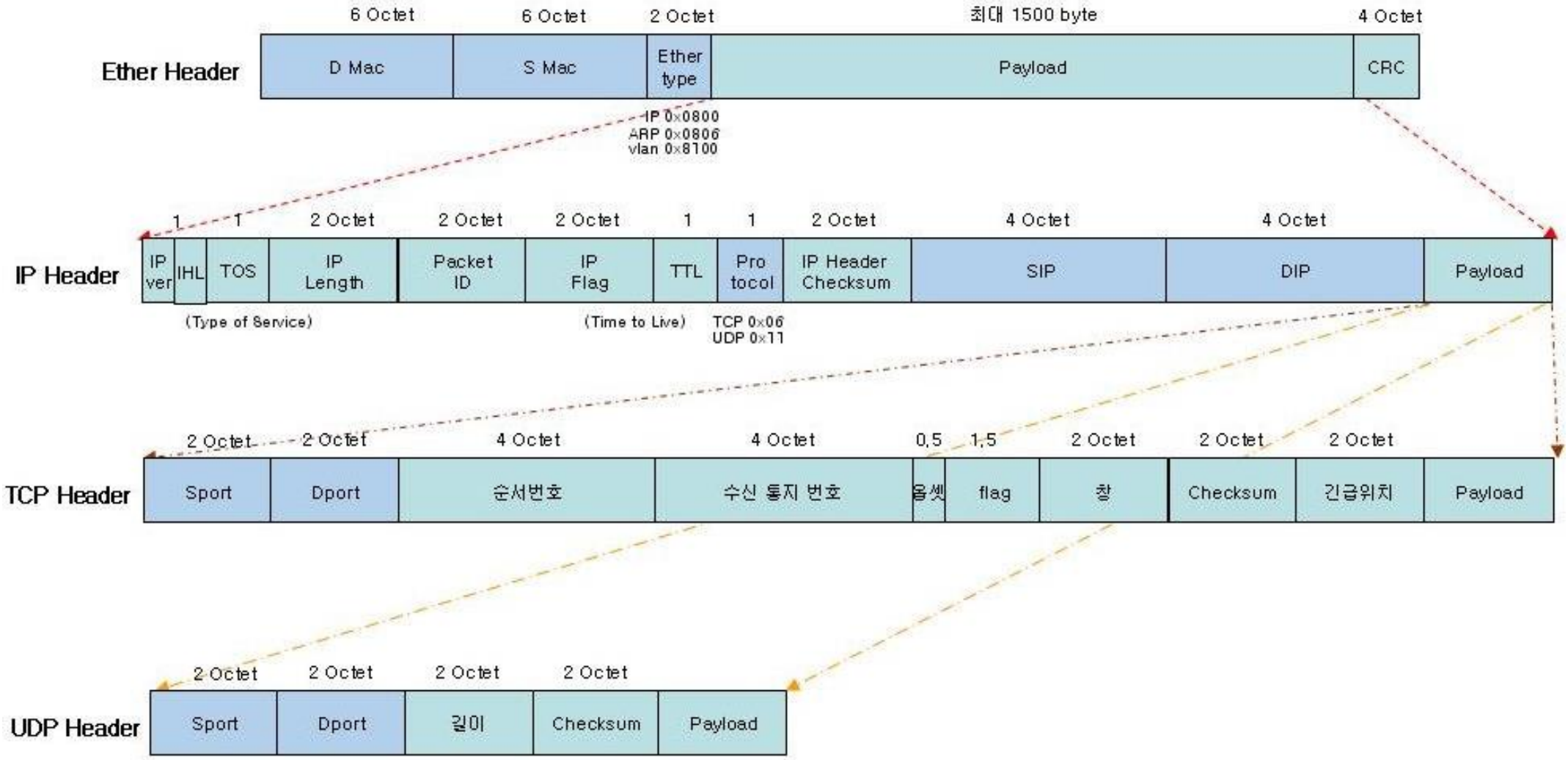
PCAP

- Packet CAPture
- 네트워크 트래픽을 캡처하기 위한 API로 구성
- 이용 라이브러리
  - 유닉스 : Libpcap
  - 윈도우 : WinPcap (Libpcap가 윈도우로 포팅된 것)

패킷

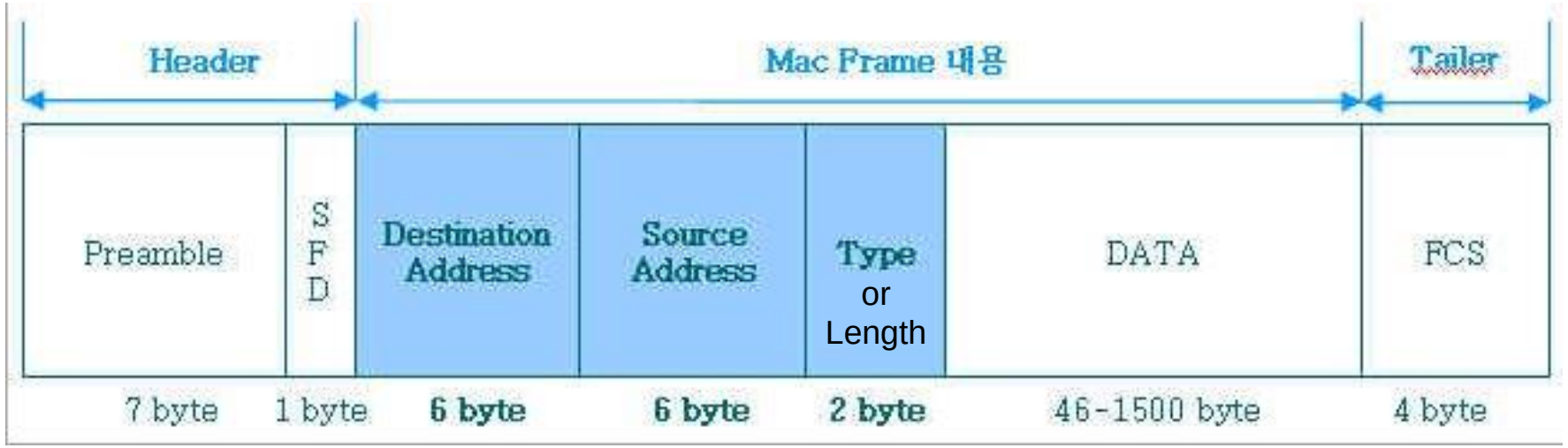
- 데이터를 일정 크기로 자른 것
- 헤더, 데이터, 트레일러로 구성
- 통신망을 통하여 노드에서 노드로 전해짐으로써 전송됨

패킷


Ether Header
6 Octet
6 Octet
2 Octet
최대 1500 byte
4 Octet
D Mac
S Mac
Ether type
Payload
CRC
IP 0x0800
ARP 0x0806
vlan 0x8100
IP Header
1
1
2 Octet
2 Octet
2 Octet
1
1
2 Octet
4 Octet
4 Octet
IP ver
IHL
TOS
IP Length
Packet ID
IP Flag
TTL
Pro tocol
IP Header Checksum
SIP
DIP
Payload
(Type of Service)
(Time to Live)
TCP 0x06
UDP 0x11
TCP Header
2 Octet
2 Octet
4 Octet
4 Octet
0,5
1,5
2 Octet
2 Octet
2 Octet
Sport
Dport
순서번호
수신 통지 번호
옵셋
flag
창
Checksum
긴급위치
Payload
UDP Header
2 Octet
2 Octet
2 Octet
2 Octet
Sport
Dport
길이
Checksum
Payload

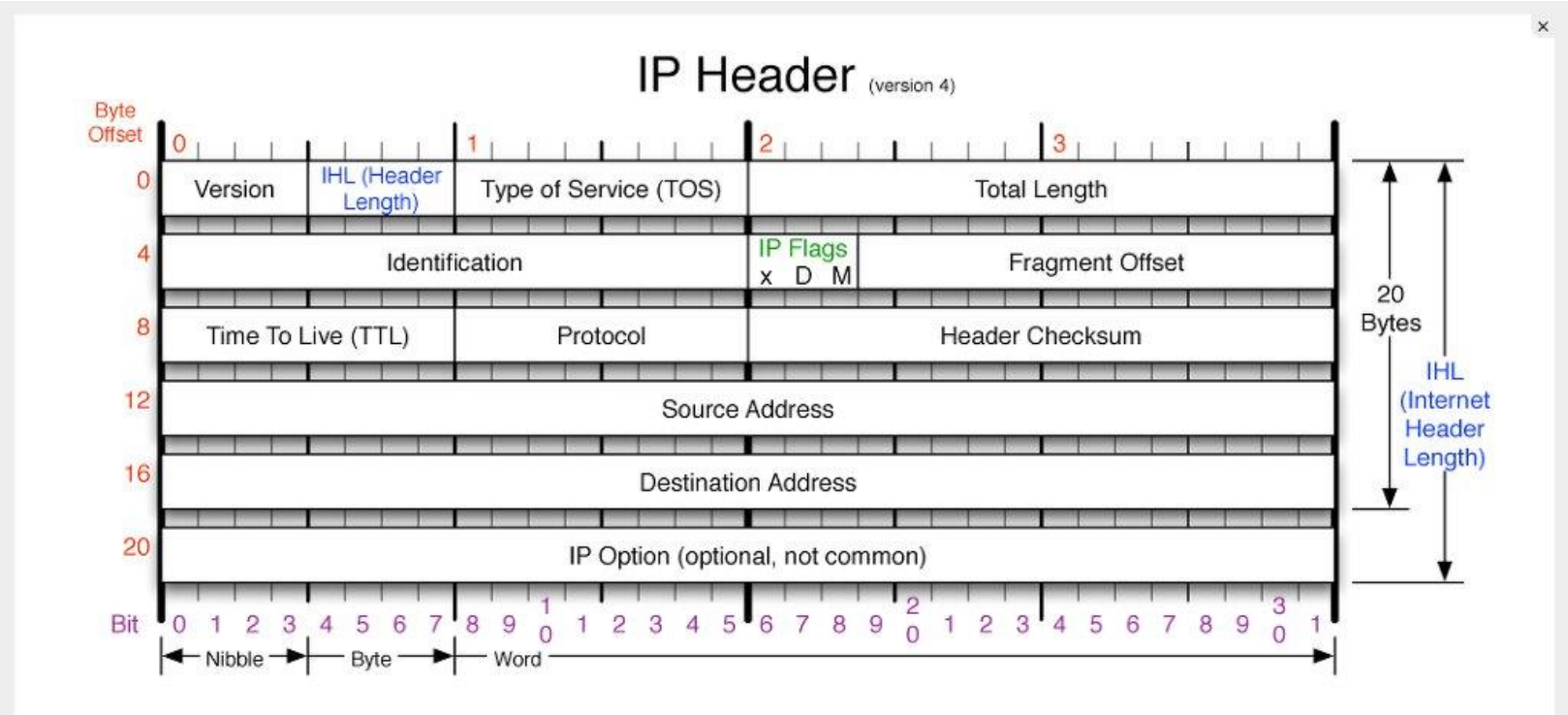
Frame 패킷

- 이더넷 프레임 구조
- OSI 7계층 중 데이터 링크 계층에 존재하며, 대부분의 데이터 링크 계층은 이더넷으로 구성



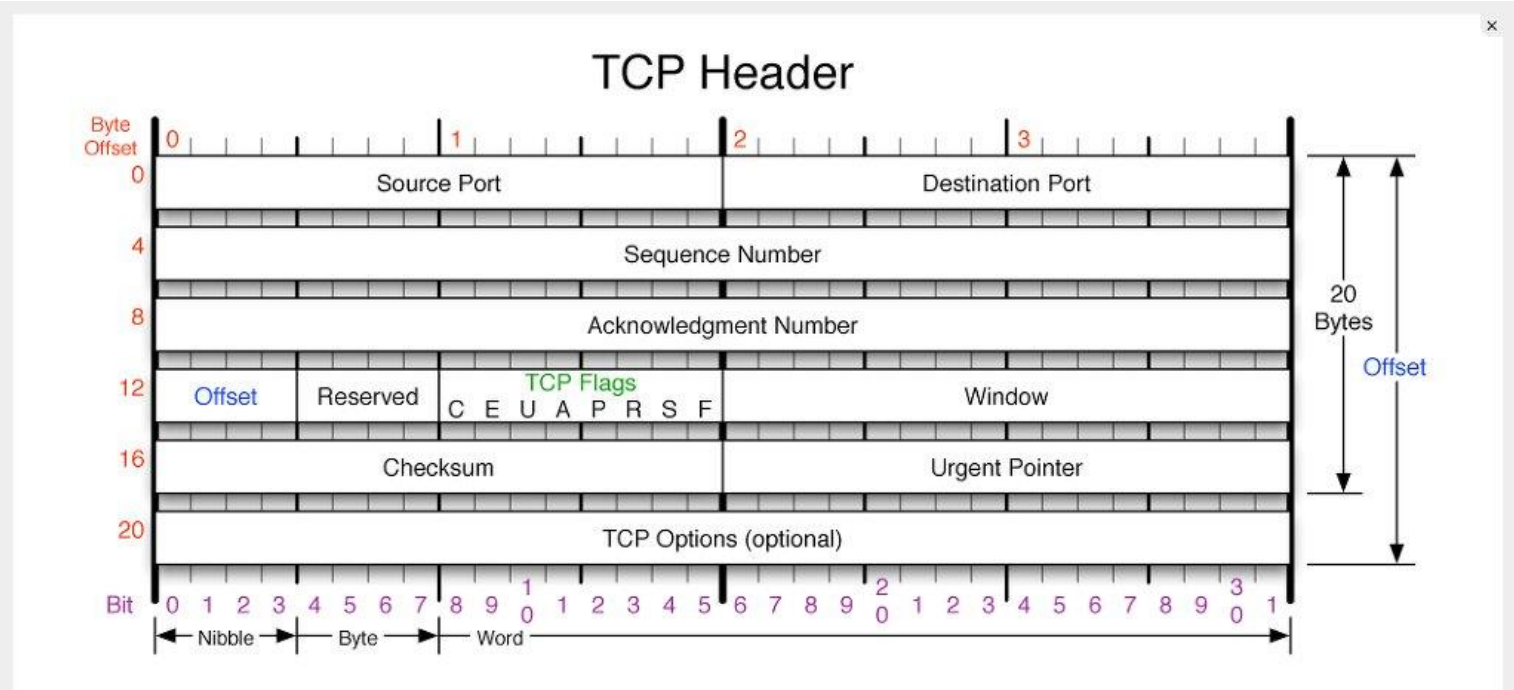
- Preamble : 수신 시스템에 프레임이 도착하는 것을 알려주며, 동기화할 수 있게 함
- SFD (Start of Frame Delimeter) : 프레임의 시작을 알림
- FCS (Frame Check Sequence) : 유효한지 계산을 통해 에러를 판별

IP 패킷



- Type Of Service : 패킷이 얼마나 빨리 처리/전달 되어야 하는가에 대한 정보
- Protocol : 상위 계층의 프로토콜 (TCP/UDP 등)
- Header Checksum : 헤더 부분에 대해 에러 발생 시 정정을 위한 체크

TCP 패킷



- Sequence Number : 패킷 송신 데이터의 일련번호
- Acknowledgement Number : 수신 데이터의 일련번호
- TCP Flags : TCP의 연결 및 종료를 제어
- Window : 메시지 전송 시 흐름 제어

패킷 캡처

- 유닉스에서는 tcpdump, 윈도우에서는 tcpdump가 win32로 포팅된 windump 사용
- tcpdump : CUI에서 실행되는, 패킷 스니퍼 소프트웨어
- 옵션을 통해 특정 조건에 대한 네트워크 인터페이스를 거치는 패킷들을 출력하거나 파일로 저장
- tcpdump -nn -vvv -A -G 3600 -w ./github-%H%M%S.pcap host github.com and port 443
  - 1시간 (3600초) 단위로 github.com과 통신하는 패킷을 pcap 파일로 저장
- 옵션들에 대한 설명

PCAP

- [Global Header] | [Packet Header | Packet Data] | [Packet Header | Packet Data] | ..

PCAP

Global Header

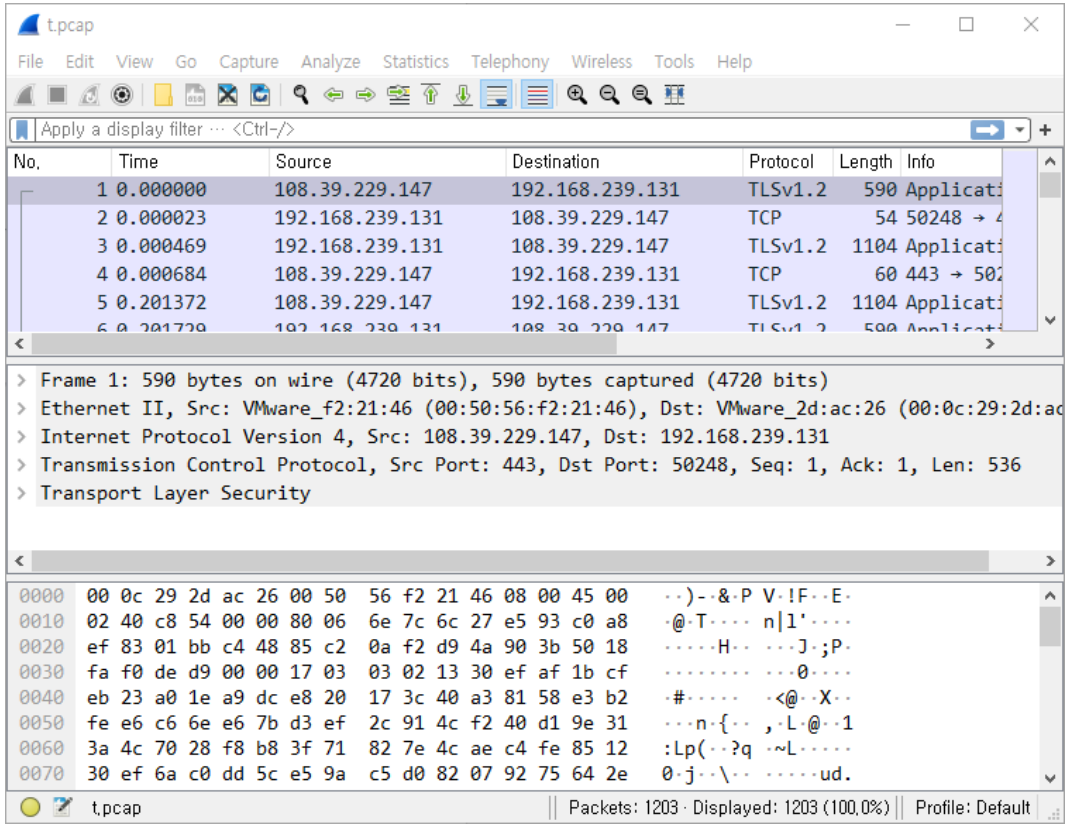
```c
typedef struct pcap_hdr_s {
    uint32_t magic_number; /* 파일 포맷과 바이트오더를 확인하기 위한 필드 */
    uint16_t version_major; /* 파일 포맷의 버전 (메이저) */
    uint16_t version_minor; /* 파일 포맷의 버전 (마이너) */
    int32_t thiszone; /* 타임존과 관련된 정보 */
    uint32_t sigfigs; /* 플러그 정보 */
    uint32_t snaplen; /* 버퍼의 크기 (일반적으로 65535) */
    uint32_t network; /* 데이터 링크 레이어의 헤더 타입 */
} pcap_hdr_t;
```

Packet Header

```c
typedef struct pcaprec_hdr_s {
    uint32_t ts_sec; /* 초단위로 저장된 타임스탬프 */
    uint32_t ts_usec; /* 패킷이 캡처된 마이크로 초 */
    uint32_t incl_len; /* 실제 파일에 저장된 패킷 데이터의 실제 바이트 크기 */
    uint32_t orig_len; /* 네트워크 상의 패킷의 바이트 크기 */
} pcaprec_hdr_t;
```

Wireshark

- 네트워크 패킷을 캡처하고 분석하는 오픈소스 도구



호스트 간 주고받은 패킷

패킷의 상세 내용

16진수 형태의 전체 데이터

파이썬을 이용한 PCAP 분석

- 파이썬에는 PCAP 파일을 분석할 수 있는 다양한 라이브러리 존재
- dpkt, PyShark, PyPCAP 등

```python
import dpkt
import datetime
import socket
import os
import binascii

def mac_addr(address):
    address = binascii.hexlify(address).decode('utf-8')
    return ':'.join('%02x' % ord(b) for b in address)

def ip_addr(address):
    return socket.inet_ntoa(address)

with open('t.pcap', 'rb') as f:
    pcap = dpkt.pcap.Reader(f)

    for timestamp, buf in pcap:
        eth = dpkt.ethernet.Ethernet(buf)
        ip = eth.data

        print('############################################################')
        print('packet length:', ip.len)
        print('timestamp:', timestamp)
        print('mac_src:', mac_addr(eth.src))
        print('mac_dst:', mac_addr(eth.dst))
        print('ip_src:', ip_addr(ip.src))
        print('ip_dst:', ip_addr(ip.dst))
        print('############################################################\n')
```

```
############################################################
packet length: 424
timestamp: 1601978665.735532
mac_src: 30:30:35:30:35:36:66:32:32:31:34:36
mac_dst: 30:30:30:63:32:39:32:64:61:63:32:36
ip_src: 108.39.229.147
ip_dst: 192.168.239.131
############################################################

############################################################
packet length: 2632
timestamp: 1601978665.740276
mac_src: 30:30:30:63:32:39:32:64:61:63:32:36
mac_dst: 30:30:35:30:35:36:66:32:32:31:34:36
ip_src: 192.168.239.131
ip_dst: 108.39.229.147
############################################################

############################################################
packet length: 40
timestamp: 1601978665.740472
mac_src: 30:30:35:30:35:36:66:32:32:31:34:36
mac_dst: 30:30:30:63:32:39:32:64:61:63:32:36
ip_src: 108.39.229.147
ip_dst: 192.168.239.131
############################################################
```

PCAPNG

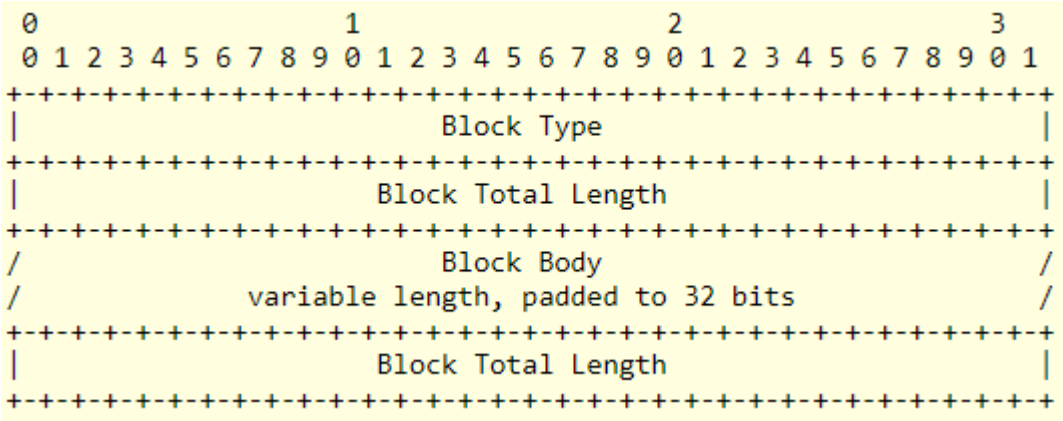
- Packet CAPture Next Generation
- 기존의 pcap을 발전시킨 파일 형식
- 인터페이스 정보, 주석, 이름 확인 정보 등의 메타데이터 추가
- 와이어샤크 최신 버전에서의 패킷파일 저장은 기본적으로 pcapng로 저장됨

```
Section Header
|
+- Interface Description
|  +- Simple Packet
|  +- Enhanced Packet
|  +- Interface Statistics
|
+- Name Resolution
```

블록의 논리 구조

PCAPNG

- Block Type (32비트) : 블록의 식별 ID값
- Block Total Length : 블록의 전체 사이즈
- Block Body : 블록의 내용
- Block Total Length : 블록의 전체 사이즈
  - 뒤에서부터의 탐색 허용을 위해 존재



기본적인 블록의 구조

Block Type Code	Description
0x00000000	Reserved ???
0x00000001	Interface Description Block [section_idb]
0x00000002	Packet Block [appendix_pb]
0x00000003	Simple Packet Block [section_spb]
0x00000004	Name Resolution Block [section_nrb]
0x00000005	Interface Statistics Block [section_isb]
0x00000006	Enhanced Packet Block [section_epb]
0x00000007	IRIG Timestamp Block (requested by Gianluca Varenni <gianluca.varenni@cacetech.com>, CACE Technologies LLC)
0x00000008	ARINC 429 in AFDX Encapsulation Information Block (requested by Gianluca Varenni <gianluca.varenni@cacetech.com>, CACE Technologies LLC)
0x00000BAD	Custom Block that rewriters can copy into new files [section_custom_block]
0x40000BAD	Custom Block that rewriters should not copy into new files [section_custom_block]
0x0A0D0D0A	Section Header Block [section_shb]
0x0A0D0A00-0x0A0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0A-0xFF0A0D0A	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0D-0xFF0A0D0D	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x0D0D0A00-0x0D0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the FTP protocol in text mode.
0x80000000-0xFFFFFFFF	Reserved for local use.

Block Type Codes

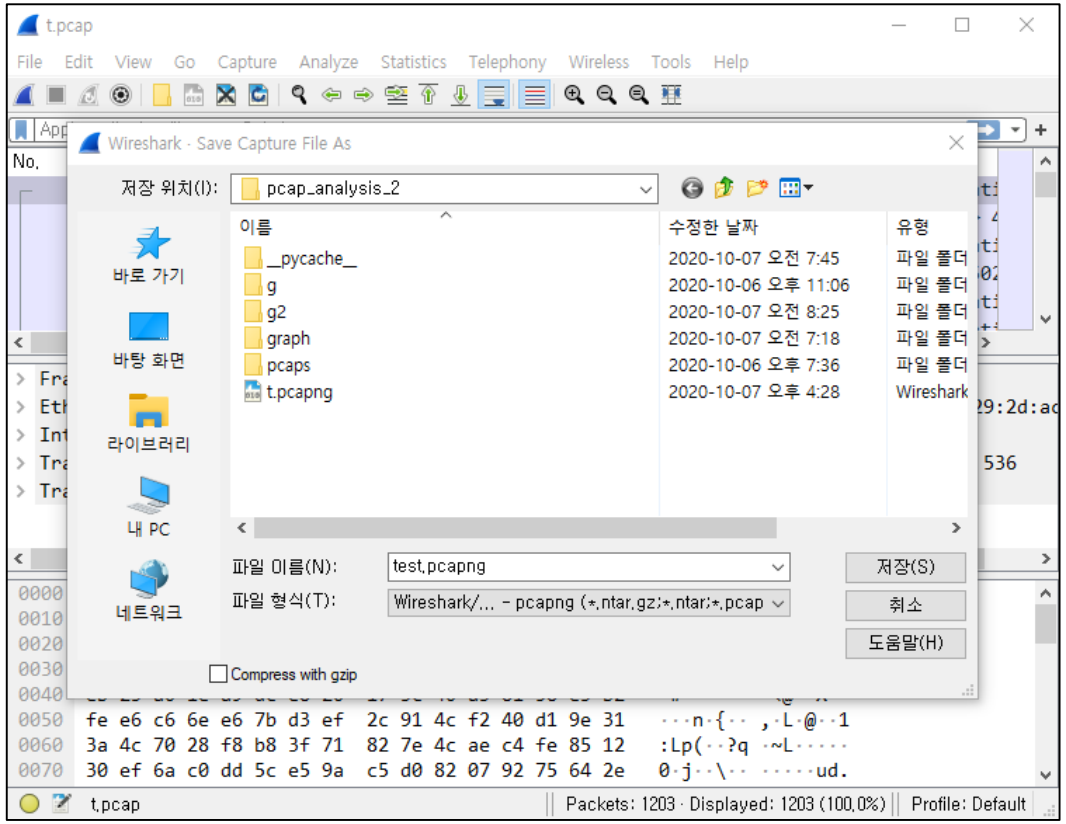
PCAPNG

- wireshark의 editcap을 이용하여 pcap ↔ pcapng 간 변환 가능

- pcapng → pcap
- editcap -F libpcap -T ether test.pcapng test.pcap

- pcap → pcapng
- editcap -F pcapng test.pcap test.pcapng
- tshark -F pcapng -r test.pcap test.pcapng

혹은 wireshark 프로그램을
직접 실행시켜서도 변환 가능



Python-PCAPNG

- https://blog.ackhax.com/2018/11/04/pcap-parsing-with-python-pcapng/

```python
from pcapng import FileScanner
from pcapng import blocks
import binascii
import socket

def get_pcap_packet_blocks(filename):
    """
    Reads a pcap file and creates a list of EnhancedPacket blocks from the file.
    """
    packet_blocks = []
    with open(filename, 'rb') as fp:
        scanner = FileScanner(fp)
        for block in scanner:
            if isinstance(block, blocks.EnhancedPacket):
                packet_blocks.append(block)
    return packet_blocks

class EthernetFrame():
    def __init__(self, packet_bytes):
        self._parse_packet(packet_bytes)

    def _parse_packet(self, packet_bytes):
        self.dst = packet_bytes[0:6]
        self.src = packet_bytes[6:12]
        self.type = packet_bytes[12:14]
        self.data = packet_bytes[14:]

    def __str__(self):
        return 'EthernetFrame:\nDestination: {}\nSource: {}\nType: {}\nData: {}'.format(self.dst, self.src, self.type, self.data)

def get_eth_frame(packet_block):
    # Check that the block's associated interface is the LINKTYPE_ETHERNET (link_type = 1)
    # Src: https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html#appendixLinkTypes
    if not packet_block.interface.link_type == 1:
        return None

    packet_data = packet_block.packet_data
    ethernet_frame = EthernetFrame(packet_data)
    return ethernet_frame

class IPv4_Packet():
    def __init__(self, data):
        self._parse_packet(data)


    def _parse_packet(self, data):
        self.version = data[0] >> 4

        # extract header length (number of 32-bit words in the header)
```

```
############################################################
packet_len: 60
timestamp: 1601978665.7404819
src_mac: 30:30:35:30:35:36:66:32:32:31:34:36
dst_mac: 30:30:30:63:32:39:32:64:61:63:32:36
src_ip: 108.39.229.147
dst_ip: 192.168.239.131
proto: 6
############################################################

############################################################
packet_len: 1104
timestamp: 1601978665.750236
src_mac: 30:30:30:63:32:39:32:64:61:63:32:36
dst_mac: 30:30:35:30:35:36:66:32:32:31:34:36
src_ip: 192.168.239.131
dst_ip: 108.39.229.147
proto: 6
############################################################

############################################################
packet_len: 60
timestamp: 1601978665.750429
src_mac: 30:30:35:30:35:36:66:32:32:31:34:36
dst_mac: 30:30:30:63:32:39:32:64:61:63:32:36
src_ip: 108.39.229.147
dst_ip: 192.168.239.131
proto: 6
############################################################
```

Q & A