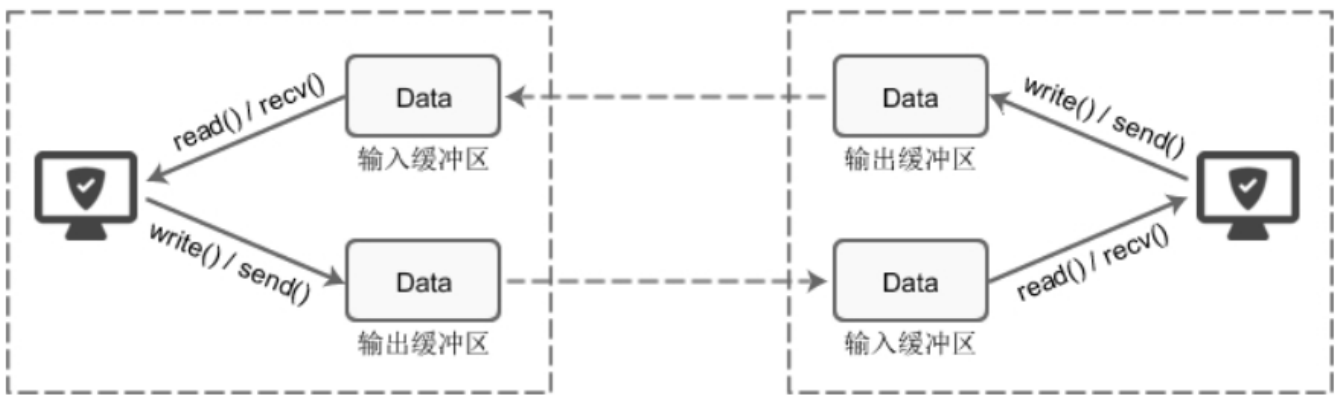
20220417-socket

1.过程描述

1.1 Socket

send()和recv()函数并不立即向网络中传输数据，而是先将数据写入缓冲区中，再由TCP协议将数据从缓冲区发送到目标机器。一旦将数据写入到缓冲区，函数就可以成功返回（返回发送或接收的字节数）。



图：TCP套接字的I/O缓冲区示意图

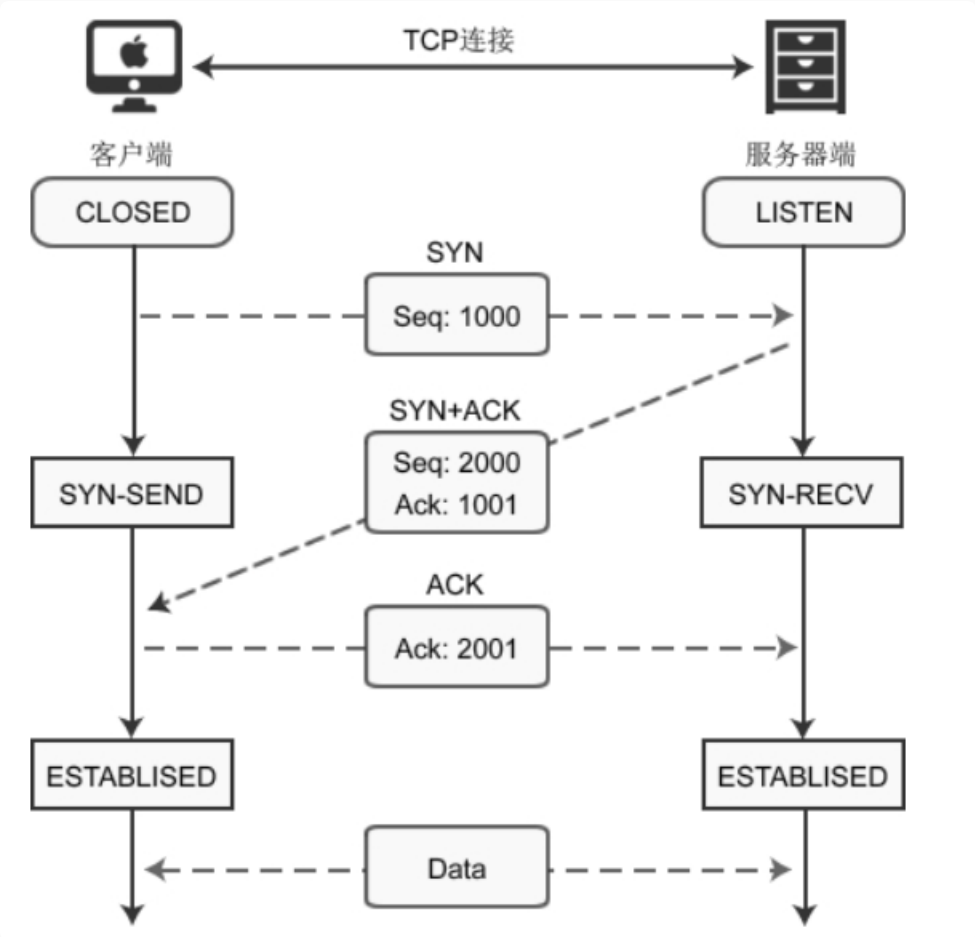
阻塞模式

- 所谓阻塞就是上一步动作没有完成下一步动作将暂停
- 当缓冲区的可用空间长度小于要发送的数据，那么send会被阻塞
- 当TCP协议正在向网络发送数据，则输出缓冲区会被锁定，send也会被阻塞
- recv会先检查缓冲区会有数据，如果有就读取，否则函数会被阻塞，直到网络上有数据到来

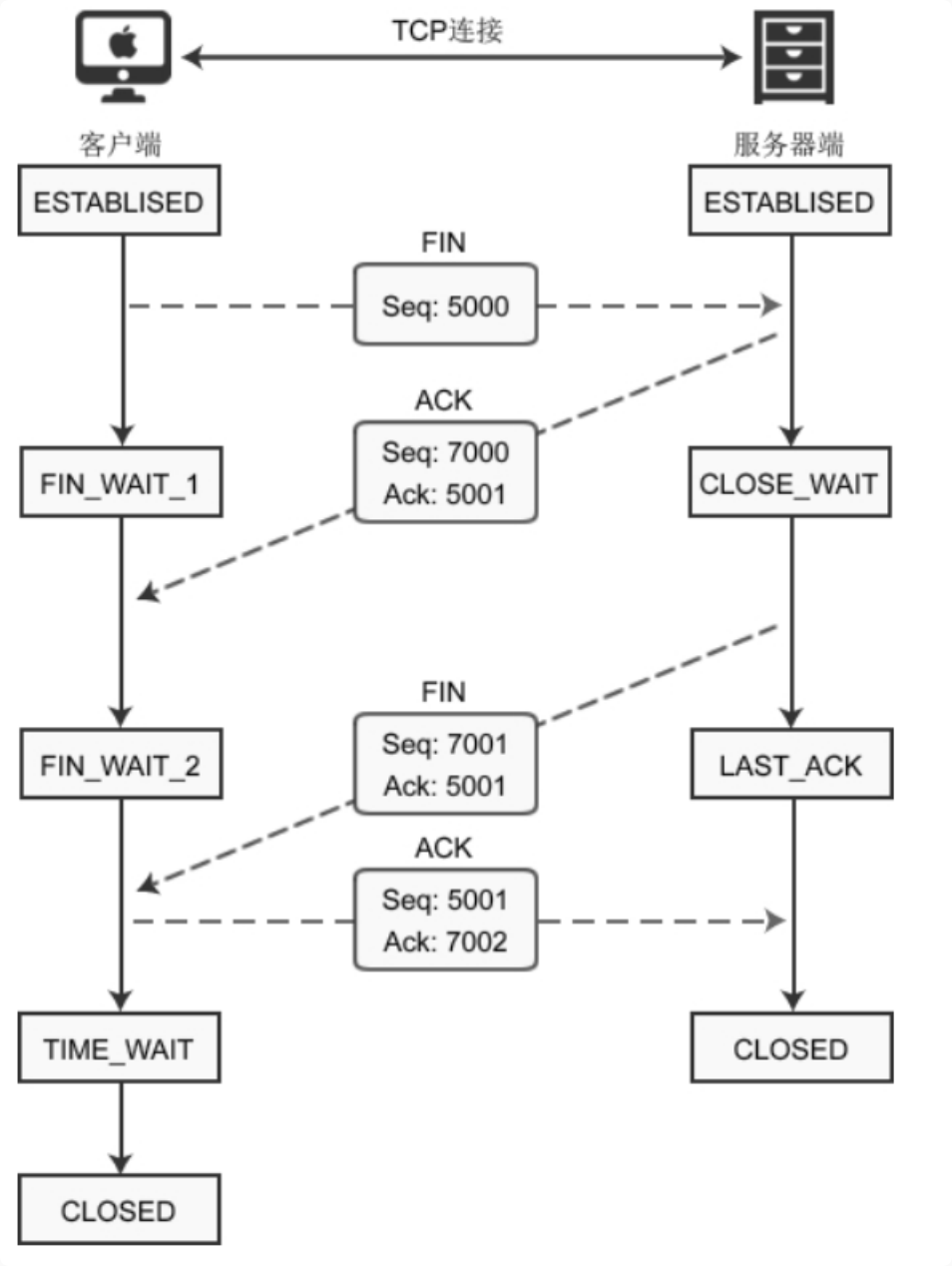
粘包问题

- 粘包问题及客户端发送的多个数据包被当作一个数据包接收
- 数据的接收和发送是无关的，即recv()和send()的执行次数可能不同
- send执行三次，每次发送相同的字符串，目标机器可能分三次、两次、一次接收

三次握手（建立连接时



四次握手（断开连接时


TCP连接
客户端
服务器端
ESTABLISED
ESTABLISED
FIN
Seq: 5000
ACK
Seq: 7000
Ack: 5001
FIN_WAIT_1
CLOSE_WAIT
FIN
Seq: 7001
Ack: 5001
FIN_WAIT_2
LAST_ACK
ACK
Seq: 5001
Ack: 7002
TIME_WAIT
CLOSED
CLOSED

优雅的断开

shutdown函数 C++ 复制代码

```
int shutdown(SOCKET s, int howto);
howto有以下取值：
1）SD_RECEIVE:关闭接收操作，也就是断开输入流；
2）SD_SEND:关闭发送操作，也就断开输出流；
3）SD_BOTH:同时关闭接收和发送操作。

closesocket用来关闭套接字，而shutdown不管调用多少次，套接字依然存在；
关闭输出流时，closesocket会立即向网络发送FIN包，不管输出缓冲区中是否还有数据，而shutdown会等输出缓冲区中的数据传输完毕再发送FIN包，这意味着调用closesocket会丢失输出缓冲区中的数据，而shutdown不会
```

文件传输

ServerFile.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>
#pragma comment (lib,"ws2_32.lib")
#define BUF_SIZE 1024
int main()
{
    //先检查文件是否存在
    const char* filename = "D:\\send.avi";
    FILE* fp = fopen(filename, "rb");
    if (fp == NULL)
    {
        printf("Cannot open file,press any key to exit!\n");
        system("pause");
        exit(0);
    }

    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);

    SOCKET serSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    SOCKADDR_IN serverAddr;
    int len = sizeof(SOCKADDR);
    memset(&serverAddr, 0, len);
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(1234);
    serverAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
    bind(serSock, (LPSOCKADDR)&serverAddr, len);
    listen(serSock, 20);

    SOCKADDR_IN clientAddr;
    SOCKET clientSock = accept(serSock, (LPSOCKADDR)&clientAddr, &len);
    char sendBuf[BUF_SIZE] = { 0 };
    int nCount;
    while( (nCount = fread(sendBuf, 1, BUF_SIZE, fp)) > 0)
    {
        send(clientSock, sendBuf, nCount, 0);
    }
    shutdown(clientSock, SD_SEND);
    recv(clientSock, sendBuf, BUF_SIZE, 0);//recv() 并没有接收到 client 端的数据，当 client 端调用 closesocket() 后，server 端会收到FIN包，recv() 就会返回，后面的代码继续执行。
    fclose(fp);
    closesocket(clientSock);
```

```
44        closesocket(serSock);
45        WSACleanup();
46        system("pause");
47        return 0;
48    }
49
```

ClientFile.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>
#pragma comment (lib,"ws2_32.lib")
#define BUF_SIZE 100
int main()
{
    //先输入文件名，看文件是否能创建成功
    char filename[100] = { 0 };
    printf("Input filename to save:");
    gets_s(filename);
    FILE* fp = fopen(filename, "wb");
    if (fp == NULL)
    {
        printf("Cannot open file, press any key to exit!\n");
        system("pause");
        exit(0);
    }


    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);

    SOCKET ClientSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    SOCKADDR_IN serverAddr;
    int len = sizeof(SOCKADDR);
    memset(&serverAddr, 0, len);
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(1234);//将short类型数据从主机字节序转化为网络字节序
    serverAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");//将点分十进制字符串转化为4字节整数，还可以检测无效IP地址

    connect(ClientSock, (LPSOCKADDR)&serverAddr, len);

    //循环接收数据，直到文件传输完毕
    char recvBuf[BUF_SIZE] = { 0 };
    int nCount;
    while ((nCount = recv(ClientSock, recvBuf, BUF_SIZE, 0)) > 0)
    {
        fwrite(recvBuf, nCount, 1, fp);
    }
    puts("File transfer success");

```

```
        fclose(fp);
        closesocket(ClientSock);
        WSACleanup();
        system("pause");
        return 0;
    }
```

网络编程知识补充

- TCP中，server创建了一个socket用于监听，然后每次接收了客户端的请求，则创建一个新的socket用于通信
- UDP的服务器和客户端无需经过连接过程，不必调用listen和accept函数。不管服务器还是客户端都只需要一个套接字
- 创建好TCP套接字后，传输数据时无需再添加地址信息；UDP套接字不会保持连接状态，每次传输数据都要添加目标地址信息

域名转IP

DomainIP.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>
#pragma comment(lib, "ws2_32.lib")

int main()
{
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);

    struct hostent* host = gethostbyname("www.baidu.com");
    if (!host)
    {
        puts("get IP address error!");
        system("pause");
        exit(0);
    }

    for (int i = 0; host->h_aliases[i];i++)
    {
        printf("Aliases %d:%s\n", i + 1, host->h_aliases[i]);
    }

    printf("Address type:%s\n", (host->h_addrtype == AF_INET) ? "AF_INET" : "AF_INET6");

    for (int i = 0; host->h_addr_list[i]; i++)
    {
        printf("IP addr %d:%s\n", i + 1, inet_ntoa(*(struct in_addr*)host->h_addr_list[i]));
    }
    system("pause");
    return 0;

}
```

UDP编程

C++ 复制代码

```cpp
#include <stdio.h>
#include <winsock2.h>
#pragma comment (lib, "ws2_32.lib")  //加载 ws2_32.dll
#define BUF_SIZE 100
int main() {
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);
    //创建套接字
    SOCKET sock = socket(AF_INET, SOCK_DGRAM, 0);
    //绑定套接字
    sockaddr_in servAddr;
    memset(&servAddr, 0, sizeof(servAddr));  //每个字节都用0填充
    servAddr.sin_family = PF_INET;  //使用IPv4地址
    servAddr.sin_addr.s_addr = htonl(INADDR_ANY); //自动获取IP地址
    servAddr.sin_port = htons(1234);  //端口
    bind(sock, (SOCKADDR*)&servAddr, sizeof(SOCKADDR));
    //接收客户端请求
    SOCKADDR clntAddr;  //客户端地址信息
    int nSize = sizeof(SOCKADDR);
    char buffer[BUF_SIZE];  //缓冲区
    while (1) {
        int strlen = recvfrom(sock, buffer, BUF_SIZE, 0, &clntAddr,
&nSize);
        sendto(sock, buffer, strlen, 0, &clntAddr,nSize);
    }
    closesocket(sock);
    WSACleanup();
    return 0;
}
```

ClientUDP.cpp C++ 复制代码

```cpp
#include <stdio.h>
#include <WinSock2.h>
#pragma comment(lib, "ws2_32.lib")  //加载 ws2_32.dll
#define BUF_SIZE 100
int main() {
    //初始化DLL
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);
    //创建套接字
    SOCKET sock = socket(PF_INET, SOCK_DGRAM, 0);
    //服务器地址信息
    sockaddr_in servAddr;
    memset(&servAddr, 0, sizeof(servAddr));  //每个字节都用0填充
    servAddr.sin_family = PF_INET;
    servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servAddr.sin_port = htons(1234);
    //不断获取用户输入并发送给服务器，然后接受服务器数据
    sockaddr fromAddr;
    int addrLen = sizeof(fromAddr);
    while (1) {
        char buffer[BUF_SIZE] = { 0 };
        printf("Input a string: ");
        gets_s(buffer);
        sendto(sock, buffer, strlen(buffer), 0, (struct sockaddr*)&servAddr, sizeof(servAddr));
        int strLen = recvfrom(sock, buffer, BUF_SIZE, 0, &fromAddr, &addrLen);
        buffer[strLen] = 0;
        printf("Message form server: %s\n", buffer);
    }
    closesocket(sock);
    WSACleanup();
    return 0;
}
```

2.结果输出

今天5点多自然醒，搞得一上午都没状态。下午把socket剩下的内容快速过了一下，其中文件传输以及UDP两块感觉收获比较大。明天打算开始C++项目，晚上再花点时间确定下思路。