# Packet Sniffing with RAW Socket

유명성

#### 1.1 Packet



- ❖ Packet은 네트워크에서 한 번에 전송되는 데이터의 전송 단위이다.
- ❖ 데이터를 쪼개서 전송하는 이유는 전송 매체를 더 효율적으로 사용하기 위해서이다.
- ❖ 100MByte의 데이터를 한 번에 전송하면 오류가 발생할 확률이 크고, 오류 발생 시 100MByte를 처음부터 다시 받아야한다.
- ❖ Ethernet과 같이 전송 매체를 공유하는 환경에서 이처럼 매체를 오랜 시간 독점해서 사용하는 것은 전체 네트워크의 성능에 악영향을 끼칠 수 있다.
- ❖ 프로토콜별 데이터 전송 단위
  - L2:Frame
  - L3: Packet
  - L4 : Segment(TCP), Message(UDP)

#### 1.2 Packet Sniffing

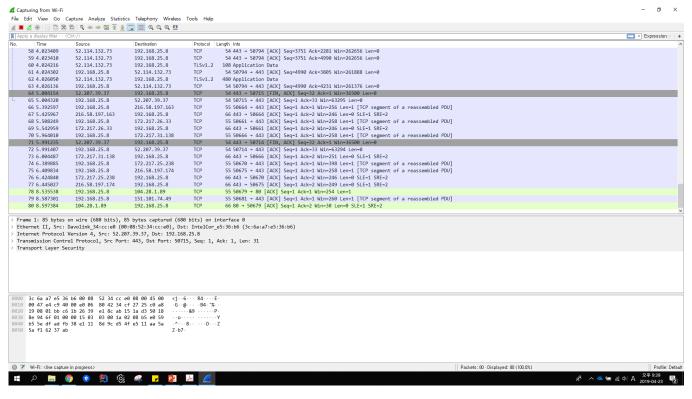
## Packet Sniffing

- ❖ 패킷 스니핑은 패킷 캡쳐, 패킷 분석으로도 불리며 네트워크 상에서 발생하는 일을 이해하기 위해 네트워크를 통해 전달되는 패킷을 수집하고 분석하는 행위이다.
- ❖ 주로 대역폭 분석 등 네트워크 상태를 분석을 위해 사용되며, 새로운 프로토콜 개발에도 쓰인다.
- ❖ 보안상 악의적인 공격을 탐지하고 공격자를 추적하기 위해서도 사용된다.
- ❖ 대표적인 패킷 스니핑 툴에 tcpdump, wireshark, kismet 등이 있다.

#### 1.3 Wireshark

- Wireshark
  - ❖ 오픈소스 GUI 패킷 스니핑 프로그램
  - ❖ 내부적으로 Libpcap을 사용하며 GUI는 Qt로 작성되어 있어 Windows, Linux, MAC OS 등 다양한 운영체제에서

동작할 수 있다.(크로스 플랫폼)

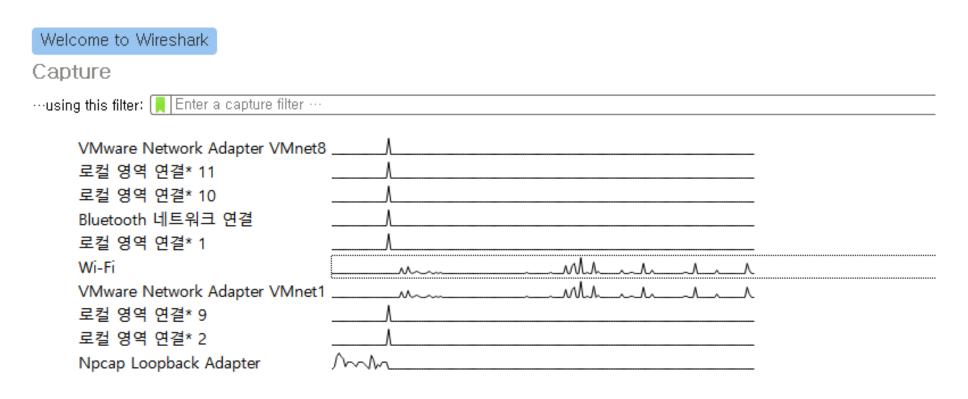


#### 1.4 pcap



- ❖ OS에서 패킷을 캡처할 수 있게 해주는 C 라이브러리로 tcpdump.org에서 제작하였다.
- ❖ Libpcap(Linux), Winpcap(Windows), Npcap(Windows) 등 다양한 버전이 존재한다.
- ❖ 다양한 언어에서 pcap을 wrapping한 라이브러리를 지원한다.
  - C++: Libtins
  - Python: python-libpcap, Pcapy, WinPcapy, scapy
  - Java: jpcap, jNetPcap

#### 1.5 packet capture



원하는 인터페이스를 선택하여 패킷을 캡처할 수 있다.

#### 1.5 packet capture

캡처된 패킷 목록

패킷 프로토콜 및 필드 정보

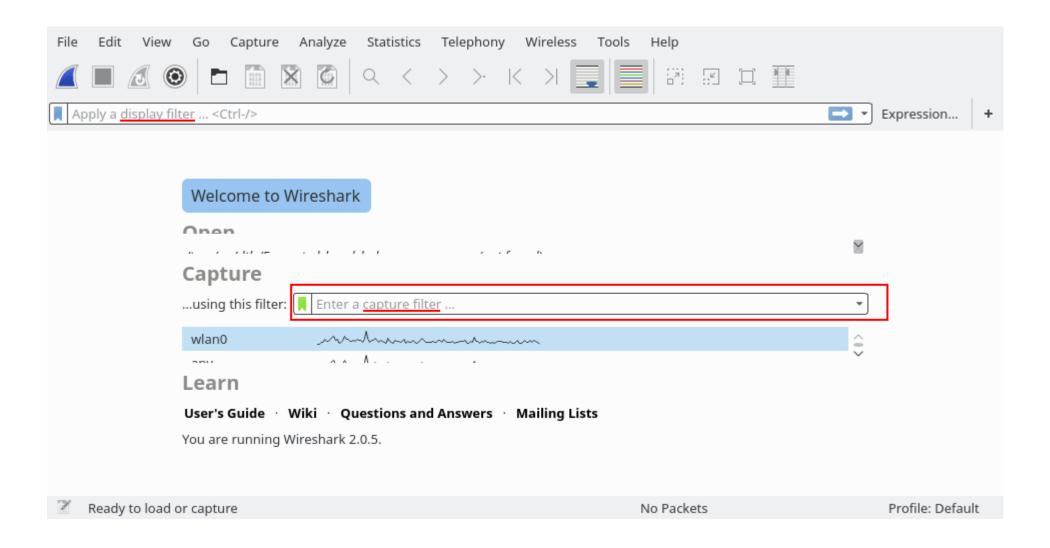
Raw Data

```
Apply a display filter ··· <Ctrl-/>
                                           Destination
    753 9.741544
                      192,168,25,8
                                          13.107.136.9
                                                               TCP
                                                                         1494 53981 → 443 [ACK] Seq=108580 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
    754 9.741544
                      192.168.25.8
                                          13.107.136.9
                                                                        1494 53981 → 443 [ACK] Seq=110020 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
    755 9.741545
                      192.168.25.8
                                          13.107.136.9
                                                                        1494 53981 → 443 [ACK] Seq=111460 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
                      13.107.136.9
    756 9.743186
                                          192.168.25.8
                                                               TCP
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=50980 Win=525568 Len=0
    757 9.743187
                      13.107.136.9
                                          192.168.25.8
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=52420 Win=524032 Len=0
    758 9.743187
                      13.107.136.9
                                          192.168.25.8
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=61060 Win=525568 Len=0
    759 9.743232
                      192.168.25.8
                                          13.107.136.9
                                                                        1494 53981 → 443 [ACK] Seq=112900 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
    760 9.743233
                      192.168.25.8
                                          13.107.136.9
                                                                        1494 53981 → 443 [ACK] Seq=114340 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
    761 9.743235
                      192.168.25.8
                                          13.107.136.9
                                                               TLSv1.2 1494 Application Data [TCP segment of a reassembled PDU]
    762 9.743235
                      192,168,25,8
                                          13.107.136.9
                                                                        1494 53981 → 443 [ACK] Seq=117220 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
                      192.168.25.8
                                                               TCP
                                                                         1494 53981 → 443 [ACK] Seq=118660 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU]
    763 9.743235
                                          13.107.136.9
    764 9.743236
                     192.168.25.8
                                          13.107.136.9
                                                               TLSv1.2 961 Application Data
    765 9.745347
                      13.107.136.9
                                          192.168.25.8
                                                                          54 443 → 53981 [ACK] Seg=5987 Ack=62500 Win=524032 Len=0
                                                                          54 443 → 53981 [ACK] Seg=5987 Ack=65380 Win=525568 Len=0
    766 9.745347
                      13.107.136.9
                                          192.168.25.8
                                                               TCP
    767 9.745347
                      13.107.136.9
                                          192.168.25.8
                                                               TCP
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=66820 Win=524032 Len=0
    768 9.745347
                      13.107.136.9
                                          192,168,25,8
                                                                          54 443 → 53981 [ACK] Seg=5987 Ack=68260 Win=525568 Len=0
    769 9.745348
                      13.107.136.9
                                          192,168,25,8
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=69700 Win=524032 Len=0
                     13.107.136.9
                                          192.168.25.8
                                                               TCP
    770 9.745348
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=71140 Win=525568 Len=0
    771 9.745348
                      13.107.136.9
                                          192.168.25.8
                                                                          54 443 → 53981 [ACK] Sea=5987 Ack=74020 Win=525568 Len=0
    772 9.745348
                     13.107.136.9
                                          192.168.25.8
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=75460 Win=524032 Len=0
    773 9.745349
                      13.107.136.9
                                          192.168.25.8
                                                               TCP
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=79780 Win=525568 Len=0
    774 9.746043
                      13.107.136.9
                                          192.168.25.8
                                                               TCP
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=81220 Win=524032 Len=0
                                                                          54 443 → 53981 [ACK] Seq=5987 Ack=82660 Win=525568 Len=0
    775 9.746043
                      13.107.136.9
                                          192.168.25.8
```

- Frame 769: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
- Ethernet II, Src: Davolink 34:cc:e0 (00:08:52:34:cc:e0), Dst: IntelCor e5:36:b6 (3c:6a:a7:e5:36:b6)
- Internet Protocol Version 4, Src: 13.107.136.9, Dst: 192.168.25.8
- Transmission Control Protocol, Src Port: 443, Dst Port: 53981, Seq: 5987, Ack: 69700, Len: 0

```
3c 6a a7 e5 36 b6 00 08 52 34 cc e0 08 00 45 00
0010 00 28 77 10 40 00 78 06 1c 9b 0d 6b 88 09 c0 a8
0020 19 08 01 bb d2 dd 25 a2 01 de d4 d2 de e4 50 10
0030 07 ff 88 e0 00 00
```

#### 1.5 packet capture



#### 1.5 packet capture

#### Examples

Capture only traffic to or from IP address 172.18.5.4:

host 172.18.5.4

Capture traffic to or from a range of IP addresses:

net 192.168.0.0/24

or

net 192.168.0.0 mask 255.255.255.0

Capture traffic from a range of IP addresses:

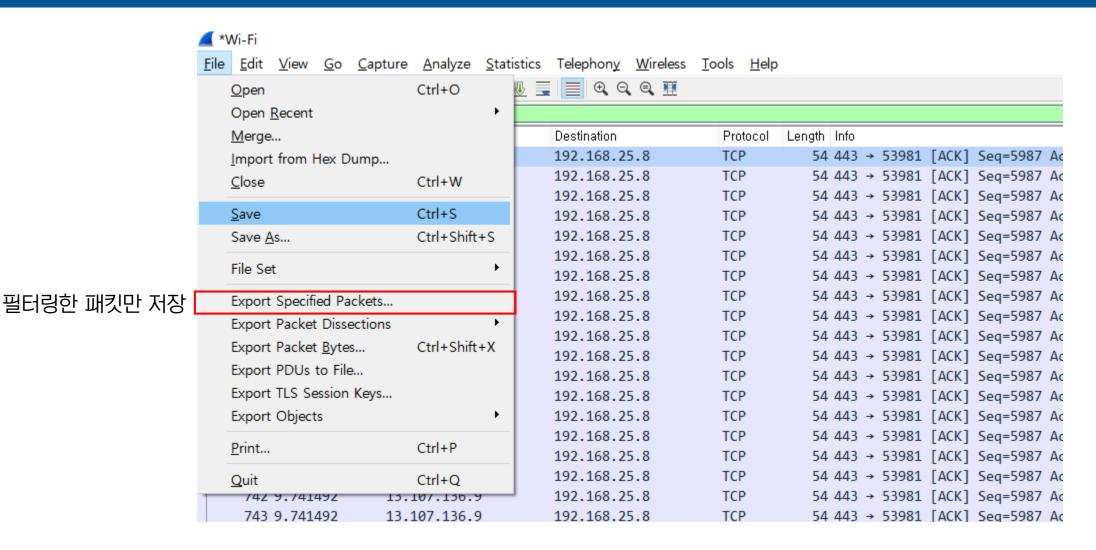
src net 192.168.0.0/24

or

src net 192.168.0.0 mask 255.255.255.0

Libpcap의 pcap\_filter를 이용해 원하는 패킷만 필터링할 수 있다. <a href="http://www.tcpdump.org/manpages/pcap-filter.7.html">http://www.tcpdump.org/manpages/pcap-filter.7.html</a>

#### 1.5 packet capture



### 1.5 packet capture

Second   S	Time	Source	Destination	Protocol	Length Info				
13.107.136.9   192.168.25.8   TCP   54 443 + 53981   ACK   Seq=5987   Ack=17860   Min=525568   Len=0					-	[ACK] Sen=5987	Δck=16420 W	in=524032 L	en=0
Expand Subtrees   .25.8   TCP   54 443 + 53981   ACK   Seq=5987   Ack=19300   Win=524032   Len=0									
Collapse Subtrees   .25.8   TCP   .54 443 + .53981   TCK   .54 53981   TCK   .56 53987   TCK -42340   TCK   .52 5588   TCP   .54 443 + .53981   TCK   .56 53987   TCK -43380   TCK   .56 5389   TCK   .54 543 + .53981   TCK   .56 5389   TCK   .54 543 + .53981   TCK   .56 5389   TCK   .54 543 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK   .55 568   TCP   .54 443 + .53981   TCK   .56 5389   TCK									
697 9.7353		·							
Collapse All		· ·		TCP					
718 9.7370 719 9.7370 719 9.7370 720 9.7370 721 9.7370 722 9.7370 723 9.7370 724 9.7370 724 9.7370 725 9.7370 726 Prepare a Filter  1 Not Selected 727 9.7370 728 9.7370 729 9.7370 729 9.7370 720 9.7370 720 9.7370 721 9.7370 722 9.7370 723 9.7370 724 9.7370 725 9.7370 726 9.7370 727 9.7370 728 9.7370 729 9.7370 729 9.7370 720 9.7370 720 9.7370 721 9.7370 722 9.7370 723 9.7370 724 9.7370 725 9.7370 726 9.7370 727 9.7370 727 9.7370 728 9.7370 729 9.7370 729 9.7370 729 9.7370 720 9.7370 720 9.7370 721 9.7370 722 9.7370 723 9.7370 724 9.7370 725 9.7370 726 9.7370 727 9.7370 728 9.7370 729 9.7370 729 9.7370 729 9.7370 729 9.7370 729 9.7370 720 9.7370 720 9.7370 721 9.7370 722 9.7370 723 9.7370 724 9.7370 725 9.7370 726 9.7370 727 9.7310 728 9.7370 729 9.7310 729 9.7370 729 9.7310 720 9.7310 720 9.7310 721 9.7320 722 9.7370 723 9.7370 724 9.7370 725 9.7370 726 9.7310 727 9.7310 728 9.7310 729 9.7310 729 9.7310 720 9.7310 720 9.7310 721 9.7320 722 9.7370 723 9.7370 724 9.7370 725 9.7320 726 9.7320 727 9.7320 728 9.7370 729 9.7310 729 9.7310 720 9.7310 720 9.7310 721 9.7320 722 9.7370 723 9.7370 724 9.7370 725 9.7320 726 9.7320 727 9.7320 727 9.7320 728 9.7320 729 9.7	698 9.7353	Expand All	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=27940 W	in=525568 Le	en=0
719 9.7370 729 9.7370 Apply as Filter Apply as	699 9.7353	Collapse All	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=30820 W	in=525568 Le	en=0
719 9.7376	718 9.7370	Amala as Calaman	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=32260 W	in=524032 Le	en=0
721 9.7370 Prepare a Filter Prepare a Filter Prepare a Filter  Not Selected  54 443 + 53981 [ACK] Seq=5987 Ack=36580 Win=525568 Len=0  54 443 + 53981 [ACK] Seq=5987 Ack=38020 Win=524032 Len=0  55 443 + 53981 [ACK] Seq=5987 Ack=38020 Win=524032 Len=0  56 443 + 53981 [ACK] Seq=5987 Ack=39460 Win=525568 Len=0  57 443 + 53981 [ACK] Seq=5987 Ack=39460 Win=525568 Len=0  58 443 + 53981 [ACK] Seq=5987 Ack=39460 Win=525568 Len=0  59 443 + 53981 [ACK] Seq=5987 Ack=3980 Win=525568 Len=0  59 443 + 53981 [ACK] Seq=5987 Ack=3980 Win=525568 Len=0  59 443 + 53981 [ACK] Seq=5987 Ack=3980 Win=525568 Len=0  59 443 + 53981 [ACK] Seq=5987 Ack=3980 Win=525568 Len=0  5	719 9.7370	Apply as Column	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=33700 W	in=525568 Le	en=0
722 9.7370 723 9.7370 724 9.7370 724 9.7370 739 9.7414 740 9.7414 741 9.7414 742 9.7414 743 9.7414 743 9.7414 745 9.7431 756 9.7431 757 9.7431 758 9.7431 759 9.7433 759 9.7431 759 9.7433 759 9.7431 759 9.7433 759 9.7434 759 9.7433 759 9.7433 759 9.7434 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 759 9.7444 750 9.7444 750 9.7444 750 9.7444 750 9.7444 750 9.7444 750 9.7444	720 9.7370	Apply as Filter	Selected		54 443 → 53981	[ACK] Seq=5987	Ack=35140 W	in=524032 Le	en=0
Conversation Filter   Colorize with Filter	721 9.7370	Prepare a Filter	▶ Not Selec	ted	54 443 → 53981	[ACK] Seq=5987	Ack=36580 W	in=525568 Le	en=0
723 9.7370 724 9.7370 739 9.7414 740 9.7414 741 9.7414 742 9.7414 743 9.7414 744 9.7414 745 9.7431 756 9.7431 757 9.7431 757 9.7431 758 9.7431 758 9.7431 759 9.7431 759 9.7431 756 9.7431 757 9.7431 758 9.7431 759 9.7433 766 9.7435 766 9.7435 766 9.7453 767 9.7454 768 9.7453 769 9.7453 760 9.7454 760 9.7453 760 9.7454	722 9.7370	•	and Sel	ected	54 443 → 53981	[ACK] Seq=5987	Ack=38020 W	in=524032 Le	en=0
724 9.7370 739 9.7414 740 9.7414 740 9.7414 741 9.7414 742 9.7414 743 9.7414 756 9.7431 756 9.7431 757 9.7431 758 9.7431 758 9.7431 759 9.7431 759 9.7431 759 9.7431 750 9.7433 760 9.7433 760 9.7433 760 9.7453	723 9.7370				54 443 → 53981	[ACK] Seq=5987	Ack=39460 W	in=525568 Le	en=0
749 9.7414 740 9.7414 741 9.7414 742 9.7414 743 9.7414 756 9.7431 758 9.7431 758 9.7431 765 9.7433 766 9.7453 766 9.7453 766 9.7453 Decode As  0101 Differenti  Total Lengure 70  Copy or not Selectedor not Au43 → 53981 [ACK] Seq=5987 Ack=42370 Win=52568 Len=0or not Selectedor no	724 9.7370				54 443 → 53981	[ACK] Seq=5987	Ack=40900 W	in=524032 Le	en=0
741 9.7414 742 9.7414 743 9.7414 745 9.7414 756 9.7431 758 9.7431 765 9.7431 765 9.7433 766 9.7433 766 9.7433 766 9.7433 766 9.7433 766 9.7433 766 9.7433 766 9.7433 767 9.7433 768 9.7433 769 9.7433 769 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7433 760 9.7453	739 9.7414	Follow			54 443 → 53981	[ACK] Seq=5987	Ack=42340 W	in=525568 Le	en=0
742 9.7414 743 9.7414 756 9.7431 757 9.7431 758 9.7431 758 9.7431 759 9.7431 759 9.7431 750 9.7431 750 9.7431 750 9.7431 750 9.7431 750 9.7431 751 9.7431 752 9.7431 753 9.7431 754 9.7431 755 9.7431 755 9.7431 756 9.7433 757 9.7433 758 9.7433 759 9.7433 759 9.7433 750 9.7433	740 9.7414	Сору	P	Selected	54 443 → 53981	[ACK] Seq=5987	Ack=43780 W	in=524032 Le	en=0
742 9.7414 743 9.7414 756 9.7431 757 9.7431 758 9.7431 765 9.7431 765 9.7432 766 9.7433 766 9.7453 767 9.7453 768 9.7453 768 9.7453 769 9.7453 769 9.7453 760 9.7453	741 9.7414	Show Packet Bytes			54 443 → 53981	[ACK] Seq=5987	Ack=45220 W	in=525568 Le	en=0
743 9.7414 756 9.7431 757 9.7431 758 9.7431 765 9.7433 766 9.7453 767 9.7453 768 9.7453 769 9.7453 760 9.7453	742 9.7414	*	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=48100 W	in=525568 Le	en=0
757 9.7431 758 9.7431 765 9.7453 766 9.7453 766 9.7453 769 0.7453 760 1011 Total Lengun. → □  Filter Field Reference  125.8  125.8  127  125.8  127  125.8  128  125.8  129  125.8  126  125.8  126  125.8  126  125.8  126  125.8  126  125.8  126  125.8  126  125.8  126  125.8  126  126  126  126  126  126  126  12	743 9.7414	Export Facket Bytes	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=49540 W	in=524032 Le	en=0
758 9.7431 765 9.7453 766 9.7453 Decode As 0101 On total Lengun. 40  Protocol Preferences Decode As  25.8  TCP Decode As	756 9.7431	Wiki Protocol Page	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=50980 W	in=525568 Le	en=0
765 9.7453  766 9.7453  Decode As  0101  Go to Linked Packet  Differenti  Show Linked Packet in New Window  Total Lengur. →  Protocol Preferences  1.25.8  TCP  54 443 → 53981 [ACK] Seq=5987 Ack=62500 Win=524032 Len=0  25.8  TCP  54 443 → 53981 [ACK] Seq=5987 Ack=65380 Win=52568 Len=0  ECN: Not-ECT)  ECN: Not-ECT)	757 9.7431	Filter Field Reference		TCP	54 443 → 53981	[ACK] Seq=5987	Ack=52420 W	in=524032 Le	≥n=0
765 9.7453  766 9.7453  Decode As  0101  Differenti  Show Linked Packet in New Window  Total Length. → □	758 9.7431	Protocol Preferences	•	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=61060 W	in=525568 Le	≥n=0
0101 Go to Linked Packet  Differenti Show Linked Packet in New Window  Total Length. 10	765 9.7453	Trotocol Frontieres	.25.8	TCP	54 443 → 53981	[ACK] Seq=5987	Ack=62500 W	in=524032 Le	en=0
Differenti Show Linked Packet in New Window ECN: Not-ECT)  Total Lengua. 10		Decode As	25. 8	TCP	54 443 → 53981	[ACK] Sen=5987	Δck=65380 W	in=525568 le	n=0
Total Lengun. +v	0101	Go to Linked Packet							
	Differenti	Show Linked Packet in New W	indow , ECN: Not-	ECT)					
Identification: 0x7701 (30465)	Total Lengum.	40							
	Identification	n: 0x7701 (30465)							

특정 필드 선택 후 필더 지정 가능

> Source: 13.107.136.9 Destination: 192.168.25.8

#### 1.5 packet capture

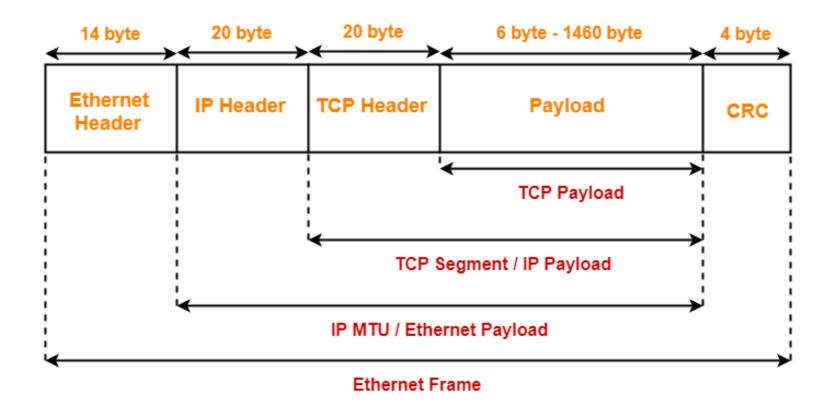
✓ \*Wi-Fi File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help 📙 🔚 🔀 🖺 | ९ 👄 👄 堅 🕜 🌡 🕎 🗐 📵 ९ ९ ९ 🕮 tcp,stream eq 7 No. Time Source Destination Protocol Length Info 700 9.735393 192.168.25.8 13.107.136.9 TCP 1494 53981 → 443 [ACK] Seq=49540 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] 701 9.735394 192.168.25.8 13.107.136.9 TLSv1.2 1494 Application Data [TCP segment of a reassembled PDU] 702 9.735395 192.168.25.8 13.107.136.9 1494 53981 → 443 [ACK] Seg=52420 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] TCP 703 9.735396 192.168.25.8 13.107.136.9 1494 53981 → 443 [ACK] Seq=53860 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] 704 9.735396 192.168.25.8 13.107.136.9 TCP 1494 53981 → 443 [ACK] Seq=55300 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] TCP 705 9.735396 1494 53981 → 443 [ACK] Seq=56740 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Mark/Unmark Packet 706 9.735396 192 TCP 1494 53981 → 443 [ACK] Seq=58180 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Ignore/Unignore Packet 707 9.735397 192 TCP 1494 53981 → 443 [ACK] Seq=59620 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Set/Unset Time Reference 1494 53981 → 443 [ACK] Seq=61060 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] 708 9.735397 192 TCP 709 9.735398 192 Time Shift... TCP 1494 53981 → 443 [ACK] Seq=62500 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] 710 9.735398 192 TCP 1494 53981 → 443 [ACK] Seq=63940 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Packet Comment... 711 9.735398 192 TCP 1494 53981 → 443 [ACK] Seq=65380 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Edit Resolved Name 712 9.735399 192 1494 Application Data [TCP segment of a reassembled PDU] 713 9.735399 192 1494 53981 → 443 [ACK] Seq=68260 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] TCP Apply as Filter 714 9.735399 192 TCP 1494 53981 → 443 [ACK] Seq=69700 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Prepare a Filter 715 9.735400 192 TCP 1494 53981 → 443 [ACK] Seq=71140 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Conversation Filter 192 716 9.735400 TCP 1494 53981 → 443 [ACK] Seg=72580 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] Colorize Conversation 717 9.735400 192 TCP 1494 53981 → 443 [ACK] Seq=74020 Ack=5987 Win=65792 Len=1440 [TCP segment of a reassembled PDU] 718 9.737087 13. SCTP TCP 54 443 → 53981 [ACK] Seq=5987 Ack=32260 Win=524032 Len=0 719 9.737087 13. 54 443 → 53981 [ACK] Seq=5987 Ack=33700 Win=525568 Len=0 Follow 720 9.737087 13. 54 443 → 53981 [ACK] Seg=5987 Ack=35140 Win=524032 Len=0 **UDP Stream** Copy 13. 721 9.737087 54 443 → 53981 [ACK] Seg=5987 Ack=36580 Win=525568 Len=0 TLS Stream 722 9.737088 13. 54 443 → 53981 [ACK] Seq=5987 Ack=38020 Win=524032 Len=0 Protocol Preferences HTTP Stream 54 443 → 53981 [ΔCK] Sea=5987 Δck=39460 Win=525568 Len=θ 723 9 737088 Decode As... > Frame 705: 1494 bytes o captured (11952 bits) on interface 0 Show Packet in New Window > Ethernet II, Src: Intel ), Dst: Davolink 34:cc:e0 (00:08:52:34:cc:e0) Internet Protocol Version 4, Src: 192.168.25.8, Dst: 13.107.136.9 0100 .... = Version: 4 .... 0101 = Header Length: 20 bytes (5) > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 1480 Identification: 0x4450 (17488) > Flags: 0x4000, Don't fragment Time to live: 128 Protocol: TCP (6)

특정 TCP/UDP 스트림만 출력할 수 도 있다.

0000 00 08 52 34 cc e0 3c 6a a7 e5 36 b6 08 00 45 00

### 1.6 Ethernet sniffing

TCP/IP Packet 구조



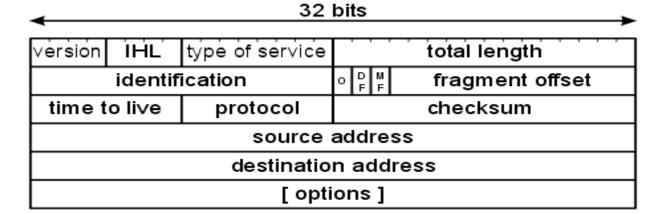
#### 1.6 Ethernet sniffing

```
> Frame 10: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
Ethernet II, Src: IntelCor e5:36:b6 (3c:6a:a7:e5:36:b6), Dst: Davolink 34:cc:e0 (00:08:52:34:cc:e0)
  > Destination: Davolink_34:cc:e0 (00:08:52:34:cc:e0)
  > Source: IntelCor_e5:36:b6 (3c:6a:a7:e5:36:b6)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.25.8, Dst: 108.177.125.188
> Transmission Control Protocol, Src Port: 58449, Dst Port: 5228, Seq: 1, Ack: 1, Len: 1
> Data (1 byte)
      00 08 52 34 cc e0 3c 6a a7 e5 36 b6 08 00 45 00
                                                         ..R4..<j ..6...E.
0000
0010 00 29 05 bf 40 00 80 06 30 f2 c0 a8 19 08 6c b1
                                                         .)..@... 0....1.
                                                         }--0-1!- ----a-P-
0020 7d bc e4 51 14 6c 21 dd 93 08 a5 0b 71 aa 50 10
                                                          · - &Z · - -
0030 01 02 26 5a 00 00 00
                                         Ethernet Header (14 byte)
                                                                         46 to 1500
                                                                                         4 byte
         7 byte
                       1 byte
                                     6 byte
                                                                            byte
                                                   6 byte
                                                               2 byte
                                                                                         Frame
                        Start
                                                                                         Check
                                    Destination
                                                   Source
                                                               Length
                                                                           Data
        Preamble
                       Frame
                                                                                       Sequence
                                     Address
                                                  Address
                      Delimiter
                                                                                         (CRC)
```

#### 1.7 IP sniffing

```
> Frame 11847: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: IntelCor_e5:36:b6 (3c:6a:a7:e5:36:b6), Dst: Davolink 34:cc:e0 (00:08:52:34:cc:e0)
Internet Protocol Version 4, Src: 192.168.25.8, Dst: 13.107.136.9
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x005b (91)
  > Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
0000 00 08 52 34 cc e0 3c 6a a7 e5 36 b6 08 00 45 00
                                                       --R4--<j --6---<u>E</u>
0010
     00 28 00 5b 40 00 80 06 8b 50 c0 a8 19 08 0d 6b
     88 09 ed ec 01 bb 06 ab c0 1e c2 ab 4b 5f 50 10
0020
0030 01 02 7b 31 00 00
                                                         ..{1..
```

#### IP header format



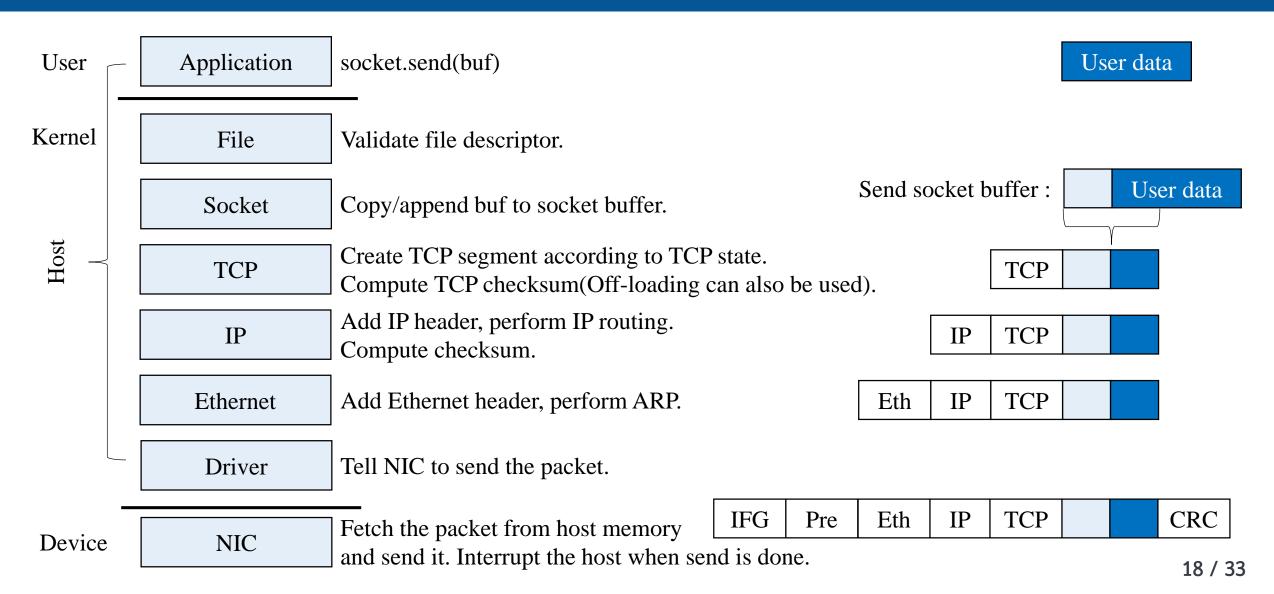
#### 1.8 HTTP sniffing

```
> Frame 475: 529 bytes on wire (4232 bits), 529 bytes captured (4232 bits) on interface 0
> Ethernet II, Src: IntelCor e5:36:b6 (3c:6a:a7:e5:36:b6), Dst: Davolink 34:cc:e0 (00:08:52:34:cc:e0)
> Internet Protocol Version 4, Src: 192.168.25.8, Dst: 175.213.35.39
> Transmission Control Protocol, Src Port: 61332, Dst Port: 80, Seq: 1, Ack: 1, Len: 475

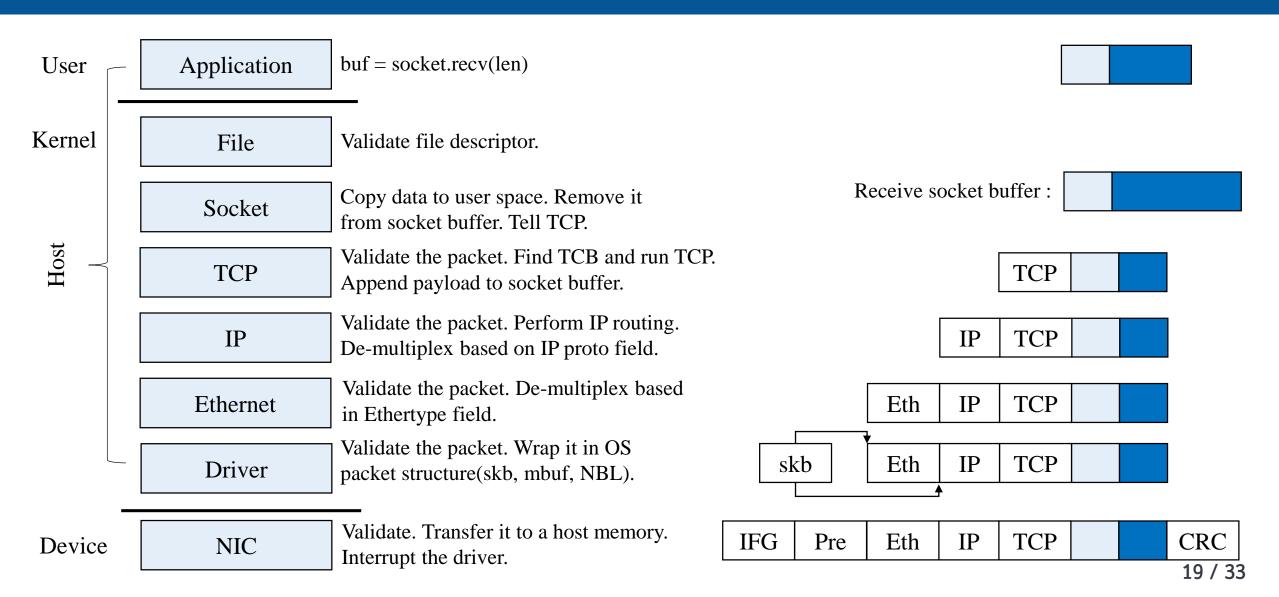
∨ Hypertext Transfer Protocol

  > GET / HTTP/1.1\r\n
    Host: gilgil.net\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    DNT: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36\r\n
0030 01 00 83 0b 00 00 47 45 54 20 2f 20 48 54 54 50
                                                          ·····GE T / HTTP
      2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 67 69 6c 67
                                                          /1.1 \cdot \cdot \cdot Ho st: gilg
0040
      69 6c 2e 6e 65 74 0d 0a 43 6f 6e 6e 65 63 74 69
                                                          il.net∴ Connecti
0050
0060
      6f 6e 3a 20 6b 65 65 70  2d 61 6c 69 76 65 0d 0a
                                                          on: keep -alive.
0070
      55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65
                                                          Upgrade- Insecure
0080
      2d 52 65 71 75 65 73 74   73 3a 20 31 0d 0a 44 4e
                                                          -Request s: 1..DN
0090
      54 3a 20 31 0d 0a 55 73  65 72 2d 41 67 65 6e 74
                                                          T: 1⋅⋅Us er-Agent
      3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 20 28 57
                                                          : Mozill a/5.0 (W
00a0
      69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b 20
                                                          indows N T 10.0;
00b0
      57 69 6e 36 34 3b 20 78 36 34 29 20 41 70 70 6c
                                                          Win64; x 64) Appl
00c0
```

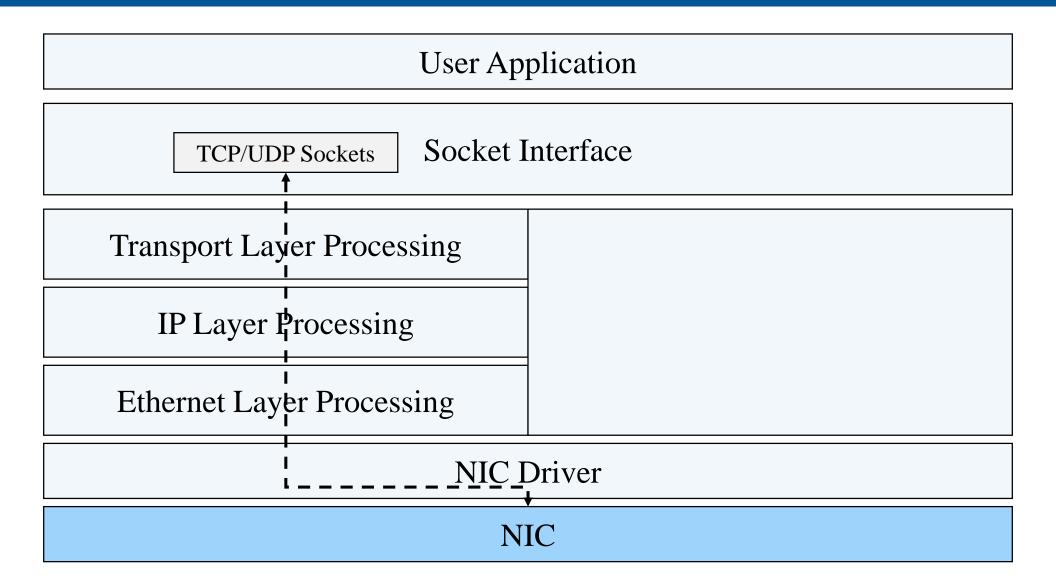
### 2.1 kernel protocol stack(when sending a packet)



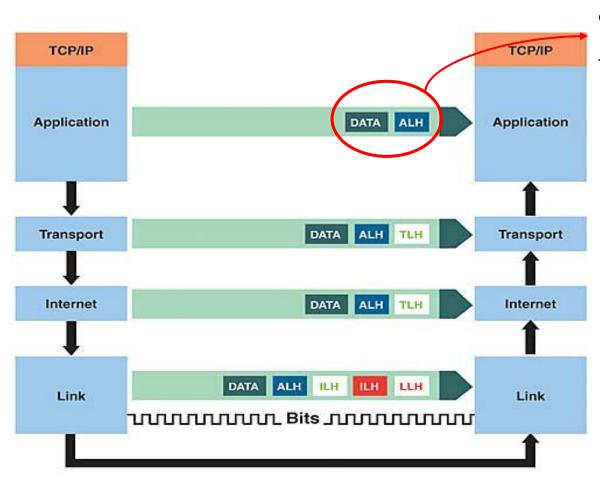
### 2.1 kernel protocol stack(when receiving a packet)



#### 2.2 Socket interface



#### 2.2 Socket interface



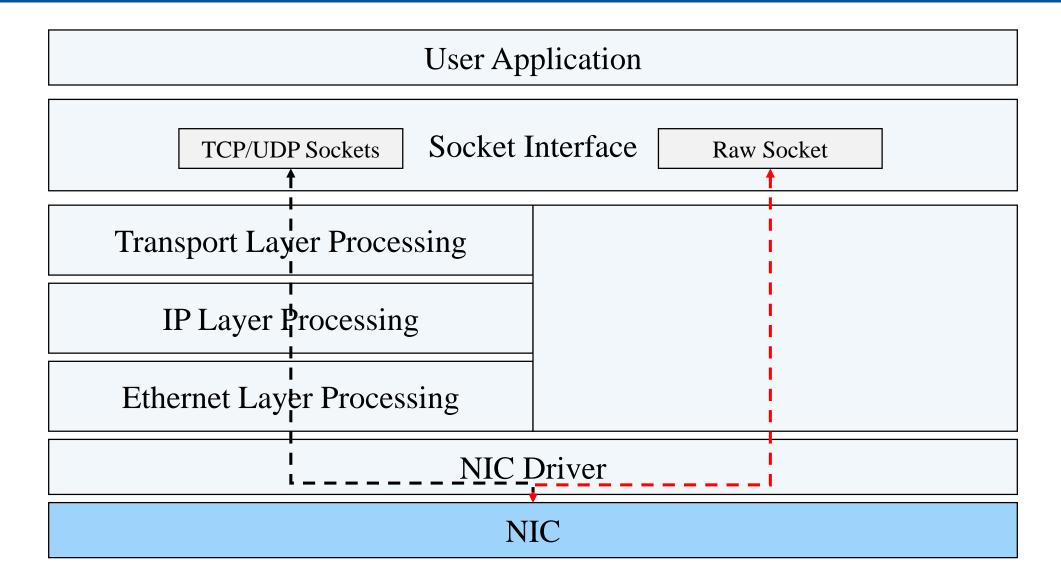
일반 Socket은 응용프로그램에 응용계층 데이터만 전달한다. 모든 프로토콜 헤더는 프로토콜 스택에서 처리하며 제거된다.

#### 2.3 Raw Socket

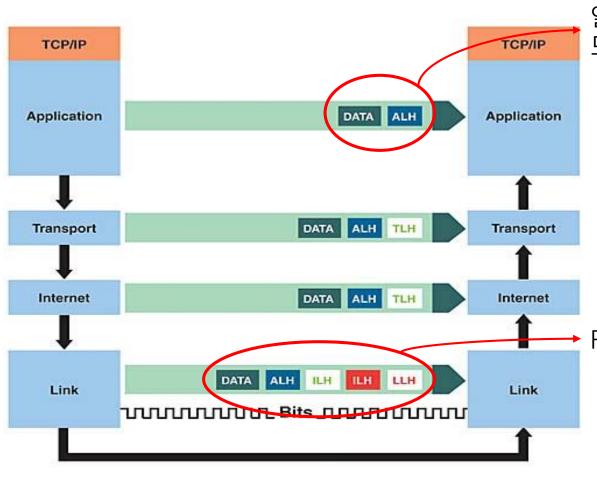


- ❖ raw 소켓은 어느 특정한 프로토콜 용의 전송 계층 포매팅 없이 인터넷 프로토콜 패킷을 직접적으로 주고 받게 해주는 인터넷 소켓이다.
- ❖ 새로운 프로토콜을 설계하여 구현하거나, 네트워크 장비를 만드는 경우와 같이 패킷을 세밀하게 조작할 때 사용하는 특수한 Socket이다.
- ❖ 커널 수준에서만 다룰 수 있었던 프로토콜 헤더를 직접 다룰 수 있다. 또한 사용자 응용프로그램에서 Raw Socket을 통해 직접 만든 패킷을 전송할 수 있다.
- ❖ 일반적으로 Raw Socket을 사용하기 위해서는 관리자(root) 권한이 필요하다.

#### 2.3 Raw Socket



#### 2.3 Raw Socket



일반 Socket은 응용프로그램에 응용계층 데이터만 전달한다. 모든 프로토콜 헤더는 프로토콜 스택에서 처리하며 제거된다.

Row Socket은 헤더를 모두 포함한 데이터를 전달한다.

### 2.4 Creating a raw socket

```
IPPROTO_ICMP

IPPROTO_UDP

ETH_P_IP

IPPROTO_TCP

ETH_P_ALL
```

socket.socket(\${family}, socket.SOCK\_RAW, \${protocol})

AF\_INET L3 이후의 프로토콜 헤더 포함

AF\_PACKET L2 이후의 프로토콜 헤더 포함

AF\_PACKET은 바인드된 인터페이스가 송/수신하는 모든 패킷을 애플리케이션에 전달

#### 2.4 Creating a raw socket

### Raw Socket in Linux

- ❖ 일반적으로 아래와 같이 Raw Socket을 생성하면 모든 타입의 프레임을 수집할 수 있다.
- ❖ ETH\_P\_ALL은 Python에 선언이 되어 있지 않기 때문에 C 언어 라이브러리의 값을 따로 선언해야 한다.
- ❖ AF\_PACKET으로 생성한 Raw Socket은 특정 NIC에 바인드할 수 있다. 이때 NIC 이름을 사용한다.
- ❖ 바인드할 경우 해당 NIC에서 수신되는 패킷만 가져오며, 바인드하지 않을 경우 모든 NIC에서 수신되는 패킷을 가져온다.
- ❖ 일반적으로 바인딩할 때 포트번호는 0(Default)를 사용한다.

```
## Linux OS
ETH_P_ALL = 0x0003
sniff_sock = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(ETH_P_ALL))
# Optional
sniff_sock.bind("Interface name", 0) # NICO(름을 입력해 해당 NIC로 들어오는 패킷만 리스님
# bind 하지 않으면 모든 NIC를 대상으로 리스님
```

### 2.4 Creating a raw socket

Raw Socket in Linux

```
# AF_INET(인터넷 소켓)을 이용한 RAW_SOCKET in linux with python 3,x
socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP) # TCP Packet
socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_UDP) # UDP Packet
socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP) # ICMP Packet
# AF_PACKET(패킷 소켓)을 이용한 RAW_SOCKET in linux with python 3,x
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.IPPROTO_TCP) # TCP Packet
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.IPPROTO_UDP) # UDP Packet
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.IPPROTO_ICMP) # ICMP Packet
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.IPPROTO_ICMP) # ICMP Packet
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.IPPROTO_ICMP) # IP Packet -> ETH_P_IP
socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.Intohs(0x00003)) # Every Packets -> ETH_P_ALL
```

### 2.4 Creating a raw socket



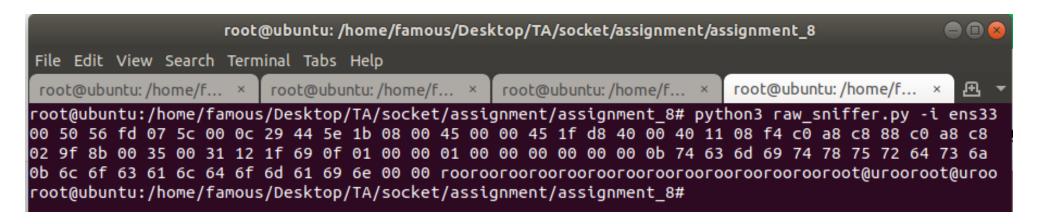
- ❖ 일반적으로 아래와 같이 Raw Socket을 생성하면 모든 IP 패킷을 수집할 수 있다.
- ❖ Windows에선 AF\_PACKET을 사용할 수 없다.
- ❖ AF\_INET만 사용하며 때문에 특정 NIC의 IP 주소에 바인딩해야 한다.

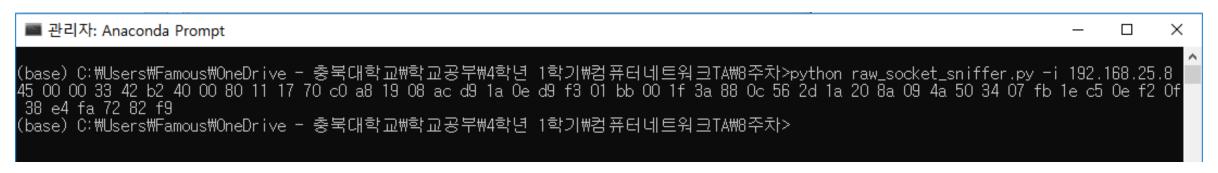
```
## Windows NT
sniff_sock = socket.socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_IP)
sniff_sock.bind(("NIC IP address",0))
sniff_sock.setsockopt(socket.IPPROTO_IP,socket.IP_HDRINCL,1)
sniff_sock.ioctl(socket.SIO_RCVALL,socket.RCVALL_ON)
```

```
def sniffing(nic):
    if os.name == 'nt':
        address familiy = socket.AF INET
        protocol type = socket.IPPROTO IP
    else:
        address familiy = socket.AF PACKET
        protocol_type = socket.ntohs(ETH_P_ALL)
    with socket.socket(address familiy, socket.SOCK RAW, protocol type) as sniffe sock:
        sniffe sock.bind((nic, 0))
        if os.name == 'nt':
            sniffe sock.setsockopt(socket.IPPROTO IP,socket.IP HDRINCL,1)
            sniffe sock.ioctl(socket.SIO RCVALL,socket.RCVALL ON)
        data, = sniffe sock.recvfrom(65535)
        for p in data:
            print("%02x" % p, end=" ")
        if os.name == 'nt':
            sniffe sock.ioctl(socket.SIO RCVALL,socket.RCVALL OFF)
```

ETH P ALL =  $0 \times 00003$ 

#### 2.5 Raw socket example





```
def dumpcode(buf):
    print("%7s"% "offset ", end='')
   for i in range(0, 16):
        print("%02x " % i, end='')
        if not (i%16-7):
            print("- ", end='')
    print("")
    for i in range(0, len(buf)):
        if not i%16:
            print("0x%04x" % i, end= ' ')
        print("%02x" % buf[i], end= ' ')
        if not (i % 16 - 7):
            print("- ", end='')
        if not (i % 16 - 15):
            print(" ")
    print("")
```

#### 2.5 Raw socket example

#### 2.6 Assignment 8

## Assignment #8

- 수업 Github assignment\_8에 있는 raw\_sniffer.py를 사용한 패킷 분석
  - Linux에서 수행할 것
  - raw\_niffer.py로 Assignment#2(문자열 거꾸로 전송)가 실행되면서 서버-클라이언트간 주고받은 첫 번째 TCP 패킷을 캡처해 사진 첨부(문자열은 팀 이름을 전달)
  - 캡처한 패킷을 상세히 분석
  - 보고서는 2장 내로 작성
- 팀 대표가 <u>barcel@naver.com</u>으로 제출 (5.7일까지)
  - Title: [컴퓨터네트워크][학번][이름][과제\_N]
  - Content: github repo url

팀명: 길동이네

팀원: 홍길동(학번), 고길동(학번)