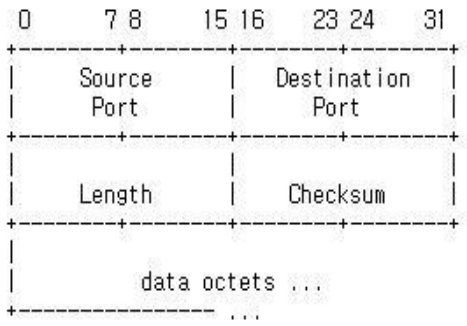
인터넷 프로토콜 4장

UDP 소켓

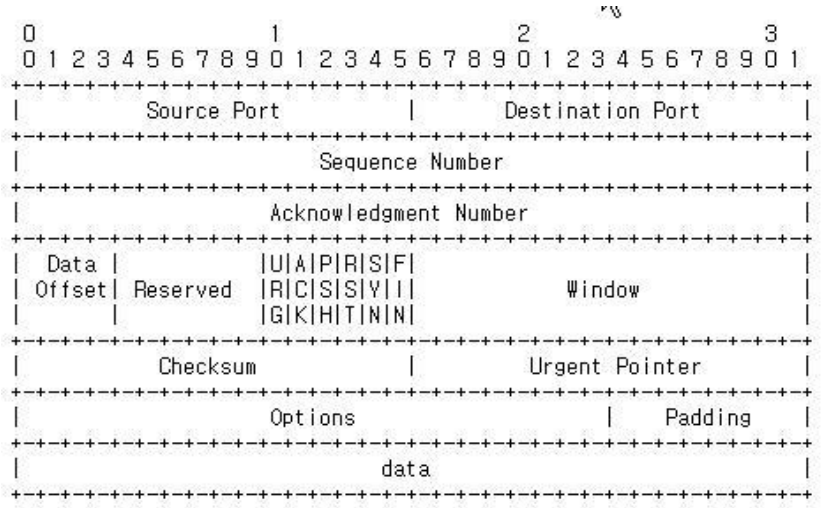
목차

UDP 소켓의 특징

- **UDP 소켓의 특성**
  - 신뢰할 수 없는 데이터 전송 방식
    - 목적지에 정확하게 전송된다는 보장이 없음. 별도의 처리 필요
  - 비 연결지향적, 순서 바뀌는 것이 가능
  - 흐름제어(flow control)를 하지 않음
  - 메시지의 경계가 있다!
    - TCP는 스트림 전송으로 send()와 recv()의 횟수가 상호 관련이 없다
    - UDP는 데이터그램 전송으로 메시지의 경계가 존재하며 송신 메시지와 수신메시지 사이의 상관관계가 존재



[UDP 헤더]



[TCP 헤더]

UDP 서버의 특징 (1)

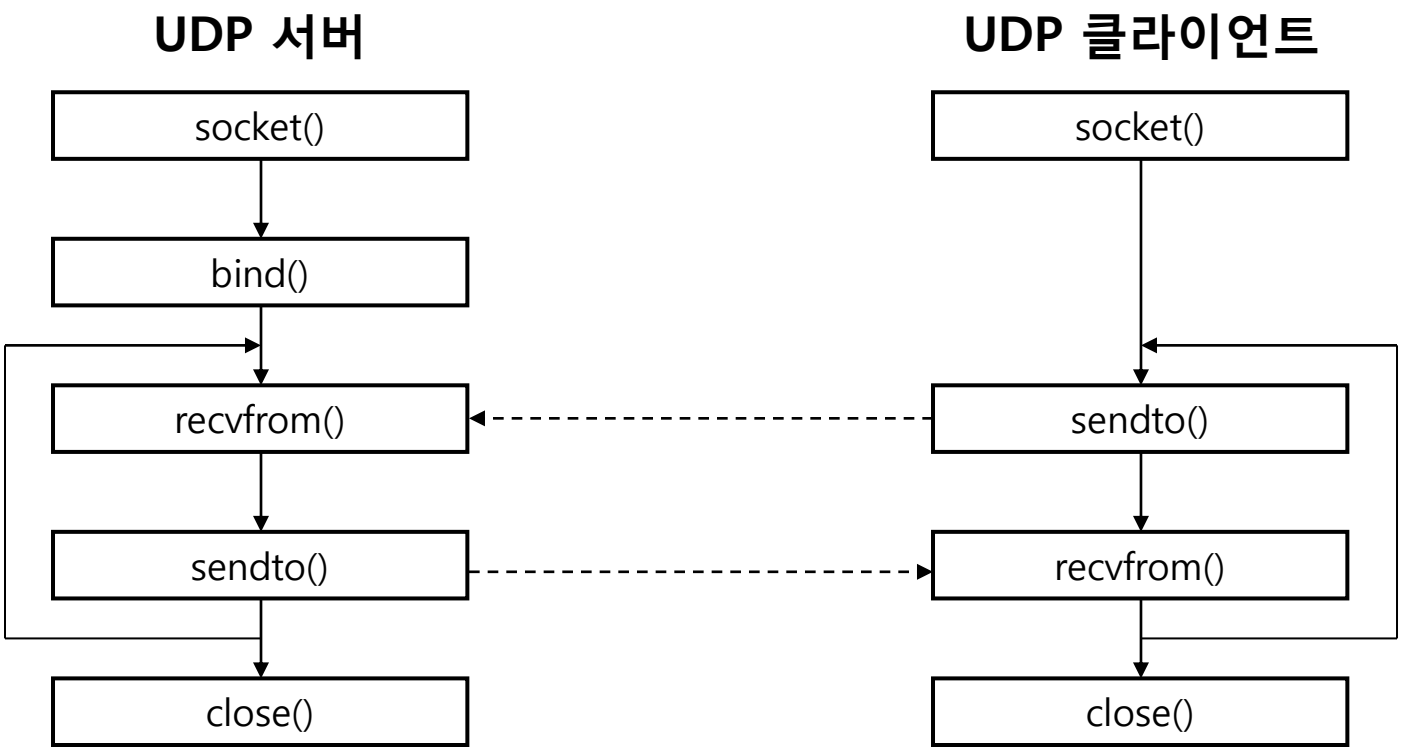
- **TCP와 UDP의 공통점**
  - 포트 번호를 이용하여 종단(응용)간 전송
  - 데이터 위변조 확인(checksum 존재)
- **TCP서버의 특징**
  - 처음 생성한 소켓은 연결만을 담당
  - 연결과정에서 반환된 소켓은 데이터 송수신을 담당
  - 서버 쪽의 데이터 송수신 소켓과 클라이언트의 소켓은 1:1연결
  - 스트림 전송으로 전송 데이터의 크기 무제한

UDP 서버의 특징 (2)

- **UDP서버의 특징**
  - UDP는 연결 자체가 없음
  - UDP서버는 다수의 클라이언트 소켓과 동시에 데이터 송수신 처리
    - 1 : 1 혹은 1: many 연결
  - 데이터그램(메시지) 단위 전송이며 하나의 데이터그램은 65535바이트 크기로 제한됨
    - 그 이상의 크기는 잘라서 보내야 함

UDP 서버-클라이언트 전송 프로세스

- **비 연결형 전송 프로세스**



UDP 소켓 생성

```
int socket(int family,int type,int proto);

int sock;
    sock = socket(PF_INET, SOCK_DGRAM,0);
    if (sock<0) { /* ERROR */ }
```

- **용도 : 메시지를 상대방에게 전송**

UDP 기반의 데이터 송수신 함수 (1)

```
ssize_t sendto (int sock, const void *msg, int len, unsigned flags
                const struct sockaddr *addr, int addrlen)
```

- **용도: 메시지를 상대방에게 전송**
- **반환 값: 성공 시 전송된 바이트 수, 실패 시 -1**
  - sock: 소켓의 파일 디스크립터, UDP 소켓
  - msg: 전송하고자 하는 데이터를 저장해 놓은 버퍼
  - len: 보낼 데이터의 크기
  - flags: 옵션(일반적으로 0)
  - addr: 전송하고자 하는 호스트의 소켓 주소 구조체
  - addrlen: 소켓주소 구조체(addr)의 크기

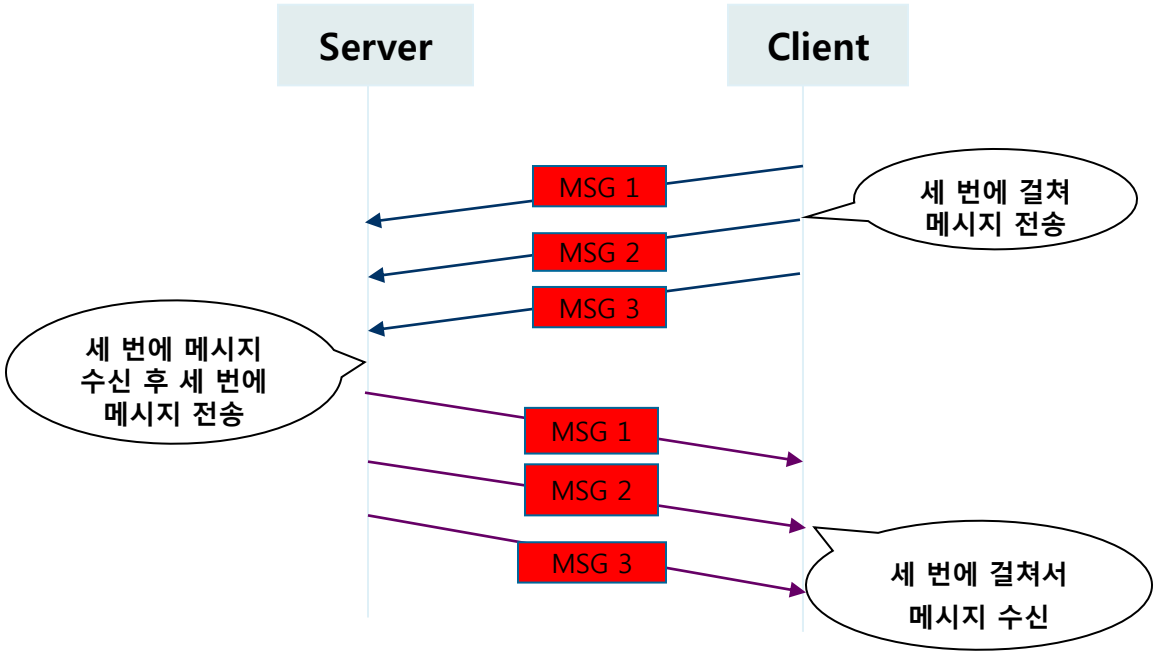
UDP 기반의 데이터 송수신 함수 (2)

```
ssize_t recvfrom (int sock, const void *buf, int len, unsigned flags
                  struct sockaddr *addr, int *addrlen)
```

- **용도 : 상대방이 전송하여 수신 버퍼에 도착한 메시지 데이터를 메모리에 복사**
- **반환 값 : 성공 시 복사된 바이트 수, 실패 시 -1**
  - sock: 데이터를 수신할 소켓의 파일 디스크립터, UDP 소켓
  - buf: 수신할 데이터를 저장할 버퍼
  - len: 수신 할 수 있는 최대 바이트 수
  - flags: 옵션
  - addr: 전송한 호스트의 소켓 주소 구조체
  - addrlen: addr이 가리키는 구조체 변수의 크기
- FIFO 버퍼에 있는 단위정보의 크기가 len보다 큰 경우, len까지만 반환되고, 읽히지 않은 **나머지 정보는 버려짐**
  - 충분한 크기의 버퍼 준비 필요 ! (UDP 데이터 최대 크기 65,507)
- MSG_PEEK 플래그 사용 가능

UDP 메시지의 처리

- 경계가 있는 메시지
  - TCP와는 달리 UDP에서는 하나의 sendto()와 하나의 recvfrom이 쌍(pair)를 이룸
  - sendto() 와 recvfrom() 호출이 서로 짝을 이루도록 순서가 맞아야 함



UDP Echo 서버 예제(simple)

```
#define BUFSIZE 65507

int main(int argc, char **argv)
{
    int serv_sock;
    char message[BUFSIZE];
    int str_len;

    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    int clnt_addr_size;

    serv_sock=socket(PF_INET, SOCK_DGRAM, 0);

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if (bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
            error_handling("bind() error");

    while(1) {
              clnt_addr_size = sizeof(clnt_addr);
              str_len = recvfrom(serv_sock, message, BUFSIZE, 0, (struct sockaddr*)&clnt_addr,  &clnt_addr_size);

              sendto(serv_sock, message, str_len, 0, (struct sockaddr*)&clnt_addr, sizeof(clnt_addr));

     }

        close(serv_sock);
        return 0;
}
```

UDP Echo 클라이언트 예제(simple)

```
#define BUFSIZE 65507

int main(int argc, char **argv)
{
    int sock;
    char message[BUFSIZE];
    int str_len, addr_size;

    struct sockaddr_in serv_addr;
    struct sockaddr_in from_addr;

    sock=socket(PF_INET, SOCK_DGRAM, 0);

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    while(1)
    {
        fgets(message, sizeof(message), stdin);

        if(!strcmp(message,"q\n")) break;

        sendto(sock, message, strlen(message), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

        addr_size = sizeof(from_addr);
        str_len = recvfrom(sock, message, BUFSIZE, 0, (struct sockaddr*)&from_addr, &addr_size);

        message[str_len] = 0; // '\0'
        printf("from server: %s", message);
    }
    close(sock);
    return 0;
}
```

sUDPEchoServer.c (1)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

#define BUFSIZE 65507

int main(int argc, char **argv)
{
        int serv_sock;
        char message[BUFSIZE];
        int str_len;
        struct sockaddr_in serv_addr;
        struct sockaddr_in clnt_addr;
        int clnt_addr_size;

        if (argc < 2) {
                fprintf(stderr, "usage: %s port_no\n", argv[0]);
                exit(1);
        }

```

sUDPEchoServer.c (2)

```
25          serv_sock=socket(PF_INET, SOCK_DGRAM, 0);
26          memset(&serv_addr, 0, sizeof(serv_addr));
27          serv_addr.sin_family = AF_INET;
28          serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
29          serv_addr.sin_port = htons(atoi(argv[1]));
30
31          if (bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
32                  perror("bind() error");
33
34          while(1) {
35                  clnt_addr_size = sizeof(clnt_addr);
36                  str_len = recvfrom(serv_sock, message, BUFSIZE, 0, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
37
38                  sendto(serv_sock, message, str_len, 0, (struct sockaddr*)&clnt_addr, sizeof(clnt_addr));
39          }
40          close(serv_sock);
41          return 0;
42  }
```

sUDPEchoClient.c (1)

```
 1 #include <stdio.h>
 2 #include <unistd.h>
 3 #include <stdlib.h>
 4 #include <string.h>
 5 #include <sys/types.h>
 6 #include <sys/socket.h>
 7 #include <netinet/in.h>
 8 #include <arpa/inet.h>
 9 #include <netdb.h>
10
11 #define BUFSIZE 65507
12 int main(int argc, char **argv)
13 {
14         int sock;
15         char message[BUFSIZE];
16         int str_len, addr_size;
17         struct sockaddr_in serv_addr;
18         struct sockaddr_in from_addr;
19
20         if (argc < 3) {
21                 fprintf(stderr, "usage: %s ip_addr port_no\n", argv[0]);
22                 exit(1);
23         }
24
```

sUDPEchoClient.c (2)

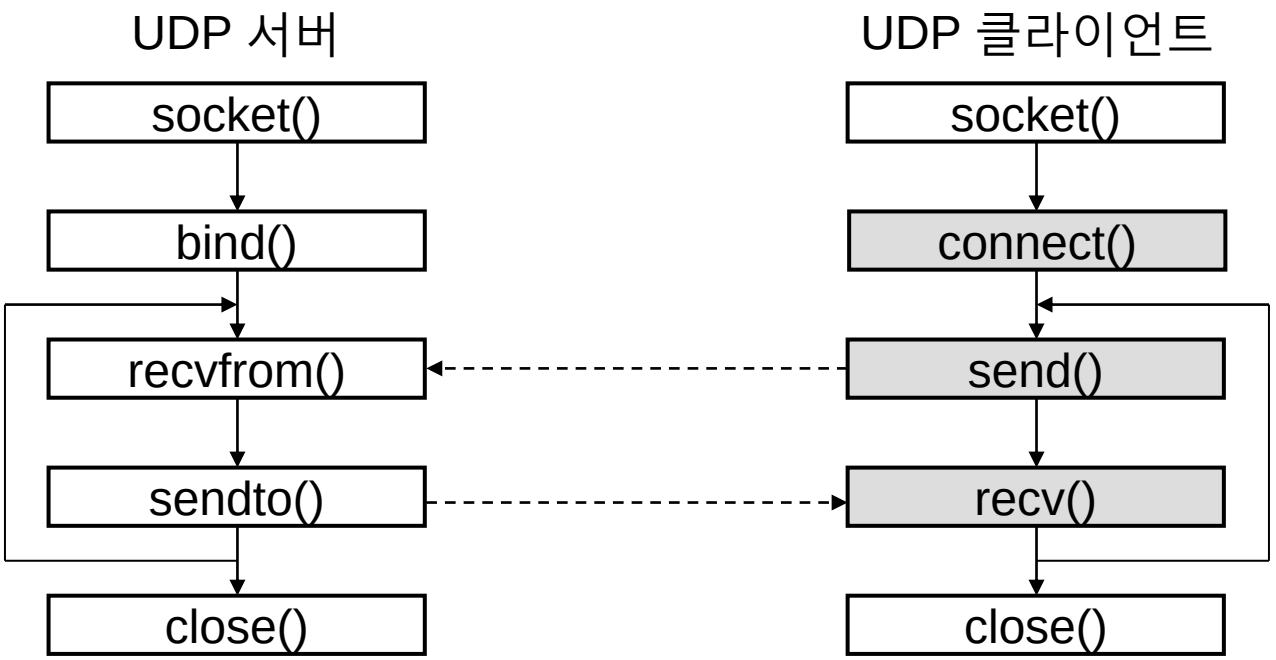
```
25          sock=socket(PF_INET, SOCK_DGRAM, 0);
26
27          memset(&serv_addr, 0, sizeof(serv_addr));
28          serv_addr.sin_family = AF_INET;
29          serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
30          serv_addr.sin_port = htons(atoi(argv[2]));
31
32          while(1)
33          {
34                  fgets(message, sizeof(message), stdin);
35                  if(!strcmp(message,"q\n")) break;
36                  sendto(sock, message, strlen(message), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
37                  addr_size = sizeof(from_addr);
38                  str_len = recvfrom(sock, message, BUFSIZE, 0, (struct sockaddr*)&from_addr, &addr_size);
39                  message[str_len] = 0; // ' \0'
40                  printf("from server: %s", message);
41          }
42          close(sock);
43          return 0;
44  }
```

교재 Makefile

```
CFLAGS = -g -std=gnu99
COM_OBJS = DieWithMessage.o AddressUtility.o
CLI_OBJS = UDPEchoClient.o
SRV_OBJS = UDPEchoServer.o
TARGETS = echo_cli echo_srv

all:    $(TARGETS)

echo_cli : $(COM_OBJS) $(CLI_OBJS)
        gcc -o echo_cli $(CLI_OBJS) $(COM_OBJS)

echo_srv : $(SRV_OBJS) $(COM_OBJS)
        gcc -o echo_srv $(SRV_OBJS) $(COM_OBJS)

clean:
        rm $(COM_OBJS) $(CLI_OBJS) $(SRV_OBJS)
        rm $(TARGETS)
```

UDP 서버-클라이언트 전송 프로세스

- **연결형 전송 프로세스**
  - 사용 목적: 사용 편의 , 송수신 성능 향상
  - 이전 행위에 대한 오류 확인 가능



응용 과제

- 2장 과제 개선2와 같이 사용자 입력을 계속 받아 전달하도록 개선
  - 동시에 여러 클라이언트를 동작하여 서버 동작 확인
  - 서버 화면 캡처