네트워크 프로그래밍

04. 소켓의 생성

프로토콜의 이해와 소켓의 생성

- 프로토콜이란?
  - 개념적으로 약속의 의미를 담고 있다.
  - 컴퓨터 상호간의 데이터 송수신에 필요한 통신규약.
  - 소켓을 생성할 때 기본적인 프로토콜을 지정해야 한다.

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

int socket(int domain, int type, int protocol);
```

➡ 성공 시 파일 디스크립터, 실패 시 -1 반환

- domain 소켓이 사용할 프로토콜 체계(Protocol Family) 정보 전달.
- type 소켓의 데이터 전송방식에 대한 정보 전달.
- protocol 두 컴퓨터간 통신에 사용되는 프로토콜 정보 전달.

우린 PF_INET use

TCP

UDP

매개변수 domain, type 그리고 protocol이 모두 프로토콜 정보와 관련이 있다.

프로토콜 체계(Protocol Family)

이름	프로토콜 체계(Protocol Family)
PF_INET	IPv4 인터넷 프로토콜 체계
PF_INET6	IPv6 인터넷 프로토콜 체계
PF_LOCAL	로컬 통신을 위한 UNIX 프로토콜 체계
PF_PACKET	Low Level 소켓을 위한 프로토콜 체계
PF_IPX	IPX 노벨 프로토콜 체계

소켓의 타입(Type)

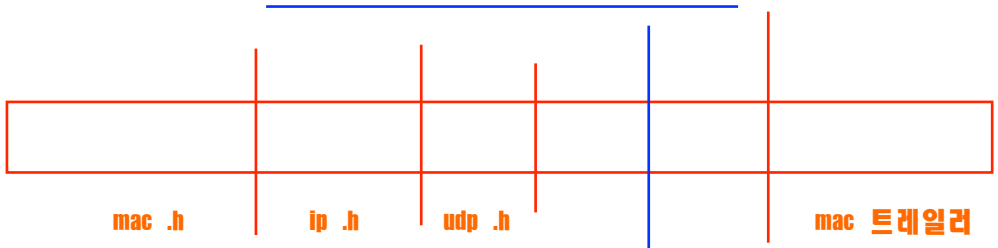
- 소켓의 타입
  - 데이터 전송방식을 의미함.
  - 소켓이 생성될 때 소켓의 타입도 결정됨.

- 프로토콜 체계 PF_INET의 대표적인 소켓 타입 둘
  - 연결 지향형 소켓 타입
  - 비 연결 지향형 소켓 타입.

FILE의 경우
read에서bufsize보다 작게 읽히는 경우는
파일의 끝인 경우만(1회)
pipe, terminal의 경우는
읽을 때 마다 bufsize보다 작을 수 있음
(여러번 가능)

*terminal의 특성
〉한 라인씩 받아서 pipe에 써줌

tip:프로세스간 데이터를 주고받을때
os 가 허락 해준 공간(shared memory)을 주는것
〉ipc(inter process communication)



두 타입의 소켓

TCP 소켓

- 연결지향형 소켓(SOCK_STREAM)
  - 중간에 데이터 소멸되지 않는다.
  - 전송 순서대로 데이터가 수신된다.
  - 데이터의 경계가 존재하지 않는다.
  - 소켓 대 소켓의 연결은 반드시 1대 1의 구조.

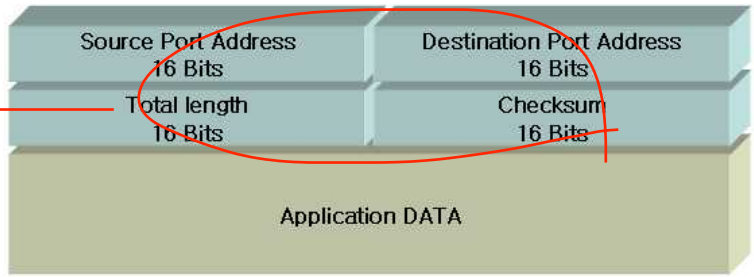
boundary
〉송신자의 송신횟수 == 수신자의 수신횟수 〉〉경계가 있다
!= 〉〉 경계가 없다

tcp는 시냇물에 비유 〉경계가 없으니 받을 때 사이즈를 정해, 나눠서 받는다
udp는 택배로 비유 받을 때 택배 단위로 받음

- 비 연결지향형 소켓(SOCK_DGRAM)
  - 전송순서 상관없이 빠른 속도의 전송을 지향
  - 데이터 손실 및 파손의 우려 있다.
  - 데이터의 경계가 존재한다.
  - 한번에 전송할 수 있는 데이터의 크기가 제한된다.

(2^16-1)-8-20 = 65507바이트
위에 8바이트 = 헤더
20바이트 =ip헤더

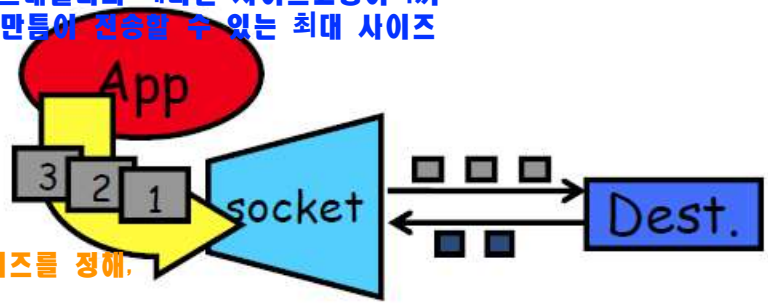
UDP헤더



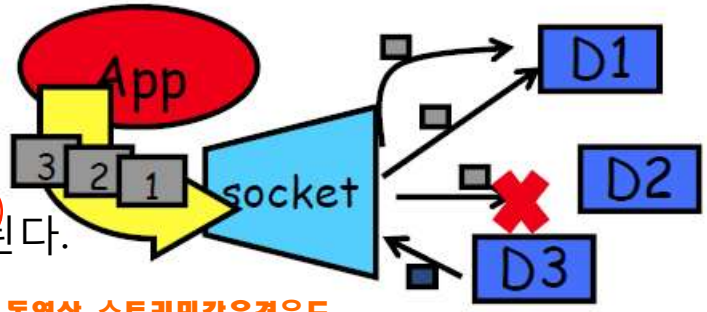
UDP 데이터그램 구조

TCP 소켓

이만큼을 MTU(maximum transmission unit)
〉맥트레일러와 헤더는 사이즈고정이니까
〉이만틈이 전송할 수 있는 최대 사이즈



UDP 소켓



동영상 스트리밍같은경우도
어느정도의 순서는 유지해야 할 경우가 있다
〉이때 UDP를 사용한다면
어느정도 순서를 맞춰주는 기능을
app에서 구현해준다.

프로토콜의 최종선택!

Remember!

- TCP 소켓

  IPv4 인터넷 프로토콜 체계에서 동작하는 연결지향형 데이터 전송 소켓

```
int tcp_socket=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

- UDP 소켓

  IPv4 인터넷 프로토콜 체계에서 동작하는 비 연결지향형 데이터 전송 소켓

```
int udp_socket=socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

첫 번째, 두 번째 인자로 전달된 정보를 통해서 소켓의 프로토콜이 사실상 결정되기 때문에 세 번째 인자로 0을 전달해도 된다!

각각의 상수값이 0이라는 뜻은 아님

TCP 소켓의 예

전송되는 데이터의 경계(boundary)가 존재하지 않음을 확인하자!

```
if(bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");
if(listen(serv_sock, 5)==-1)
    error_handling("listen() error");
clnt_addr_size=sizeof(clnt_addr);
clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
if(clnt_sock==-1)
    error_handling("accept() error");
write(clnt_sock, message, sizeof(message));
close(clnt_sock);
close(serv_sock);
```

그냥 메세지 넣으면 감
1번에 갈 수 있다는 것이
경계가 없다는 뜻

/

tcp_server.c의 데이터 전송

```
if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("connect() error!");
while(read_len=read(sock, &message[idx++], 1))
{
    if(read_len==-1)
    {
        error_handling("read() error!");
        break;
    }
    str_len+=read_len;
}
printf("Message from server: %s \n", message);
printf("Function read call count: %d \n", str_len);
```

실행결과

```
root@my_linux:/tcpip# gcc tcp_client.c -o hclient
root@my_linux:/tcpip# ./hclient 127.0.0.1 9190
Message from server: Hello World!
Function read call count: 13
```

tcp_client.c의 데이터 수신