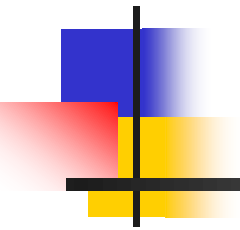
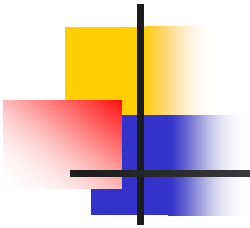
第十一章 Java网络编程



南京农业大学
谢忠红

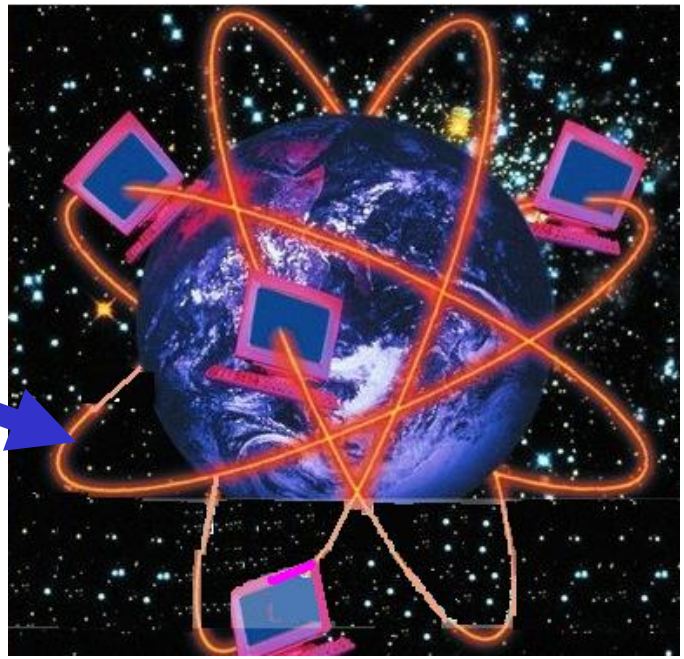


- 11.1 Java和Internet
  11.2 Socket-编程
- 11.3 URL类
- 11.4 Datagram(数据报)

11.1 Java和Internet

- **网络定义：**将分布在不同地理区域的计算机用通信线路互联起来的一个具有强大功能的网络系统

网络协议



- 网络的一些重要术语
- **DNS(域名服务器)**
- 例：**www.njau.edu.cn**
- **IP地址:TCP/IP**网络中的每台计算机都有唯一的地址**--IP**地址.
- 例**202.199.28.6**

Socket, URL , Datagram

- Socket是传统网络程序最常用的方式,可以想象为两个不同的程序通过网络的通信信道.（TCP/IP）
- URL（uniform Resouse Location）是三大功能中最高级的一种,通过URL Java程序可以使用http、file、ftp等协议，获取远端计算机上的资源信息，方便快捷地开发Internet应用程序
- Datagram更低级的网络传输方式,它把数据的目的地址在写在数据包中,然后将数据包放在网络上.(UDP)

方法索引 **InetAddress**

equals(Object)
比较该对象和指定对象。
getAddress()
返回这个 InetAddress 对象的原始 IP 地址。
getAllByName(String)
给定主机名，确定该主机的所有 IP 地址。
getByName(String)
给定主机名，确定该主机的所有 IP 地址。
getHostAddress()
返回IP 地址串 "%d.%d.%d.%d"
getHostName()
返回此地址的全限定主机名。
getLocalHost()
返回本地主机。
hashCode()
返回该IP 地址的散列码。
isMulticastAddress()
实用例行程序检查是否该 InetAddress 是IP多址发送地址。
toString()
把此IP 地址转换为 String 类型。

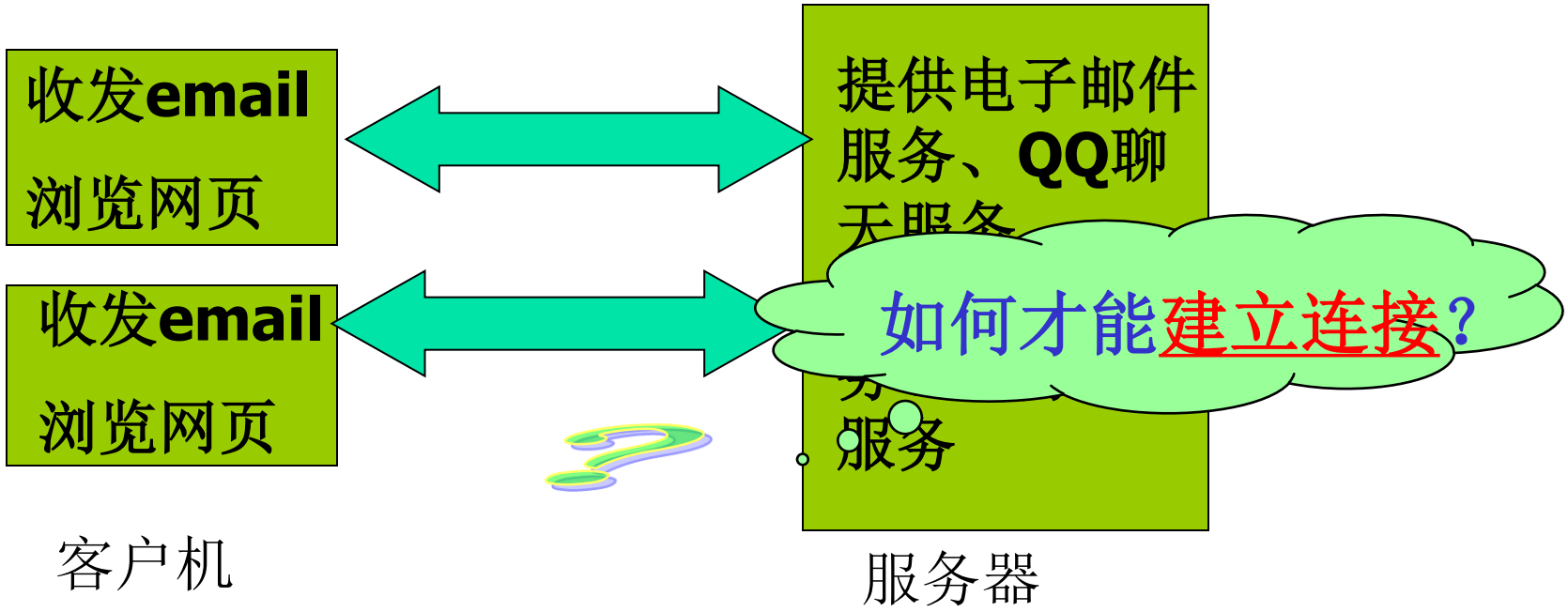
- 举例1： import java.net.*;

```
public class getLocalHostTest {
    public static void main(String args[]){
        InetAddress   myIP1=null,myIP2=null
        try {
            myIP1=InetAddress.getLocalHost();
            myIP2=InetAddress.getByName("xzh");
        }catch(UnknownHostException e){
            System.out.println(e.toString());}
        System.out.println("myIP1="+myIP1.toString());
        System.out.println("myIP2="+myIP2.toString());
    }
}
```

```
myIP1=LENOVO-A5267B58/192.168.1.100
myIP2=LENOVO-A5267B58/192.168.1.100
```

11.2 Socket-套接字编程

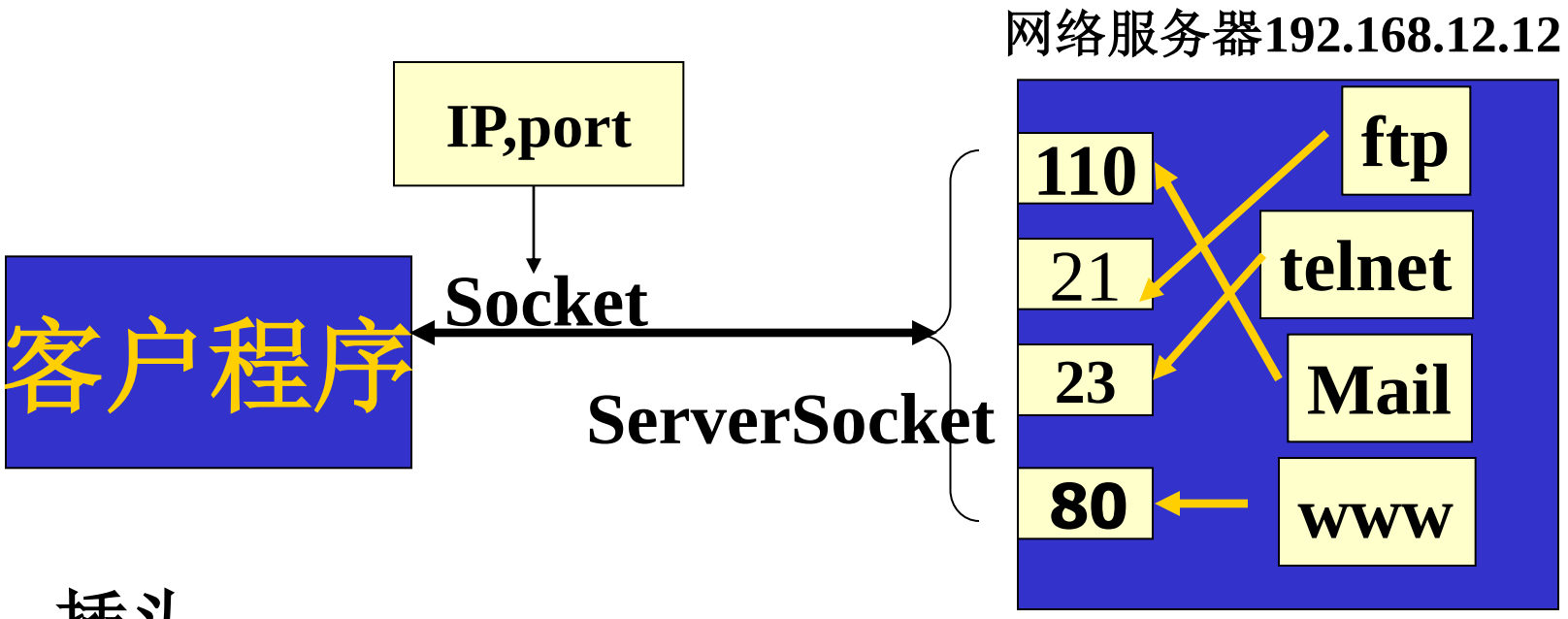
- 服务器：运行提供服务的程序的计算机
- 侦听建立连接的请求
- 客户机：使用由服务器程序所提供的服务的计算机
- 试着与某个服务器建立连接



Socket（套接字）：连接的一个终端

Socket=IP+Port

Socket的作用：分辨internet上运行各种服务程序



插头

布满插口的插座板

Java的Socket通信模式

- **ServerSocket类**
- **作用：负责 接受客户连接请求**
- **构造函数：**

ServerSocket(int port)

throws IOException

ServerSocket(int port ,int backlog)

throws Exception

backlog参数： 显示设置连接请求的队列长度，只要被加入到连接请求队列才可能与服务器连接成功

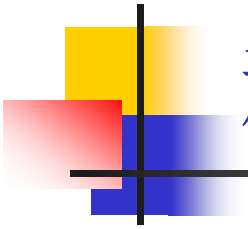
- ServerSocket实例方法：
- Socket accept()
- 作用：从连接队列中取一个客户机的连接请求，创建一个与客户连接的Socket对象
- InetAddress getInetAddress()
- 作用：返回服务器 socket 的本地地址

  getLocalPort()
- 作用：获得本连接的端口号
- close()
- 作用：使服务器释放占用的端口并断开与所有客户进程的连接

- **Socket类**
- 构造函数：
- **Socket(String host , int port)**
- **throws unknownHostException, IOException**
- 作用：试图建立与服务器的连接，如果连接成功返回一个**Socket**对象

- 客户连接服务器时可能抛出的异常
- **UnknownHostException:** 无法识别主机名字或IP地址
- **ConnectException**：服务器进程拒绝连接，或没有服务器进程监听制定端口
- **SocketTimeoutException**：等待连接超时
- **BindException**：无法把**Socket**对象与指定的本地IP地址或port绑定

- 实例方法
- public InetAddress getInetAddress()
- 作用：返回远程服务器的地址。
- public InetAddress getLocalAddress()
  作用：获得客户本地的**IP**地址
- **public InputStream getInputStream()**
- 作用：获得输入流
- **Public OutputStream getOutputStream()**
- 作用：获得输出流
- **Public void close()**
- 作用：关闭**Socket**连接



程序举例

- 功能描述：服务器程序向客户机程序发送字符串“hi”

服务器程序举例

```
import java.net.*;     import java.io.*;
public class Myserver{
   public static void main(String args[]){
        PrintStream out;
       Socket s=null;
      try{ ServerSocket  serverS=new    ServerSocket(5432);
      while(true){
            Socket s=  serverS.accept();
            if (s!=null) break;
       }
         out=new  PrintStream(S.getOutputStream());
         out.println("hi");
         out.close();       s.close();
     }catch(IOException e){  e.printStackTrace();}
}
```

客户机程序举例

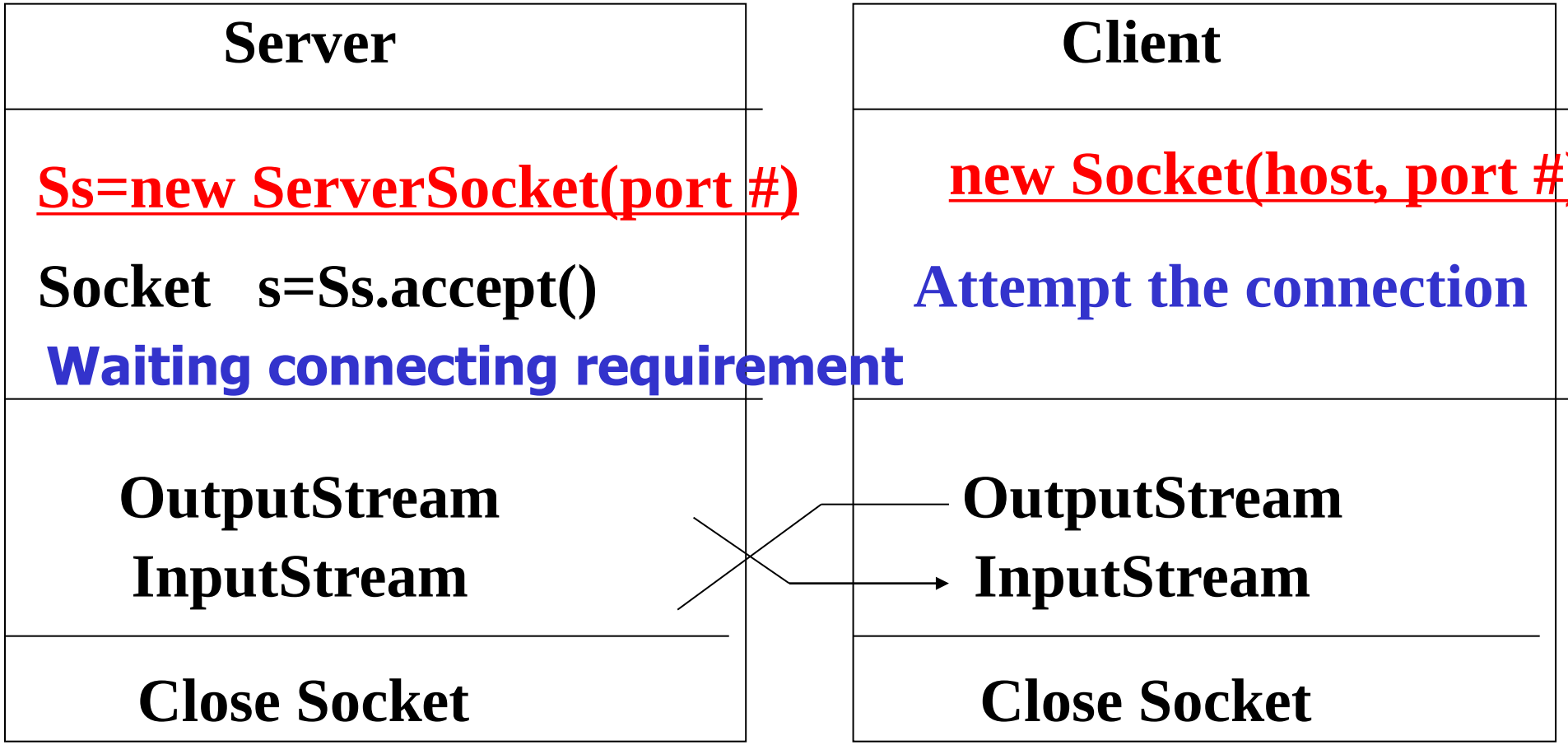
```
public class Myclient{
   public static  void main(String args[]){
      try{
         Socket s=new Socket("127.0.0.1",5432);
       BufferedReader in=new BufferedReader(new
              InputStreamReader(s.getInputStream()));
       String x=in.readLine();
       System.out.println("read from server:  "+x);
       in.close();
       s.close();
    }catch(IOException e){   }
}}
```

read from server: hi

3指定一协议编写客户/服务器程序

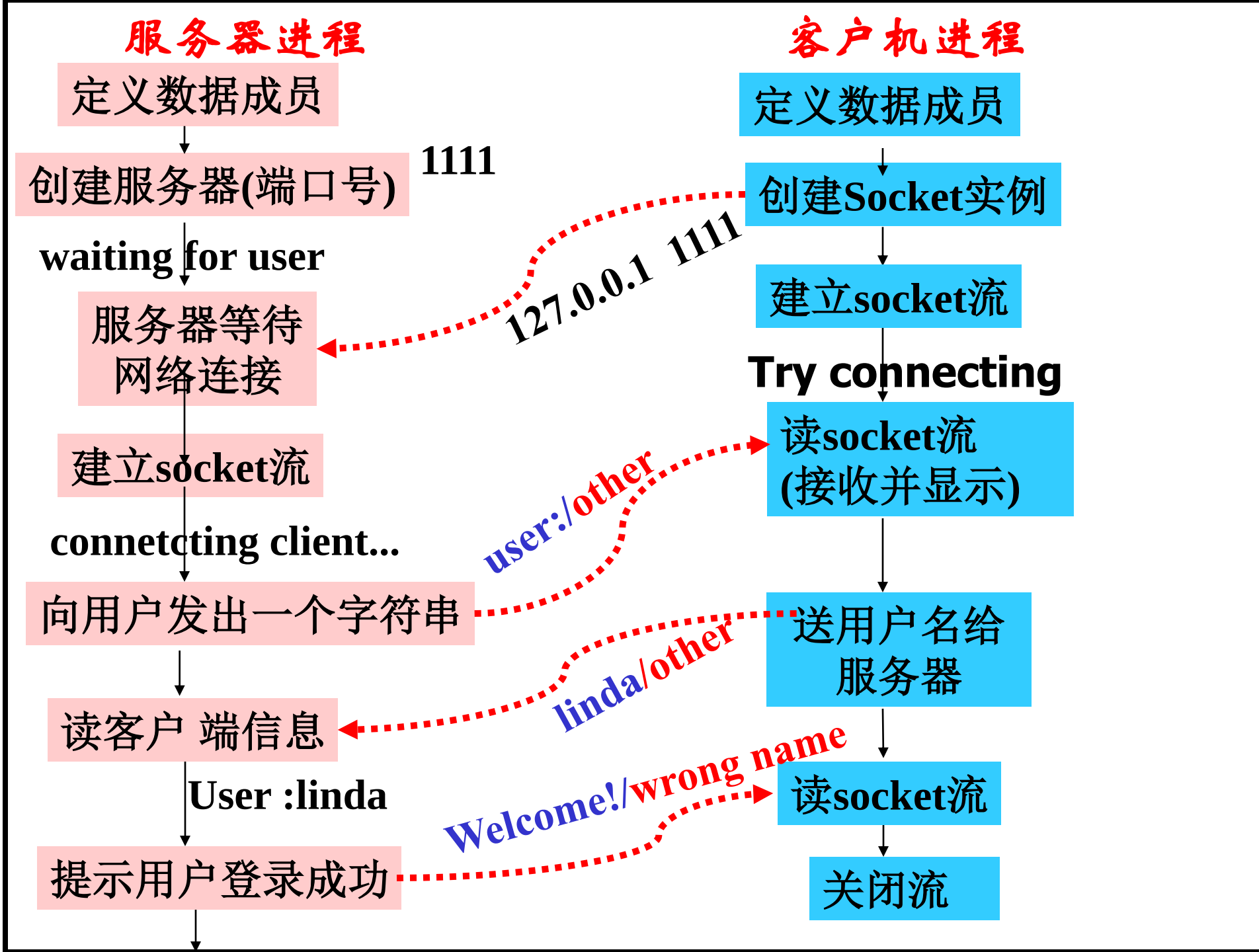
程序功能： 演示一个简单的服务器与客户的交互

即服务器客户机相互通一次信息。



■自定义协议规则

- 服务器规则：
- （1）向客户机发一个“user:”。
- （2）读取客户机提供的用户名。
- （3）如果用户名==“linda” 发送 “welcome!”。
- （4）如果用户名!=“linda” 发送 “wrong name”!。
- 相应的客户机规则：
- （1）读取服务器反馈信息。
- （2）若反馈信息是“user:”,则向服务器发送用户名并
- 执行（3），若反馈信息不“user:”退出。
- (3)读取服务器反馈信息。
- （4）若反馈信息是“welcome”!则显示登陆成功信息。
- （5）若反馈信息是”wrong name!”则显示输入错误用户名信息。


服务器进程
客户机进程
定义数据成员
定义数据成员
创建服务器(端口号)
1111
创建Socket实例
waiting for user
127.0.0.1 1111
服务器等待
网络连接
建立socket流
Try connecting
建立socket流
读socket流
(接收并显示)
connetcting client...
user:/other
向用户发出一个字符串
送用户名给
服务器
linda/other
读客户 端信息
User :linda
Welcome!/wrong name
读socket流
提示用户登录成功
关闭流

服务程序：

```
public class ServerRegister{
  public static void main(String args[]){
    ServerSocket  Ss=new ServerSocket(1111);
    Socket    s=Ss.accept();
    PrintStream out=new
            PrintStream(s.getOutputStream());
    BufferedReader in=new BufferedReader(
     new InputStreamReader(s.getInputStream()));
```

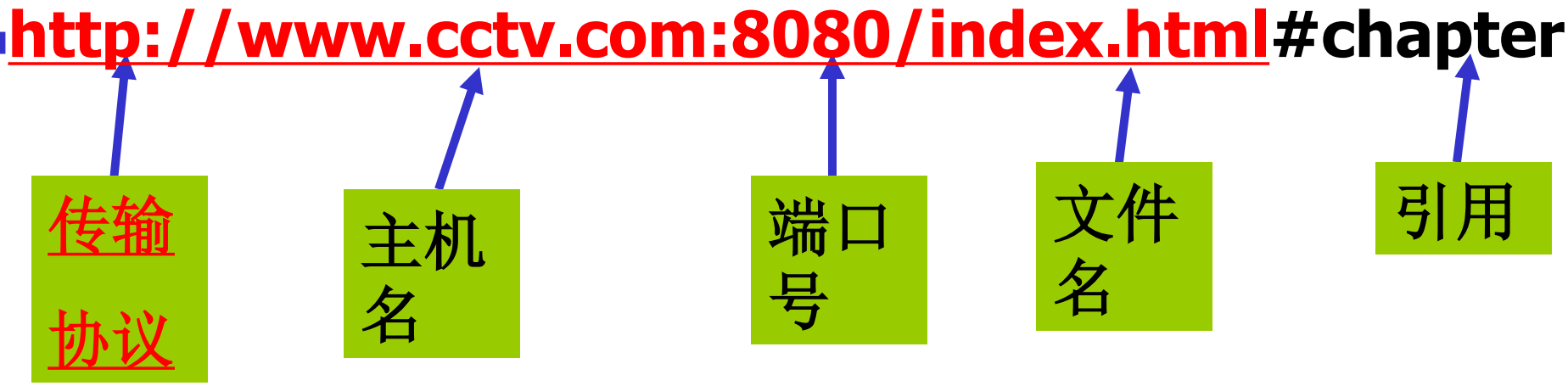
```
try{
    while(true){
        System.out.println("send to Client:"+"user:");
        out.println("user:");
        String x=in.readLine();
        System.out.println("get from Client: "+x);
        if ( x.equals("linda") ){
            System.out.println("send to Client "+"Login
                                        welcome!");
            out.println("welcome!");  }
        else out.println("wrong name!");   }
    out.close();    in.close();    s.close();
}catch(IOException e){
    System.out.println(e.toString());}
```

■客户机程序：CleintRegister

```
try{String x=in.readLine();
    if (x.equals(“user:")){
        System.out.println"get from Server:  "+x);
        System.out.println("send to Server:   "+"linda");
        out.println("linda");x=in.readLine();
        System.out.println(“get from Server"+x);
        if (x==“welcome!”)
            System.out.println(“register successfully”);
        if (x==“wrong name”)
            System.out.println(“register unsuccessfully”);
    }else { System.exit(0); }
    out.close();    in.close();    s.close();
```

11.3 URL类
（Uniform Resource Locator）

- **URL的值**：是表示网络上某个资源的地址。



构造URL类 (全名java.net.URL)

URL的构造方法:

(1) URL(String spec)

例： **URL url= new URL**
("http://www.hit.edu.cn:8080/cv/index.html")

作用：创建一个能找到该文件的**URL**对象

(2)URL(URL u, String spec)

例：**URL** base=new URL("http://rainy.hit.edu.cn/test");
URL url1=new (base, "mywork.html");

URL类的基本方法

String getProtocol()

作用：返回该**URL**对象使用的传输协议

String getHost()

int getPort()

String getFile()

String getRef()

构造URL的实例

```
import java.net.URL ; import java.io.*;
import java.net.MalformedURLException;
public class testURL{
   public static void main(String args[]){
     URL url1;
     try {
     url1=new URL
        ("http://www.njau.edu.cn:8080/cv/index.html");
     displayURL(url1);
    //displayContent(url1)  ;
     }catch (MalformedURLException e){
     System.out.println(e.toString()); }
   }
```

```
URL is
toString()=http://www.njau.edu.cn:8080/cv/index.html
Protocal=http
Host=www.njau.edu.cn
Port=8080
File=/cv/index.html
Ref=null
java.net.UnknownHostException: www.njau.edu.cn
Press any key to continue...
```

- 使用字节流访问URL资源
- URL类的一个方法：
- InputStream openStream()
- 作用：返回一个输入流对象，数据源是URL对象所指的地址。

```
public static void displayContent(URL url) {
    int c;
    InputStreamReader in=new
          InputStreamReader(url.openStream());
    BufferedReader  bin=new BufferedReader(in);
    FileOutputStream   fout=new
                  FileOutputStream("d:/njau.txt");
    while((c=(char)bin.read())!=-1){
          fout.write((char)c);
          System.out.print((char)c);     }
          bin.close(); fout.close();
}
```

- 针对HTTP协议的URLConnection类
- URLConnection类的作用：
- 作用：访问远程资源，和它建立连接。
- （探测到这个远程对象的属性下载、阅读该远程资源）
- 使用方法：
- URL myurl= new URL("http://www.edu.cn/index.htm")
- URLConnection myuc=
- myurl.openConnection();

- **URLConnection类常见实例方法**
- **int getContentLength()**
- 作用：访问对象的长度
- **String getContentType()**
- 作用：访问对象的类型
- **long getDate()**
- 作用：访问对象的当前日期
- **InputStream getInputStream()**
- 作用：返回输入流对象，数据源是访问的该对象
- **OuputStream getOuputStream()**
- 作用：返回输出流对象，输出目的地是访问的该对象

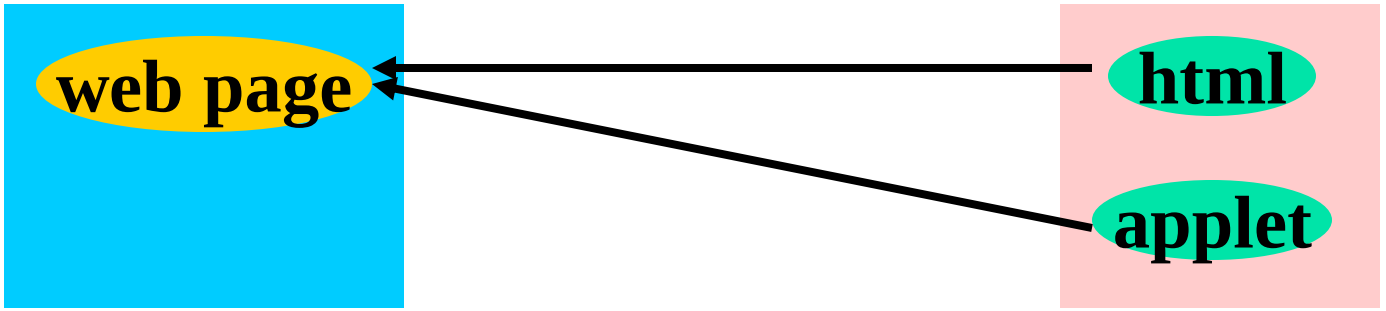
- 程序示例：
- 通过java提供的URL类和URLConnection，从指定的网址上下载网页

```
public class UseURLConnection{
    public static void main(String args[]) throws Exception{
        URL urlname = new URL("http://www.edu.cn");
        URLConnection uc =urlname.openConnection();
        UseURLConnection  x=new UseURLConnection();
       x.downloadContent(urlname,   uc);
     }
```

```
public static void download(URL urlname,
                            URLConnection uc ){
    char c;
    try{
      InputStream in=uc. getInputStream( );
      FileOutputStream   fout=new
                FileOutputStream("d:/njau.txt");
      while((c=(char)in.read())!=-1){
           fout.write((char)c);
          System.out.print((char)c); }
          in.close(); fout.close();
    } catch(IOException e){
       System.out.println(e.toString());}
  }
```

获取小应用程序HTML页面的URL和小应用程序本身的URL的方法

- URL html=getDocumentBase();
- //返回当前Applet所在的HTML文件的URL
- System.out.print(html);
- URL codebase=getCodeBase();
- //返回当前AppletL文件的URL
- System.out.print(codebase);



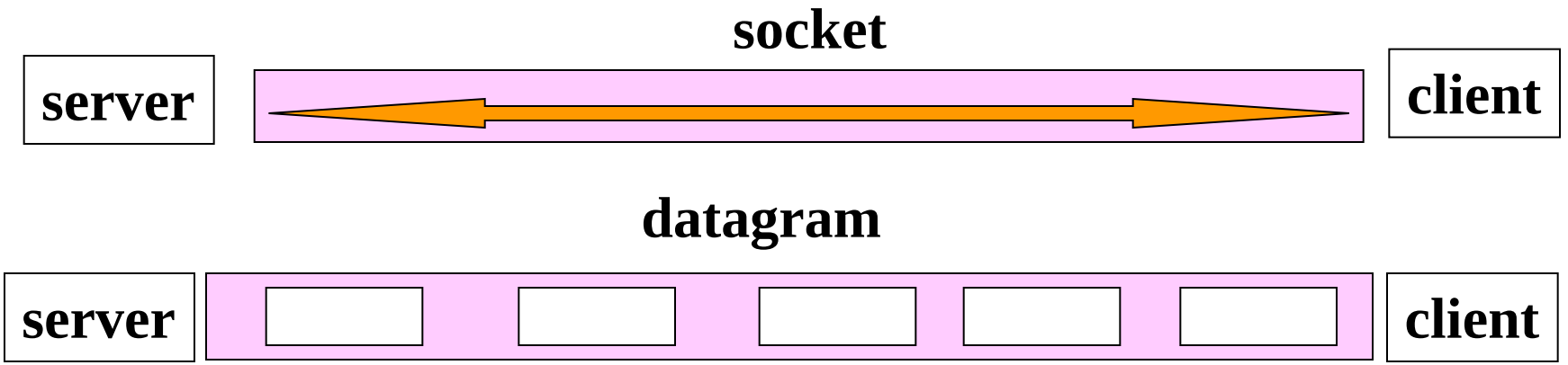
```
import java.applet.*;import java.awt.*;
public class image extends Applet {
    Image img;
   public void init() {
       img=getImage(getCodeBase(),“IMAGES/FRUIT.JPG");}
   public void paint(Graphics g) {
          int width=img.getWidth(this);
          int height=img.getHeight(this);
          g.drawRect(52,52,width+30,height+30);
          g.drawImage(img,57,57,width+20,height+20,this);}
}
```

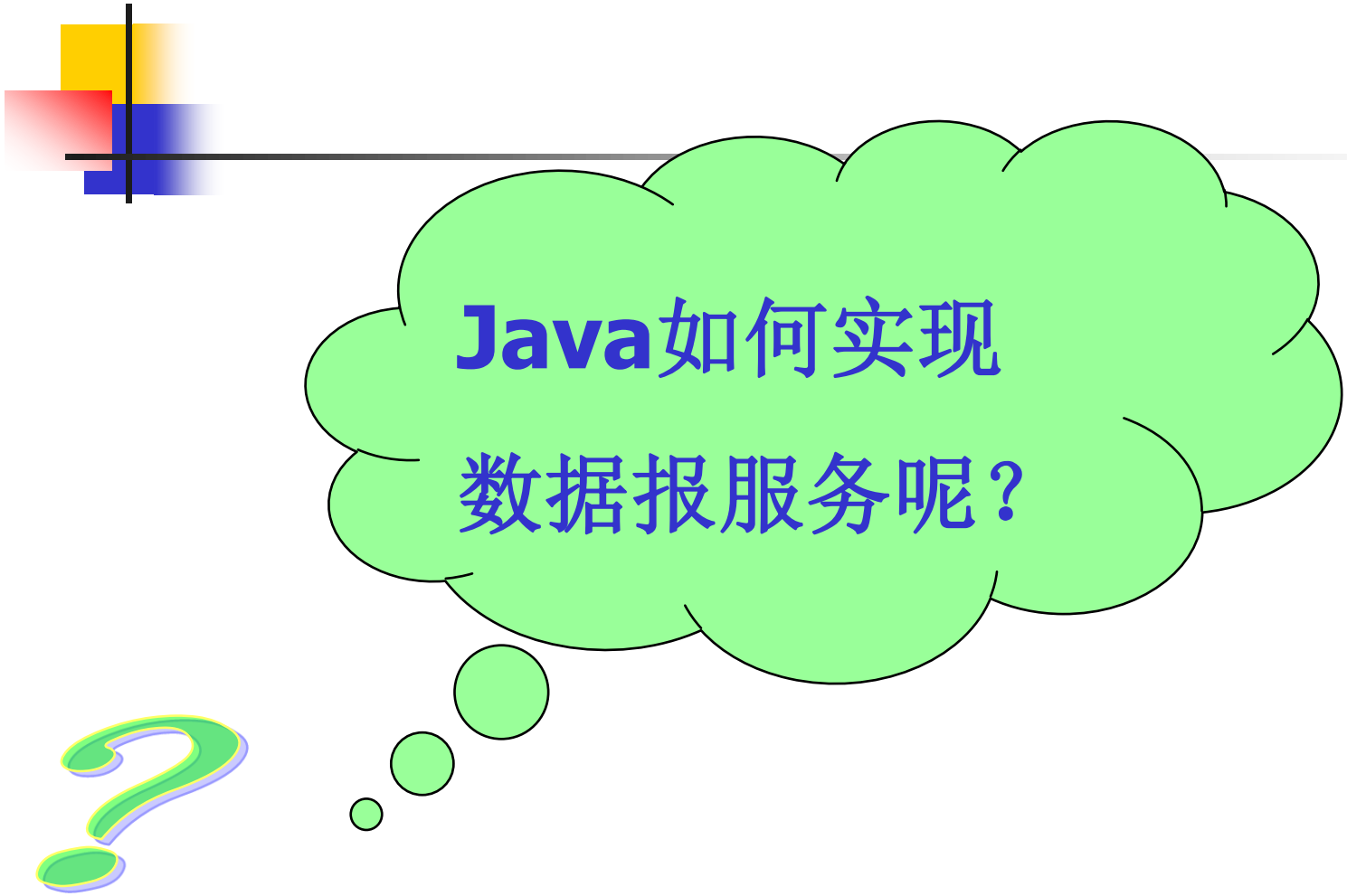
11.4Datagram(数据报)

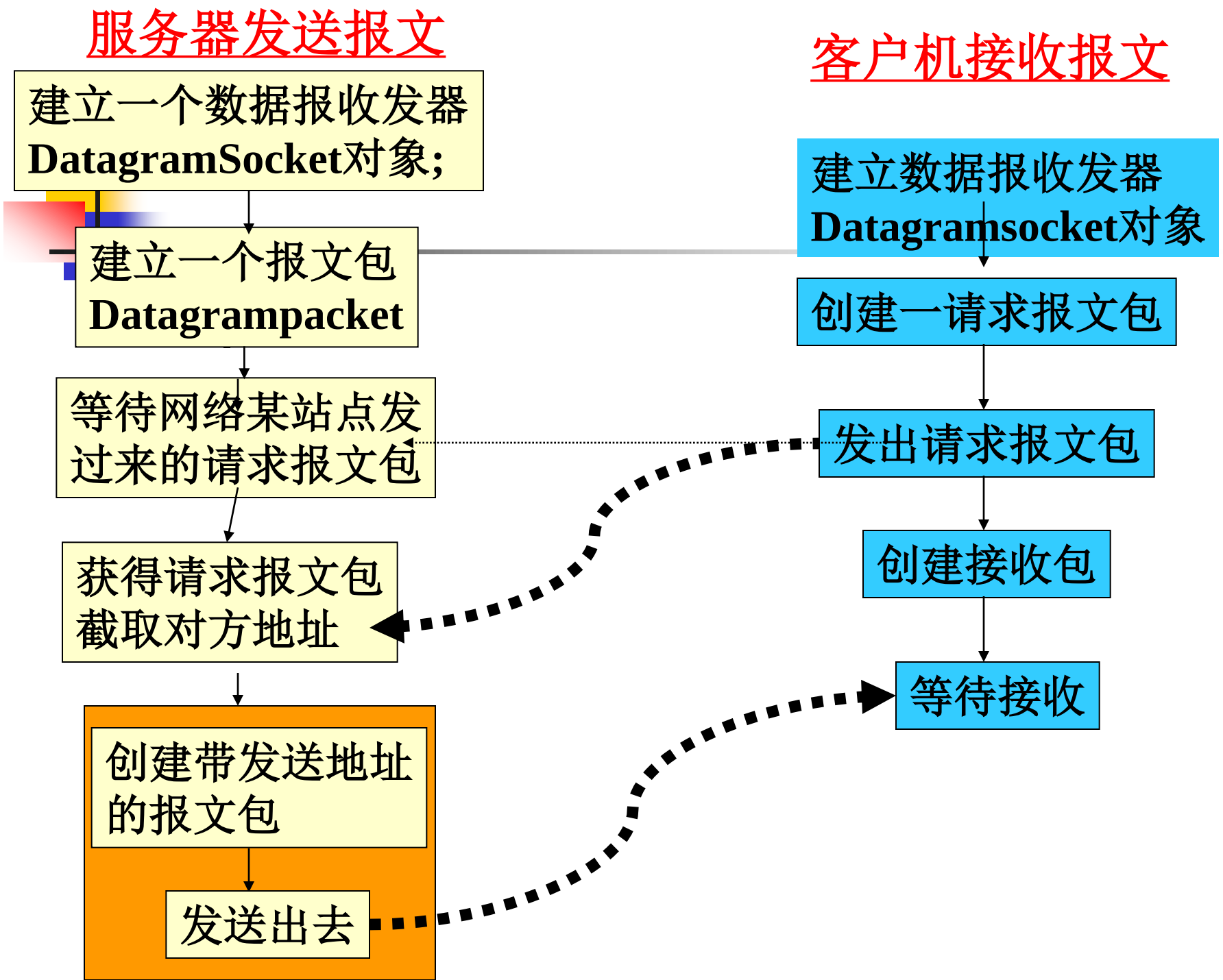
TCP/IP 协议：基于套接字具有高度的可靠性

UDP协议(User Datagram Protocol)

到达时间,以及内容不能得到保证。




Java如何实现
数据报服务呢？


服务器发送报文
客户机接收报文
建立一个数据报收发器DatagramSocket对象;
建立一个报文包Datagrampacket
等待网络某站点发过来的请求报文包
获得请求报文包截取对方地址
创建带发送地址的报文包
发送出去
建立数据报收发器Datagramsocket对象
创建一请求报文包
发出请求报文包
创建接收包
等待接收

报文收发器类 DatagramSocket

作用：收发数据包

与Socket类的不同点：不存在一对一的连接，

也不存在等待连接问题。

构造函数：**DatagramSocket(int port)**

DatagramSocket( )

方法：**receive(报文包)**

send(报文包)

- 报文包类DatagramPacket类
- 作用：存放接受或要发送出去的报文。
- 接受报文时报文包构造函数：
- **DatagramPacket (byte[] buf,**
- **int length)**
- 发送报文时报文包构造函数：
- **DatagramPacket (byte[] buf,**
- **int length, InetAdress add, int port);**

- 方法：
- **InetAddress getAddress():** 获得报文包来源的**IP**地址/或报文包正发送的目的地**IP**
- **Int getPort():** 同上
- **Byte[] getData():**返回收到报文的内容或发出报文的内容

服务器发送报文的标准步骤如下:

1. 定义数据成员

```
DatagramSocket      DataSR;
DatagramPacket      Datapacket;
InetAddress   address;//用来存放接收方的地址)
int port;           //用来存放接收方的端口号)
byte[] Buf=new byte[256]; //设定缓冲区，存放从接收/发出信息
```

2. 创建数据收发器**DatagramSocket**对象

```
try {DataSR=new DatagramSocket(5432);}
catch(Exception e) {}
```

- 4.创建一个报文包DatagramPacket对象,等待有请求报文发送过来
- pleaseP=new DatagramPacket(buf, 256);
- DataRS.receive(pleaseP);
- //在客户的请求包来之前一直等待
- 5获得请求报文包,截取对方IP地址和port号
- InetAddress addr=pleaseP.getAddress();
- int port=pleaseP.getPort();
- 7. 将需发送的数据送入缓冲区buf中
- 8. 建立发送报文包,用来用数据收发器发出
- sendP=new DatagramPacket
- (buf,buf.length, address,port);
- DataRS.send(packet);
- 10.关闭 DataRS.close();

客户端接收包的步骤:

1. 定义数据成员

```
int    port;    InetAddress    address;
DatagramSocket   DataRS;
DatagramPacket   packet;
byte[] Buf=new   byte[256];
```

2. 建立报文收发器DatagramSocket对象

```
DataRS=new DatagramSocket();
```

3. 向服务器发出请求报文

```
packet=new DatagramPacket( Buf,256,
                                    address,port);
 DataRS.send(packet);
```

// 这个包本身带有客户端的信息

4创建一接收报文包

- **packet=new DatagramPacket(Buf,256);**

5. 客户机等待服务器的应答

- **socket.receive(packet);**

6. 处理接收到的数据

String received =new

String(packet.getData());

- **System.out.println(received);**

说明

- 数据报接收器的发送方和接收方的区别:
- **(1)**一个客户机必须事先知道服务器的地址和端口,以便进行出事连接
- **(2)**一个服务器从接收到的数据报文中读取客户端的地址和端口后才知道数据报发往何处。