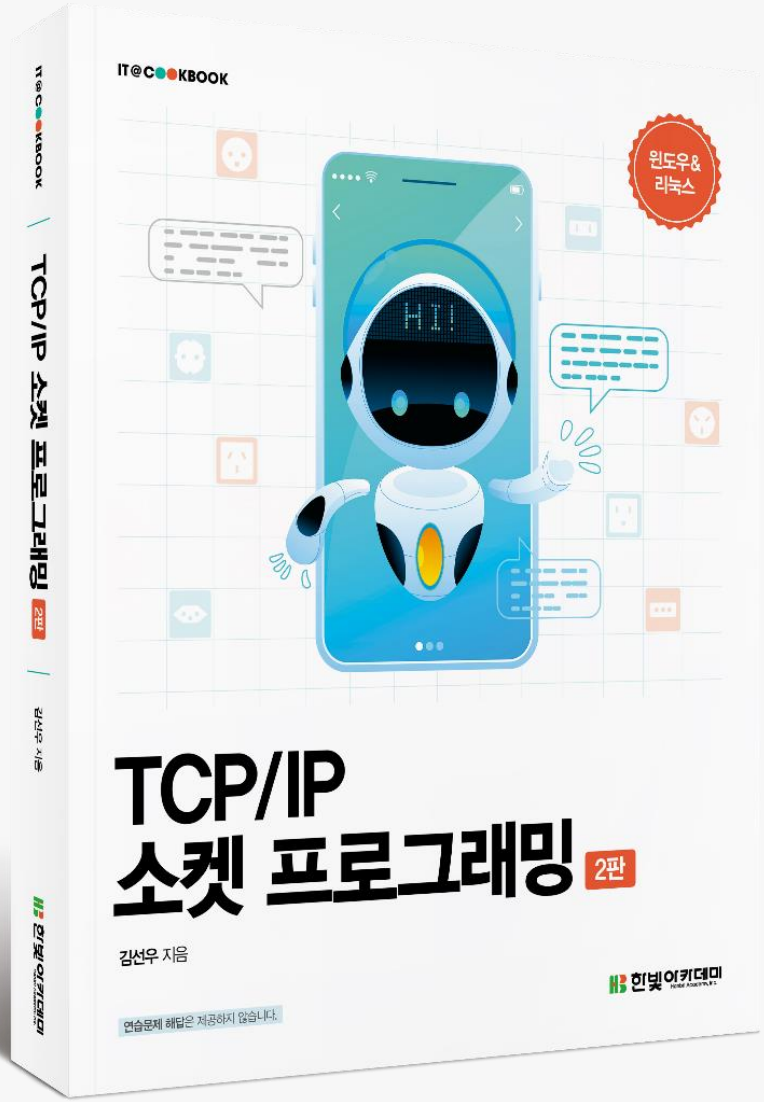


Chapter 08 UDP 서버-클라이언트

학습목표

- 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

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

TCP는 다시 전송을 요구하고
차이는 UDP는 데이터를 버려버린다.

UDP는 송신 버퍼가 필요X

5

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 +순서도 맞춰줘야 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)

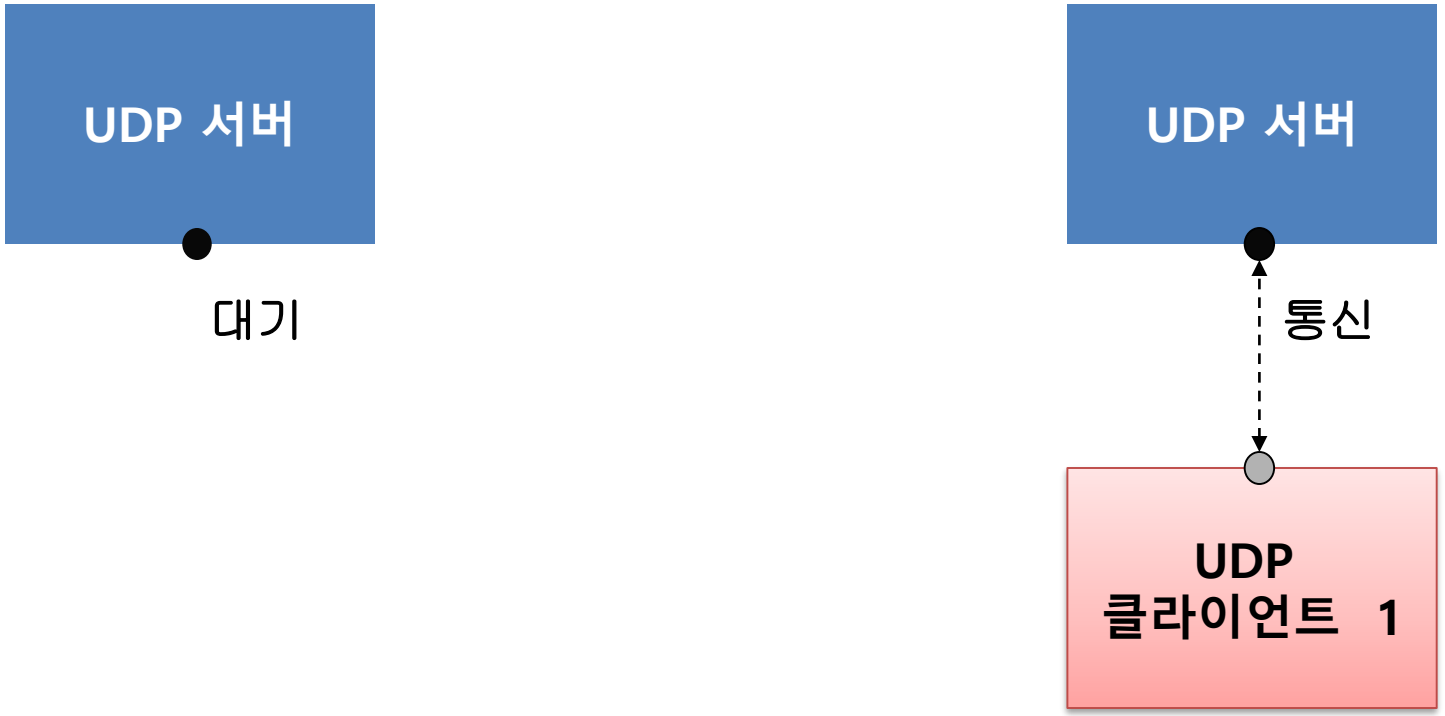
TCP와 UDP의 차이점

	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)

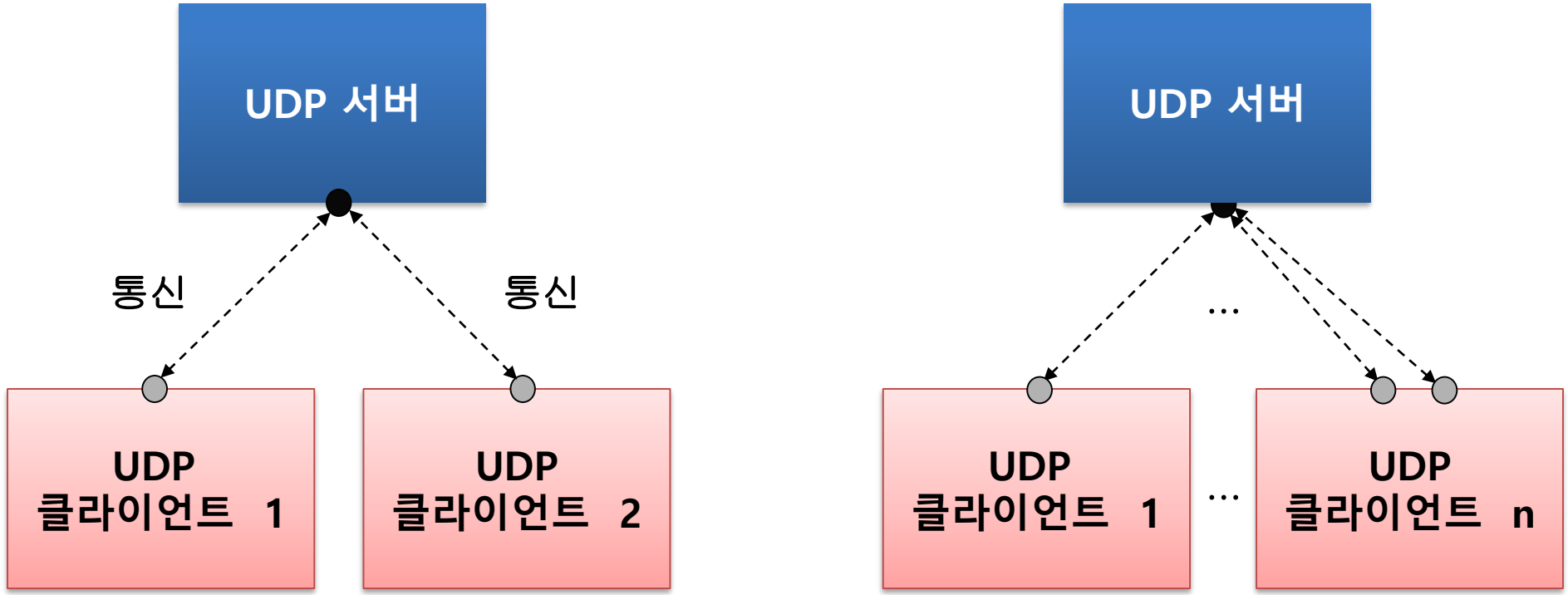
UDP 서버-클라이언트 동작 원리

TCP의 대기소켓은 클라이언트의 접속요청을 받아 전화소켓을 만들어 주는 역할



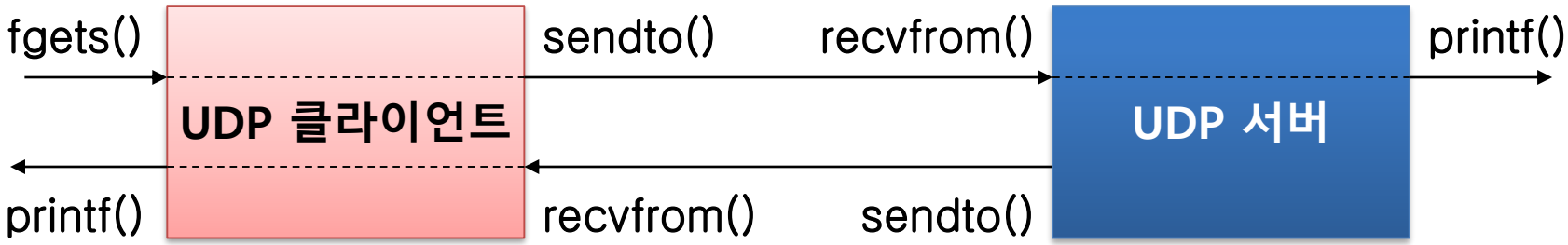
UDP 서버-클라이언트 동작 원리 (2)

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

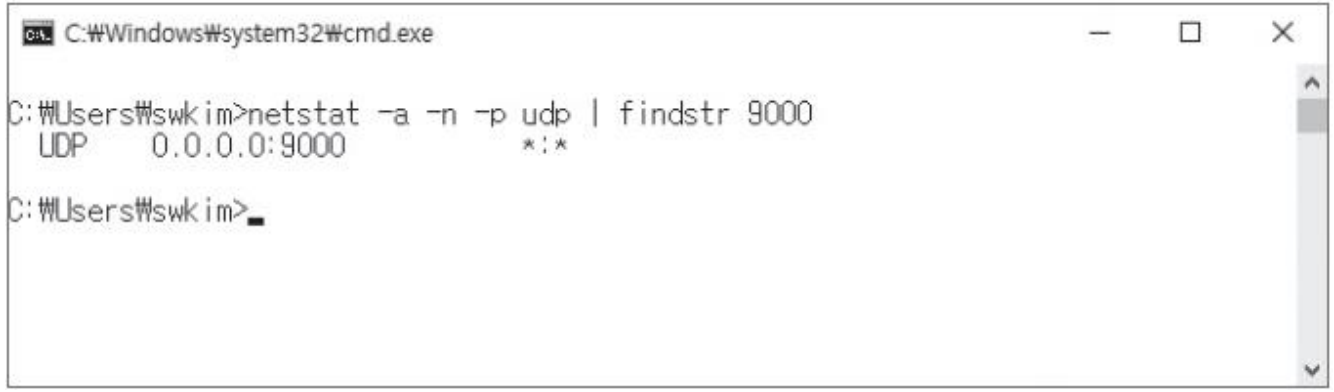
- UDPClient.cpp
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPClient/UDPClient.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPClient.cpp

UDP 서버-클라이언트 실습 (3)

실습 8-1 UDP 서버-클라이언트 작성과 테스트

❶ UDP 서버를 실행

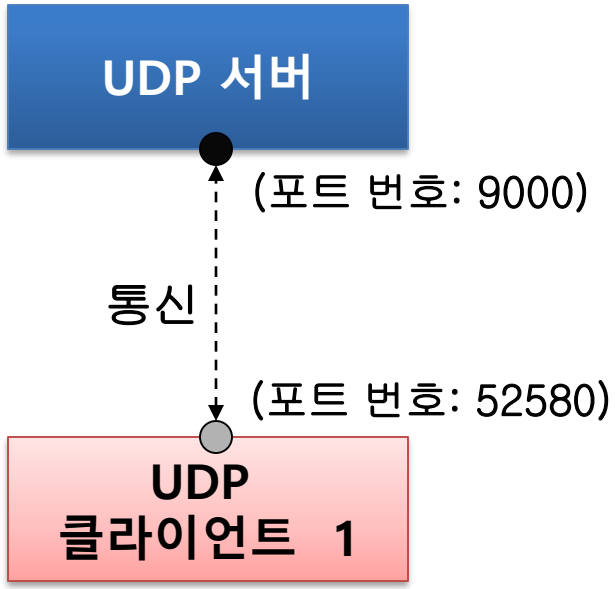
❷ 명령 프롬프트를 실행한 후 netstat -a -n -p udp | findstr 9000 명령을 실행



UDP 서버-클라이언트 실습 (4)

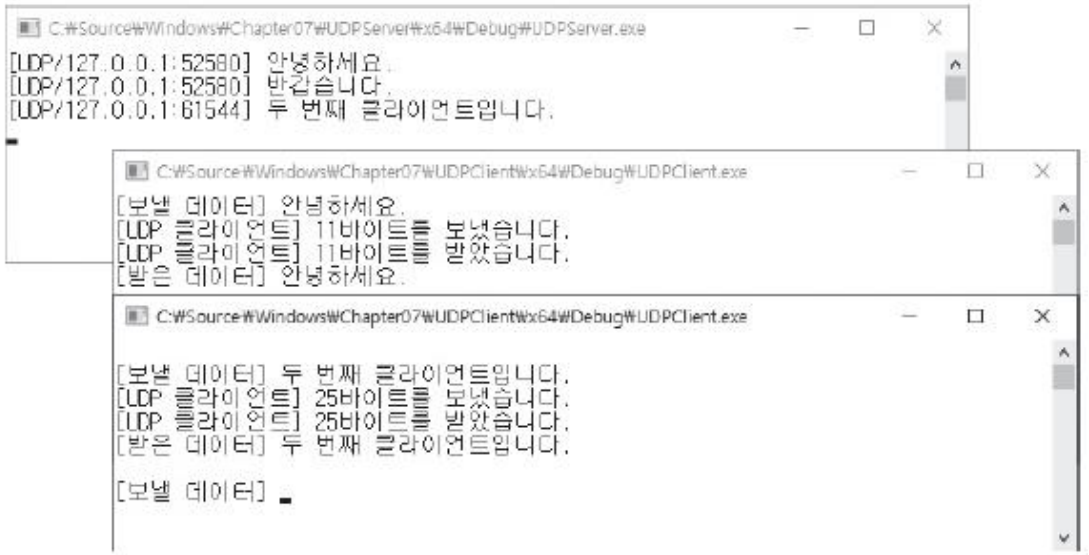
❸ UDP 클라이언트를 실행

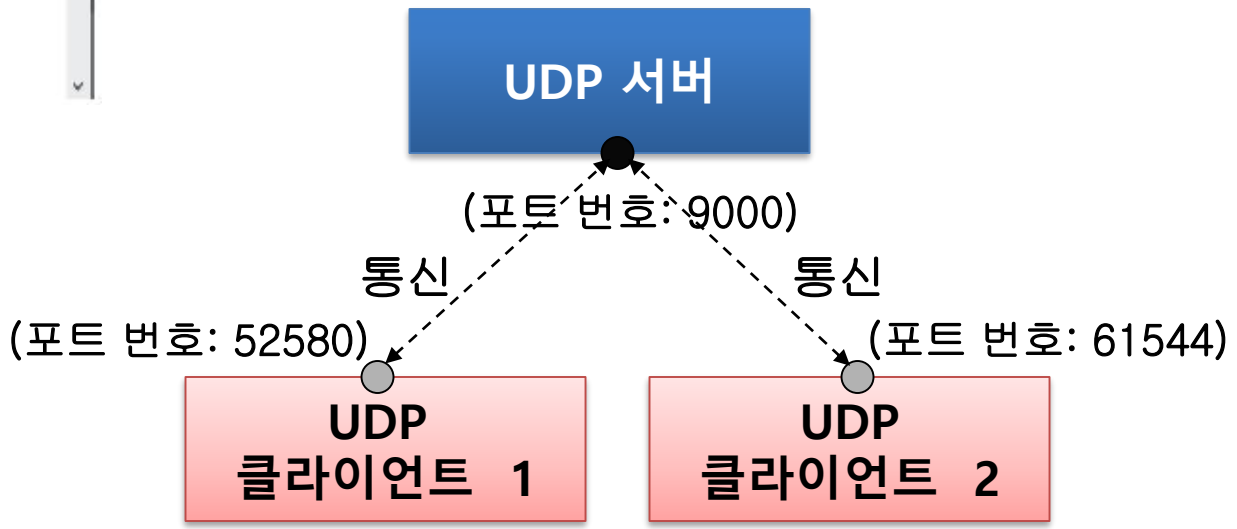
❹ 클라이언트에서 글자를 입력하고 [**Enter**]를 누름



UDP 서버-클라이언트 실습 (5)

❺ UDP 클라이언트를 하나 더 실행





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

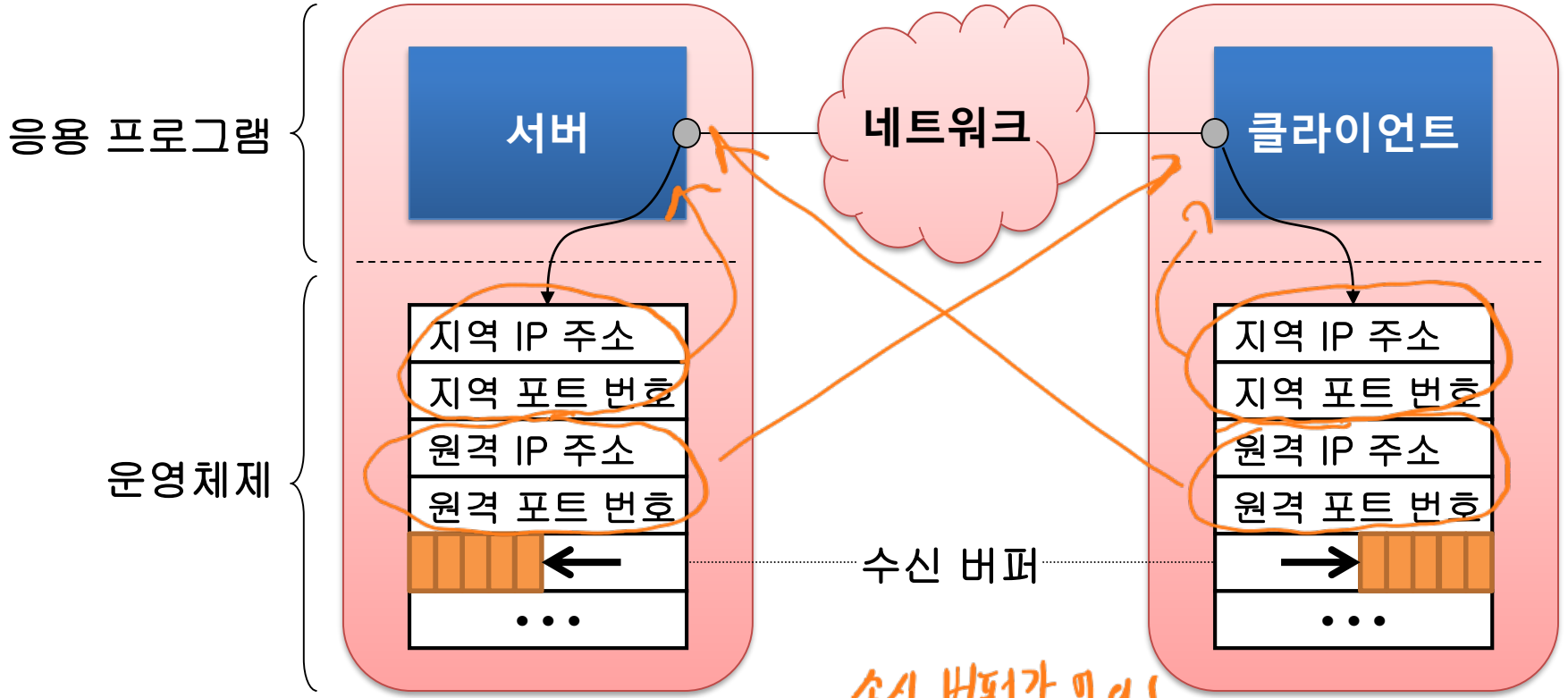
② 지역 IP 주소와 지역 포트 번호 : 서버 또는 클라이언트 자신의 주소

③ 원격 IP 주소와 원격 포트 번호 : 서버 또는 클라이언트가 통신하는 상대의 주소

UDP 서버-클라이언트 분석 (2)

- 소켓 데이터 구조체

TCP ~ ack: 데이터를 제대로 받았다는 메시지 (송신 메시지에 수신 데이터가 있을 경우)

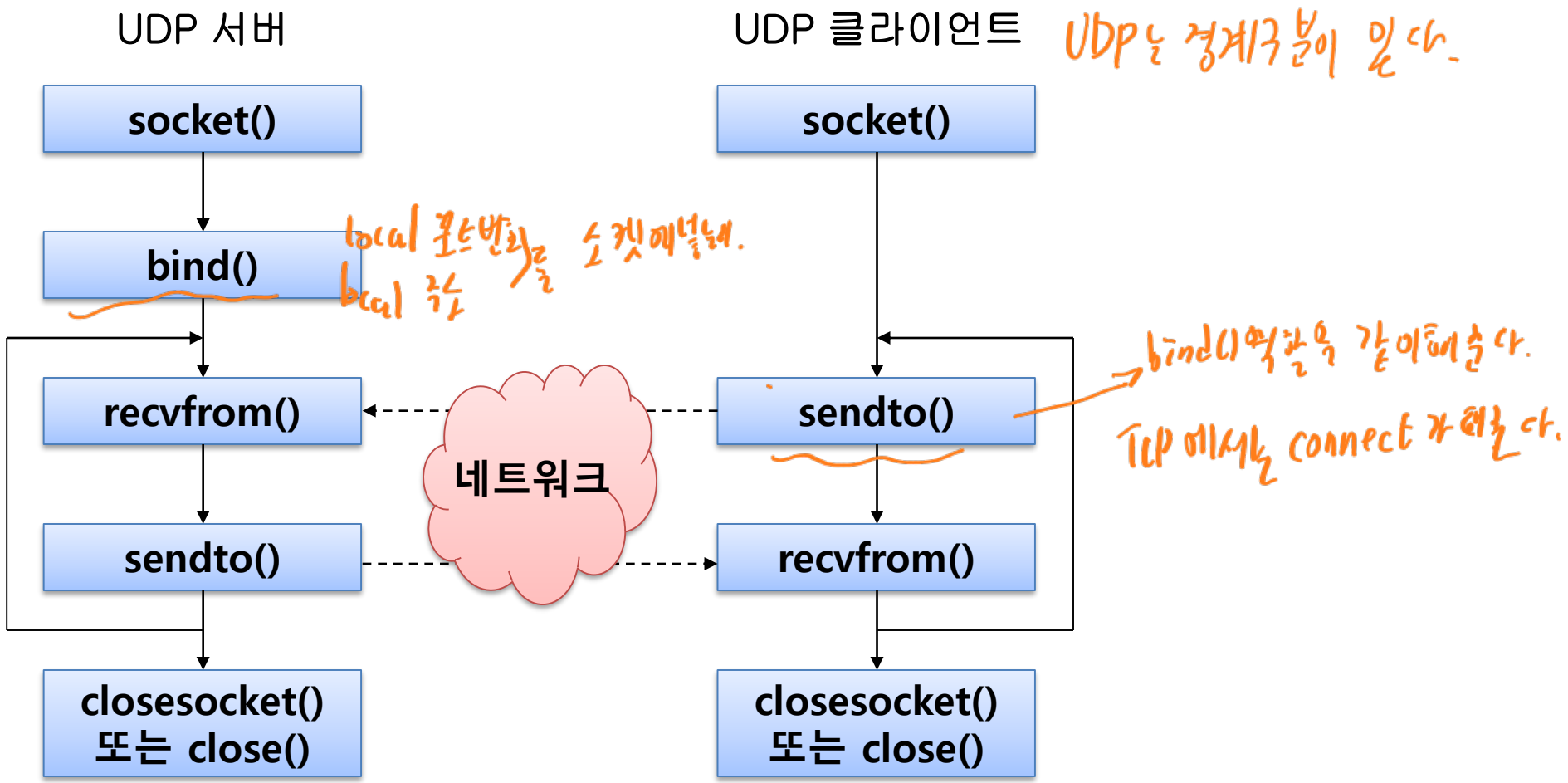


송신 버퍼가 없다!

send()를 사용하여 데이터가 커널의 복사가되고 (네에 리턴된다)

UDP 서버-클라이언트 분석 (3)

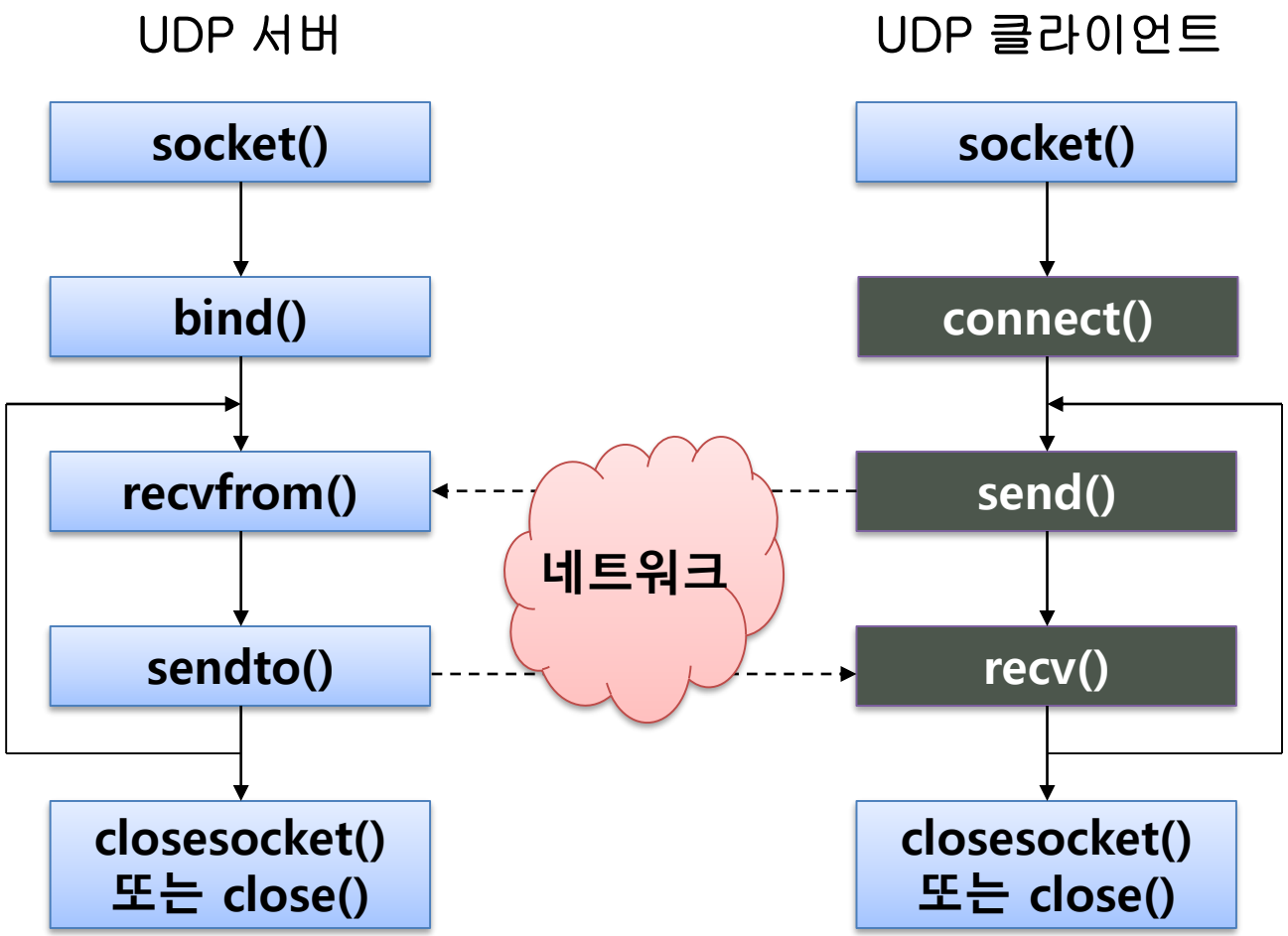
UDP 서버-클라이언트 모델 ①



UDP 서버-클라이언트 분석 (4)

UDP 서버-클라이언트 모델 ②

예외적인 사용



데이터 전송 함수 (1)

sendto() 함수

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

윈도우

```
#include <winsock2.h>
int sendto(
  ❶ SOCKET sock,
  ❷ const char *buf,
  ❸ int len,
  ❹ int flags,
  ❺ const struct sockaddr *addr,
  ❻ int addrlen
);                                  성공: 보낸 바이트 수, 실패: SOCKET_ERROR
```

리눅스

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t sendto(
  ❶ int sock,
  ❷ void *buf,
  ❸ size_t len,
  ❹ int flags,
  ❺ struct sockaddr *addr,
  ❻ 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);
```

그 보내진 데이터의 길이 리턴

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(
   ❶ SOCKET sock,
   ❷ char *buf,
   ❸ int len,
   ❹ int flags,
   ❺ struct sockaddr *addr,
   ❻ int *addrlen
);                                  성공: 받은 바이트 수, 실패: SOCKET_ERROR
```

리눅스

```
#include <sys/types.h>
#include <sys/socket.h>
ssize_t recvfrom(
   ❶ int sock,
   ❷ void *buf,
   ❸ size_t len,
   ❹ int flags,
   ❺ struct sockaddr *addr,
   ❻ 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);
```

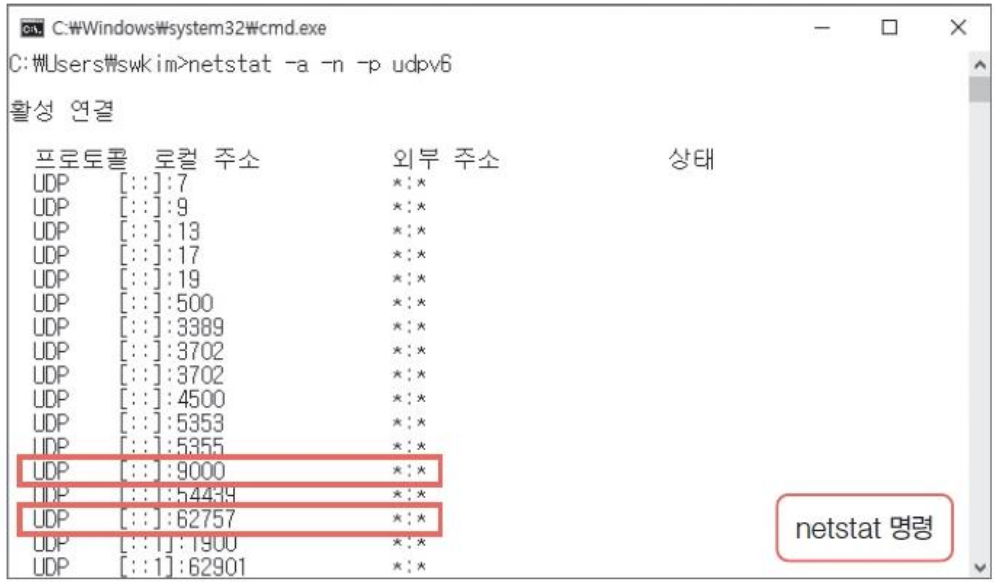
03 UDP 서버-클라이언트(IPv6)

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



UDP서버-클라이언트(IPv6) (3)

실습 8-2 UDP 서버-클라이언트(IPv6) 작성과 테스트

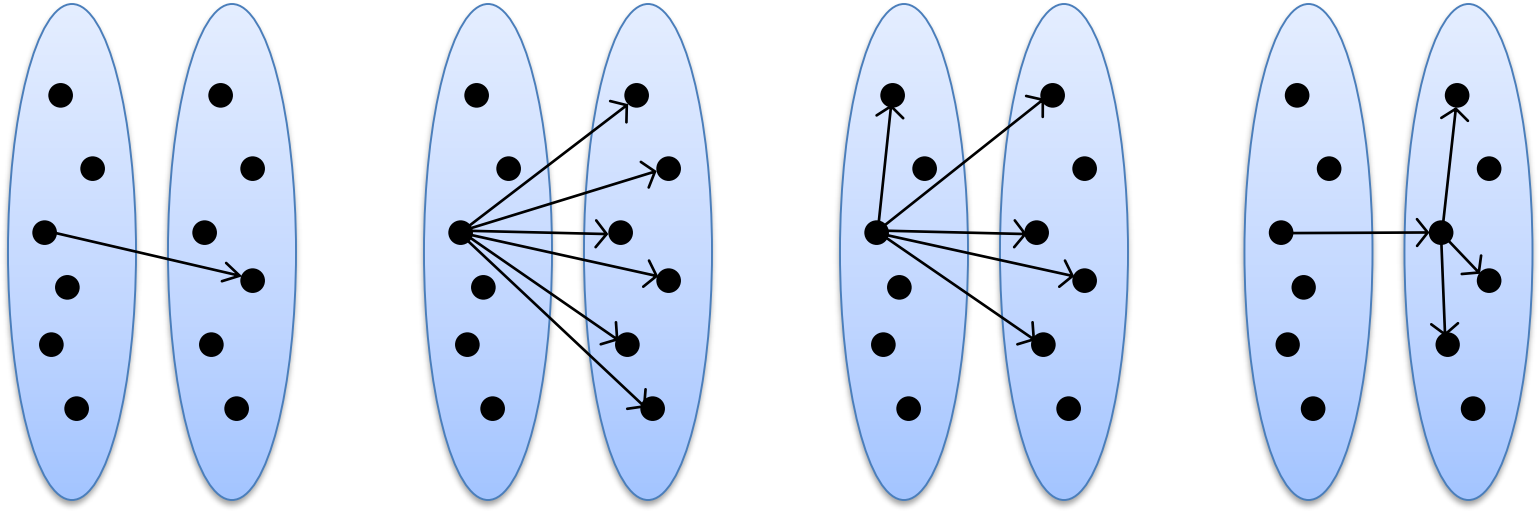
- UDPServer6
- [윈도우] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter08/UDPServer6/UDPServer6.cpp
- [리눅스] https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter08/UDPServer6.cpp

- 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

1GB를 보낸다고 하면
1GB*6 = 6GB를
보내야한다

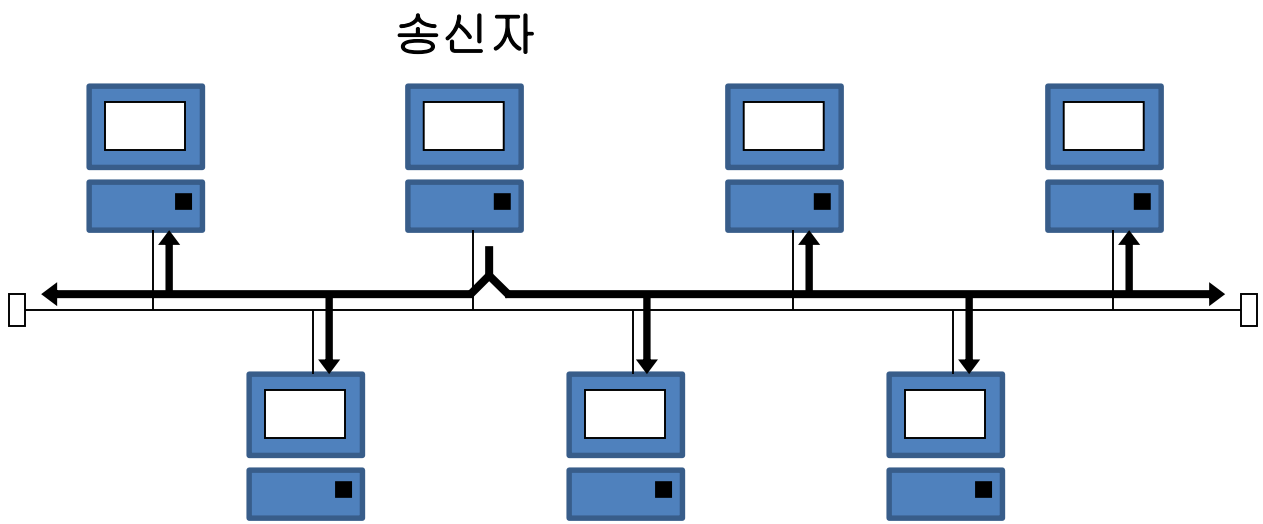
1GB 한번만 보내면
된다.

예) IPTV
같은 채널을 보고있는
사람에게만 등시비

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 → UDP 소켓을 만들어야 한다.

윈도우

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

→ 브로드캐스팅 관련 설정

리눅스

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

브로드캐스팅 (4)

■ 브로드캐스트 데이터를 보내기 위한 절차

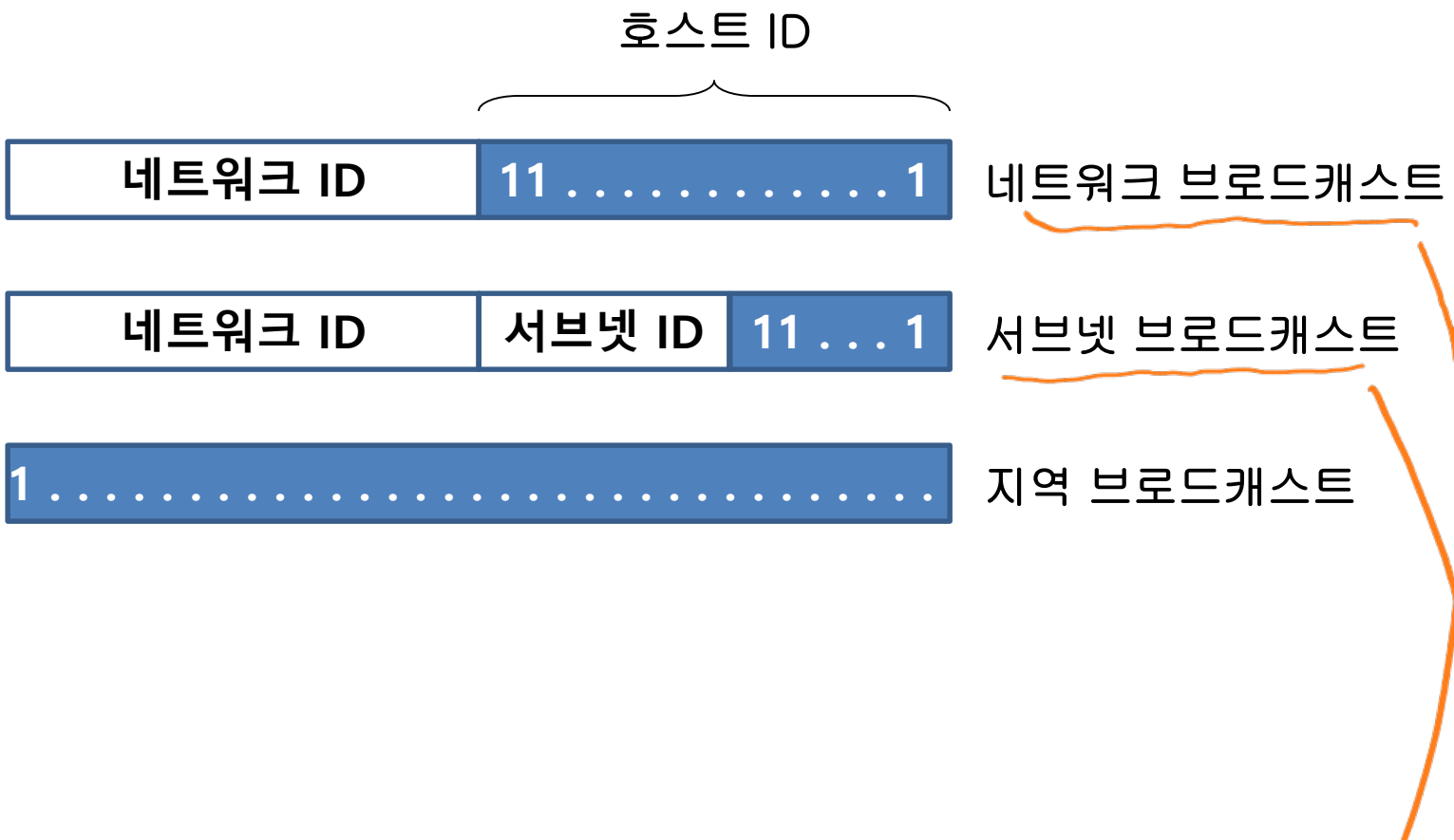
② 브로드캐스트 주소를 목적지로 설정해 데이터를 보냄

```
// 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);
```

↳ 지역 브로드캐스트 주소: 내가속해 있는 네트워크의 모든 호스트에게 데이터를 보낼때사용

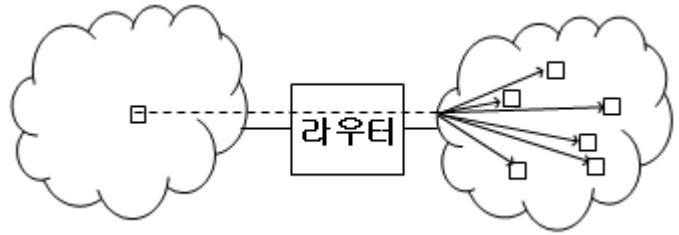
브로드캐스팅 (5)

- 브로드캐스트 주소의 종류

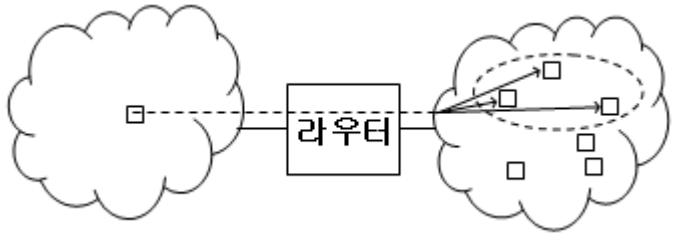


브로드캐스팅 (6)

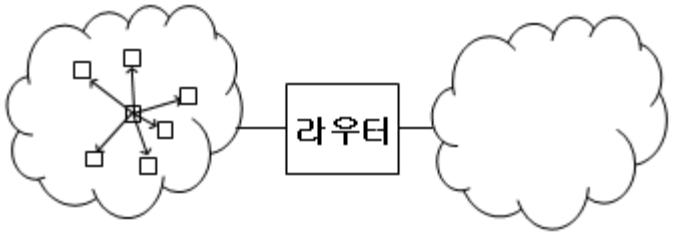
- 브로드캐스트 주소의 종류



네트워크 브로드캐스트



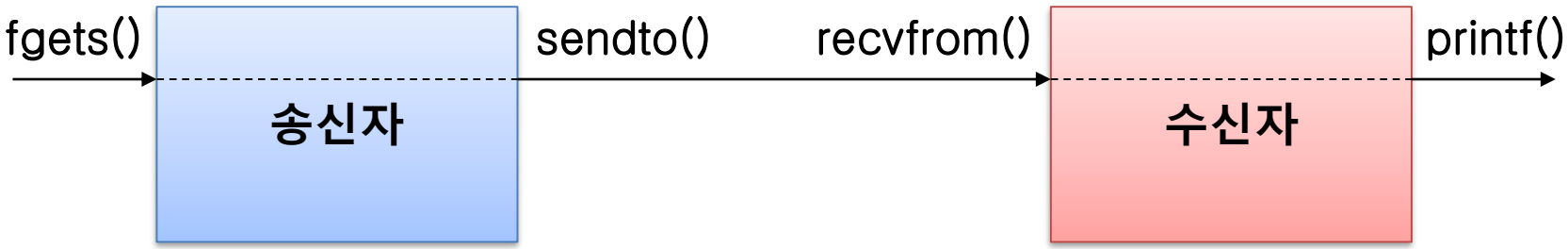
서브넷 브로드캐스트



지역 브로드캐스트

브로드캐스팅 (7)

브로드캐스팅 예제 동작

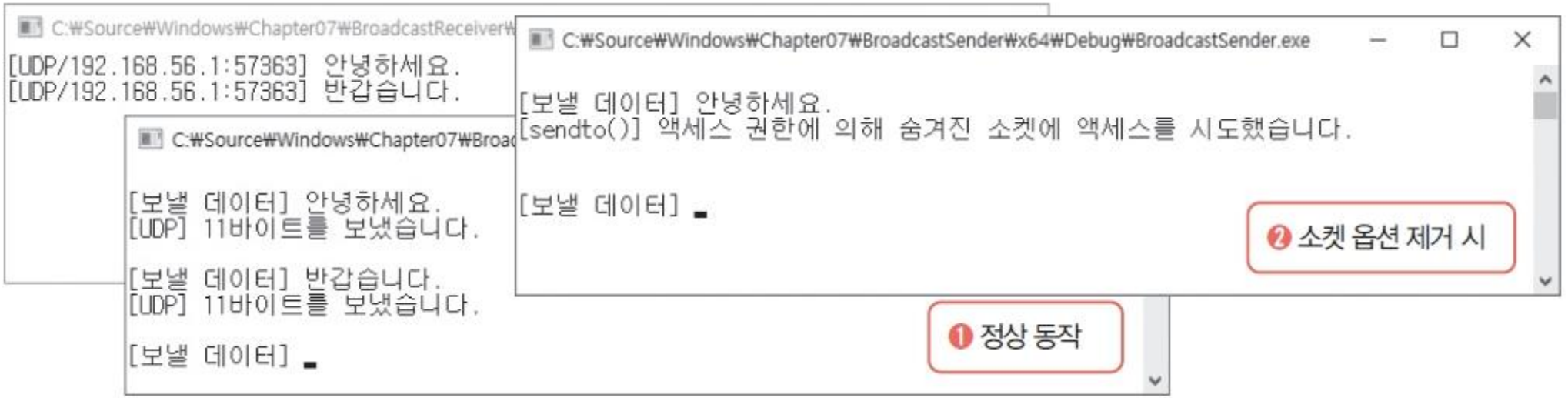


사용자가 키보드로 입력한(fgets) 문자열을 브로드캐스트 주소로 보낸다(sendto).

브로드캐스트 데이터를 받고(recvfrom), 이를 문자열로 간주해 무조건 화면에 출력한다.

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