南開大學
Nankai University

南　开　大　学

网 络 空 间 安 全 学 院

计算机网络实验报告

---

lab3-2: **滑动窗机制** UDP **的可靠数据传输协议编程实现**

姓名: 罗功成

学号: 1910487

年级 : 2019 级

专业 : 信息安全

指导教师 : 张建忠 徐敬东

2021 年 12 月 10 日

摘要

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。

关键字：UDP，**滑动窗机制，可靠数据传输**,GBN

目录

一、 实验流程

(一) 协议设计

```
unsigned char tag;//连接建立、断开标识
unsigned int seq;//序列号
unsigned int ack;//确认号
unsigned short len;//数据部分长度
unsigned short checksum;//校验和
unsigned short window;//窗口
char data[1024];//数据长度
```

图 1: 协议设计

(1)操作标记位 tag: 相当于 TCP 协议中的 FIN 和 SYN, 根据 tag 值的不同, 确定进行不同的操作: 比如设置 tag 作为 0 时, 代表客户端发起建立连接请求, tag 为 88 时, 代表断开连接请求等。

(2) 序列号 seq: 用来标记数据包是第几个。

(3) 确认号 ack: 相当于 TCP 的 ack, 确认数据包被正确接收

(4) 数据段长度 len: 标记可以容纳多少长度的数据。

(5) 数据段 data: 设计每个数据包大小为 1024 字节

(6) 窗口 window, 用来标记滑动窗。

(二) 实验核心代码

(1) 滑动窗设计:

在开头的 WINDOWSIZE 部分规定好对应的滑动窗口大小为 20, BUFFER 设置为数据包的大小, 所以缓冲区数组 buffer[WINDOWS][BUFFER], 发送和接收端的窗口大小为 WINDOWS*BUFFER.

滑动窗口

```
1
2
3 char buffer[WINDOWSIZE][BUFFER];//选择重传缓冲区
4 #define BUFFER sizeof(packet)
5 #define WINDOWSIZE 20 //窗口可容纳的数据包大小。
6 int sendwindow = WINDOWSIZE * BUFFER;//发送窗口大小
```

(2) 累计确认: 从当前的 ack 开始, 逐个向后遍历。当现在的下标号刚好超过窗口大小, 说明前面一个窗口已经放满了, 此时循环累计一个窗口数量的 ack 数量。直到 ack 总数和之前分

包计算的数据包总数一致时停止累计。

累计确认的实现

```
//累积确认
for (int j = curack; j != (index + 1) % seqnumber; j = (++j) % seqnumber)
{
ack[j % WINDOWSIZE] = 1;
++totalack;
curack = (curack + 1) % seqnumber;
}
```

（3）GO BACK N: 对于接收方而言，只要当前的数据包和等待的序列号一致，并且当前窗口还没有达到最后的 ack 时，只需要每次和窗口里最大的 ack 进行确认即可，当最大的被确认时，这一组的数据包均被正确接收。等到接受的 ACK 总数和计算的总的数据包个数一致时，完成并退出传输文件过程，并打印对应的传输时间和吞吐率。

如果发生丢失，则会对成功接受的最大的 ack 进行确认；此后，将第一个没有被确认的作为一个新的开始向后数 20 个，然后接着重新传输。(如 1-5 成功，6 缺失，则返回对 5 的 ACK,6-25 作为下一组的 20 个进行传输)。

如果传输超时，发送端将会把没有被确认的分组整个进行重传。

Go Back N

```

接收端:
if (pkt->seq == waitseq && totalrecv < totalpacket && !corrupt(pkt))
{

cout << "收到第" << pkt->seq << "号数据包" << endl << endl;
recvwindow -= BUFFER;
out_result.write(pkt->data, pkt->len);
out_result.flush();
recvwindow += BUFFER;
make_mypkt(pkt, waitseq, recvwindow);
cout << "发送对第" << waitseq << "号数据包的确认" << endl;
sendto(socketClient, (char*)pkt, BUFFER, 0, (SOCKADDR*)&addrServer, sizeof(
    SOCKADDR));
waitseq++;
waitseq %= seqnumber;
totalrecv++;
}

发送端:
pkt->init_packet();
recvSize = recvfrom(sockServer, (char*)pkt, BUFFER, 0, ((SOCKADDR*)&
    addrClient), &length);
```

```
if (recvSize < 0)
{
waitcount++;

if (waitcount > 20)
{
        timeout();
        waitcount = 0;
}
}
else
{
ackhandler(pkt->ack);
sendwindow = pkt->window;
}
break;
```

(三) 实验效果

先打开服务器端，再打开客户端，双方建立连接完成后，开始传输对应文件。

滑动窗实现效果，在对应文件传输成功过后，将会打印本次传输的延时和吞吐率信息。

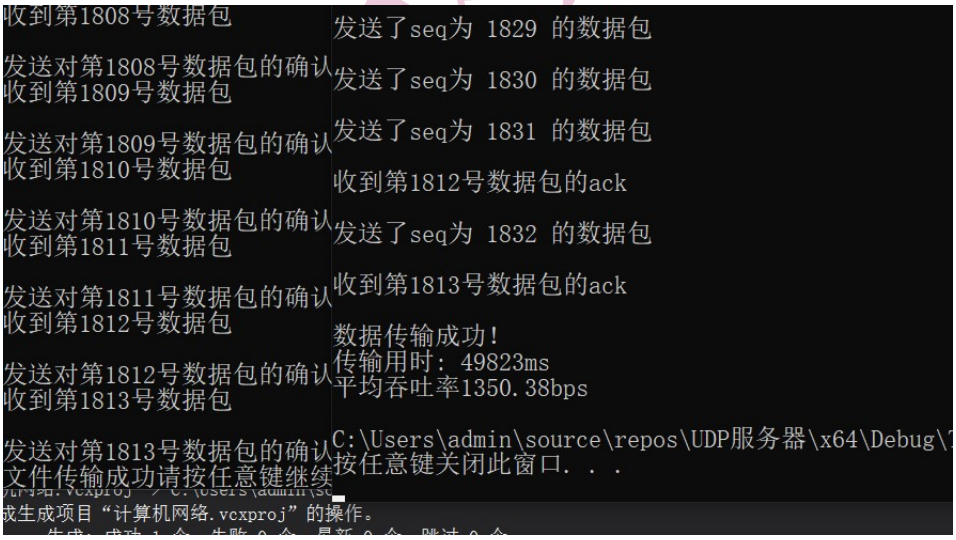

图 2: 延时和吞吐率

可以看到，向指定的文件夹输出了对应的文件。

图 3: 对应位置输出

二、 总结和收获

1. 在 3-2 的基础上实现了滑动窗机制。
2. 对 GBN 相关部分进行了知识回顾和代码实现。
3. 窗口大小应该适当，比如在这里经过实测，窗口大小设置在 10-20 左右效果比较好。窗口过小，相当于接近停等机制，并发性下降；窗口过大，近似于没有设置窗口，中间需要大量的确认交互，也会影响交互效率。