****

分布式与云计算

实验报告

|  |  |
| --- | --- |
| 姓 名： | 谭哲文 |
| 学 号： | 8202191123 |
| 学 院： | 计算机学院 |
| 专业班级： | 计科2105 |
| 指导教师： | 余腊生 |

本科生院制

2024年6月

**实验一 数据包 Socket 应用**

**1.实验目的**

1. 理解数据包 socket 的应用

2. 实现数据包 socket 通信

3. 了解 Java 并行编程的基本方法

**2. 实验要求**

1. 预习实验指导书及教材的有关内容，了解数据包 socket 的通信原理；

2. 熟悉一种 Java IDE 和程序开发过程；

3. 了解下列 Java API:Thread、Runnable;

4. 尽可能独立思考并完成实验。

**3. 实验原理**

1. UDP协议

UDP（User Datagram Protocol，用户数据报协议）是一种无连接的传输层协

议。它主要用于不要求分组顺序到达的传输中，分组传输顺序的检查与排序由应

用层完成 ，提供面向事务的简单不可靠信息传送服务。由于传输数据不建立连

接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向

多个客户机传输相同的消息。具有资源消耗小，处理速度快的优点，所以通常音

频、视频和普通数据在传送时使用UDP协议。

2. java.net.DatagramSocket类

DatagramSocket是java针对UDP协议实现的api，它的作用是接收和发送数据包。

DatagramPacket 是传输的数据报，其中封装了数据，其通信流程如下：

3. 实验原理

1. UDP协议

UDP（User Datagram Protocol，用户数据报协议）是一种无连接的传输层协

议。它主要用于不要求分组顺序到达的传输中，分组传输顺序的检查与排序由应

用层完成 ，提供面向事务的简单不可靠信息传送服务。由于传输数据不建立连

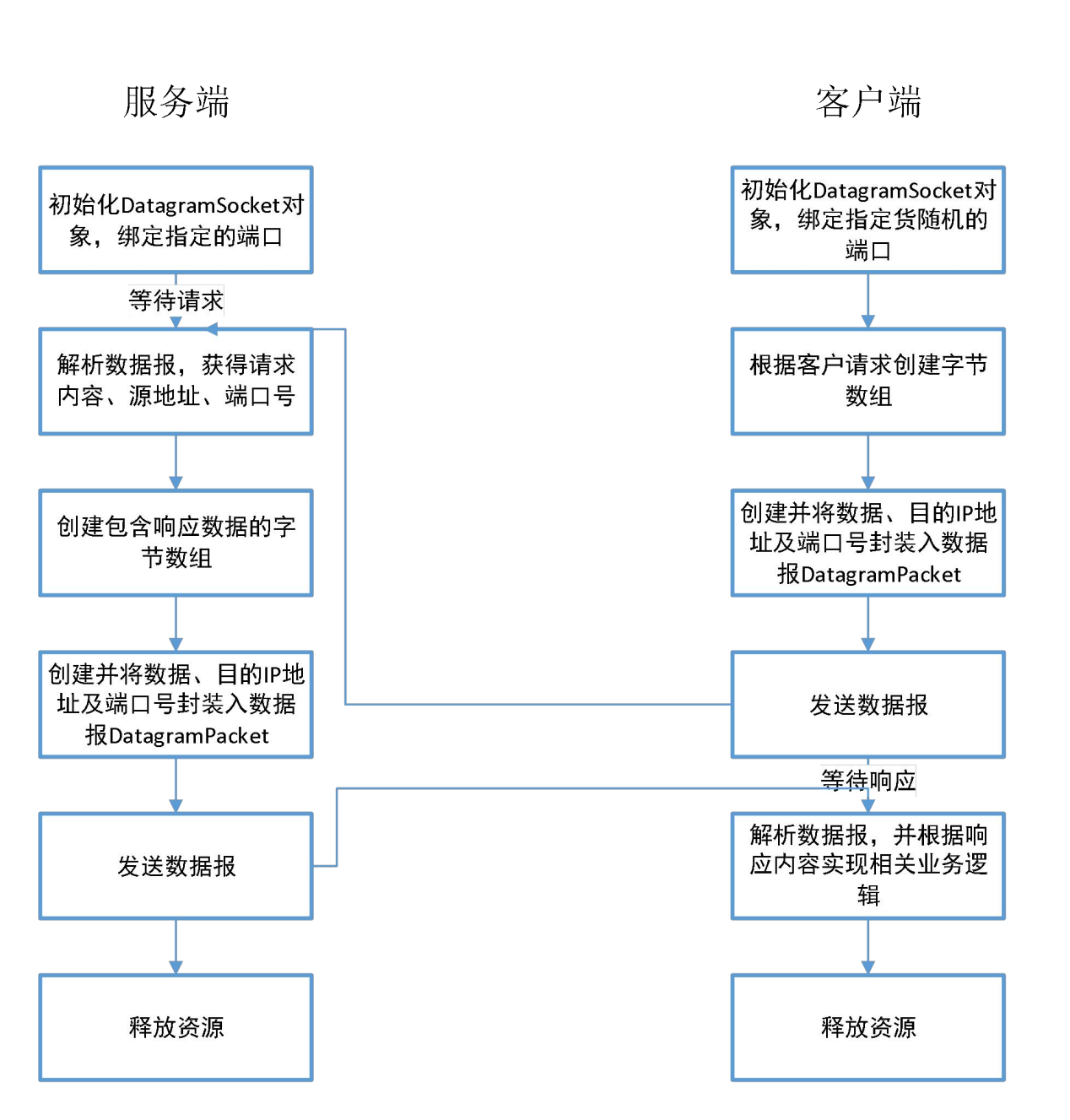
接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。具有资源消耗小，处理速度快的优点，所以通常音

频、视频和普通数据在传送时使用UDP协议。

2. java.net.DatagramSocket类

DatagramSocket是java针对UDP协议实现的api，它的作用是接收和发送数据包。

DatagramPacket 是传输的数据报，其中封装了数据，其通信流程如下：

**4. 代码设计**

本次实验主要是实现UDP协议下的echo功能，服务端会返回客户端发送的数

据。

4.1 服务端

服务端在初始化DatagramSocket需要指定端口号来监听，这样客户端才能根

据服务端的端口号确定目的地址，初始化字节数组封装入数据报对象，调用api

接收数据报，解析数据报得到客户端的源地址和端口，根据客户发来的数据，再

将字节数组封装入数据报发送回客户端。

public Server() throws SocketException, IOException {

// 指定端口

DatagramSocket datagramSocket = new DatagramSocket(8000);// 初始化字节数组

byte[] message = new byte[1024];

// 封装数据报

DatagramPacket clientPacket = new DatagramPacket(message, 1024);

// 接收请求报文

datagramSocket.receive(clientPacket);

System.out.println("Client(" + clientPacket.getAddress().getHostAddress() + ":

"

+ clientPacket.getPort() + "):" + new String(message));

// 响应报文

DatagramPacket serverPacket = new DatagramPacket(message, 0, message.l

ength, clientPacket.getAddress(),clientPacket.getPort());

System.out.println("Server:" + new String(message));

// 发送响应报文

datagramSocket.send(serverPacket);

// 关闭Socket,释放资源

datagramSocket.close()；

}

4.2 客户端

客户端不同与服务端，创建 Socket 时可以不指定端口号，jdk 会随机分配空

闲的端口。客户端获取客户的请求转为字节数组封装入数据报，进行发送。服务

端会根据原来的端口发送响应报文，接收响应报文后进行解析，释放资源。

public Client() throws IOException {

// 初始化Socket

DatagramSocket datagramSocket = new DatagramSocket();

System.out.print("Client:");

// 接收信息

Scanner scanner = new Scanner(System.in);

String message = scanner.nextLine();

// 将信息封装入数据报DatagramPacket ClientPacket = new DatagramPacket(message.getBytes(), me

ssage.length(),

InetAddress.getByName("127.0.0.1"), 8000);

// 发送数据报

datagramSocket.send(ClientPacket);

// 初始化服务端返回的数据报

DatagramPacket serverPacket = new DatagramPacket(new byte[1024], 1024);

// 接收数据报

datagramSocket.receive(serverPacket);

System.out.println("Server:" + new String(serverPacket.getData()));

// 关闭socket,scanner

scanner.close(); .

datagramSocket.close();

}

**5.实验结果**

服务端



客户端



**6. 思考题**

1. 如何避免数据包丢失而造成的无限等待问题？

可以采用以下方法来避免：

1. 设置超时时间： 发送方在发送数据包后启动一个定时器，等待接收方的

响应。如果在规定的超时时间内未收到响应，认为数据包丢失，触发重传机制。2. 重传次数限制： 为每个数据包设置一个重传次数的上限。如果数据包在

规定次数内仍未被确认，认为无法成功接收，发送端将触发重传。

3. 序号标识：在数据包中引入序号字段，确保每个数据包都有唯一的标识。

接收方可以根据序号对接收到的数据包进行排序，避免处理重复接收的数据。

4. 去重处理： 接收方在接收到数据包后，通过序号等信息判断是否是重复

接收的数据包，如果是则进行丢弃，避免对同一数据的重复处理。

2. 如何实现全双工的数据包通信？

建立双向的套接字（Socket）连接来实现的。套接字会绑定主机的端口，能

够实现接收和发送数据报，主机可以通过解析数据报获得源地址和端口，来进行

响应，发送回发送请求的主机地址。在这种连接中，两个通信方都可以独立地同

时进行发送和接收操作，实现了双向的数据传输。