

NSCAP HW1

Outcome Screenshot(using test.pcap)

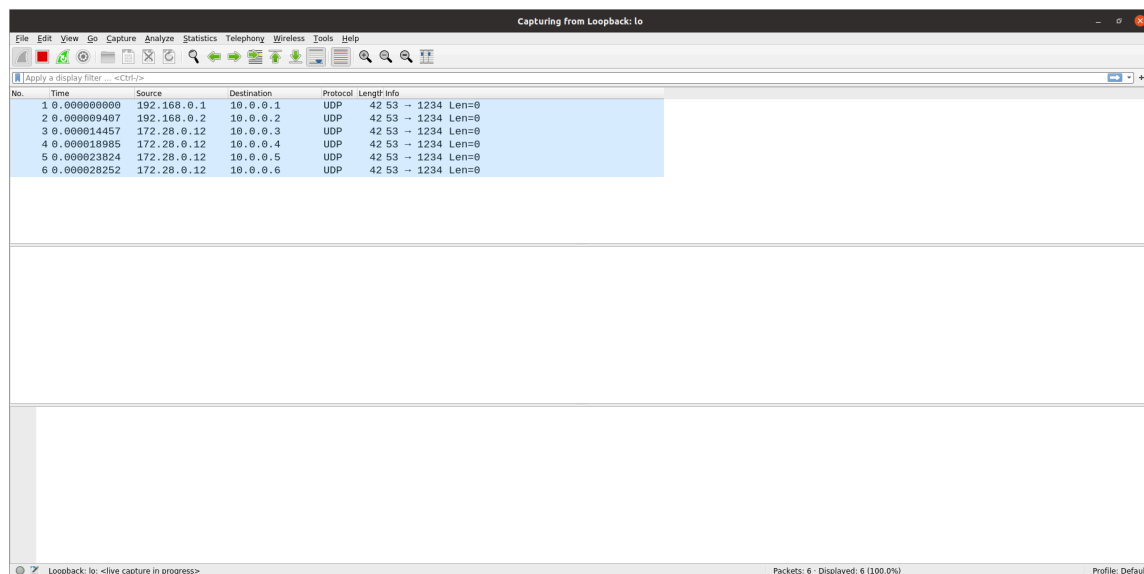
original packet

- Inject original packets in **test.pcap** to *loopback interface* without true time difference.

- o Tcpdump

```
in2023-lab1@cn2023lab1-VirtualBox:~$ sudo tcpdump -i lo -n -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
18:03:55.738738 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 192.168.0.1.53 > 10.0.0.1.1234: domain [length 0 < 12] (invalid)
18:03:55.738748 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 192.168.0.2.53 > 10.0.0.2.1234: domain [length 0 < 12] (invalid)
18:03:55.738753 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 172.28.0.12.53 > 10.0.0.3.1234: domain [length 0 < 12] (invalid)
18:03:55.738757 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 172.28.0.12.53 > 10.0.0.4.1234: domain [length 0 < 12] (invalid)
18:03:55.738762 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 172.28.0.12.53 > 10.0.0.5.1234: domain [length 0 < 12] (invalid)
18:03:55.738767 02:42:ac:1c:00:0c > 02:42:3a:0a:3c:c3, ethertype IPv4 (0x0800), length 42: 172.28.0.12.53 > 10.0.0.6.1234: domain [length 0 < 12] (invalid)
```

- o Wireshark



modified packet

- Parse **test.pcap** by `pcap_tcpreplay.cpp()` and modified
 - MAC source and destination address
 - IP source and destination address
- Inject modified packets in test.pcap to *loopback interface* with **true time difference**.

o Tcpdump

```
cn2023-lab1@cn2023lab1-VirtualBox: ~$ sudo tcpdump -i lo -n -e
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
18:12:14.522591 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
18:12:15.544961 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
18:12:17.569255 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
18:12:20.592744 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
18:12:22.617682 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
18:12:23.690645 08:00:12:34:56:78 > 08:00:12:34:ac:c2, ethertype IPv4 (0x0800), length 42: 10.1.1.3.53 > 10.1.1.4.1234: domain [length 0 < 12] (invalid)
```

o Wireshark

*Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0
2	1.022369824	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0
3	3.046664446	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0
4	6.070153213	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0
5	8.095091295	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0
6	9.168053794	10.1.1.3	10.1.1.4	UDP	42	53 → 1234 Len=0

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface lo, id 0
Ethernet II, Src: BellAtla_34:56:78 (08:00:12:34:56:78), Dst: BellAtla_34:ac:c2 (08:00:12:34:ac:c2)
Internet Protocol Version 4, Src: 10.1.1.3, Dst: 10.1.1.4
User Datagram Protocol, Src Port: 53, Dst Port: 1234

0000 08 00 12 34 ac c2 08 00 12 34 56 78 08 00 45 00 ...4... 4VX..E

0010 00 1c 00 01 00 00 40 11 b0 26 0a 01 01 03 0a 01@. .&.....

0020 01 04 00 35 04 d2 00 08 30 2d ...5... 0-

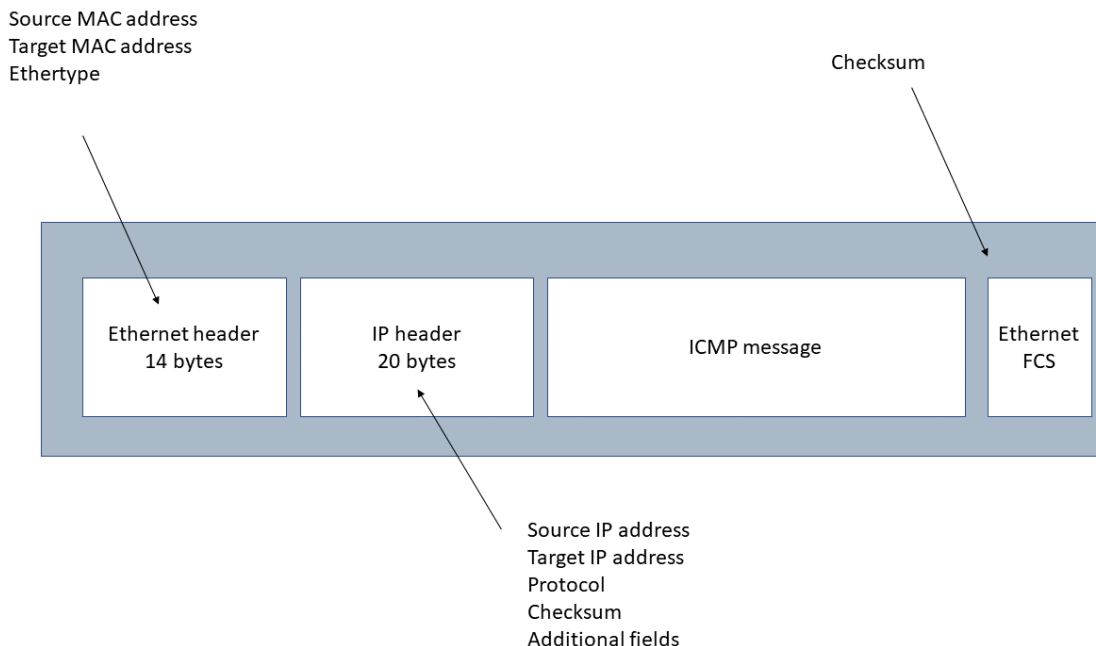
- Record of timestamp difference

```
TimeDiff: 1.02199 2.02199 3.02199 2.02199 1.02199
```

Additional Harvest

I got some additional harvest in the process of this lab.

1. Understood how to get the memory address of **IP part** in a packet



- This picture tell me that why "ETH_HLEN" is necessary to add in back of "packet" in this code.
- The reason is we need to **skip** the "ether part" to acces "IP part".

```
struct ip *ip_header = (struct ip *)(packet + ETH_HLEN); //point to IP part
```



2. To look up the library

After this lab, I have some sense to look up the related **library** when I have no idea about some components; for instance, I didn't know what is the type of **struct pcap_pkthdr** in the beginning.

```
struct pcap_pkthdr {
    struct timeval ts;      /* time stamp */
    bpf_u_int32 caplen;     /* length of portion present in data */
    bpf_u_int32 len;        /* length of this packet prior to any slicing */
};
```

After I saw the resource in some reference website (e.g. GitHub), I would realize what components in this type.

3. The format in Network is DIFFERENT

I asked ChatGPT why we need to use the function like these `ntohs` , `htons` .



You

`ntohs(eth_header->ether_type)` 是什麼意思?



ChatGPT

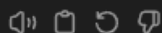
`ntohs` 和 `htons` 是网络字节序和主机字节序之间进行转换的函数，它们在 C 语言的头文件 `netinet/in.h` 中定义。这两个函数用于处理16位整数（如端口号和协议号），确保在不同的机器上正确地进行字节顺序转换。

具体来说：

- `ntohs` ("network to host short") 用于将16位无符号短整数从网络字节序转换为主机字节序。
- `htons` ("host to network short") 用于将16位无符号短整数从主机字节序转换为网络字节序。

这两个函数通常在处理网络通信时使用，因为网络通信要求数据在不同机器之间以统一的网络字节序进行传输。在这种情况下，接收方需要将接收到的数据从网络字节序转换为本地主字节序，而发送方则需要将本地数据转换为网络字节序。

在上下文中，`ntohs(eth_header->ether_type)` 可能用于将以太网帧头中的协议类型字段从网络字节序转换为主机字节序，以便更容易进行比较或其他操作。



Therefore, I learned that Network uses "Big Endian" and our system uses "Little Endian" storing information such that we need to transform the format by functions.

Example for thest two format:

Big Endian: 12 34 56 78

Little Endian: 78 56 34 12

Reference

- [<netinet/ip.h>](https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip.h.html) (<https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip.h.html>).
- [<netinet/ether.h>](https://sites.uclouvain.be/SystInfo/usr/include/net/ethernet.h.html) (<https://sites.uclouvain.be/SystInfo/usr/include/net/ethernet.h.html>).
- ChatGPT