1. 网络编程三要素
2. IP地址

网络中设备的唯一标识。

1. 端口

网络通信中应用程序的标识，

1. 协议

网络通信的协议 ，对数据的传输格式，传输速率等都有统一规定。

二． IP地址

（一） IPv4

给每个连接在网络上的主机分配一个32bit的地址，也就是4个字节， 为了方便使用，常常表示为十进制，，称为点分十进制表示法。

（二） IPv6

为了扩大地址空间，采用新的地址IPv6，用128bit表示，每16个字节一组，分成8个十六进制数。

（三）InetAddress类

此类表示IP地址。

常用方法

方法名 说明

static InetAddress getByName(String

host)

确定主机名称的IP地址。主机名称可以是机器名称，也可以

是IP地址

String getHostName() 获取此IP地址的主机名

String getHostAddress() 返回文本显示中的IP地址字符串

三． 端口

用两个字节表示的整数，取值范围是0-65535.0-1023之间的端口号用于知名的应用程序，若端口号被另一个服务或者应用占用，则当前程序会启动失败。

四．协议

（一）UDP协议

1. 概述

用户数据报协议 (User Datagram Protocol)。UDP是无连接通信协议，在数据传输时，数据的发送端和接收端不建立逻辑连接。发送端不会确认接收端的状态就直接发送，接收端收到数据也不会向发送端发送反馈。

2.应用

由于使用 UDP协议消耗资源小，通信效率高，所以通常都会用于音频、视频和普通数据的传输

（二）TCP协议

1. 概述

传输控制协议 (Transmission Control Protocol)

TCP 协议是面向连接的通信协议，即传输数据之前，在发送端和接收端建立逻辑连接，然后再传输数据，它提供了两台计算机之间可靠无差错的数据传输。在TCP连接中必须要明确客户端与服务器端，由客户端向服务端发出连接请求，每次连接的创建都需要经过“三次握手”

2.三次握手

在发送数据的准备阶段，客户端与服务器之间的三次交互，以保证连接的可靠性。

1. 第一次握手，客户端向服务器端发出连接请求，等待服务器确认
2. 第二次握手，服务器端向客户端回送一个响应，通知客户端收到了连接请求
3. 第三次握手，客户端再次向服务器端发送确认信息，确认连接

五．UDP通信实例

（一）UDP发送数据

在通信的两端各建立一个Socket对象，用于发送和接受数据。DatagramSocket类是基于UDP协议的Socket。

1. 构造方法

方法名 说明

DatagramSocket()

创建数据报套接字并将其绑定到本机地址上的任

何可用端口

DatagramPacket(byte[] buf,int len,InetAddress

add,int port)

创建数据包,发送长度为len的数据包到指定主机

的指定端口

1. 常用方法

方法名 说明

void send(DatagramPacket p) 发送数据报包

void close() 关闭数据报套接字

1. 步骤
2. 创建发送端的Socket对象。
3. 创建数据并打包
4. 发送数据
5. 关闭发送端
6. 示例

package 网络编程;  
  
import java.io.IOException;  
import java.net.\*;  
  
public class UDP发送数据 {  
 public static void main(String[] args) throws IOException {  
 DatagramSocket ds = new DatagramSocket();  
 byte[] bys = "hello world".getBytes();  
 DatagramPacket dp = new DatagramPacket(bys, bys.length, InetAddress.*getByName*("LAPTOP-R2809UF4"), 10086);  
  
 ds.send(dp);  
 ds.close();  
 }  
}

（二）UDP接受数据

1. 常用方法

方法名 说明

byte[] getData() 返回数据缓冲区

int getLength() 返回要发送的数据的长度或接收的数据的长度

void receive(DatagramPacket p) 从此套接字接受数据报包

2. 示例

package 网络编程;  
  
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.SocketException;  
  
public class UDP接受数据 {  
 public static void main(String[] args) throws IOException {  
 DatagramSocket ds = new DatagramSocket(10086);  
 byte[] bys = new byte[1024];  
 DatagramPacket dp = new DatagramPacket(bys, bys.length);  
 ds.receive(dp);  
 byte[] data = dp.getData();  
 int len = dp.getLength();  
 String dataString = new String(data, 0, len);  
 System.*out*.println("数据时："+ dataString);  
  
 ds.close();  
  
 }  
}

六．TCP通信实例

（一）TCP发送数据

TCP为客户端提供了Socket类，为服务端提供了ServerSocket类。

1. 构造方法

方法名 说明

Socket(InetAddress address,int port) 创建流套接字并将其连接到指定IP指定端口号

Socket(String host, int port) 创建流套接字并将其连接到指定主机上的指定端口号

1. 常用方法

方法名 说明

InputStream getInputStream() 返回此套接字的输入流

OutputStream getOutputStream() 返回此套接字的输出流

1. 步骤
2. 创建客户端的Socket对象。
3. 获取输出流，写数据
4. 客户端使用输入流接受服务器端的反馈数据
5. 关闭客户端
6. 示例

package 网络编程;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.net.Socket;  
  
public class TCP发送数据 {  
 public static void main(String[] args) throws IOException {  
 //创建客户端的Socket对象  
 Socket s = new Socket("192.168.0.106", 6000);  
  
 //获取输出流，写数据  
 OutputStream os = s.getOutputStream();  
 os.write("hello world".getBytes());

//接收服务器反馈

    InputStream is = s.getInputStream();

    byte[] bys = new byte[1024];

    int len = is.read(bys);

    String data = new String(bys, 0, len);

    System.out.println("客户端：" + data);

//释放资源  
 s.close();  
 }  
}

（二）TCP接受数据

1. 构造方法

方法名 说明

ServletSocket(int port) 创建绑定到指定端口的服务器套接字

2. 常用方法

方法名 说明

Socket accept() 监听要连接到此的套接字并接受它

3. 步骤

（1）创建服务器端的ServerSocket对象。

（2）监听客户端的连接

（3）获取输入流，读数据

（4）服务器端通过输出流向客户端反馈。

（5）关闭服务器端

4.示例

package 网络编程;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class TCP接受数据 {  
 public static void main(String[] args) throws IOException {  
 //创建服务器端的socket  
 ServerSocket ss = new ServerSocket(6000);  
   
 //监听客户端的链接  
 Socket s = ss.accept();  
  
 //获取输入流，读数据  
 InputStream is = s.getInputStream();  
 byte[] bys = new byte[1024];  
 int len = is.read(bys);  
 String data = new String(bys, 0, len);  
 System.*out*.println("数据是：" + data);

//给出反馈

    OutputStream os = s.getOutputStream();

    os.write("数据已经收到".getBytes());

s.close();  
 ss.close();  
 }  
}

（三）注意事项

如何记忆TCP发送数据与接受数据的输入流，输出流的顺序。客户端发送数据时，使用的是输出流，因为是向外发送；发送到服务器端，服务器端用输入流接受，因为数据是从外向里的。反馈是从服务器端开始发送，同样向外发送都是使用输出流，客户端接受这个反馈时使用的是输入流。