计算机网络编程
实验报告

班级：07111707

组长：1120171189 崔程远

成员：1120172149 吴沁璇

1120172153 张澈

1120172163 王晓媛

1120172736 张鉴昊

1120172765 曾煜瑾

1120173326 曾紫飞

北京理工大学

计算机学院

2020 年 5 月

第三章 实验 3　基于停止等待协议的可靠通信

1. 实验目的

模拟数据链路层发送数据的停止等待协议，训练编程能力

2. 实验内容

采用 UDP Socket 编程接口作为模拟物理层接口实现帧的发送和接收，协议采用单工方式进行数据通信。假设 Host1 要向 Host2 发送大文件，通过数据链路层的帧每次完成数据块的可靠传输，采用停止等待协议，差错编码采用 CRC-CCITT 标准。以教材协议 3 为基础，在帧末尾增加 CRC 校验字段。

发送程序配置文件关键要点：
数据传输目的 UDP 端口
UDPPort=8888
增添发送过滤程序，模拟传输出错或丢数据帧，下面两项指明每发送多少帧出现一次出错或丢帧，此例表示每 10 帧中一帧出错，每 10 帧中一帧丢失
FilterError=10
FilterLost=10

发送程序运行屏幕输出关键要点：
显示 next_frame_to_send 变量的值，以及正在发送帧的编号
显示经过过滤器后是正确发送、模拟传输出错还是模拟帧丢失（实际没有发送）
显示接收到确认帧，确认帧的确认序号
或者显示超时
回到开始重复一直到文件发送完成

接收程序运行屏幕输出关键要点：
显示 frame_expected 变量的值，
接收帧是否出错（CRC 余数是否为零），正确则显示接收帧的发送帧序号
显示发送回确认帧，以及确认帧的确认序号
回到开始重复一直到文件接收完成

三、实验原理

这是单工传输，即只有一个方向的通信，一端负责发，另一端负责收。发送端发送的数据帧中加上 seq，在停止等待协议中 01 交替。接收端每收到一个帧，把帧中 seq 的值和 ack_expected 比较，如果相等，表示帧的序号没有问题，另外还需要根据数据和 CRC 码计算余数，如果余数为 0，表示帧在传输中没有出错。以上两点若满足，接收端发送回确认帧，接收端在超时间隔内收到确认帧，继续发送下一帧，如果在超时间隔内没有收到确认帧，重复发送此帧。

四、实验环境

语言	集成开发环境	编译器
C++	Visual Studio 2017	gcc version 4.8.1
Java	Eclipse 2019	java version ″1.8.0_65″
Python	Pycharm 2017	Python 3.7.0

五、实验步骤

三份代码的整体结构基本一致，输出格式完全一致，均采用面向对象的方法，下面以 C++代码为例进行分析。

一、发送端

- 定义全局变量

```
    SOCKET Sock;
    SOCKADDR_IN receiverAddress;
    int receiverPort = 8888; //接收端端口号

    int dataLen = 20; //数据帧长度
    int sendFrameLen = 40; //发送帧长度
    int receiveFrameLen = 1; //接收帧长度
    int serialPos = 0; //序列号位置
    int dataStartPos = 1; //数据帧起始位置
    int validDataLenPos = 21; //有效数据长度的位置
    int crcStartPos = 22; //16 位 CRC 校验码的起始位置
    int crcLen = 16; //CRC 检验码的长度
    int isEndPos = 38; //文件是否发送完毕的标识位置

    int nextFrameToSend = 0;
    long seq = 0; //数据帧的编号
    long filterSeq = 0; //发送帧的编号
    int filterError = 10; //每 10 帧 1 帧出错
    int filterLost = 10; //每 10 帧 1 帧丢失
    int firstError = 3; //第一个出错发送帧的编号
    int firstLost = 8;  //第一个丢失发送帧的编号

    //三种处理模式的代号
    const int right = 0;
    const int error = 1;
    const int lost = 2;
```

- 辅助函数

```
//获得单个字符的八位二进制码
    string getSingleBinaryString(int a)
    {
        char s1[10];
        _itoa_s(a, s1, 2);
        string s2(s1);
        while (s2.length() < 8)
        {
            s2 = "0" + s2;
        }
        return s2;
    }
```

```
//获得字符串中每个字符的八位二进制码组合而成的二进制字符串
string getBinaryString(string source)
{
    string s = "";
    for (int i = 0; i < source.length(); i++)
    {
        s += getSingleBinaryString(int(source[i]));
    }
    return s;
}
```

- 计算 CRC 码

```
//计算余数
    string getRemainderStr(string dividendStr, string divisorStr)
    {
        int dividendLen = dividendStr.length();
        int divisorLen = divisorStr.length();
        for (int i = 0; i < divisorLen - 1; i++)
        {
            dividendStr += "0";
        }
        for (int i = 0; i < dividendLen; i++)
        {
            if (dividendStr[i] == '1')
            {
                dividendStr[i] = '0';
                for (int j = 1; j < divisorLen; j++)
                {
                    if (dividendStr[i + j] == divisorStr[j])
                    {
                        dividendStr[i + j] = '0';
                    }
                    else
                    {
                        dividendStr[i + j] = '1';
                    }
                }
            }
        }
        string remainderStr = dividendStr.substr(dividendLen, divisorLen);
        return remainderStr;
    }

    string getCRCString(string s)
    {
        string gxStr = "10001000000100001";
        return getRemainderStr(s, gxStr);
    }
```

- 打印函数

```
void printTime()
    {
        SYSTEMTIME time;
        GetLocalTime(&time);
        printf("Current time: %4d-%02d-%02d %02d:%02d:%02d\n", time.wYear,
time.wMonth, time.wDay, time.wHour, time.wMinute, time.wSecond);
    }

    void Print(int method)
    {
        printTime();
        cout << "next_frame_to_send: " << nextFrameToSend << endl;
        cout << "seq: " << seq << endl;
        if (method == right)
        {
            cout << "Simulate sending right." << endl;
        }
        else if (method == error)
        {
            cout << "Simulate sending wrong." << endl;
        }
        else if (method == lost)
        {
            cout << "Simulate sending lost." << endl;
        }
        cout << endl;
    }
```

- 判断超时间隔内是否收到确认帧

```
//判断超时间隔内是否收到确认帧
    bool waitForEvent()
    {
        char *receiveFrame = new char[receiveFrameLen];
        int len = sizeof(SOCKADDR);
        int a = recvfrom(Sock, receiveFrame, 1024, 0, (SOCKADDR*)&receiverAddress,
&len);
        if (a <= 0)
        {
            printTime();
            cout << "Reveiving ack overtime, be ready to resend." << endl << endl;
            return false;
        }
        printTime();
        cout << "Received ack, ack is: " << receiveFrame[0] << endl << endl;
        return true;
    }
```

- Send 函数

```
void Send()
    {
        //加载套接字库
        WSADATA WSAdata;
        WSAStartup(MAKEWORD(2, 2), &WSAdata);

        //绑定端口
        Sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
        receiverAddress.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
        receiverAddress.sin_family = AF_INET;
        receiverAddress.sin_port = htons(receiverPort);
        int len = sizeof(SOCKADDR);

        //设置 recvfrom 的接收超时为 3 秒
        timeval tv_out;
        tv_out.tv_sec = 3000;
        tv_out.tv_usec = 0;
        setsockopt(Sock, SOL_SOCKET, SO_RCVTIMEO, (char *)&tv_out,
sizeof(timeval));

        ifstream in("D:\\desktop\\text.txt");
        char *data = new char[dataLen];

        bool flag = true;
        int pos = dataLen;
        while (true)
        {
            for (int i = 0; i < dataLen; i++)
            {
                data[i] = '\a';
            }

            in.read(data, dataLen);

            if (in.eof())//如果已读完文件，需要识别出读的字符个数
            {
                if (flag == false)//最后一个包含数据的帧收到确认帧，发送没有数据
```

```
的帧，退出循环
                {
                    char *sendFrame = new char[sendFrameLen];
                    sendFrame[isEndPos] = '1';
                    sendto(Sock, sendFrame, strlen(sendFrame), 0,
(SOCKADDR*)&receiverAddress, len);
                    cout << "Send the file finished." << endl;
                    break;
                }
                pos = 0;
                while (data[pos] != '\a')
                {
                    pos++;
                    continue;
                }
                flag = false;
            }
            seq++;

            //计算 CRC 校验码
            string s = getBinaryString(string(data).substr(0, pos));
            string crcStr = getCRCString(s);

            //为发送帧赋值
            char *sendFrame = new char[sendFrameLen];
            sendFrame[serialPos] = nextFrameToSend + '0';
            for (int i = 0; i < dataLen; i++)
            {
                sendFrame[i + dataStartPos] = data[i];
            }
            sendFrame[validDataLenPos] = pos;
            for (int i = 0; i < crcLen; i++)
            {
                sendFrame[i + crcStartPos] = crcStr[i];
            }
            sendFrame[isEndPos] = '0';

            bool mark = false;
            while (mark == false)
            {
                //模拟传输出错
                if ((filterSeq - firstError) % filterError == 0)
                {
                    char pre = sendFrame[crcStartPos];
                    sendFrame[crcStartPos] = '1' - pre + '0';
                    sendto(Sock, sendFrame, strlen(sendFrame), 0,
(SOCKADDR*)&receiverAddress, len);
                    Print(error);
                    sendFrame[crcStartPos] = pre;
                    filterSeq++;
                }
                //模拟传输丢失
                else if ((filterSeq - firstLost) % filterLost == 0)
                {
                    Print(lost);
                    filterSeq++;
                }
                //模拟传输正确
                else
                {
                    sendto(Sock, sendFrame, strlen(sendFrame), 0,
(SOCKADDR*)&receiverAddress, len);
                    Print(right);
                    filterSeq++;
                }
                // 调节传输速度
                Sleep(1000);

                mark = waitForEvent();
                if (mark == true)
                {
                    nextFrameToSend = (nextFrameToSend + 1) % 2;
                }
            }
        }
        in.close();
        closesocket(Sock);
        WSACleanup();
    }
```

- 主函数

```
int main()
{
    //需要先打开接收端，否则 recefrom 不阻塞
    string confirm;
    cout << "Please open UDPReceiver.exe and input yes!" << endl;
    cin >> confirm;
    if (confirm.compare("yes"))
    {
        return 0;
    }
    cout << endl;
    cout << "Be ready to send file..." << endl;
    UDPSender operation;
    operation.Send();
    system("pause");
}
```

二、接收端

- 定义全局变量

```
    SOCKET Sock;
    SOCKADDR_IN senderAddress;
    SOCKADDR_IN receiverAddress;
    int receiverPort = 8888; //接收端端口号

    int dataLen = 20; //数据帧长度
    int sendFrameLen = 1; //发送帧长度
    int receiveFrameLen = 40; //接收帧长度
    int serialPos = 0; //序列号位置
    int dataStartPos = 1; //数据帧其实位置
    int validDataLenPos = 21; //有效数据长度的位置
    int crcStartPos = 22; //16 为 CRC 校验码的起始位置
    int crcLen = 16; //CRC 检验码的长度
    int isEndPos = 38; //文件是否发送完毕的标识位置
    int frameExpected = 0;
```

- 辅助函数和计算 CRC 函数和发送端一致
- Receive 函数

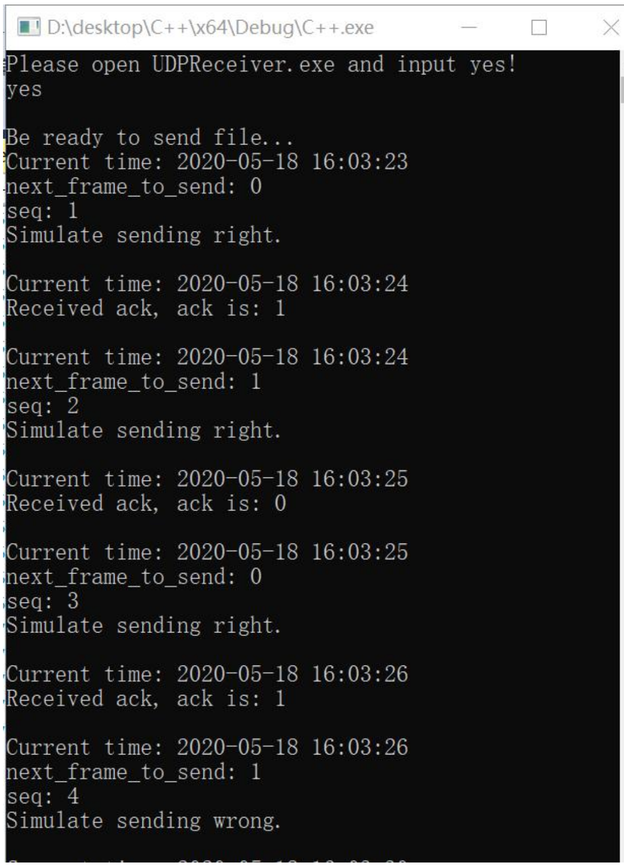
```
void Receive()
    {
        //加载套接字库
        WSADATA WSAdata;
        WSAStartup(MAKEWORD(2, 2), &WSAdata);

        //设置 UDP 通信的相关属性
        Sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
        receiverAddress.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
        receiverAddress.sin_family = AF_INET;
        receiverAddress.sin_port = htons(receiverPort);
        bind(Sock, (SOCKADDR*)&receiverAddress, sizeof(SOCKADDR));
        int len = sizeof(SOCKADDR);

        ofstream out("D:\\desktop\\copyText.txt");
        while (true)
        {
            char *receiveFrame = new char[receiveFrameLen];
            memset(receiveFrame, 0, sizeof(receiveFrame));

            recvfrom(Sock, receiveFrame, 1024, 0, (SOCKADDR*)&senderAddress,
&len);
            if (receiveFrame[isEndPos] == '1') //如果收到没有数据的帧，表示文件
全部发送完毕，退出循环
            {
                cout << "Receive the file finished." << endl;
                break;
            }
            int serial = receiveFrame[serialPos] - '0';
            int validDataLen = receiveFrame[validDataLenPos];

            //计算余数
            string dataStr =
getBinaryString(string(receiveFrame).substr(dataStartPos, validDataLen));
            string CRC = string(receiveFrame).substr(crcStartPos, crcLen);
            string crcStr = getCRCString(dataStr + CRC);

            if (atoi(crcStr.c_str()) == 0)//数据正确
            {
                printTime();
                cout << "frame_expected: " << frameExpected << endl;
                cout << "Data of the frame is right, serial is: " << serial << endl;

                char *sendFrame = new char[sendFrameLen];
                sendFrame[0] = 1 - frameExpected + '0';
                sendto(Sock, sendFrame, strlen(sendFrame), 0,
(SOCKADDR*)&senderAddress, len);
                cout << "Sending ack, ack is: " << (1 - frameExpected) << endl <<
endl;

                if (serial - 0 == frameExpected)
                {
                    char *dataFrame = new char[validDataLen];
                    for (int i = 0; i < validDataLen; i++)
                    {
                        dataFrame[i] = receiveFrame[dataStartPos + i];
                    }
                    out.write(dataFrame, validDataLen);
                    frameExpected = (frameExpected + 1) % 2;
                }
            }
            else //数据出错
            {
                printTime();
                cout << "Data of the frame is wrong, doesn't send ack." << endl <<
endl;
            }
        }
        out.close();
        closesocket(Sock);
        WSACleanup();
    }
```

- 主函数

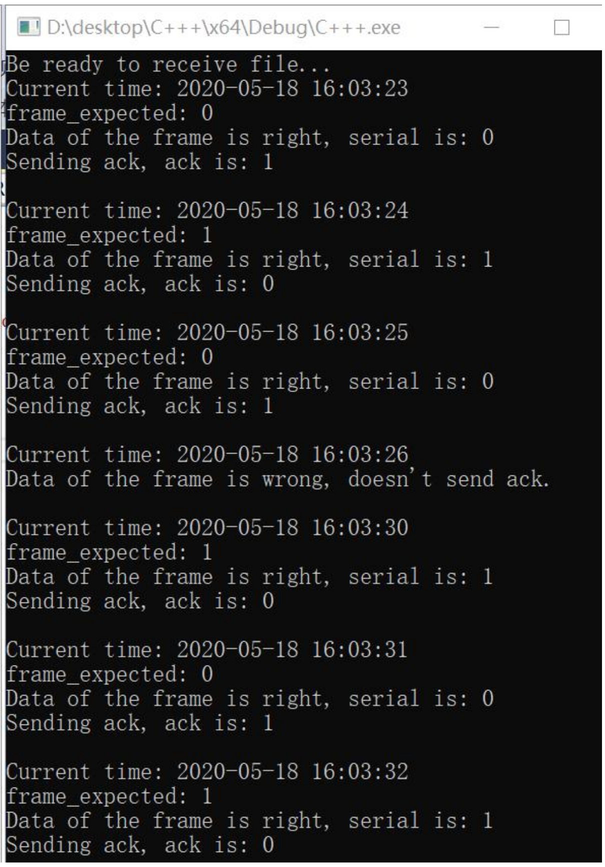
```
int main()
{
    cout << "Be ready to receive file..." << endl;
    UDPReceiver operation;
    operation.Receive();
    system("pause");
}
```

6. 实验结果

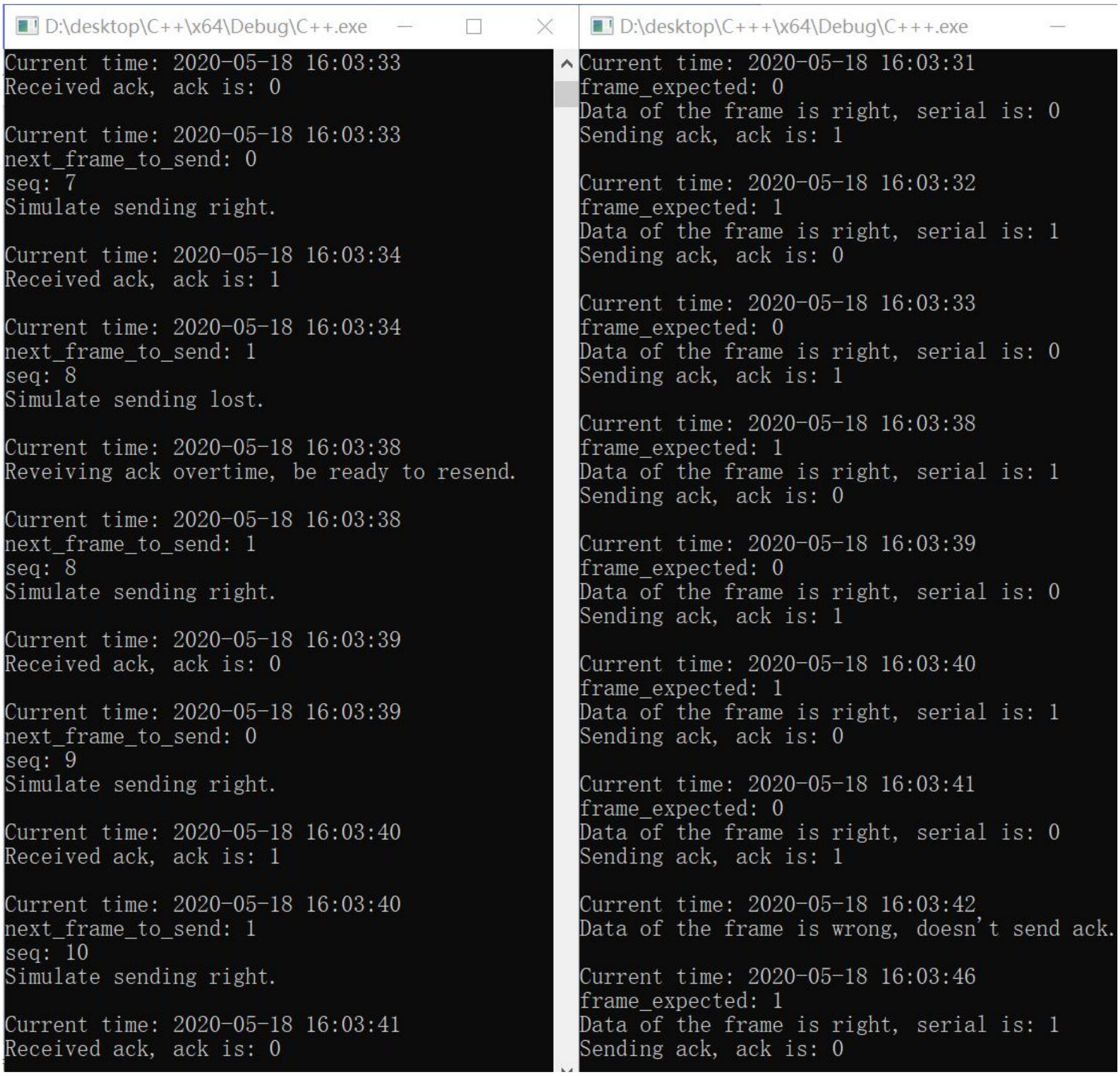
三份代码的输出格式完全一致，下面以 C++代码输出为例



```
Please open UDPReceiver.exe and input yes!
yes

Be ready to send file...
Current time: 2020-05-18 16:03:23
next_frame_to_send: 0
seq: 1
Simulate sending right.

Current time: 2020-05-18 16:03:24
Received ack, ack is: 1

Current time: 2020-05-18 16:03:24
next_frame_to_send: 1
seq: 2
Simulate sending right.

Current time: 2020-05-18 16:03:25
Received ack, ack is: 0

Current time: 2020-05-18 16:03:25
next_frame_to_send: 0
seq: 3
Simulate sending right.

Current time: 2020-05-18 16:03:26
Received ack, ack is: 1

Current time: 2020-05-18 16:03:26
next_frame_to_send: 1
seq: 4
Simulate sending wrong.
```

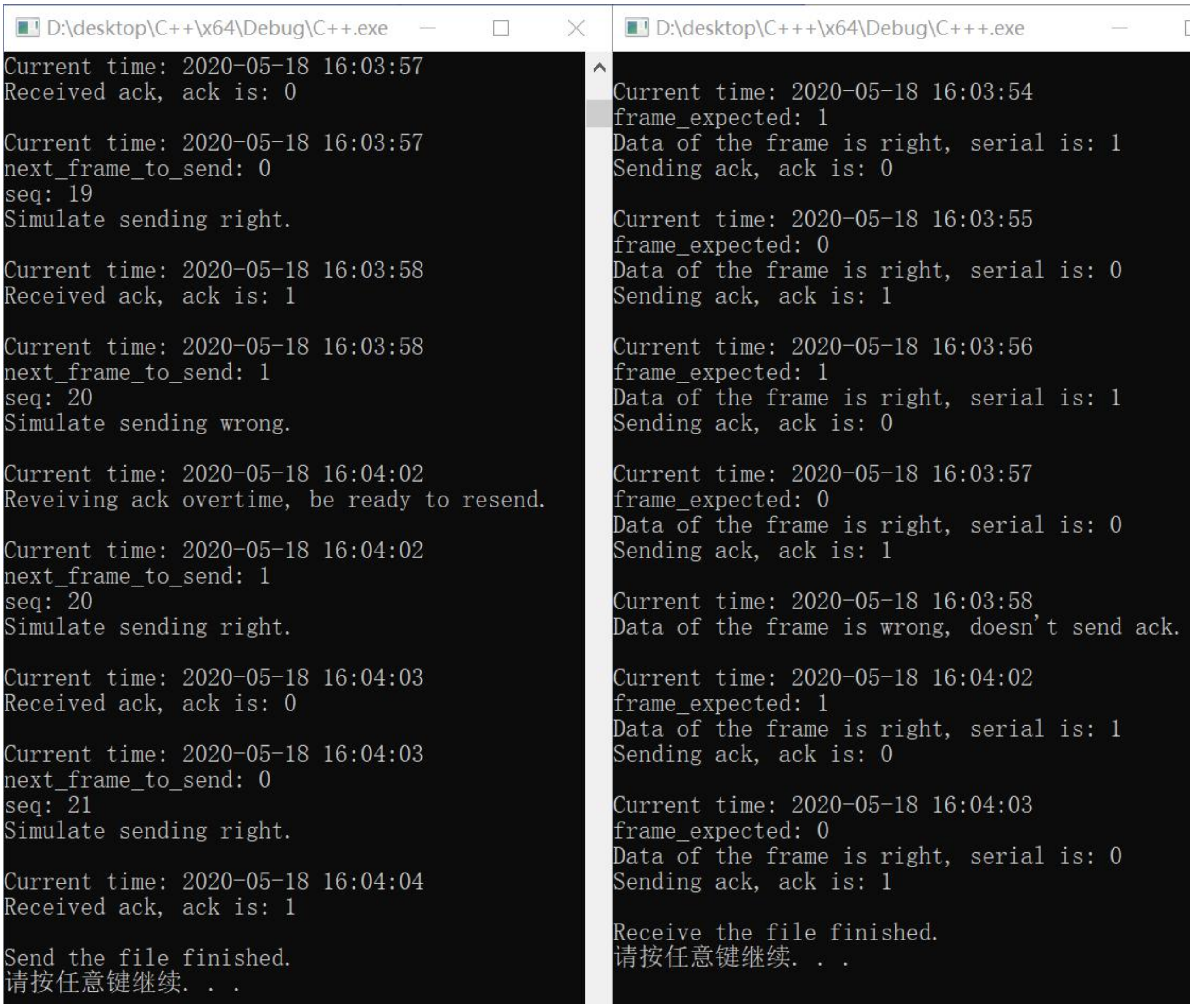


```
Be ready to receive file...
Current time: 2020-05-18 16:03:23
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:24
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:25
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:26
Data of the frame is wrong, doesn't send ack.

Current time: 2020-05-18 16:03:30
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:31
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:32
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0
```


D:\desktop\C++\x64\Debug\C++.exe
D:\desktop\C+++\x64\Debug\C+++.exe

```
Current time: 2020-05-18 16:03:33
Received ack, ack is: 0

Current time: 2020-05-18 16:03:33
next_frame_to_send: 0
seq: 7
Simulate sending right.

Current time: 2020-05-18 16:03:34
Received ack, ack is: 1

Current time: 2020-05-18 16:03:34
next_frame_to_send: 1
seq: 8
Simulate sending lost.

Current time: 2020-05-18 16:03:38
Reveiving ack overtime, be ready to resend.

Current time: 2020-05-18 16:03:38
next_frame_to_send: 1
seq: 8
Simulate sending right.

Current time: 2020-05-18 16:03:39
Received ack, ack is: 0

Current time: 2020-05-18 16:03:39
next_frame_to_send: 0
seq: 9
Simulate sending right.

Current time: 2020-05-18 16:03:40
Received ack, ack is: 1

Current time: 2020-05-18 16:03:40
next_frame_to_send: 1
seq: 10
Simulate sending right.

Current time: 2020-05-18 16:03:41
Received ack, ack is: 0
```

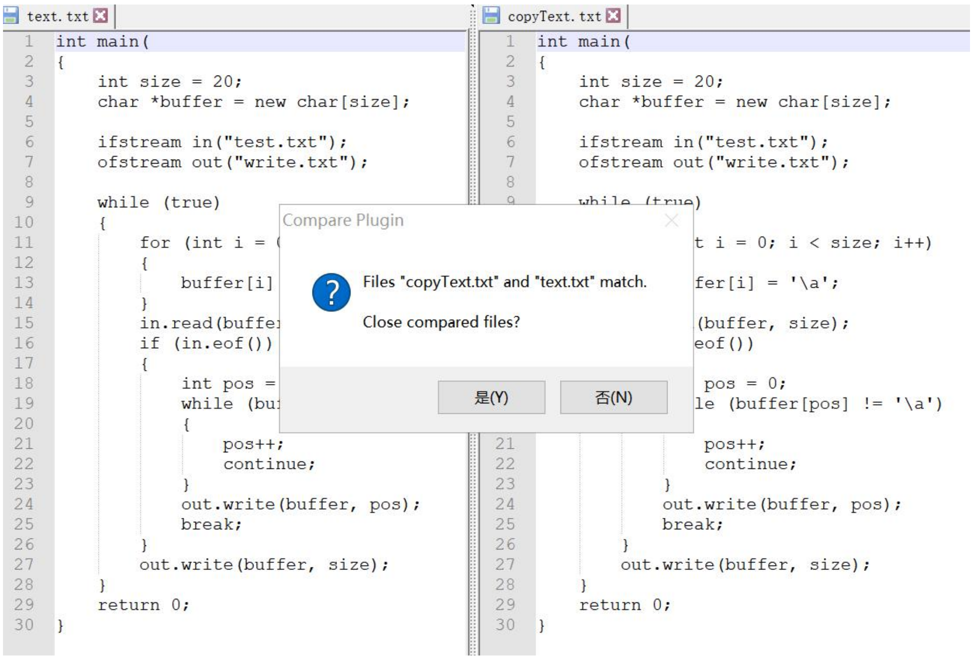
```
Current time: 2020-05-18 16:03:31
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:32
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:33
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:38
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:39
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:40
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:41
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:42
Data of the frame is wrong, doesn't send ack.

Current time: 2020-05-18 16:03:46
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0
```


D:\desktop\C++\x64\Debug\C++.exe
D:\desktop\C+++\x64\Debug\C+++.exe

```
Current time: 2020-05-18 16:03:57
Received ack, ack is: 0

Current time: 2020-05-18 16:03:57
next_frame_to_send: 0
seq: 19
Simulate sending right.

Current time: 2020-05-18 16:03:58
Received ack, ack is: 1

Current time: 2020-05-18 16:03:58
next_frame_to_send: 1
seq: 20
Simulate sending wrong.

Current time: 2020-05-18 16:04:02
Reveiving ack overtime, be ready to resend.

Current time: 2020-05-18 16:04:02
next_frame_to_send: 1
seq: 20
Simulate sending right.

Current time: 2020-05-18 16:04:03
Received ack, ack is: 0

Current time: 2020-05-18 16:04:03
next_frame_to_send: 0
seq: 21
Simulate sending right.

Current time: 2020-05-18 16:04:04
Received ack, ack is: 1

Send the file finished.
请按任意键继续. . .
```

```
Current time: 2020-05-18 16:03:54
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:55
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:56
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:03:57
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Current time: 2020-05-18 16:03:58
Data of the frame is wrong, doesn't send ack.

Current time: 2020-05-18 16:04:02
frame_expected: 1
Data of the frame is right, serial is: 1
Sending ack, ack is: 0

Current time: 2020-05-18 16:04:03
frame_expected: 0
Data of the frame is right, serial is: 0
Sending ack, ack is: 1

Receive the file finished.
请按任意键继续. . .
```

比较发送文件和接收文件



```
int main(
{
    int size = 20;
    char *buffer = new char[size];

    ifstream in("test.txt");
    ofstream out("write.txt");

    while (true)
    {
        for (int i = (
        {
            buffer[i]
        }
        in.read(buffe
        if (in.eof())
        {
            int pos =
            while (bu
            {
                pos++;
                continue;
            }
            out.write(buffer, pos);
            break;
        }
        out.write(buffer, size);
    }
    return 0;
}
```

Compare Plugin

Files "copyText.txt" and "text.txt" match.

Close compared files?

```
int main(
{
    int size = 20;
    char *buffer = new char[size];

    ifstream in("test.txt");
    ofstream out("write.txt");

    while (true)
          t i = 0; i < size; i++)
          fer[i] = '\a';
          (buffer, size);
          eof())
            pos = 0;
          le (buffer[pos] != '\a')
                pos++;
                continue;
            }
            out.write(buffer, pos);
            break;
        }
        out.write(buffer, size);
    }
    return 0;
}
```

7. 实验总结

这个实验的三份代码太折腾了，难点有以下几点：

1. 读文件，发送端每发送完一帧收到确认帧后，从文件中读取一定长度的数据，最后一次读取的数据长度不是之前的固定长度，所以需要判断，接收端同样需要识别出有效数据。

2. UDP 传输，Java 和 python 的 UDP 是传输字节数据，而 C++的 UDP 是传输字符串，所以在 Java 和 Python 中的 CRC 校验是两个字节，而在 C++中是长度为 16 的 01 字符串，这需要一系列不同的转换函数。

3. 模拟三种模式，题目要求每 10 帧 1 帧出错和丢失，其他发送帧就是模拟正确传输。模拟出错通过把计算后的 CRC 校验码中的某一位更改，模拟丢失就不发送。

4. 处理超时间隔，三种语言的 UDP 传输都可以设置接收函数的阻塞时间，如果超时退出阻塞，Java 和 Python 中是接收异常，C++中是返回值为-1。

5. 传输完毕退出，发送端收到最后一个数据帧的确认帧后，发送没有数据的标识帧，打印并退出；接收端收到一帧，识别出是标识帧，打印并退出。

6. 调试时间过长，UDP 传输和不同类型数据之间的转换很多坑点，用到了很多奇怪的函数，每份代码都需要调试很长时间，远超过写代码的时间，经过数天调试，最终成功将三份代码的输出格式统一。

7. 更复杂更心累的体验的请见下一个实验——GBN。

8. 参考博客

- C++

1. C++ udp 实现简单的通信
https://blog.csdn.net/qq_39200996/article/details/89314881

2. 关于 recvfrom 接收超时
https://blog.csdn.net/u010951692/article/details/38657173

- Java

3. UDP 发送数据和接收数据
https://blog.csdn.net/weixin_44706512/article/details/89955175

4. Java 网络编程 之 UDP 文件传输
https://blog.csdn.net/verejava/article/details/80620553?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task

5. Java UDP 实现文件传输
https://blog.csdn.net/Ramer42/article/details/83582240

6. java 基础知识之 FileInputStream 流
https://blog.csdn.net/ai_bao_zi/article/details/81097898

7. Java UDP 通信：Java DatagramSocket 类和 DatagramPacket 类
http://c.biancheng.net/view/1203.html

- Python

8. Python 实现 socket——udp 的传输与接收
https://blog.csdn.net/weixin_44321116/article/details/96475120

9. Python 中的 socket 网络编程(TCP/IP，UDP)讲解
https://blog.csdn.net/csdn15698845876/article/details/78311576

10. python - udp socket 通信循环发送和接收数据
https://blog.csdn.net/xuezhangjun0121/article/details/88786590

11. python socket 编程详细介绍
https://blog.csdn.net/weixin_34112900/article/details/85070202?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task

12. Python 实现 UDP 协议下的文件传输
https://blog.csdn.net/qq_38898129/article/details/89319767