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-停等协议				
姓名	董琦		院系	数据科学与大数据	
班级	2003501		学号	120L020701	
任课教师	刘亚维		指导教师	刘亚维	
实验地点	G001		实验时间	2022/10/13	
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：

理解可靠数据传输的基本原理；掌握停等协议的工作原理；掌握基于 UDP 设计并实现一个停等协议的过程与技术。

实验内容：

1) 基于 UDP 设计一个简单的停等协议，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) 改进所设计的停等协议，支持双向数据传输；（4）基于所设计的停等协议，实现一个 C/S 结构的文件传输应用。

实验过程：

1. 报文结构

一个MSS共长1500B，第0B是本端发送报文的序号，第1B是确认对方发送报文的序号，2、3B合起来用以标志报文内容的长度，剩下的1496B为报文内容部分。

由于考虑到后续扩展为双向通信的协议，故客户端和服务端的发送和接收报文格式一致，均为上述格式。

例：若本次发送的包在文件中序号为3，而且接收到对方序号为2的包，且此次包内容大小800B，则报文第0B内容为3，第1B内容为2，第2、3B合并内容为800，余下内容长度为800B，总长度为804B。

2. 协议流程图：

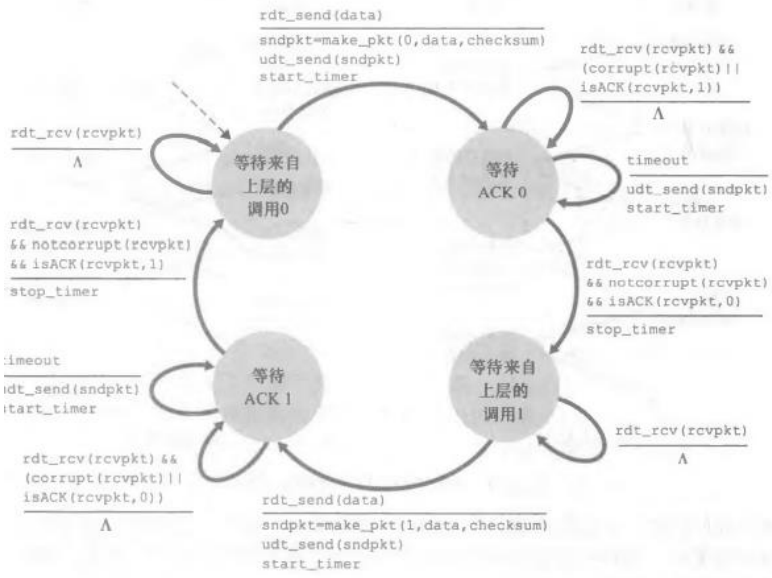


图 3-15 rdt3.0 发送方

这里只实现了C/S架构的程序发送，故服务端按照图上的状态机格式来设计。

只有当接收到来自对方的确认报文后，才会更新序号，在停等协议中，序号只需在0、1间跳变即可。

3. 数据丢失验证模拟方法

数据丢失分为三种情况：

1） 发送包丢失

这里由发送方概率不发送包来模拟：

```
if(Math.random()>0.1)
    this.socket.send(send);
else
    System.out.println("发送的包丢失。");
```

2） 确认包丢失

这里由接收方概率不发送确认包来模拟：

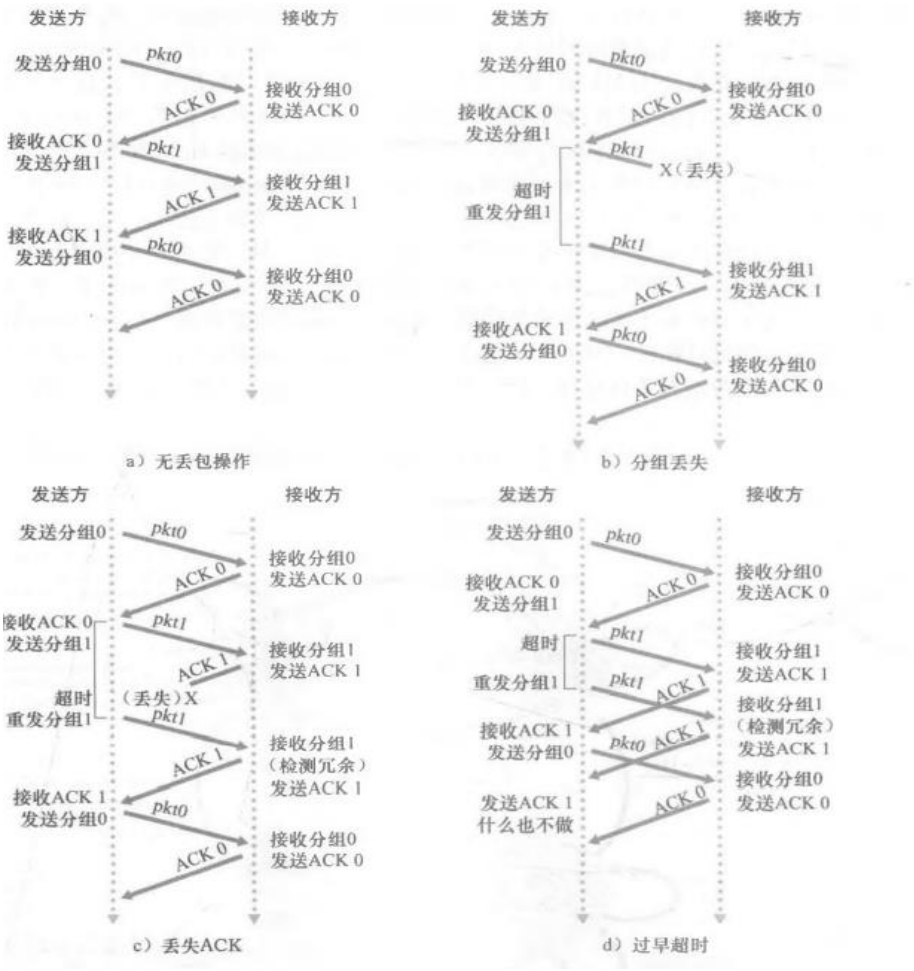
```
//模拟返回包丢失，方法是不发送
if(Math.random()>0.1){
    this.socket.send(packet);
}else{
    System.out.println("返回的包丢失");
}
```

3） 确认包超时

这里由一个线程延时函数来模拟：

```
//模拟延时收到，方法是延迟接收
if(Math.random()<0.1)
    Thread.sleep( millis: 1000);
```

发生以上情况时，客户端会执行重发的操作，示例图如下：



4. 交互过程

1） 发送方每次仅将当前信息帧作为备份保留在缓冲存储器中；

2） 当发送方开始发送信息帧时，赋予该信息帧一个帧序号，随即启动计时器；

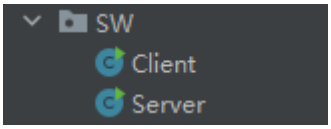
3） 当接收方收到无差错的信息帧后，即向发送方返回一个与该帧序号相同序号的ACK确认帧；

4） 当接收方检测到一个含有差错的信息帧时，便舍弃该帧；

5） 若发送方在规定时间内收到ACK确认帧，即将计时器清零，需而开始下一帧的发送；

6） 若发送方在规定时间内未收到ACK确认帧，重发存于缓冲器中的待确认信息帧。

5. 程序实现：



分别是服务器和客户端俩个程序。

我使用了JAVA作为编程语言，在JAVA中，使用DataFrameSocket类作为UDP-socket，使用DataFramePacket类作为接收的报文。

1） 服务器

-time、-quit这些功能实现如下，之后俩个程序中不再叙述：

```
this.socket.receive(packet);
switch (new String(buf,  offset: 0, packet.getLength())) {
    case "-time":
        Date date = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd :hh:mm:ss");
        byte[] time = dateFormat.format(date).getBytes(StandardCharsets.UTF_8);
        DatagramPacket recall = new DatagramPacket(time, time.length, packet.getAddress(), packet.getPort());
        this.socket.send(recall);
        break;
    case "-quit":
        byte[] bye = "Good bye!".getBytes();
        this.socket.send(new DatagramPacket(bye, bye.length, packet.getAddress(), packet.getPort()));
        break;
    case "-testsw":
        String filename = "src\\main\\resources\\swTest.txt";
        send(filename, packet.getAddress(), packet.getPort());
        break;
    default:
        socket.send(packet);
        break;
```

这个实现原理很简单，只需检测接收到的报文内容字符串即可，然后做出对应动作即可。

Send()函数，用于实现文件的传输，在这里我们实现了SW协议。

核心代码如下：

```
do {
    int length = 0;
    if (ack) {
        sendBuf[0] = this.serial;
        length = reader.read(sendBuf,  off: 4, mss);
        if (length == -1) {
            break;
        }
        sendBuf[2] = (byte) (length & 0xFF);
        sendBuf[3] = (byte) (length >> 8  & 0xFF);
        //模拟发出后未收到，方法是不发送
        if(Math.random()>0.1)
            this.socket.send(send);
    }
    try {
        this.socket.receive(recv);
        if (recv.getData()[1] == this.serial) {
            updateSerial();
            ack = true;
        } else {
            System.out.println("检测冗余,不做反应。");
            ack = false;
        }
    } catch (SocketTimeoutException e) {
        System.out.println("超时，重新发送分组。");
        ack = false;
        if(Math.random()>0.1)
            this.socket.send(send);
        else
            System.out.println("发送的包丢失。");
    }
} while (true);
```

当收到接收方的ACK报文时，则读取下一个包的缓存内容，构建新的包并更新序号发送给客户端；当发生ACK序号错误，则对应检测冗余情况；当报文超时（由于只有一个包，故直接令端口监听时长当作定时器）时，执行重发操作。

2） 客户端

```
1 个用法  DQ
public void qTime(){...}
1 个用法  DQ
public void qQuit(){...}
1 个用法  DQ
public void qTest(){...}
1 个用法  DQ
```

分别为测试三个功能的函数，前俩个比较简单，不作赘述。

qTest核心代码如下：

```
do {
    //模拟延时收到，方法是延迟接收
    if(Math.random()<0.1)
        Thread.sleep( millis: 1000);
    this.socket.receive(recv);
    buf[1] = recv.getData()[0];
    if (recv.getData()[0] == this.serial) {
        System.out.println("收到正确的分组。");
        count++;
        //收到正确分组，更新序号，保存数据
        updateSerial();

        length = recv.getData()[2]& 0xFF;
        length |= recv.getData()[3] << 8;

        out.write(buf, off: 4, length);
    }else{
        System.out.println("收到冗余分组。");
    }
    //模拟返回包丢失，方法是不发送
    if(Math.random()>0.1){
        this.socket.send(packet);
    }else{
        System.out.println("返回的包丢失");
    }
} while (length == mss);
```

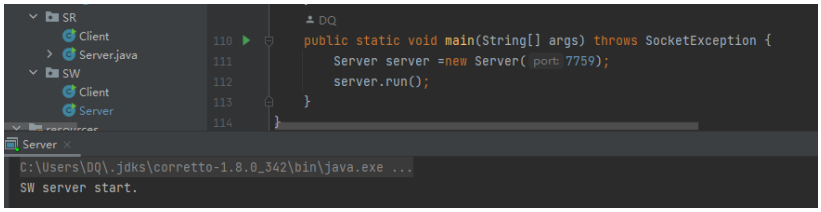
当接收到服务器发来的报文后，检测其序号是否为期望的序号。若是则发送确认报文，并且保存内容；若否则为冗余分组，不做处理，返回一个包含期望序号的报文。

6. UDP编程主要特点

UDP是一个面向无连接的协议，我们可以直接向目标地址发送报文而不用提前建立连接。UDP也不提供差错检验等纠错功能，采用尽力而为的发送策略，由此我们可以使用UDP来模拟我们网络中较底层的结构，来实现网络层的发送协议。

实验结果：

打开服务器，监听7759端口：

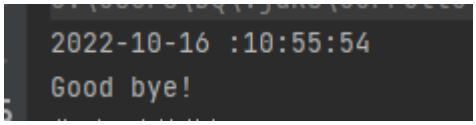


客户端运行程序，分别进行三个功能的测试：

```
public static void main(String[] args) throws SocketException {
    Client client=new Client( port: 7760);
    client.qTime();
    client.qTest();
    client.qQuit();
}
```

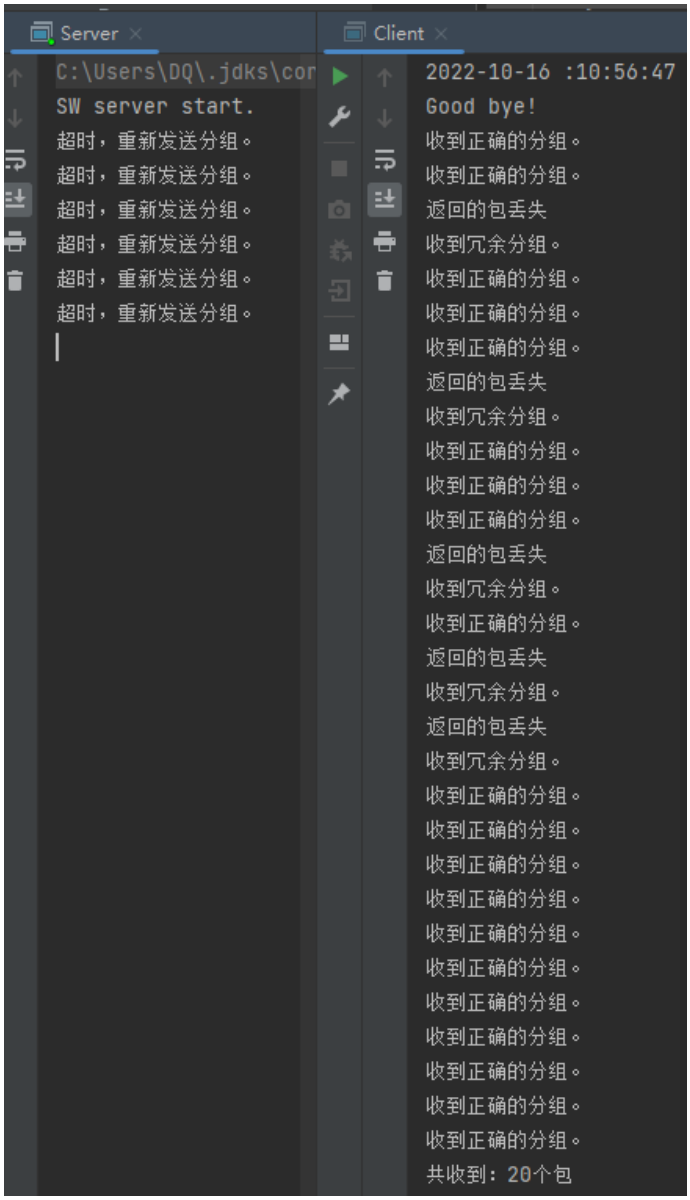
测试结果输出如下。

首先是前两个功能：



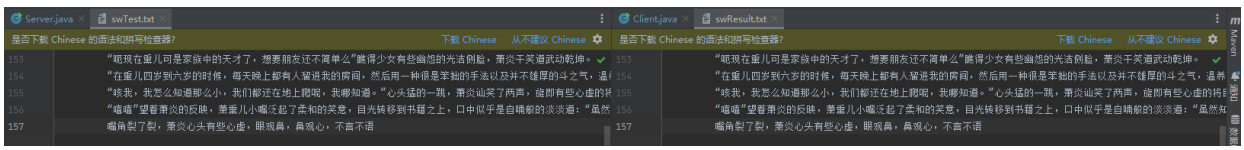
与期望功能一致。

然后是文件传输功能：



可以看到俩端对于这个文件传输过程的不同视角，当发送丢包、超时情况时，服务端可以执行重发，从而完成可靠传输。

然后检查传输的文件，发现是一模一样的：



至此，SW实验内容全部完成。

问题讨论：
Q:rdt3.0协议可否应用在实际中？ A:rdt3.0可以工作, 但是性能很差1 Gbps 链路, 15 ms 传播延迟, 8000 bit数据报: 每30 msec发送 1KB pkt -> 33kB/sec (1 Gbps 链路)，这是一个网络协议严重影响链路资源利用的一个例子!所以我们设计了滑动窗口协议解决此类问题.
心得体会：
通过用代码实现停等协议，我了解了停等协议的接收、发送双方仅需设置一个帧的缓冲存储空间和帧序号只取0或1的两个状态标志位，便可有效地实现数据重发并确保接收方接受的数据不会重复。同时也知道了rdt协议的效率过低的原因：发送窗口固定为1，不能滑动。从而产生了滑动窗口协议。