华东师范大学数据科学与工程学院实验报告

课程名称：计算机网络与编程	**年级：**2021 级	**上机实践成绩：**
指导教师：张召	**姓名：**彭一珅	**学号：**10215501412
上机实践名称：基于 UDP 的 Socket 编程		**上机实践日期：**2023.5.5
上机实践编号：9	**组号：**	**上机实践时间：**9:50

一、实验目的

学习使用Datagram Socket 实现 UDP 通信

二、实验任务

使用DatagramSocket 和 DatagramPacket 编写代码

三、使用环境

IntelliJ IDEA

JDK 版本: Java 19

四、实验过程

Task1: 完善 UDPProvider 和 UDPSearcher，使得接受端在接受到发送端发送的信息后，将该信息向发送端回写，发送端将接收到的信息打印在控制台上，将修改后的代码和运行结果附在实验报告中。

运行原始代码，得到如下结果：

```
阻塞等待发送者的消息...
我是接受者，172.30.213.226:9092 的发送者说： 用户名admin; 密码123
```

回写消息的步骤是准备字节形式的回送数据、创建数据报、调用 send 方法发回给发送端。

仿照 UDPSearcher 的发送消息，在 UDPProvider 中实现如下代码：

```
// Task 1 TODO: 准备回送数据; 创建数据报，用于发回给发送端; 发送数据报
        String newData="接受者已收到消息:"+data;
        byte[] sendBytes = newData.getBytes(StandardCharsets.UTF_8);
        DatagramPacket returnPacket = new DatagramPacket(sendBytes, offset: 0,sendBytes.length,
                InetAddress.getLocalHost(), port: 9092);
        datagramSocket.send(returnPacket);
        System.out.println("数据回复完毕...");
```

接收消息的步骤是创建读数据缓冲区、调用 receive 方法接收数据报、将数据写入缓冲区并输出。

仿照 UDPProvider 的发送消息，在 UDPSearcher 中实现如下代码：

```
// Task 1 TODO: 准备接收Provider的回送消息; 查看接受信息并打印
        byte[] buf = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(buf, offset: 0, buf.length);
        System.out.println("阻塞等待接受者的回复...");
        datagramSocket.receive(receivePacket);
        int len = receivePacket.getLength();
        String data = new String(receivePacket.getData(), offset: 0, len);
        System.out.println("我是发送者，接受者回复消息:"+ data);
```

输出结果如下：

```
阻塞等待发送者的消息...
我是接受者，172.30.213.226:9092 的发送者说: 用户名admin; 密码123
数据回复完毕...
```

```
数据发送完毕...
阻塞等待接受者的回复...
我是发送者，接受者回复消息:接受者已收到消息:用户名admin; 密码123
```

Task2: 改写 UDPProvider 和 UDPSearcher 代码完成以下功能，并将实验结果附在实验报告中：

```
广播地址: 255.255.255.255

现需要设计完成如下场景:
    UDPSearcher将UDP包发送至广播地址的9091号端口（这表示该UDP包将会被广播至局域网下所有主机的对应端口）。
    如果有UDPProvider在监听，解析接受的UDP包，通过解析其中的data得到要回送的端口号，并将自己的一些信息写回，UDPSearcher接收到UDPProvider的消息后打印出来。

现提供发送消息的格式:
    UDPSearcher请使用如下buildWithPort构建消息，port在实验中指定为30000。
    UDPProvider请使用如下parsePort解析收到的消息并得到要回写的端口号，然后用buildWithTag构建消息，tag可以是 String tag = UUID.randomUUID().toString()，然后回写。
    UDPSearcher请使用parseTag得到Tag。
```

首先根据 port=30000 构建发送的数据：

```
String sendData= MessageUtil.buildWithPort(30000);
```

设置广播发送，并用 getByName 方法获取要广播的 ip 地址：

```
datagramSocket.setBroadcast(true);
建数据报，包含要发送的数据
DatagramPacket sendPacket = new DatagramPacket(sendBytes, offset: 0, sendBytes.length,
        InetAddress.getByName( host: "255.255.255.255"), port: 9091);
```

接收数据后，解析出发送的端口号：

```
int pt=MessageUtil.parsePort(data);
```

随机生成 UUID 作为 Tag，将 Tag 写到消息里：

```
String tag = UUID.randomUUID().toString();
String newData=MessageUtil.buildWithTag(tag);
```

解析出回复消息中的 Tag：

```
String tag=MessageUtil.parseTag(data);
System.out.println("我是发送者，接受者回复Tag:"+ tag);
```

最终实验结果如图：

```
阻塞等待发送者的消息...
我是接受者，192.168.192.1:9092 的发送者输出的端口号是: 30000
数据回复完毕...
```

```
数据发送完毕...
阻塞等待接受者的回复...
我是发送者，接受者回复Tag:cce373b5-a6d0-41b6-bfe6-0110f2c2771a
```

Task3: 现使用UDP 实现文件传输功能，给出 UDPFileSender 类、请完善 UDPFileReceiver 类，实现接收文件的功能。请测试在文件参数为 1e3 和 1e8 时的情况，将修改后的代码和运行时截图附在实验报告中，并对实验现象进行解释说明。

用 while 循环接收数据，用 for 循环将指定长度的数据写入文件，如果遇到空数组（长度为 0）则跳出循环：

```
// TODO 实现不断接收数据报并将其通过文件输出流写入文件，以数据报长度为零作为终止条件
        while(true){
            ds.receive(dp);
            int len;
            if((len=dp.getLength())==0){
                break;
            }
//              System.out.println("发送数据报长度为："+len);
            byte[] d;
            d=dp.getData();
            for (int i=0;i<len;i++){
                output.write(d[i]);
            }
        }
```

运行结果显示，发送的和接受到的文件的 md5 编码相同：

```
发送文件生成完毕
发送文件的md5为: 28b8ac4cecba9a6f46071edd09c0cb12
向DESKTOP-CNIAOJM/172.31.0.239发送文件完毕！端口号为:9091
```

```
接收来自/172.31.0.239的文件已完成！对方所使用的端口号为：52357
接收文件的md5为: 28b8ac4cecba9a6f46071edd09c0cb12
```

以 1e8 数量级发送，文件生成时间显著变慢，接收方无法输出结束信息 md5 编码：

```
发送文件生成完毕
发送文件的md5为: 8b458801249a2ddc48240b1581f1ff1c
向DESKTOP-CNIAOJM/172.31.0.239发送文件完毕! 端口号为:9091
```

```
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
发送数据报长度为：1024
```

将逐个字节读取的 for 循环改为输出字符串：

```
String d=new String(dp.getData(), offset: 0,len);
output.write(d.getBytes());
```

可以看到，虽然能接收到文件结束，但是 md5 编码与发送的文件不同。

```
接收来自/172.31.0.239的文件已完成! 对方所使用的端口号为：53512
接收文件的md5为: c437180468c423b07e8fac52f112552d
```

观察文件大小，可以看到发送的文件有 290.48MB，而接收到的文件只有 276.12MB。

这是由于数据包发送的过程中，可能存在接受者还在向文件写入，而发送方已经发送到下一个数据包，而这个数据包就丢失了的情况。

通过理论课的学习，UDP 协议是不可靠、无连接、面向报文的，可能发生丢包、差错、乱序情况。

Bonus Task1: (2 选 1) 试完善文件传输功能，可选择 1.使用基于 TCP 的 Socket 进行改写；2.优化基于 UDP文件传输，包括有序发送、接收端细粒度校验和发送端数据重传。请测试在文件参数为 1e8 时的情况，将修改后的代码和运行时截图附在实验报告中。

TCPServer 代码实现（接收文件）：

```
public void start(int port) throws IOException {
    serverSocket = new ServerSocket(port);
    System.out.println("阻塞等待客户端连接中...");
    clientSocket = serverSocket.accept();
    InputStream is = clientSocket.getInputStream();
    File file = new File( pathname: "checksum_recv.txt");
    FileOutputStream output = new FileOutputStream(file);
    while(true){
        int data;
        data = is.read();
        if(data==-1){
            break;
        }
        System.out.println("发送数据报为："+data);
        output.write(data);
    }
}
```

TCPClient 代码实现（发送文件）：

```
try (FileWriter fileWriter = new FileWriter( fileName: "checksum.txt")) {
    Random r = new Random( seed: 2023);
1e3 and 1e8
    for (int i = 0; i < 1e8; i++) {
        fileWriter.write(r.nextInt());
    }
}
File file = new File( pathname: "checksum.txt");
System.out.println("发送文件生成完毕");
System.out.println("发送文件的md5为：" + MD5Util.getMD5(file));
FileInputStream fis = new FileInputStream(file);
byte[] bytes = new byte[1024];
int len;
Socket socket = new Socket( host: "127.0.0.1", port);
OutputStream os = socket.getOutputStream();
while(true){
    len = fis.read(bytes);
    System.out.println(len);
    if(len==-1) break;
    for(int i=0;i<len;i++){
        os.write(bytes[i]);
    }
      os.write(bytes);
}
byte[] a = new byte[0];
os.write(a);
fis.close();
socket.close();
```

实验结果：

```
发送文件生成完毕
发送文件的md5为: 8b458801249a2ddc48240b1581f1ff1c
```

```
阻塞等待客户端连接中...
接收文件已完成
接收文件的md5为: 8b458801249a2ddc48240b1581f1ff1c
```

五、总结

UDP 是一种无连接的协议，它不保证数据传输的可靠性和顺序性，但具有传输速度快、实时性强等特点。UDP 的信息传输特征包括：无连接、不可靠、无拥塞控制、支持广播和多播等。

DatagramSocket 是 Java 中用于实现 UDP 协议的类，它可以实现 UDP 发送和接收端的功能。编写 UDP 发送端的步骤包括：创建 DatagramSocket 对象、创建 DatagramPacket 对象、将数据封装到 DatagramPacket 对象中、使用 DatagramSocket 对象发送 DatagramPacket 对象。编写 UDP 接收端的步骤包括：创建 DatagramSocket 对象、创建 DatagramPacket 对象、使用 DatagramSocket 对象接收 DatagramPacket 对象、从 DatagramPacket 对象中获取数据。

TCP 和 UDP 的区别主要在于可靠性、速度、连接方式等方面。TCP 是一种面向连接的协议，它保证数据传输的可靠性和顺序性，但具有传输速度慢、连接建立时间长等特点。TCP 的准确传输主要是通过三次握手、数据校验和等机制实现的。

因此，UDP 适用于对数据传输速度和实时性要求较高的场景，而 TCP 适用于对数据传输可靠性和顺序性要求较高的场景。