13주차 3차시 UDP 기반 프로그래밍

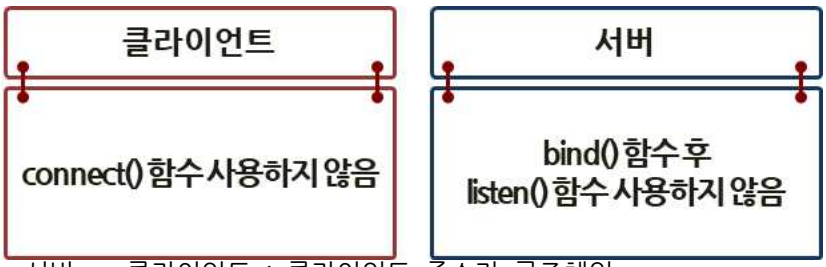
【학습목표】

1. UDP 기반 프로그래밍의 원리를 설명할 수 있다.
2. UDP 기반 서버/클라이언트를 설명할 수 있다.

학습내용1 : UDP 기반 프로그래밍

1. 개념

* 클라이언트로부터 사전에 연결 요청을 받지 않음

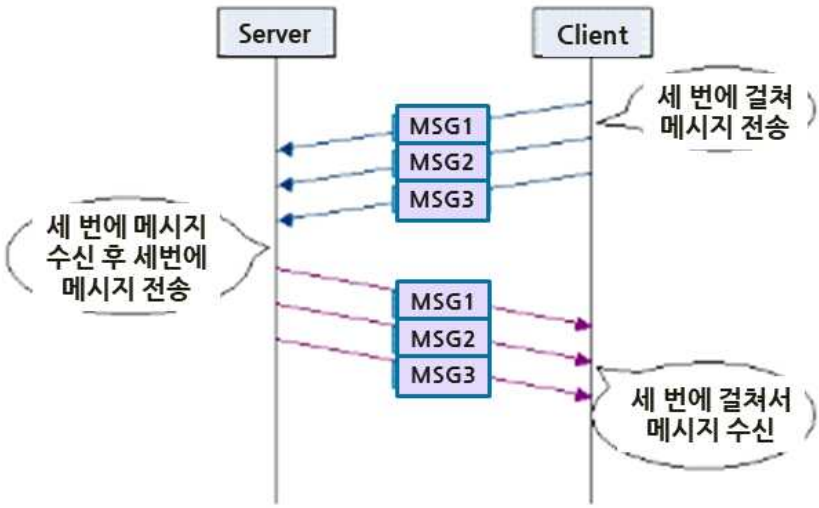


- 서버 → 클라이언트 : 클라이언트 주소가 구조체임

① IP를 기반으로 데이터를 전송함 (TCP와 공통점)
② 흐름제어(flow control)을 하지 않기 때문에 데이터 전송을 보장 받지 못함
③ 연결설정 및 연결 종료 과정도 존재하지 않음
④ 연결 상태가 존재하지 않음

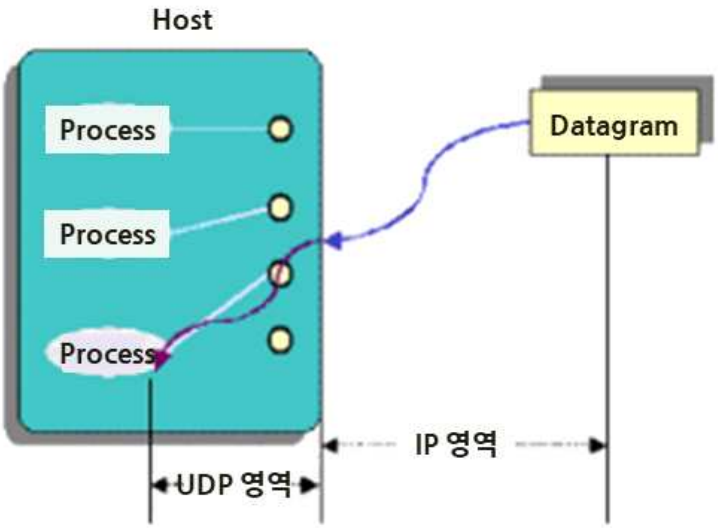
2. 특징

* 데이터의 경계(boundary)가 존재하는 UDP 소켓
- UDP 소켓은 데이터를 송수신하는데 필요한 함수 호출의 수를 정확히 일치 시켜야 함



3. UDP의 역할

* 포트 정보에 의한 프로세스의 구분
- UDP 패킷 = 데이터 그램(Datagram)



* 소켓을 생성하고 bind까지만 하면 바로 입출력함수를 사용하면 됨
- socket~bind : 서버, socket : 클라이언트
- 일반적으로 연결 설정 과정을 거치지 않음

* 데이터를 주고 받기 위한 소켓(우체통에 비유)은 하나만 생성해도 됨 (패킷은 우편)
- UDP는 비연결 프로토콜이기 때문에 주소 정보를 알아야 함

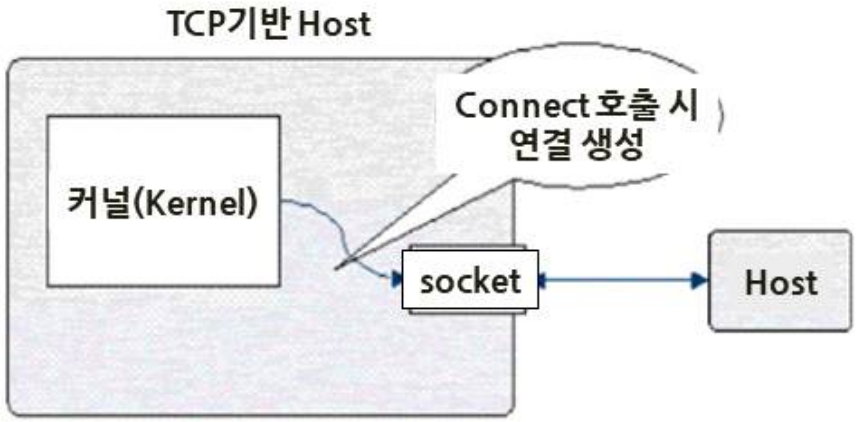
4. connect 함수 호출을 통한 성능의 향상

① TCP 소켓에서의 connect 함수의 의미
- IP와 포트의 할당
- Three-way handshaking

② UDP 소켓에서의 connect 함수의 의미
- IP와 포트의 할당
- connect함수가 없으면 sendto 함수가 제일 처음 호출되는 시점에 IP와 Port가 할당됨
- 클라이언트에서 IP와 Port가 한 번 할당되면 close()함수 호출시까지 변하지 않음

5. TCP/UDP 소켓에서 공통적으로 가지는 connect의 의미

* connect함수의 의미
- 커널과 소켓이 논리적으로 연결하고 그것을 유지함



* connect 함수 호출이 주는 이점
- 데이터를 주고 받는 속도가 빨라짐
- TCP 소켓 기반의 데이터 입출력 함수를 그대로 사용할 수 있음

6. 데이터 전송함수

```
int sendto(int sock, const void* msg, int len, unsigned flags, const struct sockaddr
*addr, int addrlen);
int sendto(SOCKET s, const char FAR *buf, int len, int flags, const struct sockaddr FAR
*to, int tolen);
```

- 1~4번째 인자는 TCP send함수와 비슷
- addr : 목적지 주소 정보
- addrlen : 목적지 주소 정보 구조체 크기

7. 데이터 수신함수

```
int recvfrom(int sock, void *buf, int len, unsigned flags, struct sockaddr *addr, int
*addrlen);
int recvfrom(SOCKET s, char FAR *buf, int len, int flags, struct sockaddr FAR *from,
int FAR *fromlen);
```

- 1~4번째 인자는 TCP recv함수와 비슷
- addr : 목적지 주소 정보
- addrlen : 목적지 주소 정보 구조체 크기

학습내용2 : UDP 기반 서버/클라이언트

1. UDP 프로그래밍 - 서버

```
...
09  #define PORTNUM 9005
10
11  int main(void) {
12      char buf[256];
13      struct sockaddr_in sin, cli;
14      int sd, clientlen = sizeof(cli);
15
16      if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
25
26      if (bind(sd, (struct sockaddr *)&sin, sizeof(sin))) {
27          perror("bind");
28          exit(1);
29      }
```

포트번호

소켓 생성(데이터그램)

소켓 주소 구조체 생성

소켓기술자와 소켓 주소 구조체 연결

```
31      while (1) {
32          if ((recvfrom(sd, buf, 255, 0,
33                  (struct sockaddr *)&cli, &clientlen)) == -1) {
34              perror("recvfrom");
35              exit(1);
36          }
37          printf("** From Client : %s\n", buf);
38          strcpy(buf, "Hello Client");
39          if ((sendto(sd, buf, strlen(buf)+1, 0,
40                  (struct sockaddr *)&cli, sizeof(cli))) == -1) {
41              perror("sendto");
42              exit(1);
43          }
44      }
45
46      return 0;
47  }
```

클라이언트의 메시지 수신

클라이언트로 데이터 보내기

2. UDP 프로그래밍 - 클라이언트

```
...
09  #define PORTNUM 9005
10
11  int main(void) {
12      int sd, n;
13      char buf[256];
14      struct sockaddr_in sin;
15
16      if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
17          perror("socket");
18          exit(1);
19      }
20
21      memset((char *)&sin, '\0', sizeof(sin));
22      sin.sin_family = AF_INET;
23      sin.sin_port = htons(PORTNUM);
24      sin.sin_addr.s_addr = inet_addr("192.168.162.133");
25
26      strcpy(buf, "I am a client.");
27      if (sendto(sd, buf, strlen(buf)+1, 0,
28                  (struct sockaddr *)&sin, sizeof(sin)) == -1) {
29          perror("sendto");
30          exit(1);
31      }
```

포트번호

소켓 생성

소켓 주소 구조체 생성

서버에 메시지 전송

```
33      n = recvfrom(sd, buf, 255, 0, NULL, NULL);
34      buf[n] = '\0';
35      printf("** From Server : %s\n", buf);
36
37      return 0;
38  }
```

서버가 보낸 데이터 읽기

서버

```
# ex12_5s.out
** From Client : I am a client.
```

클라이언트

```
# ex12_5c.out
** From Server : Hello Client
```

【학습정리】

1. UDP 기반 프로그래밍

* 개념

- 클라이언트로부터 사전에 연결 요청을 받지 않는다.
- 클라이언트 : connect() 함수 사용하지 않음.
- 서버 : bind() 함수 후 listen() 함수 사용하지 않음.
- 서버 → 클라이언트 : 클라이언트 주소가 구조체임
- IP를 기반으로 데이터를 전송한다. (TCP와 공통점)
- 흐름제어(flow control)을 하지 않기 때문에 데이터 전송을 보장 받지 못한다.
- 연결설정 및 연결 종료 과정도 존재하지 않는다.
- 연결 상태가 존재하지 않는다.

* 특징

- 데이터의 경계(boundary)가 존재하는 UDP 소켓
- UDP 소켓은 데이터를 송수신하는데 필요한 함수 호출의 수를 정확히 일치 시켜야 한다.

2. UDP 기반 프로그래밍

* UDP의 역할

- 포트 정보에 의한 프로세스의 구분
- UDP 패킷 = 데이터 그램(Datagram)

3. UDP 기반 서버/클라이언트

* 구현

- 소켓을 생성하고 bind까지만 하면 바로 입출력함수를 사용하면 된다.
- (socket~bind : 서버, socket : 클라이언트)
- 일반적으로 연결 설정 과정을 거치지 않는다.
- 데이터를 주고 받기 위한 소켓(우체통에 비유)은 하나만 생성해도 된다. (패킷은 우편)
- UDP는 비연결 프로토콜이기 때문에 주소 정보를 알아야 한다.

* connect 함수 호출을 통한 성능의 향상

- TCP 소켓에서의 connect 함수의 의미 : IP와 포트의 할당, Three-way handshaking

* UDP 소켓에서의 connect 함수의 의미

- IP와 포트의 할당
- connect함수가 없으면 sendto 함수가 제일 처음 호출되는 시점에 IP와 Port가 할당된다.
- 클라이언트에서 IP와 Port가 한 번 할당되면 close()함수 호출시까지 변하지 않는다.

4. 데이터 전송함수

- 1~4번째 인자는 TCP send함수와 비슷.
- addr : 목적지 주소 정보
- addrlen : 목적지 주소 정보 구조체 크기

```
int sendto(int sock, const void* msg, int len, unsigned flags, const struct sockaddr
*addr, int addrlen);
int sendto(SOCKET s, const char FAR *buf, int len, int flags, const struct sockaddr FAR
*to, int tolen);
```

5. 데이터 수신함수

- 1~4번째 인자는 TCP recv함수와 비슷.
- addr : 목적지 주소 정보
- addrlen : 목적지 주소 정보 구조체 크기

```
int recvfrom(int sock, void *buf, int len, unsigned flags, struct sockaddr *addr, int
*addrlen);
int recvfrom(SOCKET s, char FAR *buf, int len, int flags, struct sockaddr FAR *from,
int FAR *fromlen);
```