

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：2020 级

上机实践成绩：

指导教师：张召

姓名：张熙翔

学号：10205501427

上机实践名称：基于 UDP 的 Socket 通信

上机实践日期：2022/5/7

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

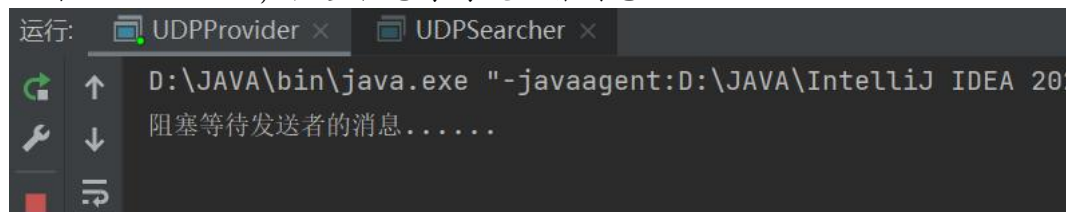
IntelliJ IDEA 2020.3.2

JDK 11.0.6

四、实验过程

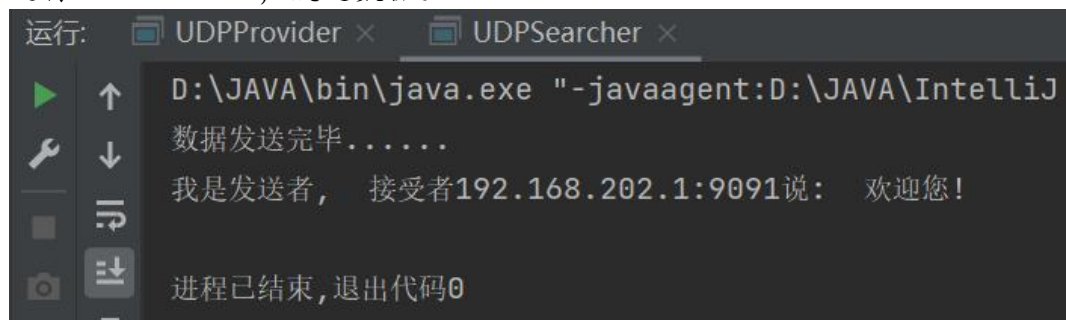
Task1：分别运行完善后的 UDPProvider 和 UDPSearcher，将实验结果附在实验报告中。

运行 UDPSearcher，阻塞状态等待发送者消息。



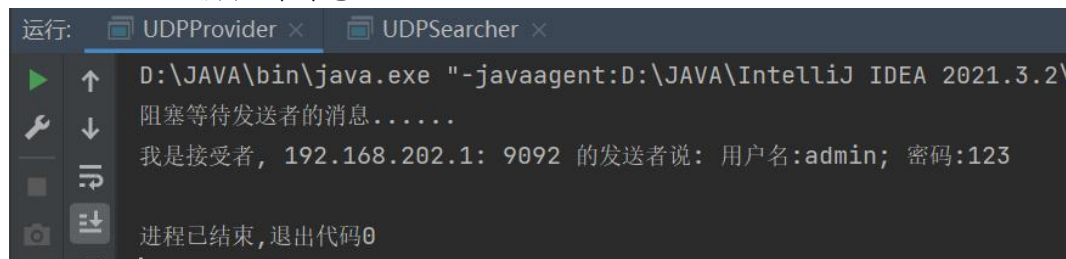
```
运行: UDPProvider x UDPSearcher x
D:\JAVA\bin\java.exe "-javaagent:D:\JAVA\IntelliJ IDEA 2020.3.2\lib\idea_rt.jar" 192.168.202.1:9091
阻塞等待发送者的消息.....
```

运行 UDPProvider，发送数据。



```
运行: UDPProvider x UDPSearcher x
D:\JAVA\bin\java.exe "-javaagent:D:\JAVA\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar" 192.168.202.1:9091
数据发送完毕.....
我是发送者, 接受者192.168.202.1:9091说: 欢迎您!
进程已结束,退出代码0
```

UDPSearcher 接收到消息。



```
运行: UDPProvider x UDPSearcher x
D:\JAVA\bin\java.exe "-javaagent:D:\JAVA\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar" 192.168.202.1:9091
阻塞等待发送者的消息.....
我是接受者, 192.168.202.1: 9092 的发送者说: 用户名:admin; 密码:123
进程已结束,退出代码0
```

Task2: 改写 UDPPProvider 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UDPPProvider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UDPPProvider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。
UDPPProvider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。

UDPPProvider 代码:

```
package UDP;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.UUID;
public class UDPPProvider {
    public static void main(String[] args) throws IOException {
        // 创建接受者端的 datagramsocket 并指定接口
        DatagramSocket datagramSocket = new DatagramSocket(9091);
        // 创建数据报 用于接收客户端传来的数据
        byte[] buf = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(buf, 0, buf.length);
        // 接收客户端发来的数据 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);
        // 解析数据 取得 新的 port
        int len = receivePacket.getLength();
        String newPort = new String(receivePacket.getData(), 0, len);
        int port = MessageUtil.parsePort(newPort);
        System.out.println(port);
        String tag = UUID.randomUUID().toString();
        String data = MessageUtil.buildWithTag(tag);
        //取得 新的 ip
        String newIP = receivePacket.getAddress().getHostAddress();
        System.out.println("我是接受者," + newIP + ":" + port + "的发送者说:" +
            new String(receivePacket.getData(), 0, len));
        //准备回送数据
        byte[] responseDataBytes = data.getBytes();
        // 创建数据报 用于发回给发送端
        DatagramPacket responsePack = new DatagramPacket(responseDataBytes,
            responseDataBytes.length, receivePacket.getAddress(), port);
        datagramSocket.send(responsePack);
        // 关闭 UDPPProvider
        datagramSocket.close();
    }
}
```

UDPSearcher 代码:

```
package UDP;

import java.io.IOException;
import java.net.*;
public class UDPSearcher {
    public static void main(String[] args) throws IOException {
        DatagramSocket datagramSocket = null;
        try {
            datagramSocket = new DatagramSocket();
        } catch (SocketException e) {
            e.printStackTrace();
        }

        // 定义要发送的数据
        String sendData = MessageUtil.buildWithPort(30000);
        byte[] sendBytes = sendData.getBytes();

        // 发送至localhost: 9091
        DatagramPacket sendPack = null;
        try {
            sendPack = new DatagramPacket(sendBytes, sendBytes.length,
                                           InetAddress.getByName("255.255.255.255"), 9091);
            datagramSocket.send(sendPack);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("数据发送完毕.....");

        // 准备接收 Provider 的回送数据
        try {
            datagramSocket = new DatagramSocket(30000);
        } catch (SocketException e) {
            e.printStackTrace();
        }
        byte[] buf = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(buf, buf.length);
        datagramSocket.receive(receivePacket);
        // 查看 receiverPacket 的相关信息

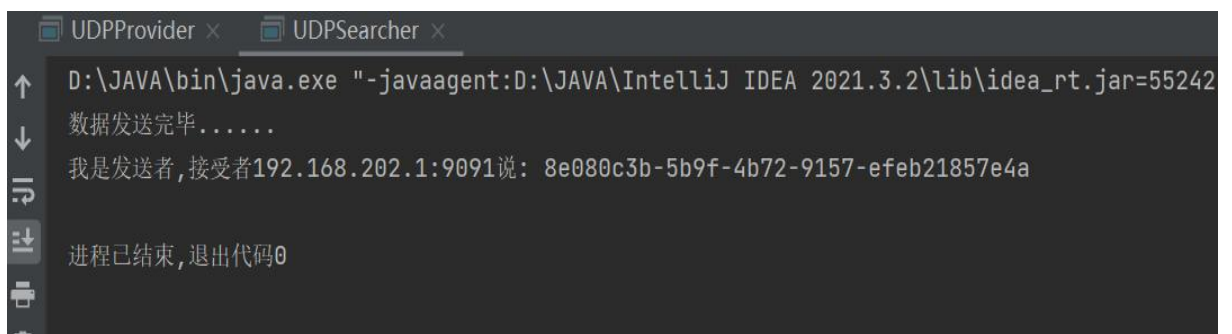
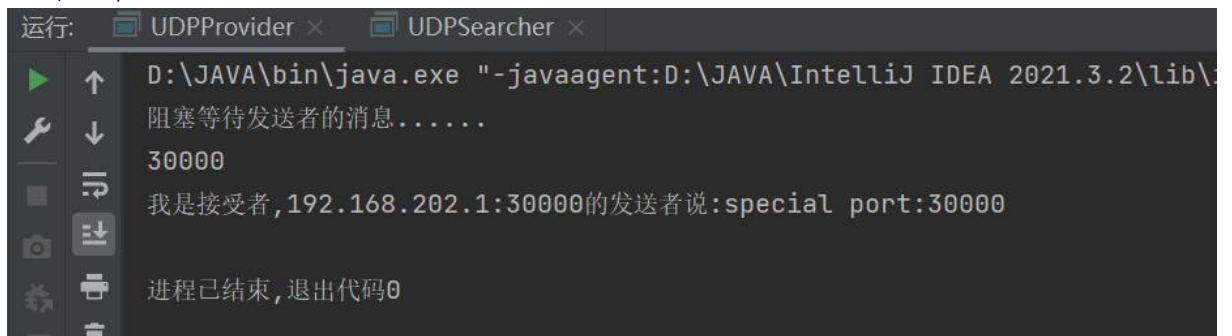
        String ip = receivePacket.getAddress().getHostAddress();
        int port = receivePacket.getPort();
        int len = receivePacket.getLength();
        String data = new String(receivePacket.getData(), 0, len);
        System.out.println("我是发送者," + "接受者" + ip + ":" + port + "说: " +
            MessageUtil.parseTag(data));
        // 关闭 UDPSearcher
        datagramSocket.close();
    }
}
```

MessageUtil 代码:

```
package UDP;
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;
    }
}
```

运行结果:



五、总结

通过本次实验掌握了使用 DatagramSocket 和 DatagramPacket 编写代码, 实现了基于 UDP 的 Socket 通信。