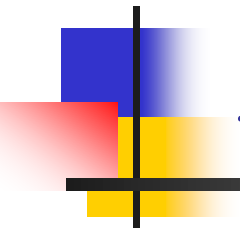
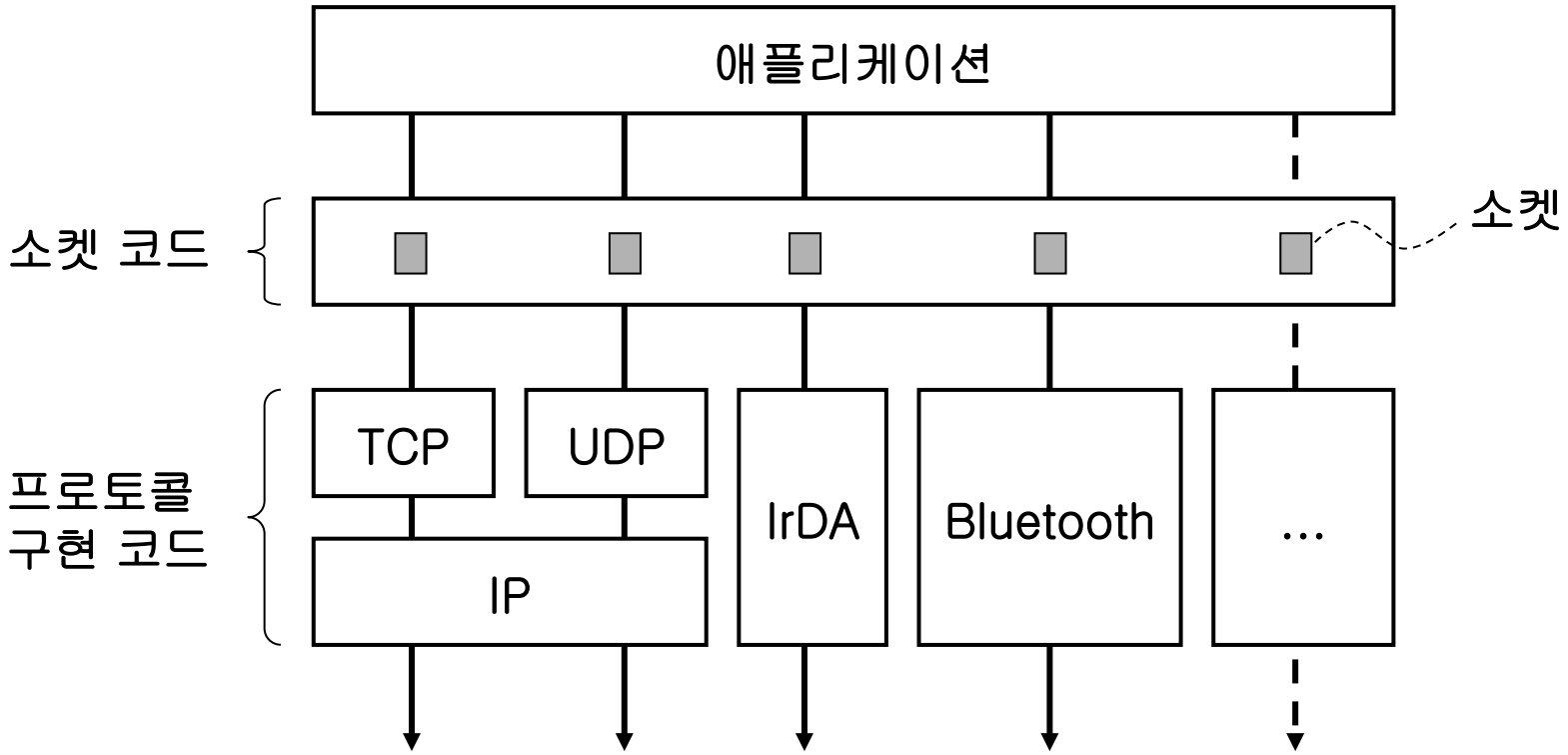


소켓옵션

홍익대학교 게임소프트웨어전공
편집: 김 혜영

개요 (1/7)

- 소켓 프로그래밍 모델



개요 (2/7)

- 소켓 옵션(socket options)
  - 소켓 함수의 기본 동작을 변경
    - 소켓 코드와 프로토콜 구현 코드에 대한 세부적인 제어 가능
- 소켓 옵션의 종류

① 소켓 코드가 담당하는 부분

- 옵션을 설정하면 소켓 코드에서 해석하고 처리함

② 프로토콜 구현 코드가 담당하는 부분

- 옵션을 설정하면 프로토콜 구현 코드에서 해석하고 처리함

개요 (3/7)

- 소켓 옵션 설정하기

```
int setsockopt (
    SOCKET s,
    int level,
    int optname,
    const char* optval,
    int optlen
) ;
```

성공: 0, 실패: SOCKET_ERROR

개요 (4/7)

- 소켓 옵션 얻기

```
int getsockopt (
    SOCKET s,
    int level,
    int optname,
    char* optval,
    int* optlen
) ;
```

성공: 0, 실패: SOCKET_ERROR

Ioctlsocket()

ioctlsocket

ioctlsocket 함수는 소켓의 입출력 모드를 제어하는 함수입니다.

```
int ioctlsocket (
        SOCKET   s,
        long   cmd,
        u_long FAR*   argp
);
```

Parameters

s

작업대상 소켓의 기술자(descriptor)를 명시.

cmd

소켓 s가 수행할 컴맨드(command)

argp

command에 대한 입/출력 파라메터로 사용.

FIONBIO

argp 매개변수가 0이 아닐 경우 소켓 s의 넌블럭킹(비동기) 모드를 활성화

argp 매개변수가 0일 경우는 넌블럭킹(비동기) 모드는 비활성화 됩니다. *argp* 매개변수는 unsigned long 값을 포인트.

FIONREAD

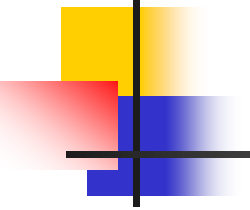
:네트웍 입력 버퍼에서 기다리고 있는, 소켓 *s*로부터 읽을 수 있는 데이터의 크기(amount)를 얻어내는데 사용

argp 매개변수는 데이터의 크기를 의미하는 unsigned long 형태로 포인트

(- 만약 *s* 매개변수가 연결지향형(stream oriented) 소켓(예:SOCK_STREAM) 경우 FIONREAD 컴맨드에 의한 ioctlsocket 함수의 호출은 recv 함수의 호출로 수 있는 데이터의 크기(amount)를 반환)

-만약 소켓이 메시지 지향형(message oriented) 소켓(예:SOCK_DGRAM) 일

- FIONREAD 컴맨드는 소켓에 큐된 첫 번째 데이터그램의 크기를 반환



SIOCATMARK

:소켓으로부터 out-of-band 데이터가 모두 읽혀졌는지를 판단하기 위해 사용됩니다.
argp 파라메터는 반환값이 저장된 boolean 로 포인트
읽혀지기를 원하는 out-of-band 데이터가 없을 경우 TRUE가 반환되고,
그렇지 않은 경우 FALSE가 반환됨.
(setsockopt 함수로 SO_OOBINLINE 옵션을 설정한 SOCK_STREAM 타입의 스트림 소켓에만 적용될 수 있음)

개요 (5/7)

- 소켓 옵션 – SOL_SOCKET

optname	optval 타입	get	set	설명
SO_BROADCAST	BOOL	•	•	브로드캐스팅 허용
SO_DONTROUTE	BOOL	•	•	데이터 전송시 라우팅 테이블 참조 과정 생략
SO_KEEPALIVE	BOOL	•	•	주기적으로 연결 여부 확인
SO_LINGER	struct linger{}	•	•	보낼 데이터가 있을 경우 closesocket() 함수 리턴 지연
SO_SNDBUF SO_RCVBUF	int	•	•	소켓 송/수신 버퍼 크기 설정
SO_SNDTIMEO SO_RCVTIMEO	int	•	•	send(), recv() 등의 함수에 대한 타임아웃 설정
SO_REUSEADDR	BOOL	•	•	지역 주소(IP 주소, 포트 번호) 재사용 허용

개요 (6/7)

- 소켓 옵션 – IPPROTO_IP

optname	optval 타입	get	set	설명
IP_HDRINCL	BOOL	•	•	데이터를 보낼 때 IP 헤더를 포함
IP_TTL	int	•	•	IP 패킷의 TTL(time-to-live) 변경
IP_MULTICAST_IF	IN_ADDR{}	•	•	멀티캐스트 패킷을 보낼 인터페이스 설정
IP_MULTICAST_TTL	int	•	•	멀티캐스트 패킷의 TTL 변경
IP_MULTICAST_LOOP	BOOL	•	•	멀티캐스트 패킷의 루프백 여부 설정
IP_ADD_MEMBERSHIP IP_DROP_MEMBERSHIP	struct ip_mreq{}		•	멀티캐스트 그룹 가입과 탈퇴

개요 (7/7)

- 소켓 옵션 – IPPROTO_TCP

optname	optval 타입	get	set	설명
TCP_NODELAY	BOOL	•	•	Nagle 알고리즘 작동 중지

SO_BROADCAST 옵션

- 용도
  - 해당 소켓을 이용하여 브로드캐스트 데이터 전송 가능
  - UDP 소켓에만 사용 가능

SO_DONTROUTE 옵션

- 용도
  - 데이터 전송시 라우팅 테이블 참조를 생략하고, 곧바로 bind() 함수로 설정한 네트워크 인터페이스로 모든 데이터를 보냄
- 사용 예

```
BOOL optval = TRUE;
if(setsockopt(listen_sock, SOL_SOCKET, SO_DONTROUTE,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

SO_KEEPALIVE 옵션

- 용도
  - TCP 프로토콜 수준에서 연결 여부를 확인하기 위해 상대 TCP에게 주기적으로(약 2시간 간격) TCP 패킷을 보냄
- 사용 예

```
BOOL optval = TRUE;
if(setsockopt(listen_sock, SOL_SOCKET, SO_KEEPALIVE,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

SO_LINGER 옵션 (1/3)

- 용도
  - closesocket() 함수의 디폴트 동작 변경

```
send(sock, ...); // 데이터를 보낸다.
closesocket(sock); // 소켓을 닫는다.
```

- 옵션값

```
struct linger {
    u_short l_onoff; /* option on/off */
    u_short l_linger; /* linger time */
};
typedef struct linger LINGER;
```

SO_LINGER 옵션 (2/3)

- 사용 예

```
LINGER optval;
optval.l_onoff = 1; /* linger on */
optval.l_linger = 10; /* linger time = 10초 */
if(setsockopt(sock, SOL_SOCKET, SO_LINGER,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

SO_LINGER 옵션 (3/3)

- 옵션값에 따른 closesocket() 함수의 동작

struct linger{}		closesocket() 함수 동작	추가 설명
l_onoff	l_linger		
0	사용 안함	①과 동일	closesocket() 함수의 디폴트 동작
1	0	②와 동일	
1	양수	③과 동일	

- ① closesocket() 함수는 곧바로 리턴하고 송신 버퍼의 데이터는 백그라운드로 보낸 후 TCP 연결을 정상 종료

  ② closesocket() 함수는 곧바로 리턴하고 송신 버퍼의 데이터는 삭제한 후 TCP 연결을 강제 종료

  ③ 송신 버퍼의 데이터를 모두 보내고 TCP 연결을 정상 종료한 후 closesocket() 함수 리턴. 일정 시간 내에 송신 버퍼의 데이터를 모두 보내지 못하면 TCP 연결을 강제 종료한 후 closesocket() 함수 리턴. 이때 송신 버퍼에 남은 데이터는 삭제함.

SO_SNDBUF, SO_RCVBUF 옵션

- 용도
  - 소켓의 송신 버퍼와 수신 버퍼 크기 변경
- 사용 예

```
int optval;
int optlen = sizeof(optval);
if(getsockopt(listen_sock, SOL_SOCKET, SO_RCVBUF,
    (char *)&optval, &optlen) == SOCKET_ERROR)
    err_quit("getsockopt()");
printf("수신 버퍼 크기 = %d 바이트₩n", optval);

optval *= 2;
if(setsockopt(listen_sock, SOL_SOCKET, SO_RCVBUF,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
    err_quit("setsockopt()");
```

SO_SNDTIMEO, SO_RCVTIMEO 옵션

- 용도
  - 데이터 전송 함수(send(), recv(), sendto(), recvfrom())가 작업 완료와 상관없이 일정 시간 후 리턴하도록 함
- 사용 예

```
int optval = 3000;
if(setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

SO_REUSEADDR 옵션

- 용도
  - 사용 중인 IP 주소와 포트 번호를 재사용
    - 사용 중인 IP 주소와 포트 번호로 bind() 함수를 (성공적으로) 호출할 수 있음
- 목적

① 서버 종료 후 재실행시 bind() 함수에서 오류가 발생하는 것을 방지

② 두 개 이상의 IP 주소를 가진 호스트에서 각 IP 주소별로 서버를 따로 운용

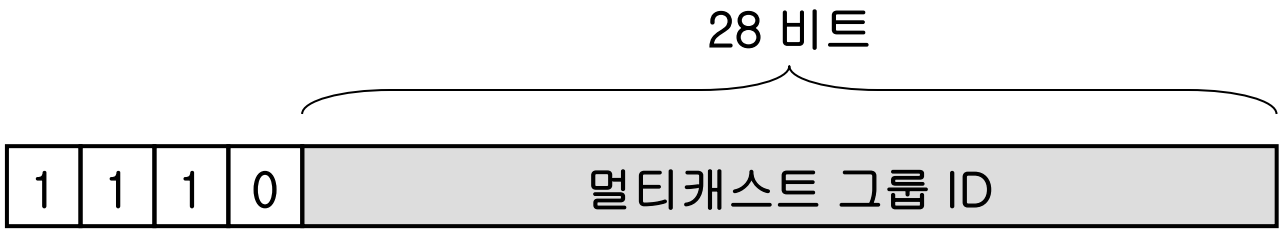
③ 멀티캐스팅 애플리케이션이 동일한 포트 번호를 사용할 수 있도록 함

예제분석 실습

* 본인들의 server에서 다음을 추가하시오.
- 송수신 버퍼의 크기를 3배로 늘림
  (교재 P.242-243참조)

멀티캐스팅 (1/3)

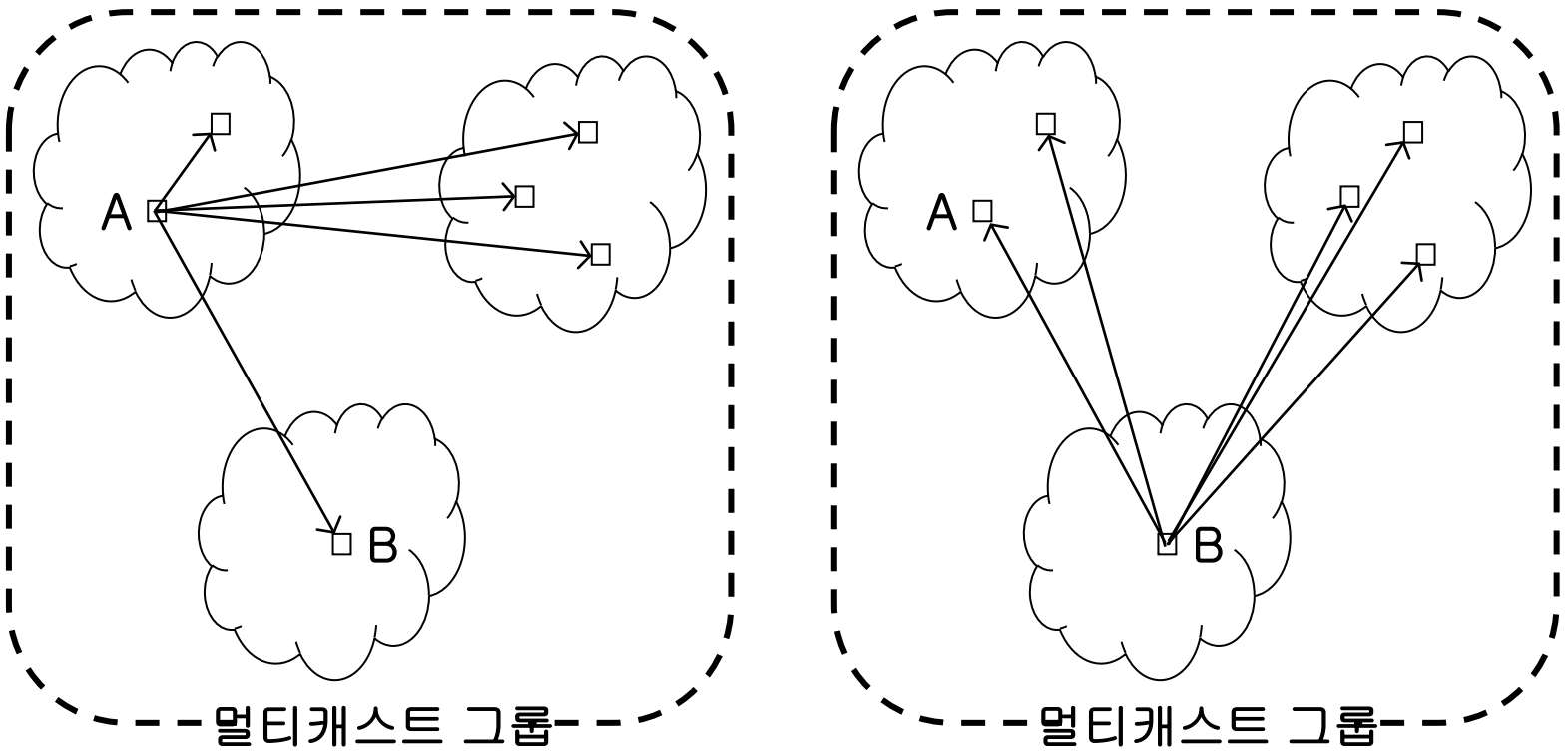
- 멀티캐스트 주소



- 특징
  - 그룹 가입과 탈퇴가 자유롭고, 그룹 구성원 모두가 평등
  - 멀티캐스트 데이터를 받으려면 그룹에 가입해야 함
  - 멀티캐스트 데이터를 보내기 위해 그룹에 가입할 필요는 없음

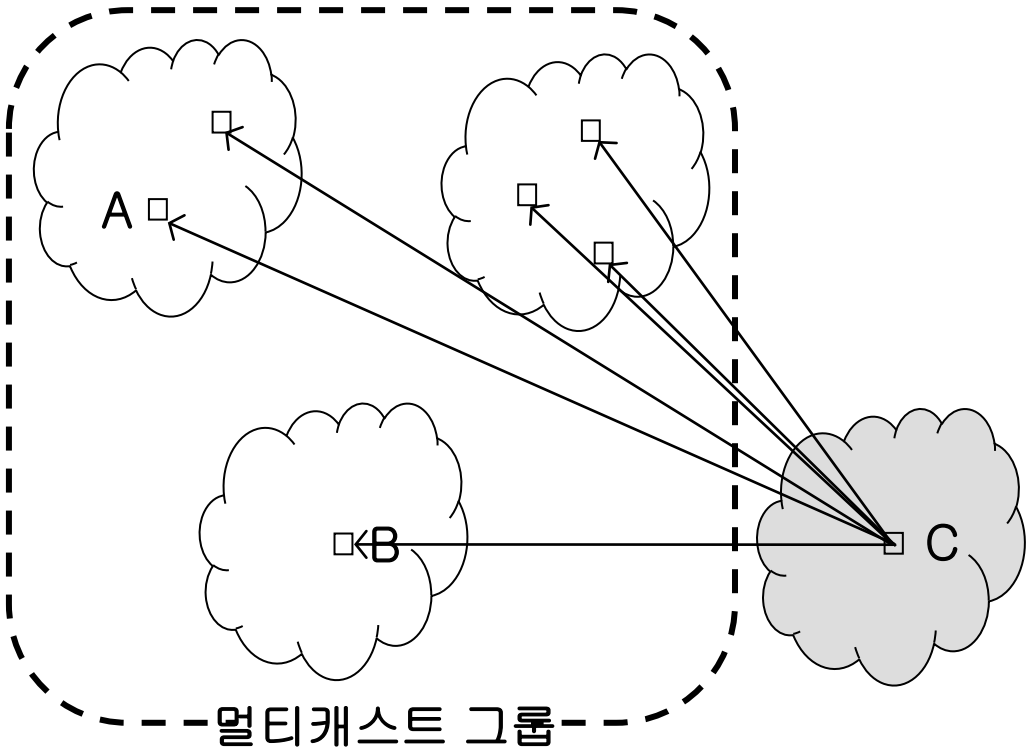
멀티캐스팅 (2/3)

- 멀티캐스트 데이터 전송(1)



멀티캐스팅 (3/3)

- 멀티캐스트 데이터 전송(2)



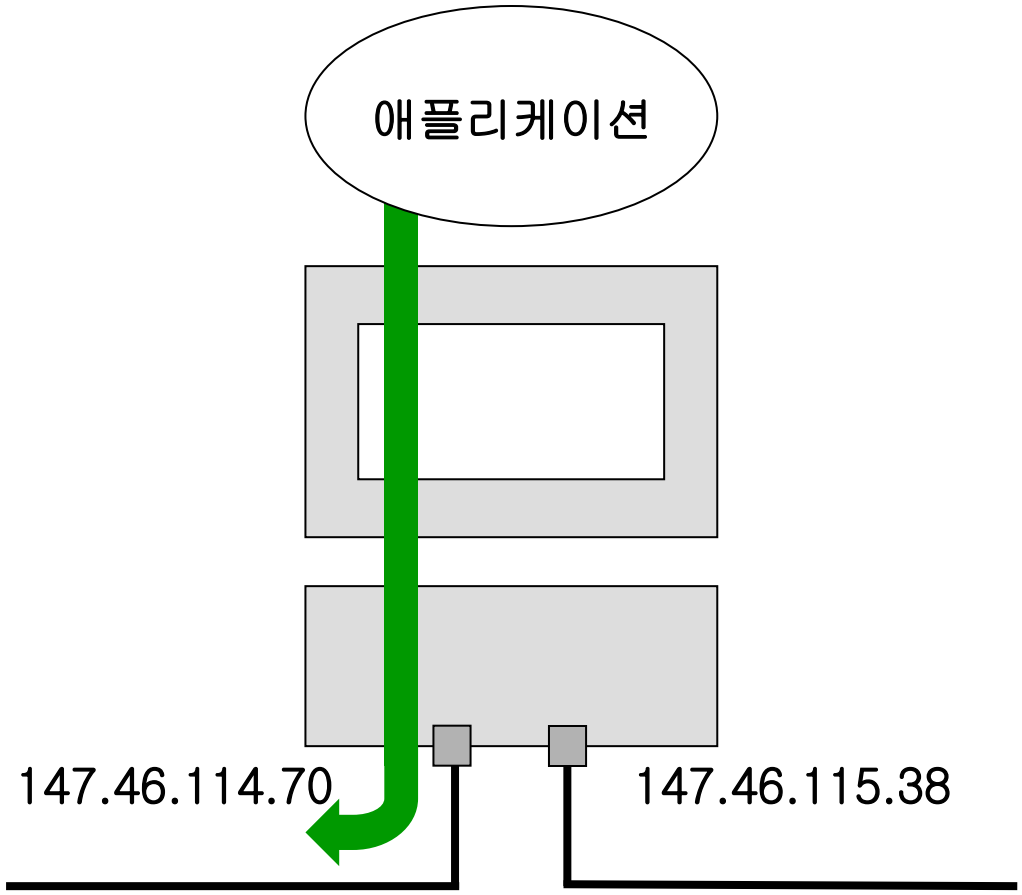
IP_MULTICAST_IF 옵션 (1/2)

- 용도
  - 두 개 이상의 IP 주소를 가진 호스트에서 멀티캐스트 데이터를 보낼 네트워크 인터페이스를 설정
- 사용 예

```
IN_ADDR localaddr;
localaddr.s_addr = inet_addr("147.46.114.70");
if(setsockopt(sock, IPPROTO_IP, IP_MULTICAST_IF,
    (char *)&localaddr, sizeof(localaddr)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

IP_MULTICAST_IF 옵션 (2/2)

- 옵션 설정 결과



IP_MULTICAST_TTL 옵션

- 용도
  - IP 헤더의 TTL 값을 변경
- 사용 예

```
// 멀티캐스트 TTL 설정
int ttl = 2;
retval = setsockopt(sock, IPPROTO_IP, IP_MULTICAST_TTL,
    (char *)&ttl, sizeof(ttl));
if(retval == SOCKET_ERROR) err_quit("setsockopt()");
```

IP_MULTICAST_LOOP 옵션

- 용도
  - 애플리케이션이 보낸 멀티캐스트 데이터를 자신도 받을 지 여부를 결정
- 사용 예

```
BOOL optval = FALSE; // 자신이 보낸 데이터는 받지 않는다.
if(setsockopt(sock, IPPROTO_IP, IP_MULTICAST_LOOP,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

IP_ADD_MEMBERSHIP,

- 용도
  - 멀티캐스트 그룹에 가입 또는 탈퇴
- 옵션값

```
#include <ws2tcpip.h>

struct ip_mreq {
    struct in_addr imr_multiaddr;/* IP multicast address of group */
    struct in_addr imr_interface;   /* local IP address of interface */
};
```

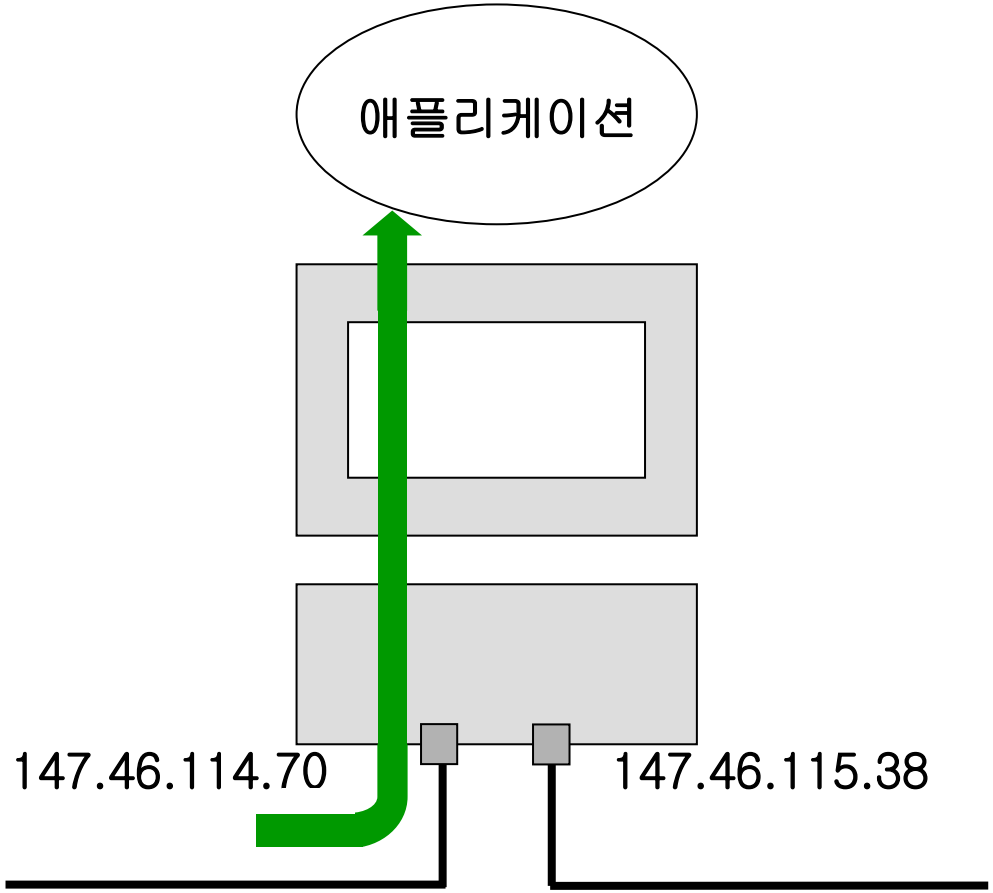
IP_ADD_MEMBERSHIP,

- 사용 예

```
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr = inet_addr("235.7.8.9");
mreq.imr_interface.s_addr = inet_addr("147.46.114.70");
if(setsockopt(sock, IPPROTO_IP, IP_ADD_MEMBERSHIP,
    (char *)&mreq, sizeof(mreq)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```

IP_ADD_MEMBERSHIP, IP_DROP_MEMBERSHIP 옵션 (3/3)

- 옵션 설정 결과



IPPROTO_TCP 옵션 (1/2)

- 용도
  - Nagle 알고리즘 작동 여부 결정
- Nagle 알고리즘

  ① 보낼 데이터가 MSS(maximum segment size)로 정의된 크기만큼 쌓이면, 상대편에게 무조건 보냄

  ② 보낼 데이터가 MSS보다 작을 경우, 이전에 보낸 데이터에 대한 ACK가 오기를 기다림. ACK가 도달하면 보낼 데이터가 MSS보다 작더라도 상대에게 보냄

IPPROTO_TCP 옵션 (2/2)

- Nagle 알고리즘의 장단점
  - 장점: 작은 패킷이 불필요하게 많이 생성되는 것을 미연에 방지함으로써 네트워크 트래픽을 감소시킴
  - 단점: 데이터가 충분히 쌓일 때까지 또는 ACK가 도달할 때까지 대기하는 시간 때문에 애플리케이션의 반응 시간(response time)이 길어질 가능성이 있음
- 사용 예

```
BOOL optval = TRUE; // Nagle 알고리즘 작동 중지
if(setsockopt(sock, IPPROTO_TCP, TCP_NODELAY,
    (char *)&optval, sizeof(optval)) == SOCKET_ERROR)
{
    err_quit("setsockopt()");
}
```