

REPORT



과 목 :	네트워크프로그래밍
제출일자 :	2021. 11. 17
담당교수 :	황성호
학 과 :	컴퓨터공학과
학 번 :	201720970
이 름 :	권대한

1. UDP_Server 소스코드

```
#pragma warning(disable : 4996)

#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>

#define SERVERPORT 9000
#define BUFFERSIZE 512

// 소켓 함수 오류 출력 후 종료
//general error message output Func
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    printf("[%s] %s", msg, (char*)lpMsgBuf);
}
```

```

        LocalFree(lpMsgBuf);
    }

int main(int argc, char* argv[])
{
    int retval;

    // 원속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        return 1;
    }

    // Socket()
    SOCKET sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock == INVALID_SOCKET)
    {
        err_quit("socket()");
    }

    // bind()
    SOCKADDR_IN serveraddr;
    ZeroMemory(&serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serveraddr.sin_port = htons(SERVERPORT);
    retval = bind(sock, (SOCKADDR*)&serveraddr, sizeof(serveraddr));
    if (retval == SOCKET_ERROR)
    {
        err_quit("bind()");
    }

    // 데이터 통신에 사용할 변수
    SOCKADDR_IN clientaddr;
    int addrlen;
    char buf[BUFFERSIZE + 1];

    // 클라이언트와 데이터 통신
    while (1)
    {

```

```

// 데이터 받기
addrlen = sizeof(clientaddr);
retval = recvfrom(sock, buf, BUFFERSIZE, 0, (SOCKADDR*)&clientaddr, &addrlen);
if (retval == SOCKET_ERROR)
{
    err_display("recvfrom()");
    continue;
}

// 받은 데이터 출력
buf[retval] = '\0';
printf("[UDP /%s:%d] %s\n", inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port),
buf);

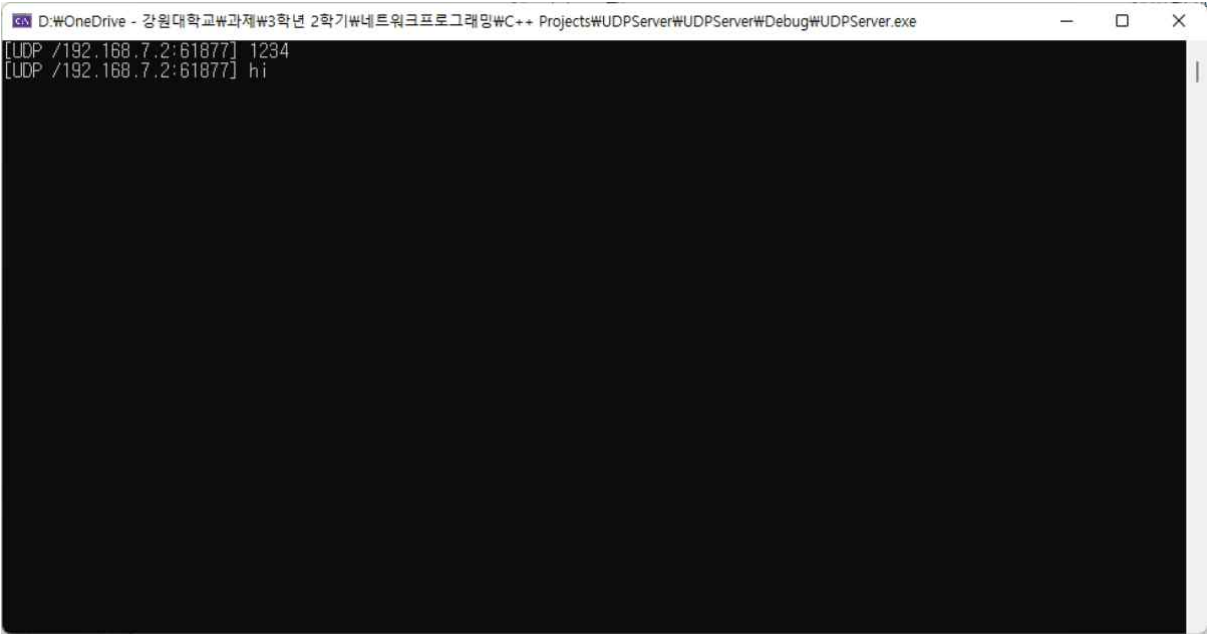
// 데이터 보내기
retval = sendto(sock, buf, retval, 0, (SOCKADDR*)&clientaddr, sizeof(clientaddr));
if (retval == SOCKET_ERROR)
{
    err_display("sendto()");
    continue;
}
}

// Closesocket()
closesocket(sock);

// 윈속 종료
WSACleanup();
return 0;
}

```

2. UDP_Server 출력결과



The screenshot shows a C++ console window titled "D:\OneDrive - 강원대학교\학과제\3학년 2학기\네트워크프로그래밍\C++ Projects\UDPServer\UDPServer\Debug\UDPServer.exe". The console output is as follows:

```
[UDP /192.168.7.2:61877] 1234
[UDP /192.168.7.2:61877] hi
```

- UDP_Server Wireshark Activity

35	28.827919	192.168.7.2	192.168.7.1	UDP	60	61876 → 9000	Len=4
36	28.829545	192.168.7.1	192.168.7.2	UDP	46	9000 → 61876	Len=4
37	29.139985	Cisco_cc:4b:0e	Cisco_cc:4b:0e	LOOP	60	Reply	
38	29.572268	Cisco_cc:4b:0e	CDP/VTP/DTP/PAGP/UD...	DTP	60	Dynamic Trunk Protocol	
39	29.572987	Cisco_cc:4b:0e	CDP/VTP/DTP/PAGP/UD...	DTP	90	Dynamic Trunk Protocol	
40	29.623095	Cisco_cc:4b:0e	Spanning-tree-(for-...	STP	60	Conf. Root = 32768/1/00:07:50:cc:4b:00 Cost = 0 Port = 0x800e	
41	31.631059	Cisco_cc:4b:0e	Spanning-tree-(for-...	STP	60	Conf. Root = 32768/1/00:07:50:cc:4b:00 Cost = 0 Port = 0x800e	
42	32.522624	192.168.7.2	192.168.7.1	UDP	60	61876 → 9000	Len=2
43	32.522813	192.168.7.1	192.168.7.2	UDP	44	9000 → 61876	Len=2

3. UDP_Client 소스코드

```
#pragma warning(disable : 4996)

#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>

#define SERVERIP "192.168.7.2"
#define SERVERPORT 9000
#define BUFFERSIZE 512

// 소켓 함수 오류 출력 후 종료
//general error message output Func
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
```

```

    printf("[%s] %s", msg, (char*)lpMsgBuf);
    LocalFree(lpMsgBuf);
}

int main(int argc, char* argv[])
{
    int retval;

    // 윈속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        return 1;
    }

    // Socket()
    SOCKET sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock == INVALID_SOCKET)
    {
        err_quit("socket()");
    }

    // 소켓 주소 구조체 초기화
    SOCKADDR_IN serveraddr;
    ZeroMemory(&serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = inet_addr(SERVERIP);
    serveraddr.sin_port = htons(SERVERPORT);

    // 데이터 통신에 사용할 변수
    SOCKADDR_IN peeraddr;
    int addrlen;
    char buf[BUFFERSIZE + 1];
    int len;

    // 서버와 데이터 통신
    while (1)
    {
        // 데이터 입력
        printf("\n[보낼 데이터] ");
    }

```

```

if (fgets(buf, BUFFERSIZE + 1, stdin) == NULL)
{
    break;
}

// '\n' 문자 제거
len = strlen(buf);
if (buf[len - 1] == '\n')
{
    buf[len - 1] = '\0';
}
if (strlen(buf) == 0)
{
    break;
}

// 데이터 보내기
retval = sendto(sock, buf, strlen(buf), 0, (SOCKADDR*)&serveraddr, sizeof(serveraddr));
if (retval == SOCKET_ERROR)
{
    err_display("sendto()");
    continue;
}

printf("[UDP 클라이언트] %d바이트를 보냈습니다.\n", retval);

// 데이터 받기
addrlen = sizeof(peeraddr);
retval = recvfrom(sock, buf, BUFFERSIZE, 0, (SOCKADDR*)&peeraddr, &addrlen);
if (retval == SOCKET_ERROR)
{
    err_display("recvfrom()");
    continue;
}

// 송신자의 IP 주소 체크
if (memcmp(&peeraddr, &serveraddr, sizeof(peeraddr)))
{
    printf("[오류] 잘못된 데이터입니다!\n");
    continue;
}

```



```

    }

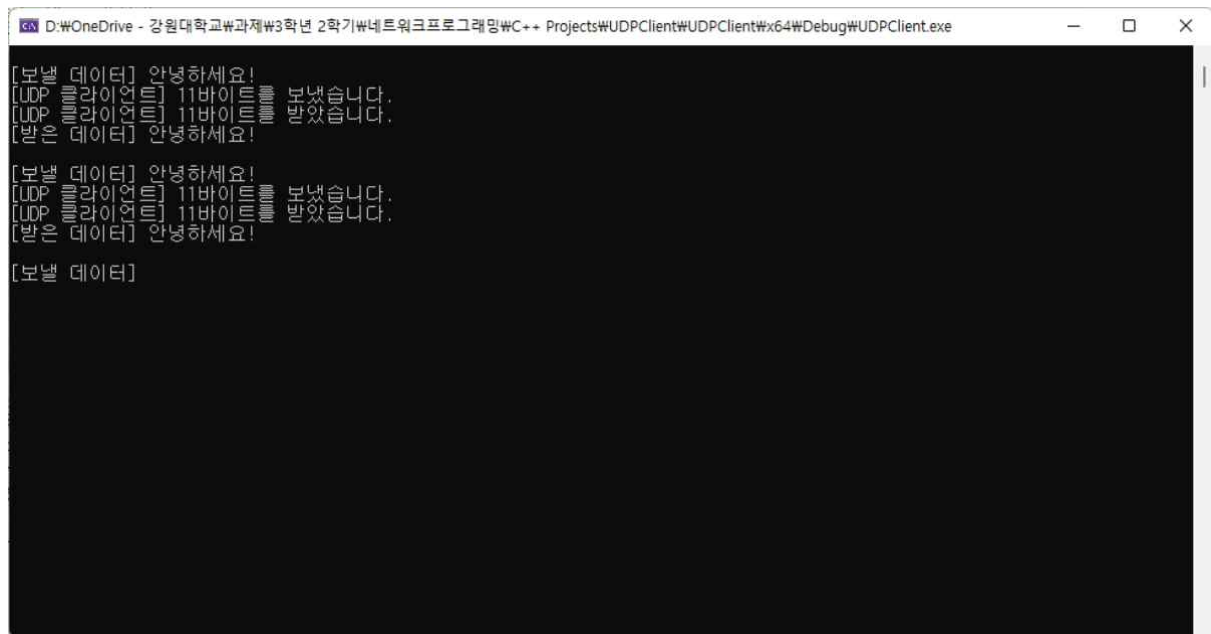
    // 받은 데이터 출력
    buf[retval] = '\0';
    printf("[UDP 클라이언트] %d바이트를 받았습니다.\n", retval);
    printf("[받은 데이터] %s\n", buf);
}

// Closesocket()
closesocket(sock);

// 윈속 종료
WSACleanup();
return 0;
}

```

4. UDP_Client 출력결과



```
D:\OneDrive - 강원대학교원격교육3학년 2학기\네트워크프로그래밍\C++ Projects\UDPClient\UDPClient\x64\Debug\UDPClient.exe
[보낼 데이터] 안녕하세요!
[UDP 클라이언트] 11바이트 보냈습니다.
[UDP 클라이언트] 11바이트 받았습니다.
[받은 데이터] 안녕하세요!

[보낼 데이터] 안녕하세요!
[UDP 클라이언트] 11바이트 보냈습니다.
[UDP 클라이언트] 11바이트 받았습니다.
[받은 데이터] 안녕하세요!

[보낼 데이터]
```

- UDP_Client Wireshark Activity

19	25.620885	192.168.7.1	192.168.7.2	UDP	53 64705 → 9000 Len=11	
20	25.622421	192.168.7.2	192.168.7.1	UDP	60 9000 → 64705 Len=11	
21	26.000766	Cisco_cc:4b:0e	Spanning-tree-(for-...	STP	60 Conf. Root = 32768/1/00:07:50:cc:4b:00 Cost = 0 Port = 0x800e	
22	28.000847	Cisco_cc:4b:0e	Spanning-tree-(for-...	STP	60 Conf. Root = 32768/1/00:07:50:cc:4b:00 Cost = 0 Port = 0x800e	
23	30.002095	Cisco_cc:4b:0e	Spanning-tree-(for-...	STP	60 Conf. Root = 32768/1/00:07:50:cc:4b:00 Cost = 0 Port = 0x800e	

5. BroadcastSender 소스코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>

#define REMOTEIP "255.255.255.255"
#define REMOTEPORT 9000
#define BUFSIZE 512

// 소켓 함수 오류 출력 후 종료
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;

    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf,
        0,
        NULL
    );

    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;

    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
```

```

        (LPTSTR)&lpMsgBuf,
        0,
        NULL
    );
    printf("[%s] %s", msg, (char*)lpMsgBuf);
    LocalFree(lpMsgBuf);
}

```

```

int main(int argc, char* argv[])
{

```

```

    int retval;

```

```

    // 윈속 초기화

```

```

    WSADATA wsa;

```

```

    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

```

```

    {

```

```

        return 1;

```

```

    }

```

```

    // Socket()

```

```

    SOCKET sock = socket(AF_INET, SOCK_DGRAM, 0);

```

```

    if (sock == INVALID_SOCKET)

```

```

    {

```

```

        err_quit("socket()");

```

```

    }

```

```

    // 브로드캐스팅 활성화

```

```

    BOOL bEnable = TRUE;

```

```

    retval = setsockopt(sock, SOL_SOCKET, SO_BROADCAST, (char*)&bEnable, sizeof(bEnable));

```

```

    if (retval == SOCKET_ERROR)

```

```

    {

```

```

        err_quit("setsockopt()");

```

```

    }

```

```

    // 소켓 주소 구조체 초기화

```

```

    SOCKADDR_IN remoteaddr;

```

```

    ZeroMemory(&remoteaddr, sizeof(remoteaddr));

```

```

    remoteaddr.sin_family = AF_INET;

```

```

    remoteaddr.sin_addr.s_addr = inet_addr(REMOTEIP);

```

```

    remoteaddr.sin_port = htons(REMOTEPORT);

```

```

// 데이터 통신에 사용할 변수
char buf[BUFSIZE + 1];
int len;

// 브로드캐스트 데이터 보내기
while (1)
{
    // 데이터 입력
    printf("\n[보낼 데이터] ");
    if (fgets(buf, BUFSIZE + 1, stdin) == NULL)
    {
        break;
    }

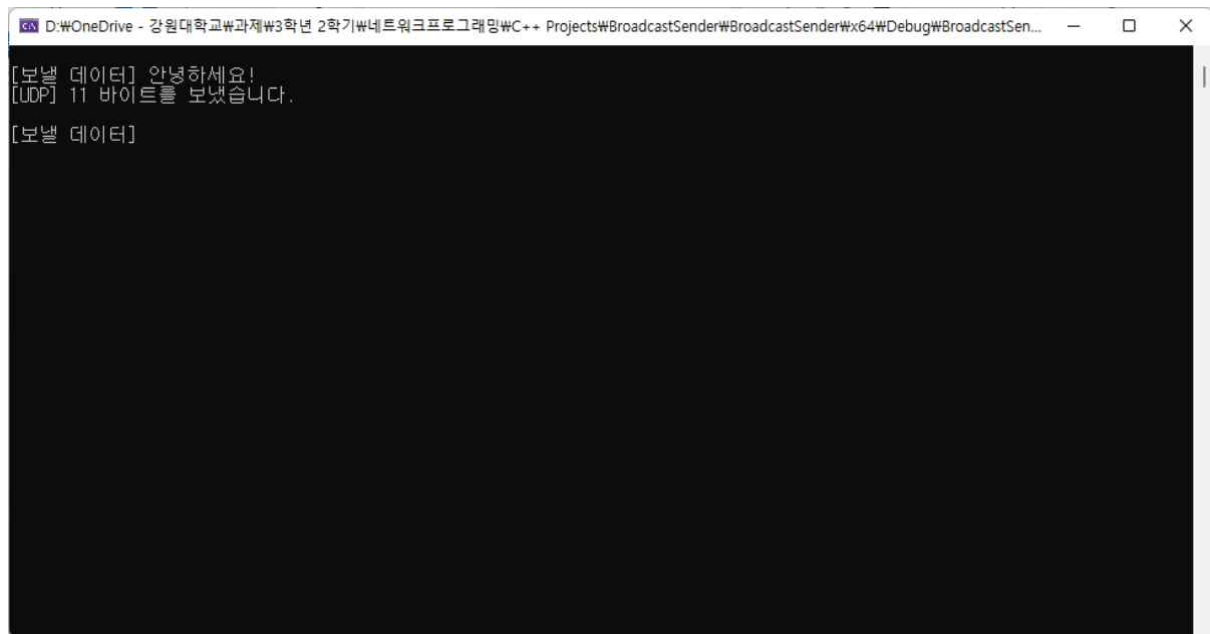
    // '\n' 문자 제거
    len = strlen(buf);
    if (buf[len - 1] == '\n')
    {
        buf[len - 1] = '\0';
    }
    if (strlen(buf) == 0)
        break;

    // 데이터 보내기
    retval = sendto(sock, buf, strlen(buf), 0, (SOCKADDR*)&remoteaddr,
sizeof(remoteaddr));
    if (retval == SOCKET_ERROR)
    {
        err_display("sendto()");
        continue;
    }
    printf("[UDP] %d 바이트를 보냈습니다.\n", retval);
}

// Closesocket()
closesocket(sock);
// 윈속 종료
WSACleanup();
return 0;
}

```

6. BroadcastSender 출력결과



```
D:\OneDrive - 강원대학교\학과제\3학년 2학기\네트워크프로그래밍\C++ Projects\BroadcastSender\BroadcastSender\Debug\BroadcastSen...  
[보낼 데이터] 안녕하세요!  
[UDP] 11 바이트를 보냈습니다.  
[보낼 데이터]
```

- BroadcastSender Wireshark Activity

Server: 192.168.7.2

Case. 192.168.7.1과 192.168.7.4가 동시에 x.2를 찾기 위해 브로드캐스트 하였을 경우

15	10.278336	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
16	11.028003	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
17	11.779973	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
19	12.599490	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
20	13.351078	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
23	14.101756	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
70	61.649857	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
73	62.400244	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
76	63.151641	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
91	71.626119	192.168.7.2	192.168.7.255	NBNS	92 Name query NB	DESKTOP-1TQHT8D<1c>
93	72.391551	192.168.7.2	192.168.7.255	NBNS	92 Name query NB	DESKTOP-1TQHT8D<1c>
94	73.155471	192.168.7.2	192.168.7.255	NBNS	92 Name query NB	DESKTOP-1TQHT8D<1c>
115	88.688378	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>
116	89.439277	192.168.7.4	192.168.7.255	NBNS	92 Name query NB	DESKTOP-N6NI4UF<1c>

7. BroadcastReceiver 소스코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>

#define LOCALPORT 9000
#define BUFSIZE 512

// 소켓 함수 오류 출력 후 종료
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;

    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf,
        0,
        NULL
    );
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
```

```

        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf,
        0,
        NULL
    );
    printf("[%s] %s", msg, (char*)lpMsgBuf);
    LocalFree(lpMsgBuf);
}

```

```

int main(int argc, char* argv[])
{
    int retval;

    // 윈속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        return 1;
    }

    // Socket()
    SOCKET sock = socket(AF_INET, SOCK_DGRAM, 0);
    if (sock == INVALID_SOCKET)
    {
        err_quit("socket()");
    }

    // Bind()
    SOCKADDR_IN localaddr;
    ZeroMemory(&localaddr, sizeof(localaddr));
    localaddr.sin_family = AF_INET;
    localaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    localaddr.sin_port = htons(LOCALPORT);
    retval = bind(sock, (SOCKADDR*)&localaddr, sizeof(localaddr));
    if (retval == SOCKET_ERROR)
    {
        err_quit("bind()");
    }

    // 데이터 통신에 사용할 변수
    SOCKADDR_IN peeraddr;
    char buf[BUFSIZE + 1];
}

```



```

int addrlen;

// 브로드캐스트 데이터 받기
while (1)
{
    // 데이터 받기
    addrlen = sizeof(peeraddr);
    retval = recvfrom(sock, buf, BUFSIZE, 0, (SOCKADDR*)&peeraddr, &addrlen);
    if (retval == SOCKET_ERROR)
    {
        err_display("recvfrom()");
        continue;
    }

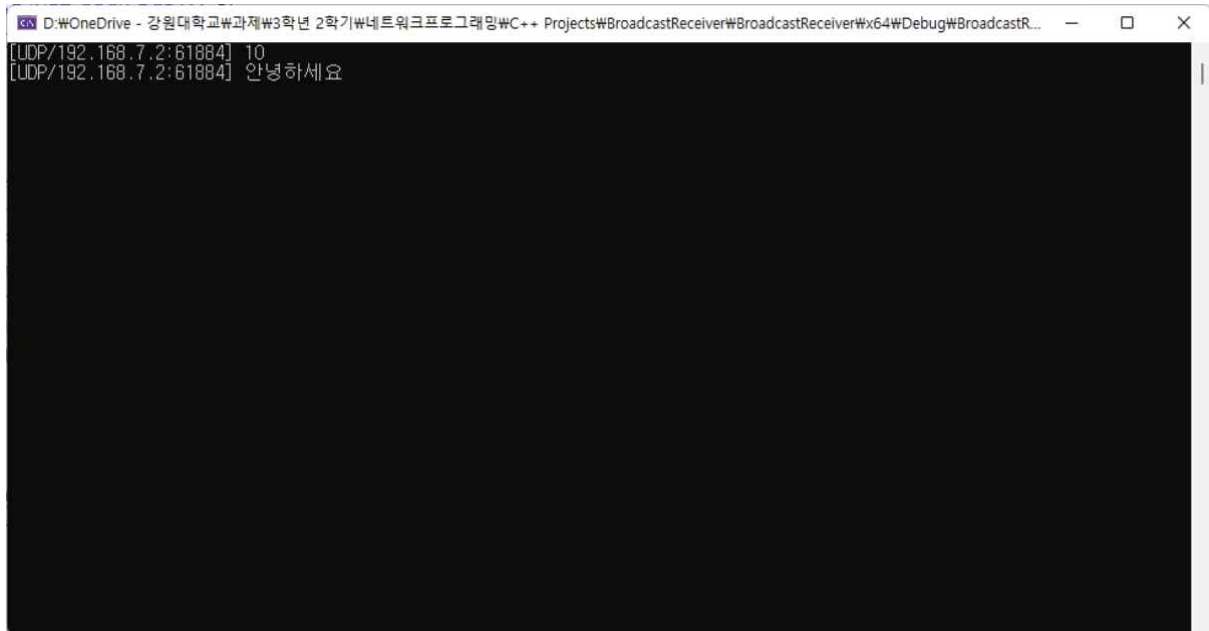
    // 받은 데이터 출력
    buf[retval] = '\0';
    printf("[UDP/%s:%d] %s\n", inet_ntoa(peeraddr.sin_addr), ntohs(peeraddr.sin_port),
buf);
}

// Closesocket()
closesocket(sock);

// 윈속 종료
WSACleanup();
return 0;
}

```

8. BroadcastReceiver 출력결과



```
D:\OneDrive - 강원대학교\학과\제3학년 2학기\네트워크프로그래밍\C++ Projects\BroadcastReceiver\BroadcastReceiver\Debug\BroadcastR...  
[UDP/192.168.7.2:61884] 10  
[UDP/192.168.7.2:61884] 안녕하세요
```

- BroadcastReceiver Wireshark Activity

Server: 192.168.7.1(본인)

Case. 192.168.7.2와 192.168.7.4가 동시에 x.1를 찾기 위해 브로드캐스트 하였을 경우

36	22.325245	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
37	23.075693	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
53	42.539230	192.168.7.2	192.168.7.255	NBNS	92 Name query NB DESKTOP-1TQHT8D<1c>
54	43.293692	192.168.7.2	192.168.7.255	NBNS	92 Name query NB DESKTOP-1TQHT8D<1c>
55	44.057261	192.168.7.2	192.168.7.255	NBNS	92 Name query NB DESKTOP-1TQHT8D<1c>
62	48.785639	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
63	49.536266	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
65	50.286943	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
67	53.214927	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
68	53.964709	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
70	54.715463	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
71	56.072237	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
74	56.823042	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>
76	57.574094	192.168.7.4	192.168.7.255	NBNS	92 Name query NB DESKTOP-N6NI4UF<1c>

9. PC에서의 고정 IP address 설정

인터넷 프로토콜 버전 4(TCP/IPv4) 속성

일반

네트워크가 IP 자동 설정 기능을 지원하면 IP 설정이 자동으로 할당되도록 할 수 있습니다. 지원하지 않으면, 네트워크 관리자에게 적절한 IP 설정값을 문의해야 합니다.

☐ 자동으로 IP 주소 받기(O)

☒ 다음 IP 주소 사용(S)

IP 주소(I):

192 . 168 . 7 . 1

서브넷 마스크(U):

255 . 255 . 255 . 0

기본 게이트웨이(D):

. . .

☐ 자동으로 DNS 서버 주소 받기(B)

☒ 다음 DNS 서버 주소 사용(E)

기본 설정 DNS 서버(P):

. . .

보조 DNS 서버(A):

. . .

☐ 끝낼 때 설정 유효성 검사(L)

고급(V)...

확인

취소

10. ping 테스트 결과를 제출하시오.

Case 1. 192.168.7.2간 ICMP 통신 결과

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\Users\daeha> ping 192.168.7.2

Ping 192.168.7.2 32바이트 데이터 사용:
192.168.7.2의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.2의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.2의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.2의 응답: 바이트=32 시간<1ms TTL=128

192.168.7.2에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
PS C:\Users\daeha> |
```

Case 2. 192.168.7.4간 ICMP 통신 결과

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\Users\daeha> ping 192.168.7.4

Ping 192.168.7.4 32바이트 데이터 사용:
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128

192.168.7.4에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
PS C:\Users\daeha> |
```