9주차 실습

2023. 05

데이터통신

TA 임병현 - lbhzzang@o.cnu.ac.kr

9주차 실습

- Layer communication
  - 코드 설명
- ARP
  - ARP 패킷 캡처
  - ARP 코드 구현

9주차 첨부 파일 설명

9주차.pptx	2023-04-27 오전 11:33	Microsoft PowerP...	561KB
addr_receiver.c	2015-06-23 오후 3:05	C Source file	6KB
addr_sender.c	2015-06-23 오후 2:49	C Source file	6KB
ARP_capture.c	2023-04-27 오전 11:36	C Source file	7KB
ARP_dest.c	2015-06-23 오후 2:29	C Source file	6KB
ARP_sender.c	2015-06-23 오후 2:38	C Source file	11KB
dataComm_final.c	2023-04-27 오전 10:57	C Source file	17KB
dataComm_ver2.c	2023-04-28 오후 4:37	C Source file	12KB

addr_sender.c	ip,mac 발신자(입력값 전송)
addr_receiver.c	ip,mac 수신자(고정값과 비교)
dataComm_ver2.c	sender+receiver 통합
ARP_capture.c	로컬에서 ARP 캡처
ARP_sender.c	ARP cache table 확인후 전송
ARP_dest.c	수신자
dataComm_final.c	7주차 과제 소스

과제 확인

```
lbh@lbh-VirtualBox:~/datacomm/202250245$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.6  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a6f5:388a:775:b2ca  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:27:e9:b1  txqueuelen 1000  (Ethernet)
        RX packets 114025  bytes 23796827 (23.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 119101  bytes 22276072 (22.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 552  bytes 56619 (56.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 552  bytes 56619 (56.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lbh@lbh-VirtualBox:~/datacomm/202250245$ ls
202250245.c  a.out
lbh@lbh-VirtualBox:~/datacomm/202250245$ ./a.out
Session 2 starting...
#####################################################
[READY]sndsock port: 8811, rcvsock port: 8810
[READY]IP: 127.0.0.1
#####################################################
[Find Adress Mode - Request ...]
[L2_receive] your MAC --> 00:10:00:0A:00:00
[L1_receive] your IP --> 192.168.0.1
[Mode Change]
hi
```

```
lbh@lbh-VirtualBox:~/datacomm/202250245$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.6  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a6f5:388a:775:b2ca  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:27:e9:b1  txqueuelen 1000  (Ethernet)
        RX packets 114001  bytes 23794937 (23.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 119086  bytes 22273626 (22.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 552  bytes 56619 (56.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 552  bytes 56619 (56.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lbh@lbh-VirtualBox:~/datacomm/202250245$ ls
202250245.c  a.out
lbh@lbh-VirtualBox:~/datacomm/202250245$ ./a.out
Session 1 starting...
#####################################################
[READY]sndsock port: 8810, rcvsock port: 8811
[READY]IP: 127.0.0.1
#####################################################
[Find Adress Mode - Response ...]
[L1_send] my IP --> 192.168.0.1
[L2_send] my MAC --> 00:10:00:0A:00:00
[Find Adress Mode - Success ...]
Received: hi
```

dataComm_final.c 확인하기

dataComm_final.c

L1_send

```c
// 형식상 맞춰줌
/*
    구현. IP 주소 헤더에 붙임


*/
L2_send(temp, size);
```

```c
length = sizeof(struct Addr); // Added Code
data.length = length;
memset(data.L1_data, 0x00, MAX_SIZE);
memcpy(data.L1_data, (void *)&addrData, sizeof(struct Addr));    // Added Code

size = sizeof(struct L1) - sizeof(data.L1_data) + length;
memset(temp, 0x00, 350);
memcpy(temp, (void *)&data, size);
L2_send(temp, size);
```

L2_send

```c
else
{
    struct L1 *tempL1 = (struct L1 *)input;
    struct Addr *tempAddr = (struct Addr *)tempL1->L1_data;
    memcpy(addrData.ip, tempAddr->ip, sizeof(addrData.ip));

    unsigned char mac[] = {0x00, 0x10, 0x00, 0x0A, 0x00, 0x00};
    printf("[%s] my MAC --> ", __func__);
    for (int i = 0; i < sizeof(mac); i++)
    {
        // 구현. addrData.mac 과 addr에 각각 mac[]의 값 할당

        // hint. 아래는 출력문임
        printf("%02X", addrData.mac[i]);
        if (i != 5)
            printf(":");
    }
    printf("₩n");
```

```c
struct L1 *tempL1 = (struct L1 *)input;
struct Addr *tempAddr = (struct Addr *)tempL1->L1_data;
memcpy(addrData.ip, tempAddr->ip, sizeof(addrData.ip));

unsigned char mac[] = {0x00, 0x10, 0x00, 0x0A, 0x00, 0x00};
printf("[%s] my MAC --> ", __func__);
for (int i = 0; i < sizeof(mac); i++)
{
    addrData.mac[i] = mac[i];
    addr.mac[i]= mac[i];
    printf("%02X", addrData.mac[i]);
    if (i != 5)
        printf(":");
}
printf("₩n");
```

dataComm_final.c

L2_send

```c
    data.length = length;

    // 구현. memset, cpy
    memset();
    memcpy();

    memset();
    memcpy();

    size = sizeof(struct L2) - sizeof(data.L2_data) + length;

    memset(temp, 0x00, 350);
    memcpy(temp, (void *)&data, size);
    L3_send(temp, size);
}
```

```c
struct L1 *tempL1 = (struct L1 *)input; // L1 데이터 복사용
struct Addr *tempAddr = (struct Addr *)tempL1->L1_data;  // L1 데이터 주소 가져오기
memcpy(addrData.ip, tempAddr->ip, sizeof(addrData.ip));  //size만큼 데이터를 addrData.ip에 복사

unsigned char mac[] = {0x00, 0x10, 0x00, 0x0A, 0x00, 0x00};
printf("[%s] my MAC --> ", __func__);
for (int i = 0; i < sizeof(mac); i++)
{
    addrData.mac[i] = mac[i];
    addr.mac[i]= mac[i];
    printf("%02X", addrData.mac[i]);
    if (i != 5)
        printf(":");
}
printf("₩n");

data.saddr[0] = 0x11;
data.saddr[1] = 0x12;
data.saddr[2] = 0x13;
data.saddr[3] = 0x14;
data.saddr[4] = 0x15;
data.saddr[5] = 0x16;

data.daddr[0] = 0x21;
data.daddr[1] = 0x22;
data.daddr[2] = 0x23;
data.daddr[3] = 0x24;
data.daddr[4] = 0x25;
data.daddr[5] = 0x26;


data.length = length;

memset(tempL1->L1_data, 0x00, MAX_SIZE);  //L1데이터 메모리 초기화
memcpy(tempL1->L1_data, (void *)&addrData, sizeof(struct Addr));   //주소 데이터 L1에 복사

memset(data.L2_data, 0x00, MAX_SIZE); //메모리 초기화
memcpy(data.L2_data, (void *)tempL1, length); // 데이터 복사

size = sizeof(struct L2) - sizeof(data.L2_data) + length;

memset(temp, 0x00, 350);
memcpy(temp, (void *)&data, size);
L3_send(temp, size);
```

dataComm_final.c

L2_send

```c
else if (control.type == 1)
{
    for (int i = 0; i < sizeof(addr.mac); i++)
    {
        // 구현. 데이터 값 대입
    }

    data.saddr[0] = 0x11;
    data.saddr[1] = 0x12;
    data.saddr[2] = 0x13;
    data.saddr[3] = 0x14;
    data.saddr[4] = 0x15;
    data.saddr[5] = 0x16;

    data.length = length;
    memset(data.L2_data, 0x00, MAX_SIZE);
    memcpy(data.L2_data, (void *)input, length);

    size = sizeof(struct L2) - sizeof(data.L2_data) + length;

    memset(temp, 0x00, 350);
    memcpy(temp, (void *)&data, size);
    L3_send(temp, size);
}
```

```c
else if (control.type == 1)
{
    //printf("[%s] daddr(MAC) setting --> ", __func__);
    for (int i = 0; i < sizeof(addr.mac); i++)
    {
        // printf("%02X", addr.mac[i]);
        // if (i != 5)  printf(":");
        data.daddr[i] = addr.mac[i];
    }
    //printf("₩n");
    /*printf("[%s] daddr(MAC) setting FINISH --> ", __func__);
    for (int i = 0; i < sizeof(addr.mac); i++)
    {
        printf("%02X", data.daddr[i]);
        if (i != 5)
            printf(":");
    }
    printf("₩n");   */
    data.saddr[0] = 0x11;
    data.saddr[1] = 0x12;
    data.saddr[2] = 0x13;
    data.saddr[3] = 0x14;
    data.saddr[4] = 0x15;
    data.saddr[5] = 0x16;
    /*
    data.daddr[0] = 0x21;
    data.daddr[1] = 0x22;
    data.daddr[2] = 0x23;
    data.daddr[3] = 0x24;
    data.daddr[4] = 0x25;
    data.daddr[5] = 0x26;
    */
    data.length = length;
    memset(data.L2_data, 0x00, MAX_SIZE);
    memcpy(data.L2_data, (void *)input, length);

    size = sizeof(struct L2) - sizeof(data.L2_data) + length;

    memset(temp, 0x00, 350);
    memcpy(temp, (void *)&data, size);
    L3_send(temp, size);
}
```

dataComm_final.c

L1_receive

```c
if (control.type == 2)
{
    data = (struct L1 *)L2_receive(length);
    if (is_server == 1)
    {
        // server
        printf("₩033[0;31m[Find Adress Mode - Response ...]₩n₩033[0m");
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    }
    else if (is_server == 0)
    {
        // client
        addrData = (struct Addr *)data->L1_data;
        // 구현  addrData 에 데이터 파싱된 값들 전역 변수에 할당 징행

        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    }
}
```

```c
if (control.type == 2)
{
    data = (struct L1 *)L2_receive(length);
     //if (strcmp(data->L1_data, "hello") == 0 && is_server == 1)
    if (is_server == 1)
    {
        // server
        // control.type=1;
        printf("₩033[0;31m[Find Adress Mode - Response ...]₩n₩033[0m");
        //printf("[Find Adress Mode - Response ...]₩n");
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    }
    else if (is_server == 0)
    {
        // client
        addrData = (struct Addr *)data->L1_data;
        printf("[%s] your IP --> ", __func__);
        for (int i = 0; i < sizeof(addrData->ip); i++)
        {
            printf("%hhu", addrData->ip[i]);
            addr.ip[i] = addrData->ip[i];
            if (i != 3)
                printf(".");
        }
        printf("₩n");
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    }
}
```

dataComm_final.c

L1_receive

```c
else if (control.type == 1)
{
    data = (struct L1 *)L2_receive(length);

    // 편의상 char 형태로 두개의 값을 비교
    char str_ip[16]; // my ip
    char str_daddr[16]; // receive ip
    // 구현 . sprintf(??)
    sprintf();
    sprintf();
    int result = strcmp(str_daddr, str_ip); // 검증
    if (result == 0) {
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    } else {
        printf("daddr is not equal to %s₩n",str_ip);
    }
}
```

```c
else if (control.type == 1)
{
    data = (struct L1 *)L2_receive(length);
    /*
    printf("receive my IP --> ");
    for (int i = 0; i < 4; i++)
    {
        printf("%hhu", data->daddr[i]);
        if (i != 3)    printf(".");
    }
    printf("₩n");
    */
    char str_ip[16]; // my ip
    char str_daddr[16]; // receive ip
    sprintf(str_ip,"%hhu.%hhu.%hhu.%hhu",addr.ip[0],addr.ip[1],addr.ip[2],addr.ip[3]);
    sprintf(str_daddr,"%hhu.%hhu.%hhu.%hhu",data->daddr[0],data->daddr[1],data->daddr[2],data->daddr[3]);
    //printf("str_ip : %s str_daddr : %s₩n", str_ip, str_daddr);

    int result = strcmp(str_daddr, str_ip);
    if (result == 0) {
        //printf("daddr is equal to %s₩n",str_ip);
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L1_data;
    } else {
        printf("daddr is not equal to %s₩n",str_ip);
    }
}
```

sprintf함수 호출하여 문자열로 변환하고
strcmp함수 호출하여 문자열 비교 진행

dataComm_final.c

L2_receive

```c
if (control.type == 2)
{
    if (is_server == 1)
    {
        // server
        data = (struct L2 *)L3_receive(length);
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L2_data;
    }
    else
    {
        // client
        // 구현. 데이터 파싱과 addr.mac에 값 대입
        // data =   *length =     addrData =


        return (char *)data->L2_data;
    }
}
```

```c
if (control.type == 2)
{
    if (is_server == 1)
    {
        // server
        data = (struct L2 *)L3_receive(length);
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L2_data;
    }
    else
    {
        // client
        data = (struct L2 *)L3_receive(length);
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);

        addrData = (struct Addr *)((struct L1 *)data->L2_data)->L1_data;

        printf("[%s] your MAC --> ", __func__);
        for (int i = 0; i < sizeof(addrData->mac); i++)
        {
            printf("%02X", addrData->mac[i]);
            addr.mac[i] = addrData->mac[i];
            if (i != 5)
                printf(":");
        }
        printf("₩n");

        return (char *)data->L2_data;
    }
}
```

dataComm_final.c

L2_receive

```c
else if (control.type == 1)
{

    data = (struct L2 *)L3_receive(length);

    char mac[18]; // my ip
    char str_daddr[18]; // receive ip
    // 구현. L1_rev 참고하여 구현

    int result = strcmp(str_daddr, mac);
    if(result==0){
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L2_data;
    }else{
        printf("daddr is not equal to %s₩n",mac);
    }
}
```

```c
else if (control.type == 1)
{

    data = (struct L2 *)L3_receive(length);
    /*
    printf("receive my MAC --> ");
    for (int i = 0; i < 6; i++)
    {
        printf("%02X", data->daddr[i]);
        if (i != 5)    printf(":");
    }
    printf("₩n");
    */
    char mac[18]; // my ip
    char str_daddr[18]; // receive ip
    sprintf(mac,"%02X:%02X:%02X:%02X:%02X:%02X",addr.mac[0],addr.mac[1],addr.mac[2],addr.mac[3],addr.mac[4],addr.mac[5]);
    sprintf(str_daddr,"%02X:%02X:%02X:%02X:%02X:%02X",data->daddr[0],data->daddr[1],data->daddr[2],data->daddr[3],data->daddr[4],data->daddr[5]);
    //printf("mac : %s str_daddr : %s₩n", mac, str_daddr);
    int result = strcmp(str_daddr, mac);
    if(result==0){
        //printf("daddr is equal to %s₩n",mac);
        *length = *length - sizeof(data->daddr) - sizeof(data->length) - sizeof(data->saddr);
        return (char *)data->L2_data;
    }else{
        printf("daddr is not equal to %s₩n",mac);
    }
}
```

sprintf함수 호출하여 문자열로 변환하고
strcmp함수 호출하여 문자열 비교 진행

dataComm_final.c – version 2

dataComm_final.c 소스의 printf 주석은 디버깅할때 사용하였으며 학습을 위해 주석을 해제하고 데이터를 어떻게 사용하는지 확인하시면 됩니다.

추가로 첨부해드린 파일 addr_sender와 addr_receiver에 소스 설명을 포함하여 주석으로 적어드렸습니다. 해당 파일도 참고하시어 L1주소와 L2 주소를 어떤 방법으로 데이터를 통신하는지 확인하시면 good

*addr_sender와 receiver는 양방향 통신을 진행하지 않기 때문에 코드 구조는 다름

*Addr Struct를 사용하지 않고 입력값을 통해 검증을 진행하였기 때문에 코드 구조는 다름

```
lbh@lbh-VirtualBox:~/datacomm/week9$ ./a.out
Session 2 starting...
#######################################################
[READY]sndsock port: 8811, rcvsock port: 8810
[READY]IP: 127.0.0.1
#######################################################
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
1
Input my L1 address(IP) : 192.168.30.1
Input my dest L1 address(IP) : 192.168.0.1
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
2
Input my L2 address(MAC) : 11:11:11:11:11:11
Input my dest L2 address(MAC) : 22:22:22:22:22:22
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
3
message: hello
call--> [L1_send]
call--> [L2_send]
Sender_sequence  : 75
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
2
Input my L2 address(MAC) : 11:11:11:11:11:11
Input my dest L2 address(MAC) : 12:12:12:12:12:12
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
3
message: hello
call--> [L1_send]
call--> [L2_send]
Sender_sequence  : 29
===========Choise Menu============
1. Select L1 address
2. Select L2 address
3. Send Message
```

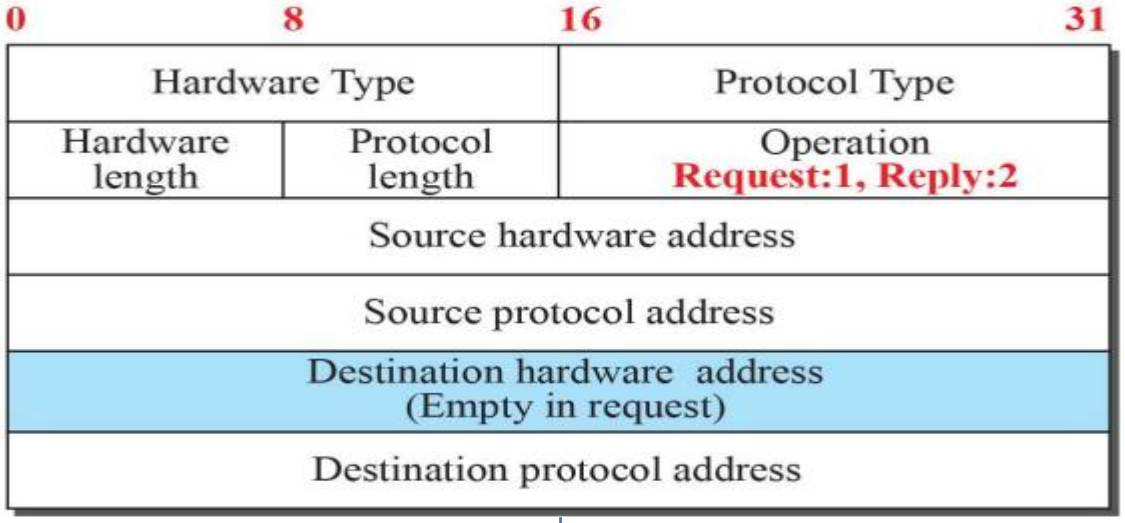
```
lbh@lbh-VirtualBox:~/datacomm/week9$ ./a.out
Session 1 starting...
#######################################################
[READY]sndsock port: 8810, rcvsock port: 8811
[READY]IP: 127.0.0.1
#######################################################
Your MAC[0]22
Your MAC[1]22
Your MAC[2]22
Your MAC[3]22
Your MAC[4]22
Your MAC[5]22
Expected_sequence  : 75
Your IP[0]192
Your IP[1]168
Your IP[2]0
Your IP[3]1
Length 6
Received: hello
Your MAC[0]12
Your MAC[1]12
Your MAC[2]12
Your MAC[3]12
Your MAC[4]12
Your MAC[5]12
Expected_sequence  : 29
Your IP[0]192
Your IP[1]168
Your IP[2]0
Your IP[3]1
L2 Address is Not Correct!!
```

ARP 패킷

ARP 헤더 선언과 나의 ARP 패킷 캡처해보기

```
lbh@lbh-VirtualBox:~/datacomm/week7$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.3                 ether   08:00:27:d9:ec:46   C                     enp0s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
lbh@lbh-VirtualBox:~/datacomm/week7$ arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.3                 ether   08:00:27:d9:ec:46   C                     enp0s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
Entries: 2      Skipped: 0      Found: 2
lbh@lbh-VirtualBox:~/datacomm/week7$ arp -a
? (10.0.2.3) at 08:00:27:d9:ec:46 [ether] on enp0s3
_gateway (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
```

명령어 $ arp –a 를 사용하여 저장되어 있는 ARP Cache 정보를 확인할 수 있음
명령어 $ arp –v / arp 를 사용하여 ARP 테이블 상태 정보를 확인할 수 있음



ARP 헤더 선언

```
struct arp_header {

    unsigned short hw_type; // 사용 가능한 전체 물리주소(MAC) 유형
    unsigned short protocl_type; // 사용중인 프로토콜 주소
    unsigned char hw_addr_len; // 패킷에 사용되는 MAC 길이
    unsigned char protocol_addr_len; // 패킷에 사용되는 IP 주소 길이
    unsigned short operation_code; // 요청(1) 또는 응답(2)
    struct ether_addr source_mac; // Sender L2 계층의 물리주소
    struct in_addr source_ip; // Sender L3 계층의 논리 주소
    struct ether_addr destination_mac; // Receiver L2 주소
    struct in_addr destination_ip; // Receiver L3 논리 주소

};
```

ARP 패킷

```
1 우분투
lbh@lbh-VirtualBox:~/datacomm/week7$ ls
lbh@lbh-VirtualBox:~/datacomm/week7$ vi test.c
lbh@lbh-VirtualBox:~/datacomm/week7$ gcc test.c
test.c:4:10: fatal error: pcap.h: No such file or directory
    4 | #include <pcap.h> // packet capture
      |          ^~~~~~~~~
compilation terminated.
```

CentOS : $ yum install libpcap libpcap-devel
Ubuntu : $ sudo apt-get install libpcap-dev

ARP 패킷캡처

```
lbh@lbh-VirtualBox:~/datacomm/week7/alpha$ sudo ./a.out
1. enp0s3 (No description available)
2. lo (No description available)
3. any (Pseudo-device that captures on all interfaces)
4. bluetooth-monitor (Bluetooth Linux Monitor)
5. nflog (Linux netfilter log (NFLOG) interface)
6. nfqueue (Linux netfilter queue (NFQUEUE) interface)
Enter the interface number (1-6):1

listening on enp0s3...

====== ARP packet ======

Src MAC : 08:00:27:27:e9:b1:
Dst MAC : 08:00:27:d9:ec:46:
 ******* request *******
 Sender IP : 10.0.2.6
 Target IP : 10.0.2.3
 pro_type : 2048, hw_addr_len : 6, protocol_addr_len : 4


====== ARP packet ======

Src MAC : 08:00:27:d9:ec:46:
Dst MAC : 08:00:27:27:e9:b1:
 ********  reply  ********
 Sender IP  : 10.0.2.3
 Sender MAC : 08:00:27:d9:ec:46:
 Target IP  : 10.0.2.6
 Target MAC : 08:00:27:d9:ec:46:
pro_type : 2048, hw_addr_len : 6, protocol_addr_len : 4
```

```
struct L1 {
    int saddr[6];
    int daddr[6];
    int length;
    char L1_data[MAX_SIZE];
};

struct L2 {
    int saddr[4];
    int daddr[4];
    int length;
    char L2_data[MAX_SIZE];
};
```

wow... MAC[6], IP[4]

컴파일 명령어 : $ gcc test.c -lpcap
프로그램 실행 : $ sudo ./a.out
packet 확인 : 1 번 입력 후 엔터

ARP 메시지 프로토콜

Sender 에서 IP주소에 매핑 되는 MAC 주소를 요청(Request)하면, Destination에서는 MAC 주소를, 요청한 Sender로 전송(Reply).

컴퓨터간 통신에는 실제적으로는 MAC 주소가 쓰임.

내 컴퓨터에서 111.222.333.444 IP로 통신할때는 간략하게 줄여서 아래와 같은 절차를 따름.

1. “해당 IP의 MAC주소가 뭐야?” 와 같은 메시지를 네트워크 대역으로 송신함.

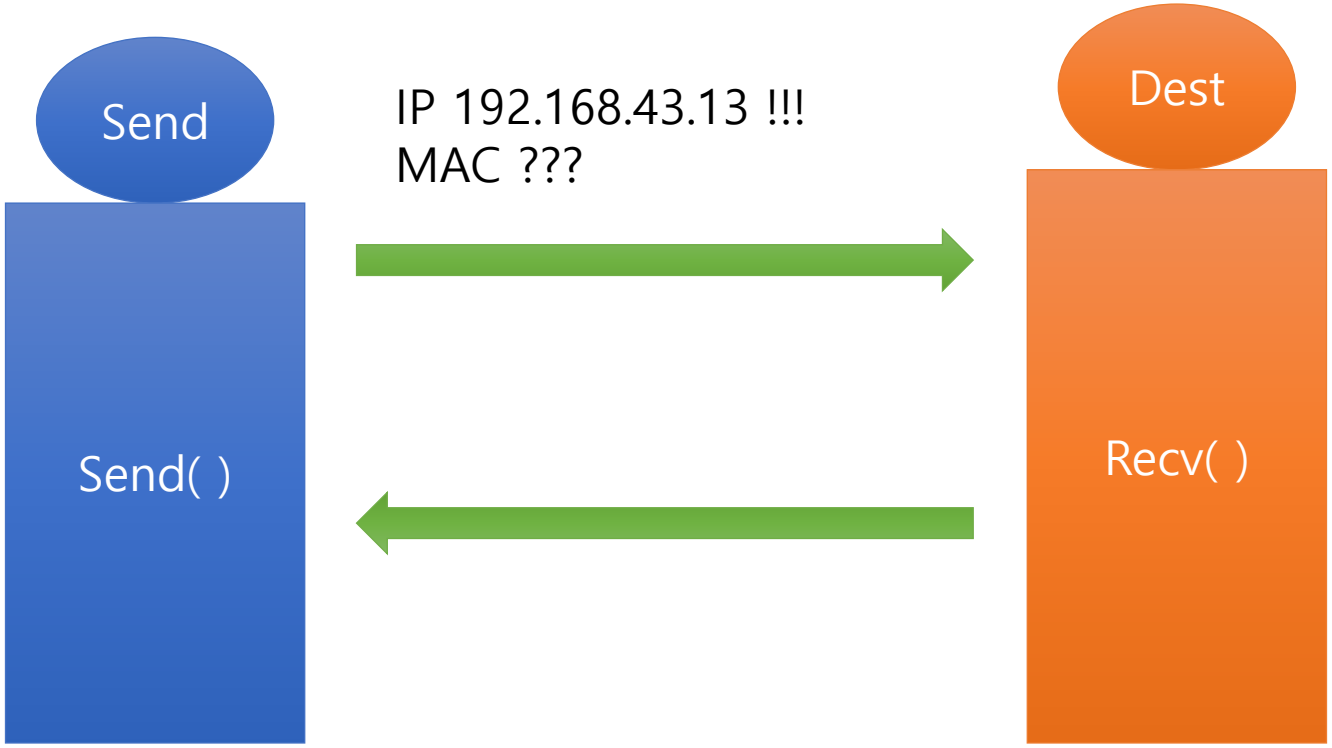
2. 같은 네트워크 대역에 있는 PC 중에서 해당되는 IP를 가진 PC가 "0A.77.00.11.2A.B3" 라고 답해줍니다.

3. 그럼 내 컴퓨터에서 “0A.77.00.11.2A.B3” 로 데이터를 전송합니다.

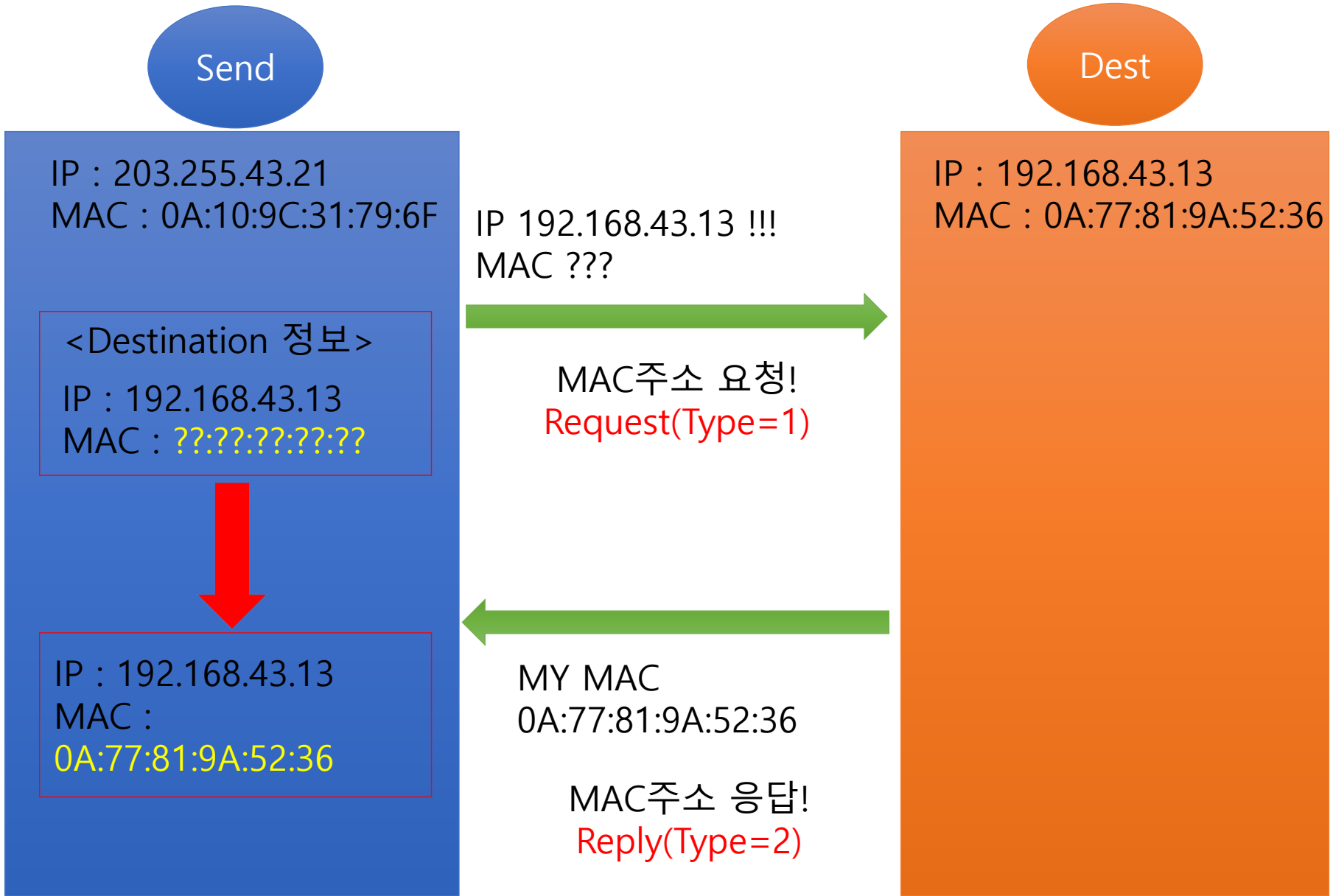
즉, 실제 개발자나 논리적인 생각상으로는 IP를 쓰지만, 컴퓨터 입장에서는 MAC주소로 통신하게 됨.

ARP 메시지 프로토콜

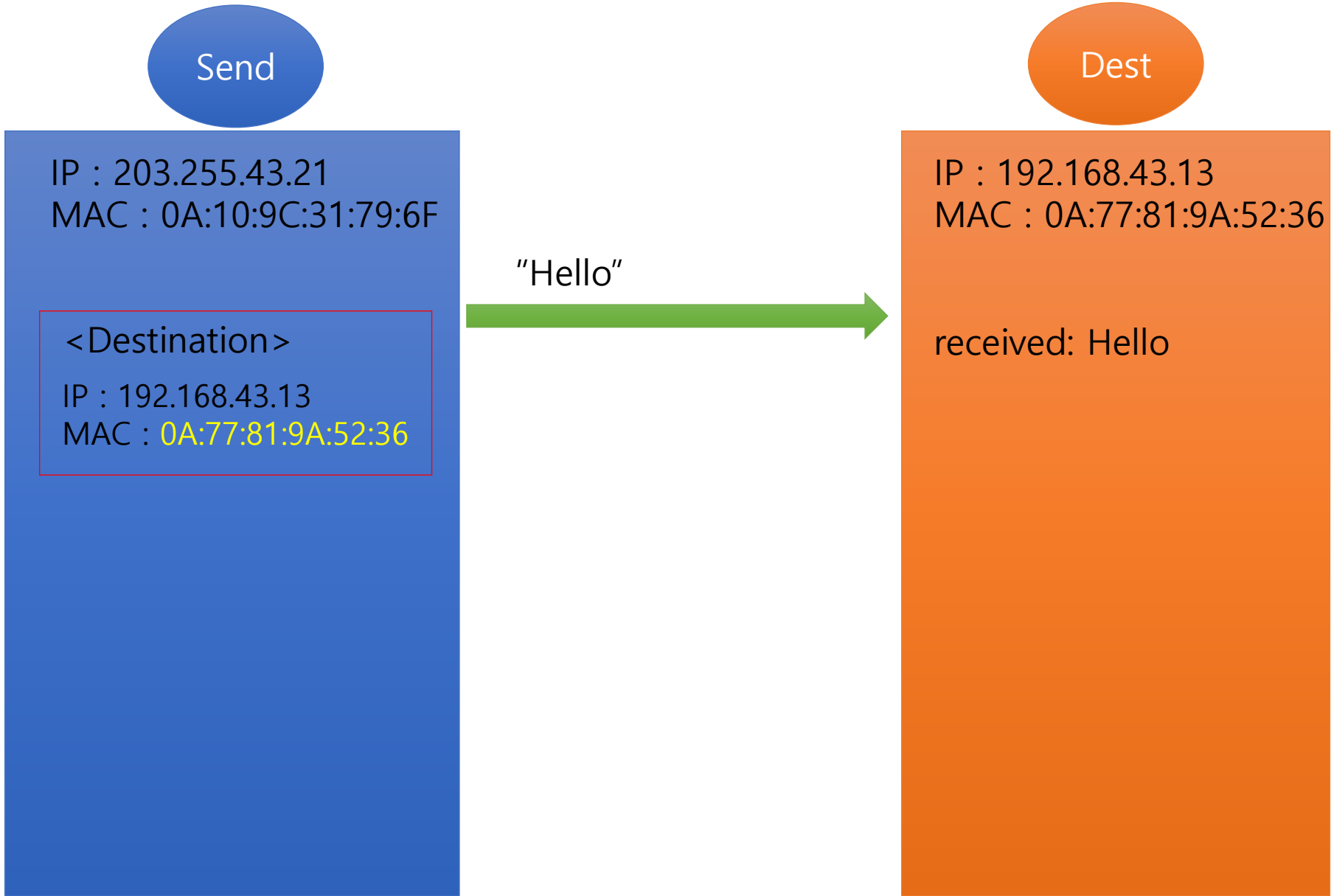
Sender 에서 IP주소에 매핑 되는 MAC 주소에 Request 하고
Destination에서는 MAC 주소를, 요청한 Sender에게 전송(Reply).



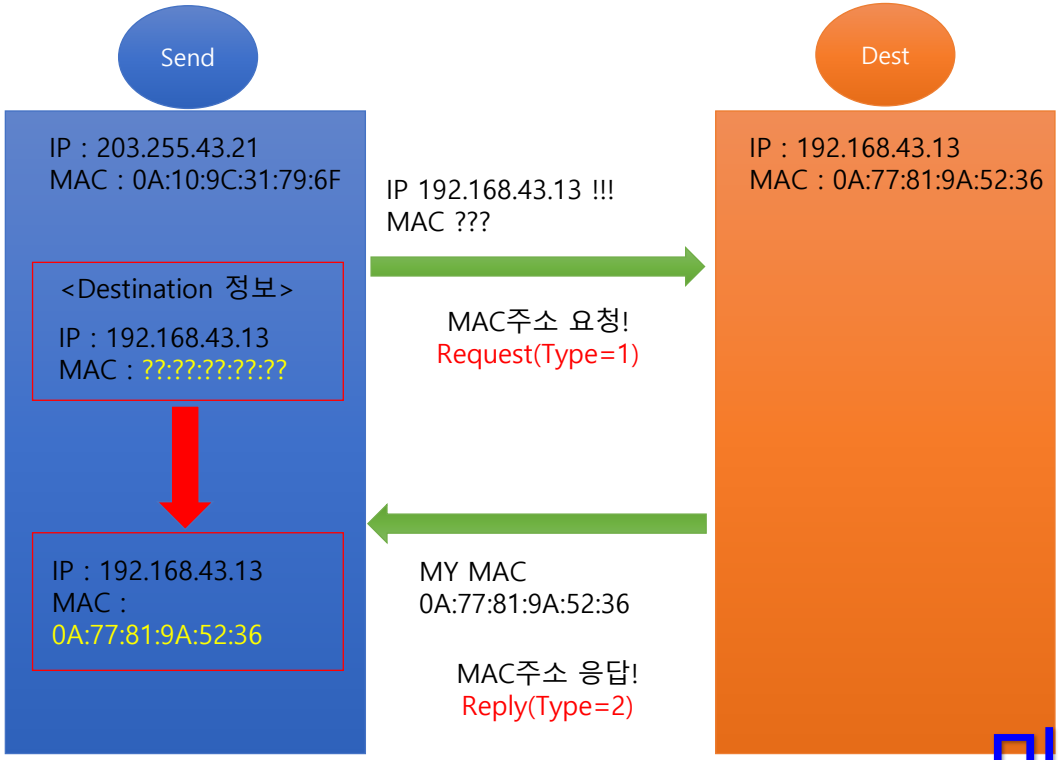
ARP 메시지 프로토콜


Send
Dest
IP : 203.255.43.21
MAC : 0A:10:9C:31:79:6F
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36
IP 192.168.43.13 !!!
MAC ???
<Destination 정보>
IP : 192.168.43.13
MAC : ??:??:??:??:??
MAC주소 요청!
Request(Type=1)
IP : 192.168.43.13
MAC :
0A:77:81:9A:52:36
MY MAC
0A:77:81:9A:52:36
MAC주소 응답!
Reply(Type=2)

ARP 메시지 프로토콜


Send
IP : 203.255.43.21
MAC : 0A:10:9C:31:79:6F
<Destination>
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36
"Hello"
Dest
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36
received: Hello

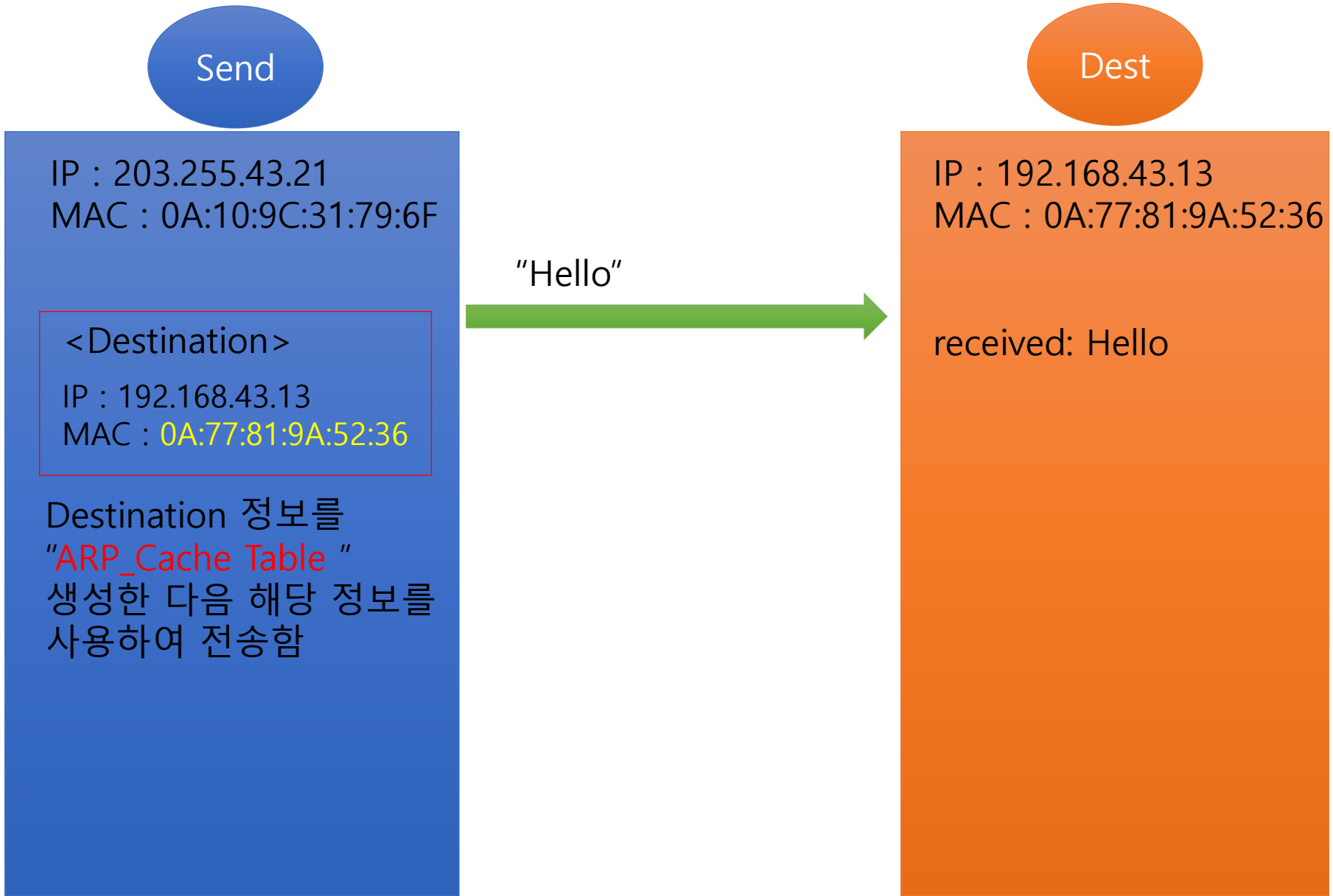
ARP 메시지 프로토콜



X 전송횟수

매번 메시지를 전송할때마다
같은 작업을 반복하는것은
비효율적임

ARP 메시지 프로토콜


Send
IP : 203.255.43.21
MAC : 0A:10:9C:31:79:6F
<Destination>
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36
Destination 정보를
"ARP_Cache Table "
생성한 다음 해당 정보를
사용하여 전송함
"Hello"
Dest
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36
received: Hello

ARP 메시지 프로토콜

Send

IP : 203.255.43.21
MAC : 0A:10:9C:31:79:6F

<ARP cache table>

IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36

Destination 정보를
"ARP cache table "
생성한 다음 해당 정보를
사용하여 전송함

ARP cache TB 확인후 전송
"Hello"

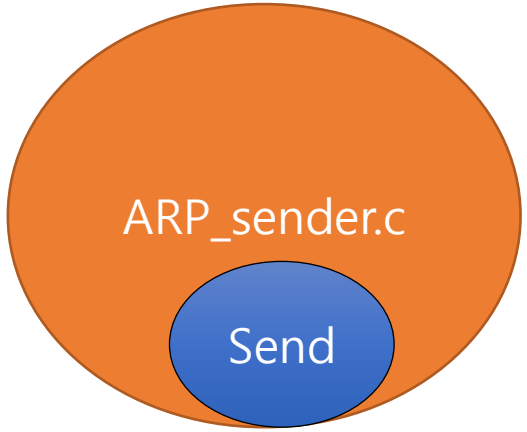
Dest

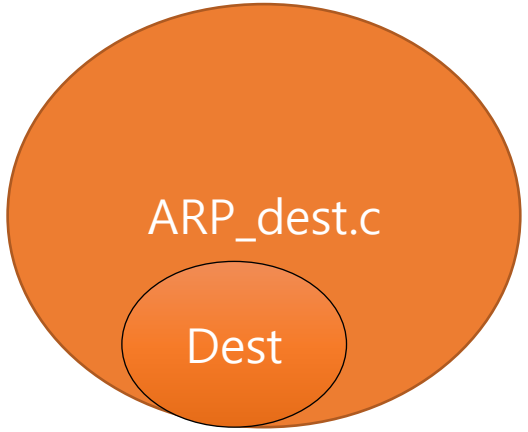
IP : 192.168.43.13
MAC : 0A:77:81:9A:52:36

received: Hello

ARP 메시지 프로토콜

이제 구현해봅시다.



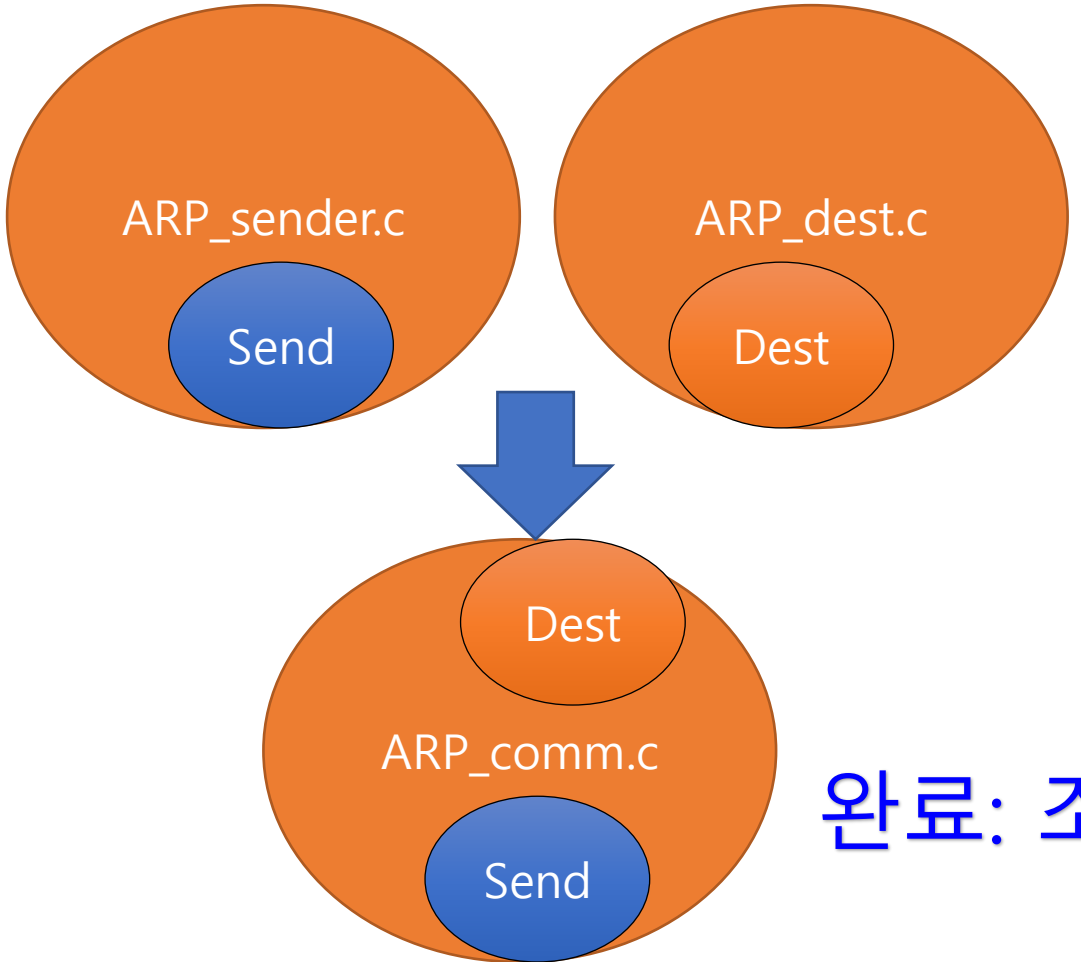


제공 소스파일: ARP_dest.c, ARP_sender.c

ARP 메시지 프로토콜

```
struct L1{
    char rec_MAC[20];
    char own_MAC[20];
    int length;
    int type;
    char L1_data[MAX_SIZE];
}; // 구조체 L1

struct L2{
    int rec_IP[4];
    int own_IP[4];
    int length;
    char L2_data[MAX_SIZE];
}; // 구조체 L2
```

1. L1, L2의 Struct 구조 변경 -> L1 IP, L2 MAC
2. sender.c + dest.c => ARP_comm.c



완료: 조교에게 확인

조교 확인

- addr_receiver.c/addr_sender.c 실행 캡처 화면, ARP 패킷 캡처 화면

```
lbh@lbh-VirtualBox:~/datacomm/week9$ ./a.out
Session 2 starting...
##################################################
[READY]sndsock port: 8811, rcvsock port: 8810
[READY]IP: 127.0.0.1
##################################################
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
1
Input my L1 address(IP) : 192.168.30.1
Input my dest L1 address(IP) : 192.168.0.1
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
2
Input my L2 address(MAC) : 11:11:11:11:11:11
Input my dest L2 address(MAC) : 22:22:22:22:22:22
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
3
message: hello
call--> [L1_send]
call--> [L2_send]
Sender_sequence  : 75
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
2
Input my L2 address(MAC) : 11:11:11:11:11:11
Input my dest L2 address(MAC) : 12:12:12:12:12:12
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
3
message: hello
call--> [L1_send]
call--> [L2_send]
Sender_sequence  : 29
==========Choise Menu==========
1. Select L1 address
2. Select L2 address
3. Send Message
```

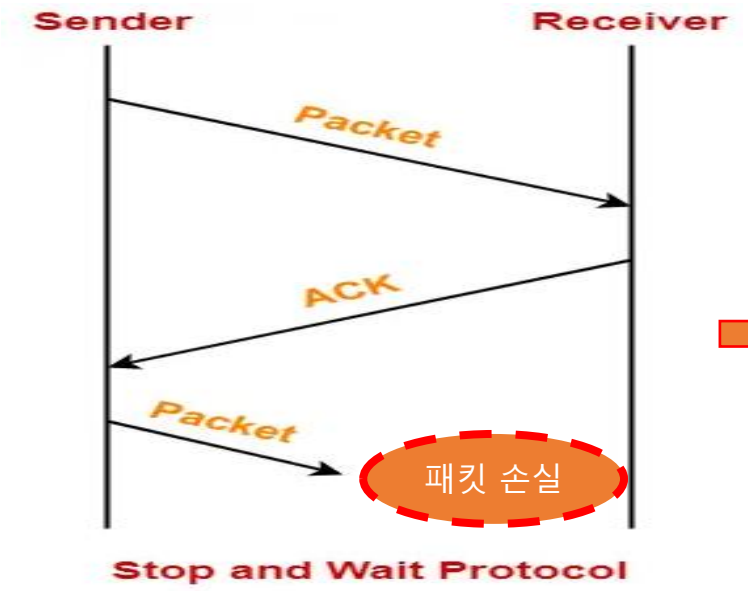
```
lbh@lbh-VirtualBox:~/datacomm/week9$ ./a.out
Session 1 starting...
##################################################
[READY]sndsock port: 8810, rcvsock port: 8811
[READY]IP: 127.0.0.1
##################################################
Your MAC[0]22
Your MAC[1]22
Your MAC[2]22
Your MAC[3]22
Your MAC[4]22
Your MAC[5]22
Expected_sequence  : 75
Your IP[0]192
Your IP[1]168
Your IP[2]0
Your IP[3]1
Length 6
Received: hello
Your MAC[0]12
Your MAC[1]12
Your MAC[2]12
Your MAC[3]12
Your MAC[4]12
Your MAC[5]12
Expected_sequence  : 29
Your IP[0]192
Your IP[1]168
Your IP[2]0
Your IP[3]1
L2 Address is Not Correct!!
```

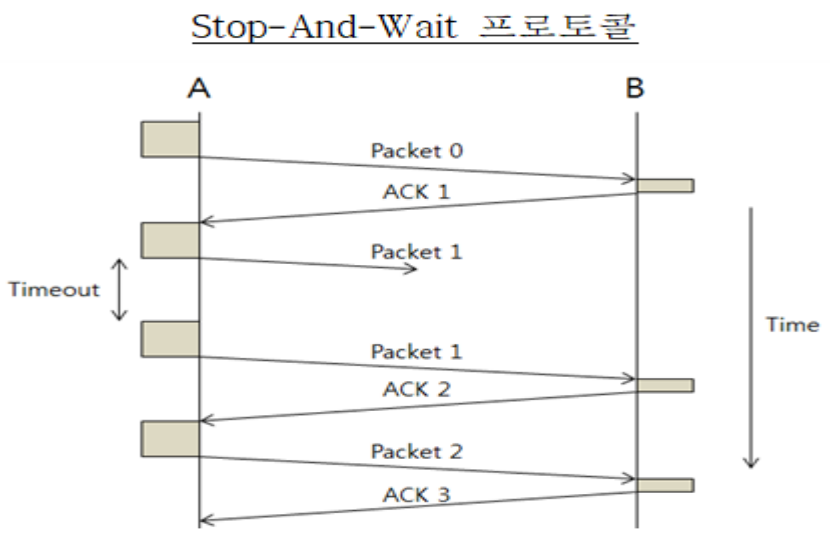
addr 캡처화면

```
lbh@lbh-VirtualBox:~/datacomm/week7/alpha$ sudo ./a.out
1. enp0s3 (No description available)
2. lo (No description available)
3. any (Pseudo-device that captures on all interfaces)
4. bluetooth-monitor (Bluetooth Linux Monitor)
5. nflog (Linux netfilter log (NFLOG) interface)
6. nfqueue (Linux netfilter queue (NFQUEUE) interface)
Enter the interface number (1-6):1

listening on enp0s3...

====== ARP packet ======

Src MAC : 08:00:27:27:e9:b1:
Dst MAC : 08:00:27:d9:ec:46:
 ******* request *******
 Sender IP : 10.0.2.6
 Target IP : 10.0.2.3
 pro_type : 2048, hw_addr_len : 6, protocol_addr_len : 4


====== ARP packet ======

Src MAC : 08:00:27:d9:ec:46:
Dst MAC : 08:00:27:27:e9:b1:
 ********  reply  ********
 Sender IP  : 10.0.2.3
 Sender MAC : 08:00:27:d9:ec:46:
 Target IP  : 10.0.2.6
 Target MAC : 08:00:27:d9:ec:46:
pro_type : 2048, hw_addr_len : 6, protocol_addr_len : 4
```

ARP 패킷 캡처화면

10주차 실습 목표





- Stop and wait 알고리즘 작성(L1,L2 구조 또는 새로운 코드 작성 예정)
- 데이터 전송시 stop and wait를 통하여 메시지 전송하도록 설계

  - 발신자는 하나의 데이터 패킷을 보낸 다음 승인을 기다림
  - 발신자는 이전 패킷에 대한 승인을 받은 후에만 다음 패킷을 전송함

11주차: Go Back N, 12주차~: HDLC 프로토콜 구현