Lab9: 基于UDP的Socket编程

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

1. 预备知识

1.1 UDP的API

- UDP（user datagram protocol）的中文叫用户数据报协议，属于传输层。UDP协议提供的服务不同于TCP协议的端到端服务，它是面向非连接的，属于不可靠的协议，UDP套接字在使用前不需要进行连接，而使得通信效率更高。
- DatagramSocket，用于接收和发送UDP数据的类，负责发送或者接收UDP数据包

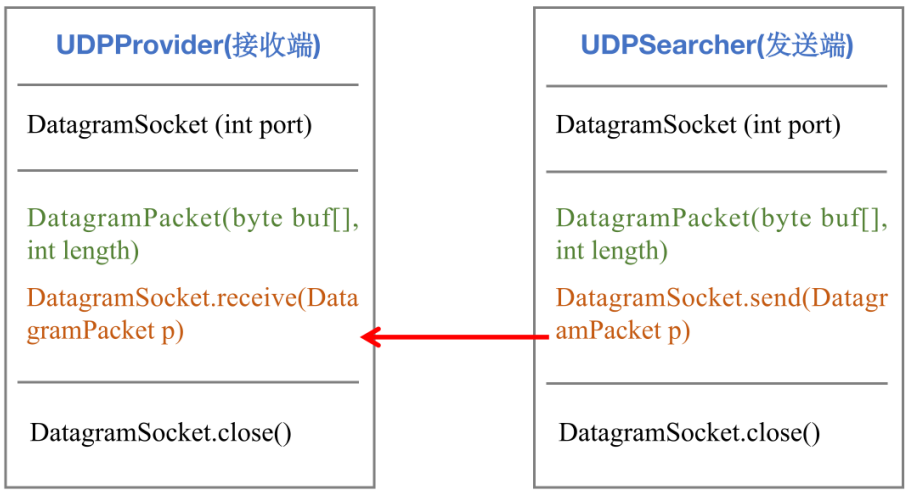
```
DatagramSocket(int port)  // 创建DatagramSocket实例(指定端口)
DatagramSocket(int port, InetAddress Addr) // 创建固定端口和ip的实例
receive(DatagramPacket d) // 接收DatagramPacket数据包
send(DatagramPacket d)    // 发送DatagramPacket数据包
setSoTimeout(int timeout) // 设置超时，毫秒为单位
```

- DatagramPacket，用于处理UDP数据的类，将字节数组、目标地址、目标端口打包成UDP报文，或者解析UDP报文

```
DatagramPacket(byte[] buf, int offset, int len, InetAddress Addr, int port) // offset和len指定了buffer数组的可用区间
setData(byte[] buf, int offset, int len)、getData()
setLength(int len)、getLength()
setPort(int port)、getPort()
setAddress(InetAddress address)、getAddress()
setSocketAddress(SocketAddress address)、getSocketAddress()
```

1.2 DatagramSocket 交互过程

UDP不分服务器端和客户端，为了更好地表示，这里采用发送者和接收者的说法



1.3 接收者端实现步骤

- 创建DatagramSocket，绑定端口号
- 创建DatagramPacket，用于接收UDP包
- 调用DatagramSocket的receive方法，接收发送者发送的UDP包
- 关闭套接字

1.4 发送者端实现步骤

- 创建DatagramSocket，绑定端口号
- 创建DatagramPacket，建立要发送的数据包，包含将要发送的信息
- 调用DatagramSocket的send方法，发送UDP数据包
- 关闭套接字

1.5 牛刀小试（发送端向接收端发送消息）

- 编写UDPProvider类

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class UDPProvider {
    public static void main(String[] args) throws IOException {
        // 1. 创建接受者端的DatagramSocket，并指定端口
        DatagramSocket datagramSocket = new DatagramSocket(9091);
        // 2. 创建数据报，用于接受客户端发来的数据
        byte[] buf = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(buf, 0, buf.length);
        // 3. 接受客户端发送的数据，此方法在收到数据报之前会一直阻塞
        System.out.println("阻塞等待发送者的消息...");
```

```
        datagramSocket.receive(receivePacket);

        // 4. 解析数据
        String ip = receivePacket.getAddress().getHostAddress();
        int port = receivePacket.getPort();
        int len = receivePacket.getLength();
        String data = new String(receivePacket.getData(),0, len);
        System.out.println("我是接受者, " + ip + ":" + port + " 的发送者说: "+ data);

        // 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

        // 5. 关闭datagramSocket
        datagramSocket.close();
    }
}
```

- 编写UDPSearcher类

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.nio.charset.StandardCharsets;

public class UDPSearcher {
    public static void main(String[] args) throws IOException {
        // 1. 定义要发送的数据
        String sendData = "用户名admin; 密码123";
        byte[] sendBytes = sendData.getBytes(StandardCharsets.UTF_8);
        // 2. 创建发送者端的DatagramSocket对象
        DatagramSocket datagramSocket = new DatagramSocket(9092);
        // 3. 创建数据报, 包含要发送的数据
        DatagramPacket sendPacket = new DatagramPacket(sendBytes, 0, sendBytes.length,
InetAddress.getLocalHost(), 9091);

        // 4. 向接受者端发送数据报
        datagramSocket.send(sendPacket);
        System.out.println("数据发送完毕...");

        // Task 1 TODO: 准备接收Provider的回送消息; 查看接受信息并打印

        // 5. 关闭datagramSocket
        datagramSocket.close();
    }
}
```

- 尝试运行UDPProvider和UDPSearcher

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**

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。

- MessageUtil工具类

```
class MessageUtil {
    private static final String TAG_HEADER = "special tag:";
    private static final String PORT_HEADER = "special port:";

    public static String buildWithPort(int port) {
        return PORT_HEADER + port;
    }

    public static int parsePort(String data) {
        if (data.startsWith(PORT_HEADER)) {
            return Integer.parseInt(data.substring(PORT_HEADER.length()));
        }
        return -1;
    }

    public static String buildWithTag(String tag) {
        return TAG_HEADER + tag;
    }
    public static String parseTag(String data) {
        if (data.startsWith(TAG_HEADER)) {
            return data.substring(TAG_HEADER.length());
        }
        return null;
    }
}
```

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。

- 提供UDPFileSender类

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.security.NoSuchAlgorithmException;
import java.util.Random;

public class UDPFileSender {
    public static void main(String[] args) throws IOException, NoSuchAlgorithmException
{
        // 生成并写入发送文件
        try (FileWriter fileWriter = new FileWriter("checksum.txt")) {
            Random r = new Random(2023);
            // 尝试 1e3 and 1e8
            for (int i = 0; i < 1e3; i++) {
                fileWriter.write(r.nextInt());
            }
        }

        File file = new File("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);
        DatagramSocket socket = new DatagramSocket();
        byte[] bytes = new byte[1024];
        DatagramPacket packet;

        // 不断从文件读取字节并将其组装成数据报发送
        int len;
        for(;;){
            len = fis.read(bytes);
            if(len==-1) break;
            packet = new DatagramPacket(bytes, len, InetAddress.getLocalHost(), 9091);
            socket.send(packet);
        }

        // 空数组作为发送终止符
        byte[] a = new byte[0];
        packet = new DatagramPacket(a, a.length, InetAddress.getLocalHost(), 9091);
```

```
        socket.send(packet);

        fis.close();
        socket.close();
        System.out.println("向" + packet.getAddress().toString() + "发送文件完毕！端口号为:" + packet.getPort());
    }
}
```

- 编写UDPFileReceiver类

```
import java.io.*;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class UDPFileReceiver {
    public static void main(String[] args) throws IOException {
        File file = new File("checksum_recv.txt");  //要接收的文件存放路径
        FileOutputStream output = new FileOutputStream(file);

        byte[] data=new byte[1024];
        DatagramSocket ds=new DatagramSocket(9091);
        DatagramPacket dp=new DatagramPacket(data, data.length);
        // 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

        output.close();
        ds.close();

        System.out.println("接收来自"+dp.getAddress().toString()+"的文件已完成！对方所使用的端口号为："+dp.getPort());
        file = new File("checksum_recv.txt");
        System.out.println("接收文件的md5为：" + MD5Util.getMD5(file));
    }
}
```

- MD5Util工具类

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;

public class MD5Util {
    public static String getMD5(File file) {
        FileInputStream fileInputStream = null;
        try {
            MessageDigest MD5 = MessageDigest.getInstance("MD5");
            fileInputStream = new FileInputStream(file);
```

```
            byte[] buffer = new byte[8192];
            int length;
            while ((length = fileInputStream.read(buffer)) != -1) {
                MD5.update(buffer, 0, length);
            }
            return new String(byte2hex(MD5.digest()));
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } finally {
            try {
                if (fileInputStream != null){
                    fileInputStream.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private static String byte2hex(byte[] b){
        String hs="";
        String stmp="";
        for (int n=0; n<b.length; n++){
            stmp=(java.lang.Integer.toHexString(b[n] & 0xFF));
            if (stmp.length()==1) hs=hs+"0"+stmp;
            else hs=hs+stmp;
        }
        return hs;
    }
}
```

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。