

네트워크 프로그래밍

08. UDP 기반 서버/클라이언트

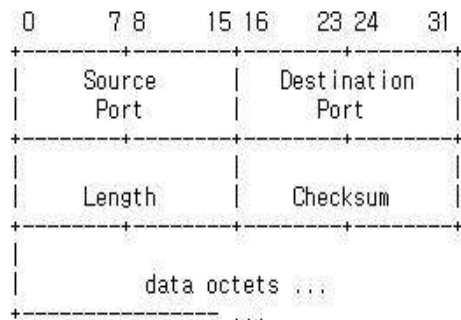
UDP(User Datagram Protocol)

□ UDP가 수행하는 두 가지 기능

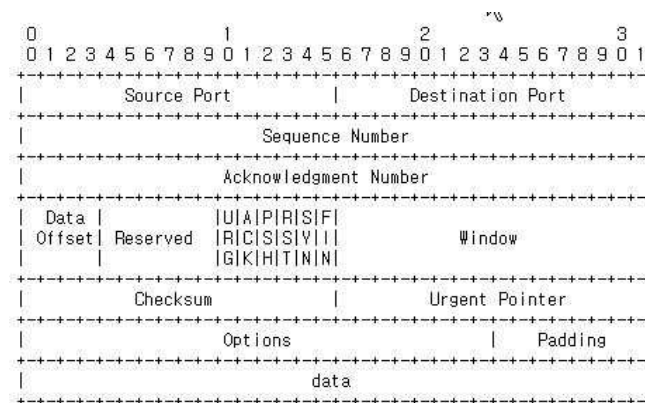
- 주소(포트)를 제공
- 데이터 변조 감지 및 변조된 데이터그램의 폐기

=> checksum
but, 모든 변조를 알아 낼 수 는 없다
운이 나쁠 경우 변조되었는데 확인 못할 수 있음

tcp > 90번 포트
udp > 90번 포트
는 서로다른 것




[UDP 헤더]



[TCP 헤더]

UDP 소켓

- TCP 소켓과 UDP 소켓의 차이점
 - ▣ Flow Control 없음
 - SEQ, ACK와 같은 메시지 전달을 하지 않음
 - ▣ 연결 단계 불필요
 - ▣ 메시지의 경계를 보존
 - ▣ Best Effort 종단 간 전송 서비스
 -  메시지 손실, 순서 바뀜

TCP 서버와 UDP 서버

□ TCP 서버의 특징

- 리스닝 소켓은 연결만을 담당
- 연결과정에서 반환된 커넥티드 소켓은 데이터 송수신을 담당
- 서버 쪽의 커넥티드 소켓과 클라이언트의 소켓은 1:1 연결
- 바이트-스트림 전송으로 전송 데이터의 크기 무제한

□ UDP 서버의 특징 여기는 소켓의 구분이 없음

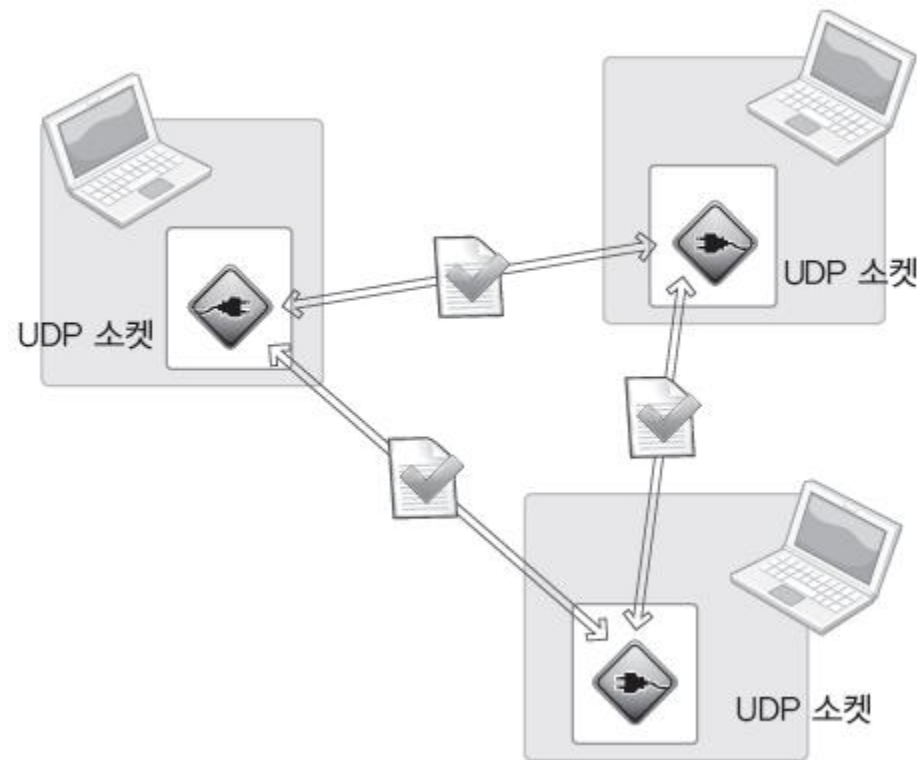
- 연결의 개념이 존재하지 않음
- 서버 소켓과 클라이언트 소켓의 구분이 없음
- 1 : 1 혹은 1 : many 데이터 송수신 가능
- 하나의 데이터그램은 65,507 바이트로 제한됨

메시지의 경계가 존재함
>>한번에 보낼 수 있는 용량의 한계가 있다

65536 - tcp , ip 헤더

UDP 소켓

- ▣ 서버 소켓과 클라이언트 소켓의 구분이 없음
- ▣ 1 : 1 혹은 1 : many 데이터 송수신 가능



UDP 기반의 데이터 입출력 함수

write와 동일

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

```
ssize_t sendto(int sock, void *buff, size_t nbytes, int flags,  
               struct sockaddr *to, socklen_t addrlen);
```

→ 성공 시 전송된 바이트 수, 실패 시 -1 반환

- sock 데이터 전송에 사용될 UDP 소켓의 파일 디스크립터를 인자로 전달.
- buff 전송할 데이터를 저장하고 있는 버퍼의 주소 값 전달.
- nbytes 전송할 데이터 크기를 바이트 단위로 전달.
- flags 옵션 지정에 사용되는 매개변수, 지정할 옵션이 없다면 0 전달.
- to 목적지 주소정보를 담고 있는 sockaddr 구조체 변수의 주소 값 전달.
- addrlen 매개변수 to로 전달된 주소 값의 구조체 변수 크기 전달.

□ sendto() 호출 시 IP와 PORT번호 자동으로 할당

MSG_PEEK
>> 메시지를 엿보다
이 플래그를 설정할 경우
읽을 때 fifo에서 빠지지 않는다. (가짜로 읽음)
다시 recvfrom하면 아직 남아있음!
>> 목적: 읽으려는 데이터의 size 파악 (buf size 조절을 위해)

UDP 기반의 데이터 입출력 함수

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

```
ssize_t recvfrom(int sock, void *buff, size_t nbytes, int flags,  
                 struct sockaddr *from, socklen_t *addrlen);
```

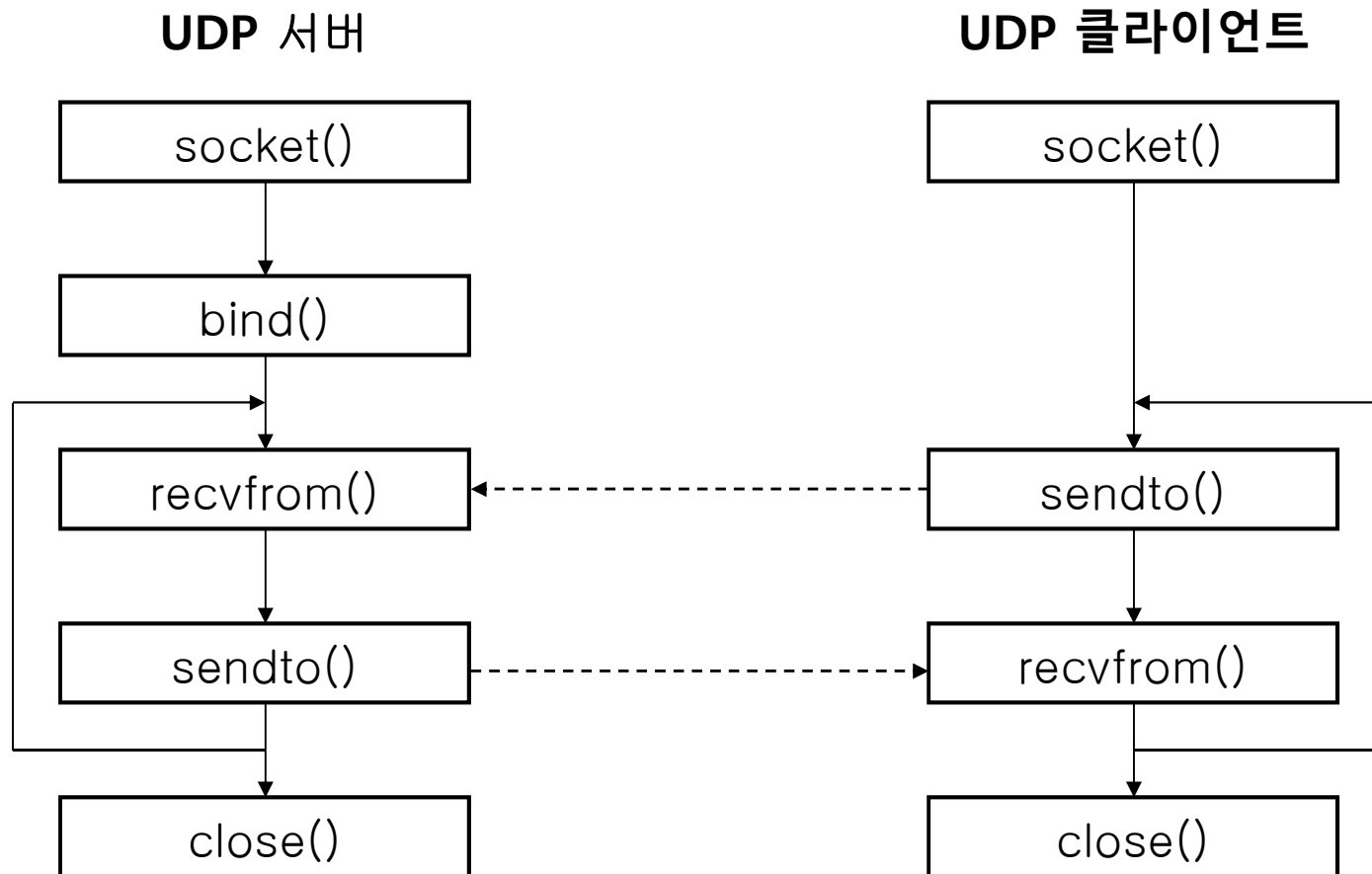
→ 성공 시 수신한 바이트 수, 실패 시 -1 반환

이 마지막 두개의 인자는 accept와 똑같다

포인터라는 것을 기억!
send to는 포인터아님

- sock 데이터 수신에 사용될 UDP 소켓의 파일 디스크립터를 인자로 전달.
- buff 데이터 수신에 사용될 버퍼의 주소 값 전달.
- nbytes 수신할 최대 바이트 수 전달, 때문에 매개변수 buff가 가리키는 버퍼의 크기를 넘을 수 없다.
- flags 옵션 지정에 사용되는 매개변수, 지정할 옵션이 없다면 0 전달.
- from 발신지 정보를 채워 넣을 struct sockaddr 구조체 변수의 주소 값 전달.
- addrlen 매개변수 from으로 전달된 주소 값의 구조체 변수의 크기 전달.

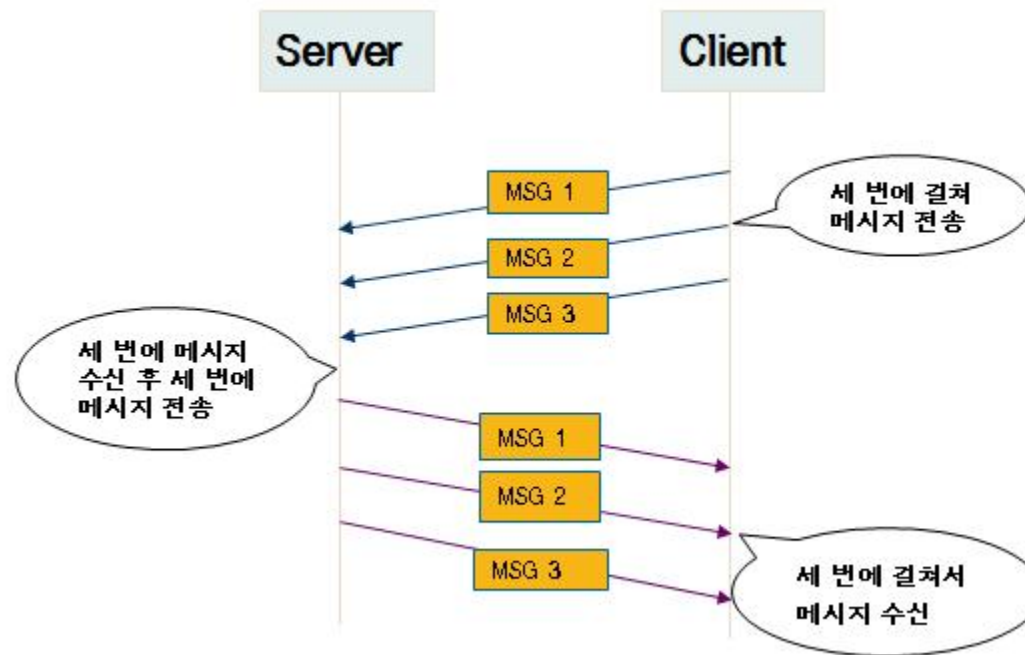
unconnected UDP 소켓



□ uecho_server.c와 uecho_client.c 참고

데이터의 경계가 존재하는 UDP 소켓

- 입력함수의 호출횟수와 출력함수의 호출횟수가 완벽히 일치해야 송신된 데이터 전부를 수신할 수 있다.



데이터의 경계가 존재하는 UDP 소켓

```
if(bind(sock, (struct sockaddr*)&my_addr, sizeof(my_addr))==-1)
    error_handling("bind() error");

for(i=0; i<3; i++)
{
    sleep(5);        // delay 5 sec.
    adr_sz=sizeof(your_addr);
    str_len=recvfrom(sock, message, BUF_SIZE, 0,
                     (struct sockaddr*)&your_addr, &adr_sz);

    printf("Message %d: %s \n", i+1, message);
}
```

bound_host1.c의 데이터 수신부분

데이터의 경계가 존재하기 때문에
5초간의 delay를 삽입해도 총 3개의
메시지를 3번의
recvfrom() 호출을 통해서 수신한다.

bound_host2.c의 데이터 전송부분

데이터의 전송에 있어서TCP와의 유일한 차이점은 사용하는 함수가 다르고 전달할 목적지 정보를 매 호출 시마다 지정한다는 점이다.

```
sendto(sock, msg1, sizeof(msg1), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
sendto(sock, msg2, sizeof(msg2), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
sendto(sock, msg3, sizeof(msg3), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
```

bounded_host1.c와 bounded_host2.c 다시 보기

- bounded_host2.c의 세 번째 sendto() 호출을 주석 처리하면?

3번째 읽을것이 없으니까 블락됨

```
if(bind(sock, (struct sockaddr*)&my_addr, sizeof(my_addr))==-1) bound_host1.c의 데이터 수신부분
    error_handling("bind() error");

for(i=0; i<3; i++)
{
    sleep(5);          // delay 5 sec.
    adr_sz=sizeof(your_addr);
    str_len=recvfrom(sock, message, BUF_SIZE, 0,
                     (struct sockaddr*)&your_addr, &adr_sz);

    printf("Message %d: %s \n", i+1, message);
}
```

bound_host2.c의 데이터 전송부분

```
sendto(sock, msg1, sizeof(msg1), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
sendto(sock, msg2, sizeof(msg2), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
sendto(sock, msg3, sizeof(msg3), 0,
        (struct sockaddr*)&your_addr, sizeof(your_addr));
```

bounded_host1.c 다시 보기

udp는 send 0가 없다
수신 Q == fifo buf

- FIFO 버퍼에 있는 첫 단위 정보의 크기가 BUF_SIZE보다 크다면?

```
if(bind(sock, (struct sockaddr*)&my_addr, sizeof(my_addr))==-1)
    error_handling("bind() error");
for(i=0; i<3; i++)
{
    sleep(5);          // delay 5 sec.
    adr_sz=sizeof(your_addr);
    str_len=recvfrom(sock, message, BUF_SIZE, 0,
                     (struct sockaddr*)&your_addr, &adr_sz);
    printf("Message %d: %s \n", i+1, message);
}
```

bound_host1.c의 데이터 수신부분

buf size만큼 읽고 남는 건 버림

그러니까 buf size를 좀 크게잡자

connected UDP 소켓

□ unconnected UDP 소켓

- sendto 함수나 recvfrom 함수를 호출했을 때만 커널과 연결된다.
 - 즉, 함수 호출이 끝나면 소켓과 커널의 연결이 해제된다.

여기가 일반적인 경우

□ connected UDP 소켓

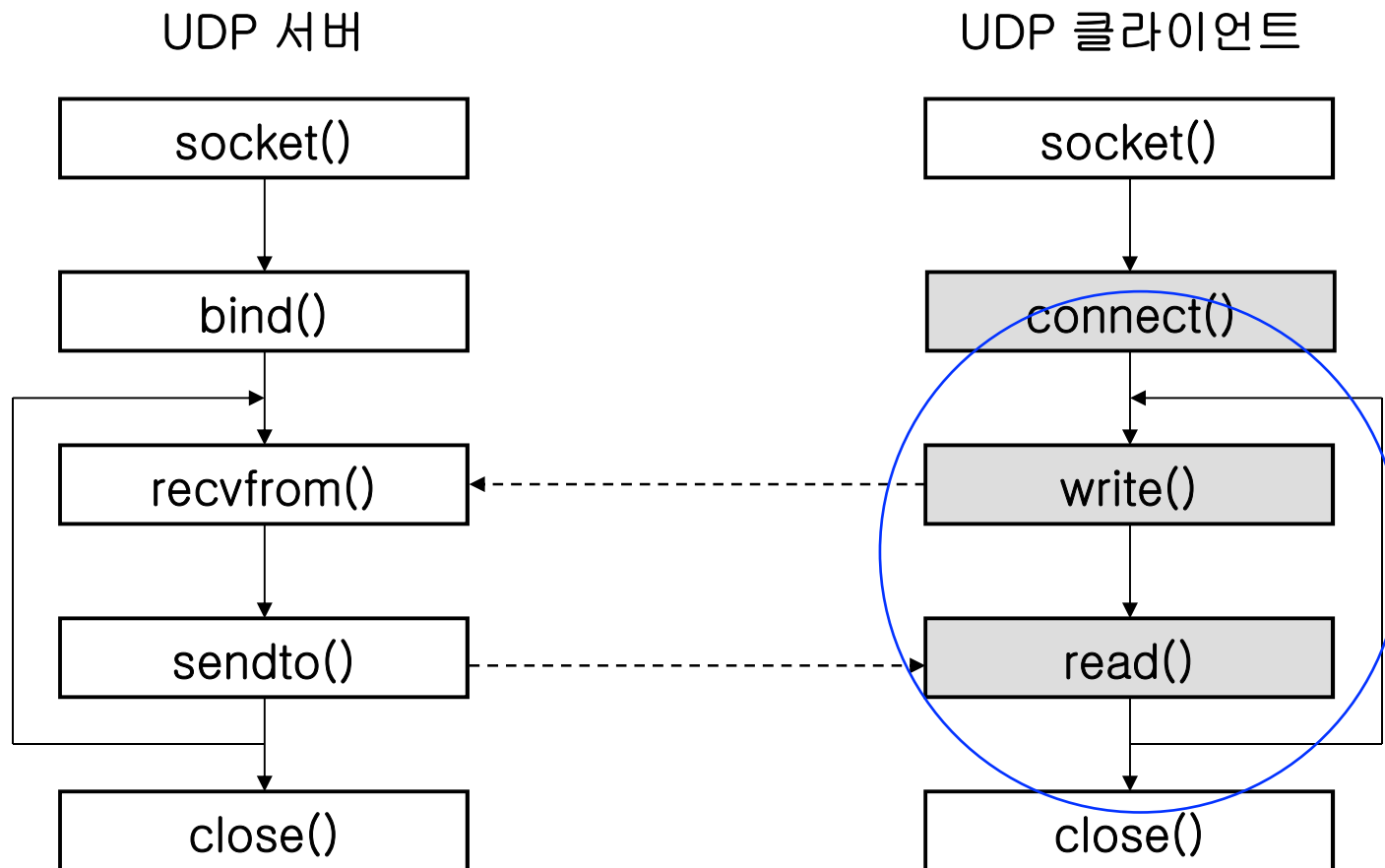
udp소켓이니까 > connect을 호출해도 syn이 가는게 아니다

```
sock=socket(PF_INET, SOCK_DGRAM, 0);
if(sock==-1)
    error_handling("socket() error");

memset(&serv_adr, 0, sizeof(serv_adr));
serv_adr.sin_family=AF_INET;
serv_adr.sin_addr.s_addr=inet_addr(argv[1]);
serv_adr.sin_port=htons(atoi(argv[2]));
connect(sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr));
```

목적지 주소

connected UDP 소켓



`sendto()`
`recvfrom()`
보다 약 30% 정도의
속도 이점이 있다.

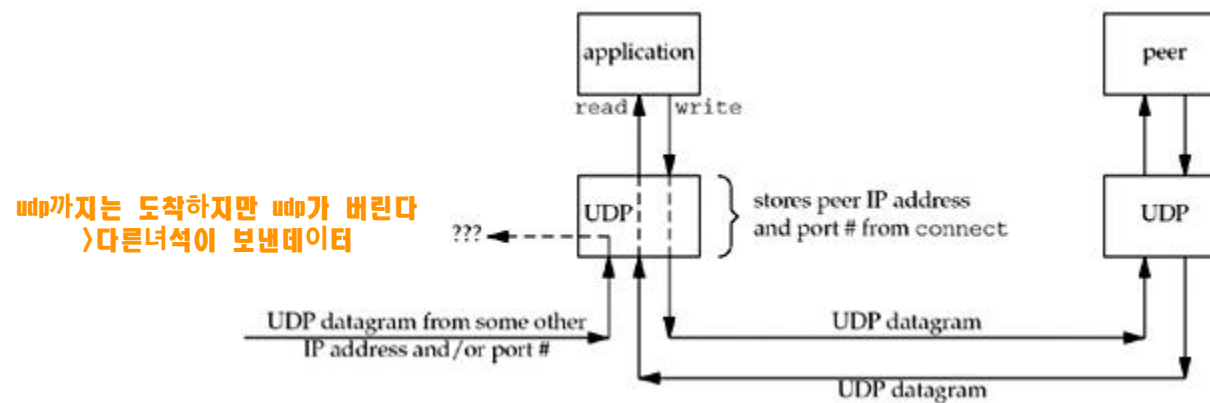
□ uecho_con_client.c와 uecho_server.c 참고

connect의 장점 > 내가 원하는 사용자와 1:1 정보교환이 가능!

connected UDP 소켓

- 사실상 연결이 이루어진 후에는 connect() 때 지정한 주소로만 데이터의 송수신이 가능

unconnected udp 소켓 인 경우
> 확인 할 수 없음



connected UDP 소켓

- ICMP 에러 메시지 수신 가능
 - ▣ 이전 행위에 대한 에러 확인 가능 (asynchronous error)

send()

recv()

존재하지 않는 포트에 데이터 전송

ICMP 메시지 도착 - "ICMP 포트 도달 불가(unreachable) 에러"

ECONNREFUSED 반환

unconnected udp소켓의 경우
이러한 메시지를 확인 할 수 없다

시간차 존재

비동기적 에러

IP와 같은 레이어에 있는 프로토콜
(internet control message protocol)

소켓 api란 read, write >> send, recv
이는 sendto, recvfrom과 다른것이니
기억 할 것!

ERROR NUMBER

connected UDP 소켓의 연결 종료(disconnect)

connected 상태에서 연결 종료
> remote ip, port number 를
밀어줌

```
struct sockaddr_in serv_adr;  
memset(&serv_adr, 0, sizeof(serv_adr));  
serv_adr.sin_family = AF_UNSPEC;  
  
connect(fd, (struct sockaddr *)&serv_adr, sizeof(serv_adr));
```