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，采用固定窗口大小，支持累积确认，完成给定测试文件的传输。

【相关知识】

·发送窗口：在任意时刻，发送方都维持了一个连续的允许发送的帧的序号，称为滑动窗口。

·接收窗口：在任意时刻，接收方也在维持的一个连续的允许接收的帧的序号。其大小和上下界均不要求和发送窗口相同。

·滑动窗口：可以理解为对发送窗口和接收窗口的统称，记录了一个连续的允许被使用的帧的序号，由于在实际程序运行过程中允许使用的帧在不断变化，因此可以理解为在进行滑动。

·GBN：发送方在发完一个数据帧后，连续发送若干个数据帧，即使在连续发送过程中收到了接收方发来的应答帧，也可以继续发送。且发送方在每发送完一个数据帧时都要设置超时定时器。只要在所设置的超时时间内仍收到确认帧，就要重发相应的数据帧。如：当发送方发送了N个帧后（N为窗口大小），若发现该N帧的前一个帧在计时器超时后仍未返回其确认信息，则该帧被判为出错或丢失，此时发送方就不得不重新发送出错帧及其后的N帧。

接受帧只允许按顺序接受帧。为了减少开销，累计确认允许接收端在连续收到好几个正确的确认帧后，只对最后一个数据帧发确认信息，或者可以在自己有数据要发送时才将对以前正确收到的帧加以捎带确认。这就是说，对某一数据帧的确认就表明该数据帧和这以前所有的数据帧均已正确无误地收到了。

GBN协议的接受窗口为1，可以保证按序接受数据帧。若采用n个比特对帧编号，则其发送窗口的尺寸应满足：1~2^(n-1)。若发送窗口的尺寸大于2^(n-1)，则会造成接受方无法分辨新帧和旧帧。

【代码实现思路】

对于发送方client而言：

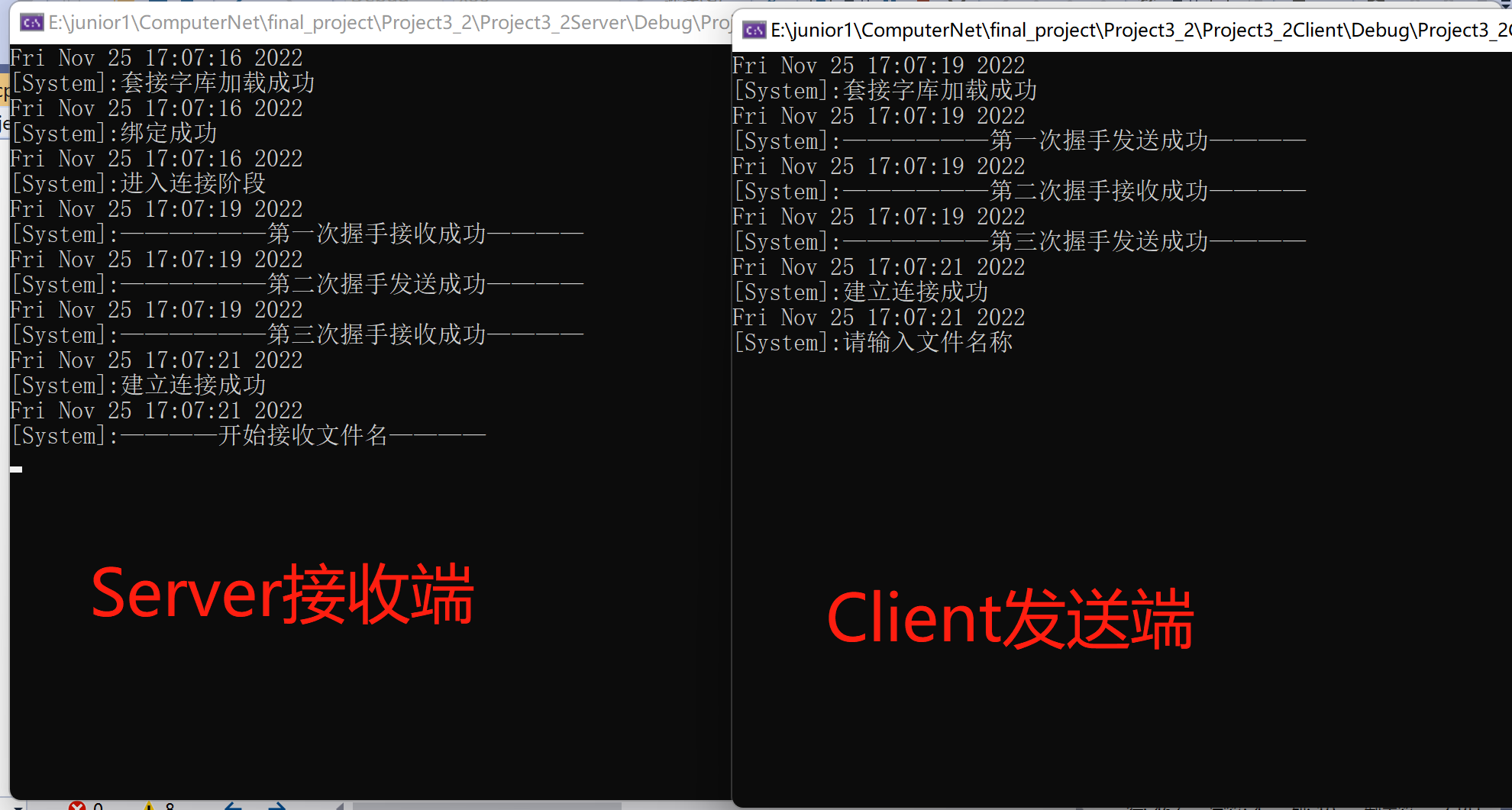
1. 维护一个发送窗口，需要记录此时的窗口下限floor和窗口上限ceiling，floor不能<=1，ceiling不能>总共发包数量packetnum。
2. 从floor到ceiling，按照序号递增的方法发送各个包，发送完第一个包时就开始计时。
3. 当收到来自于接收方的ACK时，判断接收到的包的正确性，倘若正确且当前收到的ACK包的sequence>=floor的话，则floor=sequence+1&&ceiling=floor+windowsize-1，；发送端发送窗口内新的未发送的包，重新开始计时。如果接收到的ACK包错误 || 接收到的ACK包的序号不是大于floor，那么什么也不干。
4. 直到client发现超时，之后重传从floor到ceiling的所有包并且重新开始计时。
5. 当ceiling==packetnum时，接下来窗口上限不再移动，只需要修改下限floor即可。因此当发送完最后一个数据包后，接下来对每个到来的ACK包进行判别，如果sequence==floor+1，则floor+1；否则如果收到的是最后一个数据包的确认包（头部标志位为FINAL\_ACK），则表示接收端所有数据接收成功，等待两个时钟周期后结束传送数据（此处停顿是为了和挥手包区分开），对于其余情况都不做任何处理，直到超时重发窗口内所有数据包。

对于接收方server而言，无需修改数据，维持3-1中的编写即可，逻辑如下：

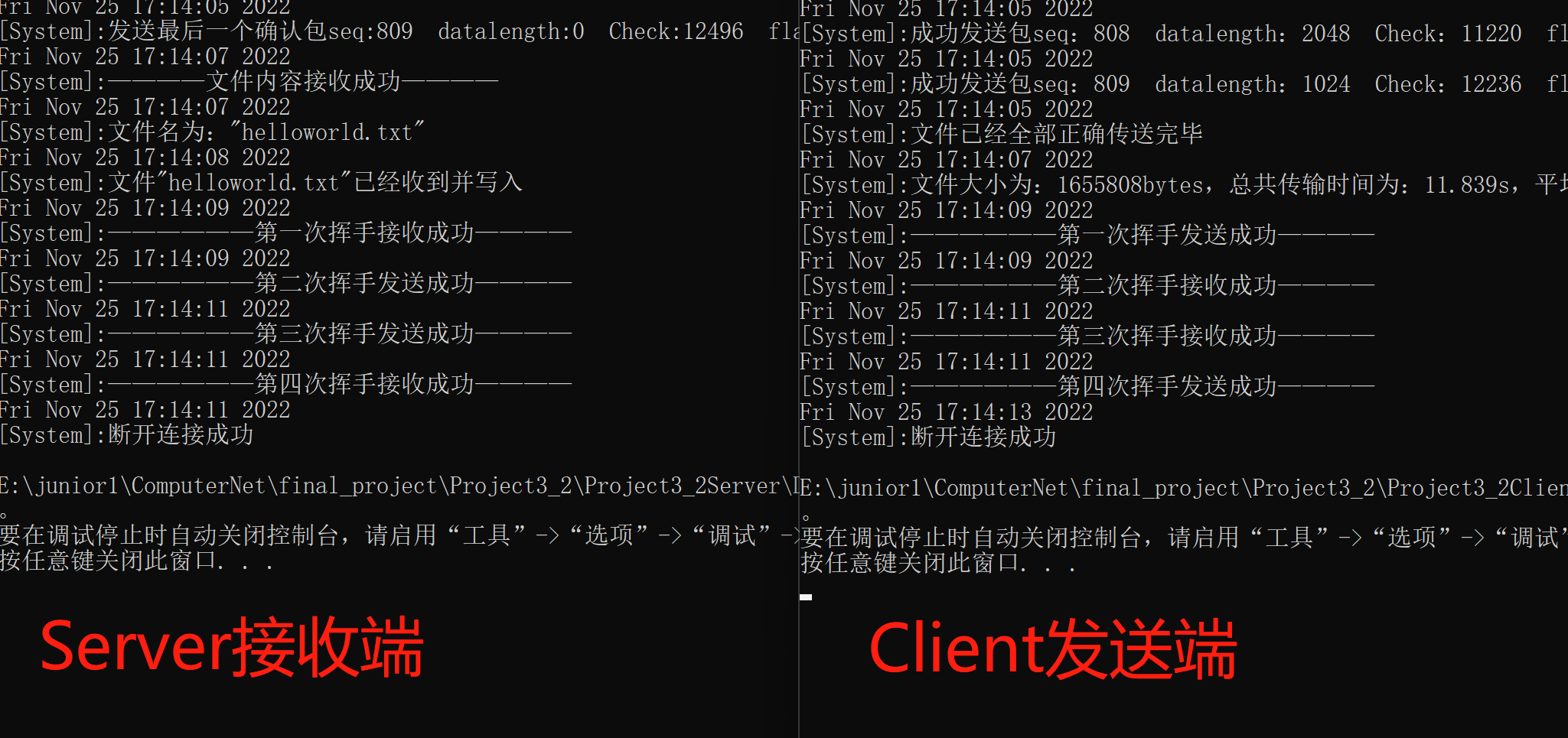
1. 本地维护一个Ssequence用于记录希望得到的数据包序列号，对于接收到的数据包进行判断。
2. 如果收到的数据包校验和正确&&序列号为期望得到的数据包序列号，那么接收此数据包并写入文件缓冲区。维护的Ssequence+1，之后发送回去的ACK确认包序列号为当前的Ssequence，表示接下来希望接收的数据包编号为Ssequence。
3. 如果收到的数据包校验和错误 || 序列号不是期望得到的数据包序列号，那么拒绝接收此数据包，维护的Ssequence不做变动，之后发送回去的ACK确认包序列号为当前的Ssequence，表示接下来希望接收的数据包编号为仍然为Ssequence。
4. 直到接收到的数据包标志位为FINAL时表示为最后一个数据包，此时发回ACK包头部置位为FINAL\_ACK，等待2MSL查看是否收到来自客户端发送的数据包，如果没收到说明FINAL\_ACK包被正确接收，结束等待。如果收到数据包说明FINAL\_ACK包丢失，此时等待客户端发来数据包后重传FINAL\_ACK包即可。

【实验结果图】

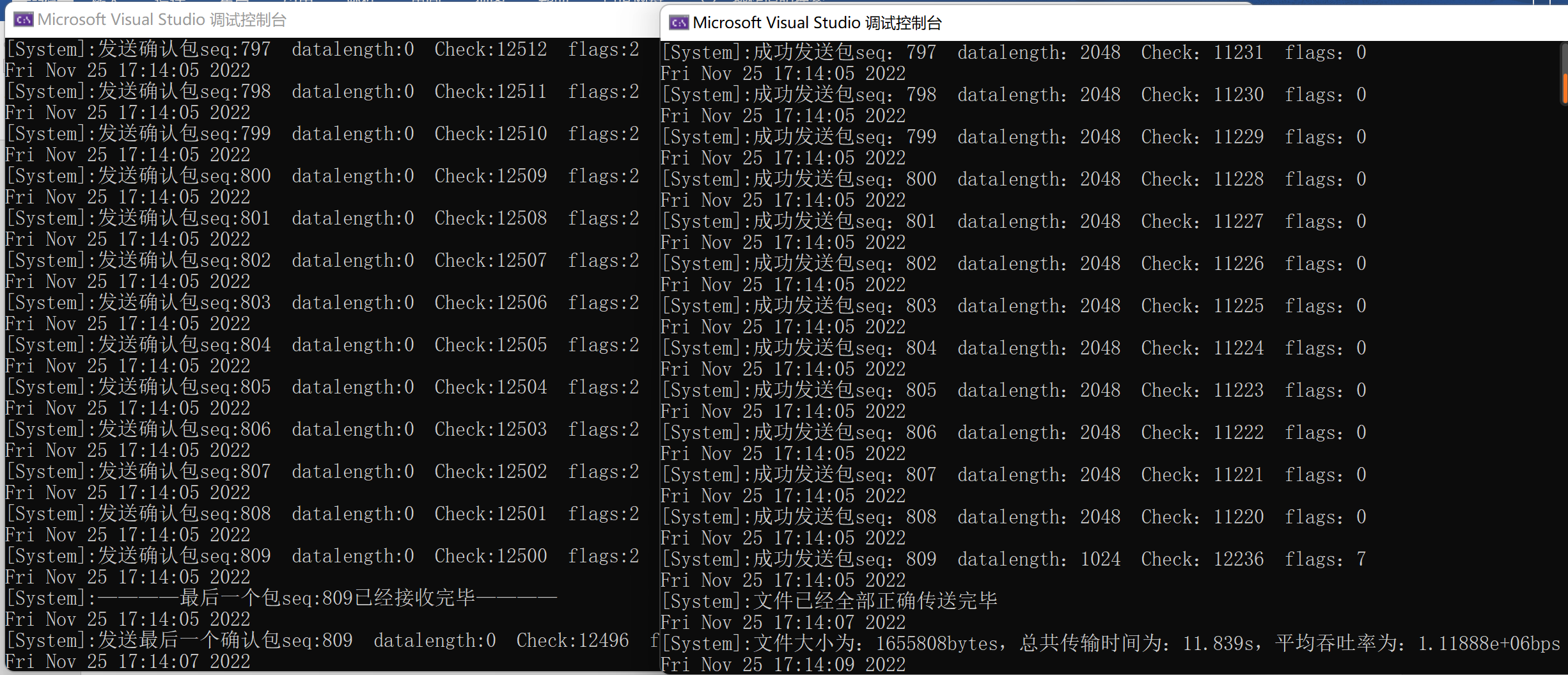
建立连接的三次握手包发送接受结果：



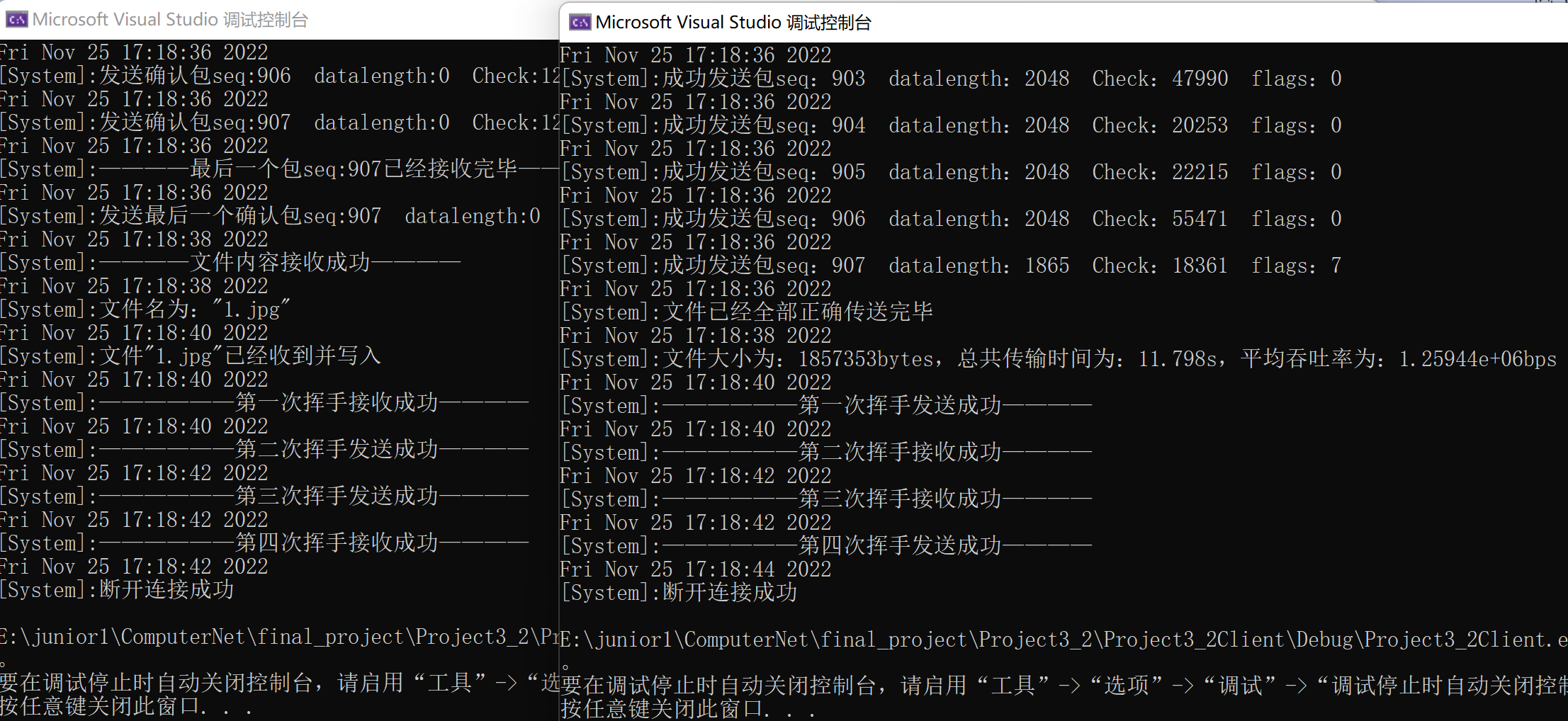
挥手结果图如下：



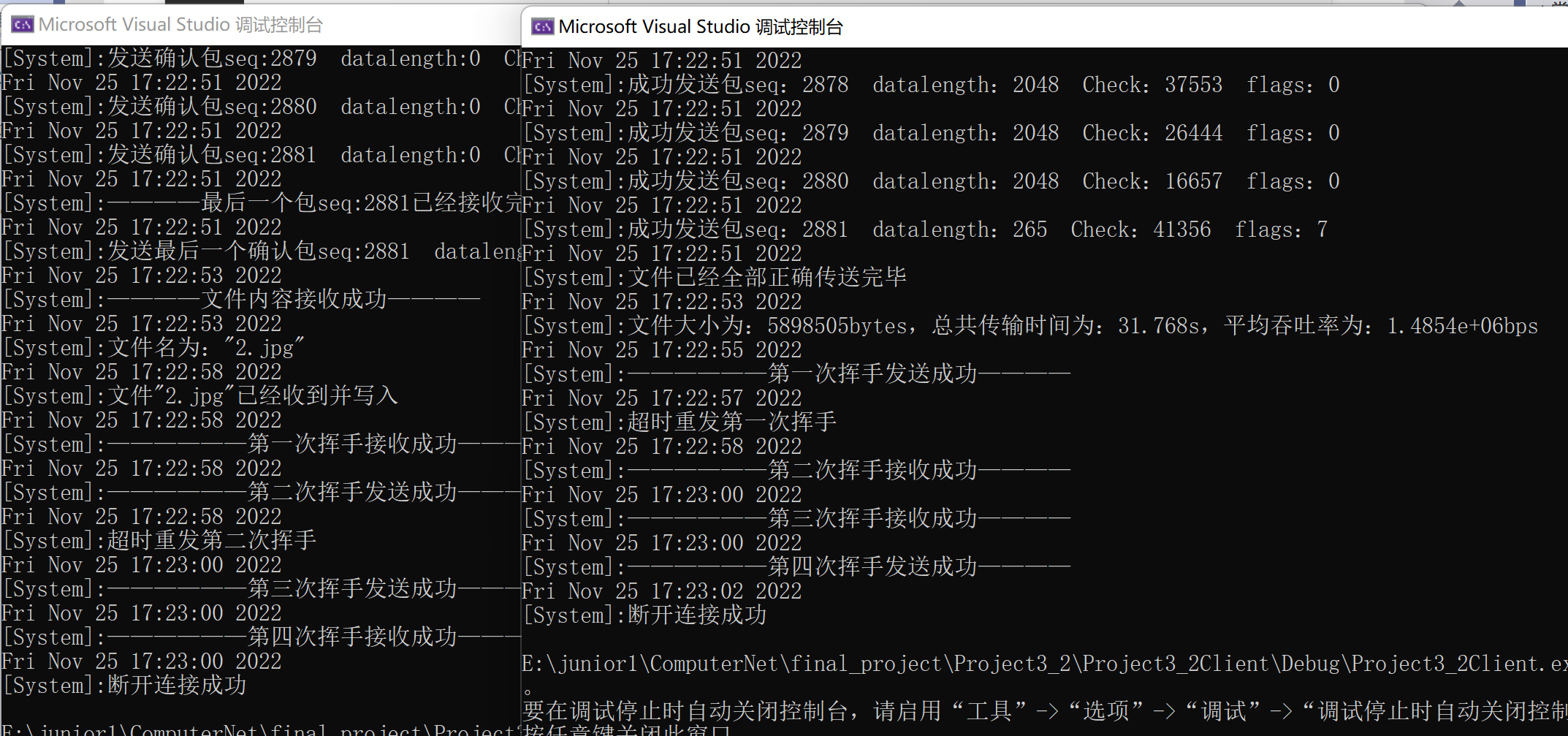
发送helloworld.txt结果图如下，传送时间为11.839s，吞吐率为1.1188e+06bps：



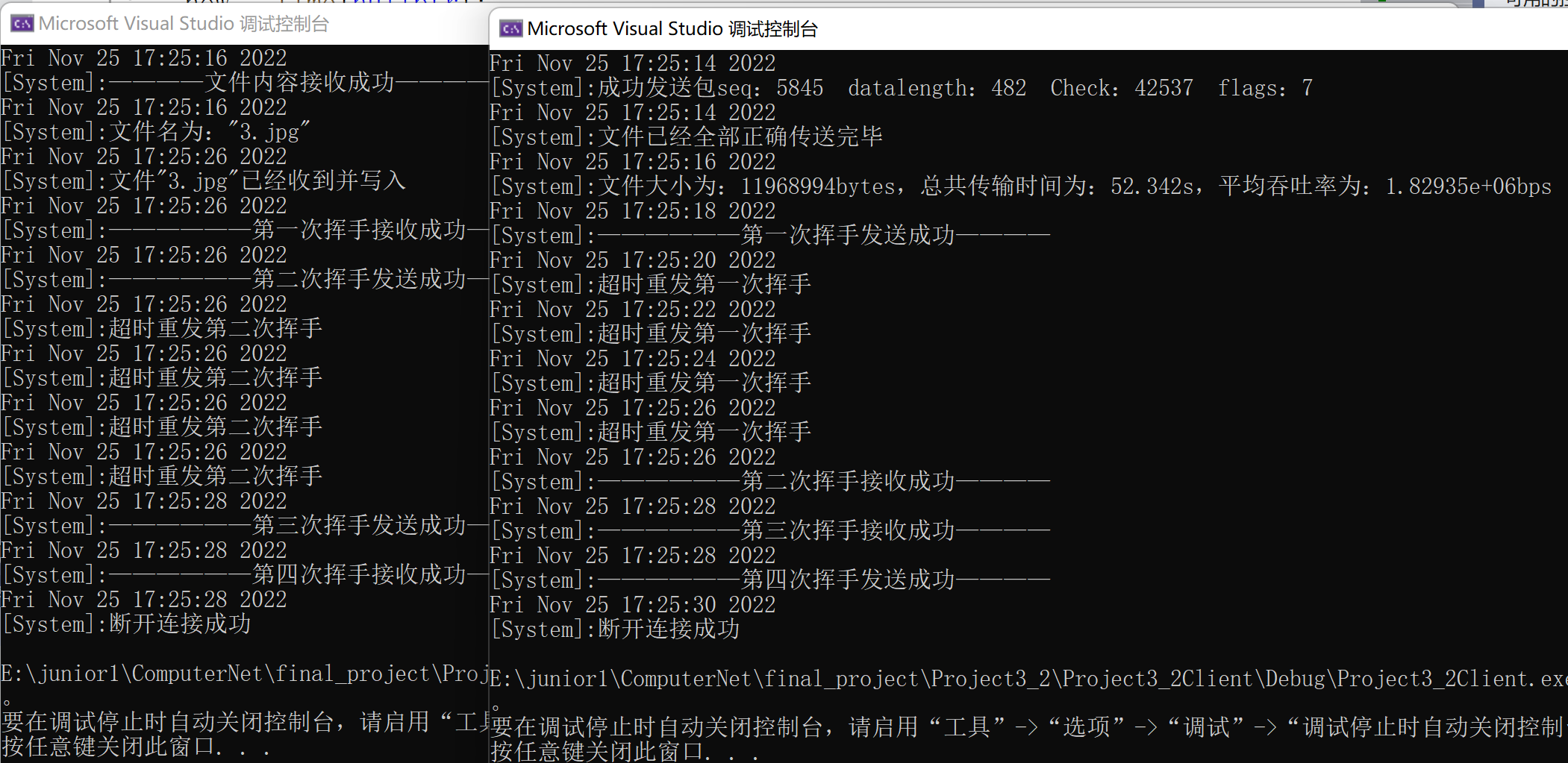
发送1.jpg结果如下图，传送时间为11.798s，平均吞吐率为1.25944e+06bps：



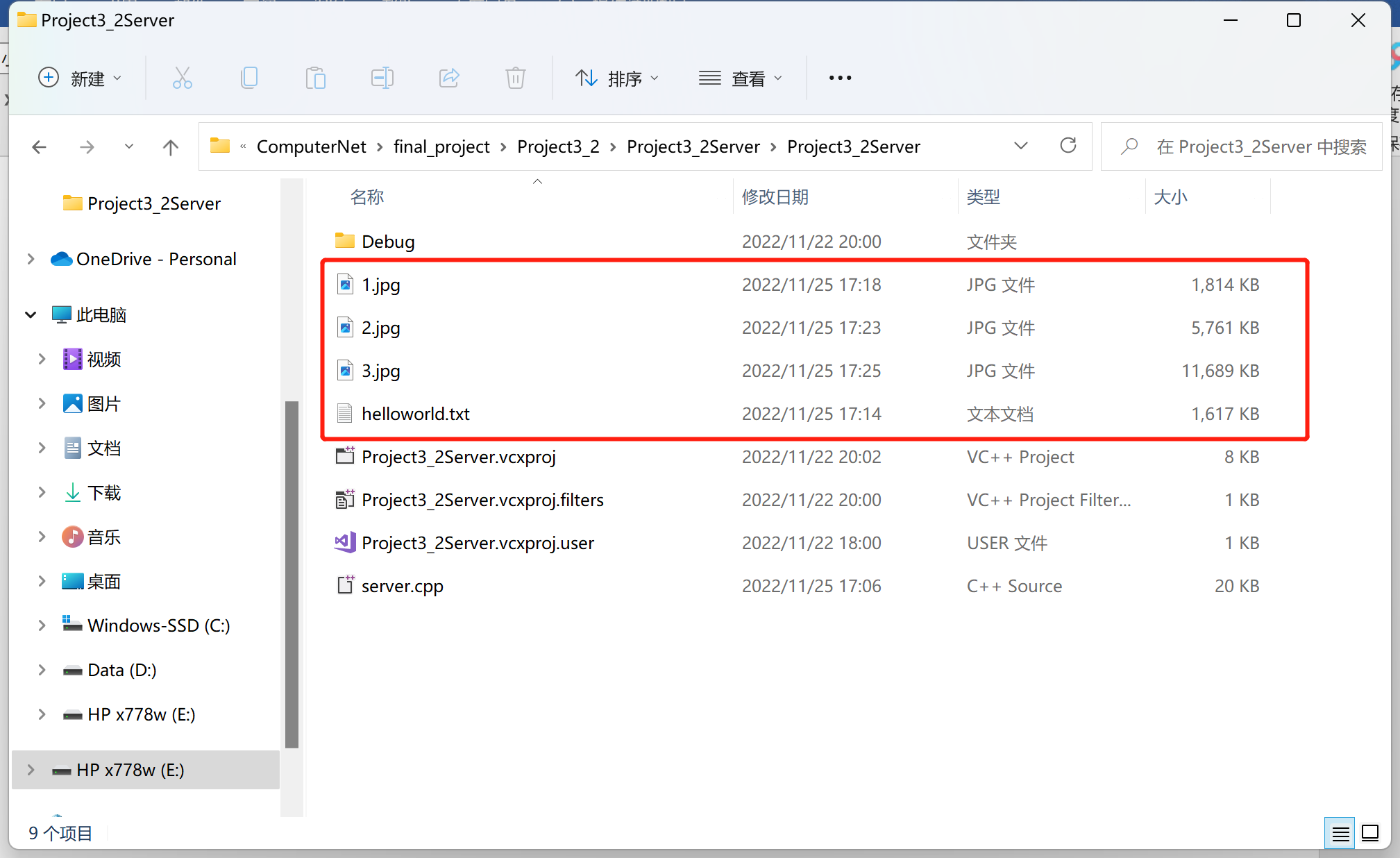
发送2.jpg结果如下图，发送时间为31.768s，平均吞吐率为1.485e+06bps：



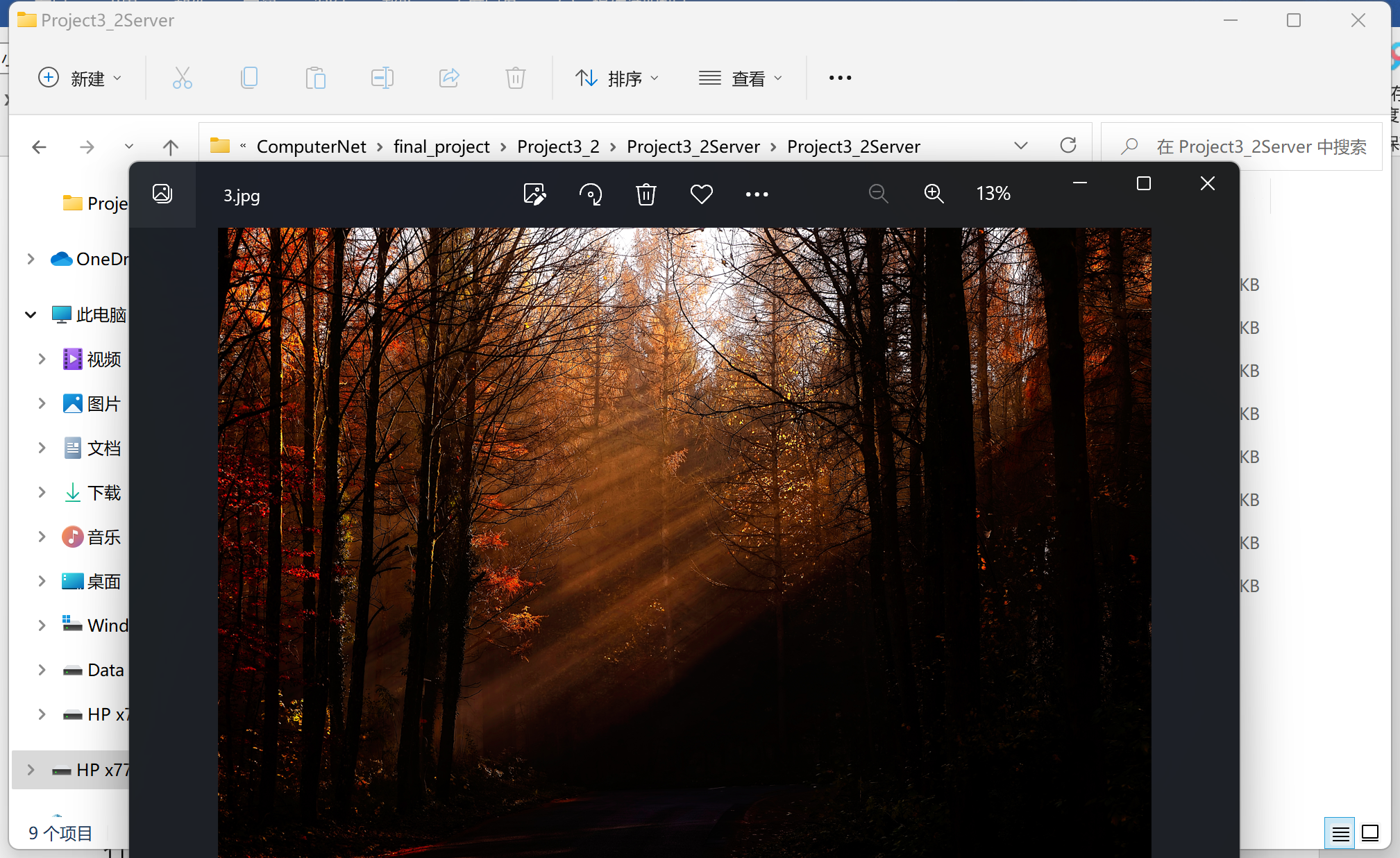
发送3.jpg结果如下图，发送时间为52.342s，平均吞吐率为1.82935e+06bps：



查看服务器端是否正确接收文件，结果如下图：



打开图片三结果如下图，显示传输结果无误：



【性能对比】

|  |  |  |
| --- | --- | --- |
| 传输时间（s）/吞吐率（bps） | 3-1 | 3-2 |
| Helloworld.txt | 21.585s/6.13688e+05bps | 11.839s/1.1188e+06bps |
| 1.jpg | 23.059s/6.44383e+05bps | 11.798s/1.25944e+06bps |
| 2.jpg | 51.151s/9.22524e+05bps | 31.768s/1.485e+06bps |
| 3.jpg | 67.173s/1.42545e+06bps | 52.342s/1.82935e+06bps |

由上表的对比可以明显看出，基于滑动窗口的GBN协议传输时间远少于之前的停等机制，吞吐率也有了很大的提高，体现了基于滑动窗口的流量控制机制远远优于停等机制。