ARP Packet

2020.07.13 김현진

ARP

ARP가 왜 필요해?

ARP가 뭐야?

ARP가 뭐야?

주소 결정 프로토콜(**A**ddress **R**esolution **P**rotocol, ARP)

네트워크 상에서 IP주소를 이용하여
물리적 주소인 MAC주소로 대응시키기
위해 사용하는 프로토콜

주로 요청과 응답을 통해 IP 주소와 MAC 주소 정보를 조회하고 알려주는 **패킷**

ARP가 뭐야?

IntelCor_d1:61:7c	ARP	42	192.168.35.1 is at 00:23:aa:de:99:91
Broadcast	ARP	42	Who has 192.168.35.1? Tell 192.168.35.105
HFR_de:99:91	ARP	60	Who has 192.168.35.1? Tell 192.168.35.2
HFR_de:99:91	ARP	60	Who has 192.168.35.1? Tell 192.168.35.2
IntelCor_d1:61:7c	ARP	42	192.168.35.1 is at 00:23:aa:de:99:91

IP 에 해당하는

MAC 주소를 알고 싶어!!!

IP

IP 에 해당하는 **MAC** 주소를

알려줄께!

MAC

ARP

ARP가 왜 필요해?

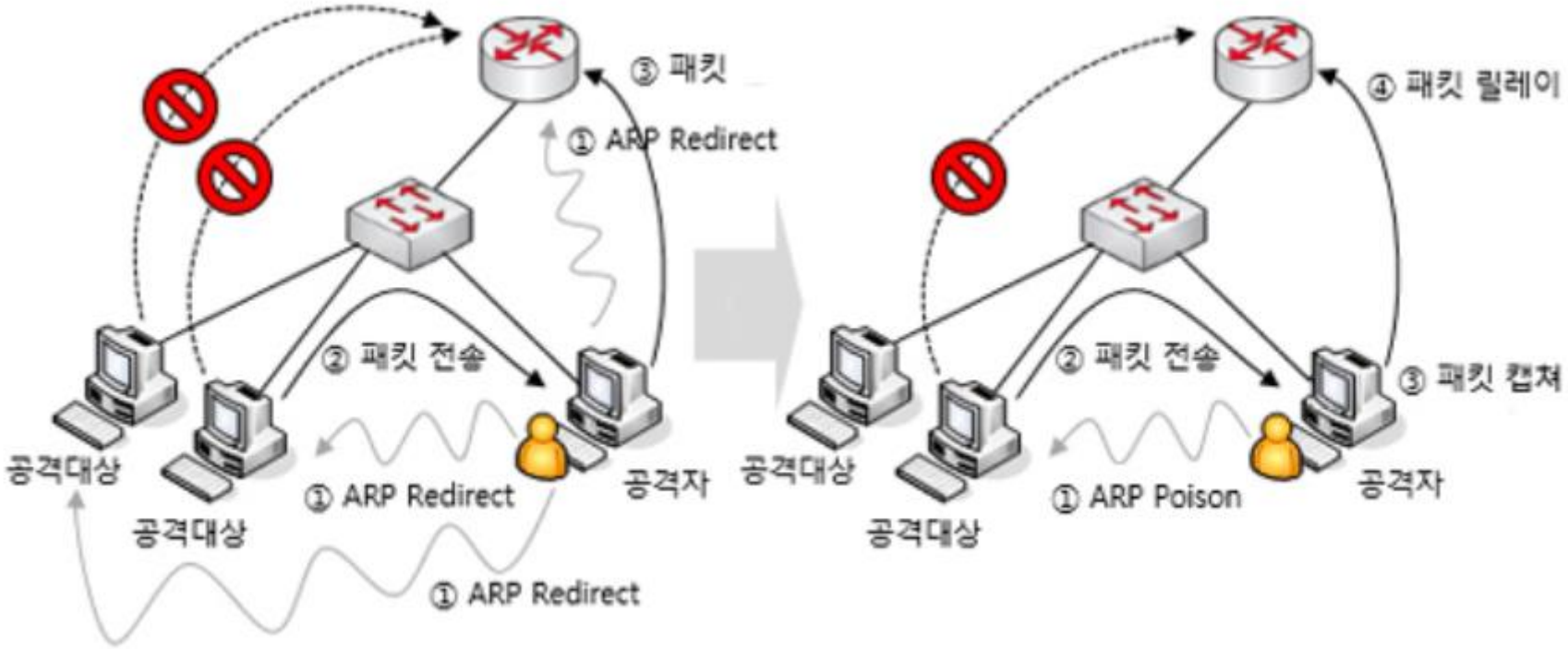
ARP가 뭐야?

ARP가 왜 필요해?

Application Layer ✓ Message format, Human-Machine Interfaces
Presentation Layer ✓ Coding into 1s and 0s; encryption, compression
Session Layer ✓ Authentication, permissions, session restoration
Transport Layer ✓ End-to-end error control
Network Layer ✓ Network addressing; routing or switching
Data Link Layer ✓ Error detection, flow control on physical link
Physical Layer ✓ Bit stream: physical medium, method of representing bits

네트워크 동작 방식이 **계층별**로 행동하기 때문이다.

ARP 공격



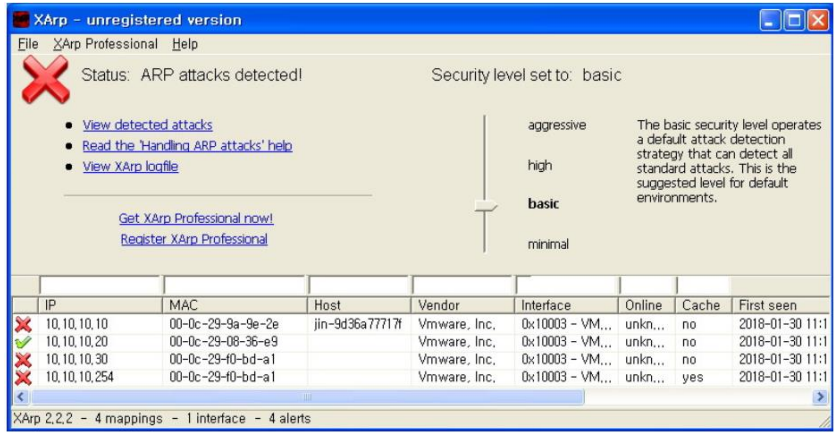
ARP redirect

모든 호스트 대 라우터

ARP spoofing

호스트 대 호스트

ARP 방어



1. 게이트 웨이 ARP 테이블을 **정적**으로 고정시켜 놓는다.
2. ***XARP*** 툴을 설치한다.

XARP

K_XARP

중복된 MAC주소가 있는지 확인하는 방법

인증 테이블과 Reply 패킷을 비교하는 방법

Request 에 대응하는 Reply 패킷이 올바른지 체크하는 방법

ARP Packet

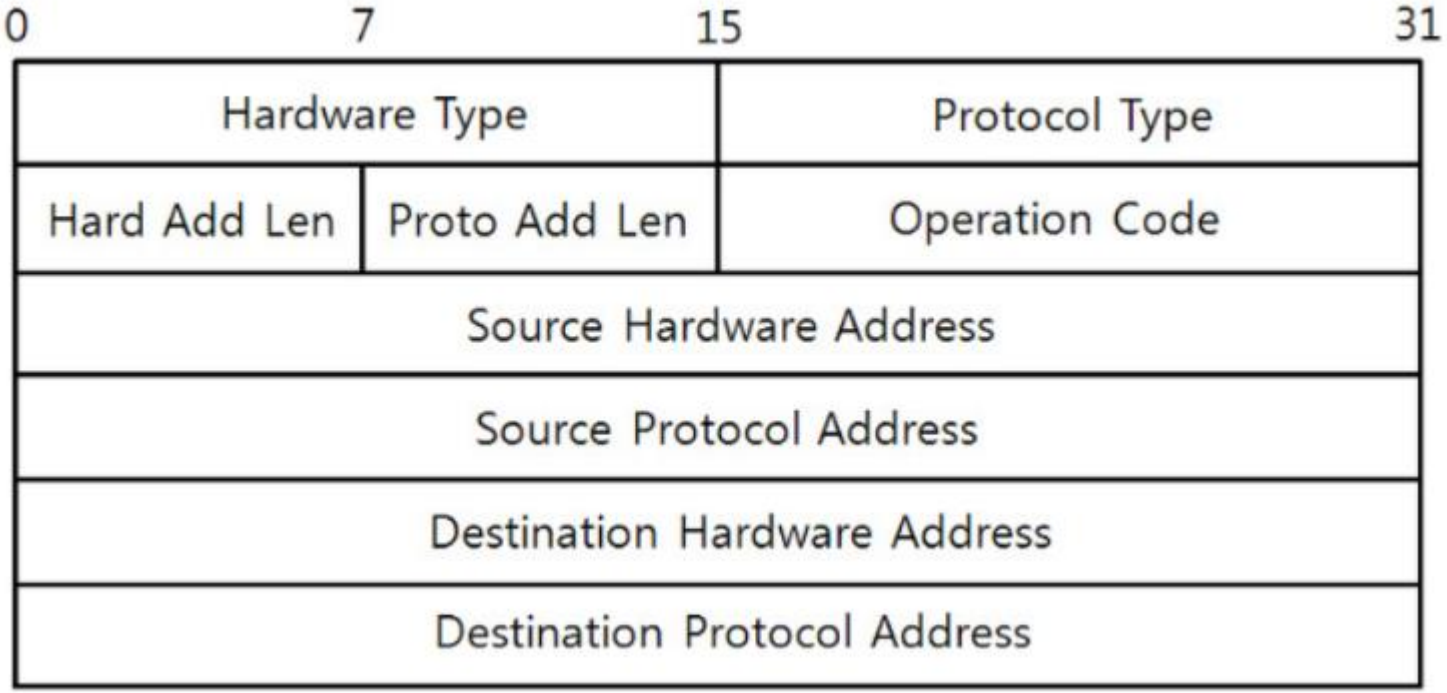
```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```

ARP Packet

```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```



ARP Packet

```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```

Type	하드웨어 타입 설 명
1	Ethernet (10Mb)
2	Experimental Ethernet (3Mb)
3	Amateur Radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802.3 networks
7	ARCNET
8	Hyperchnnel
9	Lanstar
10	Autonet Short Address
11	LoaclTalk
12	LocalNet (IBM PCNet or SYTEK LocalNET)

31

col Type

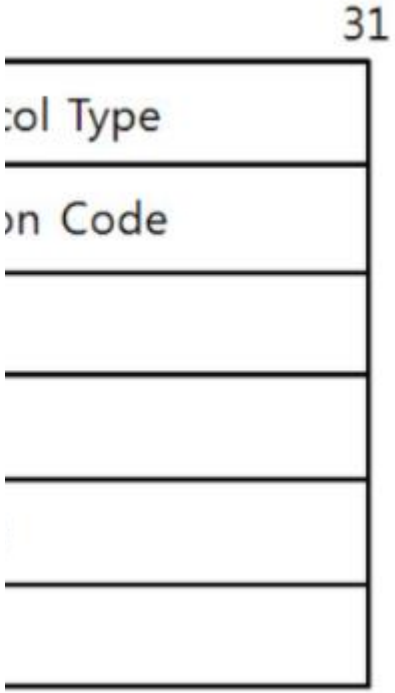
on Code

ARP Packet

```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```

```
/* Ethernet protocol ID's */
#define ETHERTYPE_PUP          0x0200
#define ETHERTYPE_SPRITE       0x0500
#define ETHERTYPE_IP           0x0800
#define ETHERTYPE_ARP          0x0806
#define ETHERTYPE_REVARP       0x8035
#define ETHERTYPE_AT           0x809B
#define ETHERTYPE_AARP         0x80F3
#define ETHERTYPE_VLAN         0x8100
#define ETHERTYPE_IPX          0x8137
#define ETHERTYPE_IPV6         0x86dd
#define ETHERTYPE_LOOPBACK     0x9000
```

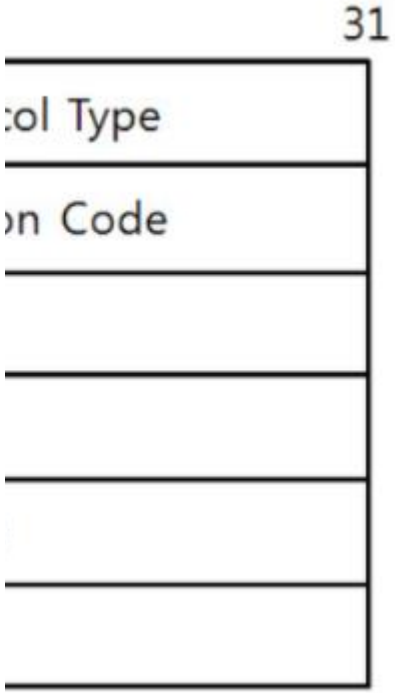


ARP Packet

```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```

```
#define ARP_REQUEST 1
#define ARP_REPLY 2
```



ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define ARP_REQUEST 1
#define ARP_REPLY 2

/* netinet/ether.h
struct ether_header {   // size: 14byte
    uint8_t ether_dhost[6];
    uint8_t ether_shost[6];
    uint16_t ether_type;
};
*/

struct arp_header {  // size: 20byte
    uint16_t hard_type;  // 하드우에어 주소 타입(
    uint16_t prto_type;  // ipv4, ipv6 등의 프로
    uint8_t hard_len;    // 6
    uint8_t prto_len;    // 4
    uint16_t opcode;     // request=1 / reply=2
    uint8_t src_mac[6];
    uint8_t src_ip[4];
    uint8_t dst_mac[6];
    uint8_t dst_ip[4];
};

```

```c
struct arp_header {  // size: 20byte
    uint16_t hard_type;  // 하드우에어 주소 타입(네트워크 유형)
    uint16_t prto_type;  // ipv4, ipv6 등의 프로토콜
    uint8_t hard_len;    // 6
    uint8_t prto_len;    // 4
    uint16_t opcode;     // request=1 / reply=2
    uint8_t src_mac[6];
    uint8_t src_ip[4];
    uint8_t dst_mac[6];
    uint8_t dst_ip[4];
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
    }

handle
            return -1;
        }

        struct ether_header * ether = (struct ether_header *)packet;
        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define AR
#define AR

/* netinet
struct eth
    uint8_
    uint8_
    uint16
};
*/

struct arp
    uint16
    uint16
    uint8_
    uint8_
    uint16
    uint8_
    uint8_
    uint8_
    uint8_
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
```

```c
    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
        printf("pcap_open_live error %s\n(%s)",dev,errbuf);
        return -1;
    }

    while(1) {
        struct pcap_pkthdr * header;
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf(" pcap_next_ex error \n");
            return -1;
        }
```

promiscuous

```c
// open handle

ket;

    u_short eth_type = ether -> ether_type;
    eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define AR
#define AR

/* netinet
struct eth
    uint8_
    uint8_
    uint16
};
*/

struct arp
    uint16
    uint16
    uint8_
    uint8_
    uint16
    uint8_
    uint8_
    uint8_
    uint8_
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
```

```c
    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
        printf("pcap_open_live error %s\n(%s)",dev,errbuf);
        return -1;
    }

    while(1) {
        struct pcap_pkthdr * header;
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf(" pcap_next_ex error \n");
            return -1;
        }
```

패킷을 읽는 시간

```c
// open handle

ket;

        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define AR
#define AR

/* netinet
struct eth
    uint8_
    uint8_
    uint16
};
*/

struct arp
    uint16
    uint16
    uint8_
    uint8_
    uint16
    uint8_
    uint8_
    uint8_
    uint8_
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
```

```c
    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
        printf("pcap_open_live error %s\n(%s)",dev,errbuf);
        return -1;
    }

    while(1) {
        struct pcap_pkthdr * header;
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf(" pcap_next_ex error \n");
            return -1;
        }
```

-1 : 에러종료
0 : 정상종료

```c
// open handle

ket;

    u_short eth_type = ether -> ether_type;
    eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define AR
#define AR

/* netinet
struct eth
    uint8_
    uint8_
    uint16
};
*/

struct arp
    uint16
    uint16
    uint8_
    uint8_
    uint16
    uint8_
    uint8_
    uint8_
    uint8_
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
```

```c
    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
        printf("pcap_open_live error %s\n(%s)",dev,errbuf);
        return -1;
    }

    while(1) {
        struct pcap_pkthdr * header;
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf(" pcap_next_ex error \n");
            return -1;
        }
```

const : 상수

```c
// open handle
ket;
u_short eth_type = ether -> ether_type;
eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define AR
#define AR

/* netinet
struct eth
    uint8_
    uint8_
    uint16
};
*/

struct arp
    uint16
    uint16
    uint8_
    uint8_
    uint16
    uint8_
    uint8_
    uint8_
    uint8_
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
```

```c
    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
        printf("pcap_open_live error %s\n(%s)",dev,errbuf);
        return -1;
    }

    while(1) {
        struct pcap_pkthdr * header;
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf(" pcap_next_ex error \n");
            return -1;
        }
```

캡처 된 패킷의 데이터를
불러오는 함수

```c
// open handle

ket;

        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define ARP_REQUEST 1
#define ARP_REPLY 2

/* netinet/ether.h
struct ether_header {   // size: 14byte
    uint8_t ether_dhost[6];
    uint8_t ether_
    uint16_t ether
};
*/

struct arp_header
    uint16_t hard_type;
    uint16_t prto_type;  // ipv4, ipv6 등의 프로토콜
    uint8_t hard_len;    // 6
    uint8_t prto_len;    // 4
    uint16_t opcode;     // request=1 / reply=2
    uint8_t src_mac[6];
    uint8_t src_ip[4];
    uint8_t dst_mac[6];
    uint8_t dst_ip[4];
};
```

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
    }

    char * dev = argv[1];
    char errbuf [PCAP_ERRBUF_SIZE];

    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
                                                                  uf);
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf( pcap_next_ex error \n");
            return -1;
        }

        struct ether_header * ether = (struct ether_header *)packet;
        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

```c
struct ether_header * ether = (struct ether_header *)packet;
u_short eth_type = ether -> ether_type;
eth_type = ntohs(eth_type);
```

packet 에서 이더넷 헤더를 가져온다.

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define ARP_REQUEST 1
#define ARP_REPLY 2

/* netinet/ether.h
struct ether_header {   // size: 14byte
    uint8_t ether_dhost[6];
    uint8_t ether_
    uint16_t ether
};
*/

struct arp_header
    uint16_t hard_type;
    uint16_t prto_type;  // ipv4, ipv6 등의 프로토콜
    uint8_t hard_len;    // 6
    uint8_t prto_len;    // 4
    uint16_t opcode;     // request=1 / reply=2
    uint8_t src_mac[6];
    uint8_t src_ip[4];
    uint8_t dst_mac[6];
    uint8_t dst_ip[4];
};
```

```c
struct ether_header * ether = (struct ether_header *)packet;
u_short eth_type = ether -> ether_type;
eth_type = ntohs(eth_type);
```

-> : 화살표 연산자
구조체 멤버에 접근

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
    }

    char * dev = argv[1];
    char errbuf [PCAP_ERRBUF_SIZE];

    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
                                                                  uf);
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf( pcap_next_ex error \n");
            return -1;
        }

        struct ether_header * ether = (struct ether_header *)packet;
        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

ARP Packet

```c
#include <stdio.h>
#include <pcap.h>
#include <string.h>  // for memcpy
#include <netinet/ether.h> //ether_header , ARP=0x0806

#define ARP_REQUEST 1
#define ARP_REPLY 2

/* netinet/ether.h
struct ether_header {   // size: 14byte
    uint8_t ether_dhost[6];
    uint8_t ether_
    uint16_t ether
};
*/

struct arp_header
    uint16_t hard_type;
    uint16_t prto_type;  // ipv4, ipv
    uint8_t hard_len;    // 6
    uint8_t prto_len;    // 4
    uint16_t opcode;     // request=1 / reply=2
    uint8_t src_mac[6];
    uint8_t src_ip[4];
    uint8_t dst_mac[6];
    uint8_t dst_ip[4];
};
```

```c
struct ether_header * ether = (struct ether_header *)packet;
u_short eth_type = ether -> ether_type;
eth_type = ntohs(eth_type);
```

네트워크 바이트 오더에서
호스트 바이트 오더로 변경

```c
int main(int argc, char * argv[])  {

    if (argc!=2) {
        return -1;
    }

    char * dev = argv[1];
    char errbuf [PCAP_ERRBUF_SIZE];

    pcap_t * handle = pcap_open_live(dev,BUFSIZ,1,1000,errbuf);   // open handle
    if (handle ==NULL) {
                                                                  uf);
        const u_char * packet;
        int res = pcap_next_ex(handle, &header, &packet);
        if (res == 0) continue;
        if (res == -1 || res == -2) {
            printf( pcap_next_ex error \n");
            return -1;
        }

        struct ether_header * ether = (struct ether_header *)packet;
        u_short eth_type = ether -> ether_type;
        eth_type = ntohs(eth_type);
```

ARP Packet

```
59          packet += sizeof(struct ether_header);
60
61          struct arp_header * arp = (struct arp_header *)packet;
62          u_short arp_hardware = arp->hard_type;
63          arp_hardware = ntohs(arp_hardware);
64          u_short arp_protocol = arp->prto_type;
65          arp_protocol = ntohs(arp_protocol);
66          u_char arp_hlen = arp->hard_len;
67          u_char arp_plen = arp->prto_len;
68          u_short arp_opcode = arp->opcode;
69          arp_opcode = ntohs(arp_opcode);
70
71          packet += sizeof(struct ether_header);
72
73
74              printf("Hardware : %02x \n",arp_hardware);
75              printf("Protocol : %02x \n",arp_protocol);
76              printf("Hlen : %d \n",arp_hlen);
77              printf("Plen : %d \n",arp_plen);
78              printf("Opcode : %d \n",arp_opcode);
79              printf("Src Mac : %02x:%02x:%02x:%02x:%02x:%02x \n", ether->ether_shost[0],ether->ether_shost[1],
80                  ether->ether_shost[2],ether->ether_shost[3],ether->ether_shost[4],ether->ether_shost[5]);
81              printf("Src Ip : %d.%d.%d.%d \n",arp->src_ip[0],arp->src_ip[1],arp->src_ip[2],arp->src_ip[3]);
82              printf("Dst Mac : %02x:%02x:%02x:%02x:%02x:%02x \n",ether->ether_dhost[0],ether->ether_dhost[1],
83                  ether->ether_dhost[2],ether->ether_dhost[3],ether->ether_dhost[4],ether->ether_dhost[5]);
84              printf("Dst Ip : %d.%d.%d.%d \n",arp->dst_ip[0],arp->dst_ip[1],arp->dst_ip[2],arp->dst_ip[3]);
85
86          }
87      }
88  }
```

이더넷 패킷을 다 읽었으므로
이더넷 크기만큼 이동

ARP Packet

```
--------- [ARP packet] ---------

Hardware : 01
Protocol : 800
Hlen : 6
Plen : 4
Opcode : 1
Src Mac : 00:0c:29:85:76:5c
Src Ip : 192.168.35.2
Dst Mac : 00:23:aa:de:99:91
Dst Ip : 192.168.35.1
```

Q & A

Thank you ☺