

Chapter 08 UDP 서버-클라이언트

- UDP 서버-클라이언트의 기본 구조와 동작 원리를 이해한다.
- UDP 응용 프로그램 작성에 필요한 핵심 소켓 함수를 익힌다.
- IPv4와 IPv6 기반 UDP 서버-클라이언트를 작성할 수 있다.
- 브로드캐스팅의 개념을 이해하고 UDP를 이용해 구현할 수 있다.

목차

- 01 UDP 서버-클라이언트 구조
- 02 UDP 서버-클라이언트 분석
- 03 UDP 서버-클라이언트(IPv6)
- 04 브로드캐스팅

01 UDP 서버-클라이언트 구조



TCP와 UDP (1)

■ TCP와 UDP의 공통점

- 포트 번호를 이용해 주소를 지정
 - 두 응용 프로그램이 TCP나 UDP를 이용해 통신하려면 반드시 포트 번호를 결정해야 함
- 데이터 오류를 체크
 - TCP와 UDP는 헤더는 물론이고 데이터에 대한 오류도 체크

TCP와 UDP (2)

■ UDP의 특징

- 연결 설정을 하지 않으므로 connect() 함수 불필요
- 프로토콜 수준에서 신뢰성 있는 데이터 전송을 보장하지 않으므로, 필요하다면 응용 프로그램 수준에서 신뢰성 있는 데이터 전송 기능을 구현해야 함
- 간단한 소켓 함수 호출 절차만 따르면 다자간 통신을 쉽게 구현할 수 있음
- TCP와 달리 응용 프로그램이 데이터 경계 구분을 위한 작업을 별도로 할 필요가 없음

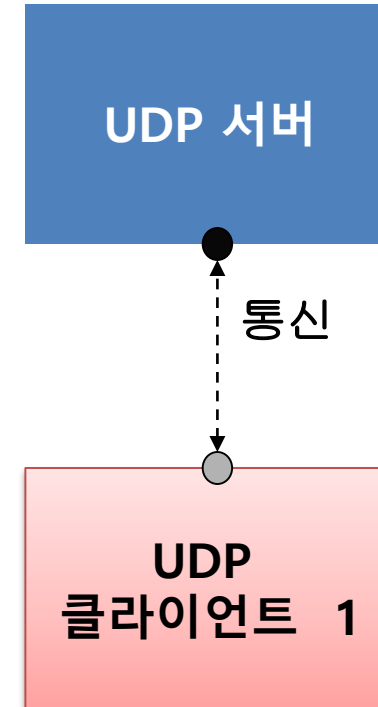
TCP와 UDP (3)

■ TCP와 UDP의 차이점

	TCP	UDP
①	연결형 프로토콜 – 연결 설정 후 통신 가능	비연결형 프로토콜 – 연결 설정 없이 통신 가능
②	신뢰성 있는 데이터 전송 – 필요시 데이터 재전송	신뢰성 없는 데이터 전송 – 데이터를 재전송하지 않음
③	일대일 통신	일대일 통신 일대다 통신
④	데이터 경계 구분 안 함 – 바이트 스트림 서비스	데이터 경계 구분함 – 데이터그램 서비스

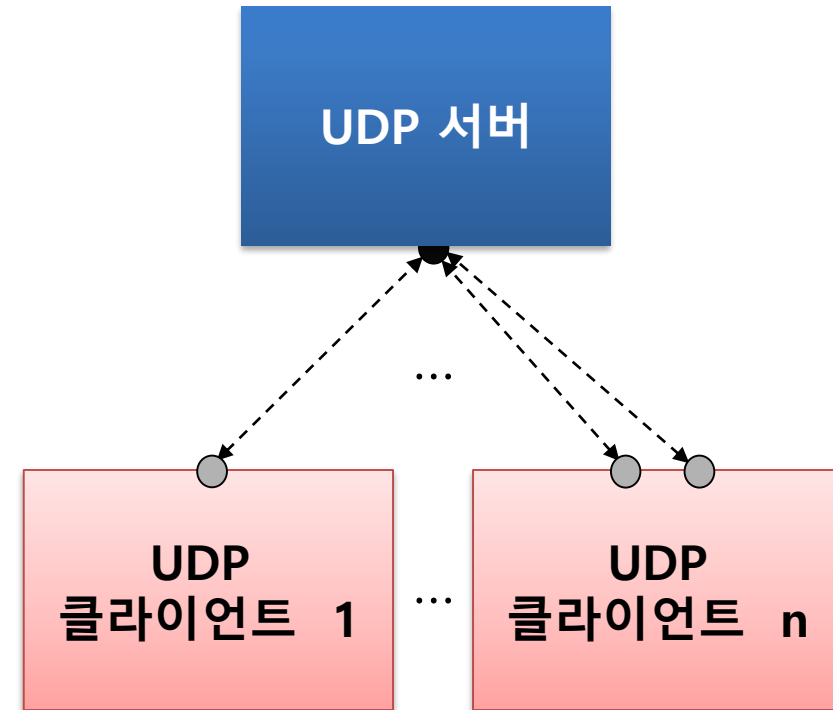
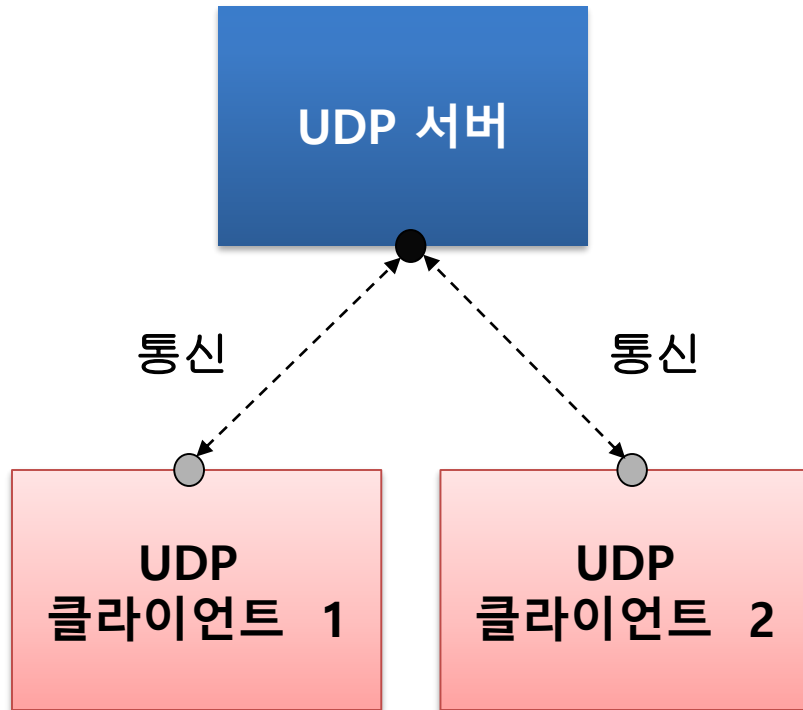
UDP 서버-클라이언트 동작 원리 (1)

■ UDP 서버-클라이언트 동작 원리



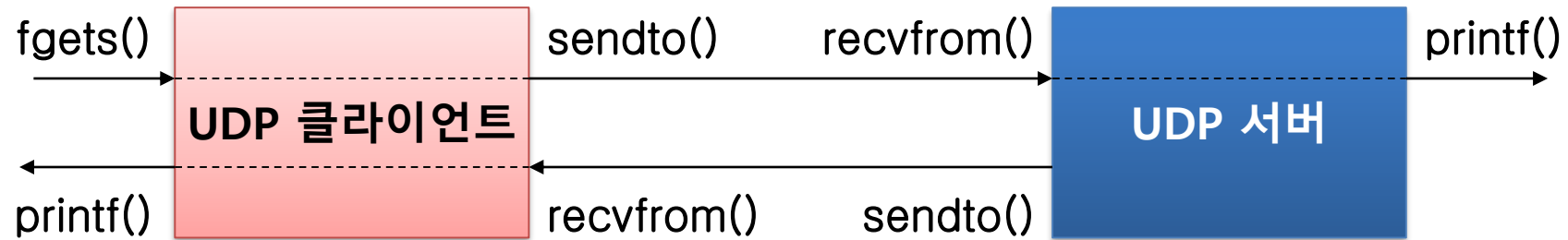
UDP 서버-클라이언트 동작 원리 (2)

■ UDP 서버-클라이언트 동작 원리



UDP 서버-클라이언트 실습 (1)

■ UDP 서버-클라이언트 예제 동작



UDP 서버-클라이언트 실습 (2)

■ 실습 8-1 UDP 서버-클라이언트 작성과 테스트

- UDPServer.cpp

- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPServer/UDPServer.cpp>

- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPServer.cpp>

- UDPClient.cpp

- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPClient/UDPClient.cpp>

- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPClient.cpp>

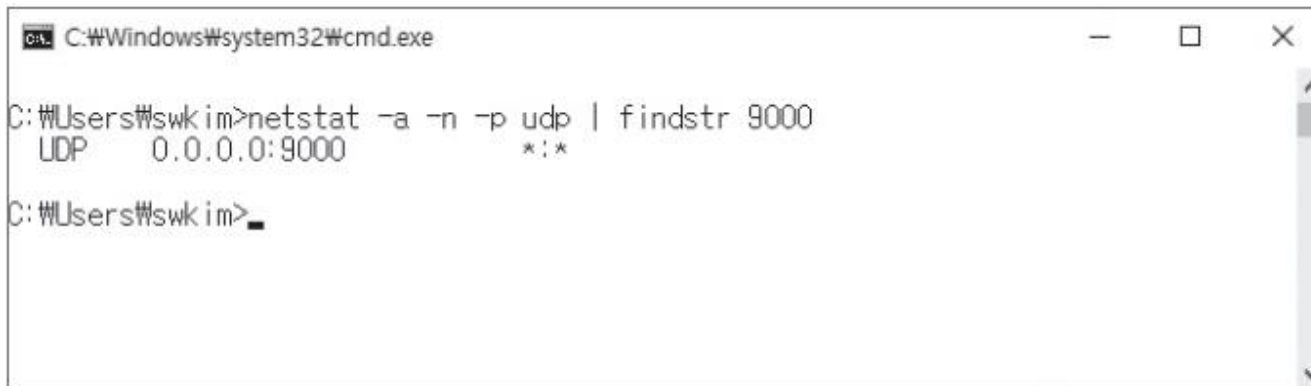
UDP 서버-클라이언트 실습 (3)

■ 실습 8-1 UDP 서버-클라이언트 작성과 테스트

① UDP 서버를 실행



② 명령 프롬프트를 실행한 후 netstat -a -n -p udp | findstr 9000 명령을 실행

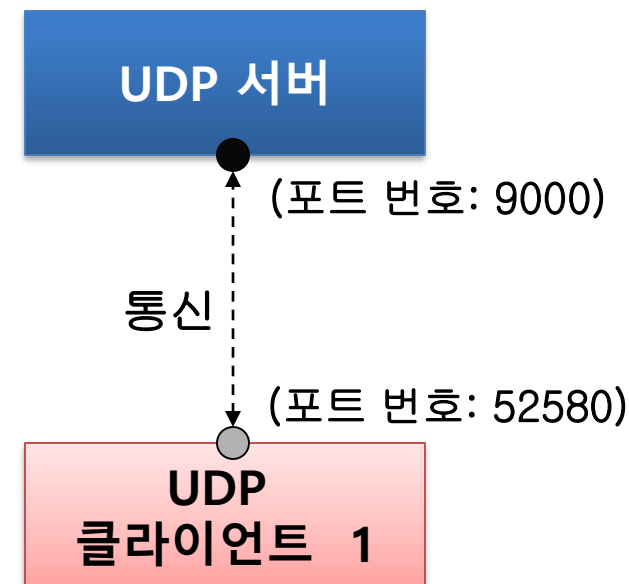


UDP 서버-클라이언트 실습 (4)

③ UDP 클라이언트를 실행



④ 클라이언트에서 글자를 입력하고 [Enter]를 누름



UDP 서버-클라이언트 실습 (5)

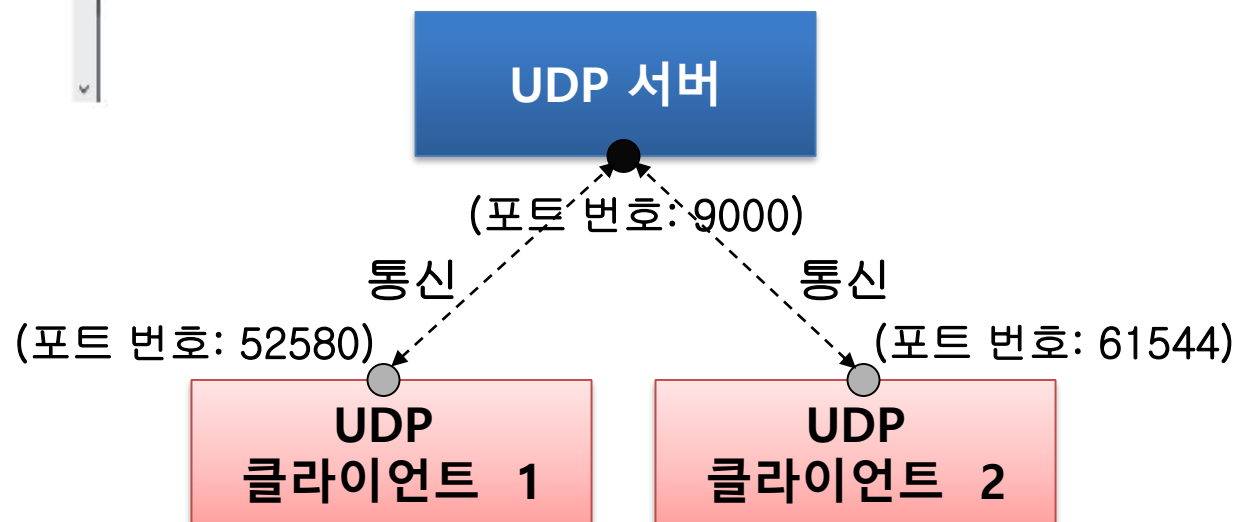
⑤ UDP 클라이언트를 하나 더 실행

```
C:\Source\Windows\Chapter07\UDPServer\Wx64\Debug\UDPServer.exe
[UDP/127.0.0.1:52580] 안녕하세요.
[UDP/127.0.0.1:52580] 반갑습니다.
[UDP/127.0.0.1:61544] 두 번째 클라이언트입니다.

C:\Source\Windows\Chapter07\UDPClient\Wx64\Debug\UDPClient.exe
[보낸 데이터] 안녕하세요.
[UDP 클라이언트] 11바이트를 보냈습니다.
[UDP 클라이언트] 11바이트를 받았습니다.
[받은 데이터] 안녕하세요.

C:\Source\Windows\Chapter07\UDPClient\Wx64\Debug\UDPClient.exe
[보낸 데이터] 두 번째 클라이언트입니다.
[UDP 클라이언트] 25바이트를 보냈습니다.
[UDP 클라이언트] 25바이트를 받았습니다.
[받은 데이터] 두 번째 클라이언트입니다.

[보낸 데이터] .
```



02 UDP 서버-클라이언트 분석



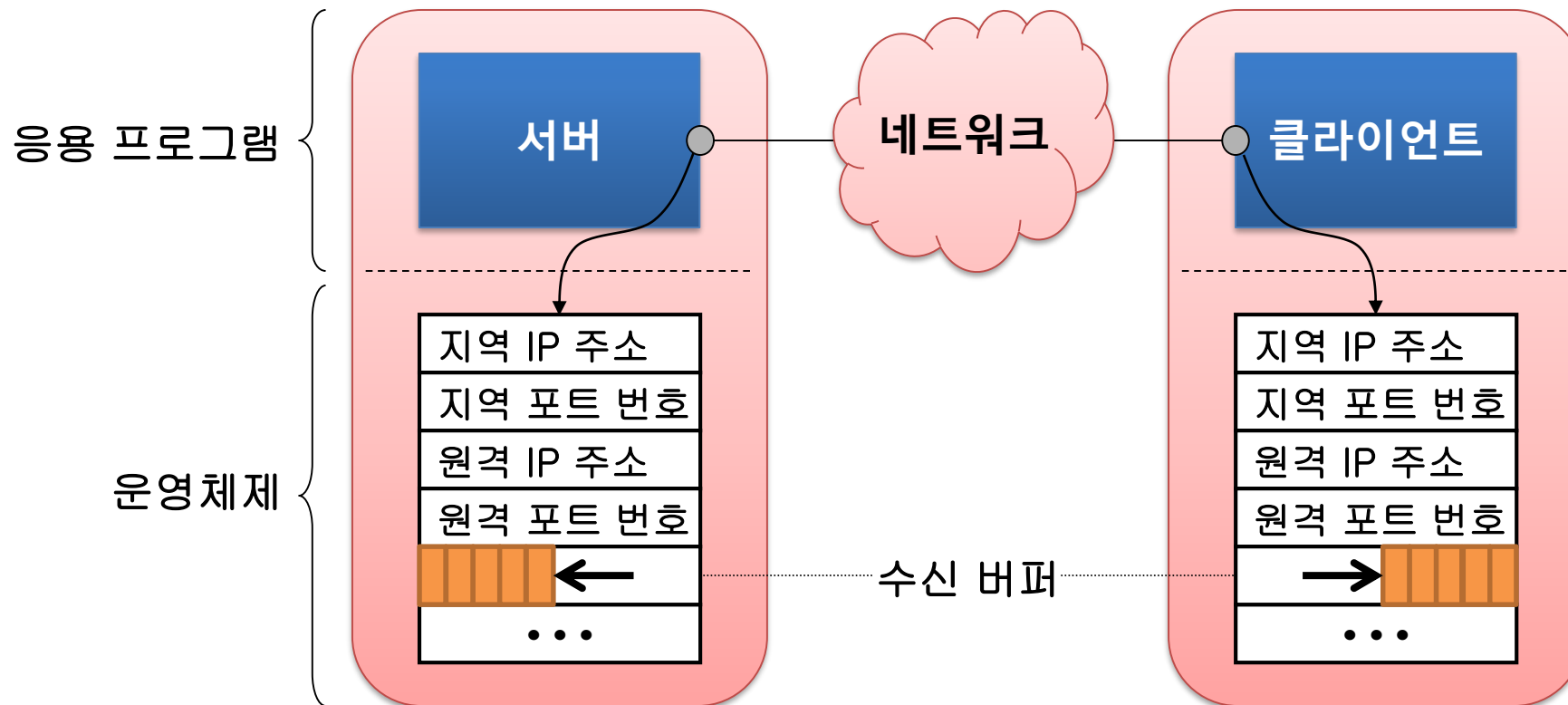
UDP 서버-클라이언트 분석 (1)

■ 응용 프로그램 통신을 위해 결정해야 할 요소

- ① 프로토콜 : 통신 규약으로, 소켓을 생성할 때 결정
- ② 지역 IP 주소와 지역 포트 번호 : 서버 또는 클라이언트 자신의 주소
- ③ 원격 IP 주소와 원격 포트 번호 : 서버 또는 클라이언트가 통신하는 상대의 주소

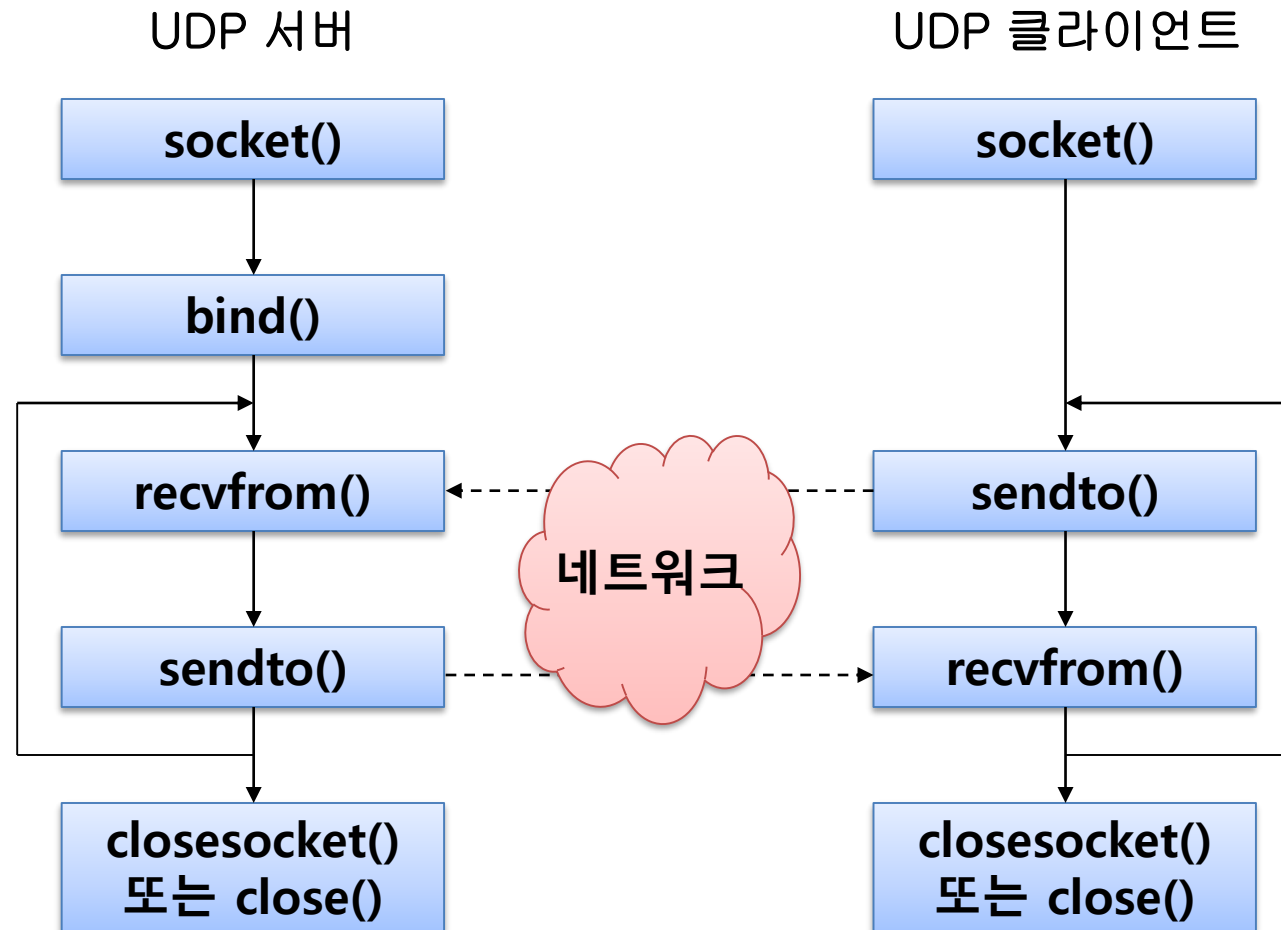
UDP 서버-클라이언트 분석 (2)

■ 소켓 데이터 구조체



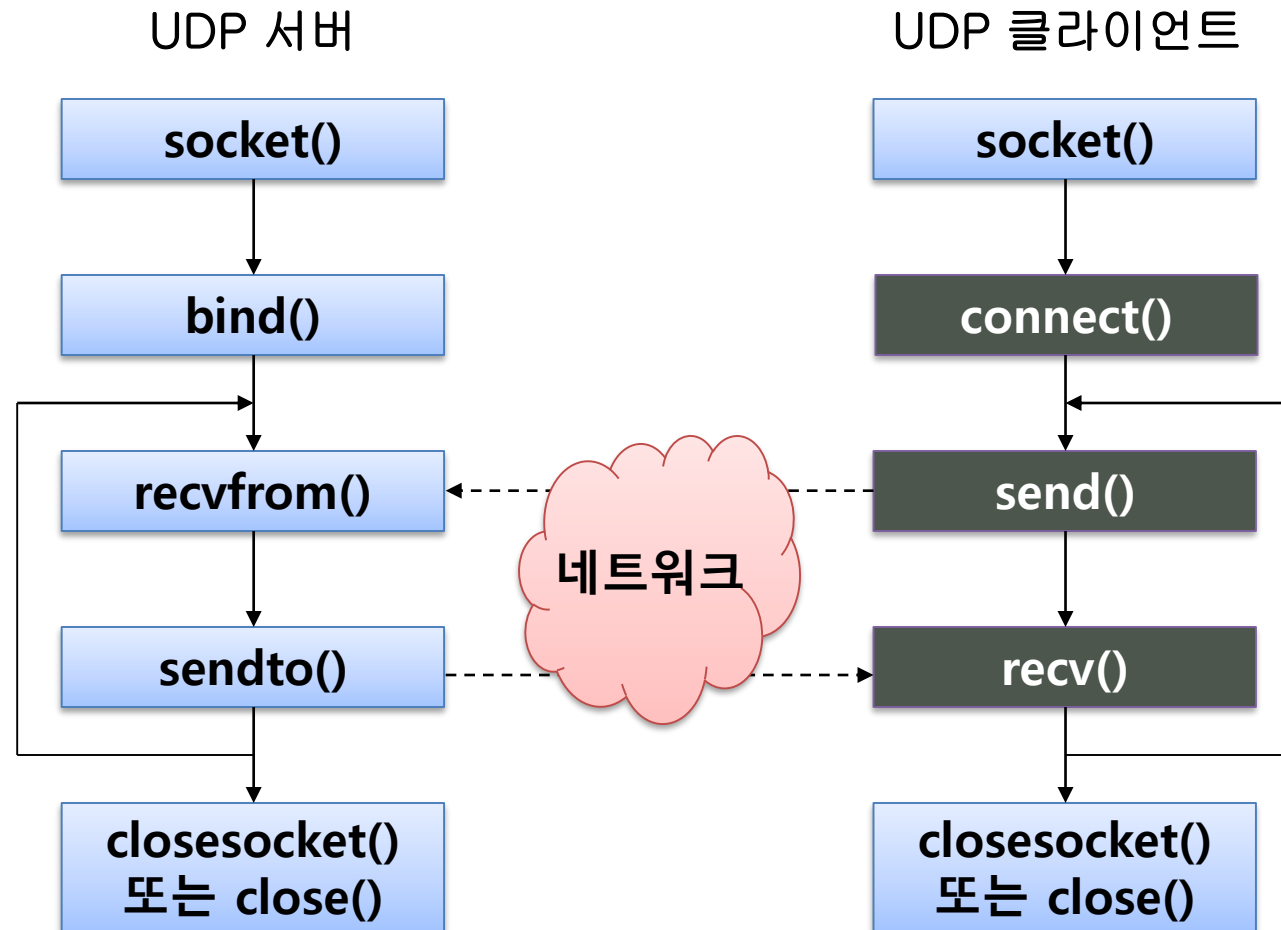
UDP 서버-클라이언트 분석 (3)

■ UDP 서버-클라이언트 모델 ①



UDP 서버-클라이언트 분석 (4)

■ UDP 서버-클라이언트 모델 ②



데이터 전송 함수 (1)

■ sendto() 함수

- 응용 프로그램 데이터를 운영체제의 커널에 복사함으로써 데이터를 전송
 - 소켓의 지역 IP 주소와 지역 포트 번호가 아직 결정되지 않은 상태라면 운영체제가 자동으로 결정

윈도우

```
#include <winsock2.h>
int sendto(
    ① SOCKET sock,
    ② const char *buf,
    ③ int len,
    ④ int flags,
    ⑤ const struct sockaddr *addr,
    ⑥ int addrlen
);
```

성공: 보낸 바이트 수, 실패: SOCKET_ERROR

리눅스

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t sendto(
    ① int sock,
    ② void *buf,
    ③ size_t len,
    ④ int flags,
    ⑤ struct sockaddr *addr,
    ⑥ socklen_t addrlen
);
```

성공: 보낸 바이트 수, 실패: -1

데이터 전송 함수 (2)

■ sendto() 함수 사용 예

```
// 소켓 주소 구조체를 수신자의 IP 주소와 포트 번호로 초기화한다.  
struct sockaddr_in serveraddr;  
  
...  
// 송신용 버퍼를 선언하고 데이터를 넣는다.  
char buf[BUFSIZE];  
  
...  
// sendto() 함수로 데이터를 보낸다.  
retval = sendto(sock, buf, (int)strlen(buf), 0,  
    (struct sockaddr *)&serveraddr, sizeof(serveraddr));  
if (retval == SOCKET_ERROR) 오류 처리;  
printf("%d바이트를 보냈습니다.\n", retval);
```

데이터 전송 함수 (3)

■ recvfrom() 함수

- 운영체제의 수신 버퍼에 도착한 데이터를 응용 프로그램 버퍼에 복사
 - UDP 패킷 데이터를 한 번에 하나만 읽을 수 있음

윈도우

```
#include <winsock2.h>

int recvfrom(
    ① SOCKET sock,
    ② char *buf,
    ③ int len,
    ④ int flags,
    ⑤ struct sockaddr *addr,
    ⑥ int *addrlen
);
```

성공: 받은 바이트 수, 실패: SOCKET_ERROR

리눅스

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

ssize_t recvfrom(
    ① int sock,
    ② void *buf,
    ③ size_t len,
    ④ int flags,
    ⑤ struct sockaddr *addr,
    ⑥ socklen_t *addrlen
);
```

성공: 받은 바이트 수, 실패: -1

데이터 전송 함수 (4)

■ recvfrom() 함수 사용 예

```
// 통신 상대의 주소를 저장할 변수를 선언한다.  
struct sockaddr_in peeraddr;  
int addrlen;  
// 수신용 버퍼를 선언한다.  
char buf[BUFSIZE];  
// recvfrom() 함수로 데이터를 받는다.  
addrlen = sizeof(peeraddr);  
retval = recvfrom(sock, buf, BUFSIZE, 0,  
    (struct sockaddr *)&peeraddr, &addrlen);  
if (retval == SOCKET_ERROR) 오류 처리;  
printf("%d바이트를 받았습니다.\n", retval);
```

03 UDP 서버-클라이언트(IPv6)



UDP서버-클라이언트(IPv6) (1)

■ IPv4 코드 ⇒ IPv6 코드

- (필요시) 새로운 헤더 파일을 포함
 - 윈도우에 한해 ws2tcpip.h 추가
- 소켓 생성 시 AF_INET 대신 AF_INET6 사용
- 소켓 주소 구조체로 sockaddr_in 대신 sockaddr_in6 사용
 - 구조체를 변경하면 구조체 필드명도 그에 따라 변경
 - 서버에서 주로 사용하는 INADDR_ANY 값은 in6addr_any로 변경
- 데이터 전송 함수는 기존의 sendto() 함수와 recvfrom() 함수를 변경 없이 그대로 사용

UDP서버-클라이언트(IPv6) (2)

■ 실습 8-2 UDP 서버-클라이언트(IPv6) 작성과 테스트

```
C:\Source\Windows\Chapter07\UDPServer6\x64\Debug\UDPServer6.exe
[UDP/::1:62757] 안녕하세요.
[UDP/::1:62757] 반갑습니다.

C:\Source\Windows\Chapter07\UDPClient6\x64\Debug\UDPClient6.exe
[보낼 데이터] 안녕하세요.
[UDP 클라이언트] 11바이트를 보냈습니다.
[UDP 클라이언트] 11바이트를 받았습니다.
[받은 데이터] 안녕하세요.

[보낼 데이터] 반갑습니다.
[UDP 클라이언트] 11바이트를 보냈습니다.
[UDP 클라이언트] 11바이트를 받았습니다.
[받은 데이터] 반갑습니다.

[보낼 데이터] .
```

윈도우 UDP 서버-클라이언트(IPv6)

```
C:\Windows\system32\cmd.exe
C:\Users\swkim>netstat -a -n -p udpv6

활성 연결

프로토콜  로컬 주소          외부 주소      상태
UDP       [::]:7         *:*
UDP       [::]:9         *:*
UDP       [::]:13        *:*
UDP       [::]:17        *:*
UDP       [::]:19        *:*
UDP       [::]:500       *:*
UDP       [::]:3389      *:*
UDP       [::]:3702      *:*
UDP       [::]:3702      *:*
UDP       [::]:4500      *:*
UDP       [::]:5353      *:*
UDP       [::]:5355      *:*
UDP       [::]:9000      *:*
UDP       [::]:54434     *:*
UDP       [::]:62757     *:*
UDP       [::1]:1900     *:*
UDP       [::1]:62901    *:*
```

netstat 명령

UDP서버-클라이언트(IPv6) (3)

■ 실습 8-2 UDP 서버-클라이언트(IPv6) 작성과 테스트

■ UDPServer6

- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPServer6/UDPServer6.cpp>
- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPServer6.cpp>

■ UDPClient6

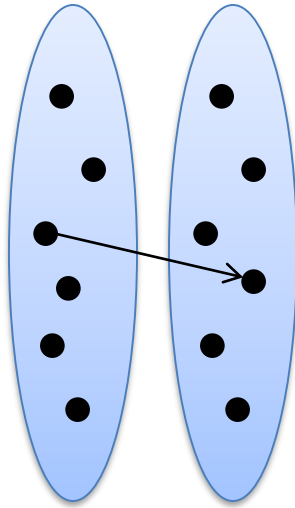
- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPClient6/UDPClient6.cpp>
- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPClient6.cpp>

04 브로드캐스팅

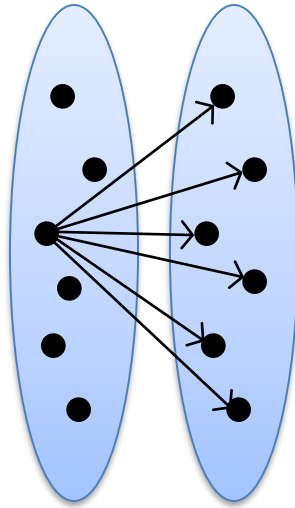


브로드캐스팅 (1)

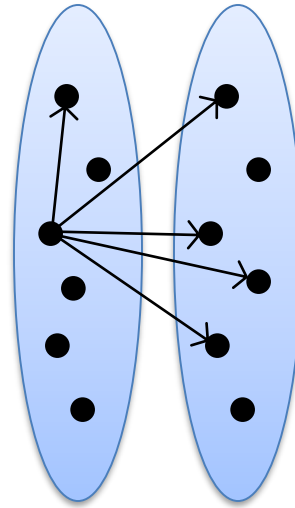
■ 통신에 참여하는 개체 간 상호 작용



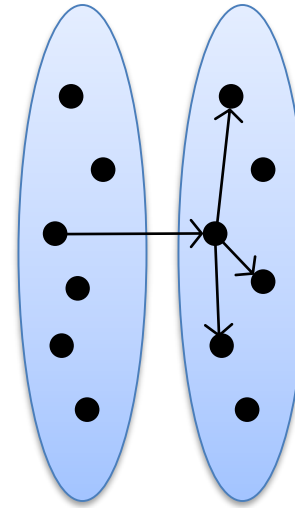
유니캐스팅



브로드캐스팅



멀티캐스팅

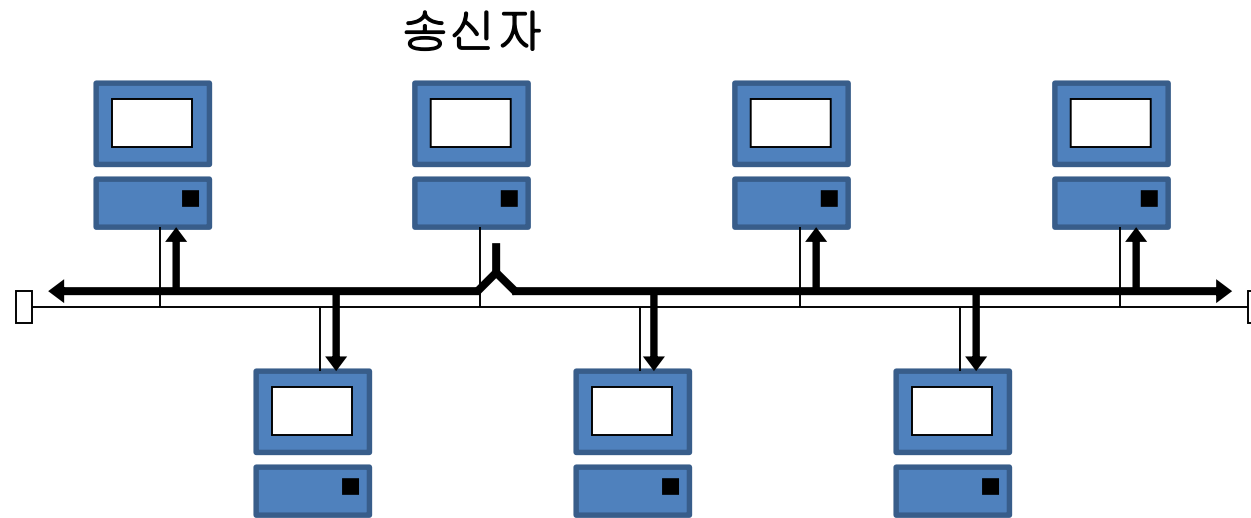


애니캐스팅

브로드캐스팅 (2)

■ 브로드캐스팅 개념

- 송신자가 보낸 데이터 하나를 다수의 수신자가 받는 방식
 - 데이터 복사본을 여러 개 만들어 보내는 것이 아니므로 송신자 관점에서 보면 상당히 효율적인 기술



브로드캐스팅 (3)

■ 브로드캐스트 데이터를 보내기 위한 절차

① 브로드캐스팅을 활성화함

윈도우

```
DWORD bEnable = 1;  
setsockopt(sock, SOL_SOCKET, SO_BROADCAST,  
           (const char *)&bEnable, sizeof(bEnable));
```

리눅스

```
int bEnable = 1;  
retval = setsockopt(sock, SOL_SOCKET, SO_BROADCAST,  
                   &bEnable, sizeof(bEnable));
```

브로드캐스팅 (4)

■ 브로드캐스트 데이터를 보내기 위한 절차

② 브로드캐스트 주소를 목적지로 설정해 데이터를 보냄

```
// 소켓 주소 구조체를 초기화한다.
struct sockaddr_in remoteaddr;
memset(&remoteaddr, 0, sizeof(remoteaddr));
remoteaddr.sin_family = AF_INET;
inet_pton(AF_INET, "255.255.255.255", &remoteaddr.sin_addr);
remoteaddr.sin_port = htons(9000);
// 송신용 버퍼를 선언하고 데이터를 넣는다.
char buf[BUFSIZE];
...
// sendto() 함수로 데이터를 보낸다.
retval = sendto(sock, buf, (int)strlen(buf), 0,
    (struct sockaddr *)&remoteaddr, sizeof(remoteaddr));
if(retval == SOCKET_ERROR) 오류 처리;
printf("%d바이트를 보냈습니다.\n", retval);
```

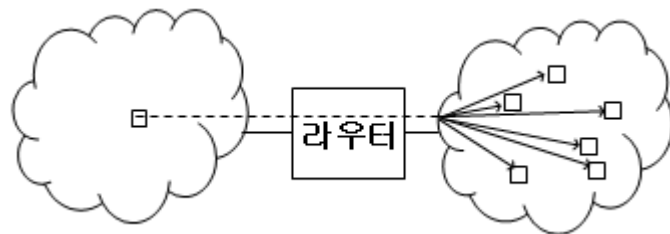

브로드캐스팅 (5)

■ 브로드캐스트 주소의 종류

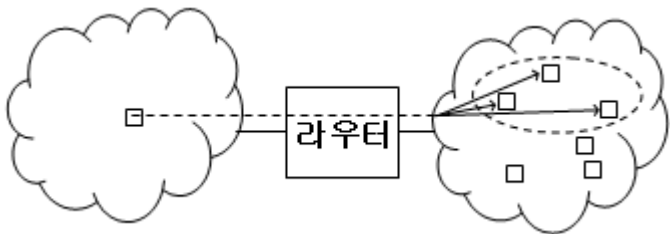


브로드캐스팅 (6)

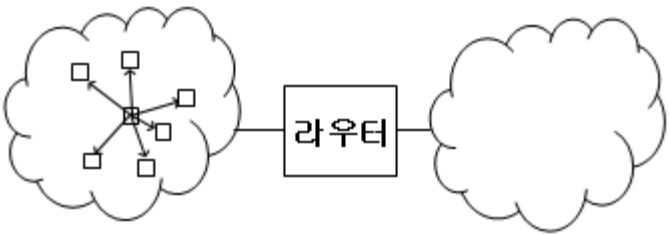
■ 브로드캐스트 주소의 종류



네트워크 브로드캐스트



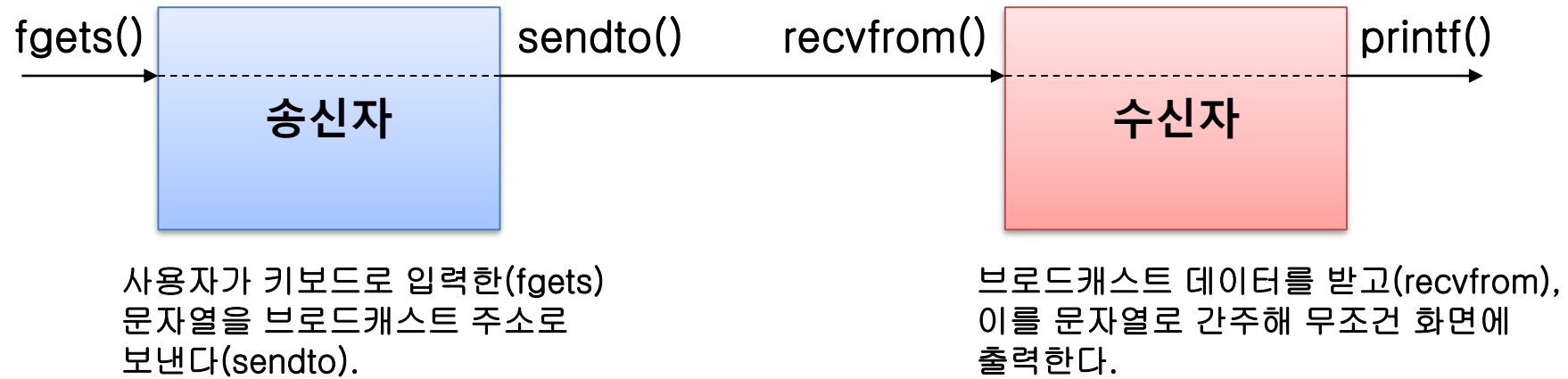
서브넷 브로드캐스트



지역 브로드캐스트

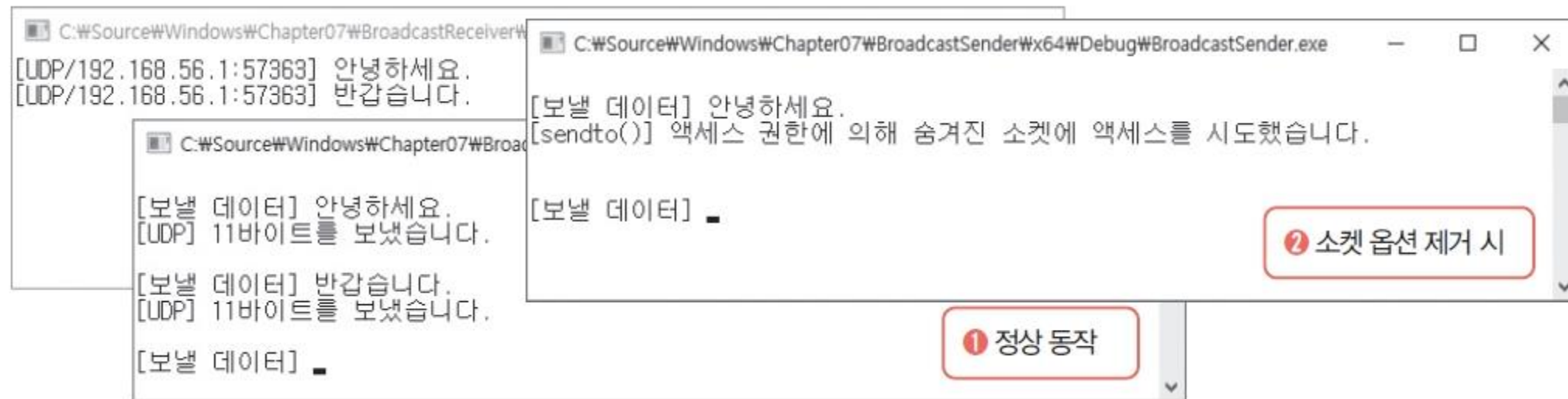
브로드캐스팅 (7)

■ 브로드캐스팅 예제 동작



브로드캐스팅 (8)

■ 실습 8-3 UDP 브로드캐스팅 예제 작성과 테스트



브로드캐스팅 (9)

■ 실습 8-3 UDP 브로드캐스팅 예제 작성과 테스트

- BroadcastReceiver.cpp
 - [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/BroadcastReceiver/BroadcastReceiver.cpp>
 - [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/BroadcastReceiver.cpp>
- BroadcastSender.cpp
 - [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/BroadcastSender/BroadcastSender.cpp>
 - [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/BroadcastSender.cpp>