

## Error 처리

```
#include <errno.h>

cIntSd = socket(PF_INET, SOCK_STREAM, 0);
if(cIntSd == -1){
    //errno가 EINVAL과 같다면 (기본적으로는 -1, 사용자 정의도 가능)
    if(errno == EINVAL){
        printf("protocol family error");
        return -1;
    }
    printf("socket create error");
    return -1;
}
```

## IPv6

IP 주소에 0이 연속될 경우 생략 가능

- fe80:0000:0000<sup>1 2</sup><sub>3 4</sub>5678:9012:3456:7890 -> fe80:<sup>1 2</sup><sub>3 4</sub>5678:9012:3456:7890
- 특수 공간 :: -> 주소의 모든 값을 0으로 세팅 ::1 -> loopback 주소(=127.0.0.1)

```
inet_pton(AF_INET, "10.12.110.57", ~); //IPv4
inet_pton(AF_INET6, "2001:db8:63b3::3490", ~); //IPv6
```

IPv4	IPv6
inet_aton()	inet_pton()
inet_addr()	
inet_ntoa()	inet_ntop()
gethostbyname()	getaddrinfo()

## Packet Socket

AF\_INET / AF\_INET6 대신 **AF\_PACKET**

-> 데이터 링크 계층까지 수정 가능

protocol	packet	result
SOCK_RAW		TCP/UDP Header
SOCK_RAW		IP Header
IP_HDRINCL=TRUE		
	AF_PACKET	IP Header

protocol	packet	result
SOCK_RAW	AF_PACKET	Link-Layer Header
SOCK_DGRAM	AF_PACKET	헤더까지 수정 가능하지만 헤더는 자동으로 맡기고 데이터만 수정

- SOCK\_DGRAM 설정시 L2의 헤더는 자동 생성됨
- SOCK\_DGRAM 설정해도 dst.mac은 수정 가능하다

### sendrecv.c

```
memset(&sockAddr, 0, sizeof(struct sockaddr_ll));
sockAddr.sll_family = AF_PACKET;
//APR 패킷으로 설정
sockAddr.sll_protocol = htons(ETH_P_ARP);
sockAddr.sll_ifindex = if_nametoindex("eth0");
sockAddr.sll_halen = ETH_ALEN;

//L2 헤더 + 데이터 수정 가능, 하지만 헤더는 수정x, 모든 패킷 수신 가능
pSocket = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_ALL));
```

## MAC address and ARP

### IP

- 32bit 주소
- L3 - 네트워크 계층에서 사용하는 주소
- IP는 동적으로 변경되기 때문에 이식성 없다

### MAC

- 48bit 주소
- LAN, physical, Ethernet이라고도 불린다.
- ntranet에서는 mac만으로 통신 가능하지만 호환성을 위해 IP까지 부여
- BroadCast: FF-FF-FF-FF-FF-FF
- NIC 칩에 박혀있는 주소이기 때문에 호환성이 존재한다.

### ARP

#### IP주소로 MAC 주소 알아오기

1. A가 B에게 데이터그램을 보내길 원함 (A, B 모두 같은 intranet)
2. 하지만 B의 MAC은 A의 ARP table에 없다.
3. A는 B의 IP로 브로드캐스트  
dstIP = B  
dstMAC = ff-ff-ff-ff-ff-ff  
LAN의 모든 노드가 ARP 패킷을 수신함
4. B가 수신 후, A에게 B의 MAC주소를 전달한다. (unicast)
5. A는 B의 MAC주소를 알게되고, 자신의 ARP Table에 TTL과 함께 저장한다.

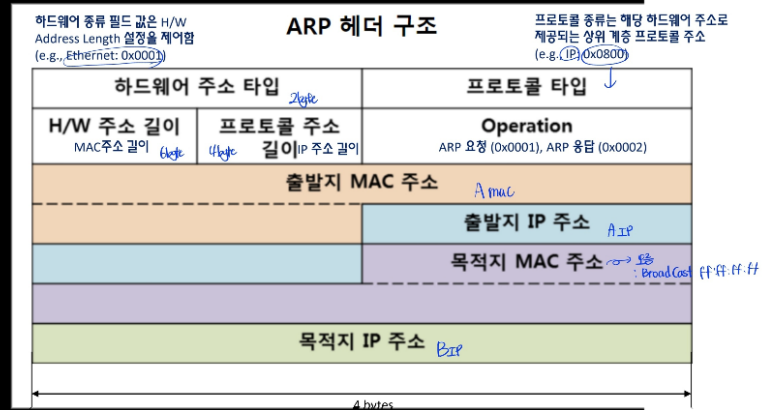
## ARP is plug-and-play

따로 세팅할 필요 없이 바로 사용 가능하다

## ARP Header

```
struct arp_ha {
    unsigned char ha[6]; mac
};
struct arp_pa {
    unsigned char pa[4]; IP
};

struct arp_packet {
    uint16_t ar_hrd; ethernet
    uint16_t ar_pro; IP Protocol
    uint8_t ar_hln; 길이
    uint8_t ar_pln; 길이
    uint16_t ar_op; operation
    struct arp_ha ar_sha; A mac
    struct arp_pa ar_spa; A IP
    struct arp_ha ar_tha; BroadCast
    struct arp_pa ar_tpa; B IP
};
```



## ARP Code

```
1 #include <sys/socket.h>
2 #include <netpacket/packet.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <linux/if_ether.h>
7 #include <net/if.h>
8 #include <errno.h>
9 #include <sys/types.h>
10 #include <stdint.h>
11 #include <ctype.h>
12 #include <errno.h>
13 #include <arpa/inet.h>
14
15 struct arp_ha {
16     unsigned char ha[6];
17 };
18 struct arp_pa {
19     unsigned char pa[4];
20 };
21
22 struct arp_packet {
23     uint16_t ar_hrd;
24     uint16_t ar_pro;
```

```

25     uint8_t ar_hln;
26     uint8_t ar_pln;
27     uint16_t ar_op;
28     struct arp_ha ar_sha;
29     struct arp_pa ar_spa;
30     struct arp_ha ar_tha;
31     struct arp_pa ar_tpa;
32 };
33
34 int convertTextToArppa(char *, struct arp_pa *);
35 int convertTextToArpha(char *, struct arp_ha *);
36 void errProc(const char*);
37 int main(int argc, char** argv)
38 {
39     int pSocket;
40     struct arp_packet * buff;
41     int hdrLen;
42     struct sockaddr_ll sockAddr;
43     int res, i;
44     int tempInt;
45     unsigned char test;
46
47     if(argc != 4)
48     {
49         fprintf(stderr, "Usage: %s <Interface> <SenderIP> <TargetIP> \n", argv[0]);
50         return -1;
51     }
52     memset(&sockAddr, 0, sizeof(struct sockaddr_ll));
53     //L2 헤더까지 제어
54     sockAddr.sll_family = AF_PACKET;
55     //ARP 패킷
56     sockAddr.sll_protocol = htons(ETH_P_ARP);
57     //이더넷 인터페이스 이름 입력
58     sockAddr.sll_ifindex = if_nametoindex(argv[1]);
59     sockAddr.sll_halen = ETH_ALEN;
60     //Broadcast 정의
61     sockAddr.sll_addr[0] = 0xff;
62     sockAddr.sll_addr[1] = 0xff;
63     sockAddr.sll_addr[2] = 0xff;
64     sockAddr.sll_addr[3] = 0xff;
65     sockAddr.sll_addr[4] = 0xff;
66     sockAddr.sll_addr[5] = 0xff;
67
68     //L2헤더까지 제어, mac주소는 수정 x, 모든 패킷을 수신할 수 있음
69     pSocket = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_ALL));
70     buff = (struct arp_packet *) malloc(sizeof(struct arp_packet));
71     hdrLen = sizeof(struct arp_packet);
72     buff->ar_hrd = htons(0x0001); //이더넷
73     buff->ar_pro = htons(0x0800); //IP 프로토콜
74     buff->ar_hln = 6; //mac주소 길이
75     buff->ar_pln = 4; //IP주소 길이
76     buff->ar_op = htons(0x0001); //ARP Request
77     convertTextToArpha("08:00:27:ea:85:20", &(buff->ar_sha)); //테스트 머신의 MAC 주소
78     convertTextToArppa(argv[2], &(buff->ar_spa)); //테스트 머신의 IP 주소
79     convertTextToArppa(argv[3], &(buff->ar_tpa)); //알고자하는 MAC 주소와 연관된 IP 주소
80     for(i = 0; i < 10; i++) {
81         //buff에 ARP헤더 존재, sockAddr = 브로드캐스트

```

```
82     res = sendto(pSocket, buff, hdrLen, 0, (struct sockaddr *)&sockAddr, sizeof
83     if(res < 0)
84     {
85         fprintf(stderr, "sendto: %s \n", strerror(errno));
86     }
87     printf("%d sent \n", res);
88 }
89 close(pSocket);
90 return 1;
91 }
92
93 int convertTextToAlpha(char *str, struct arp_ha *address)
94 {
95     char *ptrStr;
96     int len, i, j, k;
97     unsigned int temp;
98     ptrStr = str;
99     //strlen("xx:xx:xx:xx:xx:xx") == 17
100    if(strlen(str) != 17) errProc("convertTextToAlpha address length");
101    temp = k = j = 0;
102    for(i = 0; i < 17; i++)
103    {
104        j = i % 3;
105        if(j == 2)
106        {
107            ptrStr++;
108            k++;
109            continue;
110        }
111
112        switch(*ptrStr)
113        {
114            case 'a':
115            case 'A':
116                temp = 10;
117                temp *= (j==0?16:1);
118                break;
119            case 'b':
120            case 'B':
121                temp = 11;
122                temp *= (j==0?16:1);
123                break;
124
125            case 'c':
126            case 'C':
127                temp = 12;
128                temp *= (j==0?16:1);
129                break;
130
131            case 'd':
132            case 'D':
133                temp = 13;
134                temp *= (j==0?16:1);
135                break;
136
137            case 'e':
138            case 'E':
```

```
139         temp = 14;
140         temp *= (j==0?16:1);
141         break;
142
143         case 'f':
144         case 'F':
145             temp = 15;
146             temp *= (j==0?16:1);
147             break;
148         default:
149             temp = (unsigned char) *ptrStr;
150             temp -= 48;
151             if(j==0) temp *= 16;
152     }
153     if(j==0) address->ha[k] = temp;
154     if(j==1) address->ha[k] = address->ha[k] + temp;
155     ptrStr = ptrStr + 1;
156 }
157 return 1;
158 }
159
160 int convertTextToArppa(char *str, struct arp_pa *address)
161 {
162     char *ptrStr;
163     int i;
164
165     ptrStr = str;
166     i = 0;
167     if(address == NULL) return -1;
168     while(*ptrStr)
169     {
170         if(isdigit((unsigned char)*ptrStr)) {
171             address->pa[i] *= 10;
172             address->pa[i] += *ptrStr - '0';
173         }
174         else{
175             i++;
176         }
177         ptrStr++;
178     }
179
180
181     return 0;
182 }
183
184 void errProc(const char* str) {
185     fprintf(stderr,"%s: %s\n", str, strerror(errno));
186     exit(1);
187 }
188 }
189
```

CoLo