2019.12.3

Packet Sniffing

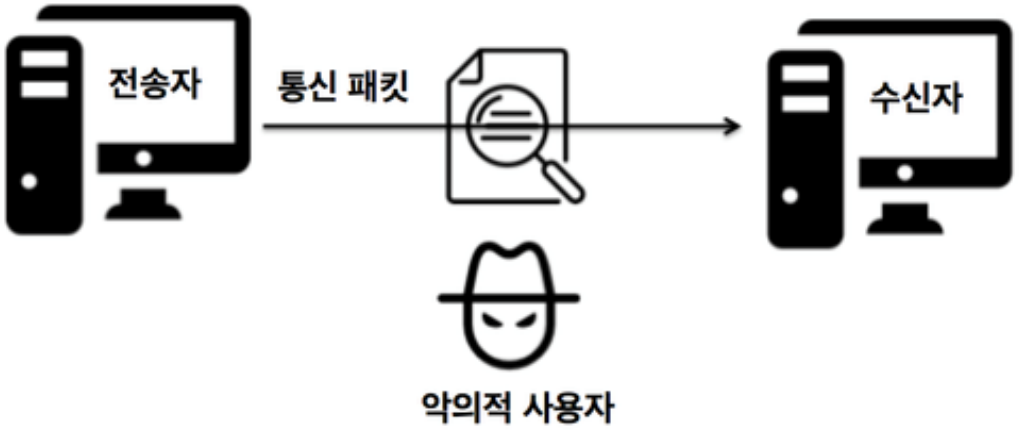
SCP_이예준

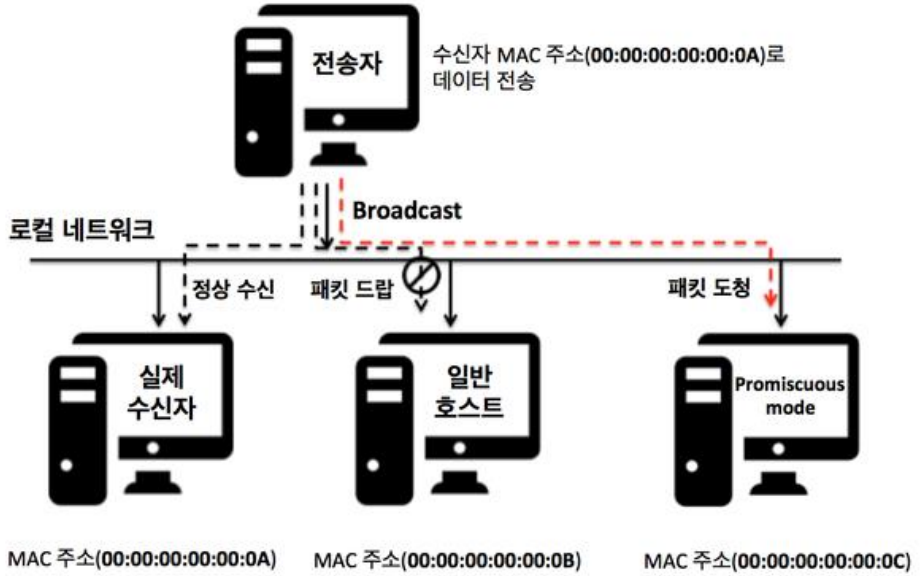
목차

- Packet Sniffing
- Ethernet Header
- ARP Header
- IP Header
- ICMP Header
- TCP Header
- UDP Header
- 실습 (Python)

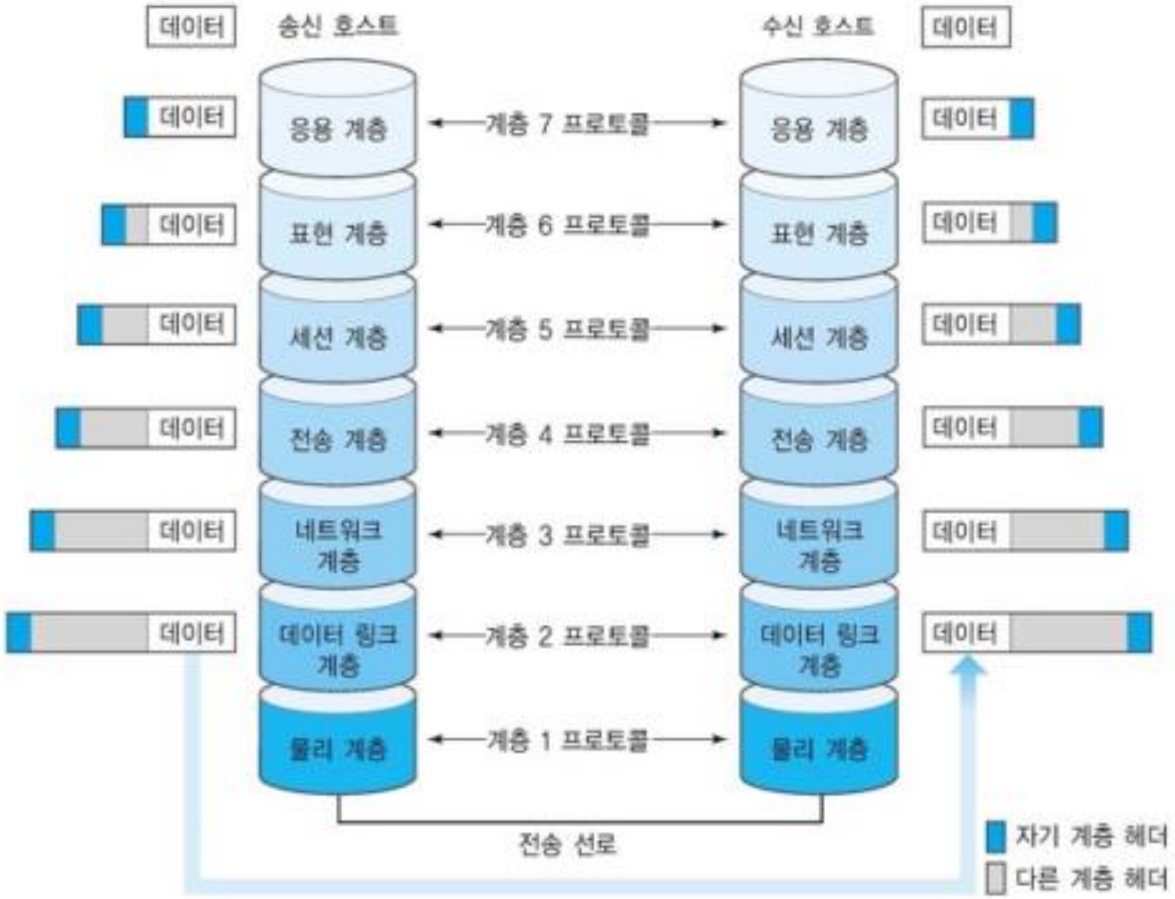
Sniffing

코를 킁킁거리다, 냄새를 맡다
네트워크 상에서 자신이 아닌 다른 상대방들의 패킷 교환을 몰래 엿보는 것
=> 네트워크 트래픽 도청



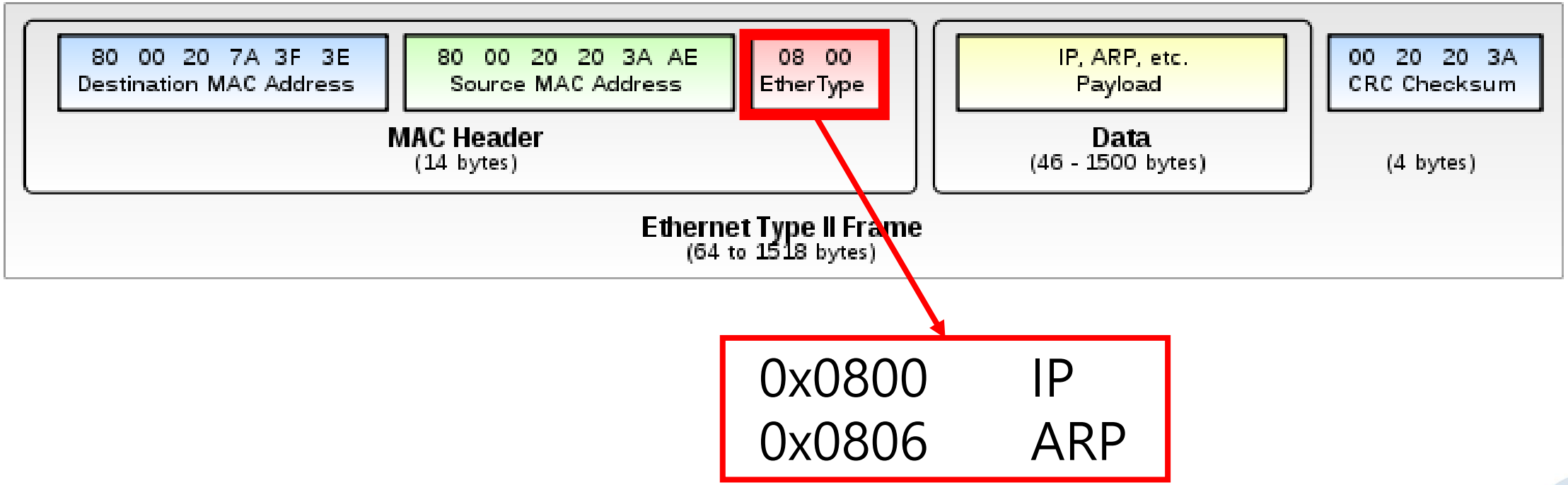


OSI 7 layer

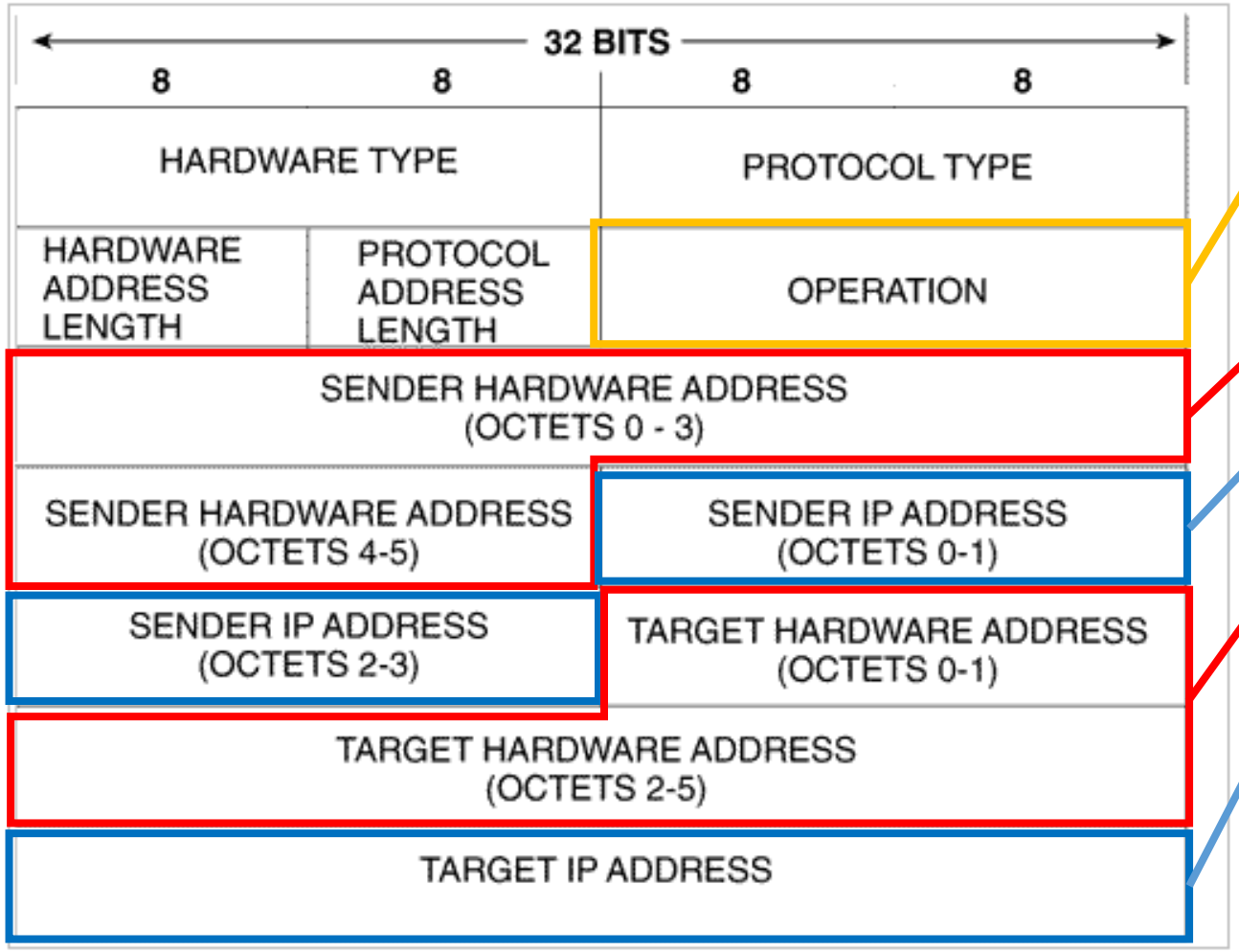

데이터
송신 호스트
수신 호스트
데이터
응용 계층
계층 7 프로토콜
표현 계층
계층 6 프로토콜
세션 계층
계층 5 프로토콜
전송 계층
계층 4 프로토콜
네트워크 계층
계층 3 프로토콜
데이터 링크 계층
계층 2 프로토콜
물리 계층
계층 1 프로토콜
전송 선로
자기 계층 헤더
다른 계층 헤더

Packet Structure

Ethernet Header

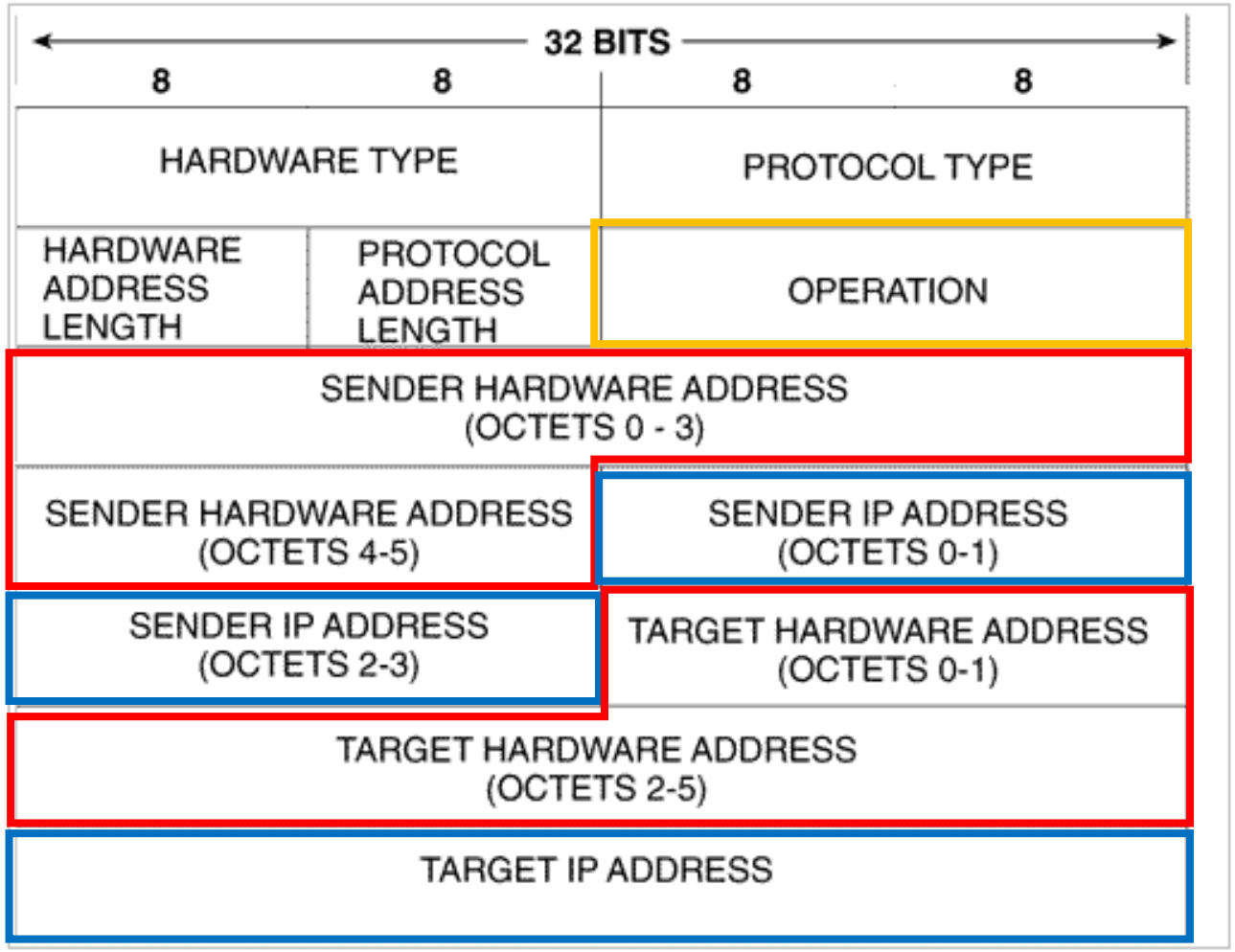


ARP Header



- OPERATION
  1 : ARP Request (요청)
  2 : ARP Reply (응답)
- 출발지 MAC
- 출발지 IP
- 목적지 MAC
- 목적지 IP

ARP Header

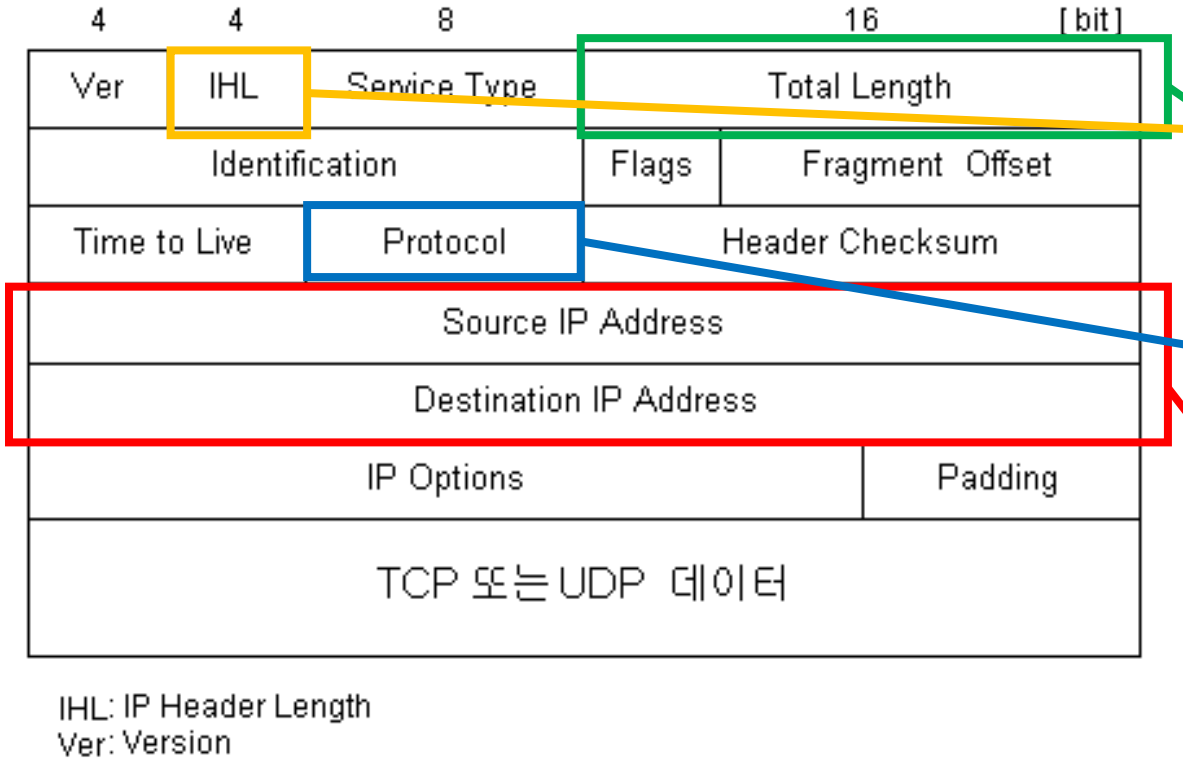


ex)
192.168.10.1 PC 1
192.168.10.128 PC 2

```
Layer 3: ARP
Operation: 1 -> ARP Request
Source MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 192.168.10.128
Target MAC Address: 00:00:00:00:00:00
Target IP Address: 192.168.10.1

Layer 3: ARP
Operation: 2 -> ARP Reply
Source MAC Address: 00:50:56:C0:00:08
Source IP Address: 192.168.10.1
Target MAC Address: 00:0C:29:2E:3D:01
Target IP Address: 192.168.10.128
```

IP Header



- IP Header Length
- Total Length
- Protocol
  1 : ICMP
  6 : TCP
  17 : UDP
- 출발지 IP, 목적지 IP

그림 1-8 IP 패킷 구조

IP Header

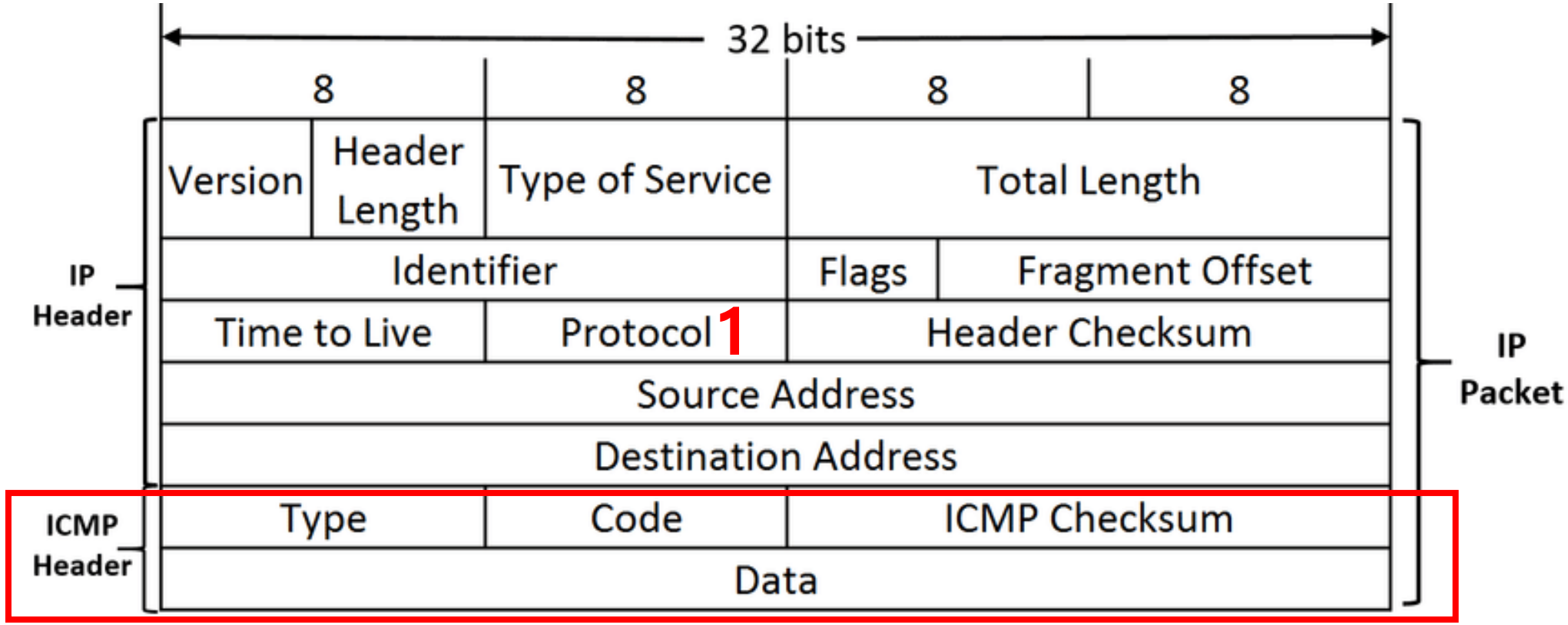
클라이언트 요청

```
Layer 3: IP
Source MAC Address: 00:0C:29:2E:3D:01
Destination MAC Address: 00:50:56:F9:1C:E0  Gateway
Source IP Address: 192.168.10.128
Destination IP Address: 175.213.35.39  server
```

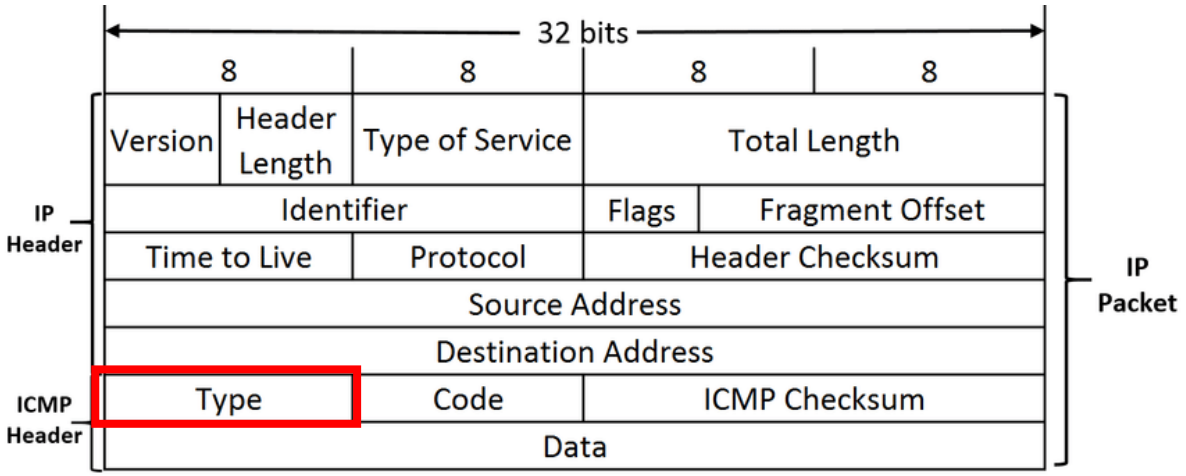
서버 응답

```
Layer 3: IP
Source MAC Address: 00:50:56:F9:1C:E0 Gateway
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 175.213.35.39 server
Destination IP Address: 192.168.10.128
```

ICMP Header


32 bits
8
8
8
8
Version
Header Length
Type of Service
Total Length
Identifier
Flags
Fragment Offset
Time to Live
Protocol 1
Header Checksum
Source Address
Destination Address
Type
Code
ICMP Checksum
Data
IP Header
ICMP Header
IP Packet

ICMP Header



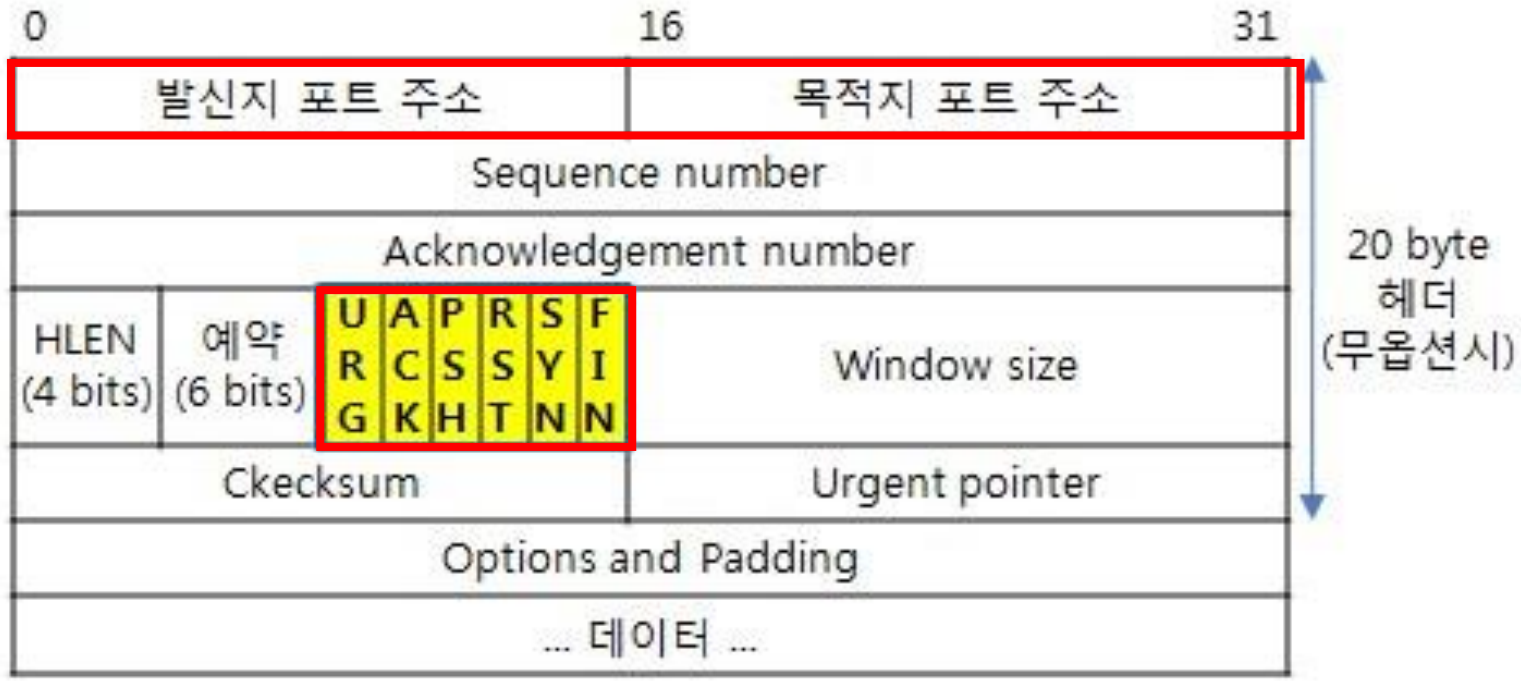

```
Layer 3: ICMP
ICMP Type: 8 -> Echo Request
Source MAC Address: 00:0C:29:2E:3D:01
Destination MAC Address: 00:50:56:F9:1C:E0
Source IP Address: 192.168.10.128
Destination IP Address: 8.8.8.8
Data: b'\x17\x04\x00\x05x\xd2\xe3]\x00\x00\x00\x00\xe2\xde\x04\x00\x00\x00\x00\x
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,
-./01234567'

Layer 3: ICMP
ICMP Type: 0 -> Echo Reply
Source MAC Address: 00:50:56:F9:1C:E0
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 8.8.8.8
Destination IP Address: 192.168.10.128
Data: b'\x17\x04\x00\x05x\xd2\xe3]\x00\x00\x00\x00\xe2\xde\x04\x00\x00\x00\x00\x
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,
-./01234567'
```

Type
0 : 응답
3 : 도달 불가능
8 : 요청
11 : TTL 만료

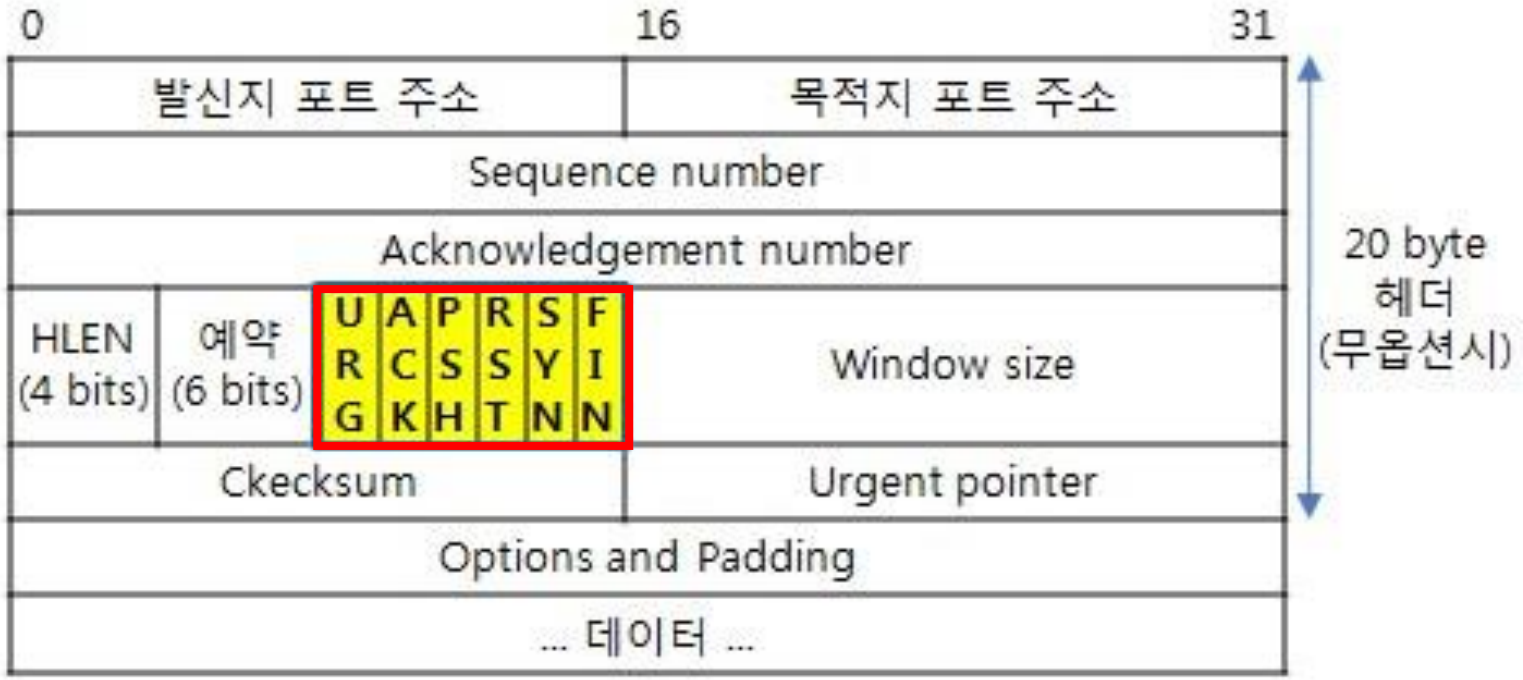
TCP Header

Port
0 ~ 1023: 잘 알려진 포트
1024~49151: IANA에서 관리하는 포트
49152 ~ 65535: 자유롭게 사용 가능한 포트



Port	Service
21	FTP
23	Telnet
25	SMTP
80	HTTP
110	POP3
194	Internet Relay Chat (IRC)
443	Secure HTTP (HTTPS)

TCP Header



Flag bit
- URG : 긴급 데이터 포함
- ACK : 확인 응답
- PSH : 데이터 포함
- RST : 수신 거부
- SYN : 연결할 때 확인
- FIN : 연결 해제

TCP Header

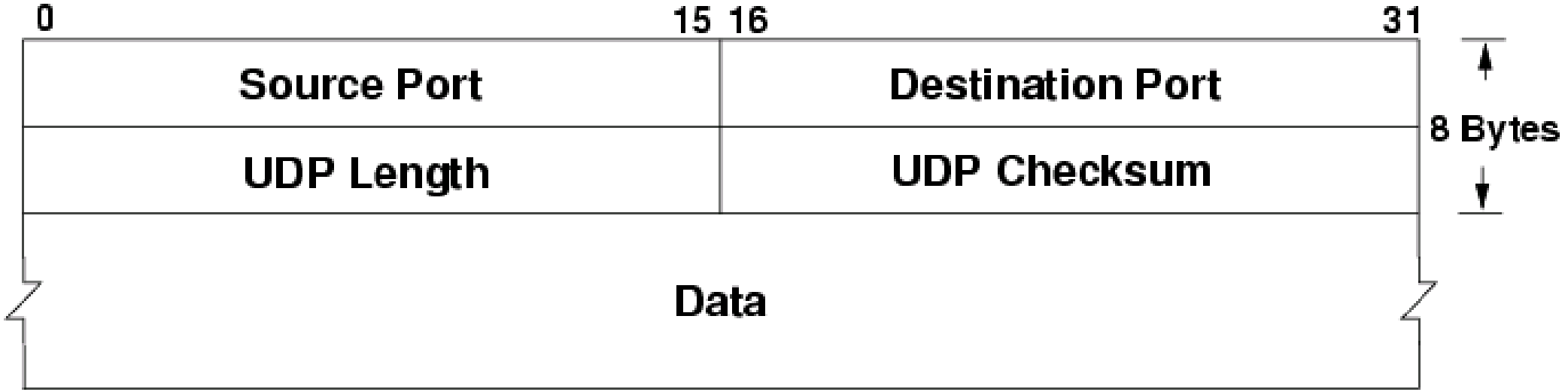
클라이언트 요청

```
Layer 3: IP
Source MAC Address: 00:0C:29:2E:3D:01
Destination MAC Address: 00:50:56:F9:1C:E0
Source IP Address: 192.168.10.128
Destination IP Address: 175.213.35.39
Layer 4: TCP
Source IP Port: Undefined 40364
Destination Port: HTTP 80
Data: b'GET /favicon.ico HTTP/1.1\r\nHost: test.gilgil.net\r\nUser-Agent: Mozilla/5.0 (X11; Lin
ux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0\r\nAccept: text/html,application/xhtml+xml,appl
ication/xml;q=0.9,*/*;q=0.8\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflat
e\r\nConnection: keep-alive\r\n\r\n'
```

서버 응답

```
Layer 3: IP
Source MAC Address: 00:50:56:F9:1C:E0
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 175.213.35.39
Destination IP Address: 192.168.10.128
Layer 4: TCP
Source IP Port: HTTP 80
Destination Port: Undefined 40364
Data: b'HTTP/1.1 404 Not Found\r\nDate: Sun, 01 Dec 2019 14:56:59 GMT\r\nServer: Apache\r\nCont
ent-Length: 311\r\nKeep-Alive: timeout=5, max=99\r\nConnection: Keep-Alive\r\nContent-Type: tex
t/html; charset=iso-8859-1\r\n\r\n<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n<html><he
ad>\n<title>404 Not Found</title>\n</head><body>\n<h1>Not Found</h1>\n<p>The requested URL /fav
icon.ico was not found on this server.</p>\n<hr>\n<address>Apache Server at <a href="mailto:adm
in@localhost">test.gilgil.net</a> Port 80</address>\n</body></html>\n'
```

UDP Header


0
15 16
31
Source Port
Destination Port
UDP Length
UDP Checksum
8 Bytes
Data

UDP Header

클라이언트 요청

```
Layer 3: IP
Source MAC Address: 00:50:56:F9:1C:E0
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 192.168.10.2
Destination IP Address: 192.168.10.128
Layer 4: UDP
Source IP Port: DNS 53
Destination Port: Undefined 52859
Data: b'q\x11\x81\x80\x00\x01\x00\x02\x00\x02\x00\x04\x03www\x08facebook\x03com\x00\x00\x01\x00
\x01\xc0\x0c\x00\x05\x00\x01\x00\x00\x00\x05\x00\x11\tstar-mini\x04c10r\xc0\x10\xc0.\x00\x01\x0
0\x01\x00\x00\x00\x05\x00\x04\x1f\rF$\xc08\x00\x02\x00\x01\x00\x00\x00\x05\x00\x07\x01a\x02ns\x
c08\xc08\x00\x02\x00\x01\x00\x00\x00\x05\x00\x04\x01b\xc0]\xc0[\x00\x01\x00\x01\x00\x00\x00\x05
\x00\x04E\xab\xef\x0b\xc0n\x00\x01\x00\x01\x00\x00\x00\x05\x00\x04E\xab\xff\x0b\xc0[\x00\x1c\x0
0\x01\x00\x00\x00\x05\x00\x10*\x03(\x80\xff\xfe\x00\x0b\xfa\xce\xb0\x0c\x00\x00\x00\x99\xc0n\x0
0\x1c\x00\x01\x00\x00\x00\x05\x00\x10*\x03(\x80\xff\xff\x00\x0b\xfa\xce\xb0\x0c\x00\x00\x00\x99
'
```

서버 응답

```
Layer 3: IP
Source MAC Address: 00:50:56:F9:1C:E0
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 192.168.10.2
Destination IP Address: 192.168.10.128
Layer 4: UDP
Source IP Port: DNS 53
Destination Port: Undefined 52859
Data: b'M\x1c\x81\x80\x00\x01\x00\x02\x00\x02\x00\x04\x03www\x08facebook\x03com\x00\x00\x1c\x00
\x01\xc0\x0c\x00\x05\x00\x01\x00\x00\x00\x05\x00\x11\tstar-mini\x04c10r\xc0\x10\xc0.\x00\x1c\x0
0\x01\x00\x00\x00\x05\x00\x10*\x03(\x80\xf1\r\x00\x83\xfa\xce\xb0\x0c\x00\x00%\xde\xc08\x00\x02
\x00\x01\x00\x00\x00\x05\x00\x07\x01a\x02ns\xc08\xc08\x00\x02\x00\x01\x00\x00\x00\x05\x00\x04\x
01b\xc0i\xc0g\x00\x01\x00\x01\x00\x00\x00\x05\x00\x04E\xab\xef\x0b\xc0z\x00\x01\x00\x01\x00\x00
\x00\x05\x00\x04E\xab\xff\x0b\xc0g\x00\x1c\x00\x01\x00\x00\x00\x05\x00\x10*\x03(\x80\xff\xfe\x0
0\x0b\xfa\xce\xb0\x0c\x00\x00\x00\x99\xc0z\x00\x1c\x00\x01\x00\x00\x00\x05\x00\x10*\x03(\x80\xf
f\xff\x00\x0b\xfa\xce\xb0\x0c\x00\x00\x00\x99'
```

스니핑 실습

운영체제 : Linux

프로그래밍 언어 : Python

Sniffer.py

포트번호, Type 분류 딕셔너리
Unpacking 함수 선언
MAC 주소 형식 포맷
IP 주소 형식 포맷

Sniffing.py

각 헤더 구별 (IF문)
로우 소켓 생성
패킷 정보 출력 (Print)

sniffer.py

```python
import socket
import struct

protocols = {1:'ICMP',6:'TCP',7:'ECHO',17:'UDP',20:'FTP',21:'FTP',22:'SSH',
             23:'Telnet',25:'SMTP',53:'DNS',67:'DHCP',68:'DHCP',69:'TFTP',
             80:'HTTP',110:'POP3',143:'IMAP4',161:'SNMP',443:'HTTPS',520:'RIP'}

arp_op = {1:'ARP Request', 2:'ARP Reply', 3:'RARP Request', 4:'RARP Reply'}

icmp_type = {0:'Echo Reply', 3:'Destination Network Unreachable',
             5:'Redirect', 8:'Echo Request',11:'TTL expired in trans'}

IP_Type = 0x0800
ARP_Type = 0x0806
Packet_LEN_ETH = 14
Packet_LEN_IP = 20
Packet_LEN_ARP = 28
Packet_LEN_TCP = 20
Packet_LEN_UDP = 8
Packet_LEN_ICMP = 4

def Get_MAC_Addr(input_MAC):
    output_MAC = map('{:02X}'.format, input_MAC)
    return ':'.join(output_MAC)

def Get_IP_Addr(input_IP):
    output_IP = map('{:0d}'.format, input_IP)
    return '.'.join(output_IP)

def Get_Eth_Header(packet):
    L3_Type = 'Undefined'
    dst_MAC, src_MAC, L3_Type = struct.unpack('! 6s 6s H',packet)
    return dst_MAC,src_MAC,L3_Type

def Get_IP_Header(packet):
    IP_Header = struct.unpack('!B B H H H B B H 4s 4s',packet[:Packet_LEN_IP])
    return IP_Header

def Get_TCP_Header(packet):
    TCP_Header = struct.unpack('!H H 2H 2H B B H H H',packet[:Packet_LEN_TCP])
    return TCP_Header

def Get_UDP_Header(packet):
    UDP_Header = struct.unpack('!H H H H',packet[:Packet_LEN_UDP])
    return UDP_Header

def Get_ARP_Header(packet):
    ARP_Header = struct.unpack('!H H B B H 6s 4s 6s 4s',packet[:Packet_LEN_ARP])
    return ARP_Header

def Get_ICMP_Header(packet):
    ICMP_Header = struct.unpack('!B B H',packet[:Packet_LEN_ICMP])
    return ICMP_Header
```


```python
import socket
import struct

protocols = {1:'ICMP',6:'TCP',7:'ECHO',17:'UDP',20:'FTP',21:'FTP',22:'SSH',
             23:'Telnet',25:'SMTP',53:'DNS',67:'DHCP',68:'DHCP',69:'TFTP',
             80:'HTTP',110:'POP3',143:'IMAP4',161:'SNMP',443:'HTTPS',520:'RIP'}

arp_op = {1:'ARP Request', 2:'ARP Reply', 3:'RARP Request', 4:'RARP Reply'}

icmp_type = {0:'Echo Reply', 3:'Destination Network Unreachable',
             5:'Redirect', 8:'Echo Request',11:'TTL expired in trans'}

IP_Type = 0x0800
ARP_Type = 0x0806
Packet_LEN_ETH = 14
Packet_LEN_IP = 20
Packet_LEN_ARP = 28
Packet_LEN_TCP = 20
Packet_LEN_UDP = 8
Packet_LEN_ICMP = 4
```

sniffer.py

```python
import socket
import struct

protocols = {1:'ICMP',6:'TCP',7:'ECHO',17:'UDP',20:'FTP',21:'FTP',22:'SSH',
             23:'Telnet',25:'SMTP',53:'DNS',67:'DHCP',68:'DHCP',69:'TFTP',
             80:'HTTP',110:'POP3',143:'IMAP4',161:'SNMP',443:'HTTPS',520:'RIP'}

arp_op = {1:'ARP Request', 2:'ARP Reply', 3:'RARP Request', 4:'RARP Reply'}

icmp_type = {0:'Echo Reply', 3:'Destination Network Unreachable',
             5:'Redirect', 8:'Echo Request',11:'TTL expired in trans'}

IP_Type = 0x0800
ARP_Type = 0x0806
Packet_LEN_ETH = 14
Packet_LEN_IP = 20
Packet_LEN_ARP = 28
Packet_LEN_TCP = 20
Packet_LEN_UDP = 8
Packet_LEN_ICMP = 4

def Get_MAC_Addr(input_MAC):
    output_MAC = map('{:02X}'.format, input_MAC)
    return ':'.join(output_MAC)

def Get_IP_Addr(input_IP):
    output_IP = map('{:0d}'.format, input_IP)
    return '.'.join(output_IP)

def Get_Eth_Header(packet):
    L3_Type = 'Undefined'
    dst_MAC, src_MAC, L3_Type = struct.unpack('! 6s 6s H',packet)
    return dst_MAC,src_MAC,L3_Type

def Get_IP_Header(packet):
    IP_Header = struct.unpack('!B B H H H B B H 4s 4s',packet[:Packet_LEN_IP])
    return IP_Header

def Get_TCP_Header(packet):
    TCP_Header = struct.unpack('!H H 2H 2H B B H H H',packet[:Packet_LEN_TCP])
    return TCP_Header

def Get_UDP_Header(packet):
    UDP_Header = struct.unpack('!H H H H',packet[:Packet_LEN_UDP])
    return UDP_Header

def Get_ARP_Header(packet):
    ARP_Header = struct.unpack('!H H B B H 6s 4s 6s 4s',packet[:Packet_LEN_ARP])
    return ARP_Header

def Get_ICMP_Header(packet):
    ICMP_Header = struct.unpack('!B B H',packet[:Packet_LEN_ICMP])
    return ICMP_Header
```

```python
def Get_MAC_Addr(input_MAC):
    output_MAC = map('{:02X}'.format, input_MAC)
    return ':'.join(output_MAC)

def Get_IP_Addr(input_IP):
    output_IP = map('{:0d}'.format, input_IP)
    return '.'.join(output_IP)
```

sniffer.py

```python
import socket
import struct

protocols = {1:'ICMP',6:'TCP',7:'ECHO',17:'UDP',20:'FTP',21:'FTP',22:'SSH',
             23:'Telnet',25:'SMTP',53:'DNS',67:'DHCP',68:'DHCP',69:'TFTP',
             80:'HTTP',110:'POP3',143:'IMAP4',161:'SNMP',443:'HTTPS',520:'RIP'}

arp_op = {1:'ARP Request', 2:'ARP Reply', 3:'RARP Request', 4:'RARP Reply'}

icmp_type = {0:'Echo Reply', 3:'Destination Network Unreachable',
             5:'Redirect', 8:'Echo Request',11:'TTL expired in trans'}

IP_Type = 0x0800
ARP_Type = 0x0806
Packet_LEN_ETH = 14
Packet_LEN_IP = 20
Packet_LEN_ARP = 28
Packet_LEN_TCP = 20
Packet_LEN_UDP = 8
Packet_LEN_ICMP = 4

def Get_MAC_Addr(input_MAC):
    output_MAC = map('{:02X}'.format, input_MAC)
    return ':'.join(output_MAC)

def Get_IP_Addr(input_IP):
    output_IP = map('{:0d}'.format, input_IP)
    return '.'.join(output_IP)

def Get_Eth_Header(packet):
    L3_Type = 'Undefined'
    dst_MAC, src_MAC, L3_Type = struct.unpack('! 6s 6s H',packet)
    return dst_MAC,src_MAC,L3_Type

def Get_IP_Header(packet):
    IP_Header = struct.unpack('!B B H H H B B H 4s 4s',packet[:Packet_LEN_IP])
    return IP_Header

def Get_TCP_Header(packet):
    TCP_Header = struct.unpack('!H H 2H 2H B B H H H',packet[:Packet_LEN_TCP])
    return TCP_Header

def Get_UDP_Header(packet):
    UDP_Header = struct.unpack('!H H H H',packet[:Packet_LEN_UDP])
    return UDP_Header

def Get_ARP_Header(packet):
    ARP_Header = struct.unpack('!H H B B H 6s 4s 6s 4s',packet[:Packet_LEN_ARP])
    return ARP_Header

def Get_ICMP_Header(packet):
    ICMP_Header = struct.unpack('!B B H',packet[:Packet_LEN_ICMP])
    return ICMP_Header
```


```python
def Get_Eth_Header(packet):
    L3_Type = 'Undefined'
    dst_MAC, src_MAC, L3_Type = struct.unpack('! 6s 6s H',packet)
    return dst_MAC,src_MAC,L3_Type

def Get_IP_Header(packet):
    IP_Header = struct.unpack('!B B H H H B B H 4s 4s',packet[:Packet_LEN_IP])
    return IP_Header

def Get_TCP_Header(packet):
    TCP_Header = struct.unpack('!H H 2H 2H B B H H H',packet[:Packet_LEN_TCP])
    return TCP_Header

def Get_UDP_Header(packet):
    UDP_Header = struct.unpack('!H H H H',packet[:Packet_LEN_UDP])
    return UDP_Header

def Get_ARP_Header(packet):
    ARP_Header = struct.unpack('!H H B B H 6s 4s 6s 4s',packet[:Packet_LEN_ARP])
    return ARP_Header

def Get_ICMP_Header(packet):
    ICMP_Header = struct.unpack('!B B H',packet[:Packet_LEN_ICMP])
    return ICMP_Header
```

sniffer.py

Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		
c	char	string of length 1	1	
b	signed char	integer	1	(3)
B	unsigned char	integer	1	(3)
?	_Bool	bool	1	(1)
h	short	integer	2	(3)
H	unsigned short	integer	2	(3)
i	int	integer	4	(3)
I	unsigned int	integer	4	(3)
l	long	integer	4	(3)
L	unsigned long	integer	4	(3)
q	long long	integer	8	(2), (3)
Q	unsigned long long	integer	8	(2), (3)
f	float	float	4	(4)
d	double	float	8	(4)
s	char[]	string		
p	char[]	string		
P	void *	integer		(5), (3)

```
30  def Get_Eth_Header(packet):
31      L3_Type = 'Undefined'
32      dst_MAC, src_MAC, L3_Type = struct.unpack('! 6s 6s H',packet)
33      return dst_MAC,src_MAC,L3_Type
34
35  def Get_IP_Header(packet):
36      IP_Header = struct.unpack('!B B H H H B B H 4s 4s',packet[:Packet_LEN_IP])
37      return IP_Header
38
39  def Get_TCP_Header(packet):
40      TCP_Header = struct.unpack('!H H 2H 2H B B H H H',packet[:Packet_LEN_TCP])
41      return TCP_Header
42
43  def Get_UDP_Header(packet):
44      UDP_Header = struct.unpack('!H H H H',packet[:Packet_LEN_UDP])
45      return UDP_Header
46
47  def Get_ARP_Header(packet):
48      ARP_Header = struct.unpack('!H H B B H 6s 4s 6s 4s',packet[:Packet_LEN_ARP])
49      return ARP_Header
50
51  def Get_ICMP_Header(packet):
52      ICMP_Header = struct.unpack('!B B H',packet[:Packet_LEN_ICMP])
53      return ICMP_Header
54
```

sniffing.py

```python
from sniffer import *

socket = socket.socket(socket.AF_PACKET,socket.SOCK_RAW,socket.htons(3))

while True:
    packet, addr = socket.recvfrom(65535)
    dst_MAC, src_MAC, L3_Type = Get_Eth_Header(packet[:Packet_LEN_ETH])
    dst_MAC = Get_MAC_Addr(dst_MAC)
    src_MAC = Get_MAC_Addr(src_MAC)

    if L3_Type == IP_Type:
        Packet_LEN_L3 = 0
        Packet_LEN_L4 = 0
        IP_Header = Get_IP_Header(packet[14:])
        IP_Port = IP_Header[6]
        IP_Header_Length = (IP_Header[0]&15)*4
        IP_total_Length = IP_Header[2]
        src_IP = Get_IP_Addr(IP_Header[8])
        dst_IP = Get_IP_Addr(IP_Header[9])
        Packet_LEN_L3 = Packet_LEN_IP
        src_Port = 0
        dst_Port = 0

        if protocols[IP_Port] == 'ICMP':
            ICMP_Header_Start = IP_Header_Length + Packet_LEN_ETH
            ICMP_Header = Get_ICMP_Header(packet[ICMP_Header_Start:])
            print('Layer 3: ICMP')
            print('ICMP Type:',ICMP_Header[0], '->', icmp_type[ICMP_Header[0]])
            print('Source MAC Address:', src_MAC)
            print('Destination MAC Address:',dst_MAC)
            print('Source IP Address:', src_IP)
            print('Destination IP Address:', dst_IP)
            Packet_LEN_L4 = Packet_LEN_ICMP
            Data_Start = Packet_LEN_ETH + IP_Header_Length + Packet_LEN_L4
            print('Data:',packet[Data_Start:])
            print()

        else:
            if protocols[IP_Port] == 'TCP':
                TCP_Header_Start = IP_Header_Length + Packet_LEN_ETH
                TCP_Header = Get_TCP_Header(packet[TCP_Header_Start:])
                src_Port = TCP_Header[0]
                dst_Port = TCP_Header[1]
                Packet_LEN_L4 = Packet_LEN_TCP

            elif protocols[IP_Port] == 'UDP':
                UDP_Header_Start = IP_Header_Length + Packet_LEN_ETH
                UDP_Header = Get_UDP_Header(packet[UDP_Header_Start:])
                src_Port = UDP_Header[0]
                dst_Port = UDP_Header[1]
                Packet_LEN_L4 = Packet_LEN_UDP
```

⋮

```python
            print('Layer 3: IP')
            print('Source MAC Address:', src_MAC)
            print('Destination MAC Address:',dst_MAC)
            print('Source IP Address:', src_IP)
            print('Destination IP Address:', dst_IP)
            print('Layer 4:', protocols[IP_Port])

            if src_Port in protocols:
                print('Source IP Port:', protocols[src_Port], src_Port)
            else:
                print('Source IP Port: Undefined', src_Port)
            if dst_Port in protocols:
                print('Destination Port:', protocols[dst_Port], dst_Port)
            else:
                print('Destination Port: Undefined',  dst_Port)

            Data_Start = Packet_LEN_ETH + IP_Header_Length + Packet_LEN_L4
            print('Data:',packet[Data_Start:])
            print()

    elif L3_Type == ARP_Type:
        ARP_Header = Get_ARP_Header(packet[Packet_LEN_ETH:])
        print('Layer 3: ARP')
        print('Operation:', ARP_Header[4], '->', arp_op[ARP_Header[4]])
        print('Source MAC Address:', Get_MAC_Addr(ARP_Header[5]))
        print('Source IP Address:', Get_IP_Addr(ARP_Header[6]))
        print('Target MAC Address:', Get_MAC_Addr(ARP_Header[7]))
        print('Target IP Address:', Get_IP_Addr(ARP_Header[8]))
        print()
        #Packet_LEN_L3 = Packet_LEN_ARP
```

Python3 sniffing.py

```
root@kali:~/Study/Network/Packet_Sniffing/Packet_Sniffer2# python3 sniffing.py
Layer 3: IP
Source MAC Address: 00:50:56:C0:00:08
Destination MAC Address: 01:00:5E:00:00:FB
Source IP Address: 192.168.10.1
Destination IP Address: 224.0.0.251
Layer 4: UDP
Source IP Port: Undefined 5353
Destination Port: Undefined 5353
Data: b'\x00\x00\x00\x00\x00\x03\x00\x00\x00\x00\x00\x00\r_apple-mobdev\x04_tcp\x05local\x00\x0
0\x0c\x00\x01\x084714054a\x04_sub\x0e_apple-mobdev2\xc0\x1a\x00\x0c\x00\x01\x0c_sleep-proxy\x04
_udp\xc0\x1f\x00\x0c\x00\x01'

Layer 3: ICMP
ICMP Type: 8 -> Echo Request
Source MAC Address: 00:0C:29:2E:3D:01
Destination MAC Address: 00:50:56:F9:1C:E0
Source IP Address: 192.168.10.128
Destination IP Address: 8.8.8.8
Data: b'\x194\x00\x01\x1c\xe3\xe3]\x00\x00\x00\x00\x8eo\x07\x00\x00\x00\x00\x00\x10\x11\x12\x13
\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

Layer 3: ICMP
ICMP Type: 0 -> Echo Reply
Source MAC Address: 00:50:56:F9:1C:E0
Destination MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 8.8.8.8
Destination IP Address: 192.168.10.128
Data: b'\x194\x00\x01\x1c\xe3\xe3]\x00\x00\x00\x00\x8eo\x07\x00\x00\x00\x00\x00\x10\x11\x12\x13
\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

Layer 3: ARP
Operation: 1 -> ARP Request
Source MAC Address: 00:0C:29:2E:3D:01
Source IP Address: 192.168.10.128
Target MAC Address: 00:00:00:00:00:00
Target IP Address: 192.168.10.2
```

QnA