

# 주요 프로토콜에 대한 이해 & 데이터 정렬 실습

인하공업전문대학 컴퓨터정보과  
최효현 교수

# 주요사항

- ❑ ICMP, ARP 등 주요 프로토콜
- ❑ DNS, DHCP, NAT 개념
- ❑ 주소 변환 및 바이트 순서 변환
- ❑ 인터넷 주소 초기화 template

# 제어 프로토콜

## □ ICMP

- ✓ 호스트 또는 라우터 사이에 오류 정보나 제어 정보를 전달하는데 사용
- ✓ Ping 같은 응용 프로그램이 직접 사용하기도 함

## □ IGMP

- ✓ 멀티캐스팅을 처리하는 프로토콜
- ✓ TCP/IP 프로토콜의 옵션 사항

## □ ARP

- ✓ LAN 내에서 특정 IP 주소를 가지고 있는 호스트의 MAC 주소를 알아내는 절차

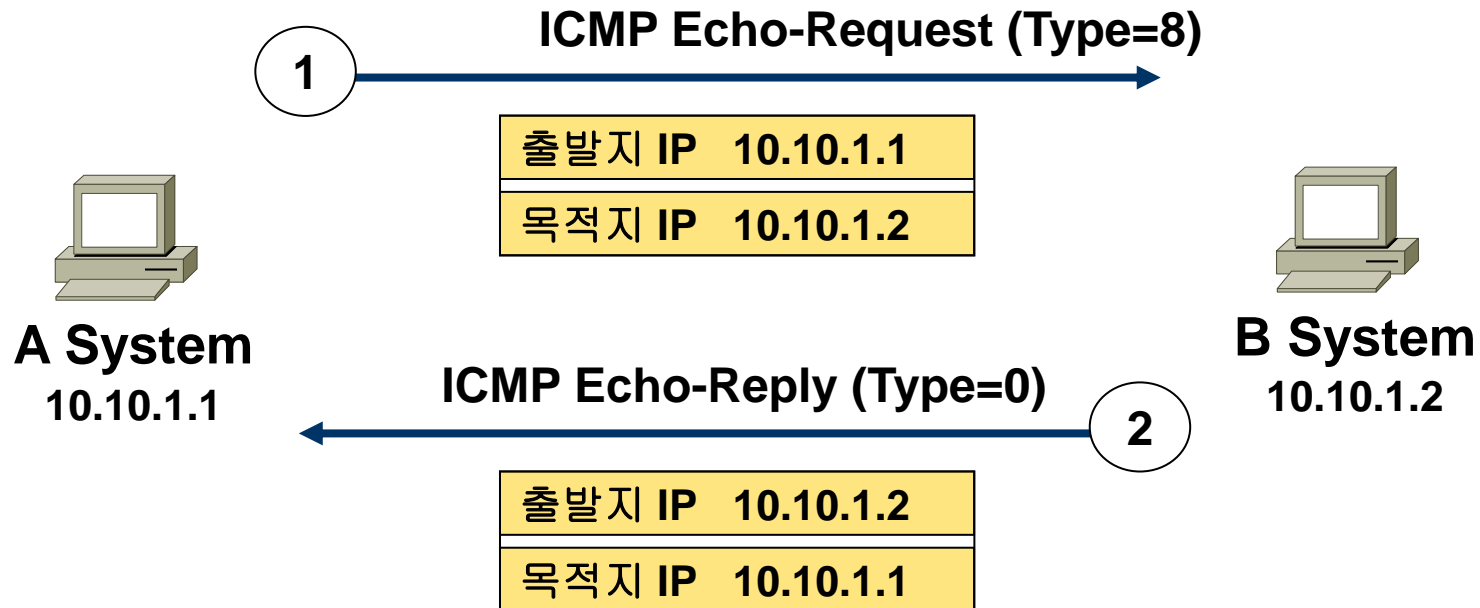
## □ RARP

- ✓ MAC 주소로부터 IP 주소를 알아내는 프로토콜

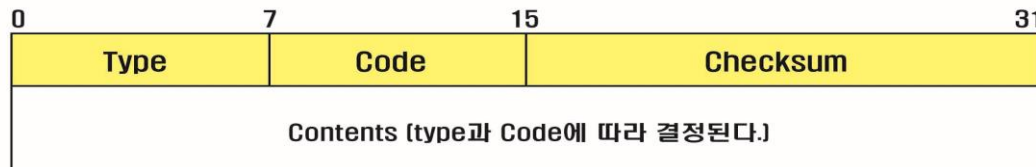


# ICMP(Internet Control Message Protocol)

**ICMP : PC나 라우터와 같은 기기의 동작 유무 확인 (예: ping)**



## ICMP 패킷 헤더



▶ ICMP 패킷 헤더의 구조

# ICMP(Internet Control Message Protocol)

1> Type(8bit)

2> Code(8bit)

3> Checksum(16bit)

4> Contents(32bit)

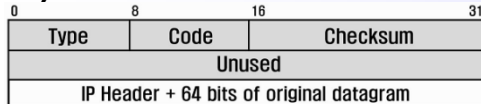
5> Data

6> 메시지 종류와 그에 따른 헤더 구조

가) Destination Unreachable

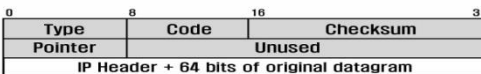
나) Time Exceeded

다) Source Quench



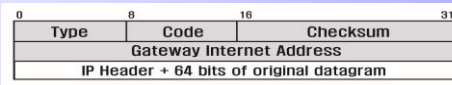
▶ Destination Unreachable,  
Source Quench, Time Exceeded

라) Parameter Problem



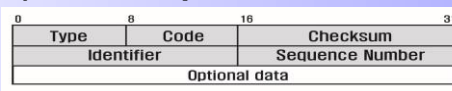
▶ Parameter Problem

마) Redirect



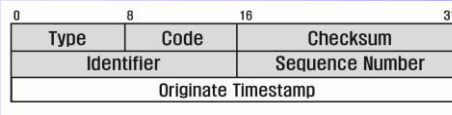
▶ Redirect

바) Echo Request and Echo Reply



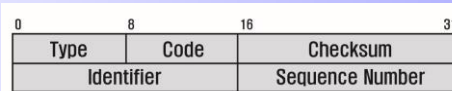
▶ Echo Request, Echo Reply

사) Timestamp Request and Reply

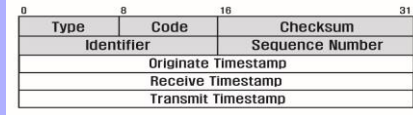


▶ Timestamp Request

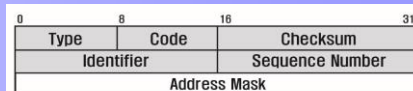
아) Address Mask Request Reply



▶ Address Mask Reply



▶ Timestamp Reply

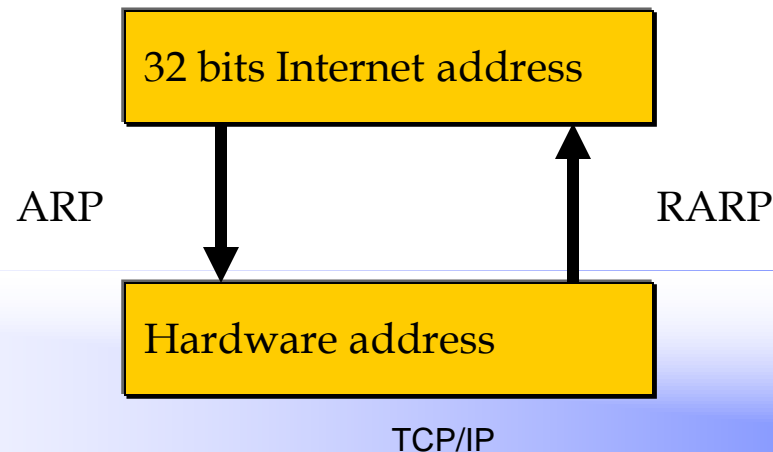


▶ Address Mask Reply

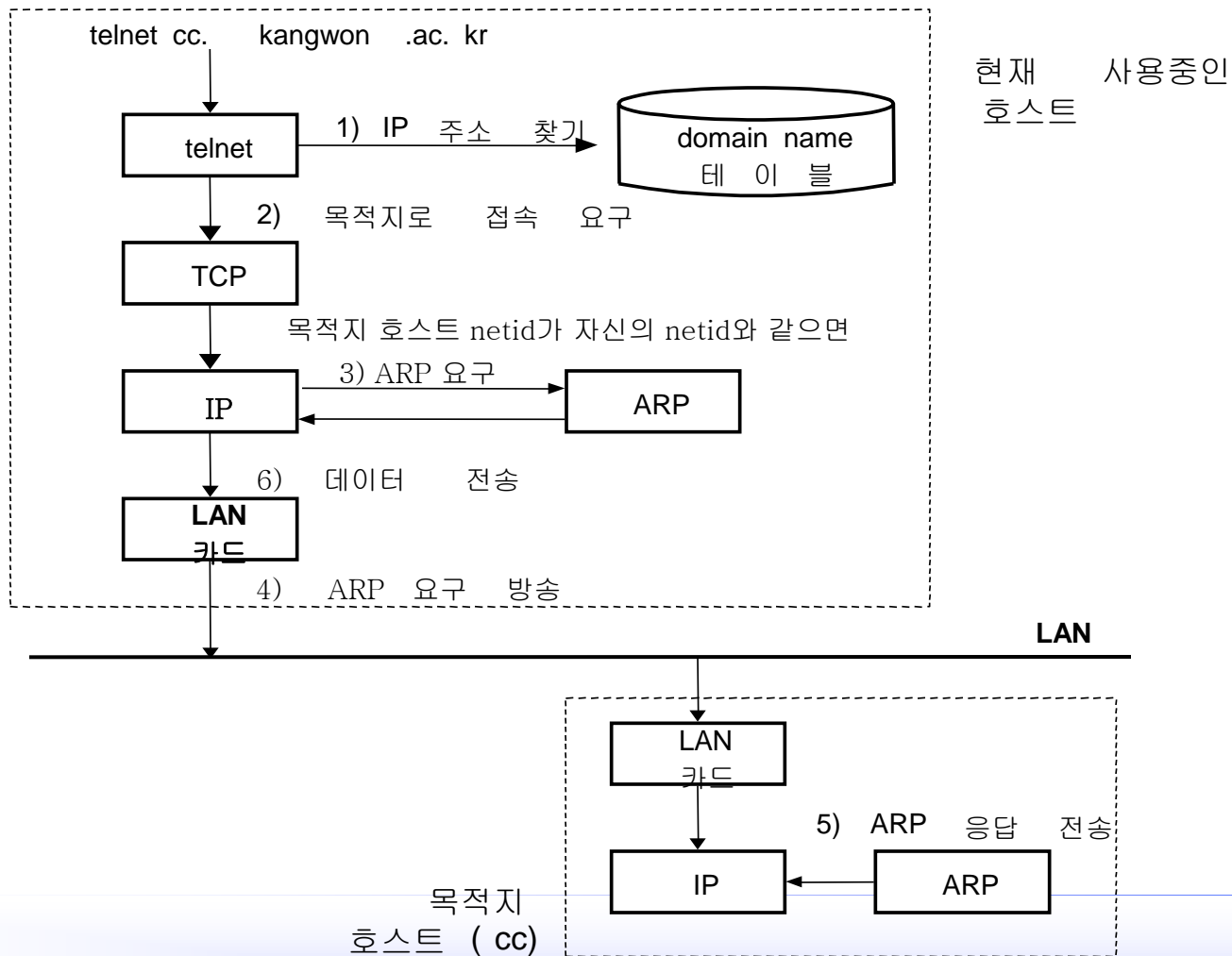
# ARP

## □ ARP(Address Resolution Protocol)

- ✓ 논리적 IP 주소와 하드웨어 MAC 주소의 mapping
- ✓ Ethernet, token ring과 같은 하드웨어는 고유의 하드웨어 주소를 가짐
- ✓ 네트워크 드라이버는 IP 주소로 프레임을 전송하지 않으며, IP 주소를 확인할 수도 없음
- ✓ Static mapping, Dynamic mapping
- ✓ RFC 826



# ARP의 동작 순서



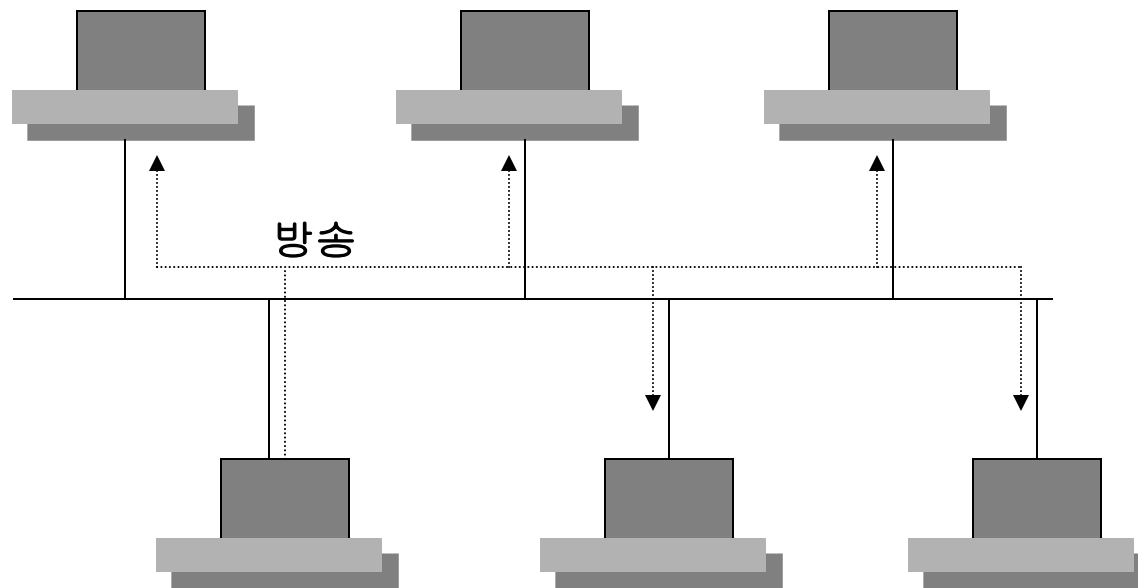
# RARP

## ❑ Reverse Address Resolution Protocol

- ✓ 특정 컴퓨터의 IP address를 알고자 할 때 필요
  - 모든 네트워크 시스템의 자신의 하드웨어 주소는 가지고 있음
  - 이를 이용하여 IP 주소를 알고자 할 때 활용
- ✓ ARP보다 복잡하고 구현하기 어려움
  - ARP는 IP주소와 하드웨어 주소를 운영체제가 알고 있음
  - RARP는 하드웨어 주소만 알고 있기 때문에 RARP 서버는 구현이 상대적으로 어려움

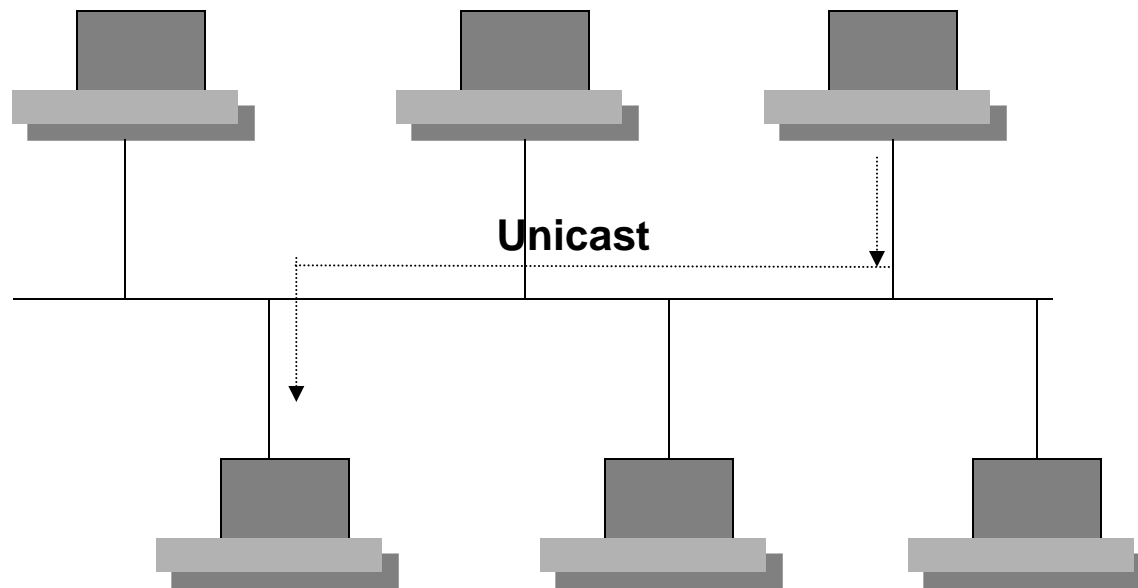


# ARP 프로토콜의 동작과정



송신지 시스템

# ARP 프로토콜의 응답과정



송신지 시스템

# DNS

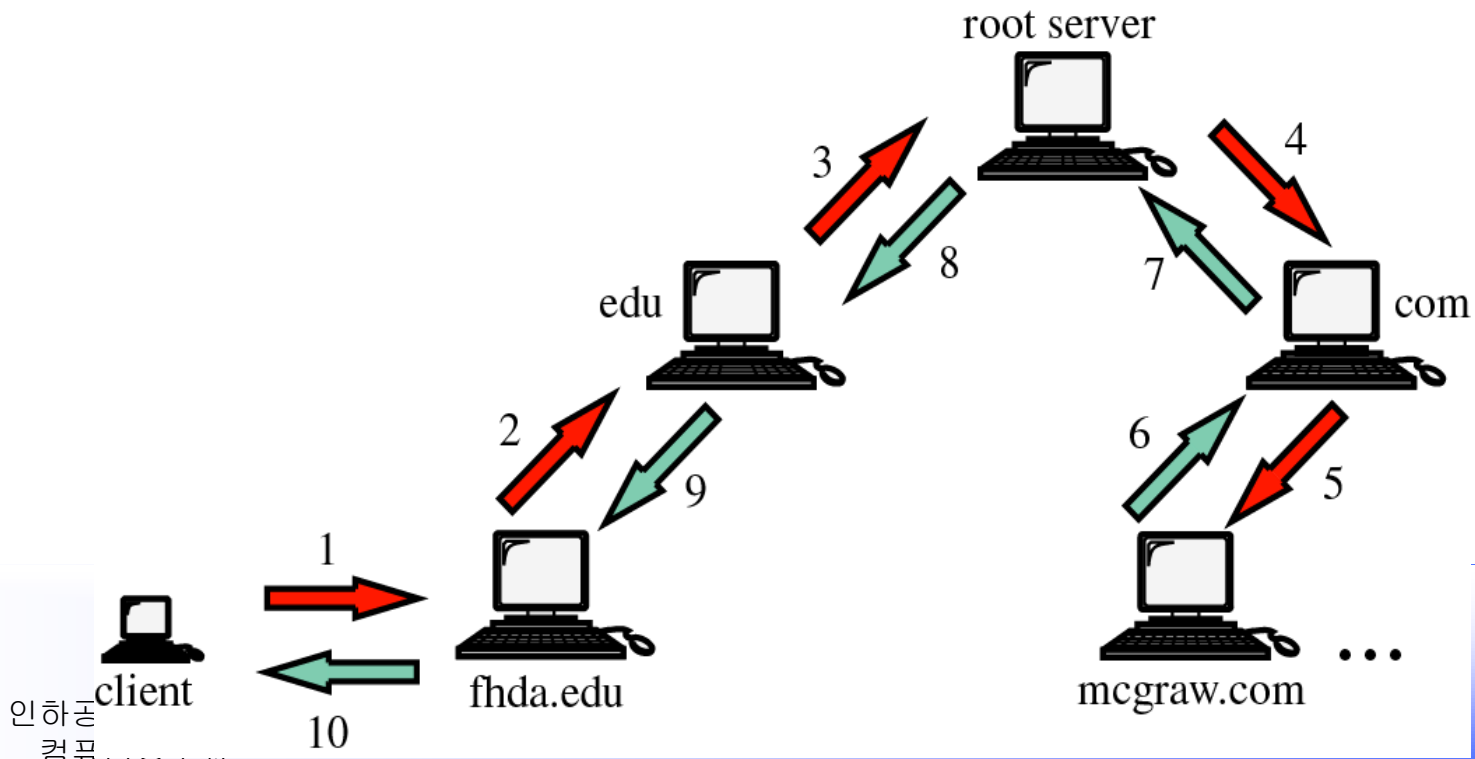
## □ DNS (Domain Name System)

- ✓ 도메인 이름과 IP 주소를 매핑 시켜주는 거대 규모의 분산 네이밍 시스템
- ✓ 인터넷에 존재하는 수많은 네임서버는 각각 도메인 계층상의 일부분을 관리하고, 정보를 요구하는 규칙에 따라 분산된 자료 중 원하는 정보를 찾을 수 있는 시스템

# DNS

## □ DNS 요청 : 순환(recursive) 쿼리 방식

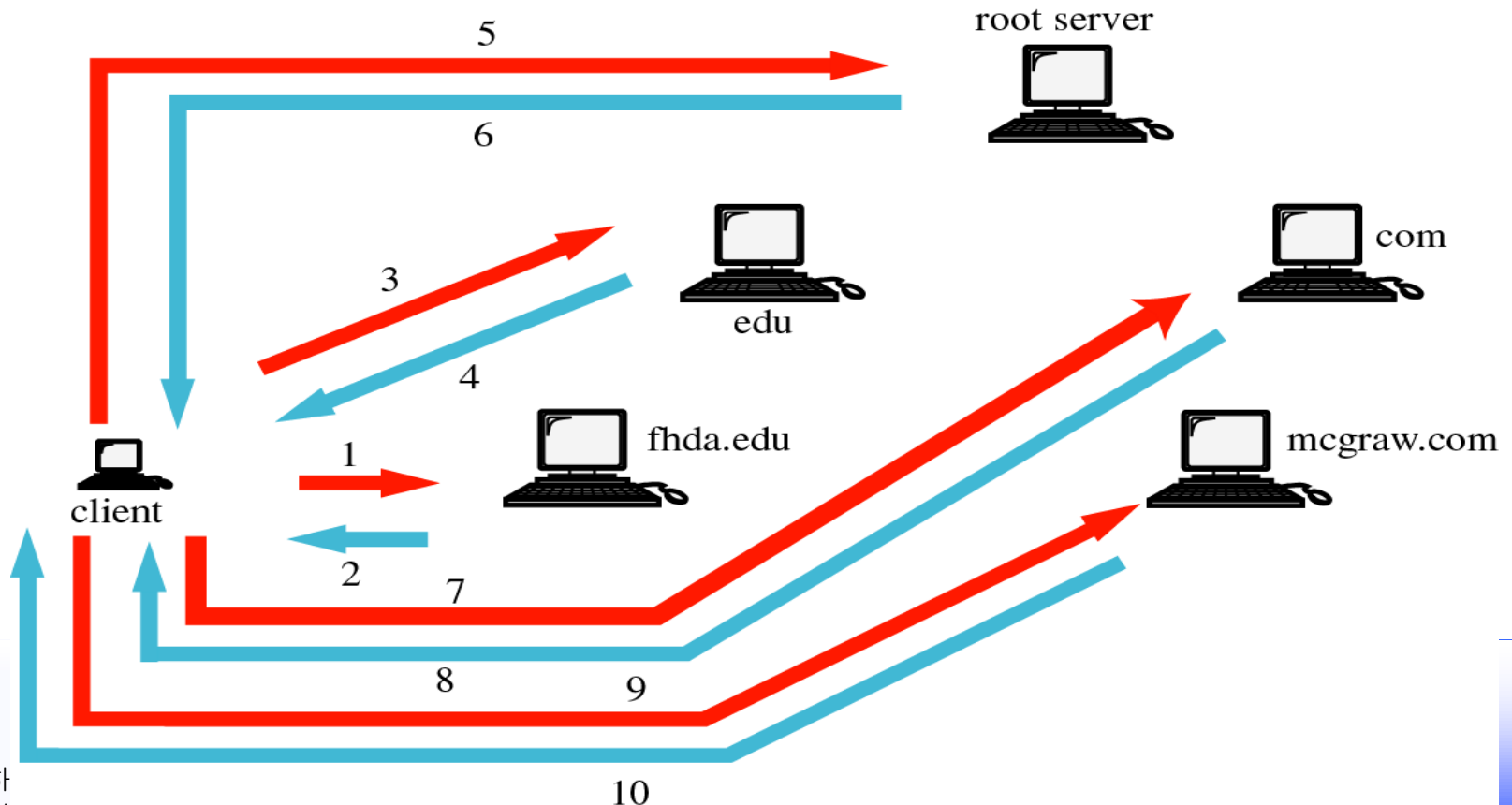
- ✓ 이름 분석을 시도하고 만일 서버가 요청한 정보를 갖고 있지 않다면 이에 대한 답을 자동으로 상위 서버에게 질의를 계속하여 전송하는 방식



# DNS

## □ DNS 요청 : 반복(iterative) 쿼리 방식

- ✓ 답을 얻을 때까지 상위 name server에게 직접 요청



# DHCP

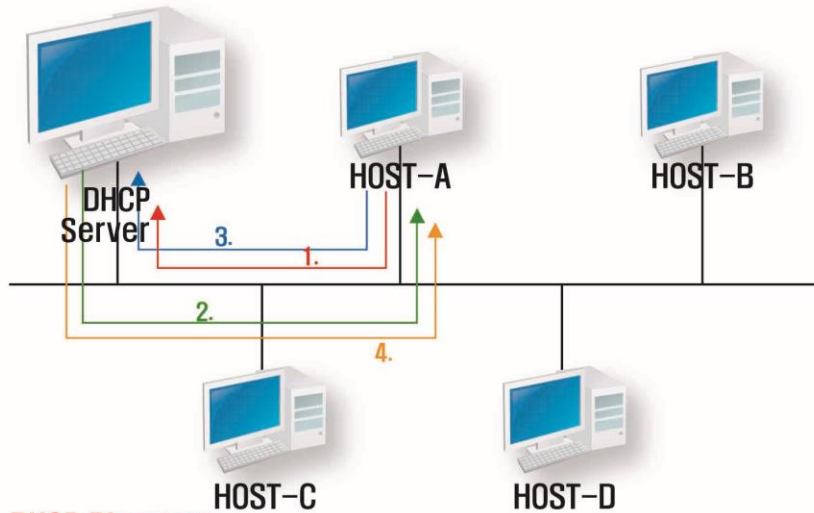
## ❑ DHCP (Dynamic Host Configuration Protocol)

- ✓ 네트워크를 구성하는 각각의 노드의 IP 관리를 쉽게 하기 위한 프로토콜

## ❑ 동적 할당

- ✓ 사용자가 TCP/IP 설정을 따로 해주지 않아도 되어서, 관리가 용이
- ✓ DHCP 기능을 다른 서브넷 상에도 적용하려면 라우터가 반드시 DHCP Relay Agent 역할을 수행.
- ✓ 관리하는 호스트 수가 많아지게 된다면 서버의 과부하 발생

# DHCP 동작 원리



1. DHCP Discover
2. DHCP Offer
3. DHCP Request
4. DHCP Acknowledgement

▶ DHCP 서버를 사용하는 네트워크의 동작 과정

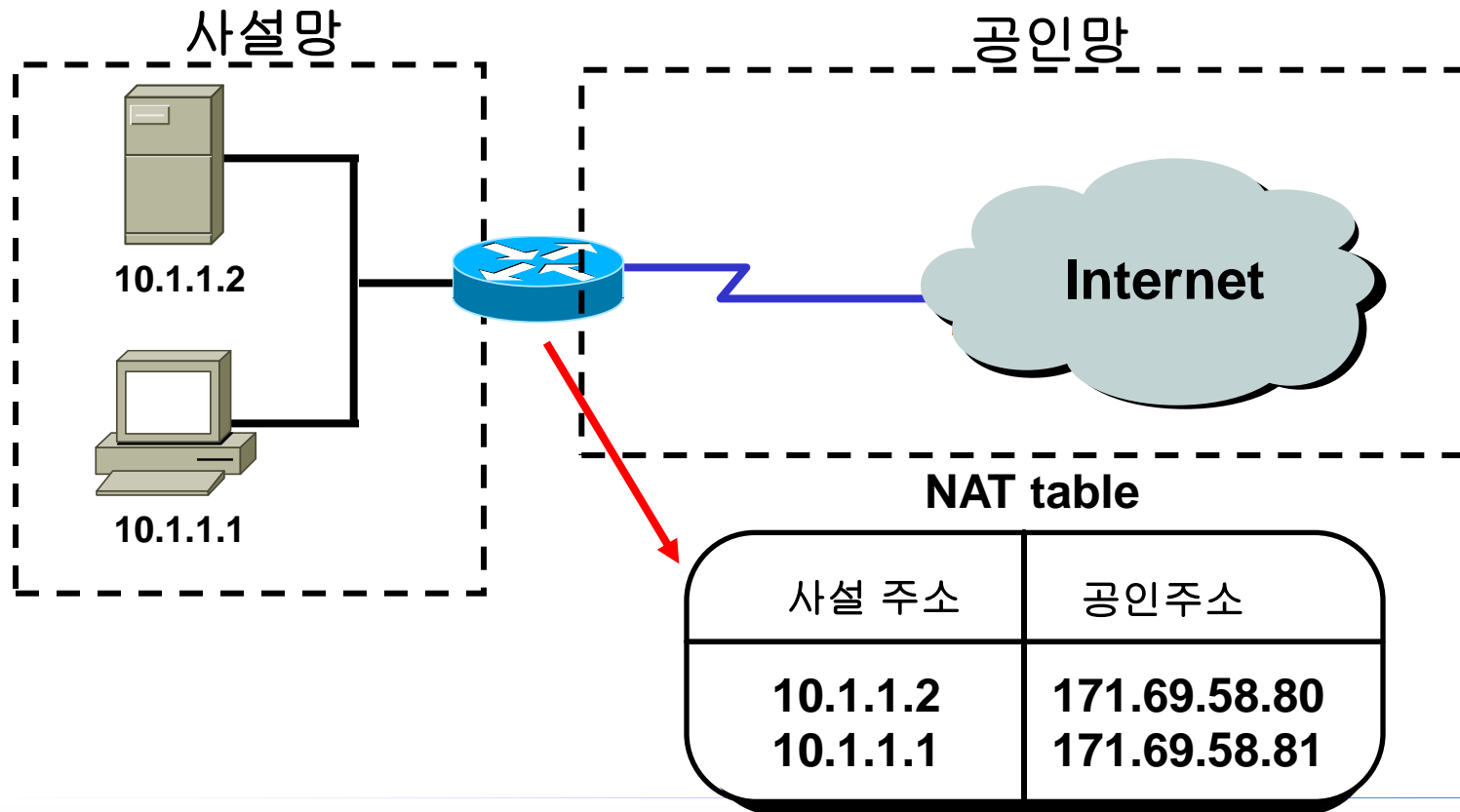
1. 최초 부팅 시 네트워크로 **DHCP Discover** 메시지를 브로드캐스팅 한다.
2. 해당 호스트가 이전에 사용했던 정보가 있을 경우 **IP 주소, Subnet Mask**등을 **DHCP Offer** 메시지를 호스트에게 전송한다.
3. 정보를 확인한 후 사용하겠다는 **DHCP Request** 메시지를 **DHCP** 서버에게 전송한다.
4. 클라이언트 정보를 자신의 테이블에 기록하고 허가 메시지인 **DHCP ACK** 메시지를 호스트에게 전송한다.

# NAT (Network Address Translation)

- ❑ NAT는 외부 네트워크에 알려진 것과 다른 IP 주소를 사용하는 내부 네트워크에서, IP 주소를 변환하는 것이다.
  - ✓ 공인 IP 주소를 다시 사설 IP 주소로 변환한다. 그 반대의 경우도 수행한다.
  - ✓ NAT는 IP 주소 고갈문제를 줄이기 위한 방법
  - ✓ 주소 변환과정을 반드시 거쳐야 하기 때문에, 보안에 도움이 됨

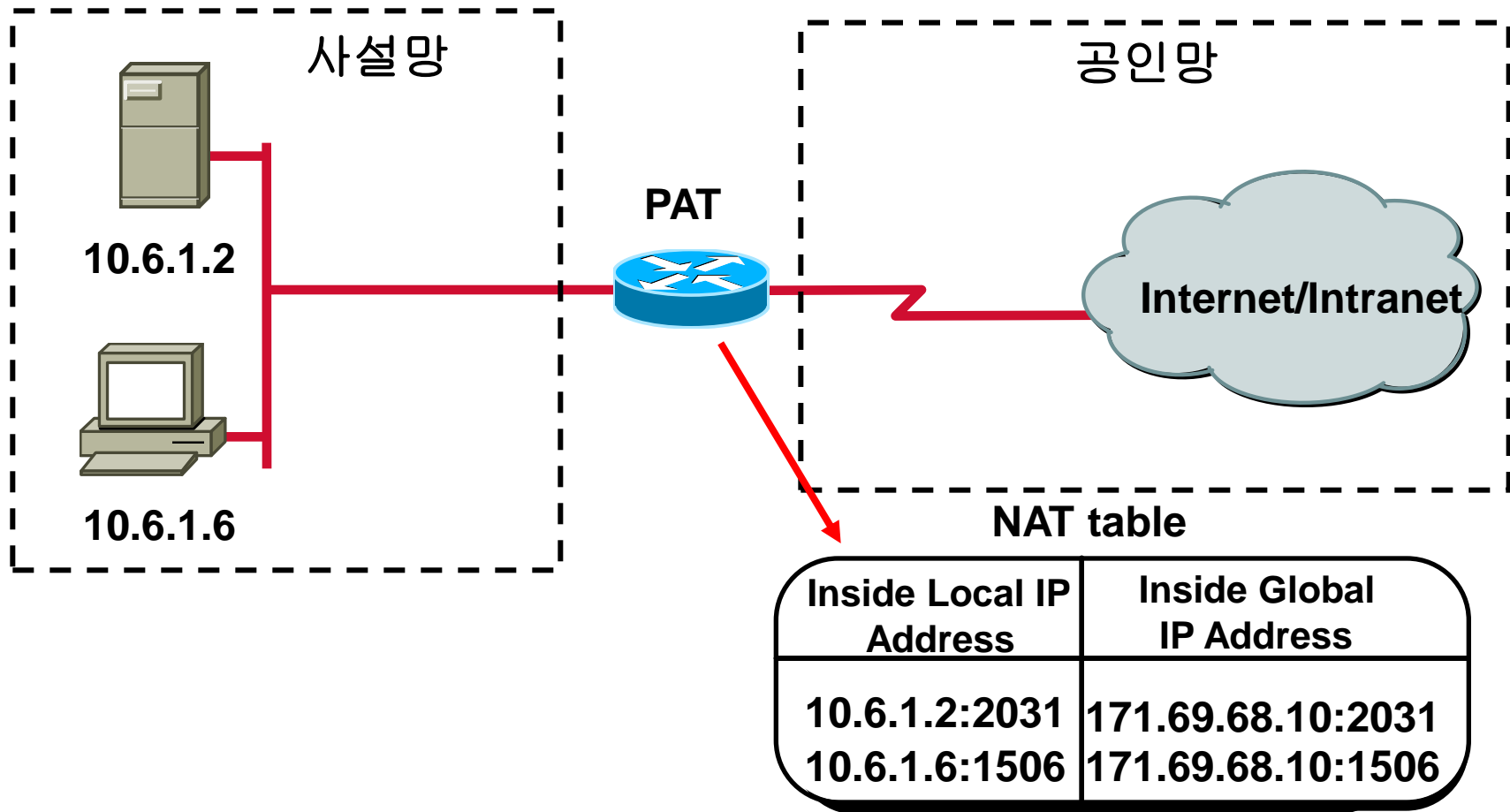


# NAT (Network Address Translation)



# NAT (Network Address Translation)

## Port Address Translation





## 3.4 인터넷주소 초기화

### • 인터넷 주소 조작 함수

<u>Domain name</u>	<u>dotted-decimal</u>	<u>network byte order</u> <u>바이너리</u>
(예 <a href="http://www.inhatc.ac.kr">www.inhatc.ac.kr</a> )	221.154.90.151	0x975a9add

→  
gethostbyname()

→  
inet\_addr() or,  
inet\_aton()

←  
gethostbyaddr()

←  
inet\_ntoa()

## 3.4 인터넷주소 초기화

- 인터넷 주소 조작 함수

- **sockaddr\_in** 구조체 주소는 **unsigned long** 타입

### 1. **FROM** Dotted-Decimal Notation **TO** Big-Endian 32 비트 정수형 데이터(Network)

```
unsigned long inet_addr(const char * string)
```

예) struct sockaddr\_in myaddr;  
char \*addr1= "1.2.3.4";  
conv\_addr = inet\_addr(addr1); // = 0x4030201  
myaddr.sin\_addr.s\_addr = conv\_addr; // 대입절차 필요  
printf(" %x", myaddr.sin\_addr.s\_addr)

인자값으로 x.x.x.x 문자열 포인터

```
unsigned long inet_aton(addr, struct in_addr addr)
```

예) char \*addr2= "1.2.3.4";  
inet\_aton(addr2, &myaddr.sin\_addr); // 자동 대입  
printf(" %x", myaddr.sin\_addr.s\_addr)

## 3.4 인터넷주소 초기화

- 인터넷 주소 조작 함수

### 2. **FROM** Big-Endian 32 비트 정수형 데이터(Network) **TO** Dotted-Decimal Notation

```
char * inet_ntoa(struct in_addr addr)
```

→ 리턴형이 문자열의 포인터, 내부 **static** 버퍼에 저장

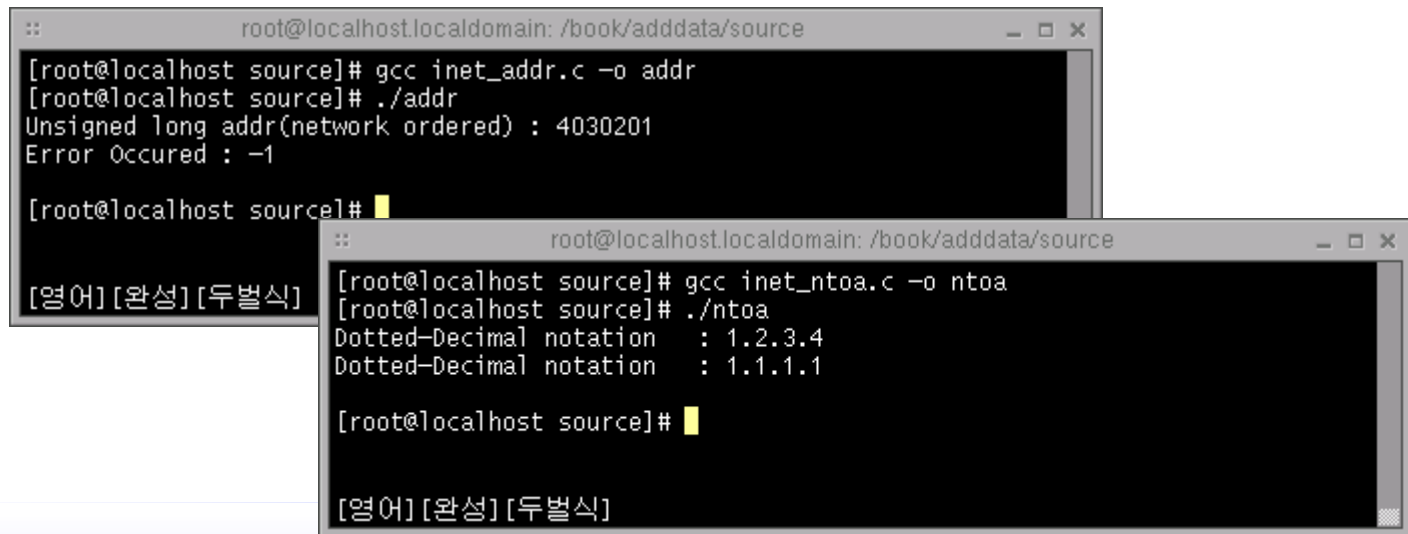
## 3.4 인터넷주소 초기화

- 예제 확인

1. 프로그램 예제

- inet\_addr.c, inet\_aton.c, inet\_ntoa.c

2. 실행결과.



```
root@localhost.localdomain: /book/adddata/source
[ root@localhost source ]# gcc inet_addr.c -o addr
[ root@localhost source ]# ./addr
Unsigned long addr(network ordered) : 4030201
Error Occured : -1

[ root@localhost source ]#

[영어] [완성] [두벌식]
```

```
root@localhost.localdomain: /book/adddata/source
[ root@localhost source ]# gcc inet_ntoa.c -o ntoa
[ root@localhost source ]# ./ntoa
Dotted-Decimal notation : 1.2.3.4
Dotted-Decimal notation : 1.1.1.1

[ root@localhost source ]#

[영어] [완성] [두벌식]
```

## 3.4 인터넷주소 초기화

### • 인터넷 주소 초기화 Template

```
1: struct sockaddr_in addr;  
2: char *serv_ip="211.217.168.13";  
3: char *serv_port="9190";
```

주소를 직접 넣어주는 것은  
다른 컴퓨터에서 실행시  
매번 바꿔주어야함

```
4: memset(&addr, 0, sizeof(addr));  
5: addr.sin_family = AF_INET;  
6: addr.sin_addr.s_addr = inet_addr(serv_ip); //스트링을 네트워크 바이트순서로 저장  
7: addr.sin_port = htons(atoi(serv_port)); //정수값으로 바꾸고, 다시  
네트워크 바이트순서로 저장
```

**memset(void \*p, int c, size\_t n)**

→ p가 가리키는 n 바이트를 상수 c로 채움



## 3.4 인터넷주소 초기화

### • 인터넷 주소 초기화 **Template**(클라이언트 쪽)

```
int main(int argc, char **argv)
```

```
1: struct sockaddr_in addr;
```

```
4: memset(&addr, 0, sizeof(addr));
```

```
5: addr.sin_family = AF_INET;
```

```
6: addr.sin_addr.s_addr = inet_addr(argv[1]);
```

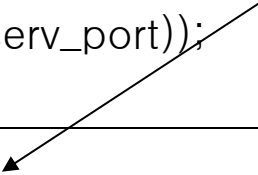
```
7: addr.sin_port = htons(atoi(argv[2]));
```

```
$> client 211.23.24.1 9190  
          argv[1]  argv[2]
```

## 3.4 인터넷주소 초기화

### • 인터넷 주소 초기화 Template(서버 쪽)

```
1: struct sockaddr_in addr;  
2: char *serv_port="9190";  
  
3: memset(&addr, 0, sizeof(addr));  
4: addr.sin_family = AF_INET;  
5: addr.sin_addr.s_addr = htonl(INADDR_ANY);  
6: addr.sin_port = htons(atoi(serv_port));
```



- 현재 시스템의 **IP**주소를 자동적으로 찾아서 할당해줌
- 하나의 시스템이 두 개 이상의 **IP**를 가지고 있을 때에도 유용함  
→ 두 **IP**로 오는 모든 패킷을 처리할 수 있음

## 3.4 인터넷주소 초기화

### • 인터넷 주소 초기화 **Template**(서버 쪽)

```
int main(int argc, char **argv)

1:  struct sockaddr_in addr;

3:  memset(&addr, 0, sizeof(addr));
4:  addr.sin_family = AF_INET;
5:  addr.sin_addr.s_addr = htonl(INADDR_ANY);
6:  addr.sin_port = htons(atoi(argv[1]));
```

## 3.4 인터넷주소 초기화

### • 주소 정보 할당하기

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int bind(int sockfd, struct sockaddr * myaddr, int addrlen);
```

→ 서버 IP주소 구조체의 포인터를 전달

- **sockfd** : 주소를 할당하고자 하는 소켓의 파일 디스크립터
- **myaddr** : 할당하고자 하는 주소 정보를 지니고 있는 **sockaddr\_in** 구조체 변수의 포인터를 인자로 전달
- **addrlen** : 인자로 전달된 주소정보 구조체의 길이
- 성공시 : **sockfd**가 가리키는 소켓에 **myaddr**이 가리키는 주소정보가 할당됨

예)

```
struct sockaddr_in serv_addr;  
bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))
```

# Q&A

