基于 winsock2 的 Windows 系统多播程序

71118415 叶宏庭

东南大学软件学院

Email: 213182964@seu.edu.cn

June 1, 2021

1 实验目的

通过课程学习，已经对多播原理有了初步的了解，接下来要做的便是完成具体的编程实现，为此本实验主要是完成一个基于 winsock2 的多播程序。

2 实验环境

2.1 操作系统:

Windows 10

2.2 编译环境:

MinGW 8.1

2.3 辅助软件:

CTEX(用于编写 tex 报告)，VS code(用于编写程序)

3 实验内容

3.1 多播介绍:

1988 年，Deering 提出了 IP 多播的概念，从此 IP 多播技术得到了广泛的关注。多播介于单播通信和广播通信之间，它可以将发送者发送的数据包发送给位于分散在不同子网中的一组接收者。

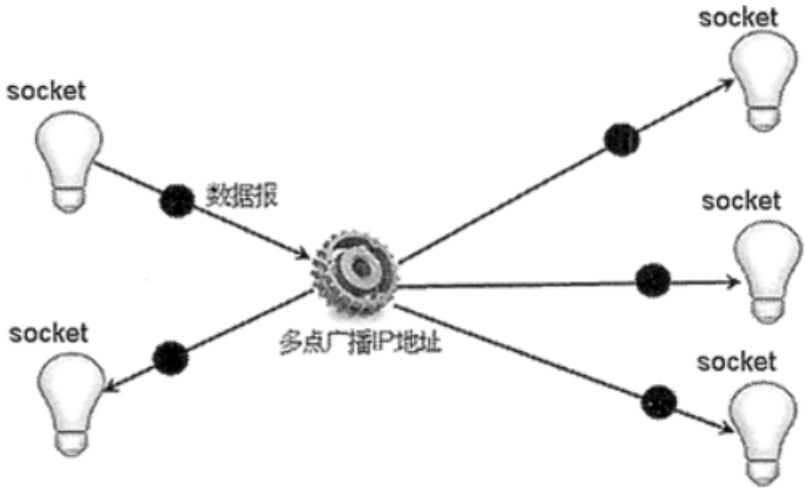
多播的基础概念是 “组”。一个多播组 (multicast group) 就是一组希望接收特定数据流的接收者。这个组没有物理或者地理的边界：组内的主机可以位于互联网或者专用网络的任何地方。多播组中的每一个节点被称为多播组成员 (multicastgroupmember)。

3.2 多播原理:

通过多点广播，可以将数据报以广播方式式发送到多个客户端。

若要使用多点广播，则需要将数据报发送到一个组目标地址，当数据报发出后，整个组的所有主机都能接收到该数据报。IP 多点广播（或多点发送）实现了将单一信息发送给多个接收者的广播，其思想是设置一组特殊的网络地址作为多点广播地址，每一个多点广播地址都被看作一个组，当客户端需要发送和接收广播信息时，加入该组即可。

IP 协议为多点广播提供了特殊的 IP 地址，这些 IP 地址的范围是 224.0.0.0 239.255.255.255。多点广播示意图如下图所示。



从上图中可以看出，当 socket 把一个数据报发送到多点广播 IP 地址时，该数据报将被自动广播到加入该地址的所有 socket。该 socket 既可以将数据报发送到多点广播地址，也可以接收其他主机的广播信息。

3.3 winsock2 介绍:

WinSock2 是连接系统和用户使用的软件之间用于交流的一个接口，这个功能就是修复软件与系统正确的通讯的作用。

Winsock2 SPI（Service Provider Interface）服务提供者接口建立在 Windows 开放系统架构 WOSA（Windows Open System Architecture）之上，是 Winsock 系统组件提供的面向系统底层的编程接口。Winsock 系统组件向上面向用户应用程序提供一个标准的 API 接口；向下在 Winsock 组件和 Winsock 服务提供者（比如 TCP/IP 协议栈）之间提供一个标准的 SPI 接口。各种服务提供者是 Windows 支持的 DLL，挂载在 Winsock2 的 Ws2_32.dll 模块下。对用户应用程序使用的 Winsock2 API 中定义的许多内部函数来说，这些服务提供者都提供了它们的对应的运作方式 (例如 API 函数 WSAConnect 有相应的 SPI 函数 WSPConnect)。多数情况下，一个应用程序在调用 Winsock2 API 函数时，Ws2_32.dll 会调用相应的 Winsock2 SPI 函数，利用特定的服务提供者执行所请求的服务。

简单来说，winsock2 为 Windows 系统提供了 socket 编程的底层架构，使用户可以在 Windows 系统下完成 socket 编程。

3.4 部分代码解析:

3.4.1 win_re.cpp(接受端程序):

定义网络参数

```
#define MCASTADDR "234.5.6.7"
#define MCASTPORT 12345
#define BUFSIZE 1024
#define DEFAULT_COUNT 500
```

套接字等变量

```
        WSADATA wsd;
        struct sockaddr_in local,
        remote,
        from;
        SOCKET sock, sockM;
        TCHAR recvbuf[BUFSIZE],
        sendbuf[BUFSIZE];
        int len = sizeof(struct sockaddr_in),
        optval,
        ret;
        DWORD i=0;
```

绑定本地 local socket

```
        local.sin_family=AF_INET;
        local.sin_port = htons(MCASTPORT);
        local.sin_addr.s_addr = dwInterface;

        if(bind(sock,(struct sockaddr *)&local ,sizeof(local))
                        ==SOCKET_ERROR){
                printf("bind failed with :%d \n", WSAGetLastError());
                closesocket(sock);
                WSACleanup();
                return -1;
        }
```

加入多播组

```
        remote.sin_family =AF_INET;
        remote.sin_port = htons(iPort);
        remote.sin_addr.s_addr = dwMulticastGroup;
        if((sockM = WSAJoinLeaf(sock,(SOCKADDR *)&remote,
                        sizeof(remote),NULL,NULL,NULL,NULL,JL_BOTH))
                        ==INVALID_SOCKET){
```

```
            printf("WSAJoinLeaf() failed : %d\n", WSAGetLastError());
            closesocket(sock);
            WSACleanup();
            return -1;
        }
```

接受数据并输出

```
        printf("Start rece...\n");
        for(i = 0;i < dwCount;i++)
        {
            //printf("hhh");
            if((ret = recvfrom(sock,recvbuf,BUFSIZE,0,
                    (struct sockaddr *)&from , &len))
                    ==SOCKET_ERROR){
                    printf("recvfrom failed with: %d\n" ,
                                    WSAGetLastError());
                    closesocket(sockM);
                    closesocket(sock);
                    WSACleanup();
                    return -1;
            }

            recvbuf[ret] = 0;
            printf("RECV: '%s' from <%s>\n", recvbuf,
                            inet_ntoa(from.sin_addr));
        }
```

3.4.2 win_se.cpp(发送端程序):

对于网络设置部分，两个程序均采用了相同配置，所以此从只给出发送数据的部分代码。
发送数据

```
        printf("Start send...\n");
        for ( i = 0; i < dwCount; i++)
        {
            sprintf(sendbuf, "server 1: This is a test: %d", i);
            printf("%s\n", sendbuf);
            if (sendto(sock,(char *)sendbuf,strlen(sendbuf),0,
                            (struct sockaddr *) &remote,
                            sizeof(remote))
                            ==SOCKET_ERROR){
                    printf("sendto failed with: %d\n",
```

```
11                                          WSAGetLastError());
12                          closesocket(sockM);
13                          closesocket(sock);
14                          WSACleanup();
15                          return -1;
16                  }
17                  Sleep(500);
18          }
```

4 实验结果与分析

首先给出程序的运行结果：

4.1 发送端结果：

```
PS E:\学习资料\课件\程序猿\网络编程\实验8_1\t2> .\send.exe
Start send...
server 1: This is a test: 0
server 1: This is a test: 1
server 1: This is a test: 2
server 1: This is a test: 3
server 1: This is a test: 4
server 1: This is a test: 5
server 1: This is a test: 6
server 1: This is a test: 7
server 1: This is a test: 8
server 1: This is a test: 9
server 1: This is a test: 10
server 1: This is a test: 11
server 1: This is a test: 12
server 1: This is a test: 13
server 1: This is a test: 14
server 1: This is a test: 15
server 1: This is a test: 16
server 1: This is a test: 17
server 1: This is a test: 18
server 1: This is a test: 19
server 1: This is a test: 20
```

4.2 接收端结果：

```
D:\QQ文件\3240540073\FileRecv\rece.exe
Start rece...
RECV: 'server 1: This is a test: 0' from <10.208.103.250>
RECV: 'server 1: This is a test: 1' from <10.208.103.250>
RECV: 'server 1: This is a test: 2' from <10.208.103.250>
RECV: 'server 1: This is a test: 3' from <10.208.103.250>
RECV: 'server 1: This is a test: 4' from <10.208.103.250>
RECV: 'server 1: This is a test: 5' from <10.208.103.250>
RECV: 'server 1: This is a test: 6' from <10.208.103.250>
RECV: 'server 1: This is a test: 7' from <10.208.103.250>
RECV: 'server 1: This is a test: 8' from <10.208.103.250>
RECV: 'server 1: This is a test: 9' from <10.208.103.250>
RECV: 'server 1: This is a test: 10' from <10.208.103.250>
RECV: 'server 1: This is a test: 11' from <10.208.103.250>
RECV: 'server 1: This is a test: 12' from <10.208.103.250>
RECV: 'server 1: This is a test: 13' from <10.208.103.250>
RECV: 'server 1: This is a test: 14' from <10.208.103.250>
```

4.3 分析

从上方两个截图中可以清楚看到，发送端发出的报文数据，被接收端准确接收，因此，本次多播实

验是一次成功的实验。

4.4 总结

通过本次实验，首先是了解了多播的原理与机制，再学习了 winsock2 的编程，掌握了 Windows 下的 socket 编程，本次实验过程中还涉及一些网络，网卡的配置，可谓是收获颇丰。