5. 소켓 옵션

- 소켓 옵션을 변경하는 방법
- 멀티캐스팅 패킷을 송수신하는 방법
- 멀티캐스트를 이용한 채팅 프로그램

5.1 소켓 옵션 변경

소켓 옵션 변경

- **소켓 옵션을 알아내는 함수와 옵션을 변경시키는 함수**

```
int setsockopt(int s, int level, int opt, const char *optval, int optlen);
int getsockopt(int s, int level, int opt, const char *optval, int *len);
```

  - s : **소켓번호**
  - level : **프로토콜 레벨**
    - SOL_SOCKET : **소켓의 일반적인 옵션 변경**
    - IPPROTO_IP : IP **프로토콜에 관한 옵션 변경**
    - IPPROTO_TCP : TCP**에 관한 옵션 변경**
  - opt : **사용하고자 하는 옵션**
  - optval : **옵션 지정에 필요한 값의 포인터**
  - optlen : optval**의 크기**

- **연결형 소켓의 옵션이** accept()**가 리턴하는 통신소켓에 상속**
  - **통신소켓에 필요한 옵션지정은** accept()**를 호출하기 전에 설정**

소켓 옵션의 종류

- **옵션 레벨**
  - SOL_SOCKET : **소켓 레벨의 옵션을 변경**
  - IPPROTO_IP : IP **레벨의 옵션을 변경**
  - IPPROTO_TCP : TCP **레벨의 옵션을 변경**

SOL_SOCKET 레벨의 옵션

- SO_BROADCAST : 방송형 메시지 전송 허용
- SO_DEBUG : DEBUG 모드를 선택
- SO_REUSEADDR : 주소 재사용 선택
- SO_LINGER : 소켓을 닫을 때 미전송된 데이터가 있어도 지정된 시간만큼 기다렸다가 소켓을 닫음
- SO_KEEPALIVE : TCP의 keep-alive 동작 선택
  - keep-alive : TCP 연결이 정성적으로 지속되고 있는지를 주기적으로 확인해 주는 기능
- SO_OOBINLINE : OOB 데이터를 일반 데이터처럼 읽음
  - Out-of-band 데이터 : 긴급한 데이터로서 일정한 순서를 가지지 않고 순차화 된 데이터보다 더 먼저 처리 되어야 할 필요가 있는 데이터
- SO_RCVBUF : 수신버퍼의 크기 변경
- SO_SNDBUF : 송신버퍼의 크기 변경

IPPROTO_IP 레벨의 옵션

- IP_TTL
  - Time To Live **변경**
- IP_MULTICAST_TTL
  - **멀티캐스트 데이터그램의** TTL **변경**
- IP_ADD_MEMBERSHIP
  - **멀티캐스트 그룹에 가입**
- IP_DROP_MEMBERSHIP
  - **멀티캐스트 그룹에서 탈퇴**
- IP_MULTICAST_LOOP
  - **멀티캐스트 데이터그램의** loopback **허용 여부**
- IP_MULTICAST_IF
  - **멀티캐스트 데이터그램 전송용 인터페이스 지정**

IPPROTO_TCP 레벨의 옵션

- TCP_KEEPALIVE
  - keep-alive **확인 메시지 전송 시간 지정**
- TCP_MAXSEG
  - TCP**의** MSSS(**최대 메시지 크기**) **지정**
- TCP_NODELAY
  - Nagle **알고리즘의 선택**
  - Nagle **알고리즘은 작은 패킷을 묶어 보내 네트워크의 부하를 줄임**

소켓 옵션 변경 예제

- IP 레벨의 TTL값을 출력하고 32로 변경하는 코드

```
int ttl, optlen;

getsockopt(s, IPPROTO_IP, IP_TTL, &ttl, &optlen);
printf("TTL = %d\n", ttl);

ttl=32;
setsockopt(s, IPPROTO_IP, IP_TTL, &ttl, sizeof(ttl));
printf("TTL = %d\n", ttl);
```

- 어떤 상태를 선택 또는 취소를 택하는 옵션은 optval 값이 1이면 선택이고 0이면 취소

```
Int  set =1;
setsockopt( s, SOL_SOCKET, SO_KEEPALIVE, &set, sizeof(set));
```

- 예제 : TCP 소켓을 개설하고 디폴트 수신버퍼 크기를 확인 후 임의의 크기(1024)로 변경(chg_rcvbuf.c)
  - SO_RCVBUF 옵션을 사용

5.2 소켓 옵션 종류

- 여러 소켓 옵션 사용방법에 대해 알아본다.

소켓 옵션의 종류 – SO_KEEPALIVE

- SO_KEEPALIVE
  - TCP 연결이 정상적으로 지속되고 있는지를 주기적으로 확인하는 기능
  - 이 옵션이 셋트되어 있으면 TCP는 확인시간(예:2시간) 동안 데이터 송수신이 없을 때 TCP 연결이 살아 있는지 물어보는 질문(keep alive prove)를 전송
  - Keep Alive Prove의 세 가지 응답
    - 상대방이 ACK를 보낸다.
      - TCP 연결이 정상적으로 동작
    - 상대방이 RST 에러를 보낸다.
      - 상대방 호스트가 꺼진 후 재부팅 된 상태
      - TCP는 소켓을 닫으며 에러코드는 ECONNRESET
    - 아무 응답이 없다.
      - 질문을 몇 번 더 보내본 후 연결 종료
      - 버클리 유닉스의 경우 75초 간격으로 8번의 질문 후 종료(에러코드는 ETIMEDOUT)
  - 중간 라우터의 응답
    - ICMP 메시지로 host unreachable error, network unreachable error를 보내면 EHOSTENREACH, ENETURNREACH가 리턴
  - 이 옵션은 디폴트로 동작하지 않는 것으로 설정되어 있다.

소켓 옵션의 종류 – SO_LINGER

- SO_LINGER
  - close() 호출시 송신버퍼에 데이터가 남아 있어도 close()는 즉시 리턴되고 TCP는 이 데이터를 모두 전송한 후 실제 연결을 종료
  - close() 함수가 리턴한 시점에서 모든 데이터가 전송된 것을 보장하지는 않음
  - SO_LINGER 옵션은 close()를 호출한 후 상대방에서 정상적으로 종료 절차가 이루어 졌는지를 확인하기 위해 사용
    - close()는 지정한 linger 시간 동안 또는 정상 종료 시까지 블록됨
    - 정상 종료전에 linger 시간이 타임아웃되면 ETIMEDOUT 에러 발생
  - 예 : 송신버퍼에 데이터가 남아있어도 타임 아웃되면 데이터를 폐기

```
struct linger { int l_onoff=1; int l_linger; } //l_onoff : linger 옵션 사용 여부 지정
//l_linger : 기다리는 시간을 지정, 0이면 송신버퍼에 남아있는 데이터를 파기, 양수이면
//close()는 남아있는 데이터를 모두 전송하고 ACK 수신 혹은 지정시간까지 블록
struct linger ling;
ling.l_onoff=1; ling.l_linger =0;
if(setsockopt(socks, SOL_SOCKET, SO_LINGER,
                    (void*)&ling, sizeof(struct linger)) != 0){
            perror("setsockopt fail ");
            exit(1);
}
```

소켓 옵션의 종류 – SO_LINGER

- 4-way handshake
  - FIN, ACK, FIN, ACK**의 순서로 동작**
  - **호스트** A**가** B**로 데이터 전송 후** close()**를 호출한 시점에서** B**가 모든 데이터를 읽었다고 확신할 수 없음**
- shutdown()
  - **상대방이** FIN**을 보낼 때 까지 기다리도록 함**

```
int shutdown(int s, int how);
```

  - how
    - SHUT_WR : **송신 스트림만 닫음**
    - SHUT_RD : **수신 스트림만 닫음**
    - SHUT_RDWR : **송수신 스트림을 보두 닫음**(close()
  - **상대방이** FIN**을 보낼 때까지 기다리는 예제 코드**

```
write(sock, buf, size); // 상대방에게 데이터를 전송
shutdown(sock, SHUT_WR); // 상대방이 FIN에 대한 ACK를 보낼 때까지 블록
while (read(sock, buf, size) > 0);// 상대방이 FIN을 보내면 read가 0을 리턴
shutdown(sock, SHUT_RD); //상대방의 FIN에 대한 ACK를 보내고 스트림을 close
```

소켓 옵션의 종류 – SO_RCVBUF, SO_SNDBUF

- TCP, UDP는 **송신버퍼와 수신버퍼를 가짐**
  - TCP**의 경우** write() **호출시 데이터를 송신 버퍼로 복사**
  - **데이터가 송신버퍼에 모두 복사되면 시스템이 데이터를 전송**
  - **전송 데이터는 유지하고 있다가** ACK**를 수신 후 삭제**
  - **송신버퍼가 가득차면** write()**는 블록**
  - **송신/수신버퍼의 크기를 사용자가 지정할 수 있음**
- SO_SNDBUF
  - **송신 버퍼의 크기 확인 및 지정**
- SO_RCVBUF
  - **수신 버퍼의 크기 확인 및 지정**
- **송신/수신 버퍼의 크기 지정 방법**
  - 3-way handshake **후에는 버퍼 크기 변경이 불가**
    - **서버의 경우** listen() **호출 이전에 설정**
    - **클라이언트의 경우** connect() **호출 이전에 설정**
- TCP**는 흐름제어를 위해 수신버퍼의 여유공간(윈도우)을 상대방에 알린다**
  - **수신버퍼의 여유공간보다 큰 데이터의 초과 부분은 버려진다**.

소켓 옵션의 종류 – SO_REUSEADDR

- **동일한 소켓주소를 여러 프로세스 또는 한 프로세스 내의 여러 소켓에서 중복 사용을 허용하는 옵션, 주로 서버에서 사용**

```
int set = 1;
setsockopt(udp_sock1, SOL_SOCKET, SO_REUSEADDR, (void*)&set, sizeof(int));
```

  - **소켓 주소 재사용 옵션은** bind() **호출 이전에 설정해야 함**
- **소켓주소 재사용 옵션이 필요한 경우**
  - TIME-WAIT **상태에서의 주소 재사용**
    - ACTIVE CLOSE**상태의 호스트는** TIME-WAIT **상태를 가지며 포트번호를 중복하여 사용할 수 없음**
  - **자식 프로세스가 서버인 경우**
    - **부모 프로세스가 종료된 후 다시 시작하면 "포트번호 사용중" 에러가 발생**
  - **멀티홈 서버의 경우**
    - **호스트가 두 개 이상의** IP**주소를 가지며 같은 포트번호를 사용해야 하는 경우가 있음**
  - **완전 중복 바인딩**
    - **동일한** IP**주소와 포트번호를 중복하여** bind()**하는 것,** UDP**에서만 가능**
    - **예 : 한 호스트에서 여러 멀티캐스트 가입자들이 데이터를 수신하는 경우**

기타 옵션

- SO_OOBINLINE
  - **상대방이 보낸 대역외**(OOB:Out of Band) **데이터를 일반 데이터의 수신버퍼에 같이 저장되도록 하여 동등한 순서로 처리**
- SO_RCVLOWAT, SO_SNDLOWAT
  - **송수신버퍼의 최소량 설정**(low-watermark) **옵션**
  - select() **함수의 읽기 조건 리턴은 최소** 1**바이트**
    - **최소량 데이터 크기 변경시** SO_RCVLOWAT **옵션 사용**
  - select() **함수의 쓰기 조건 리턴은 최소** 2048**바이트**
    - **최소량 데이터 크기 변경시** SO_SNDLOWAT **옵션 사용**
- SO_DONTROUTE
  - **데이터그램 송신시에 시스템이 배정하는 라우팅 경로를 사용하지 않도록 함**
  - send(), sendto(), sendmsg() **호출시에 사용할 수 있음**
- SO_BROADCAST
  - **소켓으로 방송 메시지를 보낼 수 있는지를 지정**(UDP**에서만 사용**)

기타 옵션

- TCP_NODELAY
  - TCP에서는 기본적으로 Neagle's 알고리즘이 실행
  - TCP_NODELAY 옵션은 Neagle's 알고리즘 실행을 중지
  - Neagle's 알고리즘
    - ACK를 받기 전까지 작은 크기의 데이터(MSS 보다 작은 데이터)는 전송하지 않고 기다리는 알고리즘
    - ACK의 도착이전에 발생한 작은 데이터를 모아서 전송하여 너무 작은 데이터가 자주 전송하지 않게 하여 전송 효율을 높이는 알고리즘
- TCP_MAXSEG
  - TCP의 최대 세그먼트 크기(MSS) 값을 읽거나 변경
  - MSS 변경은 모든 시스템이 지원하지 않음(LINUX는 지원)
  - MSS는 연결 설정 전에 해야 함
  - 너무 큰 값을 선택하면 연결 설정 과정에서 원하는 값보다 작게 지정될 수도 있음

5.3 멀티캐스트 프로그래밍

멀티캐스트

- 멀티캐스트
  - 하나의 데이터그램을 다수의 호스트로 동시에 전송
    - 하나의 데이터그램은 네트워크 내에서 복제되어 다수의 호스트로 전송
  - LAN 뿐만 아니라 인터넷에 연결되어 있는 모든 호스트로 전송 가능
    - 단 호스트는 해당 멀티캐스트 그룹에 가입되어 있어야 함
- 멀티캐스트 그룹
  - 멀티캐스트 데이터그램을 수신하고자 하는 호스트들의 집합
  - 클래스 D의 IP주소를 그룹 주소로 사용
    - 224.0.0.0부터 239.255.255.255 사이의 값을 가짐
  - 해당 그룹 주소를 목적지로 하여 UDP 데이터그램을 전송
  - 중간 라우터가 MRP(Multicast Routing Protocol)을 지원해야 함
- 멀티캐스트 데이터그램 송수신
  - 호스트가 멀티캐스트 그룹에 가입하면 인접한 라우터에 IGMP 메시지를 보내 가입한 사실을 알려준다.
  - 어떤 호스트에서 해당 멀티캐스트 그룹으로 데이터그램을 보내면 라우터들은 MRP를 이용하여 그룹의 가입자들에게 전달
  - 이더넷에서 멀티캐스트 전송
    - 48비트 MAC 주소 사용 : 24비트(0x01005e)+멀티캐스트 그룹 IP 주소

소켓 옵션 지정

- **멀티캐스트와 관련된 소켓 옵션**
  - IP_ADD_MEMBERSHIP : **그룹에 가입**
  - IP_DROP_MEMBERSHIP : **그룹에서 탈퇴**
  - IP_MULTICAST_LOOP : **멀티캐스트 패킷의** loopback **허용 여부**
  - IP_MULTICAST_TTL : **패킷의** TTL**값 지정**
  - IP_MULTICAST_IF : **패킷 전송을 위한 인터페이스 지정**

- **멀티캐스트 그룹 가입/탈퇴**
  - ip_mreq **구조체를 사용**

```
struct ip_mreq {
  struct in_addr imr_multiaddr;
  struct in_addr imr_interface;              // 자신의 인터페이스(IP주소)
}
```

멀티캐스트 데이터그램 수신

- **수신을 위해서는 그룹에 가입해야 함**

```
struct sockaddr_in mcast_group;
struct ip_mreq mreq;

// 멀티캐스트 그룹 주소 지정
memset(&mcast_group, 0, sizeof(mcast_group));
mcast_group.sin_family = AF_INET;
mcast_groupt.sin_port = htons(atoi(argv[2]));
mcast_group.sin_addr.s_addr = inet_addr(argv[1]);

// ip_mreq 구조체 지정
mreq.imr_multiaddr = mcast_group.sin_addr;
mreq.imr_interface.s_addr = htonl(INADDR_ANY);

// 멀티캐스트 그룹 가입
recv_s = socket(AF_INET, SOCK_DGRAM, 0); //수신용 UDP 소켓 개설
setsockopt(recv_s, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq));
// 주소 재사용 옵션 지정
unsigned int set = 1;
setsockopt(recv_s, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set));
// 수신용 소켓과 멀티캐스트 소켓주소를 bind
bind(recv_s, (struct sockaddr*)&mcast_group, sizeof(mcast_group));
```

멀티캐스트 데이터그램 송신

- 송신자는 멀티캐스트 그룹에 가입하지 않아도 됨
- 해당 멀티캐스트 주소로 패킷을 전송하기

```
send_s = socket(AF_INET, SOCK_DGRAM, 0); // 송신용 소켓 개설
...
char msg[MAXLINE+1]; // 전송할 메시지
...
sendto(send_s, msg, strlen(msg), 0, (struct sockaddr*)&mcast_group,
        sizeof(mcast_group));
```

- 멀티캐스트 송신 호스트가 멀티캐스트 그룹에 가입했을 경우
  - 패킷은 기본적으로 자신에게 loopback된다.
  - 소켓 옵션을 이용해서 취소할 수 있다.

```
int no = 0;
setsockopt(send_s, IPPROTO_IP, IP_MULTICAST_LOOP, &no, sizeof(no));
```

- IP_MULTICAST_IF 옵션은 멀티캐스트 데이터그램을 전송할 IP 주소를 호스트의 디폴트 IP 주소가 아닌 다른 주소로 지정 시 사용
  - Multihomed 호스트 인 경우
- 멀티캐스트 채팅 프로그램(multicast.c)

5.4 send()와 recv()의 옵션 지정

- **소켓 옵션을 한번의 송수신에만 적용시키는 방법**
  - Recv(), send() **함수의** flag **인자를 사용**

MSG_PEEK, MSG_DONTWAIT, MSG_WAITALL

- MSG_PEEK
  - recv()**가 데이터를 읽기만 하고 버퍼에서 삭제시키지 않게 하는 플래그**
  - **수신버퍼에 현재 데이터 양을 알아볼 때 사용**.
- MSG_DONTWAIT
  - **수신버퍼에 데이터가 없을 경우 데이터가 도착할 때까지** recv()**가 무한히 기다리는 것을 방지하는 플래그**
  - **이 플래그를 설정하면** recv()**를 바로 리턴시킨다**.
- **소켓** s**의 수신버퍼에 도착한 데이터 양을 알아보는 예**

```
n = recv(s, buf, sizeof(buf), MSG_PEEK | MSG_DONTWAIT);
```

- MSG_WAITALL
  - Recv() **호출시 지정된 크기의 데이터를 수신하기 전까지는 리턴하지 않게하는 플래그**

```
Recv(fd, ptr, n, MSG_WAITALL);
```

- MSG_OOB
  - **대역외 데이터를 송수신할 때사용**