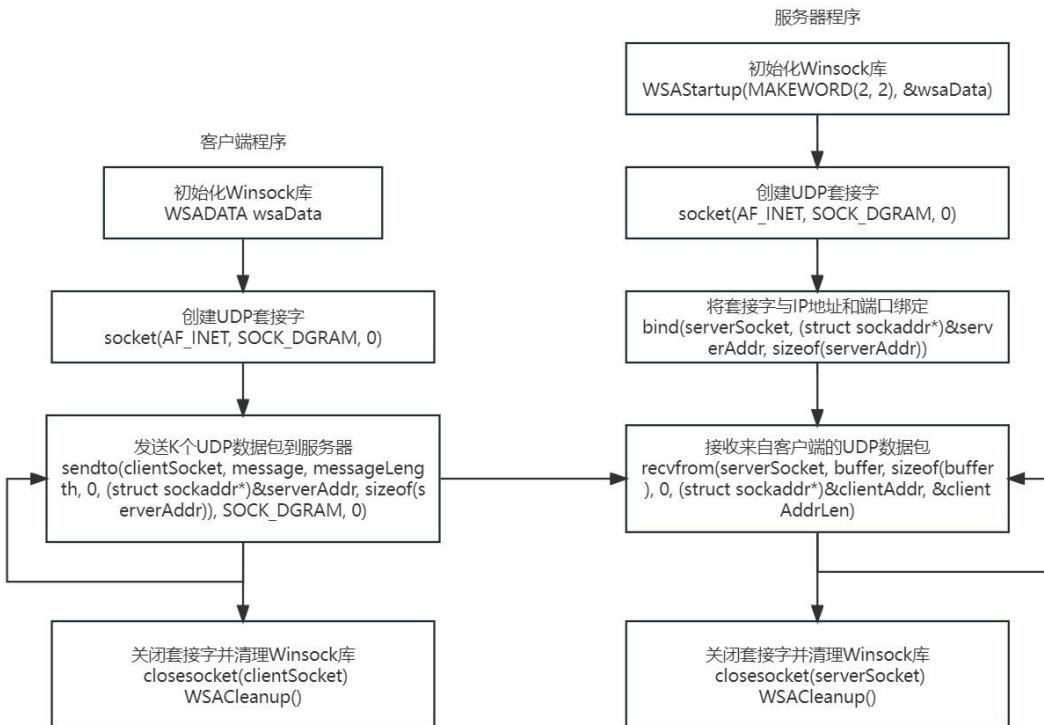


计算机网络 实验四

一、软件设计流程图



二、源代码

1、客户端程序

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS //用于屏蔽 visual studio 对 inet_addr 函数的报错
#define _CRT_SECURE_NO_WARNINGS //用于屏蔽 visual studio 对 sprintf 函数的报错

#include <iostream>
using namespace std;

#include <Winsock2.h> //静态链接库 winsock2.h 是用来编译基于 Winsock 程序的
#pragma comment(lib, "ws2_32.lib") //动态链接库 ws2_32.dll 是运行基于 WinSock 的程序所必须的

int main() {
    int K;
    cout << "请输入要发生的 UDP 数据包数量 K: ";
    cin >> K;

    // 初始化 Winsock 库 (如果不初始化, 下一步创建 UDP 套接字会失败)
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        cerr << "WSAStartup 失败" << endl;
        return 1;
    }
    // WSAStartup() 用于初始化 Winsock 库
    // 函数原型: int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData);
    // wVersionRequested: 指定请求的 Winsock 版本。通常设置为 MAKEWORD(2, 2), 以请求 Winsock 2.2 版本, 这是一个常见的版本, 支持大多数网络功能。
    // lpWSAData: 指向 WSADATA 结构的指针, 用于接收关于 Winsock 库初始化的信息。
    // 返回值: 如果函数成功, 返回值为 0。如果函数失败, 返回值为非零错误代码
```

```

// 创建 UDP 套接字
SOCKET clientSocket = socket(AF_INET, SOCK_DGRAM, 0);
if (clientSocket == INVALID_SOCKET) {
    cerr << "套接字创建失败" << endl;
    WSACleanup();
    return 1;
}
//socket()函数是用于在网络编程中创建套接字
//函数原型: int socket(int domain, int type, int protocol);
//domain: 套接字的通信域或地址族, 指定套接字的地址类型。常见的值包括 AF_INET (IPv4) 和 AF_INET6 (IPv6)。
//type: 套接字的类型, 定义了套接字的特性和行为。常见的值包括 SOCK_STREAM (流套接字, 用于 TCP 协议) 和 SOCK_DGRAM (数据报套接字, 用于 UDP 协议)。
//protocol: 套接字使用的协议, 通常设置为 0, 表示根据 domain 和 type 自动选择合适的协议。
//返回值:
//如果 socket() 函数成功创建套接字, 它将返回一个非负整数, 表示套接字描述符。这个描述符是后续网络通信的标识符。
//如果 socket() 函数失败, 它将返回 -1, 表示错误。

// 发送 K 个 UDP 数据包到服务器
sockaddr_in serverAddr;//sockaddr_in 是用于 IPv4 地址的套接字地址结构, 特定于 IPv4 地址族 (AF_INET)
serverAddr.sin_family = AF_INET;//设置地址族为 AF_INET, 表示使用 IPv4 地址。
serverAddr.sin_addr.s_addr = inet_addr("172.20.10.11");// 设置服务器的 IP 地址。inet_addr 函数将字符串形式的 IPv4 地址转换为
为网络字节序的整数表示形式
serverAddr.sin_port = htons(8080); // 设置服务器的 UDP 监听端口号。htons 函数将主机字节序 (通常是小端字节序) 的端口号转换为
网络字节序 (大端字节序), 以便在网络上传输。
char message[128];

for (int i = 0; i < K; i++) {
    sprintf(message, "%d 号 UDP 测试数据", i + 1);
    int messageLength = strlen(message);
    int bytesSent = sendto(clientSocket, message, messageLength, 0, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
    if (bytesSent == SOCKET_ERROR) {
        cerr << "发送失败" << endl;
        break;
    }
    cout << "已向服务器 8080 端口发送 UDP 数据包" << i + 1 << ", 内容为: \\" " << message << "\\\" " << endl;
    Sleep(500); // 延迟 0.5 秒
}
// sendto() 函数用于在套接字编程中用于发送数据, 通常与 UDP 套接字一起使用
// 函数原型: int sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);
// sockfd: 表示发送数据的套接字描述符, 它标识了哪个套接字将要发送数据。
// buf: 是一个指向要发送数据的缓冲区的指针。
// len: 表示要发送的数据的字节数。
// flags: 通常可以设置为 0, 表示没有特殊标志位。
// dest_addr: 是一个指向 sockaddr 结构的指针, 用于指定目标地址和端口。这个参数通常在 UDP 套接字中用于指定数据报的目标地址。
// addrlen: 表示 dest_addr 结构的长度。
// 返回值: 如果 sendto 函数成功发送数据, 其将返回发送的字节数。如果发送失败, 返回 -1。

// 关闭套接字并清理 Winsock 库
closesocket(clientSocket);
// closesocket 函数用于关闭套接字
// 返回值: 如果 closesocket 函数成功关闭套接字, 它将返回 0。如果关闭失败, 它将返回 SOCKET_ERROR
WSACleanup();
// WSACleanup 用于清理和释放 Winsock 库所占用资源
// 返回值: 如果 WSACleanup 函数成功执行, 它将返回 0。如果清理失败, 它将返回 WSAENETDOWN, 表示 Windows 套接字库已关闭或未初始化。

return 0;
}

```

2、服务器程序

```

#define _WINSOCK_DEPRECATED_NO_WARNINGS //用于屏蔽 visual studio 对 inet_addr 函数的报错
#include <iostream>
using namespace std;
#include <csignal> //信号处理
#include <unordered_map> //哈希表

#include <Winsock2.h> //静态链接库 winsock2.h 是用来编译基于 Winsock 程序的
#pragma comment(lib, "ws2_32.lib") //动态链接库 ws2_32.dll 是运行基于 WinSock 的程序所必须的

bool continueLoop = true;
SOCKET serverSocket;
void signalHandler(int) // 信号处理函数
{
    continueLoop = false; // 设置标志以退出循环

    // 关闭套接字并清理 Winsock 库
    cout << "\n套接字已关闭" << endl;
    closesocket(serverSocket);
    WSACleanup();
}

```

```

}

int main() {
    WSADATA wsaData;
    // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        cerr << "WSAStartup 失败" << endl;
        return 1;
    }

    // 创建 UDP 套接字
    serverSocket = socket(AF_INET, SOCK_DGRAM, 0);
    if (serverSocket == INVALID_SOCKET) {
        cerr << "套接字创建失败" << endl;
        WSACleanup();
        return 1;
    }

    // 将套接字与 IP 地址和端口绑定
    sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;//设置服务器套接字的本地 IP 地址, INADDR_ANY 是一个常量, 通常用于将服务器绑定到所有可用的网络接口上, 以便服务器可以接受来自任何本地接口的连接请求
    serverAddr.sin_port = htons(8080); // 服务器监听的 UDP 端口号
    if (bind(serverSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        cerr << "绑定失败" << endl;
        closesocket(serverSocket);
        WSACleanup();
        return 1;
    }

    // bind 函数是在套接字编程中用于将套接字与本地地址 (IP 地址和端口号) 绑定的函数
    // 函数原型: int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
    // sockfd: 表示要绑定的套接字的描述符, 通常是通过 socket 函数创建的套接字。
    // addr: 是一个指向 sockaddr 结构的指针, 用于指定本地地址信息, 包括 IP 地址和端口号。
    // addrlen: 表示 addr 结构的长度, 通常使用 sizeof(struct sockaddr)。
    cout << "服务器正在端口 8080 上监听 UDP 数据包" << endl;

    // 接收来自客户端的 UDP 数据包
    sockaddr_in clientAddr;//用于接受数据发送方的地址
    int clientAddrLen = sizeof(clientAddr);//地址结构体大小
    unordered_map<string, int> packetCount;//数据包数量计数器
    char buffer[1024] = "\0";//接收报文的缓冲区
    signal(SIGINT, signalHandler);//设置循环的停止条件
    printf("按下 Ctrl+C 来关闭服务器程序\n\n");

    while (continueLoop) {
        int bytesReceived = recvfrom(serverSocket, buffer, sizeof(buffer), 0, (struct sockaddr*)&clientAddr, &clientAddrLen);
        if (bytesReceived == SOCKET_ERROR) {
            cerr << "接收失败或套接字被关闭\n" << endl;
            break;
        }
        packetCount[inet_ntoa(clientAddr.sin_addr)]++;
        cout << "从地址" << inet_ntoa(clientAddr.sin_addr) << "端口" << ntohs(clientAddr.sin_port) << "收到数据包, ";
        // inet_ntoa 函数在套接字编程中用于将 IPv4 地址从网络字节序 (big-endian) 转换为点分十进制字符串表示
        // ntohs 函数是在套接字编程中用于将一个 16 位整数从网络字节序 (big-endian) 转换为主机字节序 (host byte order) 的函数
        cout << "已从此地址共收到" << packetCount[inet_ntoa(clientAddr.sin_addr)] << "个数据包, ";
        cout << "此数据包内容为: " << buffer << endl;
    }

    //recvfrom 函数是在套接字编程中用于接收数据的函数, 通常与 UDP 套接字一起使用
    //函数原型: ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);
    //sockfd: 表示要接收数据的套接字的描述符, 通常是通过 socket 函数创建的套接字。
    //buf: 是一个指向接收数据的缓冲区的指针。
    //len: 表示缓冲区的大小, 即可以接收的最大字节数。
    //flags: 通常可以设置为 0, 表示没有特殊标志位。
    //src_addr: 是一个指向 sockaddr 结构的指针, 用于存储发送数据的远程地址。通过这个参数, 可以确定数据来自哪个远程主机。在使用 recvfrom 时, 通常需要初始化这个结构体, 并将其传递给函数。
    //addrlen: 是一个指向整数的指针, 用于指定 src_addr 结构的长度。在调用 recvfrom 之前, 应将其设置为 src_addr 结构的长度
    //返回值: 如果 recvfrom 函数成功接收数据, 它将返回接收到的字节数。如果接收失败, 它将返回-1。

    return 0;
}

```

三、测试运行截图

1、在客户端 B 机器的命令行运行客户端软件 2 次，K 值分别为 20, 40

B 机器发送情况：

```
Microsoft Visual Studio 调试 × + ▾

请输入要发生的UDP数据包数量K: 20
已向服务器8080端口发送UDP数据包1, 内容为: "1号 UDP 测试数据"
已向服务器8080端口发送UDP数据包2, 内容为: "2号 UDP 测试数据"
已向服务器8080端口发送UDP数据包3, 内容为: "3号 UDP 测试数据"
已向服务器8080端口发送UDP数据包4, 内容为: "4号 UDP 测试数据"
已向服务器8080端口发送UDP数据包5, 内容为: "5号 UDP 测试数据"
已向服务器8080端口发送UDP数据包6, 内容为: "6号 UDP 测试数据"
已向服务器8080端口发送UDP数据包7, 内容为: "7号 UDP 测试数据"
已向服务器8080端口发送UDP数据包8, 内容为: "8号 UDP 测试数据"
已向服务器8080端口发送UDP数据包9, 内容为: "9号 UDP 测试数据"
已向服务器8080端口发送UDP数据包10, 内容为: "10号 UDP 测试数据"
已向服务器8080端口发送UDP数据包11, 内容为: "11号 UDP 测试数据"
已向服务器8080端口发送UDP数据包12, 内容为: "12号 UDP 测试数据"
已向服务器8080端口发送UDP数据包13, 内容为: "13号 UDP 测试数据"
已向服务器8080端口发送UDP数据包14, 内容为: "14号 UDP 测试数据"
已向服务器8080端口发送UDP数据包15, 内容为: "15号 UDP 测试数据"
已向服务器8080端口发送UDP数据包16, 内容为: "16号 UDP 测试数据"
已向服务器8080端口发送UDP数据包17, 内容为: "17号 UDP 测试数据"
已向服务器8080端口发送UDP数据包18, 内容为: "18号 UDP 测试数据"
已向服务器8080端口发送UDP数据包19, 内容为: "19号 UDP 测试数据"
已向服务器8080端口发送UDP数据包20, 内容为: "20号 UDP 测试数据"

C:\Users\28220\Desktop\code\VS\test\test\x64\Debug\test.exe (进程 21236)已退出, 代码为 0。
按任意键关闭此窗口. . .


```

```
Microsoft Visual Studio 调试 × + ▾

已向服务器8080端口发送UDP数据包4, 内容为: "4号 UDP 测试数据"
已向服务器8080端口发送UDP数据包5, 内容为: "5号 UDP 测试数据"
已向服务器8080端口发送UDP数据包6, 内容为: "6号 UDP 测试数据"
已向服务器8080端口发送UDP数据包7, 内容为: "7号 UDP 测试数据"
已向服务器8080端口发送UDP数据包8, 内容为: "8号 UDP 测试数据"
已向服务器8080端口发送UDP数据包9, 内容为: "9号 UDP 测试数据"
已向服务器8080端口发送UDP数据包10, 内容为: "10号 UDP 测试数据"
已向服务器8080端口发送UDP数据包11, 内容为: "11号 UDP 测试数据"
已向服务器8080端口发送UDP数据包12, 内容为: "12号 UDP 测试数据"
已向服务器8080端口发送UDP数据包13, 内容为: "13号 UDP 测试数据"
已向服务器8080端口发送UDP数据包14, 内容为: "14号 UDP 测试数据"
已向服务器8080端口发送UDP数据包15, 内容为: "15号 UDP 测试数据"
已向服务器8080端口发送UDP数据包16, 内容为: "16号 UDP 测试数据"
已向服务器8080端口发送UDP数据包17, 内容为: "17号 UDP 测试数据"
已向服务器8080端口发送UDP数据包18, 内容为: "18号 UDP 测试数据"
已向服务器8080端口发送UDP数据包19, 内容为: "19号 UDP 测试数据"
已向服务器8080端口发送UDP数据包20, 内容为: "20号 UDP 测试数据"
已向服务器8080端口发送UDP数据包21, 内容为: "21号 UDP 测试数据"
已向服务器8080端口发送UDP数据包22, 内容为: "22号 UDP 测试数据"
已向服务器8080端口发送UDP数据包23, 内容为: "23号 UDP 测试数据"
已向服务器8080端口发送UDP数据包24, 内容为: "24号 UDP 测试数据"
已向服务器8080端口发送UDP数据包25, 内容为: "25号 UDP 测试数据"
已向服务器8080端口发送UDP数据包26, 内容为: "26号 UDP 测试数据"
已向服务器8080端口发送UDP数据包27, 内容为: "27号 UDP 测试数据"
已向服务器8080端口发送UDP数据包28, 内容为: "28号 UDP 测试数据"
已向服务器8080端口发送UDP数据包29, 内容为: "29号 UDP 测试数据"
已向服务器8080端口发送UDP数据包30, 内容为: "30号 UDP 测试数据"
已向服务器8080端口发送UDP数据包31, 内容为: "31号 UDP 测试数据"
已向服务器8080端口发送UDP数据包32, 内容为: "32号 UDP 测试数据"
已向服务器8080端口发送UDP数据包33, 内容为: "33号 UDP 测试数据"
已向服务器8080端口发送UDP数据包34, 内容为: "34号 UDP 测试数据"
已向服务器8080端口发送UDP数据包35, 内容为: "35号 UDP 测试数据"
已向服务器8080端口发送UDP数据包36, 内容为: "36号 UDP 测试数据"
已向服务器8080端口发送UDP数据包37, 内容为: "37号 UDP 测试数据"
已向服务器8080端口发送UDP数据包38, 内容为: "38号 UDP 测试数据"
已向服务器8080端口发送UDP数据包39, 内容为: "39号 UDP 测试数据"
已向服务器8080端口发送UDP数据包40, 内容为: "40号 UDP 测试数据"

C:\Users\28220\Desktop\code\VS\test\test\x64\Debug\test.exe (进程 32700)已退出, 代码为 0。
按任意键关闭此窗口. . .


```

可以看到 B 机器分别发送了 20 和 40 个 UDP 数据包

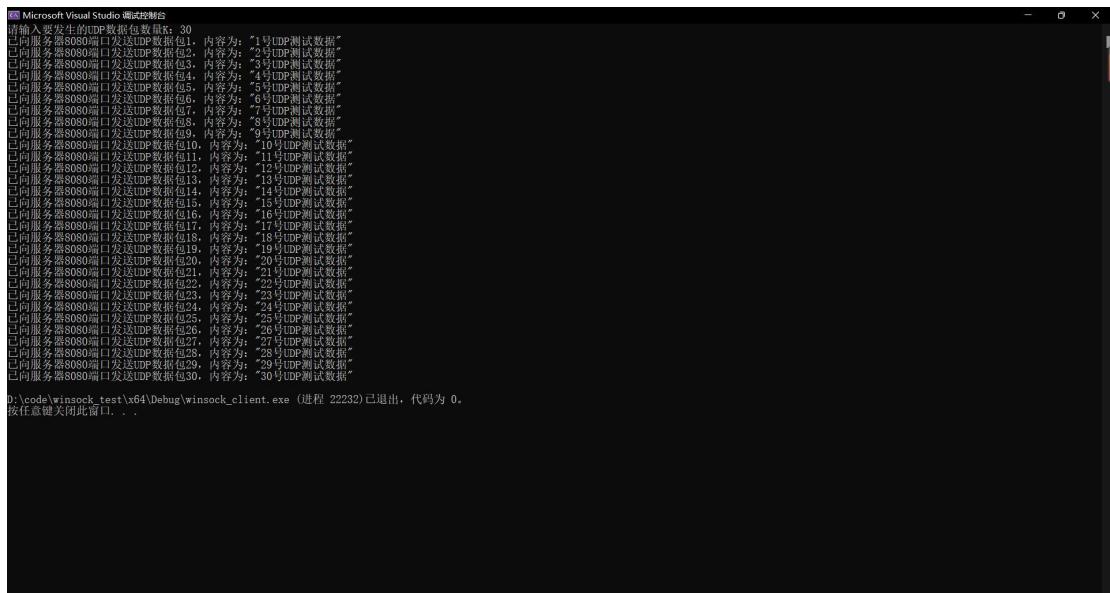
A机器接收情况:

服务器正在端口8080上监听UDP数据包
按下Ctrl+C来关闭服务器程序

可以看到，A 机器接收到了两次运行程序发送的 20 和 40 个数据包，并统计了从 B 机器的地址共收到 60 个数据包。

2、在 A 机器也同时运行客户端软件 2 次，K 值分别为 30, 50

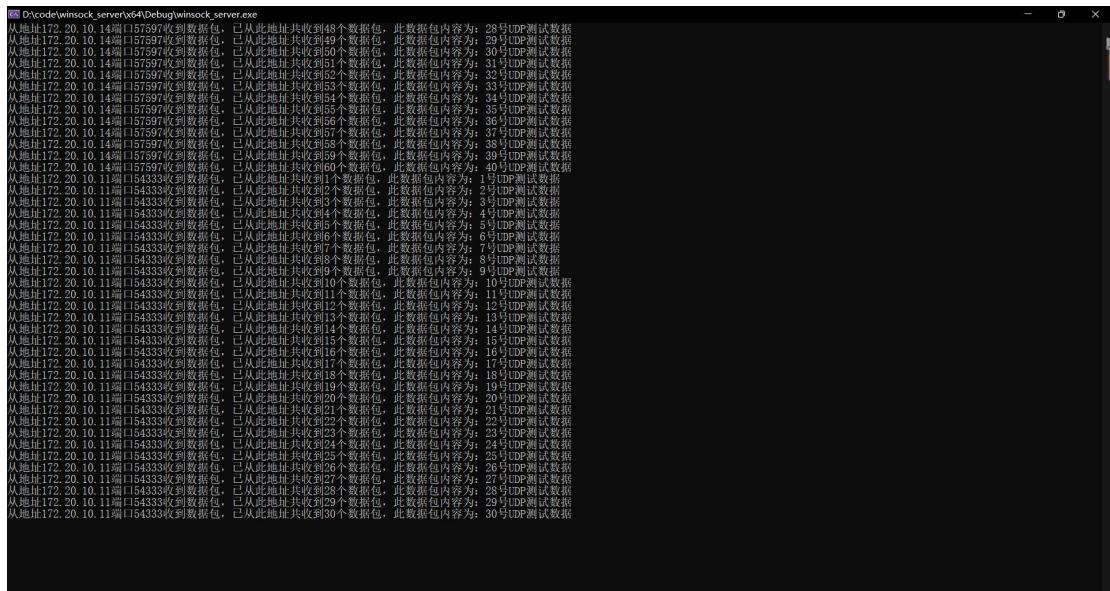
A 机器发送情况（K 值为 30）：



```
Microsoft Visual Studio 调试输出窗口
清输入要发送的UDP数据包数量为30
已向服务端8080端口发送UDP数据包1, 内容为: "1号UDP测试数据"
已向服务端8080端口发送UDP数据包2, 内容为: "2号UDP测试数据"
已向服务端8080端口发送UDP数据包3, 内容为: "3号UDP测试数据"
已向服务端8080端口发送UDP数据包4, 内容为: "4号UDP测试数据"
已向服务端8080端口发送UDP数据包5, 内容为: "5号UDP测试数据"
已向服务端8080端口发送UDP数据包6, 内容为: "6号UDP测试数据"
已向服务端8080端口发送UDP数据包7, 内容为: "7号UDP测试数据"
已向服务端8080端口发送UDP数据包8, 内容为: "8号UDP测试数据"
已向服务端8080端口发送UDP数据包9, 内容为: "9号UDP测试数据"
已向服务端8080端口发送UDP数据包10, 内容为: "10号UDP测试数据"
已向服务端8080端口发送UDP数据包11, 内容为: "11号UDP测试数据"
已向服务端8080端口发送UDP数据包12, 内容为: "12号UDP测试数据"
已向服务端8080端口发送UDP数据包13, 内容为: "13号UDP测试数据"
已向服务端8080端口发送UDP数据包14, 内容为: "14号UDP测试数据"
已向服务端8080端口发送UDP数据包15, 内容为: "15号UDP测试数据"
已向服务端8080端口发送UDP数据包16, 内容为: "16号UDP测试数据"
已向服务端8080端口发送UDP数据包17, 内容为: "17号UDP测试数据"
已向服务端8080端口发送UDP数据包18, 内容为: "18号UDP测试数据"
已向服务端8080端口发送UDP数据包19, 内容为: "19号UDP测试数据"
已向服务端8080端口发送UDP数据包20, 内容为: "20号UDP测试数据"
已向服务端8080端口发送UDP数据包21, 内容为: "21号UDP测试数据"
已向服务端8080端口发送UDP数据包22, 内容为: "22号UDP测试数据"
已向服务端8080端口发送UDP数据包23, 内容为: "23号UDP测试数据"
已向服务端8080端口发送UDP数据包24, 内容为: "24号UDP测试数据"
已向服务端8080端口发送UDP数据包25, 内容为: "25号UDP测试数据"
已向服务端8080端口发送UDP数据包26, 内容为: "26号UDP测试数据"
已向服务端8080端口发送UDP数据包27, 内容为: "27号UDP测试数据"
已向服务端8080端口发送UDP数据包28, 内容为: "28号UDP测试数据"
已向服务端8080端口发送UDP数据包29, 内容为: "29号UDP测试数据"
已向服务端8080端口发送UDP数据包30, 内容为: "30号UDP测试数据"

D:\code\winsock\test\x64\Debug\winsock_client.exe (进程 22232) 已退出, 代码为 0。
按任意键关闭此窗口。.
```

A 机器接收情况：



```
D:\code\winsock\server\x64\Debug\winsock_server.exe
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到48个数据包, 此数据包内容为: 28号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到49个数据包, 此数据包内容为: 29号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到50个数据包, 此数据包内容为: 30号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到51个数据包, 此数据包内容为: 31号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到52个数据包, 此数据包内容为: 32号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到53个数据包, 此数据包内容为: 33号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到54个数据包, 此数据包内容为: 34号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到55个数据包, 此数据包内容为: 35号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到56个数据包, 此数据包内容为: 36号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到57个数据包, 此数据包内容为: 37号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到58个数据包, 此数据包内容为: 38号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到59个数据包, 此数据包内容为: 39号UDP测试数据
从地址172.20.10.14端口15759收到数据包, 已从此地址共收到60个数据包, 此数据包内容为: 40号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到61个数据包, 此数据包内容为: 1号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到62个数据包, 此数据包内容为: 2号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到63个数据包, 此数据包内容为: 3号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到64个数据包, 此数据包内容为: 4号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到65个数据包, 此数据包内容为: 5号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到66个数据包, 此数据包内容为: 6号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到67个数据包, 此数据包内容为: 7号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到68个数据包, 此数据包内容为: 8号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到69个数据包, 此数据包内容为: 9号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到70个数据包, 此数据包内容为: 10号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到11个数据包, 此数据包内容为: 11号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到12个数据包, 此数据包内容为: 12号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到13个数据包, 此数据包内容为: 13号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到14个数据包, 此数据包内容为: 14号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到15个数据包, 此数据包内容为: 15号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到16个数据包, 此数据包内容为: 16号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到17个数据包, 此数据包内容为: 17号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到18个数据包, 此数据包内容为: 18号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到19个数据包, 此数据包内容为: 19号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到20个数据包, 此数据包内容为: 20号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到21个数据包, 此数据包内容为: 21号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到22个数据包, 此数据包内容为: 22号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到23个数据包, 此数据包内容为: 23号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到24个数据包, 此数据包内容为: 24号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到25个数据包, 此数据包内容为: 25号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到26个数据包, 此数据包内容为: 26号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到27个数据包, 此数据包内容为: 27号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到28个数据包, 此数据包内容为: 28号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到29个数据包, 此数据包内容为: 29号UDP测试数据
从地址172.20.10.11端口154333收到数据包, 已从此地址共收到30个数据包, 此数据包内容为: 30号UDP测试数据
```

可以看到，A 机器接收到了来自 A 机器的 30 个数据包，并且统计了来自 A 机器的数据包有 30 个，并没有与 B 机器的地址搞混。

A 机器发送情况 (K 值为 50) :

Microsoft Visual Studio 调试输出	
已向服务器端8080端口发送IP数据包4, 内容为:	"24号UDP测试数据"
已向服务器端8080端口发送IP数据包5, 内容为:	"25号UDP测试数据"
已向服务器端8080端口发送IP数据包6, 内容为:	"26号UDP测试数据"
已向服务器端8080端口发送IP数据包7, 内容为:	"27号UDP测试数据"
已向服务器端8080端口发送IP数据包8, 内容为:	"28号UDP测试数据"
已向服务器端8080端口发送IP数据包9, 内容为:	"29号UDP测试数据"
已向服务器端8080端口发送IP数据包10, 内容为:	"30号UDP测试数据"
已向服务器端8080端口发送IP数据包11, 内容为:	"31号UDP测试数据"
已向服务器端8080端口发送IP数据包12, 内容为:	"32号UDP测试数据"
已向服务器端8080端口发送IP数据包13, 内容为:	"33号UDP测试数据"
已向服务器端8080端口发送IP数据包14, 内容为:	"34号UDP测试数据"
已向服务器端8080端口发送IP数据包15, 内容为:	"35号UDP测试数据"
已向服务器端8080端口发送IP数据包16, 内容为:	"36号UDP测试数据"
已向服务器端8080端口发送IP数据包17, 内容为:	"37号UDP测试数据"
已向服务器端8080端口发送IP数据包18, 内容为:	"38号UDP测试数据"
已向服务器端8080端口发送IP数据包19, 内容为:	"39号UDP测试数据"
已向服务器端8080端口发送IP数据包20, 内容为:	"40号UDP测试数据"
已向服务器端8080端口发送IP数据包21, 内容为:	"41号UDP测试数据"
已向服务器端8080端口发送IP数据包22, 内容为:	"42号UDP测试数据"
已向服务器端8080端口发送IP数据包23, 内容为:	"43号UDP测试数据"
已向服务器端8080端口发送IP数据包24, 内容为:	"44号UDP测试数据"
已向服务器端8080端口发送IP数据包25, 内容为:	"45号UDP测试数据"
已向服务器端8080端口发送IP数据包26, 内容为:	"46号UDP测试数据"
已向服务器端8080端口发送IP数据包27, 内容为:	"47号UDP测试数据"
已向服务器端8080端口发送IP数据包28, 内容为:	"48号UDP测试数据"
已向服务器端8080端口发送IP数据包29, 内容为:	"49号UDP测试数据"
已向服务器端8080端口发送IP数据包30, 内容为:	"50号UDP测试数据"

A机器接收情况:

可以看到，A 机器统计了来自 A 机器地址的数据包总数为 80 个

3、退出服务器端软件

```

Microsoft Visual Studio 调试控制台

从地址172.20.10.11端口16006收到数据包，已从此地址共收到39个数据包，此数据包内容为：9号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到40个数据包，此数据包内容为：10号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到41个数据包，此数据包内容为：11号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到42个数据包，此数据包内容为：12号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到43个数据包，此数据包内容为：13号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到44个数据包，此数据包内容为：14号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到45个数据包，此数据包内容为：15号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到46个数据包，此数据包内容为：16号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到47个数据包，此数据包内容为：17号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到48个数据包，此数据包内容为：18号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到49个数据包，此数据包内容为：19号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到50个数据包，此数据包内容为：20号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到51个数据包，此数据包内容为：21号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到52个数据包，此数据包内容为：22号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到53个数据包，此数据包内容为：23号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到54个数据包，此数据包内容为：24号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到55个数据包，此数据包内容为：25号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到56个数据包，此数据包内容为：26号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到57个数据包，此数据包内容为：27号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到58个数据包，此数据包内容为：28号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到59个数据包，此数据包内容为：29号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到60个数据包，此数据包内容为：30号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到61个数据包，此数据包内容为：31号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到62个数据包，此数据包内容为：32号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到63个数据包，此数据包内容为：33号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到64个数据包，此数据包内容为：34号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到65个数据包，此数据包内容为：35号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到66个数据包，此数据包内容为：36号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到67个数据包，此数据包内容为：37号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到68个数据包，此数据包内容为：38号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到69个数据包，此数据包内容为：39号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到70个数据包，此数据包内容为：40号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到71个数据包，此数据包内容为：41号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到72个数据包，此数据包内容为：42号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到73个数据包，此数据包内容为：43号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到74个数据包，此数据包内容为：44号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到75个数据包，此数据包内容为：45号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到76个数据包，此数据包内容为：46号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到77个数据包，此数据包内容为：47号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到78个数据包，此数据包内容为：48号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到79个数据包，此数据包内容为：49号UDP测试数据
从地址172.20.10.11端口16006收到数据包，已从此地址共收到80个数据包，此数据包内容为：50号UDP测试数据

套接字已关闭
接收失败或套接字被关闭

D:\code\winsock_server\x64\Debug\winsock_server.exe (进程 28460)已退出，代码为 0。
按任意键关闭此窗口。 - -

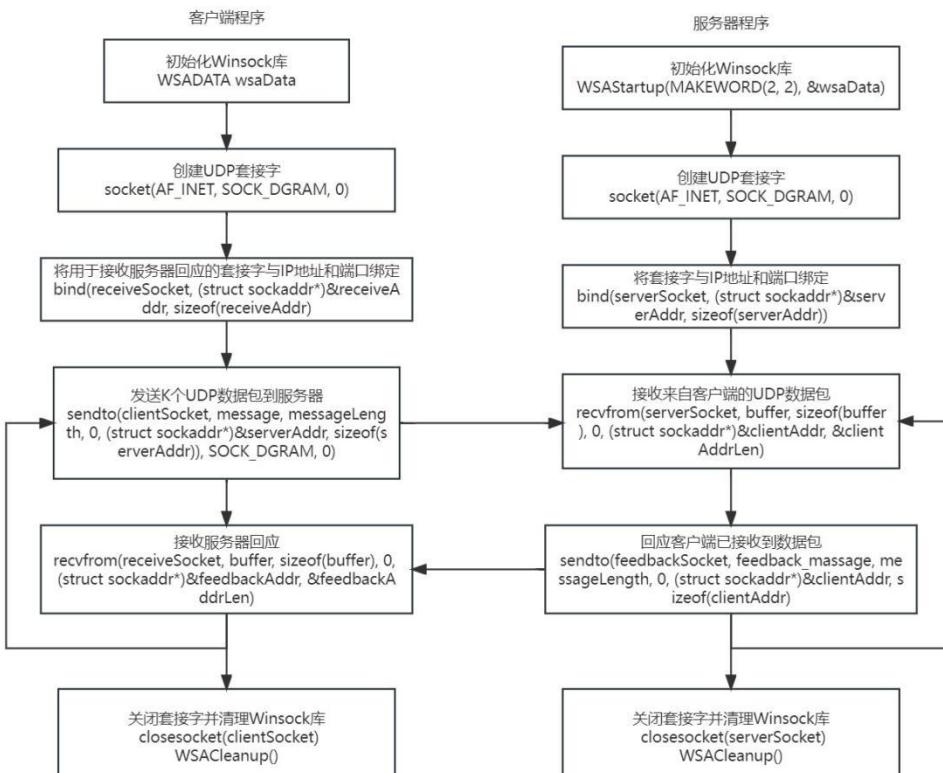
```

可以看到，程序关闭套接字正常退出

计算机网络 实验五

5.1 UDP 通信编程扩展

一、软件设计流程图



二、源代码

1、客户端程序

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS //用于屏蔽 visual studio 对 inet_addr 函数的报错
#define _CRT_SECURE_NO_WARNINGS//用于屏蔽 visual studio 对 sprintf 函数的报错
#include <iostream>
using namespace std;
#include <unordered_map>/哈希表

#include <Winsock2.h>//静态链接库 winsock2.h 是用来编译基于 Winsock 程序的
#pragma comment(lib, "ws2_32.lib")//动态链接库 ws2_32.dll 是运行基于 WinSock 的程序所必须的

int main() {
    int K;
    cout << "请输入要发送的 UDP 数据包数量 K: ";
    cin >> K;

    // 初始化 Winsock 库
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        cerr << "WSAStartup 失败" << endl;
        return 1;
    }

    // 创建 UDP 套接字
    SOCKET clientSocket = socket(AF_INET, SOCK_DGRAM, 0);
    if (clientSocket == INVALID_SOCKET) {
        cerr << "套接字创建失败" << endl;
        WSACleanup();
        return 1;
    }

    SOCKET receiveSocket = socket(AF_INET, SOCK_DGRAM, 0);
    if (receiveSocket == INVALID_SOCKET) {
        cerr << "套接字创建失败" << endl;
        WSACleanup();
        return 1;
    }

    // 将用于接收服务器回应的套接字与 IP 地址和端口绑定
    sockaddr_in receiveAddr;
    receiveAddr.sin_family = AF_INET;
    receiveAddr.sin_addr.s_addr = INADDR_ANY;//设置服务器套接字的本地 IP 地址, INADDR_ANY 是一个常量, 通常用于将服务器绑定到所有可用的网络接口上, 以便服务器可以接受来自任何本地接口的连接请求
    receiveAddr.sin_port = htons(8080); // 监听的 UDP 端口号
    if (bind(receiveSocket, (struct sockaddr*)&receiveAddr, sizeof(receiveAddr)) == SOCKET_ERROR) {
        cerr << "绑定失败" << endl;
        closesocket(receiveSocket);
        WSACleanup();
        return 1;
    }
    cout << "正在端口 8080 上监听 UDP 数据包" << endl;

    // 发送 K 个 UDP 数据包到服务器并接收服务器回应
    sockaddr_in serverAddr;//sockaddr_in 是用于 IPv4 地址的套接字地址结构, 特定于 IPv4 地址族 (AF_INET)
    serverAddr.sin_family = AF_INET;//设置地址族为 AF_INET, 表示使用 IPv4 地址。
    serverAddr.sin_addr.s_addr = inet_addr("172.20.10.11"); // 设置服务器的 IP 地址。inet_addr 函数将字符串形式的 IPv4 地址转换为网络字节序的整数表示形式
    serverAddr.sin_port = htons(8888); // 设置服务器的 UDP 监听端口号。htons 函数将主机字节序 (通常是小端字节序) 的端口号转换为网络字节序 (大端字节序), 以便在网络上传输。
    char message[128];

    sockaddr_in feedbackAddr;//用于接受数据发送方的地址
    int feedbackAddrLen = sizeof(feedbackAddr);//地址结构体大小
    unordered_map<string, int> packetCount;//数据包数量计数器
    char buffer[1024] = "\0";//接收报文的缓冲区

    for (int i = 0; i < K; i++) {
        sprintf(message, "%d 号 UDP 测试数据", i + 1);
        int messageLength = strlen(message);
        int bytesSent = sendto(clientSocket, message, messageLength, 0, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
        if (bytesSent == SOCKET_ERROR) {
            cerr << "发送失败" << endl;
            break;
        }
        cout << "已向服务器 8888 端口发送 UDP 数据包" << i + 1 << ", 内容为: \\" << message << "\\" << endl;
        Sleep(500); // 延迟 0.5 秒

        int bytesReceived = recvfrom(receiveSocket, buffer, sizeof(buffer), 0, (struct sockaddr*)&feedbackAddr,
        &feedbackAddrLen);
        if (bytesReceived == SOCKET_ERROR) {
```

```

        cerr << "接收失败或套接字被关闭\n" << endl;
        break;
    }
    packetCount[inet_ntoa(feedbackAddr.sin_addr)]++;
    cout << "从地址" << inet_ntoa(feedbackAddr.sin_addr) << "端口" << ntohs(feedbackAddr.sin_port) << "收到数据包，";
    cout << "已从此地址共收到" << packetCount[inet_ntoa(feedbackAddr.sin_addr)] << "个数据包，";
    cout << "此数据包内容为：" << buffer << endl;
}

// 关闭套接字并清理 Winsock 库
closesocket(clientSocket);
closesocket(receiveSocket);
WSACleanup();

return 0;
}

```

2、服务器程序

```

#define _WIN32_DEPRECATED_NO_WARNINGS //用于屏蔽 visual studio 对 inet_addr 函数的报错
#define _CRT_SECURE_NO_WARNINGS//用于屏蔽 visual studio 对 sprintf 函数的报错
#include <iostream>
using namespace std;
#include <csignal>//信号处理
#include <unordered_map>//哈希表

#include <Winsock2.h>//静态链接库 winsock2.h 是用来编译基于 Winsock 程序的
#pragma comment(lib, "ws2_32.lib")//动态链接库 ws2_32.dll 是运行基于 WinSock 的程序所必须的

bool continueLoop = true;
SOCKET serverSocket;
void signalHandler(int)
{
    continueLoop = false; // 设置标志以退出循环

    // 关闭套接字并清理 Winsock 库
    cout << "\n套接字已关闭" << endl;
    closesocket(serverSocket);
    WSACleanup();
}

int main()
{
    WSADATA wsaData;
    // 初始化 Winsock 库
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        cerr << "WSAStartup 失败" << endl;
        return 1;
    }

    // 创建 UDP 套接字
    serverSocket = socket(AF_INET, SOCK_DGRAM, 0);
    if (serverSocket == INVALID_SOCKET) {
        cerr << "套接字创建失败" << endl;
        WSACleanup();
        return 1;
    }

    SOCKET feedbackSocket = socket(AF_INET, SOCK_DGRAM, 0);
    if (feedbackSocket == INVALID_SOCKET) {
        cerr << "套接字创建失败" << endl;
        WSACleanup();
        return 1;
    }

    // 将套接字与 IP 地址和端口绑定
    sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;//设置服务器套接字的本地 IP 地址, INADDR_ANY 是一个常量, 通常用于将服务器绑定到所有可用的网络接口上, 以便服务器可以接受来自任何本地接口的连接请求
    serverAddr.sin_port = htons(8888); // 服务器监听的 UDP 端口号
    if (bind(serverSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        cerr << "绑定失败" << endl;
        closesocket(serverSocket);
        WSACleanup();
        return 1;
    }
    cout << "服务器正在端口 8888 上监听 UDP 数据包" << endl;

    // 接收来自客户端的 UDP 数据包
    sockaddr_in clientAddr;//用于接受数据发送方的地址
    int clientAddrLen = sizeof(clientAddr);//地址结构体大小
    unordered_map<string, int> packetCount;//数据包数量计数器

```

```

char buffer[1024] = "\0";//接收报文的缓冲区
char feedback_message[128];
signal(SIGINT, signalHandler);//设置循环的停止条件
printf("按下 Ctrl+C 来关闭服务器程序\n\n");

while (continueLoop) {
    int bytesReceived = recvfrom(serverSocket, buffer, sizeof(buffer), 0, (struct sockaddr*)&clientAddr, &clientAddrLen);
    if (bytesReceived == SOCKET_ERROR) {
        cerr << "接收失败或套接字被关闭\n" << endl;
        break;
    }
    packetCount[inet_ntoa(clientAddr.sin_addr)]++;
    cout << "从地址" << inet_ntoa(clientAddr.sin_addr) << "端口" << ntohs(clientAddr.sin_port) << "收到数据包，";
    cout << "已从此地址共收到" << packetCount[inet_ntoa(clientAddr.sin_addr)] << "个数据包，";
    cout << "此数据包内容为：" << buffer << endl;

    sprintf(feedback_message, "已收到第%d个数据包", packetCount[inet_ntoa(clientAddr.sin_addr)]);
    int messageLength = strlen(feedback_message);
    clientAddr.sin_port = htons(8080);
    int bytesSent = sendto(feedbackSocket, feedback_message, messageLength, 0, (struct sockaddr*)&clientAddr,
    sizeof(clientAddr));
    if (bytesSent == SOCKET_ERROR) {
        cerr << "发送失败" << endl;
        break;
    }
    cout << "已向此客户端 8080 端口回应收到第" << packetCount[inet_ntoa(clientAddr.sin_addr)] << "个数据包" << endl;
}

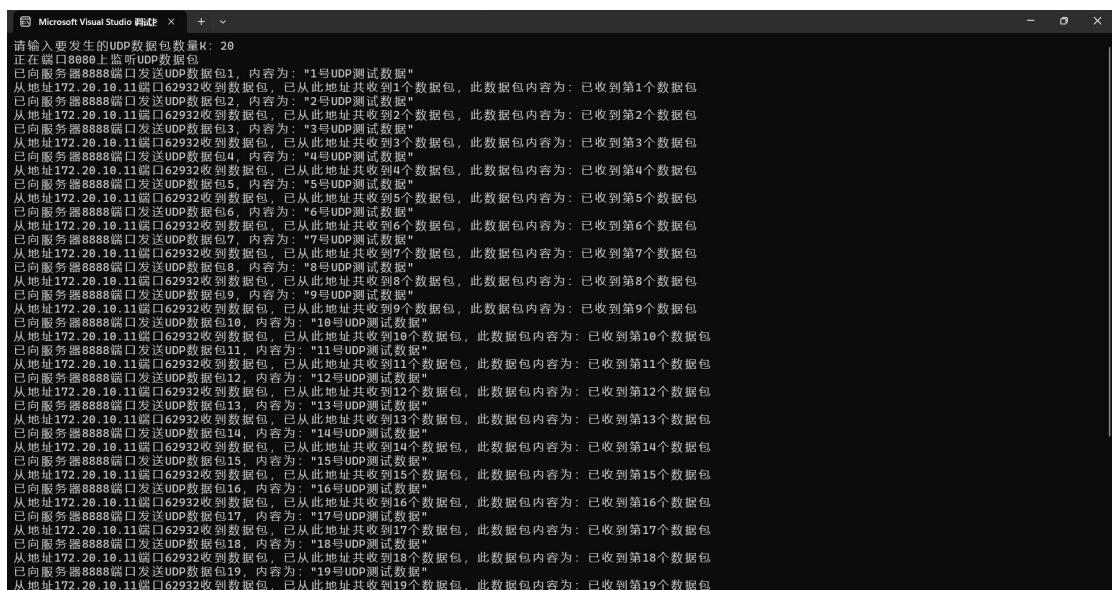
closesocket(feedbackSocket);
return 0;
}

```

三、测试运行截图

1、在客户端 B 机器的命令行运行客户端软件 2 次，K 值分别为 20，40

B 机器发送情况（K 值为 20）：



```

Microsoft Visual Studio 调试 x + v

从地址172.20.18.11端口62932收到数据包，已从此地址共收到4个数据包，此数据包内容为：已收到第4个数据包
已向服务器8888端口发送UDP数据包5，内容为：“5号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到5个数据包，此数据包内容为：已收到第5个数据包
已向服务器8888端口发送UDP数据包6，内容为：“6号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到6个数据包，此数据包内容为：已收到第6个数据包
已向服务器8888端口发送UDP数据包7，内容为：“7号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到7个数据包，此数据包内容为：已收到第7个数据包
已向服务器8888端口发送UDP数据包8，内容为：“8号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到8个数据包，此数据包内容为：已收到第8个数据包
已向服务器8888端口发送UDP数据包9，内容为：“9号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到9个数据包，此数据包内容为：已收到第9个数据包
已向服务器8888端口发送UDP数据包10，内容为：“10号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到10个数据包，此数据包内容为：已收到第10个数据包
已向服务器8888端口发送UDP数据包11，内容为：“11号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到11个数据包，此数据包内容为：已收到第11个数据包
已向服务器8888端口发送UDP数据包12，内容为：“12号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到12个数据包，此数据包内容为：已收到第12个数据包
已向服务器8888端口发送UDP数据包13，内容为：“13号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到13个数据包，此数据包内容为：已收到第13个数据包
已向服务器8888端口发送UDP数据包14，内容为：“14号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到14个数据包，此数据包内容为：已收到第14个数据包
已向服务器8888端口发送UDP数据包15，内容为：“15号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到15个数据包，此数据包内容为：已收到第15个数据包
已向服务器8888端口发送UDP数据包16，内容为：“16号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到16个数据包，此数据包内容为：已收到第16个数据包
已向服务器8888端口发送UDP数据包17，内容为：“17号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到17个数据包，此数据包内容为：已收到第17个数据包
已向服务器8888端口发送UDP数据包18，内容为：“18号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到18个数据包，此数据包内容为：已收到第18个数据包
已向服务器8888端口发送UDP数据包19，内容为：“19号UDP测试数据”
从地址172.20.18.11端口62932收到数据包，已从此地址共收到19个数据包，此数据包内容为：已收到第19个数据包
从地址172.20.18.11端口62932收到数据包，已从此地址共收到20个数据包，此数据包内容为：已收到第20个数据包

C:\Users\28222\Desktop\code\VS\test\test\x64\Debug\test.exe (进程 19052)已退出，代码为 0。
按任意键关闭此窗口。 . .

```

可以看到，B 机器发送了 20 个数据包，并且每个数据包都接收到了 A 机器的回应

A 机器接收情况：

```

D:\code\winsock server\x64\Debug\winsock server.exe
服务器正在端口8888上监听UDP数据包
按下Ctrl+C来关闭服务 帮助程序

从地址172.20.10.14端口1562300收到数据包，已从此地址共收到1个数据包，此数据包内容为：1号UDP测试数据
已向客户机端8080端口回应收到第1个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到2个数据包，此数据包内容为：2号UDP测试数据
已向客户机端8080端口回应收到第2个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到3个数据包，此数据包内容为：3号UDP测试数据
已向客户机端8080端口回应收到第3个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到4个数据包，此数据包内容为：4号UDP测试数据
已向客户机端8080端口回应收到第4个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到5个数据包，此数据包内容为：5号UDP测试数据
已向客户机端8080端口回应收到第5个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到6个数据包，此数据包内容为：6号UDP测试数据
已向客户机端8080端口回应收到第6个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到7个数据包，此数据包内容为：7号UDP测试数据
已向客户机端8080端口回应收到第7个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到8个数据包，此数据包内容为：8号UDP测试数据
已向客户机端8080端口回应收到第8个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到9个数据包，此数据包内容为：9号UDP测试数据
已向客户机端8080端口回应收到第9个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到10个数据包，此数据包内容为：10号UDP测试数据
已向客户机端8080端口回应收到第10个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到11个数据包，此数据包内容为：11号UDP测试数据
已向客户机端8080端口回应收到第11个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到12个数据包，此数据包内容为：12号UDP测试数据
已向客户机端8080端口回应收到第12个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到13个数据包，此数据包内容为：13号UDP测试数据
已向客户机端8080端口回应收到第13个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到14个数据包，此数据包内容为：14号UDP测试数据
已向客户机端8080端口回应收到第14个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到15个数据包，此数据包内容为：15号UDP测试数据
已向客户机端8080端口回应收到第15个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到16个数据包，此数据包内容为：16号UDP测试数据
已向客户机端8080端口回应收到第16个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到17个数据包，此数据包内容为：17号UDP测试数据
已向客户机端8080端口回应收到第17个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到18个数据包，此数据包内容为：18号UDP测试数据
已向客户机端8080端口回应收到第18个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到19个数据包，此数据包内容为：19号UDP测试数据
已向客户机端8080端口回应收到第19个数据包
从地址172.20.10.14端口1562300收到数据包，已从此地址共收到20个数据包，此数据包内容为：20号UDP测试数据
已向客户机端8080端口回应收到第20个数据包

```

可以看到，A 机器接收到了 20 个数据包，并且每个数据包都进行了回应

B 机器发送情况 (K 值为 40) :

```
Microsoft Visual Studio [调试] + 

请输入要发送的UDP数据包数量K: 48
正在端口8888上监听UDP数据包。
已向服务器8888端口发送UDP数据包1, 内容为: "1号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到1个数据包, 此数据包内容为: 已收到第21个数据包
已向服务器8888端口发送UDP数据包2, 内容为: "2号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到2个数据包, 此数据包内容为: 已收到第22个数据包
已向服务器8888端口发送UDP数据包3, 内容为: "3号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到3个数据包, 此数据包内容为: 已收到第23个数据包
已向服务器8888端口发送UDP数据包4, 内容为: "4号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到4个数据包, 此数据包内容为: 已收到第24个数据包
已向服务器8888端口发送UDP数据包5, 内容为: "5号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到5个数据包, 此数据包内容为: 已收到第25个数据包
已向服务器8888端口发送UDP数据包6, 内容为: "6号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到6个数据包, 此数据包内容为: 已收到第26个数据包
已向服务器8888端口发送UDP数据包7, 内容为: "7号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到7个数据包, 此数据包内容为: 已收到第27个数据包
已向服务器8888端口发送UDP数据包8, 内容为: "8号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到8个数据包, 此数据包内容为: 已收到第28个数据包
已向服务器8888端口发送UDP数据包9, 内容为: "9号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到9个数据包, 此数据包内容为: 已收到第29个数据包
已向服务器8888端口发送UDP数据包10, 内容为: "10号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到10个数据包, 此数据包内容为: 已收到第30个数据包
已向服务器8888端口发送UDP数据包11, 内容为: "11号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到11个数据包, 此数据包内容为: 已收到第31个数据包
已向服务器8888端口发送UDP数据包12, 内容为: "12号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到12个数据包, 此数据包内容为: 已收到第32个数据包
已向服务器8888端口发送UDP数据包13, 内容为: "13号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到13个数据包, 此数据包内容为: 已收到第33个数据包
已向服务器8888端口发送UDP数据包14, 内容为: "14号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到14个数据包, 此数据包内容为: 已收到第34个数据包
已向服务器8888端口发送UDP数据包15, 内容为: "15号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到15个数据包, 此数据包内容为: 已收到第35个数据包
已向服务器8888端口发送UDP数据包16, 内容为: "16号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到16个数据包, 此数据包内容为: 已收到第36个数据包
已向服务器8888端口发送UDP数据包17, 内容为: "17号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到17个数据包, 此数据包内容为: 已收到第37个数据包
已向服务器8888端口发送UDP数据包18, 内容为: "18号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到18个数据包, 此数据包内容为: 已收到第38个数据包
已向服务器8888端口发送UDP数据包19, 内容为: "19号 UDP 测试数据"
从地址172_26_19_11端口62495收到数据包。已从此地址共收到19个数据包, 此数据包内容为: 已收到第39个数据包
```

可以看到，A 机器回应共接收到 60 个数据包，数字正确

A 机器接收情况:

可以看到，A 机器继续接收到了 40 个数据包，并且都进行了回应，统计一共从 B 机器地址接收到了 60 个数据包

2、在 A 机器也同时运行客户端软件 2 次，K 值分别为 30, 50

A 机器发送情况（K 值为 30）：

```
Microsoft Visual Studio 调试控制台
请输入要发送的UDP数据包数量: K: 30
正在端口18000上监听UDP数据包
从地址172.20.10.11端口162495收到数据包1, 内容为: "1号UDP测试数据"
已向服务8888端口发送UDP数据包1, 内容为: "1号UDP测试数据"
从地址172.20.10.11端口162495收到数据包2, 已从此地址共收到2个数据包, 此数据包内容为: 已收到第1个数据包
已向服务8888端口发送UDP数据包2, 内容为: "2号UDP测试数据"
从地址172.20.10.11端口162495收到数据包3, 已从此地址共收到3个数据包, 此数据包内容为: 已收到第2个数据包
已向服务8888端口发送UDP数据包3, 内容为: "3号UDP测试数据"
从地址172.20.10.11端口162495收到数据包4, 已从此地址共收到4个数据包, 此数据包内容为: 已收到第3个数据包
已向服务8888端口发送UDP数据包4, 内容为: "4号UDP测试数据"
从地址172.20.10.11端口162495收到数据包5, 已从此地址共收到5个数据包, 此数据包内容为: 已收到第4个数据包
已向服务8888端口发送UDP数据包5, 内容为: "5号UDP测试数据"
从地址172.20.10.11端口162495收到数据包6, 已从此地址共收到6个数据包, 此数据包内容为: 已收到第5个数据包
已向服务8888端口发送UDP数据包6, 内容为: "6号UDP测试数据"
从地址172.20.10.11端口162495收到数据包7, 已从此地址共收到7个数据包, 此数据包内容为: 已收到第6个数据包
已向服务8888端口发送UDP数据包7, 内容为: "7号UDP测试数据"
从地址172.20.10.11端口162495收到数据包8, 已从此地址共收到8个数据包, 此数据包内容为: 已收到第7个数据包
已向服务8888端口发送UDP数据包8, 内容为: "8号UDP测试数据"
从地址172.20.10.11端口162495收到数据包9, 已从此地址共收到9个数据包, 此数据包内容为: 已收到第8个数据包
已向服务8888端口发送UDP数据包9, 内容为: "9号UDP测试数据"
从地址172.20.10.11端口162495收到数据包10, 已从此地址共收到10个数据包, 此数据包内容为: 已收到第9个数据包
已向服务8888端口发送UDP数据包10, 内容为: "10号UDP测试数据"
从地址172.20.10.11端口162495收到数据包11, 内容为: "11号UDP测试数据"
从地址172.20.10.11端口162495收到数据包12, 内容为: "12号UDP测试数据"
从地址172.20.10.11端口162495收到数据包13, 内容为: "13号UDP测试数据"
从地址172.20.10.11端口162495收到数据包14, 内容为: "14号UDP测试数据"
从地址172.20.10.11端口162495收到数据包15, 已从此地址共收到14个数据包, 此数据包内容为: 已收到第14个数据包
已向服务8888端口发送UDP数据包15, 内容为: "15号UDP测试数据"
从地址172.20.10.11端口162495收到数据包16, 内容为: "16号UDP测试数据"
从地址172.20.10.11端口162495收到数据包17, 内容为: "17号UDP测试数据"
从地址172.20.10.11端口162495收到数据包18, 已从此地址共收到16个数据包, 此数据包内容为: 已收到第16个数据包
已向服务8888端口发送UDP数据包18, 内容为: "18号UDP测试数据"
从地址172.20.10.11端口162495收到数据包19, 已从此地址共收到17个数据包, 此数据包内容为: 已收到第17个数据包
已向服务8888端口发送UDP数据包19, 内容为: "19号UDP测试数据"
从地址172.20.10.11端口162495收到数据包20, 已从此地址共收到18个数据包, 此数据包内容为: 已收到第18个数据包
已向服务8888端口发送UDP数据包20, 内容为: "20号UDP测试数据"
从地址172.20.10.11端口162495收到数据包21, 已从此地址共收到19个数据包, 此数据包内容为: 已收到第19个数据包
已向服务8888端口发送UDP数据包21, 内容为: "21号UDP测试数据"
从地址172.20.10.11端口162495收到数据包22, 已从此地址共收到20个数据包, 此数据包内容为: 已收到第20个数据包
已向服务8888端口发送UDP数据包22, 内容为: "22号UDP测试数据"
从地址172.20.10.11端口162495收到数据包23, 已从此地址共收到21个数据包, 此数据包内容为: 已收到第21个数据包
已向服务8888端口发送UDP数据包23, 内容为: "23号UDP测试数据"
从地址172.20.10.11端口162495收到数据包24, 已从此地址共收到22个数据包, 此数据包内容为: 已收到第22个数据包
已向服务8888端口发送UDP数据包24, 内容为: "24号UDP测试数据"
```

```
Microsoft Visual Studio 调试控制台
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到10个数据包, 此数据包内容为: 已收到第10个数据包
已向服务8888端口发送UDP数据包11, 内容为: "11号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到11个数据包, 此数据包内容为: 已收到第11个数据包
已向服务8888端口发送UDP数据包12, 内容为: "12号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到12个数据包, 此数据包内容为: 已收到第12个数据包
已向服务8888端口发送UDP数据包13, 内容为: "13号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到13个数据包, 此数据包内容为: 已收到第13个数据包
已向服务8888端口发送UDP数据包14, 内容为: "14号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到14个数据包, 此数据包内容为: 已收到第14个数据包
已向服务8888端口发送UDP数据包15, 内容为: "15号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到15个数据包, 此数据包内容为: 已收到第15个数据包
已向服务8888端口发送UDP数据包16, 内容为: "16号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到16个数据包, 此数据包内容为: 已收到第16个数据包
已向服务8888端口发送UDP数据包17, 内容为: "17号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到17个数据包, 此数据包内容为: 已收到第17个数据包
已向服务8888端口发送UDP数据包18, 内容为: "18号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到18个数据包, 此数据包内容为: 已收到第18个数据包
已向服务8888端口发送UDP数据包19, 内容为: "19号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到19个数据包, 此数据包内容为: 已收到第19个数据包
已向服务8888端口发送UDP数据包20, 内容为: "20号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到20个数据包, 此数据包内容为: 已收到第20个数据包
已向服务8888端口发送UDP数据包21, 内容为: "21号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到21个数据包, 此数据包内容为: 已收到第21个数据包
已向服务8888端口发送UDP数据包22, 内容为: "22号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到22个数据包, 此数据包内容为: 已收到第22个数据包
已向服务8888端口发送UDP数据包23, 内容为: "23号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到23个数据包, 此数据包内容为: 已收到第23个数据包
已向服务8888端口发送UDP数据包24, 内容为: "24号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到24个数据包, 此数据包内容为: 已收到第24个数据包
已向服务8888端口发送UDP数据包25, 内容为: "25号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到25个数据包, 此数据包内容为: 已收到第25个数据包
已向服务8888端口发送UDP数据包26, 内容为: "26号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到26个数据包, 此数据包内容为: 已收到第26个数据包
已向服务8888端口发送UDP数据包27, 内容为: "27号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到27个数据包, 此数据包内容为: 已收到第27个数据包
已向服务8888端口发送UDP数据包28, 内容为: "28号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到28个数据包, 此数据包内容为: 已收到第28个数据包
已向服务8888端口发送UDP数据包29, 内容为: "29号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到29个数据包, 此数据包内容为: 已收到第29个数据包
已向服务8888端口发送UDP数据包30, 内容为: "30号UDP测试数据"
从地址172.20.10.11端口162495收到数据包, 已从此地址共收到30个数据包, 此数据包内容为: 已收到第30个数据包
```

D:\code\winsock_test\win32\Debug\winsock_client.exe (进程 18300) 已退出, 代码为 0。

按键盘关闭此窗口。

可以看到, A 机器发送了 30 个数据包, 并且每个数据包都接收到了 A 机器的回应。A 机器回应共接收到了 30 个数据包, 数字正确, 没有与 B 机器地址搞混。

A 机器接收情况:

可以看到，A 机器接收到了 30 个数据包，并且每个数据包都进行了回应。

A 机器发送情况 (K 值为 50) :

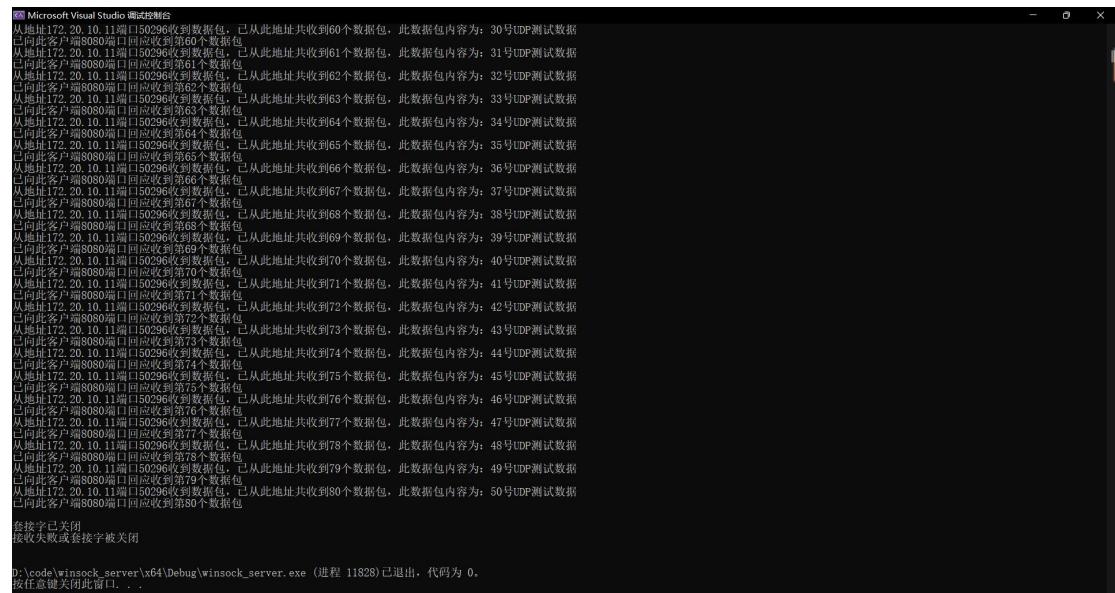
Microsoft Visual Studio 调试输出
捕获到本地发出的 UDP 数据包信息数:50
正在扫描 8050 上所有端口数据包
向服务器8888#1口发送IDP数据包1, 内容为: "1号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了1个数据包, 此数据包内容为: 已收到第31个数据包
向服务器8888#1口发送IDP数据包2, 内容为: "2号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了1个数据包, 此数据包内容为: 已收到第32个数据包
向服务器8888#1口发送IDP数据包3, 内容为: "3号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了3个数据包, 此数据包内容为: 已收到第33个数据包
向服务器8888#1口发送IDP数据包4, 内容为: "4号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了4个数据包, 此数据包内容为: 已收到第34个数据包
向服务器8888#1口发送IDP数据包5, 内容为: "5号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了5个数据包, 此数据包内容为: 已收到第35个数据包
向服务器8888#1口发送IDP数据包6, 内容为: "6号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了6个数据包, 此数据包内容为: 已收到第36个数据包
向服务器8888#1口发送IDP数据包7, 内容为: "7号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了7个数据包, 此数据包内容为: 已收到第37个数据包
向服务器8888#1口发送IDP数据包8, 内容为: "8号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了8个数据包, 此数据包内容为: 已收到第38个数据包
向服务器8888#1口发送IDP数据包9, 内容为: "9号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了9个数据包, 此数据包内容为: 已收到第39个数据包
向服务器8888#1口发送IDP数据包10, 内容为: "10号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了10个数据包, 此数据包内容为: 已收到第40个数据包
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了11个数据包, 此数据包内容为: 已收到第41个数据包
向服务器8888#1口发送IDP数据包12, 内容为: "12号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了12个数据包, 此数据包内容为: 已收到第42个数据包
向服务器8888#1口发送IDP数据包13, 内容为: "13号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了13个数据包, 此数据包内容为: 已收到第43个数据包
向服务器8888#1口发送IDP数据包14, 内容为: "14号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了14个数据包, 此数据包内容为: 已收到第44个数据包
向服务器8888#1口发送IDP数据包15, 内容为: "15号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了15个数据包, 此数据包内容为: 已收到第45个数据包
向服务器8888#1口发送IDP数据包16, 内容为: "16号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了16个数据包, 此数据包内容为: 已收到第46个数据包
向服务器8888#1口发送IDP数据包17, 内容为: "17号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了17个数据包, 此数据包内容为: 已收到第47个数据包
向服务器8888#1口发送IDP数据包18, 内容为: "18号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了18个数据包, 此数据包内容为: 已收到第48个数据包
向服务器8888#1口发送IDP数据包19, 内容为: "19号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了19个数据包, 此数据包内容为: 已收到第49个数据包
向服务器8888#1口发送IDP数据包20, 内容为: "20号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了20个数据包, 此数据包内容为: 已收到第50个数据包
向服务器8888#1口发送IDP数据包21, 内容为: "21号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了21个数据包, 此数据包内容为: 已收到第51个数据包
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了22个数据包, 此数据包内容为: 已收到第52个数据包
向服务器8888#1口发送IDP数据包23, 内容为: "23号IDP测试数据"
从地址172.20.10.11端口162495收到数据包, 以此地址共收到了23个数据包, 此数据包内容为: 已收到第53个数据包
向服务器8888#1口发送IDP数据包24, 内容为: "24号IDP测试数据"

可以看到，A 机器发送了 50 个数据包，并且都收到了回应，A 机器回应共收到 80 个数据包，数字正确。

A 机器接收情况:

可以看到，A 机器收到了 50 个数据包并且都进行了回应，总计从 A 机器的地址收到了 80 个数据包。

3、退出服务器端软件



可以看到，程序关闭套接字正常退出