소켓 네트워크 프로그램 개발

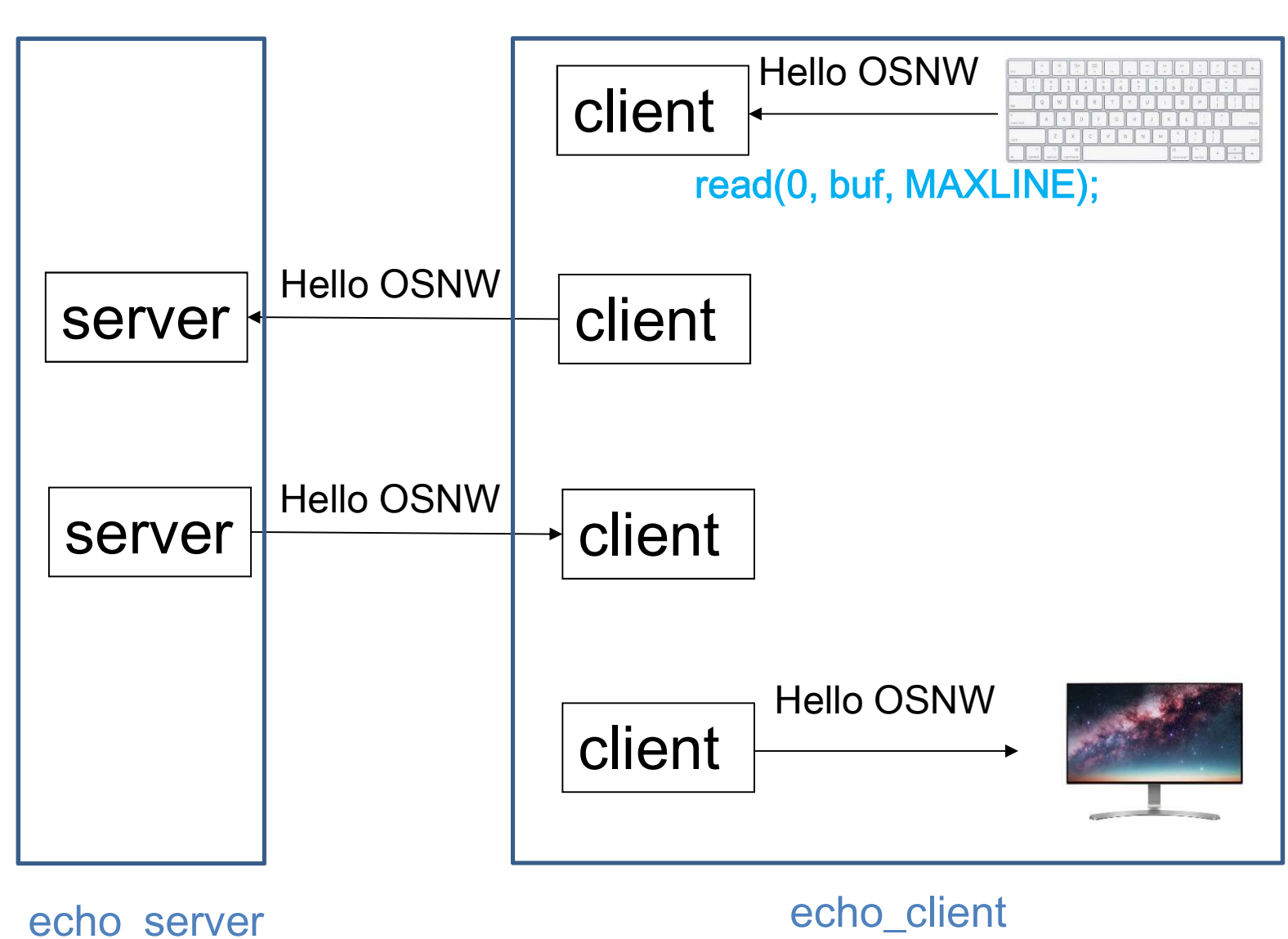
뇌를 자극하는 TCP/IP 소켓 프로그래밍

4장 네트워크 프로그램 개발

- 네트워프 프로그래밍에 대해서
- 소켓 다루기
- 서버/클라이언트 네트워크 프로그램 개발

echo_server.c and echo_client.c

Lab



echo_server.c – Linux version

```
#include <sys/socket.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>

#define MAXBUF 1024

int main(int argc, char **argv)
{
        int server_sockfd, client_sockfd, client_len, n;
        char buf[MAXBUF];
        struct sockaddr_in clientaddr, serveraddr;
        client_len = sizeof(clientaddr);
        if ( server_sockfd = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP ) ) == -1){
                perror("socket error : ");
                exit(0);
        }
        memset(&serveraddr, 0x00, sizeof(serveraddr));
        serveraddr.sin_family = AF_INET;
        serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
        serveraddr.sin_port = htons(atoi(argv[1]));     // input PORT NUMBER
```

structure

System call – network

echo_server.c – Linux version

```
        bind (server_sockfd, (struct sockaddr *)&serveraddr, sizeof(serveraddr));
        listen(server_sockfd, 5);

        while(1)
        {
                client_sockfd = accept(server_sockfd, (struct sockaddr *)&clientaddr,
                        &client_len);
                printf("New Client Connect: %s\n", inet_ntoa(clientaddr.sin_addr));
                memset(buf, 0x00, MAXBUF);
                if ((n = read(client_sockfd, buf, MAXBUF) ) <= 0) {
                        close(client_sockfd);
                        continue;
                }
                if (write(client_sockfd, buf, MAXBUF) <=0) {
                        perror("write error : ");
                        close(client_sockfd);
                }
                close(client_sockfd);
        }
        close(server_sockfd);
        return 0;
}
```

System call – general

echo_client.c – Linux version

```
#include <sys/socket.h>  /* 소켓 관련 함수*/
#include <arpa/inet.h>   /* 소켓 지원을 위한 각종 함수*/
#include <sys/stat.h>
#include <stdio.h>      /* 표준 입출력 관련*/
#include <string.h>     /* 문자열 관련*/
#include <unistd.h>     /* 각종 시스템 함수 */
#define MAXLINE    1024

int main(int argc, char **argv)
{
    struct sockaddr_in serveraddr;
    int server_sockfd,     int client_len;
    char buf[MAXLINE];

    if (( server_sockfd = socket(AF_INET, SOCK_STREAM, 0) ) == -1)  {
        perror("error :");
        return 1;
    }
    /* 연결요청할 서버의 주소와 포트번호 프로토콜등을 지정한다. */
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    serveraddr.sin_port = htons(3600);
```

echo_client.c – Linux version

```
client_len = sizeof(serveraddr);

/* 서버에 연결을 시도한다*/
if ( connect(server_sockfd, (struct sockaddr *)&serveraddr, client_len)  == -1) {
    perror("connect error :");
    return 1;
}

memset(buf, 0x00, MAXLINE);
read(0, buf, MAXLINE);                          /* 키보드 입력을 기다린다 */
if ( write(server_sockfd, buf, MAXLINE) <= 0) {    /* 입력 받은 데이터를 서버로 전송한다. */
    perror("write error : ");
    return 1;
}
memset(buf, 0x00, MAXLINE);
/* 서버로 부터 데이터를 읽는다 */
if ( read(server_sockfd, buf, MAXLINE) <= 0) {
    perror("read error : ");
    return 1;
}
printf("read : %s", buf);
close(server_sockfd);
return 0;
}
```

echo_server.c – Windows version

```
#include <winsock2.h>
#include <stdio.h>
#define MAX_PACKETLEN 1024

int main( )
{
    WSADATA wsaData;
    SOCKET listen_s, client_s;
    int nPort = 3500;
    struct sockaddr_in server_addr, client_addr;
    char szReceiveBuffer[MAX_PACKETLEN];
    int readn, writen;

    if( WSAStartup(MAKEWORD(2,2),&wsaData) != 0 )
            return 1;

    listen_s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(listen_s == INVALID_SOCKET)
            return 1;

    ZeroMemory(&server_addr, sizeof(struct sockaddr_in ));
```

Windows version

echo_server.c – Windows version

```
server_addr.sin_family = PF_INET;
server_addr.sin_port = htons(nPort);
server_addr.sin_addr.S_un.S_addr = htonl(INADDR_ANY);

if( bind(listen_s, (struct sockaddr*)&server_addr, sizeof(struct sockaddr_in) ) ==
      SOCKET_ERROR)
{
        return 0;
}
if( listen(listen_s, 5) == SOCKET_ERROR)
{
        return 0;
}

ZeroMemory(&client_addr, sizeof( struct sockaddr_in ));

int nAcceptClientInfo = sizeof(struct sockaddr_in);
client_s = accept(listen_s, (struct sockaddr*) &client_addr, &nAcceptClientInfo);
int nReceiveBytes = 0;
```

echo_server.c – Windows version

```
    while( 1 )
    {
            readn = recv(client_s, szReceiveBuffer, MAX_PACKETLEN,0);
            if( nReceiveBytes > 0 )
            {
                    writen = send(client_s, szReceiveBuffer, nReceiveBytes, 0);
            }
            closesocket(client_s);
    }

    closesocket(listen_s);
    WSACleanup();
    return 0;
}
```

echo_lient.c – Windows version

```
#include <winsock2.h>
#include <stdio.h>

#define PORT            3600
#define IP              "127.0.0.1"

int main( )
{
    WSADATA             WSAData;
    SOCKADDR_IN addr;
    SOCKET s;
    char buffer[1024];
    int readbytes, i, len;

    if ( WSAStartup(MAKEWORD(2,0), &WSAData) != 0)    {
            return 1;
    }
    s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (s == INVALID_SOCKET)  {
            return 1;
    }
```

echo_lient.c – Windows version

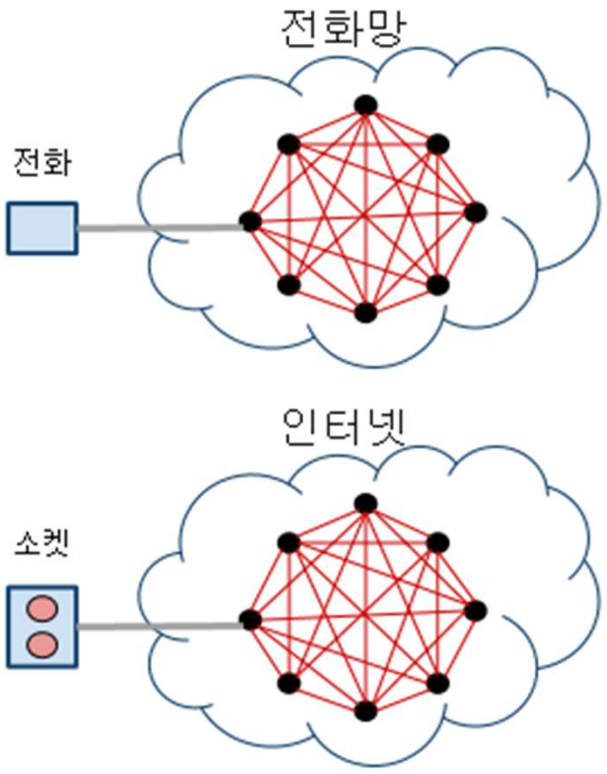
```
addr.sin_family = AF_INET;
addr.sin_port = htons(PORT);
addr.sin_addr.S_un.S_addr = inet_addr(IP);

if ( connect(s,(struct sockaddr *)&addr, sizeof(addr)) == SOCKET_ERROR) {
        printf("fail to connect\n");
        closesocket(s);
        return 1;
}
printf("enter messages\n");
for(i=0; 1; i++)  {
        buffer[i] = getchar( );
        if (buffer[i] == '\n') {
                buffer[i++] = '\0';
                break;
        }
}
len = i;
printf("send messages (%d bytes)\n", len);
send(s, buffer, len, 0);
```

echo_lient.c – Windows version

```
    for(readbytes=0; readbytes<len;)
            readbytes += recv(s, buffer+readbytes, len-readbytes, 0);

    printf("recv messages = %s\n", buffer);

    closesocket(s);
    WSACleanup( );
    return 0;
}
```

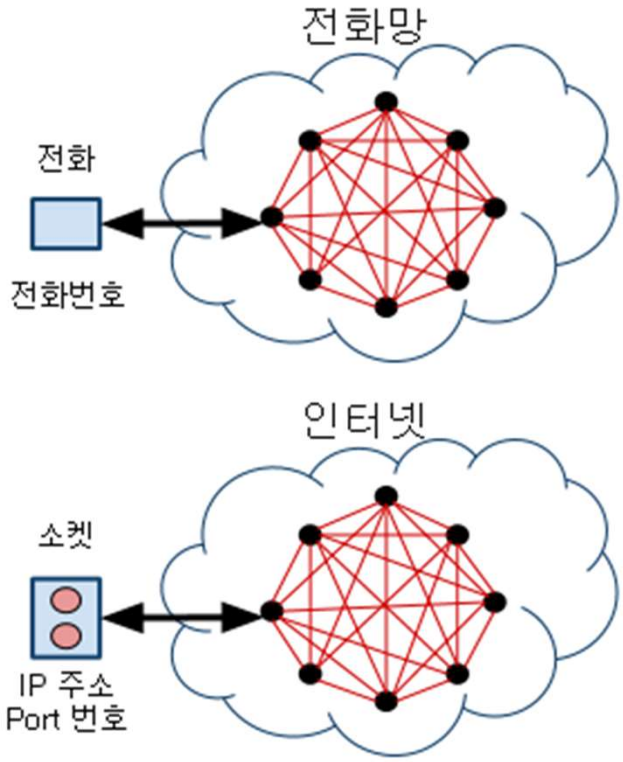
네트워크 프로그램의 흐름

- 전화망(PSTN: **Public switched telephone network**)과 동일한 흐름
- 소켓 : 전화기의 역할을 함. 인터넷과의 접점 혹은 관문



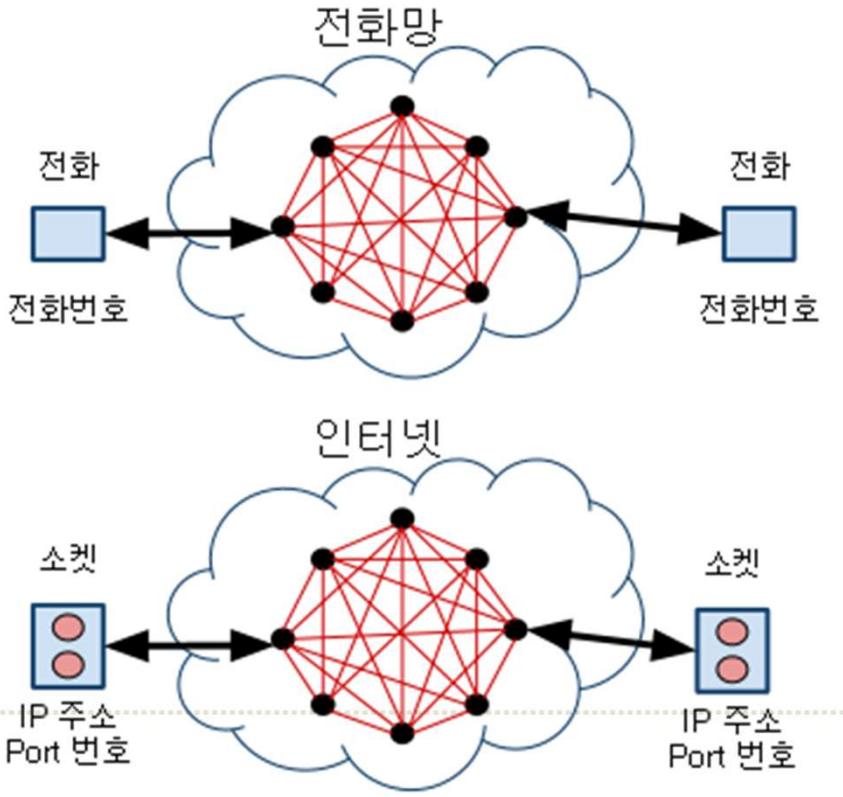
네트워크 프로그램의 흐름

- 송신자 Node를 인터넷에 연결하기
  - 전체 인터넷 노드에서 유일하게 식별가능한 **IP주소**를 소켓에 부여한다.
  - 프로그램을 찾을 수 있도록 **포트 번호**를 bind 한다.



네트워크 프로그램의 흐름

- 수신자 Node 찾기
  - 수신자 Node도 IP 주소와 Port 번호를 가지고 있다.
  - IP 주소를 이용해서 컴퓨터의 위치를 찾고
  - Port 번호를 이용해서 프로그램을 찾는다.
- 연결된 통신채널을 이용하여 데이터 송수신



네트워크 프로그램의 흐름

- 서버 측

  소켓 생성 → 포트 부여 → 상대편 연결 기다리기 → 통신 → 종료

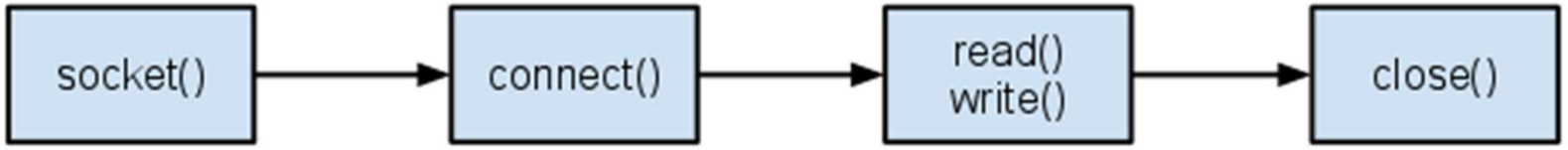
- 클라이언트 측

  소켓 생성 → 포트 부여 → 상대편 IP/Port 주소로 연결 → 통신 → 종료

클라이언트 네트워크 프로그램의 흐름

클라이언트 프로그램의 흐름

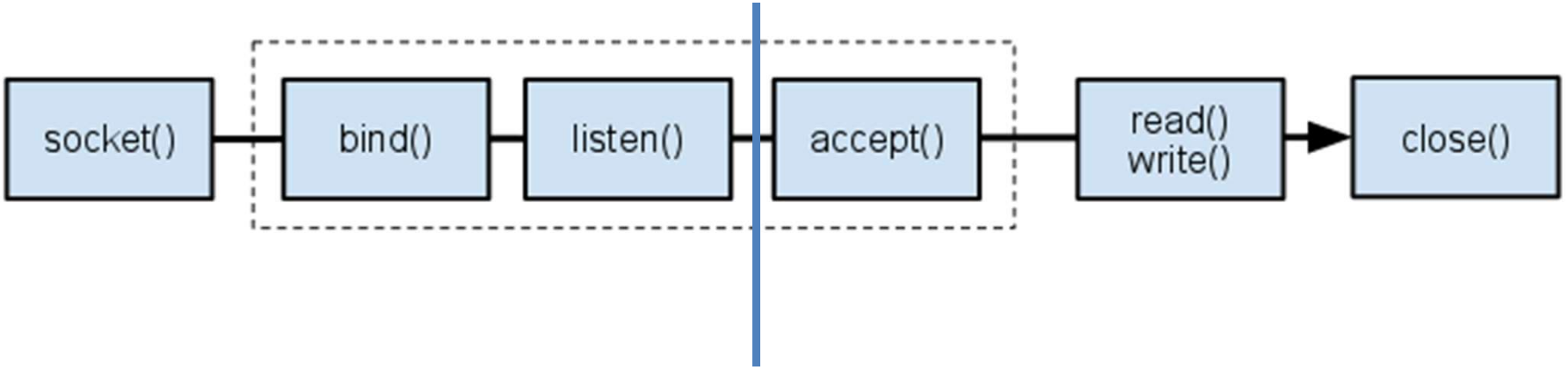
- socket() : 소켓 생성
- connect() : 연결 요청
- read()/write() : 데이터 통신
- close : 소켓 닫기



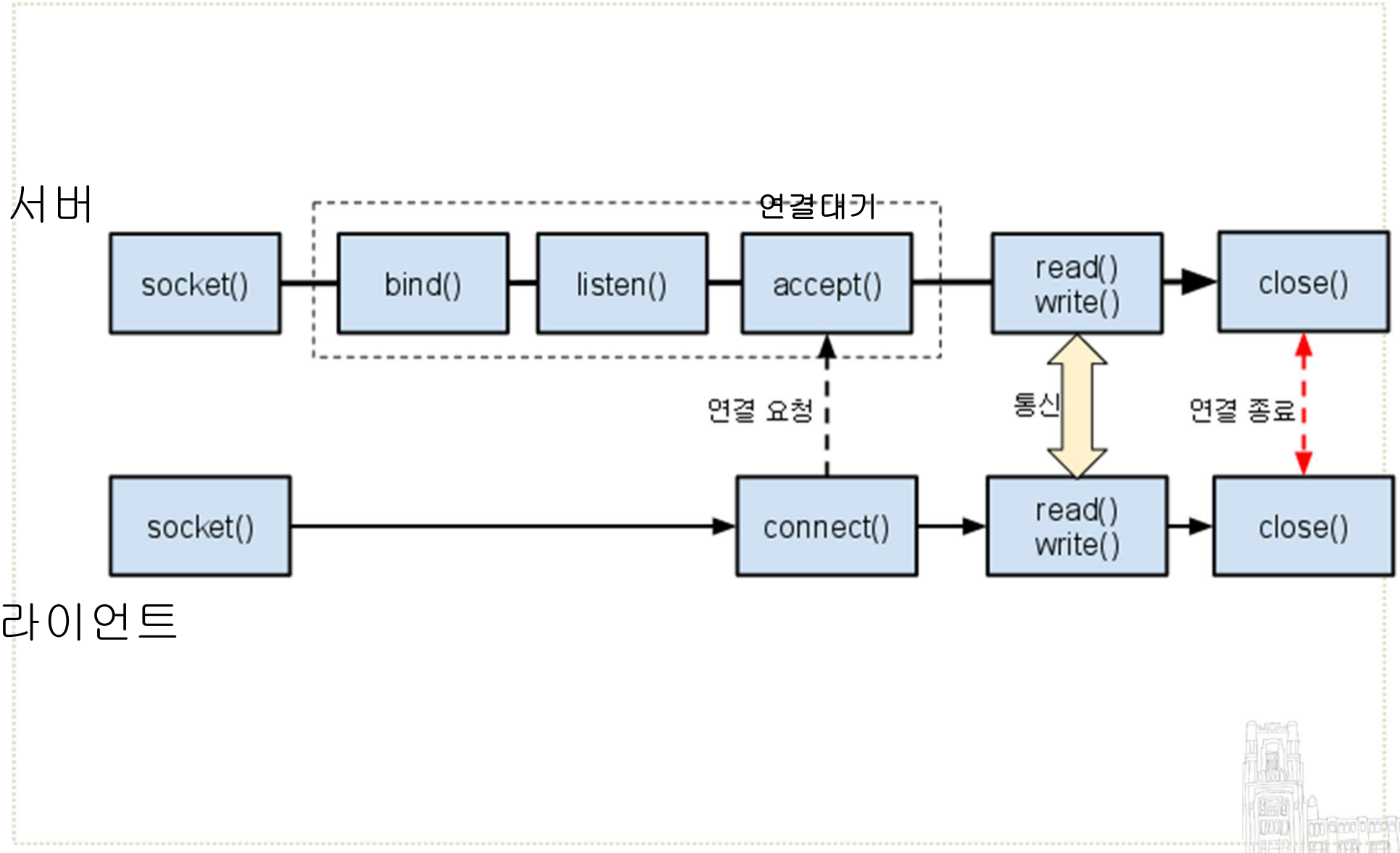
서버 네트워크 프로그램의 흐름

서버 프로그램의 흐름

- socket() : 소켓 생성
- bind() : 소켓을 인터넷 주소와 포트번호로 묶음
- listen() : 수신 대기열 생성 (listen queue)
- accept() : 연결 대기
- read()/write : 데이터 통신
- close() : 소켓 닫기



서버와 클라이언트 네트워크 프로그램 흐름



1. 소켓 생성

- 인터넷과 연결하기 위한 접점소켓(endpoint socket) 생성

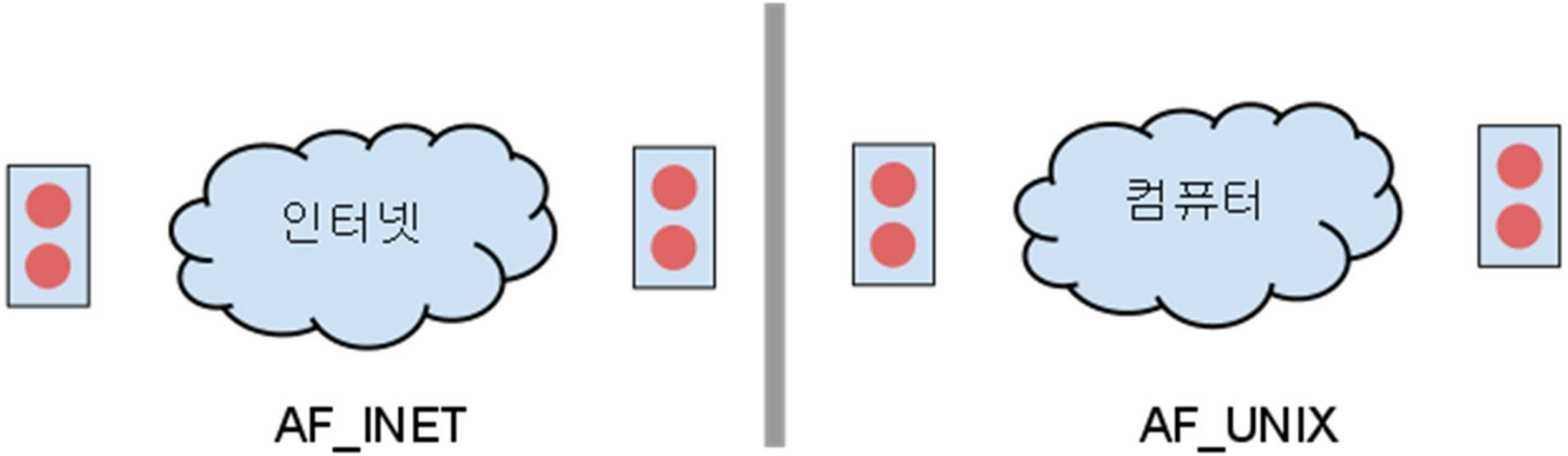
```
#include <sys/types.h>
#include <sys/socket.h>
int socket (int domain, int type, int protocol);
```

- domain : 소켓의 사용 영역을 정의한다.
- type : 소켓 유형을 정의한다.
- protocol : 소켓이 사용할 프로토콜을 정의한다.

A function declaration tells the compiler
about a function's name, return type, and parameters.

1. 소켓 생성

- domain : 소켓이 사용되는 네트워크의 영역을 정의



TYPE	설명
AF_UNIX	프로세스간 통신(IPC)용
AF_INET	**IPv4 TCP/IP 인터넷 통신**
AF_INET6	**IPv6 TCP/IP 인터넷 통신**
AF_IPX	노벨의 IPX
AF_X25	X.25 프로토콜

1. 소켓 생성

Type & Protocol

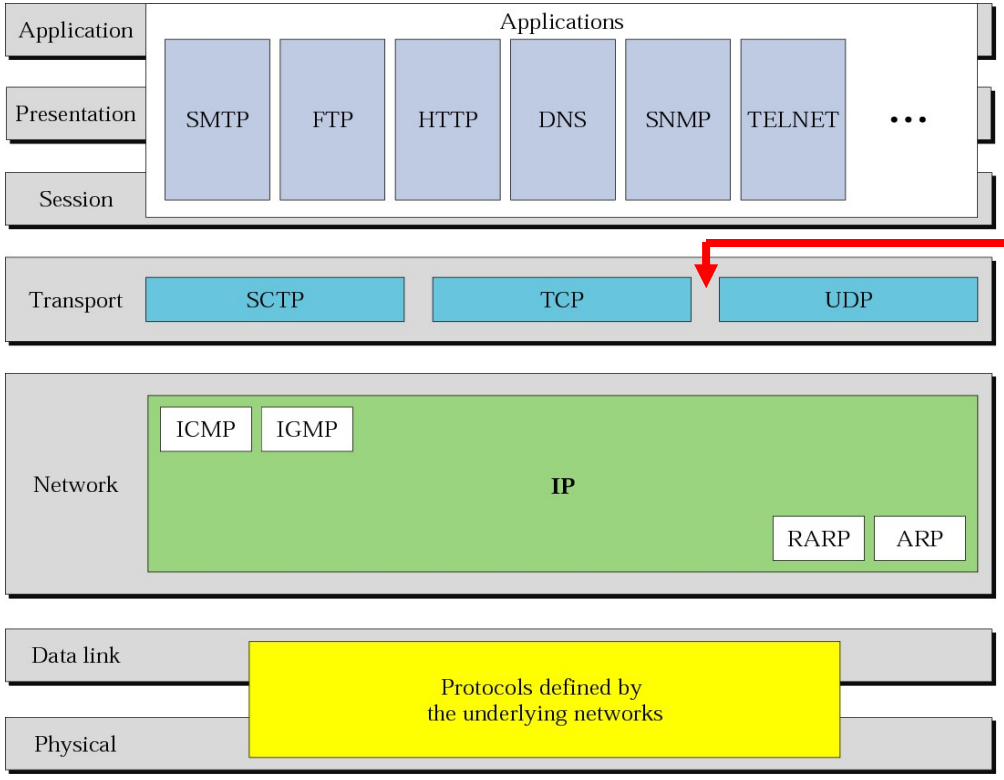
- Type : 통신에 사용할 패킷의 타입을 지정
- Protocol : 통신에 사용할 프로토콜 지정
- Type에 따라서 Protocol이 정해짐.

Type	Protocol
SOCK_STREAM	IPPROTO_TCP
SOCK_DGRAM	IPPROTO_UDP
SOCK_RAW	IPPROTO_ICMP

- SOCK_STREAM & IPPROTO_TCP : TCP 기반의 통신에 사용
- SOCK_DGRAM & IPPROTO_UDP : UDP 기반의 통신에 사용
- SOCK_RAW & (원하는 프로토콜, e.g. **IPPROTO_ICMP**) : RAW Socket으로 저수준에서 프로토콜을 직접 다룰 때 사용

1. 소켓 생성

- SOCK_STREAM & IPPROTO_TCP : TCP 기반의 통신에 사용
- SOCK_DGRAM & IPPROTO_UDP : UDP 기반의 통신에 사용
- SOCK_RAW & (원하는 프로토콜, e.g. IPPROTO_ICMP) : RAW Socket으로 저수준에서 프로토콜을 직접 다룰 때 사용
  - ICMP(Internet Control Message Protocol)을 이용하여 서버의 상태를 측정하거나, 특정 port로 향하는 패킷의 헤더내용을 분석



트랜스포트 계층 프로그래밍

1. 소켓 생성

socket 함수를 이용한 소켓 생성의 예.

- TCP 소켓 :
  socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
- UDP 소켓
  socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
- RAW 소켓
  socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)

1. 소켓 생성

socket 함수 반환 값.

- 성공적으로 소켓을 만들면 **0** 이상인 **int** 값 반환
- 소켓지정번호, **socket descriptor(file descriptor)** 라고 부른다.
- 소켓을 지시하며, 이를 이용해서 소켓을 제어한다.

소켓 관련 구조체들

```
struct sockaddr {
    unsigned short    sa_family;      // address family, AF_XXXX
    char              sa_data[14];    // 14 bytes of protocol address
};

struct sockaddr_in {                  // IPv4 AF_INET sockets:
    short             sin_family;     // e.g. AF_INET, AF_INET6
    unsigned short    sin_port;       // e.g. htons(3490)
    struct            in_addr  sin_addr;          // see struct in_addr, below
    char              sin_zero[8];    // padding or filler
};

struct in_addr {
    unsigned long  s_addr;
};
```

2. 소켓에 연결하기

connect 함수를 이용한 소켓 연결

- 연결하고자 하는 상대 **Node**의 **IP** 주소와
- 연결하고자 하는 프로그램의 **Port** 번호를 명시

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd,   const struct sockaddr *serv_addr,   socklen_t addrlen);
```

- sockfd : 파일 지정 번호
- serv_addr : 연결할 인터넷 주소와 Port 번호를 포함한 구조체 포인터
- addrlen : 두번째 매개 변수로 넘길 데이터의 크기

- 반환값 : 함수 성공 여부
  - 서버와는 달리 클라이언트는 듣기 소켓과 연결소켓을 구분하지 않음

2. 소켓에 연결하기

connect 함수의 사용 예.

struct **sockaddr_in** serveraddr;

server_sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = inet_addr("218.234.19.87");
serveraddr.sin_port = htons(8080);

client_len = sizeof(serveraddr);

connect(server_sockfd, (struct sockaddr *) &serveraddr, client_len);

- Intetnet TCP/IP 통신
- 218.234.19.87 주소로 연결 요청
- 8080 포트에 연결된 프로그램을 요청

3. 데이터 통신하기

데이터 입출력

- 소켓 함수 : send, recv, sendto, recvfrom
- 파일 함수 : **read, write**

입력함수

- recv, recvfrom, **read**

출력함수

- send, sendto, **write**

- 유닉스는 소켓을 포함한 모든 자원을 파일로 본다.
- 파일 입출력에 사용하는 함수를 소켓에 사용 가능

3. 데이터 통신하기

데이터 쓰기

```
ssize_t write(int fd, const void *buf, size_t count);
int send(int fd, const void *msg, size_t len, int flags);
```

데이터 읽기

```
ssize_t read(int fd, void *buf, size_t count);
int recv(int fd, void *buf, size_t len, int flags);
```

- fd : 소켓 지정 번호.
- buf : 통신에 사용할 데이터를 가리키는 포인터
- count : 통신에 사용할 데이터의 크기.

4. 연결 종료

- 데이터 통신이 끝났다면, close 함수를 이용해서 소켓을 닫아야 한다.
- 소켓을 닫지 않을 경우 자원 누수 발생

```
close(int sockfd);
closesocket(SOCKET sockfd);
```

서버 프로그램 만들기

bind 함수

- 소켓을 인터넷 주소에 묶어준다. **(IP 주소 & Port 번호)**

int bind(int sockfd, struct **sockaddr** *my_addr, socklen_t addrlen);

- sockfd : 파일 지정 번호
- serv_addr : 연결할 인터넷 주소와 Port 번호를 포함한 구조체
- addrlen : 두번째 매개 변수로 넘길 데이터의 크기
- 반환 값 : 성공하면 0, 실패하면 -1

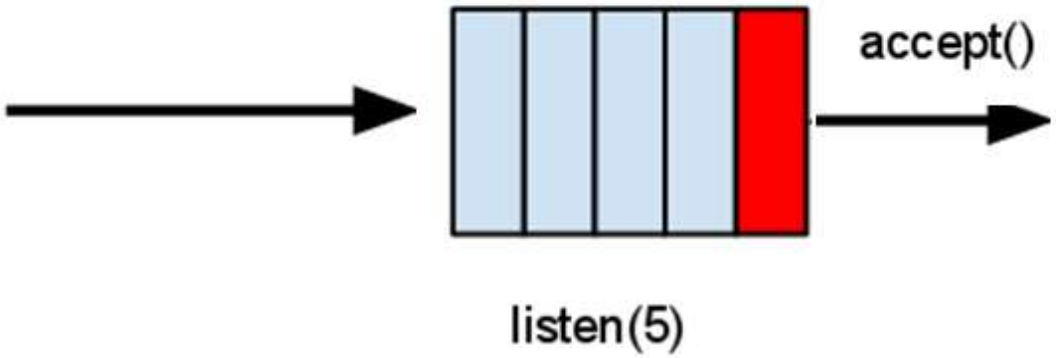
```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl(INADDR_ANY); // 모든 클라이언트로부터 접속대기
addr.sin_port = htons(8080);

state = bind(sockfd , (struct sockaddr *)&addr, sizeof(addr));
```

서버 프로그램 흐름 만들기

listen 함수 : 수신 대기열 생성

- 클라이언트의 요청은 먼저 수신 대기열(FIFO queue)에 들어간다.
- 현재 접속요청중인 클라이언트 요청에 대한 처리가 끝나기 전에 새로운 클라이언트가 요청하는 상황을 위함

```
int listen(int queue_size);                    // 5 in general
```



서버 프로그램 흐름 만들기

accept 함수

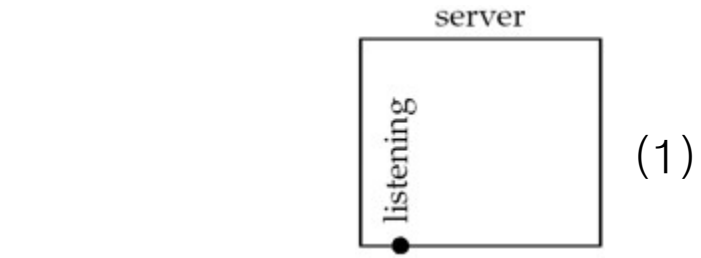
- 듣기소켓 혹은 서버소켓에 수신 대기중인 연결 요청이 있는지 확인하여 대기열의 맨 앞에 있는 클라이언트 요청을 읽는다.
- 클라이언트 요청이 있다면, **클라이언트와의 통신을 담당할 소켓 지정번호(**연결소켓 혹은 클라이언트소켓**)**를 반환한다.

```
int accept(int s,   struct sockaddr *addr,        socklen_t *addrlen);
```

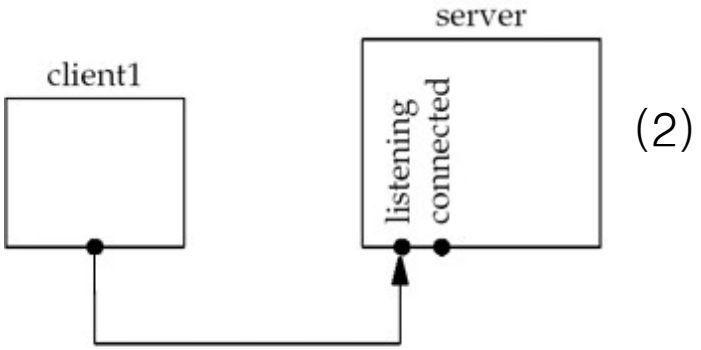
- s : 듣기 소켓의 소켓 지정 번호
- addr : 클라이언트의 주소 정보
- addrlen : 두번째 매개 변수의 데이터 크기.

- 반환값 : 소켓 지정 번호 반환
  듣기소켓과는 별도로 연결된 클라이언트와 통신을 위한 연결소켓

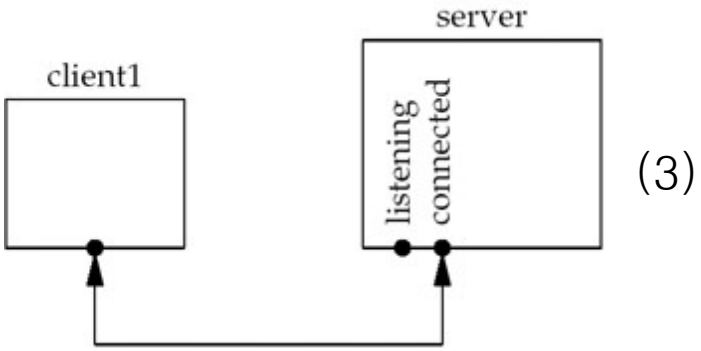
서버 프로그램 흐름 만들기



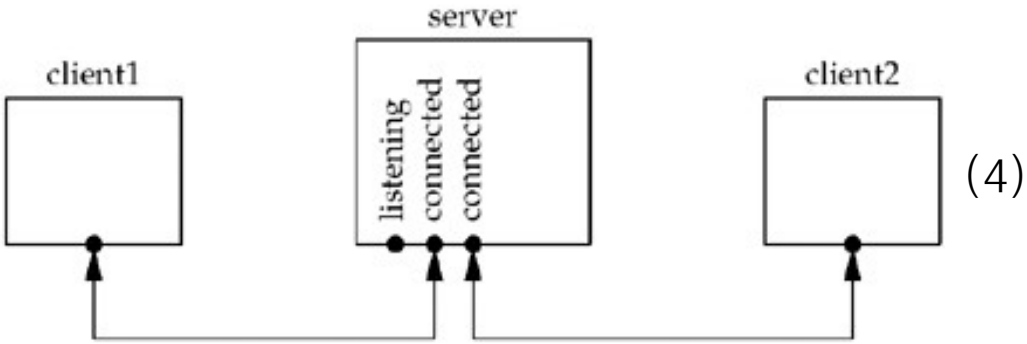
(1)



(2)



(3)



(4)

윈도우 소켓 프로그래밍(Section 11, p.108)

- 윈속 (Windows Socket API)
- load winsock.dll
- WSAStartup(), MAKEWORK() macro
- SOCKET data type
- closesocket()
- Compile option(link option), ws2_32.lib

Thank You !

뇌를 자극하는 TCP/IP 소켓 프로그래밍