# 实验4：TCP/IP编程

## UDP

## Client端主要步骤：

1. 创建套接字

ClientSocket = socket(AF\_INET, SOCK\_DGRAM)

1. 创建具有serverIP和port = x的数据报；经clientSocket发送数据报
2. 从clientSocket接受数据报
3. 关闭clientSocket

## Server端主要步骤：

1. 创建套接字，

Post = x；serverSocket = socket(AF\_INET, SOCK\_DGRAM)

1. 从serverSocket读UDP报文段
2. 向serverSocket写响应指定客户地址和端口号

## 主要类及作用

1. java.net.DatagramSocket class

是一个UDP的接口, 该类表示用于发送和接收数据包的套接字.

通过该类中的 send() 函数来发送UDP数据包

通过该类中的 receive() 函数来接收UDP数据包

1. java.net.DatagramPacket class

该类用于表示一个数据报包

该类的可以通过字节数组及其长度来实例化DatagramPacket对象

1. java.net.InetAddress class

该类表征一个IP地址

该类的实例化对象可以通过 getByName(String host) 来获得

1. Java.util.Scanner class

该类用于扫描数据流

## TCP

## Client端主要步骤：

1. 创建套接字

ClientSocket = socket(AF\_INET, SOCK\_DGRAM)

1. 创建具有serverIP和port = x的数据报；经clientSocket发送数据报
2. 从clientSocket接受数据报
3. 关闭clientSocket

## Server端主要步骤：

1. 对到来的请求创建套接字

Post = x；serverSocket = socket(AF\_INET, SOCK\_DGRAM)

1. 等待入连接请求

connectionSocket = serverSocket.accept()

1. 从connectionSocket读请求
2. 向connectionSocket写响应
3. 关闭connectionSocket

## 主要类及作用

1. java.net.Socket class

此类实现TCP客户机套接字（也称为“套接字”）。套接字是两台机器之间通信的端点。

1. java.io.BufferedReader class

字符输入流

1. java.io.InputStreamReader class

字节输入流

1. java.io.DataOutputStream class

字节输出流

## TCP与UDP主要差异和特点:

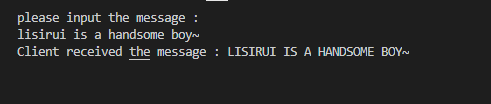
UDP: 在发送进程能够将数据分组推出套接字的门之前, 当使用UDP时, 必须将目的地址附在该分组之上. 在该分组穿过发送方的套接字之后, 因特网将使用该目的地址通过因特网为该分组选路到接收进行的套接字. 当分组到达接收套接字时, 接收进程将通过该套接字取回分组, 然后检查分组的内容并采取适当的动作.

TCP: 与UDP不同, TCP是一个面向连接的协议. 这意味着在客户和服务器能够给开始相互发送数据之前,它们先要握手和创建一个TCP连接. TCP连接的一端与客户套接字相连, 另一端与服务器套接字相连. 当创建TCP连接时, 我们将其与客户端套接字地址(IP 和 端口号) 和服务器套接字地址关联起来. 使用创建的TCP连接, 当一侧要向另一侧发送数据时, 他只需经过其套接字将数据丢进TCP连接即可. 这与UDP不同, UDP服务器再讲分组丢进套接字之前必须为其附上一个目的地址.

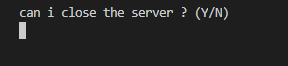
## 实验结果:

UDP:

Client



Server:



TCP:

Client:



Server:

