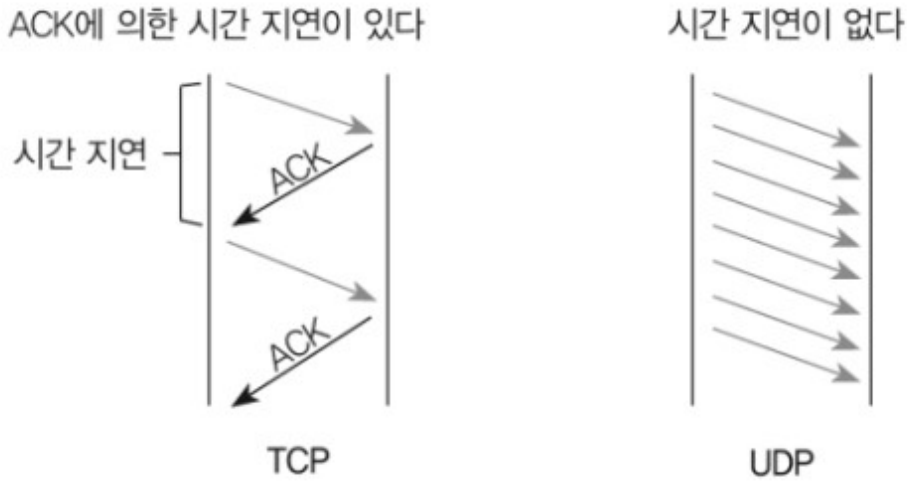
UDP 소켓 프로그래밍

뇌를 자극하는 TCP/IP 소켓 프로그래밍

TCP의 단점

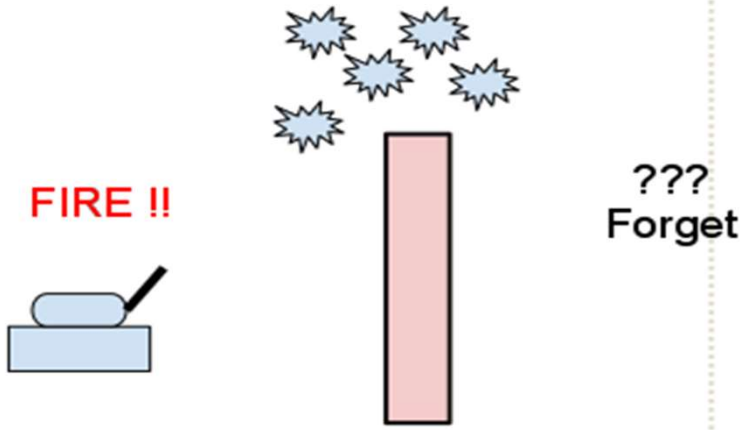


- 신뢰성 제공을 위하여
  - 패킷 재 전송
  - 패킷 순서 조합
- 낮은 성능
  - 패킷에 대한 응답**(ACK)** : 시간 지연, CPU 소모
- 연결**(connection)** 관리 **- overhead** 발생
- **Streaming** 서비스에 불리
  - 재 전송 요청에 따른 서비스 지연

  ex) 실시간 대화식**(real-time interactive) Multimedia** 데이터 통신
  - Download streaming video 응용의 경우 buffer 사용하여 서비스 제공 가능

UDP의 특징과 장점

- UDP : User Datagram Protocol
- Datagram 중심 프로토콜
- Fire and Forget
  - 전송 패킷에 대한 응답을 기다리지 않음

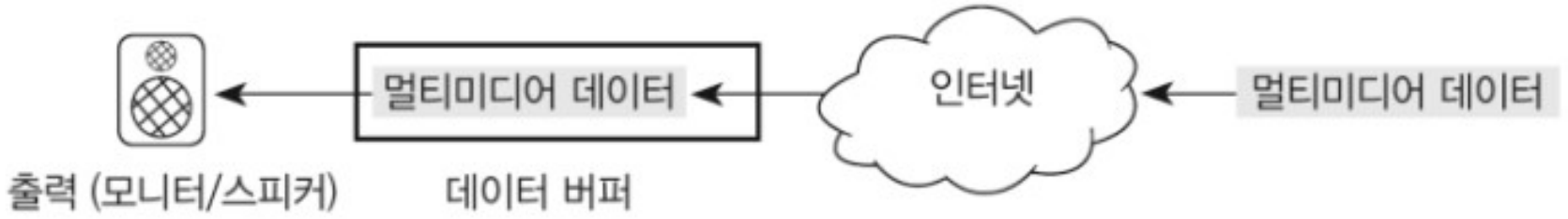


- Not ordered
- 신뢰할 수 없음(unreliable)
  - 패킷이 제대로 전송 됐는지
  - 오류가 없는지 확인할 수 없음
- 높은 성능
  - 신뢰성을 희생해서 성능 확보

- 품질, 신뢰성이 그다지 중요하지 않은 서비스, 연속성을 중시하는 서비스에 적합

TCP로 멀티미디어 데이터 다루기

- 품질과 연속성 모두를 충족하기 위하여
  - 멀티미디어 데이터를 관리하는 버퍼(응용프로그램 내의 사용자 버퍼) 운영
  - 대략적인 지연시간을 계산하여 예상되는 지연시간만큼의 데이터를 서비스 초기에 미리 받아놓음
  - “buffering…” 메시지



UDP 헤더

- TCP에 비해서 단순
  - 목적지로 보내기 위한 정보만 가지고 있음

bits	0 – 15	16 – 31
0	Source Port Number	Destination Port Number
32	Length	Checksum
64	Data	

- Source Port Number : 출발지 포트 번호
- Destination Port Number : 목적지 포트 번호
- Length : 데이터 길이
- Checksum : checksum 값

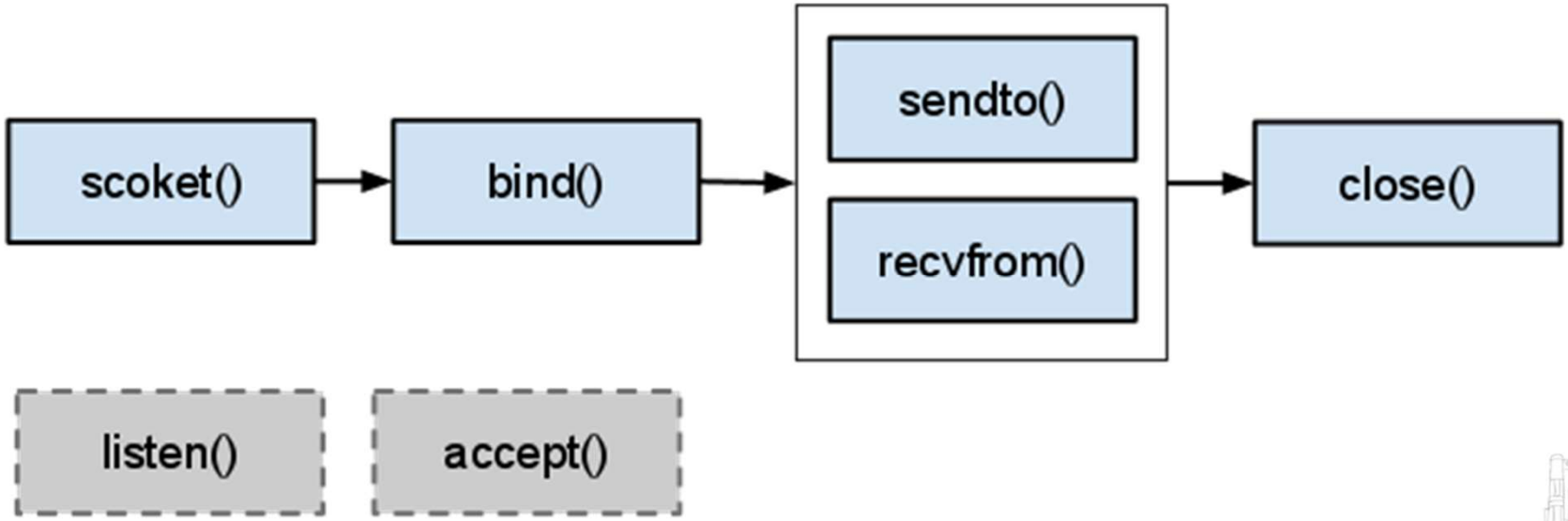
UDP 소켓 만들기

- socket 함수 이용
- Domain : AF_INET, 인터넷 영역
- Protocol
    - SOCK_DGRAM : UDP 통신
    - SOCK_STREAM : TCP 통신

```
socket(AF_INET, SOCK_DGRAM, 0);

socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

UDP 서버 프로그램 흐름

- socket() : UDP 소켓 생성
- bind() : socket을 port number /IP address에 bind
- **listen() :** 필요 없음
- **accept() :** 필요 없음
- **sendto() / recvfrom() : read(), write(), send(), recv()** 대신 사용함
  - **UDP** 데이터를 전송할 수 있는 소켓 함수 사용
- close() : 소켓 종료



UDP bind 함수사용하기.

```
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons(1234);
state = bind(sockfd, (struct sockaddr *)&serveraddr,
        sizeof(serveraddr));
```

- bind 함수의 목적은 port 번호와 기다릴 주소를 bind 하므로
- TCP 소켓의 bind와 사용방법에 차이가 없다.

UDP 통신 함수 사용하기.

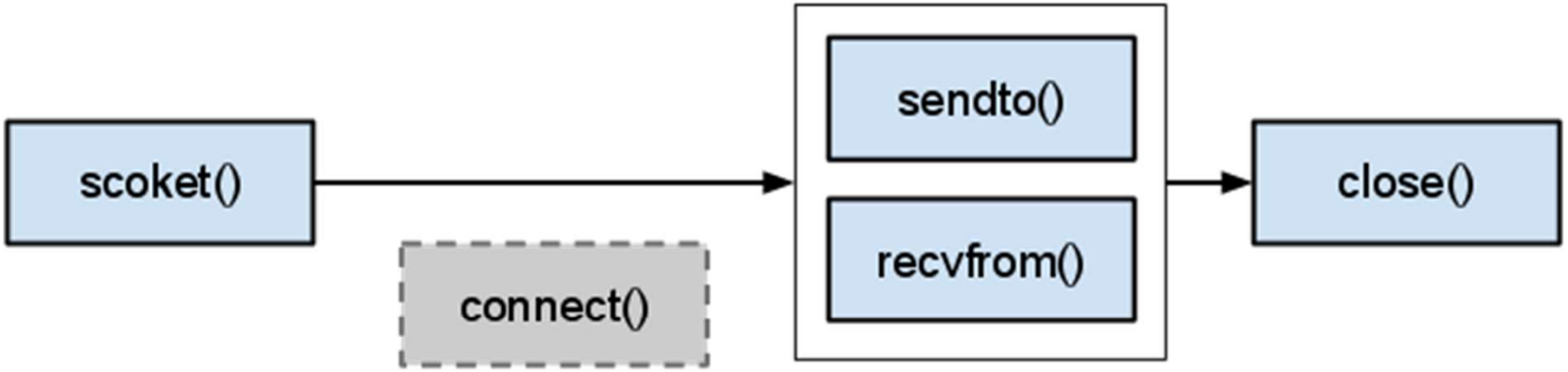
```
recvfrom(sockfd, buf, MAXLINE, 0,
        (struct sockaddr *)&cliaddr, &clilen);

sendto(sockfd, buf, strlen(buf), 0,
        (struct sockaddr *)&cliaddr, sizeof(cliaddr));
```

- 연결을 맺지 않으므로, 모든 데이터 송수신에 있어서 **IP** 주소, **port** 번호 필요
  - write(), read() 사용 불가능
  - sendto(), recvfrom() 을 대신 사용함
- IP 주소 정보를 위해서 sockaddr_in 구조체를 사용한다.

UDP 클라이언트 프로그램 흐름

- 연결을 맺지 않으므로 **connect** 함수를 사용하지 않아도 된다.

- 그러나, UDP 소켓에서 **connect** 함수를 사용하면
  - 서버를 명시할 목적으로 사용할 수는 있다.
  - 데이터를 보낼 인터넷 주소와 포트번호를 고정(즉, 생략)



UDP 클라이언트와 connect 함수

- **connect 함수가 사용되기도 함**. 이 경우 실제로 connect 과정을 수행하는 것은 아니다.
- 서버를 명시하기 위해서 사용
- 데이터를 전송할 때, **주소 복사 과정을 생략**할 수 있다.

```
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr("111.111.111.111");
addr.sin_port = htons(8081);
connect(sockfd, &addr, sizeof(addr));

// 111.111.111.111:8081로 데이터를 전송
sendto(sockfd, buf, strlen(buf), 0, NULL, len);
```

UDP 사용 이유

- 성능을 높일 수 있다.
- 실시간 멀티미디어 서비스 프로그램 개발이 용이하다.
    - 연속성이 신뢰성 보다 중요한 서비스
- 개발이 쉽다.
    - 흐름을 관리하기 위한 노력이 필요 없다.

- Agent & Manager 모델에 적합
    - Manager 가 Agent 로 데이터를 요청
    - 다수의 Agent를 효과적으로 관리
    - 오류에 민감하지 않음 (다시 요청하면 됨)

calc_linux_server.c - Linux

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT_NUM 3800
#define MAXLEN 256

struct cal_data
{
        int left_num;
        int right_num;
        char op;
        int result;
        short int error;
};
```

calc_linux_server.c - Linux

```
int main(int argc, char **argv)
{
        int sockfd;
        socklen_t addrlen;
        int cal_result;
        int left_num, right_num;
        struct sockaddr_in addr, cliaddr;
        struct cal_data rdata;

        if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
        {
                return 1;
        }
        memset((void *)&addr, 0x00, sizeof(addr));
        addr.sin_family = AF_INET;
        addr.sin_addr.s_addr = htonl(INADDR_ANY);
        addr.sin_port = htons(PORT_NUM);

        addrlen = sizeof(addr);
        if( bind(sockfd, (struct sockaddr *)&addr, addrlen) == -1)
        {
                return 1;
        }
```

calc_linux_server.c - Linux

$gcc … -DDEBUG

```
        while(1)
        {
                addrlen = sizeof(cliaddr);
                recvfrom(sockfd, (void *)&rdata, sizeof(rdata), 0,
                                (struct sockaddr *)&cliaddr, &addrlen);
#if DEBUG
        printf("Client Info : %s (%d)\n", inet_ntoa(cliaddr.sin_addr), ntohs(cliaddr.sin_port));
        printf("%02x %c %02x\n", ntohl(rdata.left_num), rdata.op, ntohl(rdata.right_num));
#endif
                left_num  = ntohl(rdata.left_num);
                right_num  = ntohl(rdata.right_num);
                switch(rdata.op)
                {
                        case '+':   cal_result = left_num + right_num;  break;
                        case '-':    cal_result = left_num - right_num;   break;
                        case 'x':    cal_result = left_num * right_num;   break;
                        case '/':    if(right_num == 0)  {   rdata.error = htons(2);  break; }
                                        cal_result = left_num / right_num;   break;
                }
                rdata.result=htonl(cal_result);
                sendto(sockfd, (void *)&rdata, sizeof(rdata), 0,
                                (struct sockaddr *)&cliaddr, addrlen);
        }
        return 1;
}
```

-DDEBUG

calc_linux_cli.c - Linux

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT_NUM 3800
#define MAXLEN 256

struct cal_data
{
        int left_num;
        int right_num;
        char op;
        int result;
        short int error;
};
```

calc_linux_cli.c - Linux

```
int main(int argc, char **argv)
{
        int sockfd;
        struct sockaddr_in addr;
        struct cal_data sdata, recvaddr;

        char msg[MAXLEN];
        int left_num;
        int right_num;
        socklen_t addrlen;

        char op[2];

        if (argc != 2)
        {
                printf("Usage : %s [ipaddress]\n", argv[0]);
                return 1;
        }
        if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1 )
        {
                return 1;
        }
```

calc_linux_cli.c - Linux

```
        memset((void *)&addr, 0x00, sizeof(addr));
        addr.sin_family = AF_INET;
        addr.sin_addr.s_addr = inet_addr(argv[1]);
        addr.sin_port = htons(PORT_NUM);

        while(1)
        {
                printf("> ");
                fgets(msg, MAXLEN-1, stdin);
                if(strncmp(msg, "quit\n",5) == 0) {  break; }
                sscanf(msg, "%d%[^0-9]%d", &left_num, op, &right_num);
                memset((void *)&sdata, 0x00, sizeof(sdata));
                sdata.left_num = htonl(left_num);
                sdata.right_num = htonl(right_num);
                sdata.op = op[0];
                addrlen = sizeof(addr);
    sendto(sockfd, (void *)&sdata, sizeof(sdata), 0, (struct sockaddr *)&addr, addrlen);
    recvfrom(sockfd, (void *)&sdata, sizeof(sdata), 0, (struct sockaddr *)&recvaddr, &addrlen);
        printf("%d %c %d = %d\n", ntohl(sdata.left_num), sdata.op, ntohl(sdata.right_num),
            ntohl(sdata.result));
        }
        close(sockfd);
}
```

Lab. calc_linux_server.c, calc_linux_cli.c

Lab

```
osnw00000000@osnw00000000-osnw: ~/week08
osnw00000000@osnw00000000-osnw:~/week08$ gcc -o cal_linux_server cal_linux_server.c
osnw00000000@osnw00000000-osnw:~/week08$ gcc -o cal_linux_debug  cal_linux_server.c -DDEBUG
osnw00000000@osnw00000000-osnw:~/week08$ gcc -o cal_linux_cli cal_linux_cli.c
osnw00000000@osnw00000000-osnw:~/week08$ ./cal_linux_server
^C
osnw00000000@osnw00000000-osnw:~/week08$ ./cal_linux_debug
00 00 00 01 00 00 00 02 2b 00 00 00 00 00 00 00 00 00 00 00
Client Info : 127.0.0.1 (49057)
01 + 02
00 00 00 01 00 00 00 02 2b 00 00 00 00 00 00 03 00 00 00 00
00 00 00 02 00 00 00 03 78 00 00 00 00 00 00 00 00 00 00 00
Client Info : 127.0.0.1 (49057)
02 x 03
00 00 00 02 00 00 00 03 78 00 00 00 00 00 00 06 00 00 00 00
```

```
osnw00000000@osnw00000000-osnw: ~/week08
osnw00000000@osnw00000000-osnw:~/week08$ ./cal_linux_cli 127.0.0.1
> 1+2
1 + 2 = 3
> 2x3
2 x 3 = 6
> quit
osnw00000000@osnw00000000-osnw:~/week08$ ./cal_linux_cli 127.0.0.1
> 1+2
1 + 2 = 3
> 2x3
2 x 3 = 6
> quit
osnw00000000@osnw00000000-osnw:~/week08$
```

Thank You !

뇌를 자극하는 TCP/IP 소켓 프로그래밍