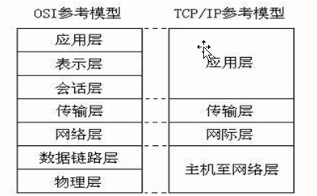
# 网络编程

网络参考模型:



应用层协议：HTTP/HTTPS协议

传输层协议：TCP/UDP协议，ip协议。

主机到网络层：传输介质：光纤，蓝牙，红外等。

网络中的数据传输：封包到拆包的过程。

ip地址 ：

网络中唯一表示设备

127.0.0.1 本地ip地址。

192.168.1.1~255 保留作为局域网地址段

ipV4已经不够用了，

端口：

用于标识不同进程的逻辑地址。

有效端口:1~65535,其中0~1024为系统端口或保留端口

传输协议：

国际通用的传输的协议 :TCP/ip

局域网的协议。

为了安全起见，一些组织定义了自己的协议。其他用户无法连接进来。

# TCP和UDP

UDP(User Datagram Protocol,用户数据报协议) (聊天,视频共享)

不需要建立连接,即面向无连接

数据会被封包(数据源和目的),大小都限制在64K内

因为是无连接,是不可靠协议

不需要建立连接,速度快

TCP (Transmission Control Protocol,传输控制协议)(下载)

建立连接,形成传输数据的通道

在连接中可以进行大数据量传输

通过三次握手完成连接,是可靠协议

必须建立连接,效率会稍低

# Socket(插座,套接字)

Socket为网络服务提供的机制

应用程序间通信,两端必须都有Socket,其实就是socke间的通信

数据在两个Socket见通过io传输

# UDP传输(客户端)

/\*

编写一个聊天程序。有收数据的部分，和发数据的部分。

这两部分需要同时执行。那就需要用到多线程技术。

一个线程控制收，一个线程控制发。

因为收和发动作是不一致的，所以要定义两个run方法。

而且这两个方法要封装到不同的类中。

\*/

**public** **class** ChatDemo {

**public** **static** **void** main(String[] args) **throws** SocketException {

DatagramSocket sendSocket = **new** DatagramSocket();

DatagramSocket receSocket = **new** DatagramSocket(5000);

**new** Thread(**new** Send(sendSocket)).start();

**new** Thread(**new** Receiver(receSocket)).start();

}

}

//发送UDP数据的线程

**class** Send **implements** Runnable{

**private** DatagramSocket socket;

//1. 接受udpSocket服务.(创建UDPSocket服务)

**public** Send(DatagramSocket socket){

**this**.socket = socket;

}

**public** **void** run() {

//输入数据

BufferedReader reader=**new** BufferedReader(

**new** InputStreamReader(System.***in***));

String line=**null**;

**try** {

**while**((line = reader.readLine())!=**null**){

//2.封装数据到数据到数据包中.

**byte** []bs= line.getBytes();

DatagramPacket packet = **new** DatagramPacket

(bs,bs.length,InetAddress.*getLocalHost*()

, 5000);

//3. 通过socket服务发送数据

socket.send(packet);

**if** ("886".equals(line)) {

**break**;

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

**finally** {

socket.close();

}

}

}

//接口消息的线程

**class** Receiver **implements** Runnable{

**private** DatagramSocket socket1;

//1. 定义udpsocket服务。并监听一个端口.

**public** Receiver(DatagramSocket socket1){

**this**.socket1 = socket1;

}

**public** **void** run() {

//2. 定义数据包,存储接受到字节数据

**byte**[] buf = **new** **byte**[1024];

DatagramPacket packet= **new** DatagramPacket(

buf, buf.length);

**while**(**true**){

**try** {

//3. 通过socket服务的receive方法将收到的数据存入已定义好的数据包中。

socket1.receive(packet);

// 4. 通过数据包对象的特有功能。将这些不同的数据取出。打印在控制台上。

String ip=packet.getAddress().getHostName();

String data = **new** String(packet.getData(),

0,packet.getLength());

System.***out***.println(ip+":"+data);

**if**("886".equals(data)){

System.***out***.println(ip+"....离开聊天室");

**break**;

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

}

# TCP传输

/\*

\* TCP客户端Socket 服务器端 ServiceSocket

\* 客户端

\* 1.建立socket服务.指定要连接的主机ip地址和端口

\* 2.获取socket流中的输出流s.getOutputStream(),将数据发送到服务端

\* 3.获取socket的输入流, s.getInputStream(),获取服务端返回的数据

\* 4.关闭客户端资源

\* 服务器端

\* 1.建立服务器端服务器socket服务 ,ServerSocket() 并监听一个端口

\* 2.获取连接过来的socket对象 ,通过accept方法,该方法为阻塞式方法

\* 3.通过服务端对象,获取输入流和输出流,

\* 4.关闭流资源 (可选操作)

\*/

//客户端

**class** Test {

**public** **static** **void** main(String[] args) **throws** Exception {

String hoString = InetAddress.*getLocalHost*().getHostAddress();

System.***out***.println(hoString);

Socket socket = **new** Socket(hoString,5000);

OutputStream os = socket.getOutputStream();

os.write("你好,服务器端".getBytes());

**byte** [] buf= **new** **byte**[1024];

**int** len = socket.getInputStream().read(buf);

String string = **new** String(buf, 0, len);

System.***out***.println(string);

os.close();

socket.close();

}

}

**class** Rece{

**public** **static** **void** main(String[] args)**throws** Exception {

ServerSocket ss = **new** ServerSocket(5000);

Socket s = ss.accept();

String name = s.getInetAddress().getHostName();

**byte** [] bs = **new** **byte**[1024];

**int** len = s.getInputStream().read(bs);

System.***out***.println(**new** String(bs,0,len)+name);

s.getOutputStream().write("你好客户端".getBytes());

s.close();

ss.close();

}

}