**【实验名称】：UDP协议网络编程**

**学生姓名：苏家铭 合作学生：无**

**实验地点：济世楼330 实验时间：2023年9月27日**

**【实验目的】**

* **理解客户机服务器模型，了解端口在网络传输中的作用。**
* **了解无连接通信方式及编程方式**
* **了解掌握基于Socket的UDP网络编程的基本步骤**

**【实验原理】**

**两个网络程序进行简单通信，基本功能是客户机从发送窗口发送任意一段字符串给服务器，服务器接收来自客户机的字符串，然后将字符串返回给客户机。**

**UdpClient.java代码解析：**

**该程序创建了一个简单的图形用户界面（GUI）来实现一个UDP客户端，允许用户连接到指定的服务器并通过GUI界面发送和接收消息。以下是程序的主要部分和功能：**

**1. `UdpClient` 类继承自 `JFrame`，表示GUI窗口。**

**2. 在 `main` 方法中，创建了一个 `UdpClient` 对象并设置其大小和可见性，以便在程序启动时显示GUI窗口。**

**3. GUI界面包含两个主要部分，分别位于 `contentPanel2` 和 `contentPanel` 中，分别用于接收和发送消息。**

**4. 用户可以在 `textFieldIP` 文本框中输入服务器的IP地址。**

**5. 当用户点击 "连接" 按钮 (`btnConnect`) 时，程序尝试解析用户输入的IP地址，并创建一个 `DatagramPacket` 对象来准备接收服务器的响应。之后，启用 "发送" 按钮 (`btnSend`)。**

**6. 当用户点击 "发送" 按钮时，程序将用户在 `textMessage` 文本框中输入的文本转换为字节数组，并将其作为UDP数据包发送到指定的服务器地址（在端口5555上）。**

**7. 一旦数据包发送成功，程序等待接收来自服务器的响应数据包，并将其显示在 `textChat` 文本区域中。**

**8. 用户可以点击 "关闭" 按钮 (`btnExit`) 来退出程序，这将关闭UDP连接。**

**UdpServer.java代码解析：**

**该程序实现了一个UDP（User Datagram Protocol）服务器，用于接收来自客户端的消息并将其回显给客户端。服务器侦听端口5555，等待来自客户端的连接。以下是程序的主要部分和功能：**

1. **UdpServer 类包含了 main 方法，用于启动UDP服务器。**
2. **在构造函数中，程序创建了一个 DatagramSocket 对象，绑定到端口号5555，这是UDP服务器将侦听的端口。**
3. **服务器开始侦听端口5555，并在控制台输出 "UDP Server Start, wait on Port 5555 ..."，表示服务器已经启动。**
4. **然后，程序创建一个 DatagramPacket 对象 request，用于接收来自客户端的UDP数据包。这个数据包将用于接收客户端发送的消息。**
5. **服务器进入一个无限循环，持续监听端口5555上的UDP数据包。一旦接收到数据包，它会将数据包中的内容转换为字符串，并在控制台上输出 "Request:" 后面跟着接收到的消息内容。**
6. **然后，服务器尝试向客户端发送相同的数据包（即将接收到的数据包内容返回给客户端）。这是一个简单的回显操作，将客户端发送的消息返回给客户端。**
7. **如果发生 SocketException 或 IOException 异常，程序将捕获并处理这些异常，但是在代码中并没有具体的处理逻辑。**

**【实验设备】**

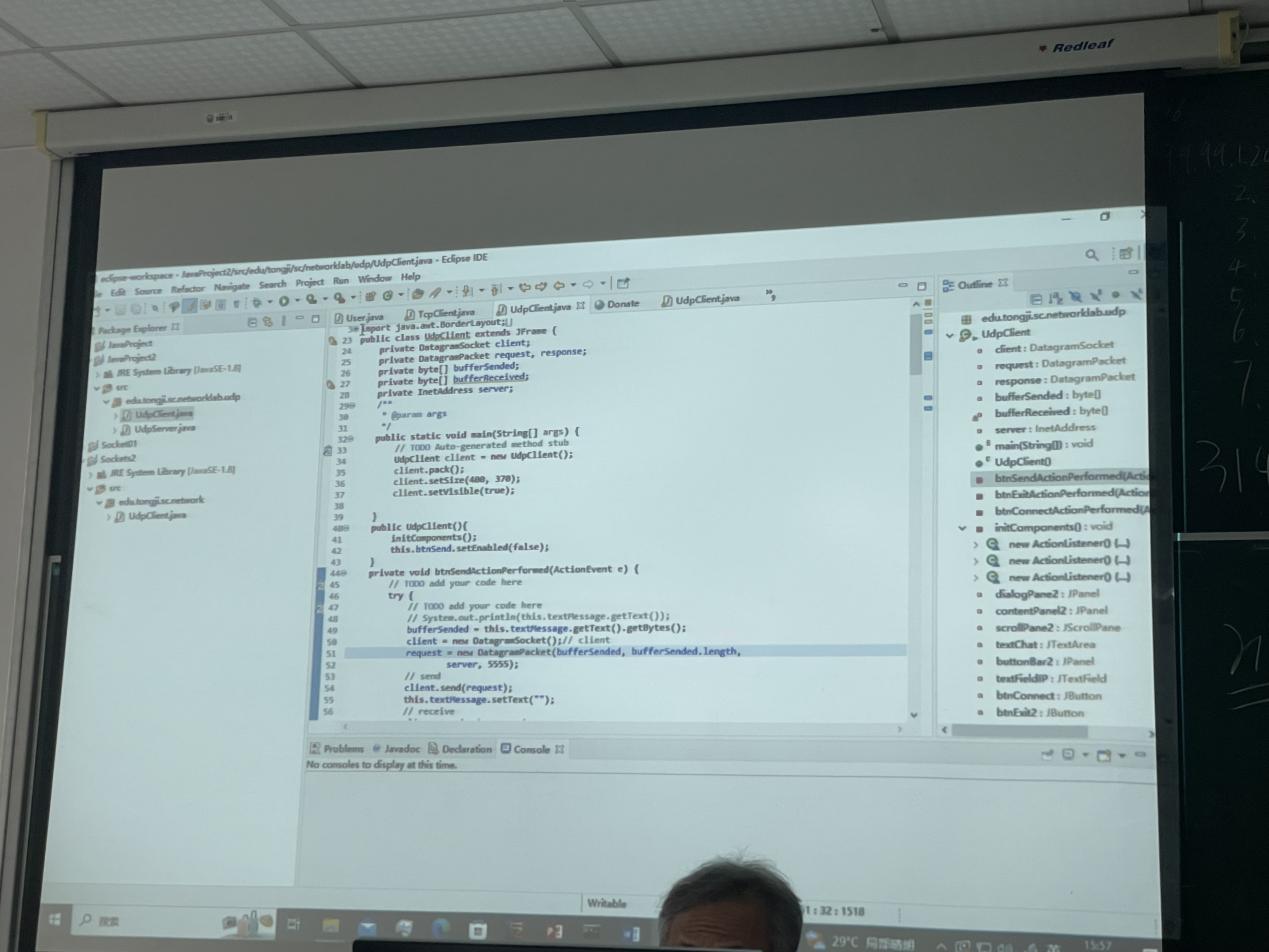
**硬件：济世楼330机房电脑**

**IDE:Eclipse**

**【实验步骤】**

**在Eclipse中新建project运行两份代码UdpServer.java和UdpClient.java，运行这两份代码将实现一个简单的UDP客户端和UDP服务器，允许客户端向服务器发送消息，并且服务器将接收这些消息并将它们回显给客户端。**

1. **启动UDP服务器（UdpServer）：在运行服务器代码后，服务器成功绑定到5555端口并开始等待客户端的连接。**
2. **启动UDP客户端（UdpClient）：客户端程序创建了一个简单的GUI窗口，允许用户输入服务器的IP地址和要发送的消息。用户可以输入服务器的IP地址（默认为127.0.0.1），然后点击 "连接" 按钮。**
3. **客户端连接成功后，用户可以在文本框中输入要发送的消息，并点击 "发送" 按钮。服务器将接收到的消息回显到客户端的文本区域中。**

****

**【实验现象】**

**下面是运行这两份代码时的结果：**

**运行UDP服务器（UdpServer）：**

**在服务器端启动时，你将在控制台看到消息 "UDP Server Start, wait on Port 5555 ..."，表示服务器已经启动并正在监听端口5555。**

**当客户端发送消息时，服务器将在控制台上显示 "Request:" 后跟随接收到的消息内容。**

**运行UDP客户端（UdpClient）：**

**客户端程序将启动一个图形用户界面（GUI）窗口，允许用户输入服务器的IP地址和要发送的消息。**

**用户可以在文本框中输入服务器的IP地址，然后点击 "连接" 按钮以连接到服务器。**

**一旦连接成功，用户可以在文本框中输入消息并点击 "发送" 按钮以将消息发送到服务器。**

**服务器将接收到的消息回显给客户端，并显示在客户端的文本区域中。**

**实验结果表明，客户端能够成功连接到服务器，发送消息并接收到服务器的响应。服务器能够成功接收来自客户端的消息，并将其回显给客户端。**

**【分析讨论】**

1. **协议分析：UDP协议是一种无连接的协议，它不保证数据传输的可靠性，但具有低延迟和高效率的特点。在实时传输和游戏等应用中，UDP常被用于快速传输数据。与TCP协议相比，UDP协议不提供连接的建立和断开，也不进行数据包的重传，因此在可靠性方面较差，但在某些场景下的性能更优越。**
2. **通过这个实验，我更深入地了解了UDP协议的工作方式，以及如何使用Java编程实现UDP客户端和服务器。我观察到UDP的快速传输特性，适用于需要快速实时通信的应用，但在丢包和数据完整性方面存在风险。实验还强调了服务器与客户端之间的简单请求-响应通信模式，其中服务器接收并回显客户端的消息。**
3. **通过这个实验，我成功地掌握了UDP协议的基本原理和实际应用。我了解了UDP协议的优点和局限性，并知道在何种情况下选择使用UDP而不是TCP。此外，我还熟悉了使用Java编程实现UDP通信的基本步骤。这些知识对于网络编程和实时通信应用的开发非常重要，将为我未来的项目提供有用的基础。**