



计算机网络 实验报告

基于 UDP 服务设计可靠传输协议并编程实现

实验 3-3



姓名：辛浩然

学号：2112514

年级：2021 级

学院：网络空间安全学院

班级：信息安全、法学

- 1 实验要求
- 2 协议设计
 - 2.1 数据包格式
 - 2.2 建立连接
 - 2.3 文件传输的基本交互流程
 - 2.4 可靠数据传输
 - 发送端
 - 接收端
 - 2.5 断开连接
- 3 代码实现
 - 3.1 项目文件结构
 - 3.2 发送端
 - 发送数据包
 - 接收ACK
 - 计时与超时重传
 - 3.3 接收端
- 4 文件传输结果分析
- 5 所有文件传输测试

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，**发送窗口和接收窗口采用相同大小**，支持**选择确认**，完成给定测试文件的传输。

- 单向传输：一端发数据，一端返回确认；
- 协议设计：数据包格式，发送端和接收端交互；
- **流水线协议**：多个序列号；
- **选择确认**：SR(Selective Repeat)；
- 建立连接、断开连接：类似TCP的握手、挥手功能；
- 差错检验：校验和；
- 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；

2 协议设计

2.1 数据包格式

本次实验中，传输数据包包括首部和数据两部分，总体格式如下图所示与前面实验一致：

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
发送序号 Sequence Number																															
确认序号 Acknowledge Number																															
校验和 Check Sum																数据部分长度 Data Length															
标记位 Flags												S	A	F	数据 Data																
数据 Data																															

2.2 建立连接

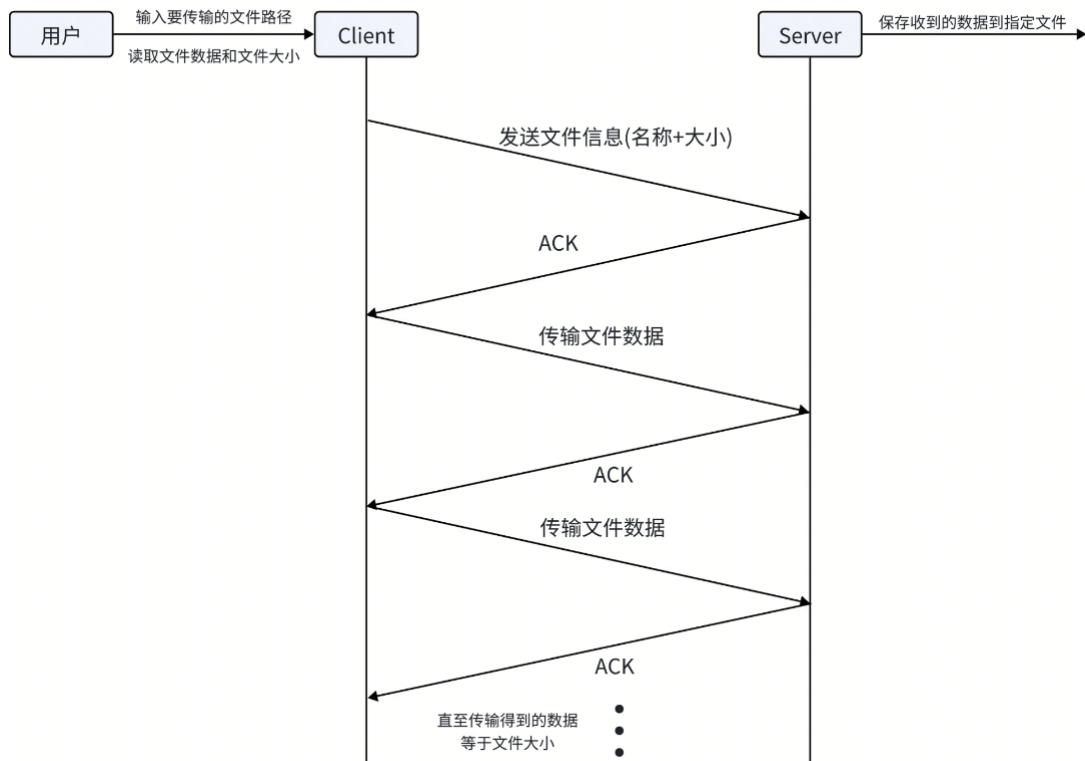
仿照TCP的三次握手，考虑到数据单向传输，因此建立连接只需要两次握手。

基本流程为：

- 第一次握手：客户端生成 **SYN** 报文，向服务器声明客户端的初始发送序号，发送至服务器，等待服务器的 **ACK** 报文；
- 第二次握手：服务器收到来自客户端的 **SYN** 报文后，判断标志位和校验和，回复 **ACK** 报文。
- 客户端超时未收到服务器的 **ACK** 报文，重传 **SYN** 报文。

2.3 文件传输的基本交互流程

本次实验中，建立连接后，客户端作为发送端，服务器作为接收端，发送文件数据的**基本交互流程**如下图所示：



如果用户在输入传输文件路径时输入 `exit`，则客户端主动与服务器断开连接。

2.4 可靠数据传输

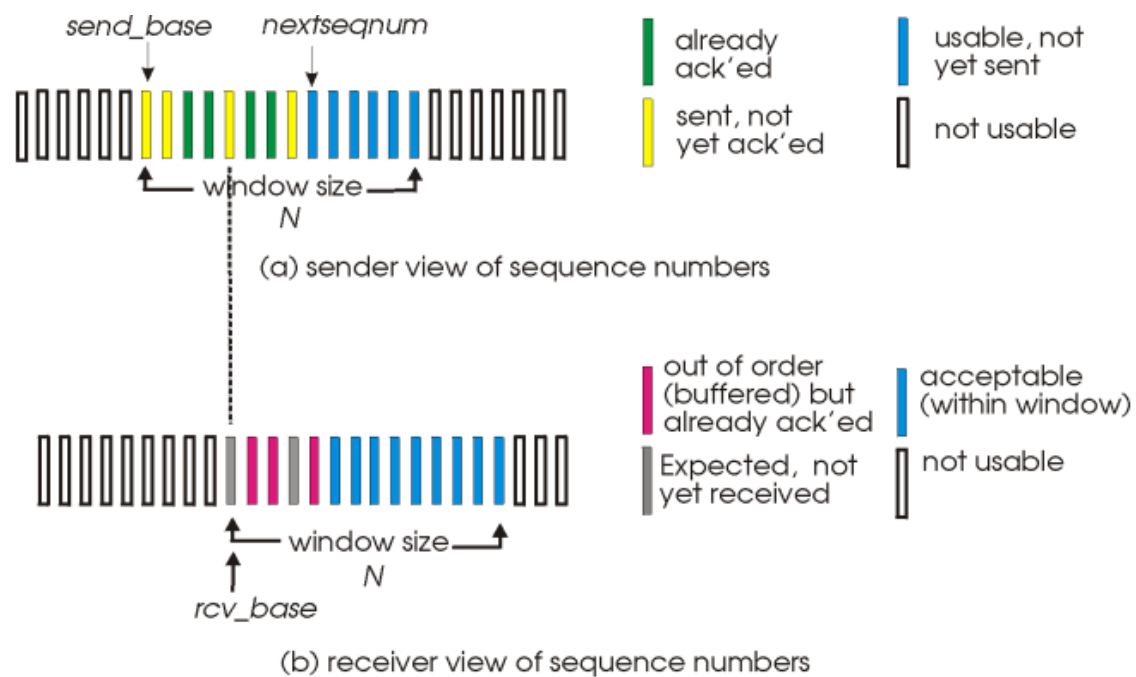
使用流水线可靠数据传输协议，采用基于滑动窗口的流量控制机制，支持选择确认，发送端选择重传丢失数据包。

对于发送端的发送的数据包，每个数据包分配递增且不同的序号。

允许发送方发送多个分组(当有多个分组可用时)而不需等待确认，但在流水线中未确认的分组数不能超过某个最大允许数 N ，即窗口大小。

接收方将确认一个正确接收的分组而不管其是否按序。失序的分组将被缓存直到所有丢失分组(即序号更小的分组)皆被收到为止，这时才可以将一批分组按序交付给上层。

发送端和接收端都维护一个窗口，且二者窗口具有相同大小。



对于发送端而言：

- 窗口的左边界(图中的`sendBase`)代表最早发送但未被确认的序号，窗口右边界(图中的`nextSeq`)代表最小的未使用序号(即下一个待发送数据包的序号)。左右边界之间的部分应当小于等于窗口大小。
- 为窗口内的序号增加确认标记：窗口内的接收到ACK的序号标记为已确认。

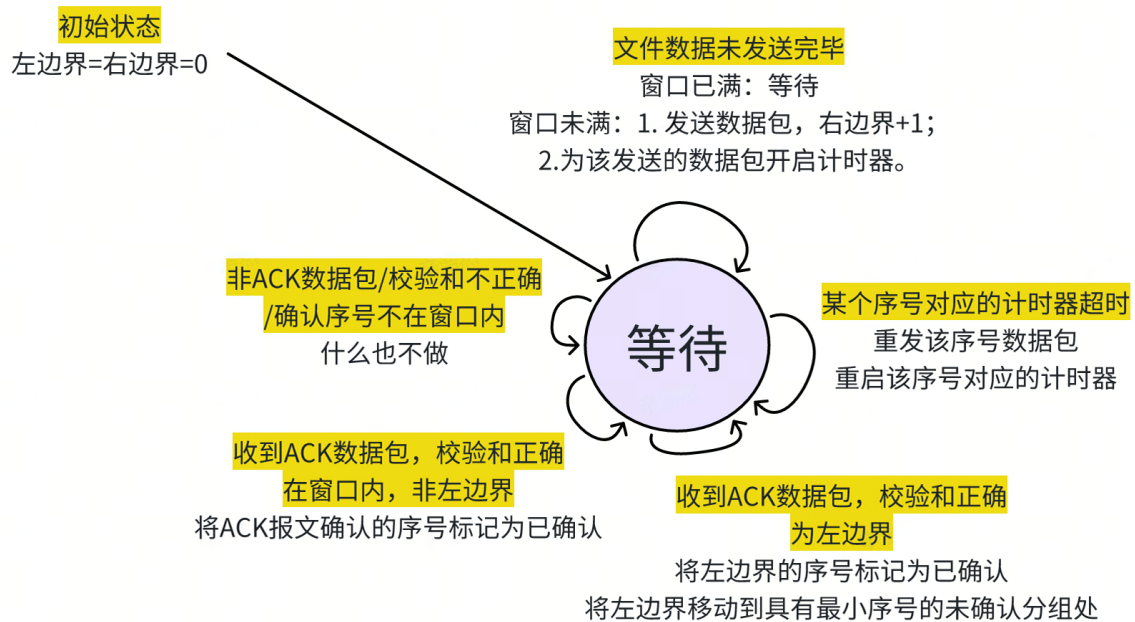
对于接收端而言：

- 窗口的左边界(图中的`recvBase`)代表期望收到但没有收到的序号，窗口内的序号是可被接收的序号；
- 为窗口内的序号增加缓存标记：如果是序号在窗口内且是失序收到的，标记为缓存。

接下来，具体分析发送端与接收端的事件与动作。

▲ 发送端

发送端状态机如下图所示：



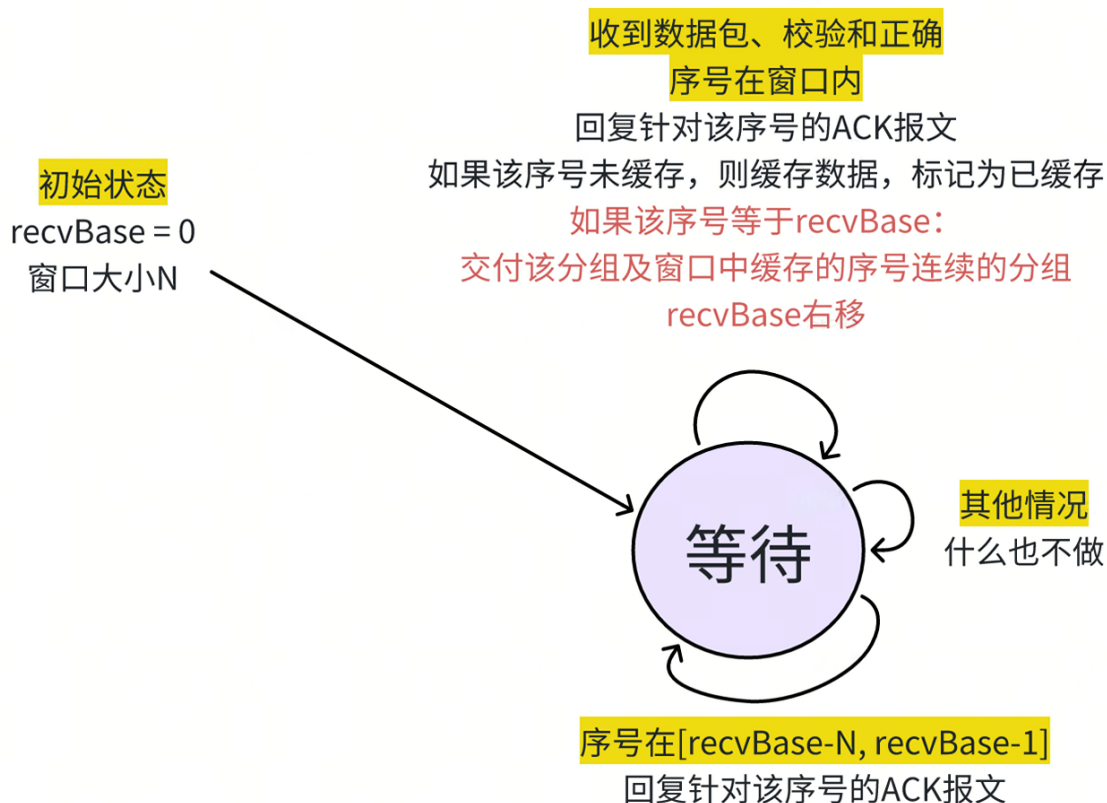
从上层收到数据: 当发送方从上层接收到数据时, 首先检查当前窗口大小。如果当前窗口不满, 就发送右边界(nextSeq)对应的数据包, 右边界右移。为该序号开启计时器。如果窗口已满, 那么稍后传输。

收到ACK: 如果发送方收到了ACK, 它首先检查ACK对应的分组的序号是否在窗口内。如果在窗口内, 说明这个分组已经被接收, 于是将该分组标记为已确认, 关闭该序号的计时器。如果ACK对应的分组的序号等于窗口的左边界, 则窗口左边界向前移动到具有最小序号的未确认分组处。

超时: 超时机制用于防止分组丢失。每个分组都有自己的逻辑定时器。如果定时器超时, 说明相应的分组可能丢失, 因此重新发送这个分组并重启对应的计时器。

▲ 接收端

接收端状态机如下图所示:



序号在 $[\text{recvBase}, \text{recvBase}+N-1]$ 内的分组被正确接收：

- 如果接收方收到的分组的序号位于接收窗口 $[\text{recvBase}, \text{recvBase}+N-1]$ 内，表示这个分组在接收窗口内，可以被正确接收。
- 在这种情况下，接收方向发送方发送一个针对该序号的选择性的ACK。这个ACK通知发送方已经成功接收了这个分组。
- 如果这个分组之前没有收到过，那么接收方缓存这个分组。
- 如果这个分组的序号等于接收窗口的基序号 recvBase ，那么说明这个分组以及接收窗口中缓存的序号连续的分组可以被交付给上层协议。接收窗口随之向前移动，以准备接收下一个序号的分组。

序号在 $[\text{recvBase}-N, \text{recvBase}-1]$ 内的分组被正确收到：

- 即使分组的序号在接收窗口之前的范围 $[\text{recvBase}-N, \text{recvBase}-1]$ 内，接收方也需要产生一个ACK，即使这个分组之前已经确认过。
- 这是因为发送方和接收方的窗口并不总是一致。如果不这样做，在ACK丢失情况下，可能会出现发送端窗口永远不能向前滑动的情况。

其他情况：如果分组的序号不在上述两个范围内，接收方会忽略该分组。

2.5 断开连接

在获取传输文件路径时，如果用户输入 `exit`，客户端主动发起断开连接。

- 客户端发送第一次挥手 `FIN + ACK` 报文。客户端发送后，等待服务器发来的第二次挥手，如果超时未收到则重发；
- 服务器收到第二次挥手后，回复 `ACK` 报文，进入等待期，等待一段时间再退出。在这段时间里，如果收到来自客户端的第一次挥手，就回复`ACK`报文。

3 代码实现

数据包定义、计算校验和、套接字初始化、建立与断开连接、读取文件数据和保存文件的代码实现与实验3-1基本一致，本次报告就不再赘述。

模拟丢包和延时、日志输出线程的实现在实验3-2中也已详细说明，此次报告也不再赘述。

接下来重点关注文件传输协议的实现。

3.1 项目文件结构

```
| - Makefile : Windows 平台下使用 mingw32-make 执行
| - include
|   | - msg.h: 数据包结构体及接口
|   | - socket.h: 建立udp连接
|   | - router.h: 模拟丢包与延时
|   | - log.h: 写日志文件与输出
| - client.cpp: 发送端
| - server.cpp: 接收端
```

3.2 发送端

左边界 `leftWin`、右边界 `rightWin` 初始为0。如前所述，窗口的左边界代表最早发送但未被确认的序号，窗口右边界代表最小的未使用序号(即下一个待发送数据包的序号)。左右边界之间的部分应当小于等于窗口大小。

`bool` 数组 `isFinished`，用于标记某序号对应的数据包是否已经确认；

`HANDLE *` 类型数组 `timers`，存放某序号对应的线程句柄。

▲ 发送数据包

发送方发送数据包时，开启循环，每次循环首先检查发送窗口是否已满和是否已经发送完毕。

- 如果**窗口未满足且未发送完毕**，则发送序号为当前右边界 `rightWin` 的数据包，发送的数据包就是右边界对应的分组。
 - 为该序号创建线程，调用计时函数作为线程函数。
 - **更新窗口右边界**。
- 若条件不满足，不做发送处理，等待至条件满足；
- 如果所有包都已发送且被确认，结束循环，终止计时器，结束其他线程。

下面是核心代码：

```
while (1)
{
    // 窗口未满足，未发送完，可以发送
    if (rightWin - leftWin < MAXWIN && rightWin < rounds)
    {
        sendFilePackage(rightWin);
        timers[rightWin] = CreateThread(nullptr, 0,
reTransmitFileThreadFunction, reinterpret_cast<LPVOID>(rightWin), 0,
nullptr);
        rightWin++; // 发送后，右窗口右移
        logger.log("[Window][AFTER SENT] LEFT: %d, RIGHT: %d", leftWin +
beforeSendNum, rightWin + beforeSendNum);
    }
    if (leftWin == rounds) // 当所有包都发送结束且确认结束后，结束循环
    {
        break;
    }
}
```

▲ 接收ACK

接收ACK线程。

如果发送方收到了ACK，首先检查ACK对应的分组的序号是否在窗口内。

- 如果在窗口内，说明这个分组已经被接收，于是将该分组标记为已确认。
- 如果ACK对应的分组的序号等于窗口的左边界，则窗口左边界向前移动到具有最小序号的未确认分组处。

下面是核心代码：

```

while (1)
{
    int bytesRecv = recvfrom(clientSock, recvFileBuf, sizeof(recvFileBuf),
0, (SOCKADDR *)&serverAddr, &sockaddrSize);

    if (bytesRecv != SOCKET_ERROR)
    {
        recvMsg->reset();
        memcpy(recvMsg, recvFileBuf, bytesRecv);
        // 判断标志位 ACK 和校验和

        if (isValidACK(recvMsg))
        {
            mtx.lock();
            if (recvMsg->header.getAckNum() <= rightWin + beforeSendNum)
            {
                isFinished[recvMsg->header.getAckNum() - beforeSendNum - 1]
= true;

                if (leftWin == recvMsg->header.getAckNum() - beforeSendNum
- 1)
                {
                    for (leftWin; leftWin < rounds; leftWin++)
                    {
                        if (!isFinished[leftWin])
                        {
                            break;
                        }
                    }
                }
                logMsg(recvMsg);
                logger.log("[Window][AFTER RECV] LEFT: %d, RIGHT: %d",
leftWin + beforeSendNum, rightWin + beforeSendNum);
            }
            mtx.unlock();
        }
    }
}

```

计时与超时重传

每个序号数据包对应的线程函数。

前面提到，在发送完该序号后，创建线程，调用线程函数，开始计时。

在该序号一直没有被确认的情况下：

- 如果超过超时时间，则重发该序号对应的数据包，重启计时器；

- 直到序号被确认，结束线程函数。

下面是核心代码：

```
DWORD WINAPI reTransmitFileThreadFunction(LPVOID lpParam)
{
    intptr_t pkg = reinterpret_cast<intptr_t>(lpParam);
    clock_t start = clock();
    while (!isFinished[pkg])
    {
        if (static_cast<double>(clock() - start) / CLOCKS_PER_SEC >=
timeOut)
        {
            logger.log("retransmit %d", pkg + beforeSendNum);
            sendFilePackage(pkg);
            start = clock();
        }
    }
    return 0;
}
```

3.3 接收端

接收端维护以下变量：

- `recvBase`：左边界；
- `MAXWIN`：窗口大小；
- `bool` 数组 `isRecv`：每个序号对应的数据包是否收到；
- `map<int, char[MSS + 1]>` 类型的 `fileBuf`：用于缓存数据包，序号到数据的映射；
- `map<int, int>` 类型的 `fileBufSize`：用于序号到数据长度的映射。

序号在 `[recvBase, recvBase+N-1]` 内的分组被正确接收：

- 发送一个针对该序号的选择性的ACK。
- 如果这个分组之前没有收到过，那么接收方缓存这个分组，将对应的序号标记为已收到。
- 如果这个分组的序号等于接收窗口的基序号 `recvBase`，那么：
 - 将这个分组以及接收窗口中缓存的序号连续的分组中的数据保存到文件中；
 - 接收窗口越过这些连续收到的序号向前移动。

序号在 `[recvBase-N, recvBase-1]` 内的分组被正确收到：

- 发送一个针对该序号的选择性的ACK。

下面是核心代码：

```
if (recvMsg->isValid())
{
    if (recvMsg->header.getSeqNum() < recvBase && recvMsg->header.getSeqNum() >= recvBase - MAXWIN)
    {
        logger.log("Unordered");
        sendflags(ACK, initialSeqNum, recvMsg->header.getSeqNum() + 1); // 发送ACK
    }
    // 收到的分组落在接收方窗口内：回复ACK，如果没收到过该分组，缓存该分组；
    else if (recvMsg->header.getSeqNum() >= recvBase && recvMsg->header.getSeqNum() < recvBase + MAXWIN)
    {
        sendflags(ACK, initialSeqNum, recvMsg->header.getSeqNum() + 1);
        // 第一次收到，放入缓存区
        if (!isRecv[recvMsg->header.getSeqNum() - beforeRecv])
        {
            isRecv[recvMsg->header.getSeqNum() - beforeRecv] = true;
            memcpy(fileBuf[recvMsg->header.getSeqNum() - beforeRecv],
recvMsg->data, recvMsg->header.getLength());
            fileBufSize[recvMsg->header.getSeqNum() - beforeRecv] =
recvMsg->header.getLength();
        }
        if (recvMsg->header.getSeqNum() == recvBase)
        {
            // 交付连续接收到的并收缩左边界
            int i = recvBase;
            for (i; i < rounds + beforeRecv; i++)
            {
                if (isRecv[i - beforeRecv])
                {
                    logger.log("write %d Size %d", i, fileBufSize[i -
beforeRecv]);
                    fileStream.write(fileBuf[i - beforeRecv], fileBufSize[i
- beforeRecv]);
                    remainingSize -= fileBufSize[i - beforeRecv];
                }
                else
                {
                    break;
                }
            }
            recvBase = i;
        }
    }
}
```

```

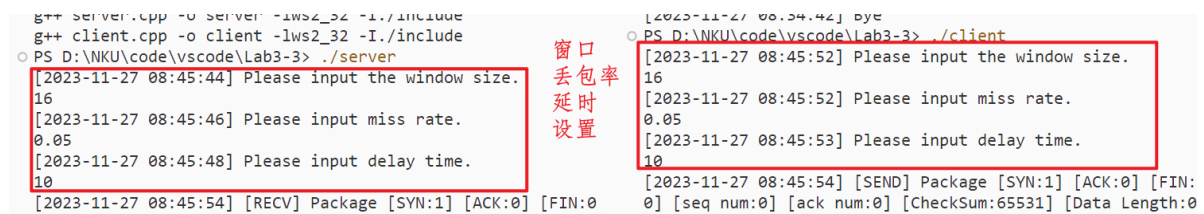
        logger.log("remainingSize %d", remainingSize);
        logger.log("rcvBase %d", rcvBase);
    }
}

```

4 文件传输结果分析

- 手动编写程序模拟丢包与延时，并且具有相应的日志输出。每次发送都调用 `sendWithRegularLoss` 函数，在该函数内先进行模拟丢包和延时的操作，只有不丢包的情况下调用真正的 `sendto` 函数。具体完整代码可参见 `router.h`。

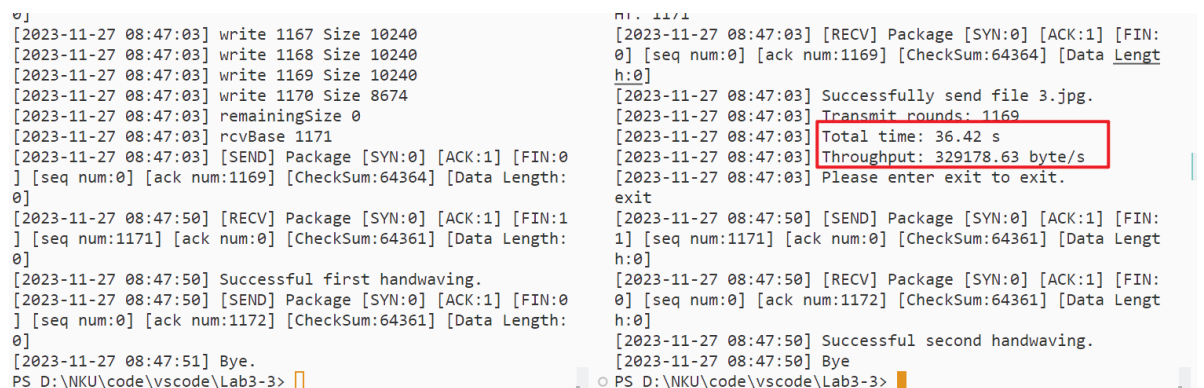
设置每个数据包大小为10240字节，发送端和接收端窗口大小为16，超时时间为500ms，丢包率为5%，延时为10ms，测试文件传输。



窗口丢包率延时设置

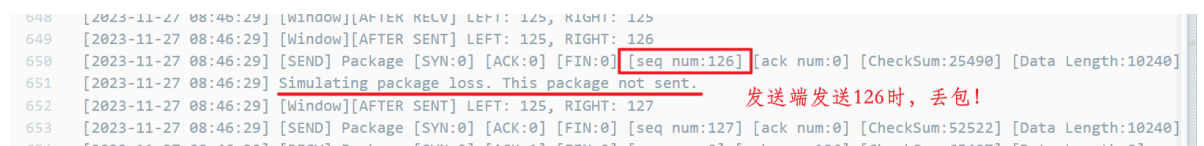
首先先针对一次测试的结果进行详细分析，在下一部分给出所有文件的测试结果。

传输3.jpg，传输时间是36.42s，吞吐率为32.917863万字节每秒。



接下来分析一下日志输出：

发送端在发送序号126的数据包时，丢包。



发送端发送126时，丢包！

接收端没有收到126的数据包，在接收127的数据包时回复针对127的ACK，但是左窗口始终是126。

```

[2023-11-27 08:46:29] rcvBase 125
[2023-11-27 08:46:29] [RECV] Package [SYN:0] [ACK:0] [FIN:0] [seq num:125] [ack num:0] [Checksum:33288] [Data Length:10240]
[2023-11-27 08:46:29] [SEND] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:126] [Checksum:65407] [Data Length:0]
[2023-11-27 08:46:29] write 125 Size 10240
[2023-11-27 08:46:29] remainingSize 10699234
[2023-11-27 08:46:29] rcvBase 126
[2023-11-27 08:46:29] [RECV] Package [SYN:0] [ACK:0] [FIN:0] [seq num:127] [ack num:0] [Checksum:52522] [Data Length:10240]
[2023-11-27 08:46:29] [SEND] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:128] [Checksum:65405] [Data Length:0]
[2023-11-27 08:46:29] remainingSize 10699234
[2023-11-27 08:46:29] rcvBase 126
[2023-11-27 08:46:29] [RECV] Package [SYN:0] [ACK:0] [FIN:0] [seq num:128] [ack num:0] [Checksum:4126] [Data Length:10240]
[2023-11-27 08:46:29] [SEND] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:129] [Checksum:65404] [Data Length:0]
[2023-11-27 08:46:29] remainingSize 10699234

```

接收端没有收到126，在接受127时，左窗口不会变化

接收端会回复针对127的ACK

而发送端没有收到针对126的ACK，在收到针对127的ACK后，左窗口边界不变。

```

[2023-11-27 08:46:29] [Window][AFTER SENT] LEFT: 125, RIGHT: 127
[2023-11-27 08:46:29] [SEND] Package [SYN:0] [ACK:0] [FIN:0] [seq num:127] [ack num:0] [Checksum:52522] [Data Length:10240]
[2023-11-27 08:46:29] [RECV] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:126] [Checksum:65407] [Data Length:0]
[2023-11-27 08:46:29] [Window][AFTER RECV] LEFT: 126, RIGHT: 127
[2023-11-27 08:46:29] [Window][AFTER SENT] LEFT: 126, RIGHT: 128
[2023-11-27 08:46:29] [SEND] Package [SYN:0] [ACK:0] [FIN:0] [seq num:128] [ack num:0] [Checksum:4126] [Data Length:10240]
[2023-11-27 08:46:29] [RECV] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:128] [Checksum:65405] [Data Length:0]
[2023-11-27 08:46:29] [Window][AFTER RECV] LEFT: 126, RIGHT: 128
[2023-11-27 08:46:29] [Window][AFTER SENT] LEFT: 126, RIGHT: 129

```

没有收到126的ACK，左窗口不变

发送端超时重传126序号的数据包，在收到ACK后，左边界越过了所有连续的已确认的序号。

```

[2023-11-27 08:46:29] [Window][AFTER RECV] LEFT: 126, RIGHT: 142
[2023-11-27 08:46:30] retransmit 126
[2023-11-27 08:46:30] [SEND] Package [SYN:0] [ACK:0] [FIN:0] [seq num:126] [ack num:0] [Checksum:25490] [Data Length:10240]
[2023-11-27 08:46:30] [RECV] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:127] [Checksum:65406] [Data Length:0]
[2023-11-27 08:46:30] [SEND] Package [SYN:0] [ACK:0] [FIN:0] [seq num:142] [ack num:0] [Checksum:29666] [Data Length:10240]
[2023-11-27 08:46:30] [Window][AFTER RECV] LEFT: 131, RIGHT: 142
[2023-11-27 08:46:30] [Window][AFTER SENT] LEFT: 131, RIGHT: 143
[2023-11-27 08:46:30] [SEND] Package [SYN:0] [ACK:0] [FIN:0] [seq num:143] [ack num:0] [Checksum:60674] [Data Length:10240]

```

超时重传126

收到ACK后，左边界越过所有连续已确认的序号

接收端收到了超时重传的126，从左边界开始的所有连续收到的包写入文件，左边界越过所有连续收到的序号。

```

[2023-11-27 08:46:29] remainingSize 10699234
[2023-11-27 08:46:29] rcvBase 126
[2023-11-27 08:46:30] [RECV] Package [SYN:0] [ACK:0] [FIN:0] [seq num:126] [ack num:0] [Checksum:25490] [Data Length:10240]
[2023-11-27 08:46:30] [SEND] Package [SYN:0] [ACK:1] [FIN:0] [seq num:0] [ack num:127] [Checksum:65406] [Data Length:0]
[2023-11-27 08:46:30] write 126 Size 10240
[2023-11-27 08:46:30] write 127 Size 10240
[2023-11-27 08:46:30] write 128 Size 10240
[2023-11-27 08:46:30] write 129 Size 10240
[2023-11-27 08:46:30] write 130 Size 10240
[2023-11-27 08:46:30] write 131 Size 10240
[2023-11-27 08:46:30] write 132 Size 10240
[2023-11-27 08:46:30] write 133 Size 10240
[2023-11-27 08:46:30] write 134 Size 10240
[2023-11-27 08:46:30] write 135 Size 10240
[2023-11-27 08:46:30] write 136 Size 10240
[2023-11-27 08:46:30] write 137 Size 10240
[2023-11-27 08:46:30] write 138 Size 10240
[2023-11-27 08:46:30] write 139 Size 10240
[2023-11-27 08:46:30] write 140 Size 10240
[2023-11-27 08:46:30] write 141 Size 10240
[2023-11-27 08:46:30] remainingSize 10535394
[2023-11-27 08:46:30] rcvBase 142

```

接收端收到了超时重传的126

从左边界开始的所有连续收到的包 写入文件
修改左边界

5 所有文件传输测试

设置每个数据包大小为10240字节，发送端和接收端窗口大小为16，超时时间为500ms，丢包率为5%，延时为10ms，测试所有文件传输。

```

PS D:\NKU\code\vscode\Lab3-3> ./server
[2023-11-27 09:13:26] Please input the window size.
10
[2023-11-27 09:13:35] Please input miss rate.
0.05
[2023-11-27 09:13:37] Please input delay time.
10
[2023-11-27 09:13:41] [RECV] Package [SYN:1] [ACK:0] [FIN:0]
[seq num:0] [ack num:0] [Checksum:65531] [Data Length:0]
[2023-11-27 09:13:41] Successful first handshake.
[2023-11-27 09:13:41] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:1] [Checksum:65532] [Data Length:0]
[2023-11-27 09:13:27] Please input miss rate.
10
[2023-11-27 09:13:30] Please input the window size.
10
[2023-11-27 09:13:40] Please input miss rate.
0.05
[2023-11-27 09:13:41] Please input delay time.
10
[2023-11-27 09:13:41] [SEND] Package [SYN:1] [ACK:0] [FIN:0]
[seq num:0] [ack num:0] [Checksum:65531] [Data Length:0]
[2023-11-27 09:13:41] [RECV] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:1] [Checksum:65532] [Data Length:0]

```

传输结果如下：

	1.jpg	2.jpg	3.jpg	helloworld.txt
传输时间 s	5.92	22.01	44.62	5.67
吞吐量 byte/s	314074	268374	268661	292538

下面是具体传输截图：

1.jpg:

```

[2023-11-27 09:14:27] [RECV] Package [SYN:0] [ACK:0] [FIN:0]
[seq num:183] [ack num:0] [Checksum:7539] [Data Length:39]
[2023-11-27 09:14:27] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:184] [Checksum:65349] [Data Length:0]
[2023-11-27 09:14:27] write 183 Size 3913
[2023-11-27 09:14:27] remainingSize 0
[2023-11-27 09:14:27] rcvBase 184
[2023-11-27 09:14:31] [RECV] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:184] [ack num:0] [Checksum:65348] [Data Length:0]
[2023-11-27 09:14:31] Successful first handwaving.
[2023-11-27 09:14:31] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:185] [Checksum:65348] [Data Length:0]
[2023-11-27 09:14:32] Bye.
PS D:\NKU\code\vscode\Lab3-3>
[2023-11-27 09:14:27] Please input miss rate.
10
[2023-11-27 09:14:27] [Window][AFTER SENT] LEFT: 183, RIGHT: 184
[2023-11-27 09:14:27] Successfully send file 1.jpg.
[2023-11-27 09:14:27] Transmit rounds: 182
[2023-11-27 09:14:27] Total time: 5.92 s
[2023-11-27 09:14:27] Throughput: 314074.79 byte/s
[2023-11-27 09:14:27] Please enter exit to exit.
exit
[2023-11-27 09:14:31] [SEND] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:184] [ack num:0] [Checksum:65348] [Data Length:0]
[2023-11-27 09:14:31] [RECV] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:185] [Checksum:65348] [Data Length:0]
[2023-11-27 09:14:31] Successful second handwaving.
[2023-11-27 09:14:31] Bye
PS D:\NKU\code\vscode\Lab3-3>

```



recv_1.jpg

位置: D:\NKU\code\vscode\Lab3-3\recv

大小: 1.77 MB (1,857,353 字节)

占用空间: 1.77 MB (1,859,584 字节)

创建时间: 2023年11月27日, 9:14:21

修改时间: 2023年11月27日, 9:14:27

访问时间: 2023年11月27日, 9:14:27

2.jpg:


```

[2023-11-27 09:16:43] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:579] [Checksum:64954] [Data Length:0]
[2023-11-27 09:16:43] write 578 Size 265
[2023-11-27 09:16:43] remainingSize 0
[2023-11-27 09:16:43] rcvBase 579
[2023-11-27 09:16:43] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:572] [Checksum:64961] [Data Length:0]
[2023-11-27 09:16:46] [RECV] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:579] [ack num:0] [Checksum:64953] [Data Length:0]
[2023-11-27 09:16:46] Successful first handwaving.
[2023-11-27 09:16:46] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:580] [Checksum:64953] [Data Length:0]
[2023-11-27 09:16:47] Bye.
PS D:\NKU\code\vscode\Lab3-3>
[2023-11-27 09:16:43] [SEND] Package [SYN:0] [ACK:0] [FIN:0]
[seq num:571] [ack num:0] [Checksum:22866] [Data Length:10240]
[2023-11-27 09:16:43] Successfully send file 2.jpg.
[2023-11-27 09:16:43] Transmit rounds: 577
[2023-11-27 09:16:43] Total time: 22.01 s
[2023-11-27 09:16:43] Throughput: 268374.91 byte/s
[2023-11-27 09:16:43] Please enter exit to exit.
exit
[2023-11-27 09:16:46] [SEND] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:579] [ack num:0] [Checksum:64953] [Data Length:0]
[2023-11-27 09:16:46] [RECV] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:580] [Checksum:64953] [Data Length:0]
[2023-11-27 09:16:46] Successful second handwaving.
[2023-11-27 09:16:46] Bye
PS D:\NKU\code\vscode\Lab3-3>

```



rcv_2.jpg

位置:	D:\NKU\code\vscode\Lab3-3\recv
大小:	5.62 MB (5,898,505 字节)
占用空间:	5.62 MB (5,902,336 字节)
创建时间:	2023年11月27日, 9:16:21
修改时间:	2023年11月27日, 9:16:43
访问时间:	2023年11月27日, 9:16:43

3.jpg:

```

[2023-11-27 09:18:14] write 1167 Size 10240
[2023-11-27 09:18:14] write 1168 Size 10240
[2023-11-27 09:18:14] write 1169 Size 10240
[2023-11-27 09:18:14] write 1170 Size 8674
[2023-11-27 09:18:14] remainingSize 0
[2023-11-27 09:18:14] rcvBase 1171
[2023-11-27 09:18:14] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:1169] [Checksum:64364] [Data Length:0]
[2023-11-27 09:18:16] [RECV] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:1171] [ack num:0] [Checksum:64361] [Data Length:0]
[2023-11-27 09:18:16] Successful first handwaving.
[2023-11-27 09:18:16] [SEND] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:1172] [Checksum:64361] [Data Length:0]
[2023-11-27 09:18:17] Bye.
PS D:\NKU\code\vscode\Lab3-3>
[2023-11-27 09:18:14] [seq num:0] [ack num:1168] [Checksum:64365] [Data Length:0]
[2023-11-27 09:18:14] [Window][AFTER RECV] LEFT: 1168, RIGHT: 1171
[2023-11-27 09:18:14] Successfully send file 3.jpg.
[2023-11-27 09:18:14] Transmit rounds: 1169
[2023-11-27 09:18:14] Total time: 44.62 s
[2023-11-27 09:18:14] Throughput: 268661.99 byte/s
[2023-11-27 09:18:14] Please enter exit to exit.
exit
[2023-11-27 09:18:16] [SEND] Package [SYN:0] [ACK:1] [FIN:1]
[seq num:1171] [ack num:0] [Checksum:64361] [Data Length:0]
[2023-11-27 09:18:16] [RECV] Package [SYN:0] [ACK:1] [FIN:0]
[seq num:0] [ack num:1172] [Checksum:64361] [Data Length:0]
[2023-11-27 09:18:16] Successful second handwaving.
[2023-11-27 09:18:16] Bye
PS D:\NKU\code\vscode\Lab3-3>

```