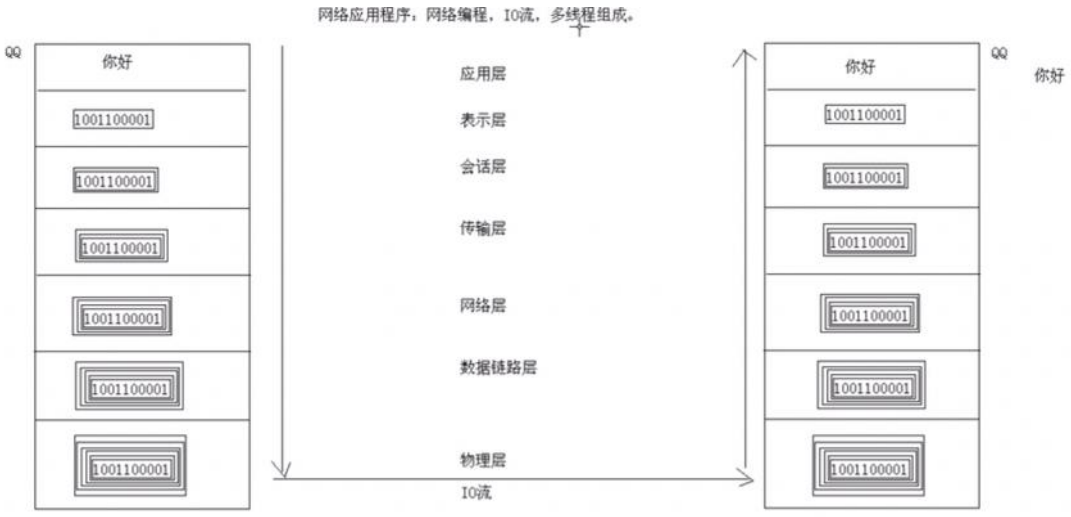
网络模型

2018年7月30日 15:50

OSI参考模型
TCP/IP参考模型

应用层
表示层
会话层
传输层
网络层
数据链路层
物理层



网络编程三要素

2018年7月30日 16:21

网络编程三要素
1.IP地址 找到要传输的对象
2.端口 找到要传输的应用程序
3.协议 按照某种规则传输

IP地址：网络中计算机的唯一标识
计算机只能识别二进制，但配置的IP地址不是二进制
IP：192.168.1.100
换算：11000000 10101000 00000001 01100100
点分十进制：把IP地址的每一个字节上的数据换算成十进制，用点隔开

IP地址的组成：网络号段+主机号段

IP地址的分类：
A类IP：从0.0.0.0 – 127.255.255.255，共有16777216个IP 第一号段为网络号段+后三段的主机号段 256*256*256
B类IP：从128.0.0.0 – 191.255.255.255，共有65536个IP 前两段为网络号段+后二段的主机号段 256*256 校园网
C类IP：从192.0.0.0 – 223.255.255.255，共有256个IP 前三段为网络号段+后一段的主机号段 192.168.X.X是私有地址
D类IP：从224.0.0.0 – 239.255.255.255
E类IP：从2240.0.0.0 – 255.255.255.255

两个DOS命令：
ipconfig 查看本机地址
ping 后面跟ip地址，测试本机与指定ip地址间的通信是否有问题
特殊的IP地址：127.0.0.1回环地址（表示本机） X.X.X.255 广播地址 X.X.X.0网络地址
ping本机网络环境

端口号：
逻辑端口：每个网络程序都至少会有一个逻辑端口
正在运行的程序标识
有效端口：0~95535，其中0~1024系统使用或保留端口

协议：通信的规则
UDP：把数据打包，数据有限制64k，不建立连接，速度快，不可靠 如：发短信，QQ
TCP：三次握手协议，建立连接通道，数据无限制，速度慢，可靠 如：打电话

InetAddress类

2018年7月30日 17:34

```
/*
 * 如果一个类没有构造方法：
 * 1.成员全部是静态(Math,Arrays,Collections)
 * 2.单例设计模式(Runtime)
 * 3.类中有静态方法返回该类对象
 *
 * 成员方法：public static InetAddress getByNames(String host)
 *                          根据主机名或者IP地址的字符串表示得到IP地址对象
 */
public class InetAddressDemo {
	public static void main(String[] args) throws UnknownHostException {
		//InetAddress address = InetAddress.getByName("DESKTOP-4O0BFJQ");
		InetAddress address = InetAddress.getByName("192.168.31.118");

		//获取主机名和IP地址
		String name = address.getHostName();
		String ip = address.getHostAddress();
		System.out.println(name+"---"+ip);
	}
}
```

Socket

2018年7月30日 17:34

Socket:网络套接字
Socket编程：网络编程，套接字编程

Socket包含了IP地址+端口
通信的两端都有Socket
网络通信其实就是socket间的通信
数据在两个Socket间通过IO传输

UDP协议发送数据

2018年7月30日 22:15

```
/*
 * UDP协议发送数据:
 * 1.创建发送端的Socket对象
 * 2.创建数据并把数据打包
 * 3.调用Socket对象的发送方法发送数据包
 * 4.释放资源
 */
public class SendDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket();

		// 创建数据
		byte[] bys = "hello,udp".getBytes();
		// 长度
		int length = bys.length;
		// IP地址对象
		InetAddress address = InetAddress.getByName("DESKTOP-4O0BFJQ");
		// 端口
		int port = 10086;
		// 创建数据
		DatagramPacket dp = new DatagramPacket(bys, length, address, port);

		ds.send(dp);

		ds.close();
	}
}
```

UDP协议接收数据

2018年7月30日 22:15

```
/*
 * UDP协议接收数据:
 * 1.创建发送端的Socket对象
 * 2.创建数据包(接收容器)
 * 3.调用Socket对象的接收方法接收数据包
 * 4.解析数据包，并显示在数据台
 * 5.释放资源
 */
public class ReceiveDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket(10086);

		byte[] bys = new byte[1024];
		int length = bys.length;
		DatagramPacket dp = new DatagramPacket(bys, length);
		ds.receive(dp);// 阻塞

		// 获取对方的IP
		InetAddress address = dp.getAddress();
		String ip = address.getHostAddress();

		byte[] bys2 = dp.getData();
		int len = dp.getLength();
		String s = new String(bys2, 0, len);
		System.out.println(ip + "传递的数据是:" + s);

		ds.close();
	}
}
```

代码改进

2018年7月30日 22:46

```
public class SendDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket();

		byte[] bys = "helloworld".getBytes();
		DatagramPacket dp = new DatagramPacket(bys, bys.length,
		InetAddress.getByName("DESKTOP-4O0BFJQ"), 12345);

		ds.send(dp);

		ds.close();
	}
}


public class RecieveDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket(12345);

		byte[] bys = new byte[1024];
		DatagramPacket dp = new DatagramPacket(bys, bys.length);

		ds.receive(dp);

		String ip = dp.getAddress().getHostAddress();

		String s = new String(dp.getData(), 0, dp.getLength());
		System.out.println("from" + ip + "data is" + s);

		ds.close();
	}
}
```

实现键盘录入

2018年7月30日 23:57

```
public class RecieveDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket(12345);
		while (true) {
			byte[] bys = new byte[1024];
			DatagramPacket dp = new DatagramPacket(bys, bys.length);

			ds.receive(dp);

			String ip = dp.getAddress().getHostAddress();

			String s = new String(dp.getData(), 0, dp.getLength());
			System.out.println("from" + ip + " data is： " + s);
		}
		// 接收端一直开着接收数据
		// ds.close();
	}
}


public class SendDemo {
	public static void main(String[] args) throws IOException {
		DatagramSocket ds = new DatagramSocket();

		// 键盘录入
		BufferedReader br = new BufferedReader(new
		InputStreamReader(System.in));
		String line = null;
		while ((line = br.readLine()) != null) {
			if ("886".equals(line)) {
				break;
			}
			byte[] bys = line.getBytes();
			DatagramPacket dp = new DatagramPacket(bys, bys.length,
			InetAddress.getByName("DESKTOP-4O0BFJQ"), 12345);
```

```
                ds.send(dp);
            }
            ds.close();
        }
    }
```

聊天室

2018年7月31日 10:11

```
/*
 * 通过多线程改进聊天室，可以实现在一个窗口发送和接收数据了
 */
public class ChatRoom {
	public static void main(String[] args) throws IOException {
		DatagramSocket dsSend = new DatagramSocket();
		DatagramSocket dsReceive = new DatagramSocket(12345);

		SendThread st = new SendThread(dsSend);
		ReceiveThread rt = new ReceiveThread(dsReceive);

		Thread t1 = new Thread(st);
		Thread t2 = new Thread(rt);

		t1.start();
		t2.start();
	}
}
```

```
public class SendThread implements Runnable {
	private DatagramSocket ds;

	public SendThread(DatagramSocket ds) {
		this.ds = ds;
	}

	@Override
	public void run() {
		try {
			// 键盘录入
			BufferedReader br = new BufferedReader(new
			InputStreamReader(System.in));
			String line = null;
			while ((line = br.readLine()) != null) {
				if ("886".equals(line)) {
					break;
				}
				byte[] bys = line.getBytes();
				DatagramPacket dp = new
				DatagramPacket(bys, bys.length,
				InetAddress.getByName("DESKTOP-4O0BFJ
				Q"),12345);

				ds.send(dp);
			}
			ds.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

}
```

```
public class ReceiveThread implements Runnable {
	private DatagramSocket ds;

	public ReceiveThread(DatagramSocket ds) {
		this.ds = ds;
	}
	@Override
	public void run() {
		try {
		while (true) {
			byte[] bys = new byte[1024];
			DatagramPacket dp = new DatagramPacket(bys, bys.length);

			ds.receive(dp);

			String ip = dp.getAddress().getHostAddress();

			String s = new String(dp.getData(), 0, dp.getLength());
			System.out.println("from" + ip + " data is： " + s);
		}}catch(IOException e) {
			e.printStackTrace();
		}
	}

}
```

TCP协议发送数据

2018年7月31日 10:15

```
/*
 * TCP协议发送数据：
 * 1.创建发送端的Socket对象：这一步如果成功，说明已建立连接
 * 2.获取输出流，写数据
 * 3.释放资源
 * TCP协议一定要先开服务器
 */
public class ClientDemo {
	public static void main(String[] args) throws IOException {
		// 创建发送端的Socket对象
		// Socket s = new Socket(InetAddress.getByName("DESKTOP-4O0BFJQ"),
		8888);
		Socket s = new Socket("DESKTOP-4O0BFJQ", 8888);

		// 获取输出流
		OutputStream os = s.getOutputStream();
		os.write("hello".getBytes());

		// 释放资源
		s.close();
	}
}
```

TCP协议接收数据

2018年7月31日 10:32

```
/*
 * TCP协议接收数据：
 * 1.创建接收端的Socket对象
 * 2.监听客户端，返回一个对象的Socket对象(三次握手)
 * 3.获取输入流，读取数据
 * 4.释放资源
 * TCP协议一定要先开服务器
 */
public class ServerDemo {
	public static void main(String[] args) throws IOException {
		// 创建接收端的Socket对象
		ServerSocket ss = new ServerSocket(12306);

		// 监听客户端，返回一个对象的Socket对象
		Socket s = ss.accept();

		// 获取输入流，读取数据
		InputStream is = s.getInputStream();
		byte[] bys = new byte[1024];
		int len = is.read(bys);
		String str = new String(bys, 0, len);

		String ip = s.getInetAddress().getHostAddress();

		System.out.println(ip + "---" + str);

		// 释放资源
		s.close();
	}
}
```

有反馈TCP

2018年7月31日 11:19

```
public class ServerDemo {
	public static void main(String[] args) throws IOException {
		ServerSocket ss = new ServerSocket(12306);
		Socket s = ss.accept();

		// 获取输入流
		InputStream is = s.getInputStream();
		byte[] bys = new byte[1024];
		int len = is.read(bys);// 阻塞
		String server = new String(bys, 0, len);
		System.out.println("server=" + server);

		// 获取输出流
		OutputStream os = s.getOutputStream();
		os.write("数据已收到".getBytes());

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

```
public class ClientDemo {
	public static void main(String[] args) throws IOException {
		Socket s = new Socket("DESKTOP-4O0BFJQ", 12306);

		OutputStream os = s.getOutputStream();
		os.write("hello".getBytes());

		InputStream is = s.getInputStream();
		byte[] bys = new byte[1024];
		int len = is.read(bys);// 阻塞
		String client = new String(bys, 0, len);
		System.out.println("client=" + client);

		s.close();
	}

}
```

键盘录入TCP

2018年7月31日 11:20

```
public class ClientDemo {
	public static void main(String[] args) throws IOException {
		Socket s = new Socket("DESKTOP-4O0BFJQ", 12306);

		BufferedReader br = new BufferedReader(new
		InputStreamReader(System.in));
		BufferedWriter bw = new BufferedWriter(new
		OutputStreamWriter(s.getOutputStream()));

		String line = null;
		while ((line = br.readLine()) != null) {
			// 键盘录入需要自定义借宿标记
			if ("886".equals(line)) {
				break;
			}
			bw.write(line);
			bw.newLine();
			bw.flush();
		}

		// bw.close();
		// br.close();
		s.close();
	}
}

public class ServerDemo {
	public static void main(String[] args) throws IOException {
		ServerSocket ss = new ServerSocket(12306);

		Socket s = ss.accept();

		// 包装通道内的流
		BufferedReader br = new BufferedReader(new
		InputStreamReader(s.getInputStream()));
		String line = null;
```

```
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }

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

TCP上传文件并给出反馈

2018年7月31日 11:45

读取文件可以以null作为结束信息的，但通道内不能这样结束信息
服务器不知道结束，更无法发送反馈，相互等待

如何解决：
1.再多些一条语句，告诉服务器，读取到此条结束
2.Socket提供了一个终止功能，会通知服务器不要等待了
s.shutdownOutput();

TCP上传图片

2018年7月31日 14:26

```
public class UploadServer {
	public static void main(String[] args) throws IOException {
		ServerSocket ss = new ServerSocket(12306);
		Socket s = ss.accept();

		BufferedInputStream bis = new BufferedInputStream(s.getInputStream());
		// 封装图片文件
		BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream("IP_0.png"));

		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = bis.read(bys)) != -1) {
			bos.write(bys, 0, len);
			bos.flush();
		}

		// 反馈
		OutputStream os = s.getOutputStream();
		os.write("图片上传成功".getBytes());

		bos.close();
		s.close();
	}
}
```

```
public class UploadClient {
	public static void main(String[] args) throws IOException {
		Socket s = new Socket("DESKTOP-4O0BFJQ", 12306);

		BufferedInputStream bis = new BufferedInputStream(new FileInputStream("IP.png"));
		BufferedOutputStream bos = new BufferedOutputStream(s.getOutputStream());

		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = bis.read(bys)) != -1) {
			bos.write(bys, 0, len);
			bos.flush();
		}

		s.shutdownOutput();

		// 读取反馈
		InputStream is = s.getInputStream();
		byte[] bys2 = new byte[1024];
		int len2 = 0;
		len2 = is.read(bys2);
		String client = new String(bys2, 0, len2);
		System.out.println(client);

		bis.close();
		s.close();
	}
}
```