哈爾濱工業大學
Harbin Institute of Technology

计算机网络
课程实验报告

实验名称	可靠数据传输协议-GBN 协议的设计与实现				
姓名	姜思琪		院系	计算机科学与技术学院	
班级	1603109		学号	1160300814	
任课教师	李全龙		指导教师	李全龙	
实验地点	格物 213		实验时间	2018.11.3 周六 3、4 节	
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)				
教师评语					

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

实验目的：

理解滑动窗口协议的基本原理；掌握 GBN 的工作原理；掌握基于 UDP 设计并实现一个 GBN 协议的过程与技术。

实验内容：

1) 基于UDP设计一个简单的GBN协议，实现单向可靠数据传输（服务器到客户的数据传输）。
2) 模拟引入数据包的丢失，验证所设计协议的有效性。
3) 改进所设计的 GBN 协议，支持双向数据传输；（选作内容，加分项目）
4）将所设计的 GBN 协议改进为 SR 协议。（选作内容，加分项目）

实验过程：

1) 基于UDP设计一个简单的GBN协议，实现单向可靠数据传输（服务器到客户的数据传输）。

首先，需要设计传输的数据分组的格式。这样才能打包生成数据分组和解析数据分组。我采用的数据分组分为四部分：序列号、传输结束标志、数据校验和、数据。序列号，取值范围 0~255；传输结束标志，若是最后一个数据分组则为 1，不是则为 0；数据校验和，长度为 8bit；传输的数据，长度为 2048 字节。

其次，确认分组ACK的格式。分为最近一次确认的数据分组的序列号和接收端期望收到的数据分组的序列号两部分，取值均为0~255。

然后，要实现协议内容。滑动窗口大小设置为4，分为接受方和发送方两部分。

发送方：

首先将数据按定义的格式打包（make_pkt函数）。使用udp发送分组，通过利用udp_send函数。发送时，通过LOSS_RATE参数设置丢失率，用生成随机数的形式判断是否丢包，然后等待确认分组到达。

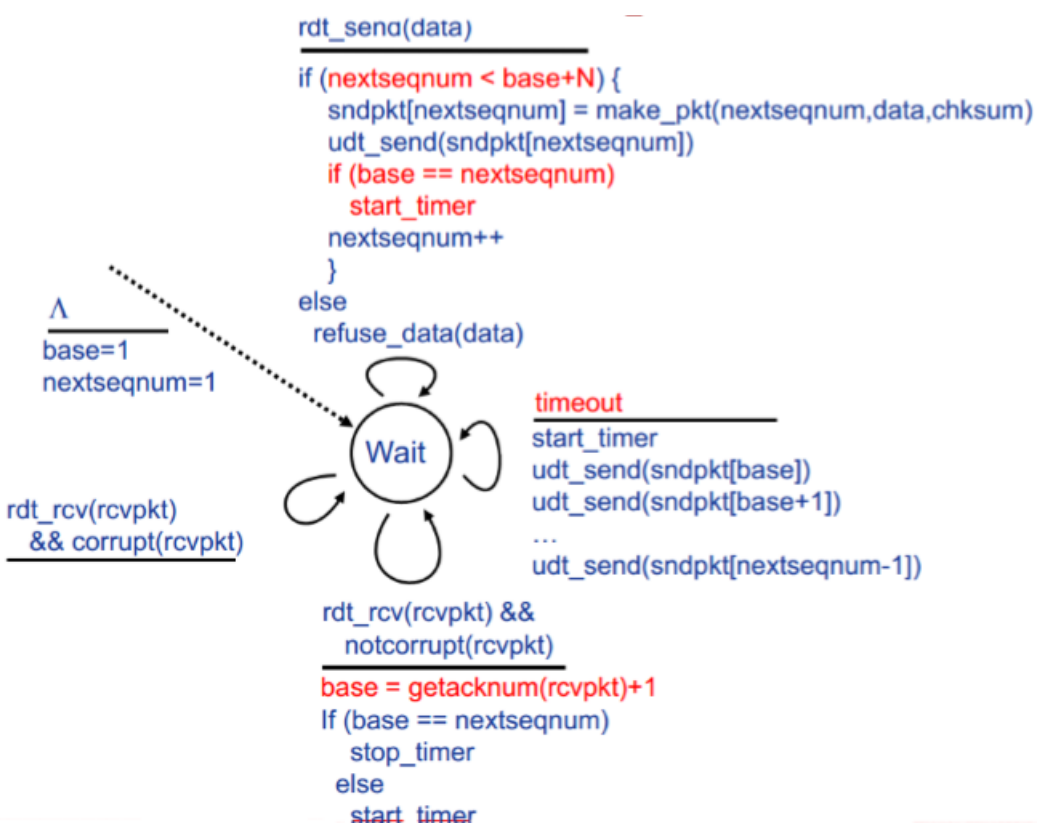
如果发生超时，则重发分组，如果连续超时10次，接收方已断开，终止。当收到ACK时，若收到重复确认, 此处应当立即重发；若没有重复，滑动窗口则向后移动。当确认到最后一个分组时，结束。关键代码如下：

```
            if (self.send_base == expect_seq % 256):
                # 收到重复确认，此处应当立即重发
                # pass
                for i in range(self.send_base, self.next_seq):
                    print('Sender resend packet:', i)
                    self.udp_send(self.packets[i])

            self.send_base = max(self.send_base, (ack_seq + 1) % 256)
            if self.send_base == self.next_seq:  # 已发送分组确认完毕
                self.sender_socket.settimeout(None)
                return True

        except socket.timeout:
            # 超时，重发分组.
            print('Sender wait for ACK timeout.')
            for i in range(self.send_base, self.next_seq):
                print('Sender resend packet:', i)
                self.udp_send(self.packets[i])
            self.sender_socket.settimeout(self.timeout)  # reset timer
            count += 1
```

流程图：



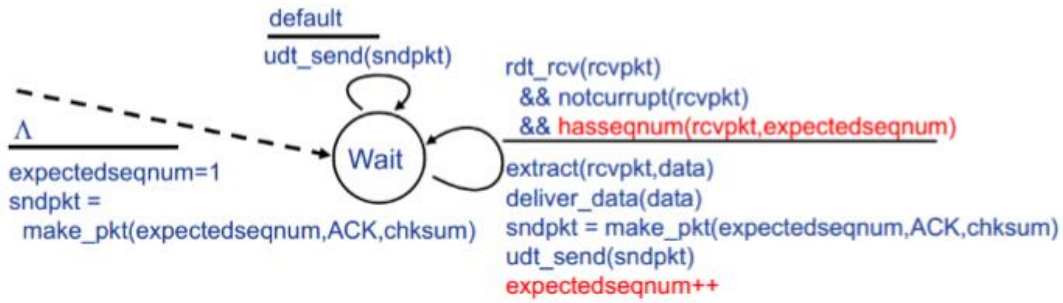
接收方：

当接收到数据传输时，因为是累积确认，所以要判断当前到达的序列号是否是期待的序列号，同时如果检查校验和未出错，则用make_pkt函数产生确认分组，返回确认分组并期待

下一个序列号分组，同时滑动窗口；如果出错或者数据包丢失，收到的不是所期待的数据分组，则重复发送ACK；如果最后一个数据包确认即标志位为1的数据包被确认，结束。关键代码如下：

```
print('Receiver receive packet:', seq_num)
# 收到期望数据包且未出错
if seq_num == self.expect_seq and getChecksum(data) == checksum:
    self.expect_seq = (self.expect_seq + 1) % 256
    ack_pkt = self.make_pkt(seq_num, self.expect_seq)
    self.udp_send(ack_pkt)
    if flag:    # 最后一个数据包
        return data, True
    else:
        return data, False
else:
    ack_pkt = self.make_pkt((self.expect_seq - 1) % 256, self.expect_seq)
```

流程图：

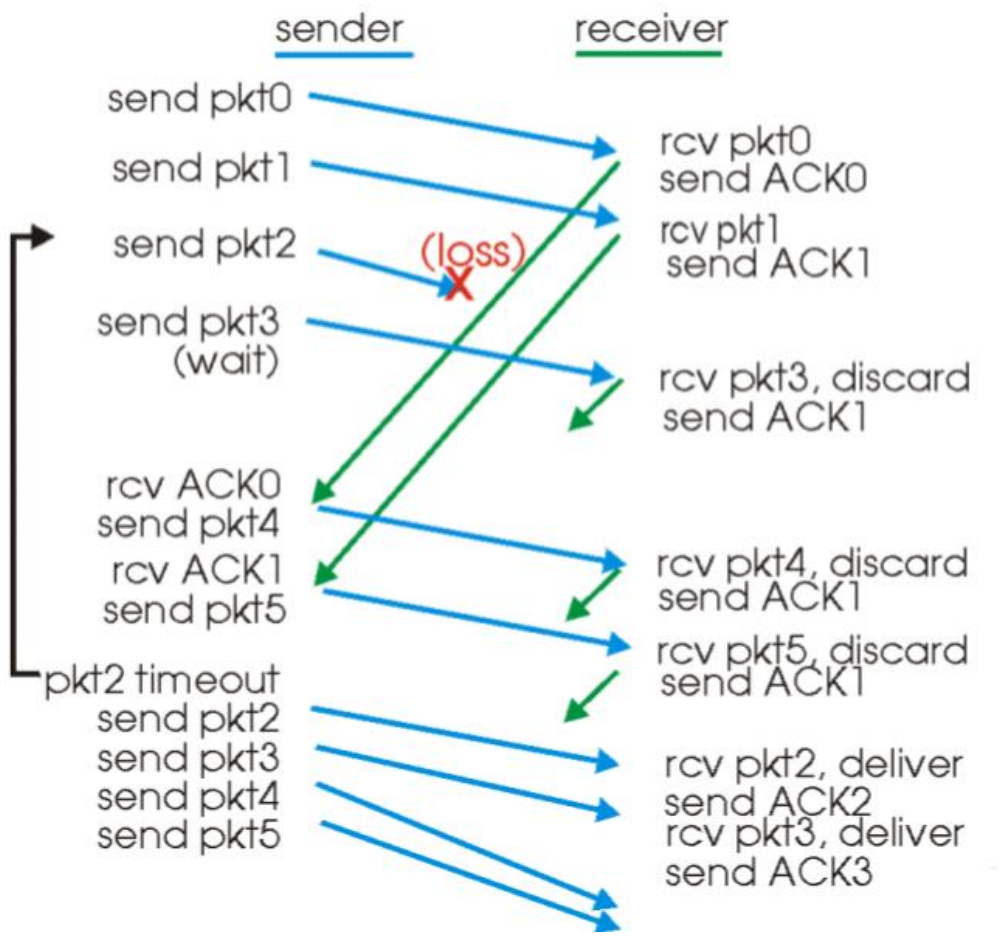


2) 模拟引入数据包的丢失，验证所设计协议的有效性。

我的发送的数据分组时随机丢失的，通过设置LOSS_RATE参数设置丢失率。然后用随机生成函数的生成整数，来判断包裹是否丢失。例如LOSS_RATE=0.1，random.randint(1, 1/0.1)，也就是[1 ， 10]之间随机生成一个数，若是我规定的数生成，则丢失。关键代码如下：

```
if self.loss_rate == 0 or random.randint(1, int(1 / self.loss_rate)) != 1:
    self.sender_socket.sendto(pkt, self.address)
else:
    print('Packet lost.')
time.sleep(0.3)
```

典型交互过程：



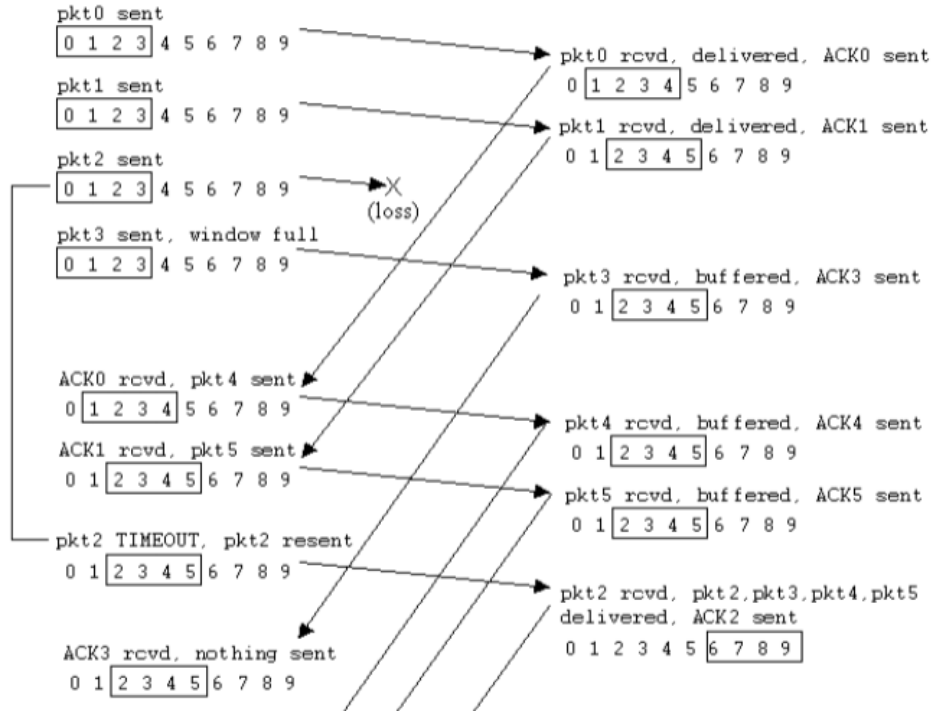
3) 改进所设计的 GBN 协议，支持双向数据传输；（选作内容，加分项目）

在客户端(client)与服务器端(server)分别建立一个 GBN 发送端(sender)和一个 GBN 接收端(receiver)，使用两对会话模拟双向数据传输。每个发送端和接受端如何编写参考第一个问题方案。结果在后面给出。

4）将所设计的 GBN 协议改进为 SR 协议。（选作内容，加分项目）

SR协议与GBN不同的是SR协议不是累积确认，是接收方对每个分组单独进行确认，会设置缓存机制，缓存乱序到达的分组。发送方只重传那些没收到ACK的分组，并为每个分组设置定时器。

传输示例：



数据分组和确认分组ACK格式同GBN。最大特点是不累积确认，而是到一个确认一个：

发送方：将窗口中数据发出，如果超时，则重传并重新设置计时器；如果传输窗口中数据分组乱序到达，也能将数据分组缓存，如果窗口最小序列号的分组被确认，则窗口滑动。

流程图：

data from above :
- if next available seq # in window, send pkt

timeout(n):
- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:
- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

接收方：接受到接受窗口中的数据则会返回ACK确认，不在窗口中的数据不接收；如果窗口最小序列号的分组被接受，则窗口滑动。

流程图：

pkt n in [rcvbase, rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]
- ACK(n)

otherwise:
- ignore

发送方与接收方窗口不一定同步。

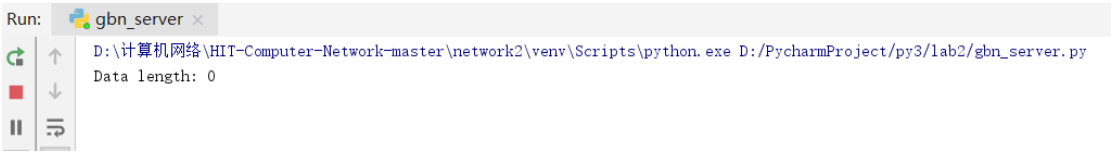
实验结果：

采用演示截图、文字说明等方式，给出本次实验的实验结果。

1) GBN 协议测试：

我传输的是图片，用时间戳作为图片名称。同时模拟引入数据包的丢失，验证所设计协议的有效性。

首先，将分组丢失概率设置为 0.1。 运行 gbn_server.py，使接收端处于监听状态：



接着，运行 gbn_client.py，通过发送端向接收端发送数据文件 client/data.jpg。数据分组个数为13。

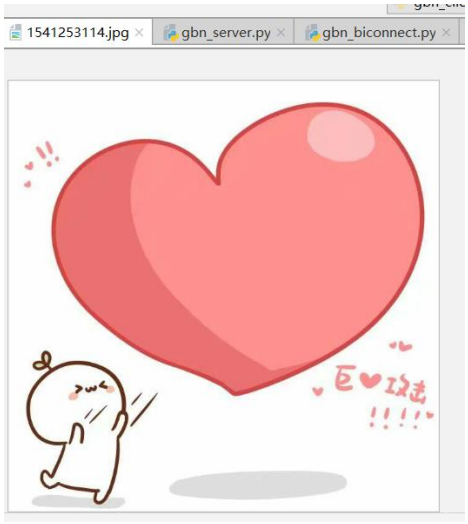
接收方：

```
============= RESTART: D:\PycharmProject\py3\lab2\gbn_server.py =======
seq:0  flag:0
Receiver receive packet: 0
Receiver send ACK: 0 1
Data length: 2048
seq:1  flag:0
Receiver receive packet: 1
Receiver send ACK: 1 2
Data length: 2048
seq:2  flag:0
Receiver receive packet: 2
Receiver send ACK: 2 3
Data length: 2048
seq:3  flag:0
Receiver receive packet: 3
Receiver send ACK: 3 4
Data length: 2048
seq:4  flag:0
Receiver receive packet: 4
Receiver send ACK: 4 5
Data length: 2048
seq:5  flag:0
Receiver receive packet: 5
Receiver send ACK: 5 6
Data length: 2048
seq:6  flag:0
Receiver receive packet: 6
Receiver send ACK: 6 7
Data length: 2048
seq:7  flag:0
Receiver receive packet: 7
Receiver send ACK: 7 8
Data length: 2048
seq:8  flag:0
Receiver receive packet: 8
Receiver send ACK: 8 9
Data length: 2048
seq:9  flag:0
Receiver receive packet: 9
Receiver send ACK: 9 10
Data length: 2048
seq:10  flag:0
Receiver receive packet: 10
Receiver send ACK: 10 11
Data length: 2048
seq:11  flag:0
Receiver receive packet: 11
Receiver send ACK: 11 12
Data length: 2048
seq:12  flag:1
Receiver receive packet: 12
Receiver send ACK: 12 13
Data length: 1659
>>>
```

发送方：

```
============= RESTART: D:\PycharmProject\py3\lab2\gbn_client.py =
The total number of data packets:  13
Sender send packet: 0
Sender send packet: 1
Sender send packet: 2
Sender send packet: 3
Sender receive ACK: 0 1
Sender receive ACK: 1 2
Sender receive ACK: 2 3
Sender receive ACK: 3 4
Sender send packet: 4
Sender send packet: 5
Sender send packet: 6
Sender send packet: 7
Sender receive ACK: 4 5
Sender receive ACK: 5 6
Sender receive ACK: 6 7
Sender receive ACK: 7 8
Sender send packet: 8
Sender send packet: 9
Sender send packet: 10
Packet lost.
Sender send packet: 11
Sender receive ACK: 8 9
Sender receive ACK: 9 10
Sender receive ACK: 9 10
Sender resend packet: 10
Sender resend packet: 11
Packet lost.
Sender receive ACK: 10 11
Sender wait for ACK timeout.
Sender resend packet: 11
Sender receive ACK: 11 12
Sender send packet: 12
Sender receive ACK: 12 13
>>>
```

传输结果（server/1541253114.jpg，文件使用时间戳命名）：



可见数据传输成功，接收端正确地接收到了发送端发送的数据文件。

2) 改进所设计的 GBN 协议，支持双向数据传输；（选作内容，加分项目）

运行 gbn_biconnect.py，进行双向数据传输测试。

其中，client 向 server 发送 client/data.jpg 文件：

server 向 client 发送 server/data.jpg 文件：

部分输出结果截图：

```
D:\计算机网络\HIT-Computer-Network-master\network2\venv\Scrip
The total number of data packets:  13
Sender send packet: 0
seq:0  flag:0
Receiver receive packet: 0
Receiver send ACK: 0 1
Data length: 2048
Sender send packet: 1
seq:1  flag:0
Receiver receive packet: 1
Receiver send ACK: 1 2
Data length: 2048
Sender send packet: 2
seq:2  flag:0
Receiver receive packet: 2
Receiver send ACK: 2 3
Data length: 2048
Sender send packet: 3
seq:3  flag:0
Receiver receive packet: 3
Receiver send ACK: 3 4
Data length: 2048
Sender receive ACK: 0 1
Sender receive ACK: 1 2
Sender receive ACK: 2 3
Sender receive ACK: 3 4
Sender send packet: 4
seq:4  flag:0
Receiver receive packet: 4
Receiver send ACK: 4 5
Data length: 2048
Sender send packet: 5
seq:5  flag:0
Receiver receive packet: 5
```

```
seq:11  flag:0
Receiver receive packet: 11
Receiver send ACK: 11 12
Data length: 2048
Sender receive ACK: 8 9
Sender receive ACK: 9 10
Sender receive ACK: 10 11
Sender receive ACK: 11 12
Sender send packet: 12
seq:12  flag:1
Receiver receive packet: 12
Receiver send ACK: 12 13
Data length: 1659
Sender receive ACK: 12 13
The total number of data packets:  15
Sender send packet: 0
seq:0  flag:0
Receiver receive packet: 0
Receiver send ACK: 0 1
Data length: 2048
Sender send packet: 1
seq:1  flag:0
Receiver receive packet: 1
Receiver send ACK: 1 2
Data length: 2048
Sender send packet: 2
seq:2  flag:0
Receiver receive packet: 2
Receiver send ACK: 2 3
Data length: 2048
Sender send packet: 3

 seq:13  flag:0
 Receiver receive packet: 13
 Receiver send ACK: 13 14
 Data length: 2048
 Sender send packet: 14
 seq:14  flag:1
 Receiver receive packet: 14
 Receiver send ACK: 14 15
 Data length: 279
 Sender receive ACK: 12 13
 Sender receive ACK: 13 14
 Sender receive ACK: 14 15

 Process finished with exit code 0
```

传输结果：

server 收到数据后写入 server/1541253713.jpg 文件：

client 收到数据后写入 client/1541253713.jpg 文件：

可见双向数据传输成功。

4）SR 协议测试（选作内容，加分项目）

运行 sr_server.py，使得接收端处于监听状态：

接着，运行 sr_client.py，通过发送端向接收端发送数据文件 client/data.jpg（数据分组个数为 13）：

接收方：

```
''' 
============== RESTART: D:\PycharmProject\py3\lab2\sr_server.py ======
seq:0  flag:0
Receiver receive packet: 0
Receiver send ACK: 0 0
Data length: 2048
seq:1  flag:0
Receiver receive packet: 1
Receiver send ACK: 1 1
Data length: 2048
seq:2  flag:0
Receiver receive packet: 2
Receiver send ACK: 2 2
Data length: 2048
seq:3  flag:0
Receiver receive packet: 3
Receiver send ACK: 3 3
Data length: 2048
seq:4  flag:0
Receiver receive packet: 4
Receiver send ACK: 4 4
Data length: 2048
seq:5  flag:0
Receiver receive packet: 5
Receiver send ACK: 5 5
Data length: 2048
seq:6  flag:0
Receiver receive packet: 6
Receiver send ACK: 6 6
Data length: 2048
seq:7  flag:0
Receiver receive packet: 7
Receiver send ACK: 7 7
Data length: 2048
seq:8  flag:0
Receiver receive packet: 8
Receiver send ACK: 8 8
Data length: 2048
seq:9  flag:0
Receiver receive packet: 9
Receiver send ACK: 9 9
Data length: 2048
seq:10  flag:0
Receiver receive packet: 10
Receiver send ACK: 10 10
Data length: 2048
seq:11  flag:0
Receiver receive packet: 11
Receiver send ACK: 11 11
Data length: 2048
seq:12  flag:1
Receiver receive packet: 12
Receiver send ACK: 12 12
Data length: 1659
>>>
```

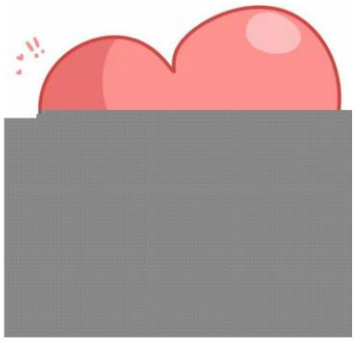
发送方：

```
============== RESTART: D:\PycharmProject\py3\lab2\sr_client.py =
The total number of data packets:  13
Sender send packet: 0
Sender send packet: 1
Sender send packet: 2
Sender send packet: 3
Sender receive ACK: 0 0
Sender receive ACK: 1 1
Sender receive ACK: 2 2
Sender receive ACK: 3 3
Sender send packet: 4
Sender send packet: 5
Sender send packet: 6
Sender send packet: 7
Sender receive ACK: 4 4
Sender receive ACK: 5 5
Sender receive ACK: 6 6
Sender receive ACK: 7 7
Sender send packet: 8
Sender send packet: 9
Sender send packet: 10
Sender send packet: 11
Sender receive ACK: 8 8
Sender receive ACK: 9 9
Sender receive ACK: 10 10
Sender receive ACK: 11 11
Sender send packet: 12
Sender receive ACK: 12 12
>>>
```

传输结果（server/1541253274.jpg，文件使用时间戳命名）：

可见数据传输成功，接收端正确地接收到了发送端发送的数据文件。

附：数据文件未传输完毕时的图像，证明传输过程的有效性



问题讨论：

我在实验过程中发现，注意细节真的很重要。在数据分组传输和接收，确认分组的传输和接受，里面有很多相似但不同的思路，一旦错用某个参数，整个程序将无法运行，卡死在某个地方，细节真的很重要。

如何使程序截至也是一个问题。在GBN中，我们可以以最后一个分组被确认而终止；在SR中，这种方法就不可以。因为SR是乱序到达，GBN是累积确认。SR这种情况，需要等窗口不能滑动且窗口内数据均被确认，才能终止。这块值得注意。

心得体会：

通过本次实验，我理解了滑动窗口协议的基本原理； 掌握了 GBN 协议的工作原理； 掌握了 SR 协议的工作原理； 掌握了基于 UDP 设计并实现一个可靠数据传输协议的过程与技术； 进一步掌握了使用 Python 语言进行 socket 编程的方法和技术。

我了解到了Pycharm和IDLE在运行中的优缺点。IDLE可以同时运行多个python，有多个输出结果框同时运行，而Pycharm只能输出一个结果框。

我知道自己还有很多不足和知识盲区，要继续努力前行。