实验 3-3 基于UDP设计可靠传输协议并实现

1. 拥塞控制实现算法——new reno算法

算法原理：在已有的程序上，若要实现拥塞控制，只需要在发送端（客户端）接收ACK消息时，以及超时重传时加上窗口的变化即可。具体地：

1、慢启动阶段

（1）实现方法：

∗ 初始化拥塞窗口：cwnd=1

∗ 每接收到一个 ACK，cwnd 增 1（每个RTT，cwnd翻倍）

（2）进入时机

∗ 连接初始建立。

∗ 报文段超时未得到确认。

2、拥塞避免阶段

（1）实现方法

∗ 每个 RTT，cwnd 增加 1

（2）进入时机

∗ cwnd 增长到阈值时，慢启动阶段结束，进入拥塞避免阶段。

∗快速阶段接收到新的ACK时，且发送的消息已经全部得到确认，进入拥塞避免阶段。

3、快速恢复阶段

（1）实现方法

∗ 阈值 ssthresh 变成窗口大小 cwnd 的一半

∗ cwnd = ssthresh + 3

∗ 此后每收到一个重复的 ACK，cwnd 增加 1

（2）进入时机

∗ 收到三次重复 ACK 后，进入快速恢复阶段

1. 从快速恢复阶段到其他阶段：

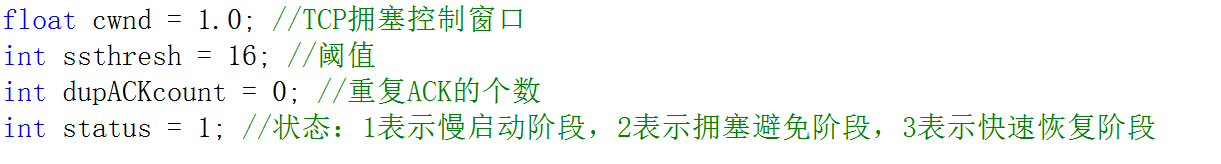
∗收到新的 ACK 后，如果发送的消息已经全部得到确认，进入拥塞避免阶段；如果没有所有发送的消息都得到确认，维持快速恢复状态

∗超时进入慢启动阶段

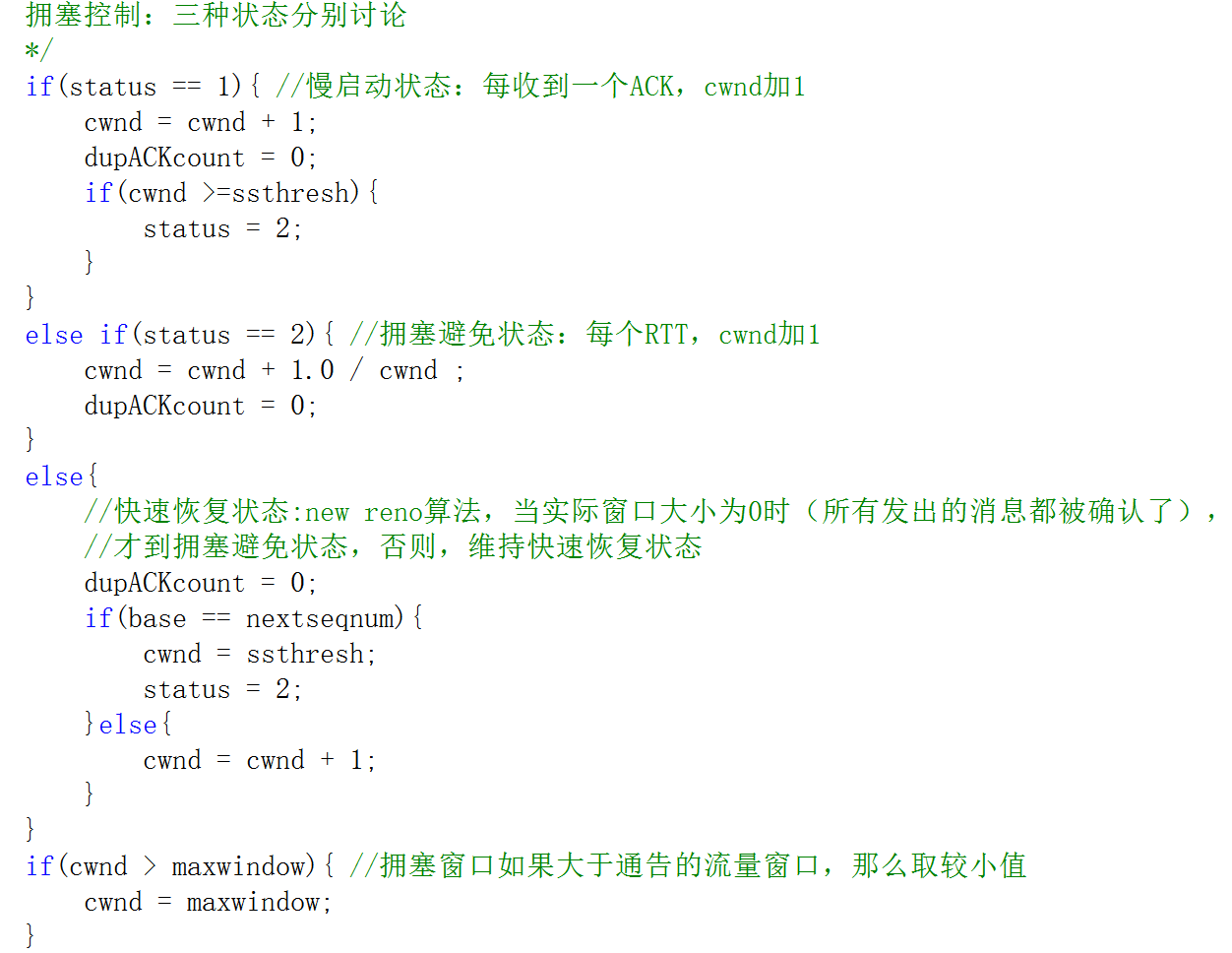
另外，除了拥塞控制，实际应用中应还有流量控制的最大窗口通告，因此，在程序中，我们设流量控制通告的最大窗口为20，如果拥塞控制增长到窗口大于20时，取二者较小值，也就是20。

1. 程序设计

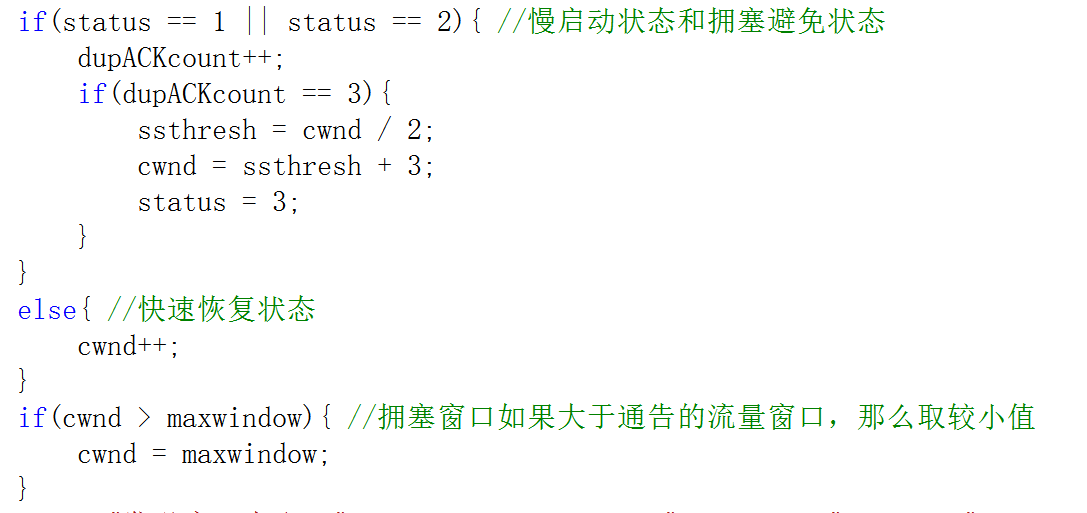
首先添加几个全局变量进行拥塞控制：



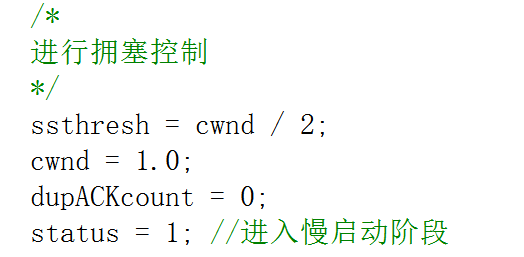
1. 当接收到新的正确的ACK时，三个阶段，对应不同的操作：



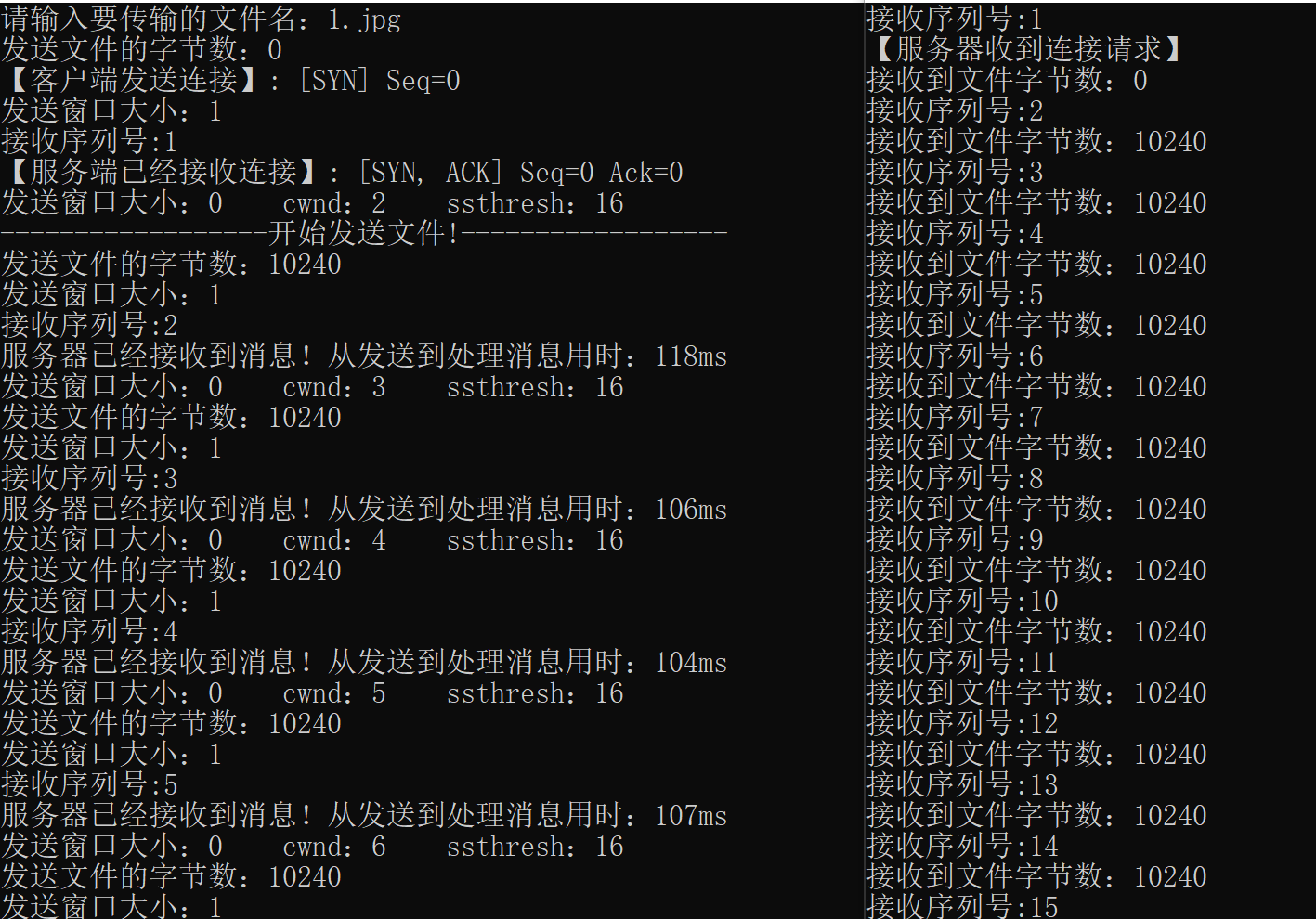
1. 收到重复的ACK时，三个阶段对应不同操作：



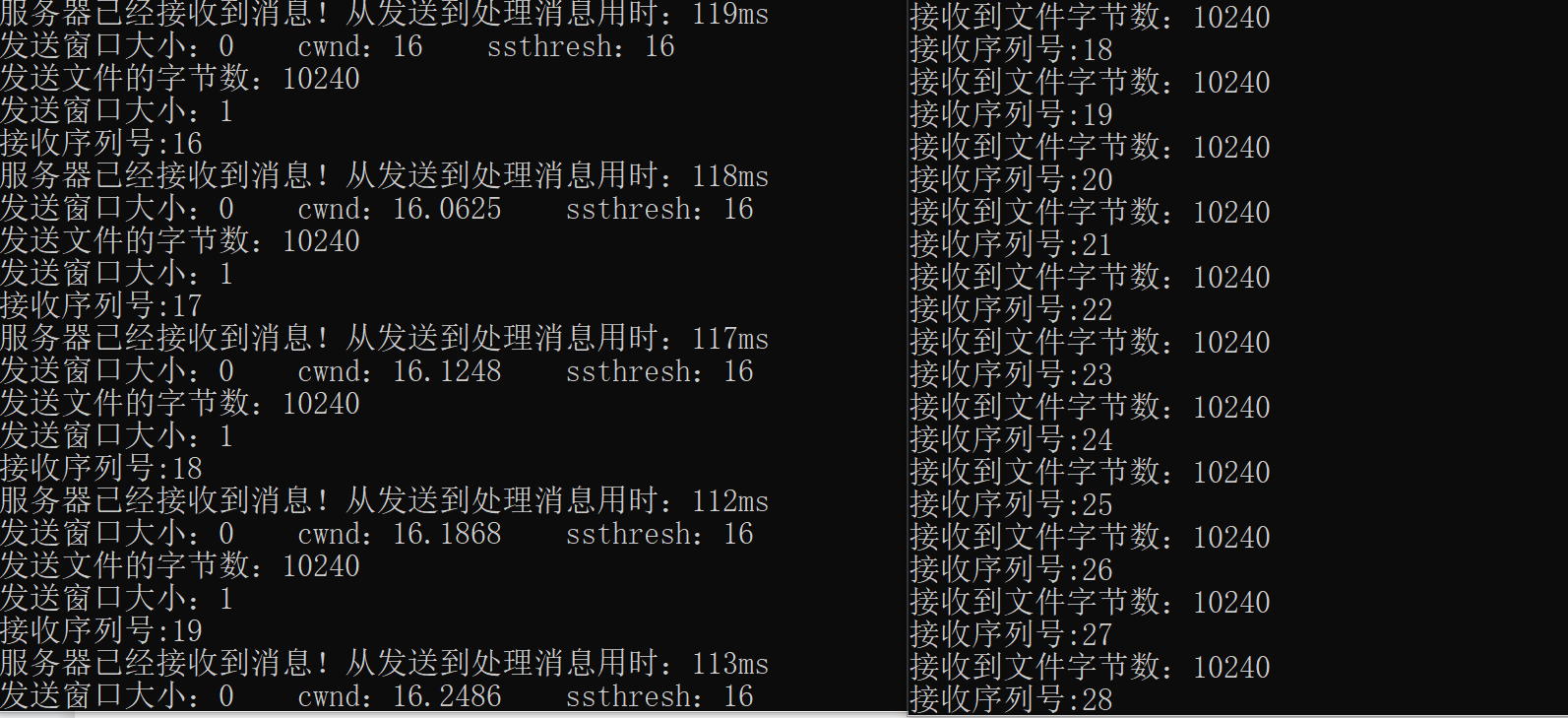
1. 超时重传时操作：



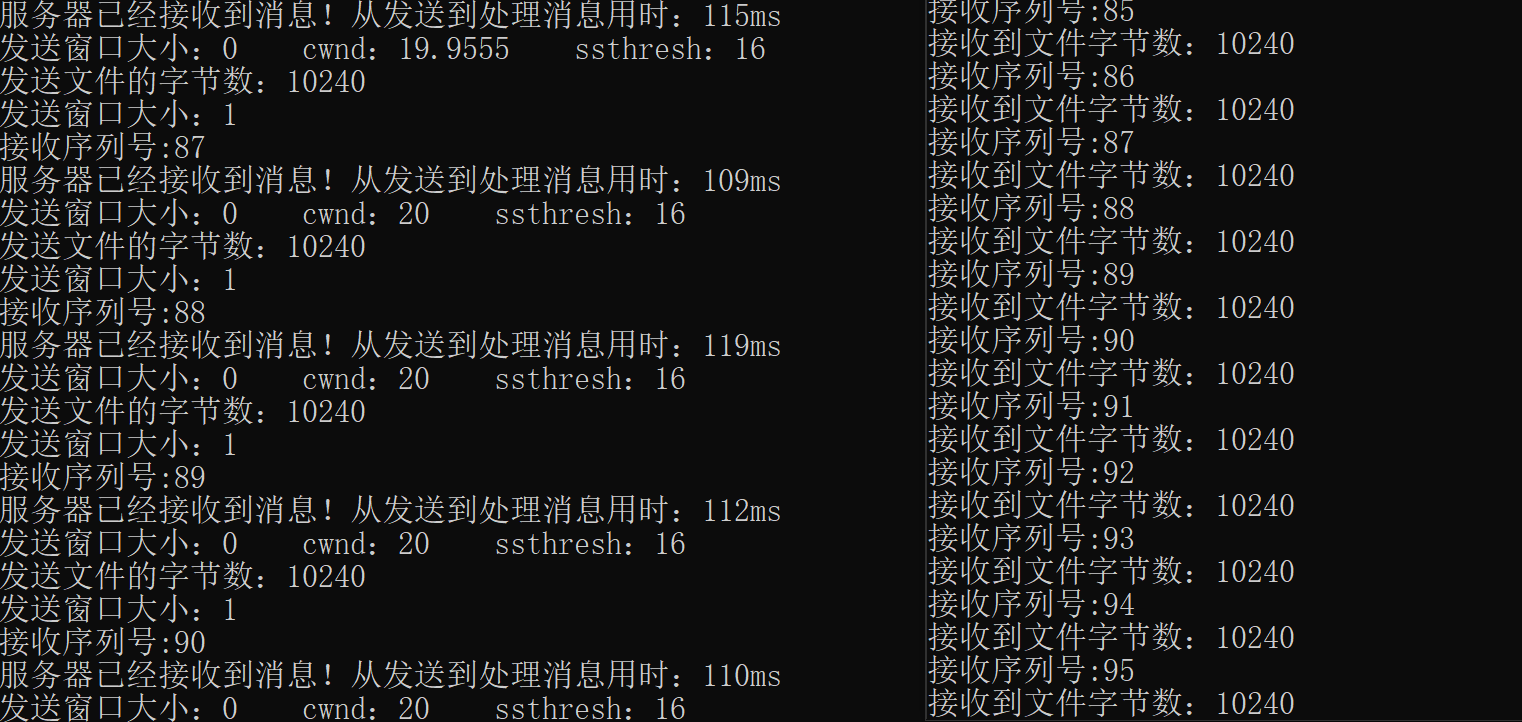
1. 实验结果
2. 刚开始慢启动阶段：左边为客户端，右边为服务器。看到cwnd每次接收完ACK以后加1，ssthresh初始为16。



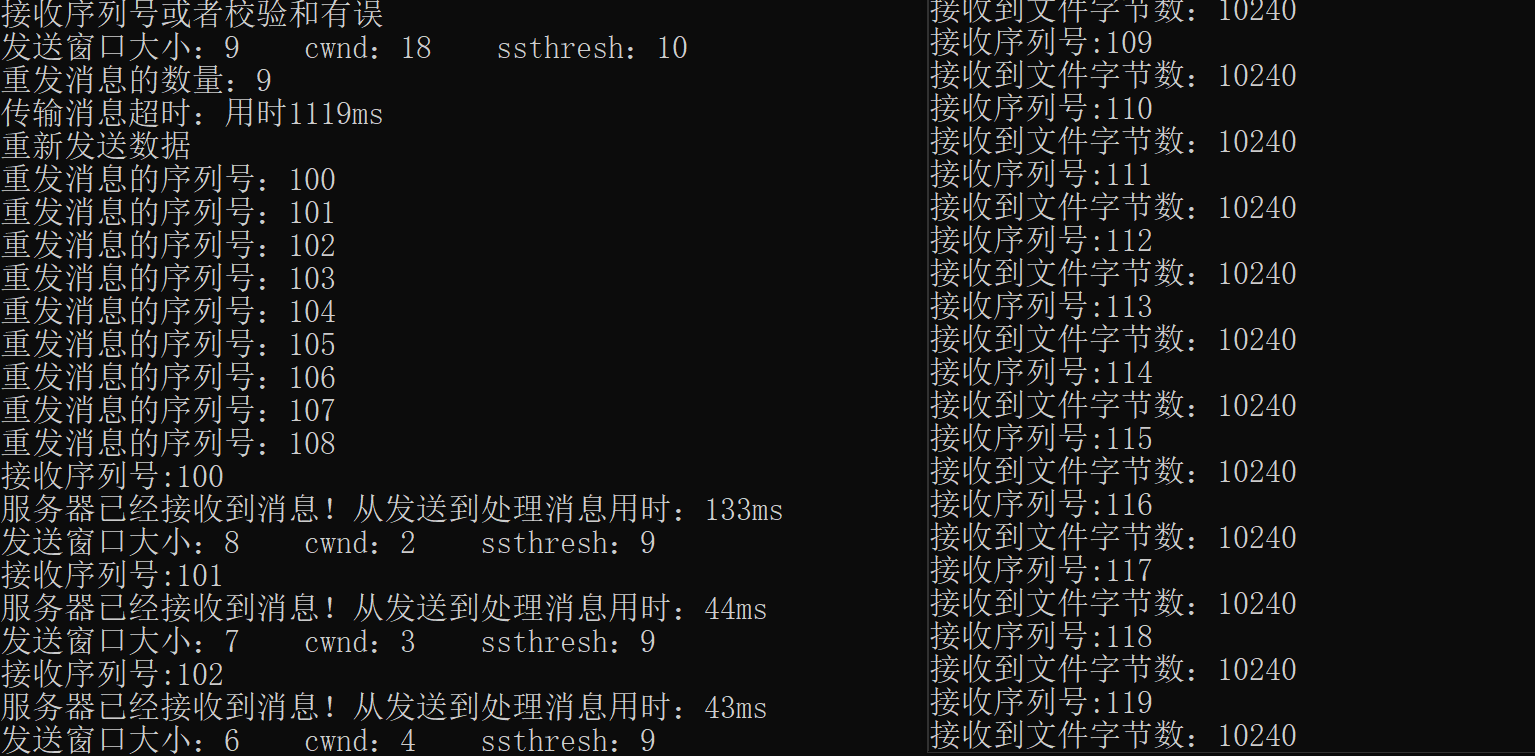
1. 拥塞避免阶段：左边客户端，右边服务器。可以看到cwnd每次不再加1，而是小数。



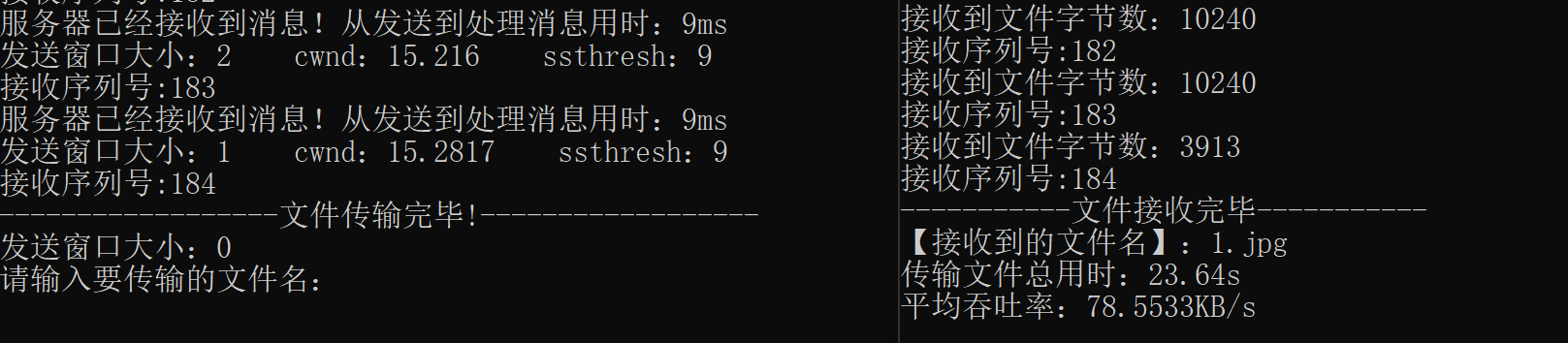
3、设置流量控制通告的窗口最大为20，因此，当拥塞控制窗口超过20时，要取较小值也就是20整并且不能再增加。



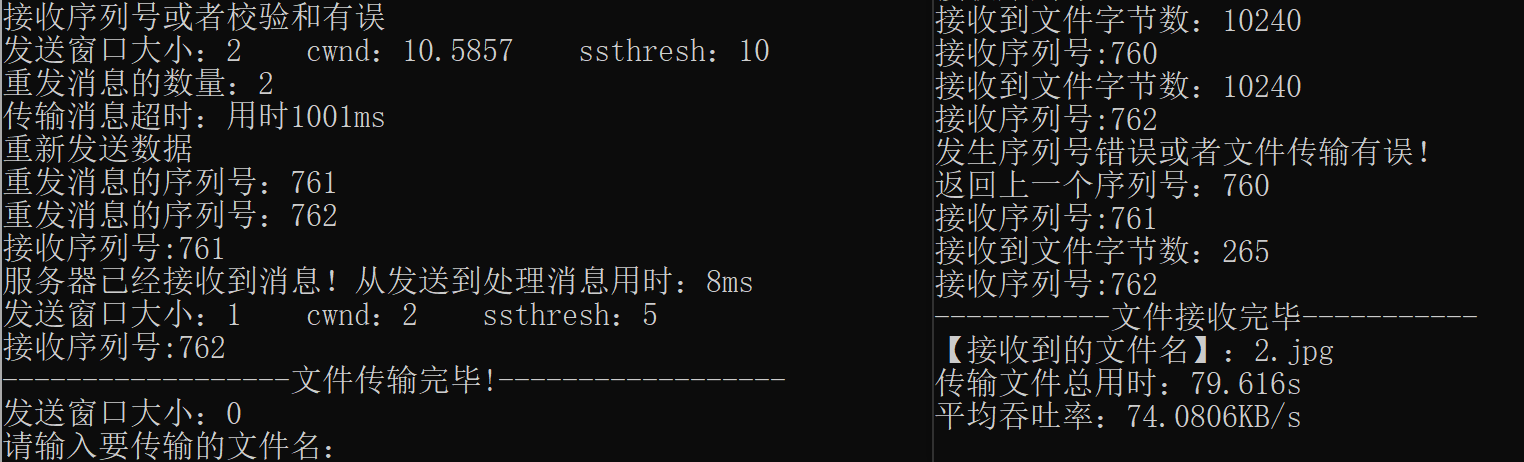
4、设置了接收端（服务器端）丢包，发送端这边显示如左图，在显示几次序号错误后，进行了超时重传，并且ssthresh变成cwnd的一半，cwnd也从1开始，然后进入了慢启动阶段。



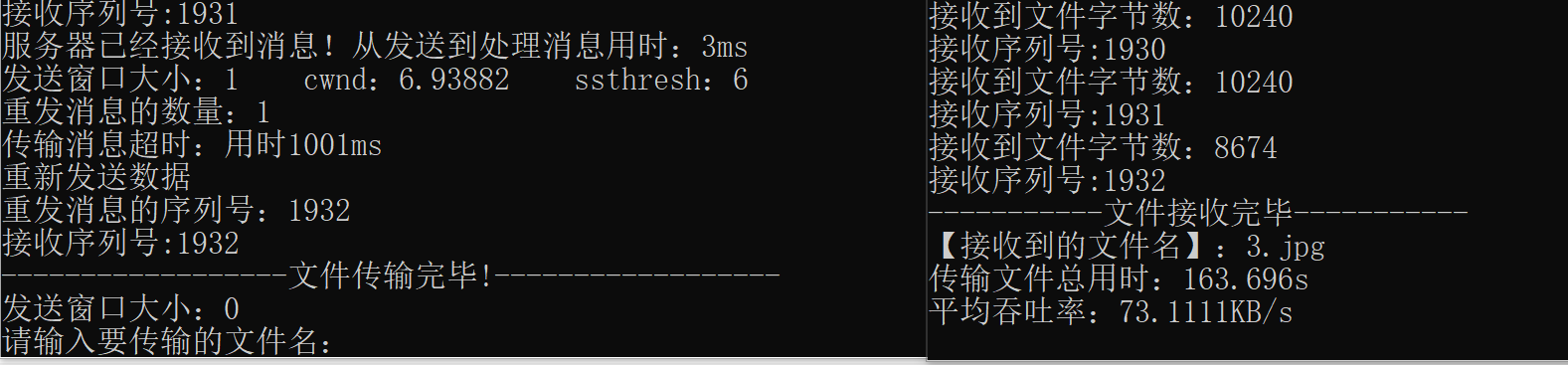
5、文件1.jpg传输完成



1. 文件2.jpg传输完成



1. 文件3.jpg传输完成



1. 文件helloworld.txt传输完成

