

**计算机网络实验报告**

**实验3-2**

****

专 业 信息安全

学 号 2113662

姓 名 张丛

班 级 信安一班

1. **实验目的**

在实验3-1的基础上，将停等机制改成基于滑动窗口的流量控制机制，发送窗口和接收窗口采用相同大小，支持累积确认，完成给定测试文件的传输。

1. **实验要求**

➢ 协议设计：数据包格式，发送端和接收端交互，详细完整

➢ 流水线协议：多个序列号

➢ 累积确认：Go Back N

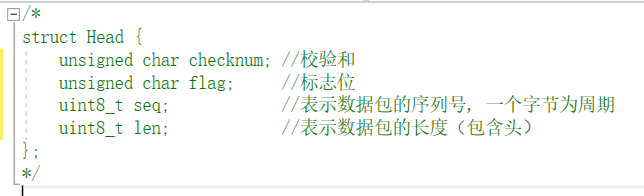
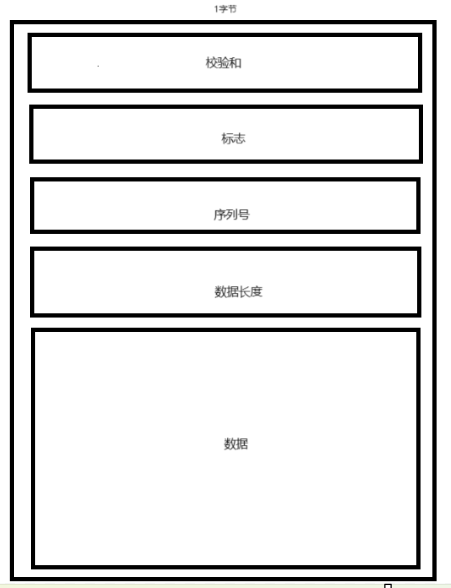
➢ 日志输出：收到/发送数据包的序号、ACK、校验和等，发送端和接收端的窗口大小等情况，传输时间与吞吐率。

➢ 测试文件：1.jpg、2.jpg、3.jpg、helloworld.txt

1. **实验内容**

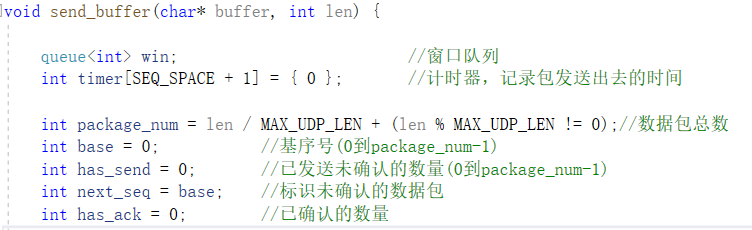
在实验3-1中的重复内容就不再赘述。

**协议设计**



**滑动窗口**

滑动窗口机制的重要**标识**如下：



发送单个数据包使用send\_package()函数：



在窗口中**发送数据**：



流程为：

1. 判断是否可以停止发送数据。通过是否接收到所有ACK来判断。
2. 给数据包赋予正确的在序号空间内的序号。
3. 如果窗口没满，并且还有数据包需要发送，就发送这个数据包，记录发送时间（用于超时重传），更新窗口信息（win.push）。
4. next\_seq++和has\_send++可以理解为指向下一个数据包。

上面的代码中，是发送一个数据包后就去接收ACK，将上图的第三个if改为while，可以实现发送窗口内所有数据包后再转到recvfrom函数。

发送方采用非阻塞的接收消息方式, 可以使收发数据包同时进行。

**窗口的滑动**：

在接收到ACK后，通过接收端回应的ACK来更新窗口。

需要明白的是，发送方（Client）接收到ACK\_X，即表明接收方（Server）已经正确接收了从最开始的数据包到数据包X间的所有数据包。

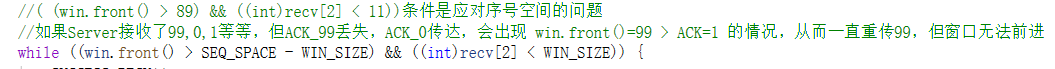
窗口队列的win.front()即为最远的已发送但未确认的数据包序号，通过win.front()与recv[2](即ACK）进行比较，来决定如何前进窗口。



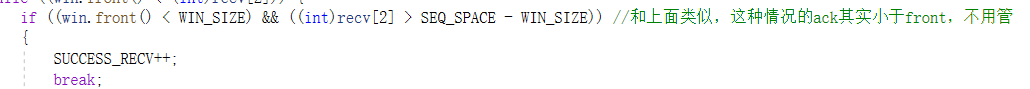
win.pop()将数据包移出窗口队列，表示这个数据包已经确认（被正确接收）。

通过一步一步的pop()，base将指向最远的已发送未被确认的数据包。（其实也可以直接通过ACK来给base赋值表示窗口的移动，但win这个队列总是要一个一个pop()的，是一样的）

代码中有下面两个判断：



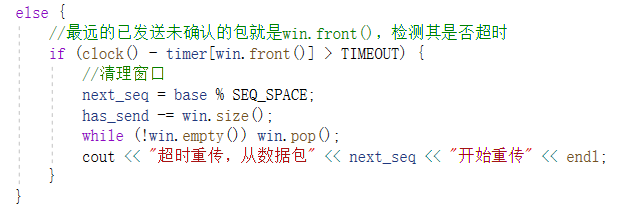
和



是因为我设置了数据包的序号空间，避免序号和ACK比较出现bug。

知识点：对于SR协议而言，窗口长度必须小于等于序号空间大小的一半。

**超时重传**：



在发送数据包时，在timer中记录了发送时间，从而可以来判断是否需要超时重传。

超时后，则重置标识，清空窗口队列，在下一个循环中重新发送窗口。

**累积确认**

**接收方（Server）**：

成功收到了连续的数据包后，它将发送一个包含最后成功接收的数据包序号的ACK。

这个确认序号表示已经成功接收了该序号之前的所有数据。

代码如下：

1. while (1) {
2. if (recv\_req >= 100)
3. {
4. recv\_req %= 100;
5. }
6. memset(recv, 0, sizeof(recv));
7. while (recvfrom(m\_ServerSocket, recv, MAX\_UDP\_LEN + 4, 0, (sockaddr\*)&m\_ClientAdress, &len\_addr) == SOCKET\_ERROR);
8. char send[3];
9. int flag = 0;
10. SUCC\_RECV++;
11. *//发送ACK确认的序列号*
12. if (check\_sum(recv, MAX\_UDP\_LEN + 4) == 0 && (uint8\_t)recv[2] == recv\_req) {
13. recv\_req++;
14. flag = 1;
15. }
16. send[1] = ACK;    *//标志*
17. if (recv\_req == 0)
18. {
19. send[2] = 99;
20. cout << "接收端确认序列号: " << 99 << endl;
21. }  *//序号*
22. else
23. {
24. send[2] = recv\_req - 1;
25. cout << "接收端确认序列号: " << recv\_req - 1 << endl;
26. }
27. send[0] = check\_sum(send + 1, 2); *//校验和*
29. sendto(m\_ServerSocket, send, 3, 0, (sockaddr\*)&m\_ClientAdress, sizeof(m\_ClientAdress));
30. if (flag)
31. break;
32. }

对于接收到的正确的序号，将回应这个正确序号的ACK；对于非期待的序号，将回应上一个正确的确认ACK。

**发送方（Client）**：根据ACK更新窗口、计时器等等。

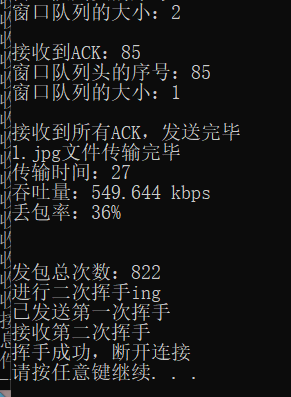
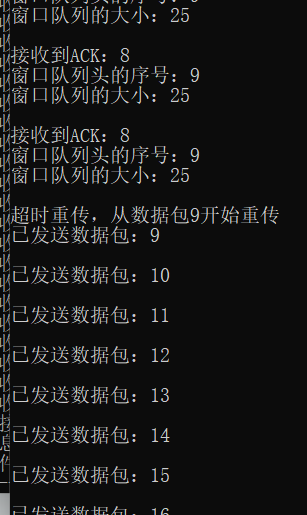
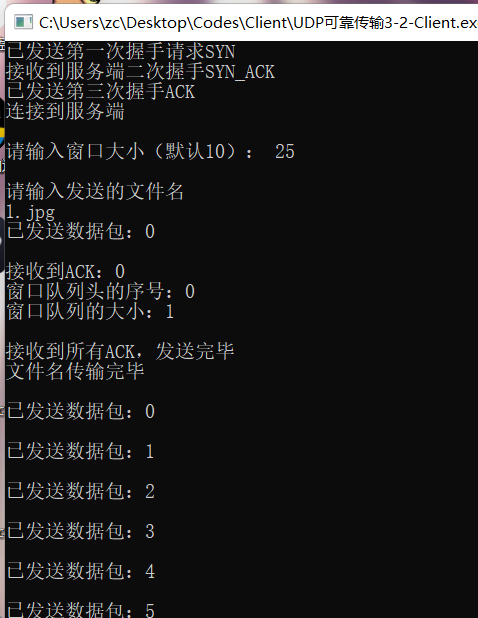
**实验效果**

在路由器设置丢包、延时后

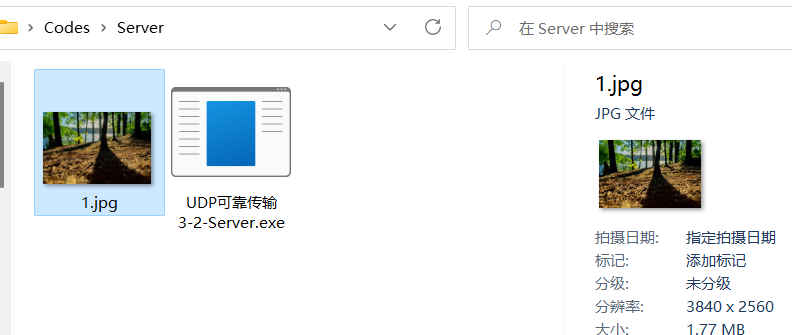


对于四个测试文件：

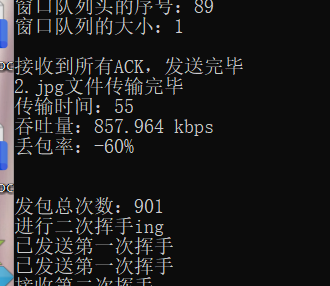
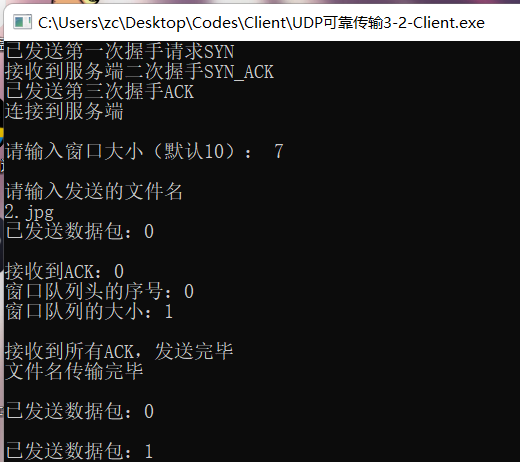
1.jpg:

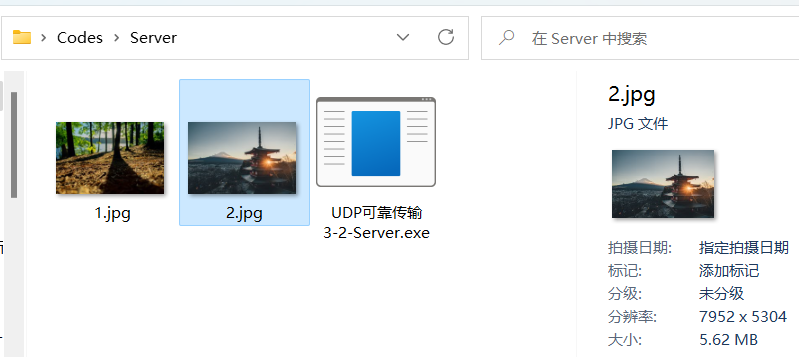


传输成功：

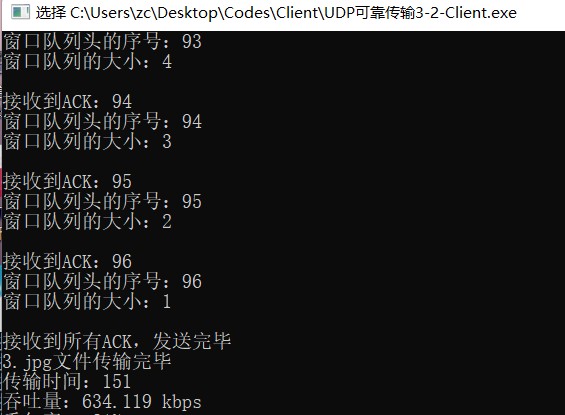


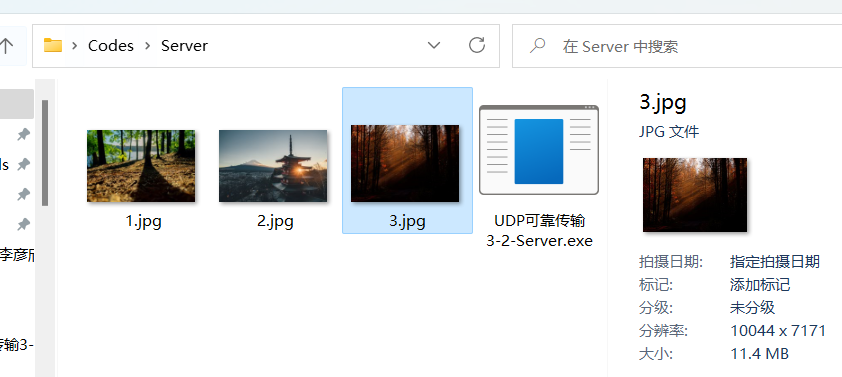
1. jpg:



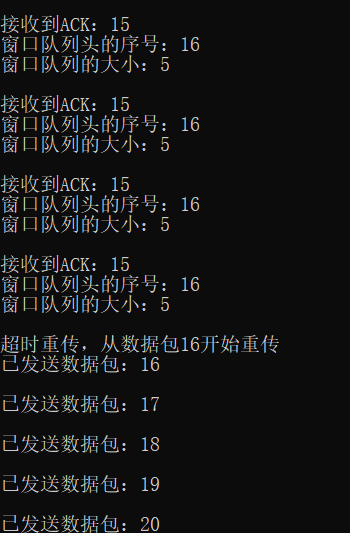
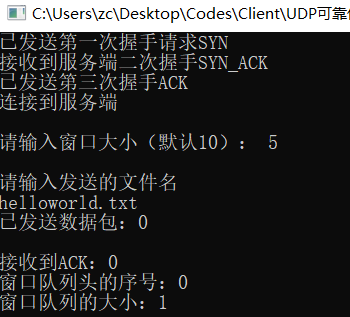


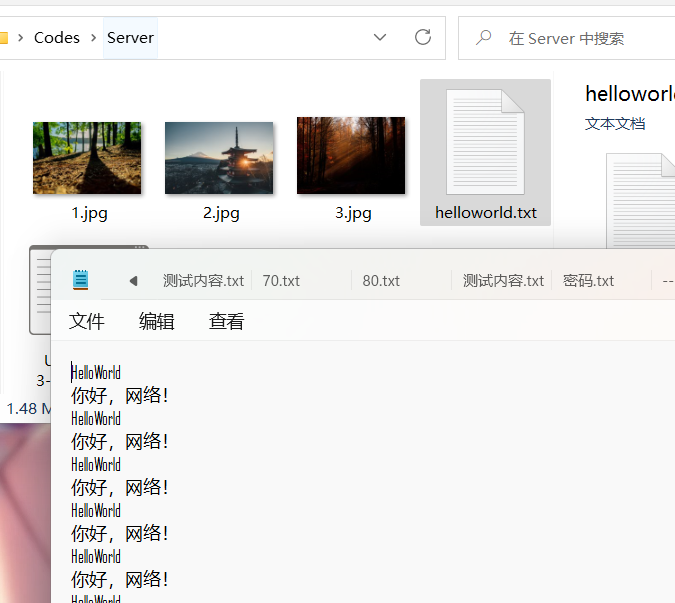
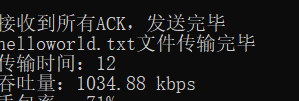
1. jpg:





helloworld.txt:





1. **思考与总结**

实现了UDP可靠传输的滑动窗口机制和累计确认机制，复习到课上的理论知识点。

对于可靠传输的实现流程等等有了更清晰的认识。

不足：实验中发送端使用的是单线程，下次实验可以使用多线程，发送和接收数据包同时进行。