实验3：基于UDP服务设计可靠传输协议并编程实现

实验3-2：改进实验3-1

# 实验要求

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。 (1)实现单向传输。

(2)对于每一个任务要求给出详细的协议设计。

(3)给出实现的拥塞控制算法的原理说明。

(4)完成给定测试文件的传输，显示传输时间和平均吞吐率。

(5)性能测试指标：吞吐率、时延，给出图形结果并进行分析。

(6)完成详细的实验报告（每个任务完成一份）。

(7)编写的程序应结构清晰，具有较好的可读性。

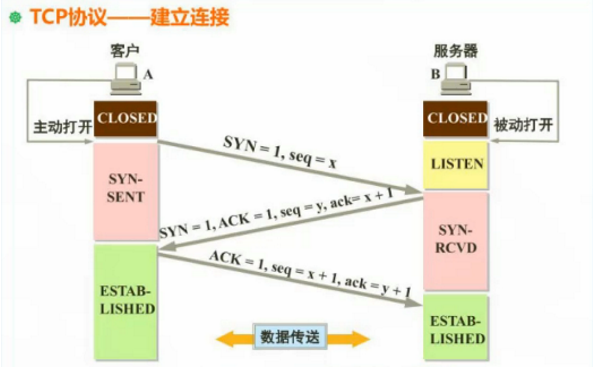
(8)提交程序源码和实验报告。

# 协议设计

## 面向连接

建立连接和断开连接的设计参考了TCP协议的三次握手和四次挥手。

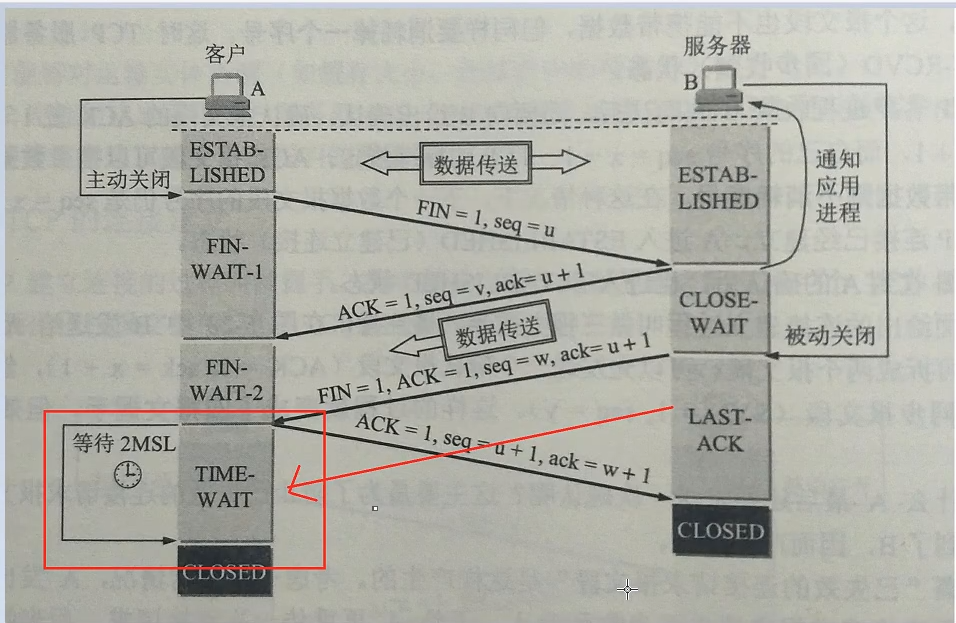
在TCP协议中，三次握手过程如下：



基于此，设计建立连接过程：

1. 首先客户端向服务器端发送一个报文，其SYN标志位置1，标志请求建立连接
2. 服务器收到请求后，向客户端回复一个报文，SYN和ACK标志位置1，标志允许建立连接
3. 客户端收到服务器反馈后，向服务器发送一个报文，ACK置1，标志可以开始传输

TCP中四次挥手过程如下：



基于此，设计断开连接过程：

①客户端向服务器端发送一个报文，将FIN标志位置1，标识文件传输完毕请求断开连接

②服务器端收到断开请求后，回应一个报文，将ACK标志位置1，标识接到断开请求

③服务器端向客户端发送一个报文，将A=FIN标志位置1，标识请求断开连接

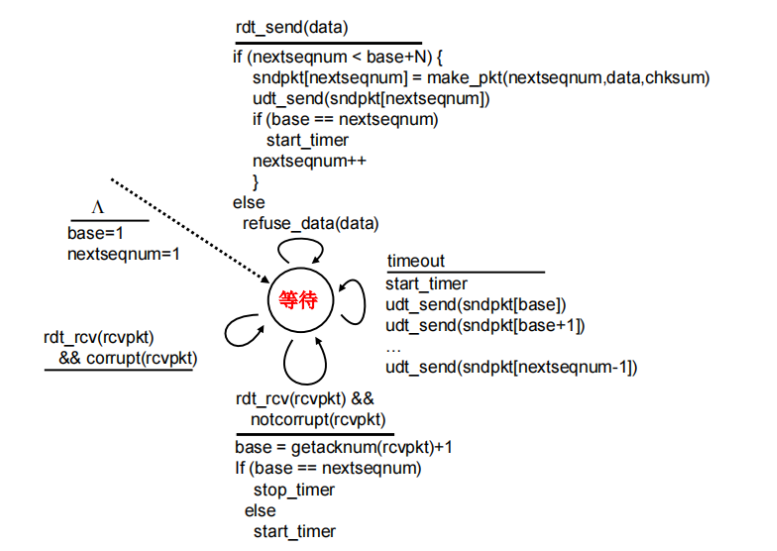
④客户端收到断开请求后，回应一个报文，将ACK标志位置1，标识接到断开请求。之后客户端在等待两个MSL时间后，确保不再接收到服务端的数据包（即服务端已经收到客户端的回应，不会再进行重传操作）后，再关闭。

## 拥塞设置：阻塞模式与非阻塞模式

由于重传机制的实现需要对数据报的发送和接收进行计时，但是初始化socket时默认是阻塞态的socket，调用recvfrom函数后线程被阻塞，计时函数也不能正常运行。如果我们在阻塞态调用recvfrom那么计时函数就需要新开一个线程，为了避免这种麻烦，我们需要计时重传的阶段调用以下代码，将socket切换为非阻塞态。

## 可靠数据传输：使用基于滑动窗口的流量控制机制的rdt3.0

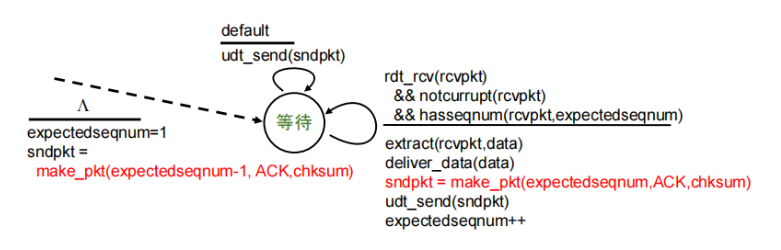
发送端有限状态状态机如图：



### 重传策略采用GBN（滑动窗口）：

1. 当滑动窗口有空闲时，持续发送数据包；
2. 发送一个数据包后，看是否收到了确认包，收到了则进行判断是否需要确认重传或者滑动窗口；
3. 判断是否有超时的情况出现，有则进行重传

接收端有限状态机如图：



### 采取累计确认：

1. 当收到正确seq的数据包，将其存储到缓存空间表示，并等待下一个数据包。
2. 当收到错误seq的数据包，则丢弃，并发送对最新seq数据包的确认包

## 超时重传

服务器端对于窗口中第一个数据包进行计时，当超时时，重发窗口中的所有数据报，并重新开始计时。

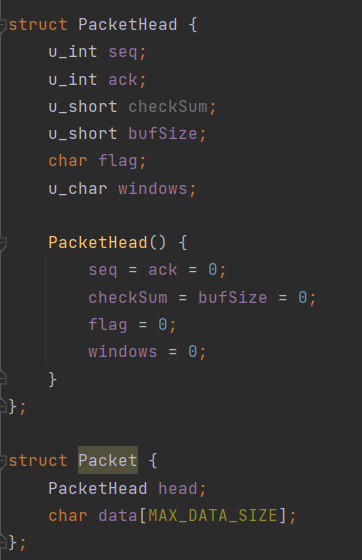
## 差错检测

模仿tcp，发送方发送报文前先计算checksum并封装到包内，接收方收到包进行校验，如果正确则正确接收，如果错误则忽略。发送方与接收方相似，但当发送方接收到错误的校验和时，会重发窗口中的所有数据包。

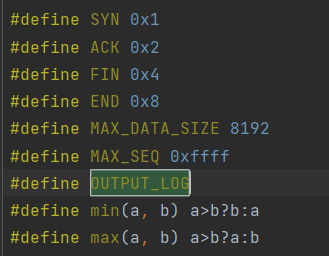
# 功能实现

## 报文结构定义和一些宏定义

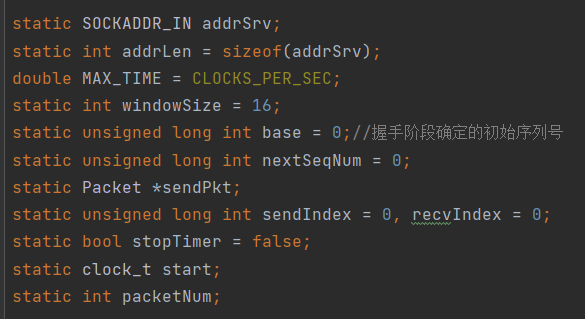
报文结构



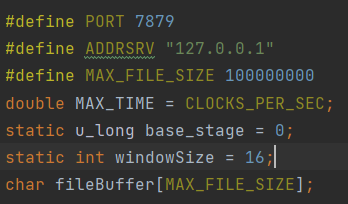
宏定义



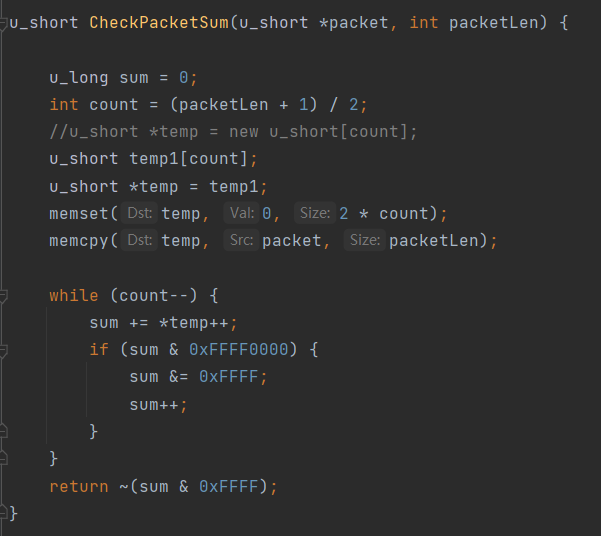
发送端：



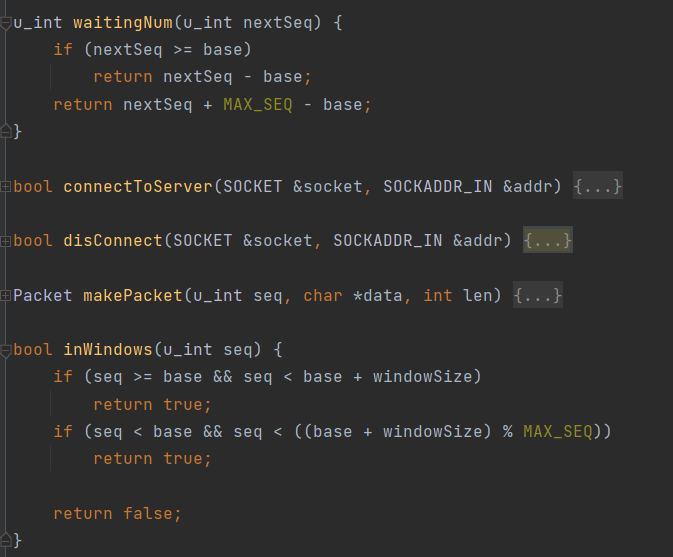
接收端：



## 计算校验和



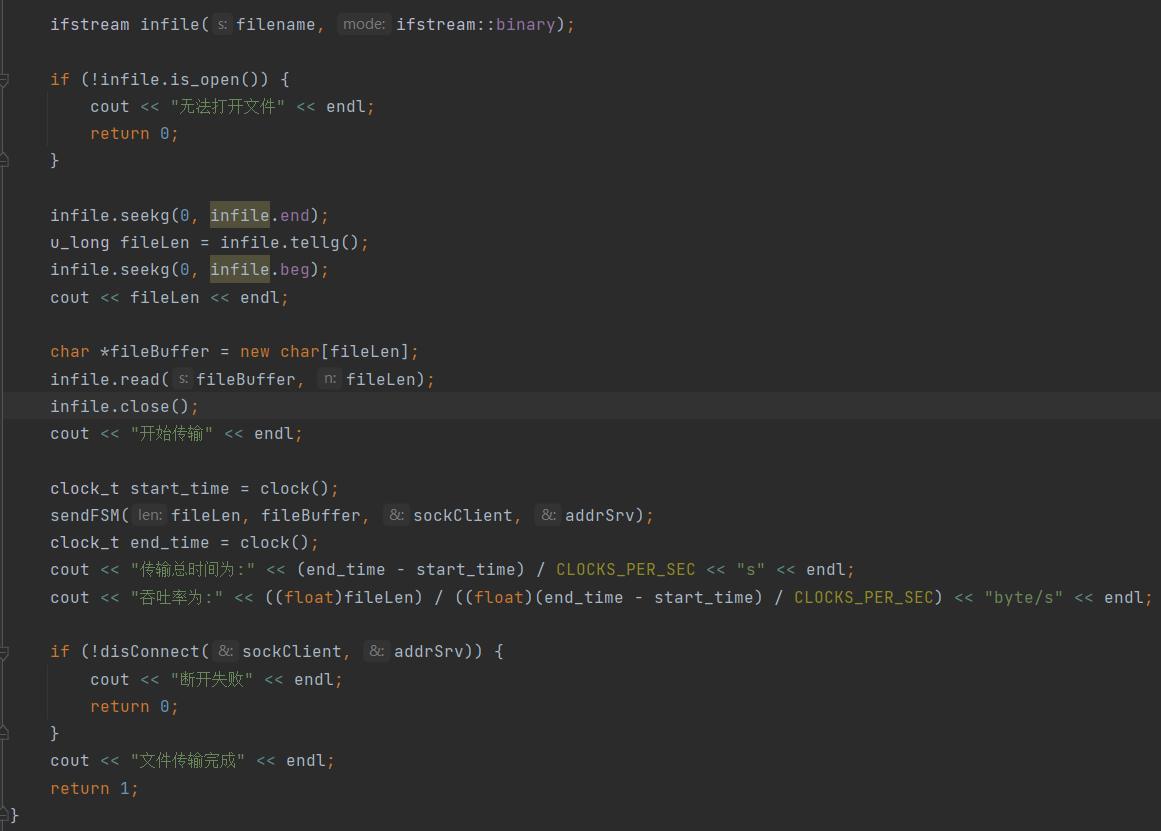
## 发送端判断是否在窗口内



## 主函数

发送端：





接收端：





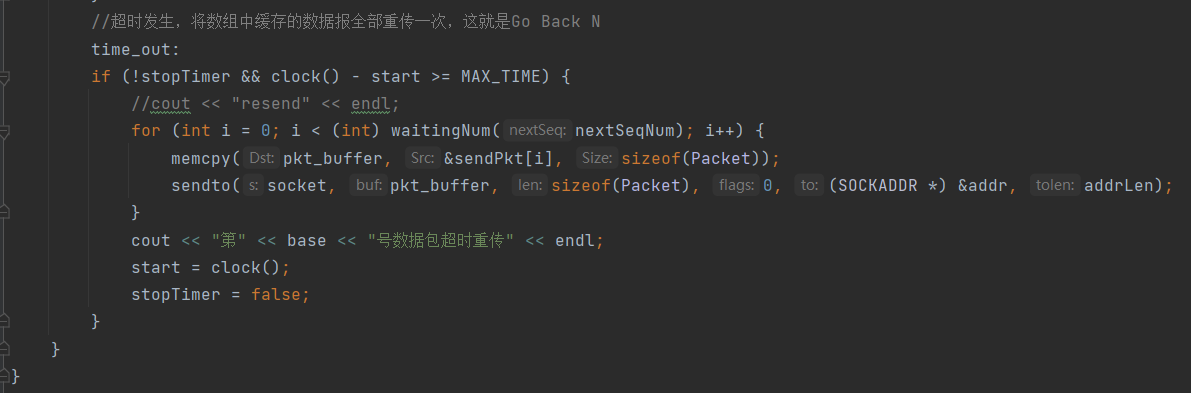
## 客户端（发送端）有限状态机





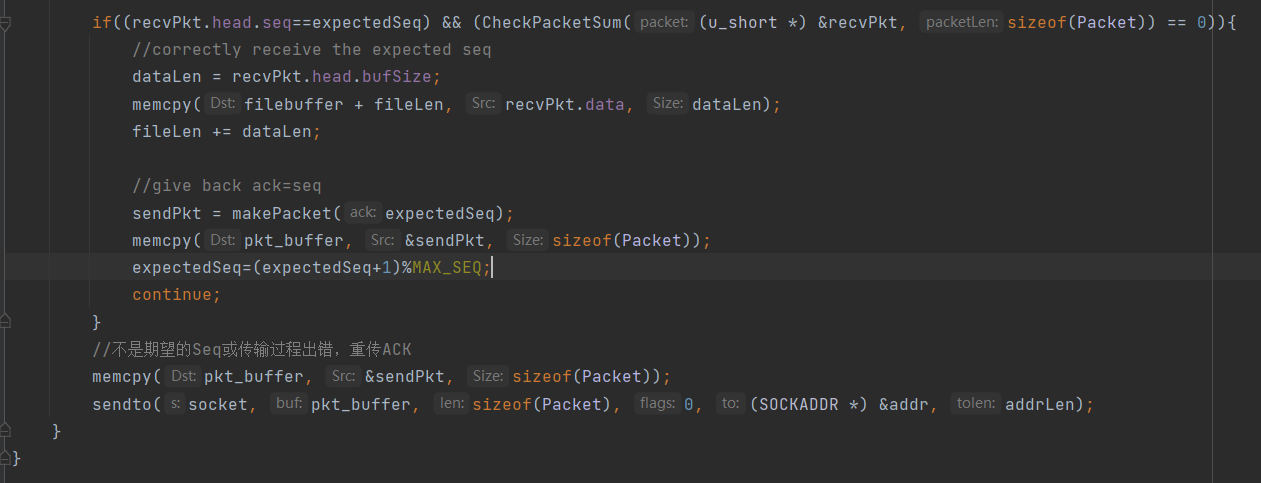






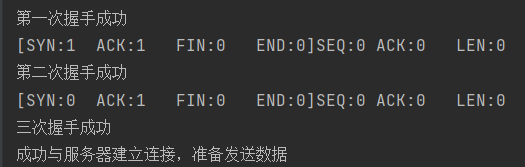
## 服务端（接收端）有限状态机

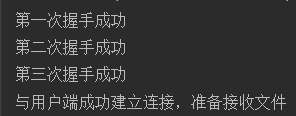




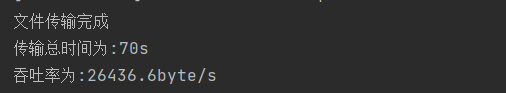
# 输出结果

建立连接

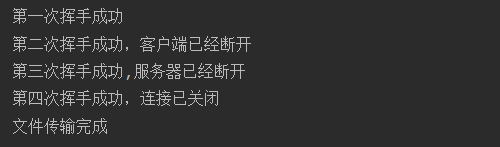


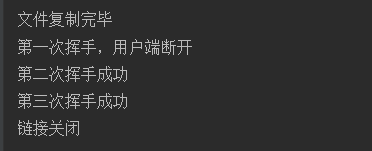


传输时间、吞吐率

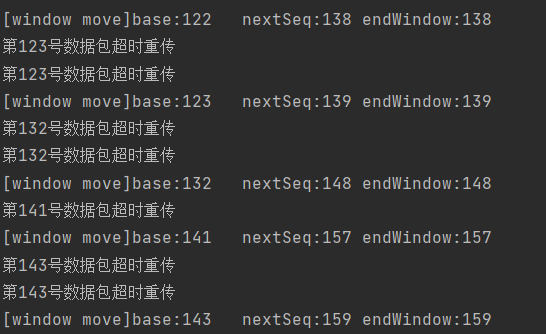


断开连接





文件传输过程部分截图（包括超时重传与窗口滑动）



# 遇到问题

Q：在clion运行项目时，选择运行时无法进行吞吐率的输出以及断开连接，但是选择调试时可以正确执行。

A：暂未解决。