Java SE程序设计 北京圣思园科技有限公司

主讲人　张龙

Java Network

- 课程目标
  - 理解Java 网络系统
  - 熟练使用java.net包中的相关类与接口进行网络编程
  - 掌握如何使用Java在一台或多台计算机之间进行基于TCP/IP协议的网络通讯
  - 为今后J2EE的学习做好准备

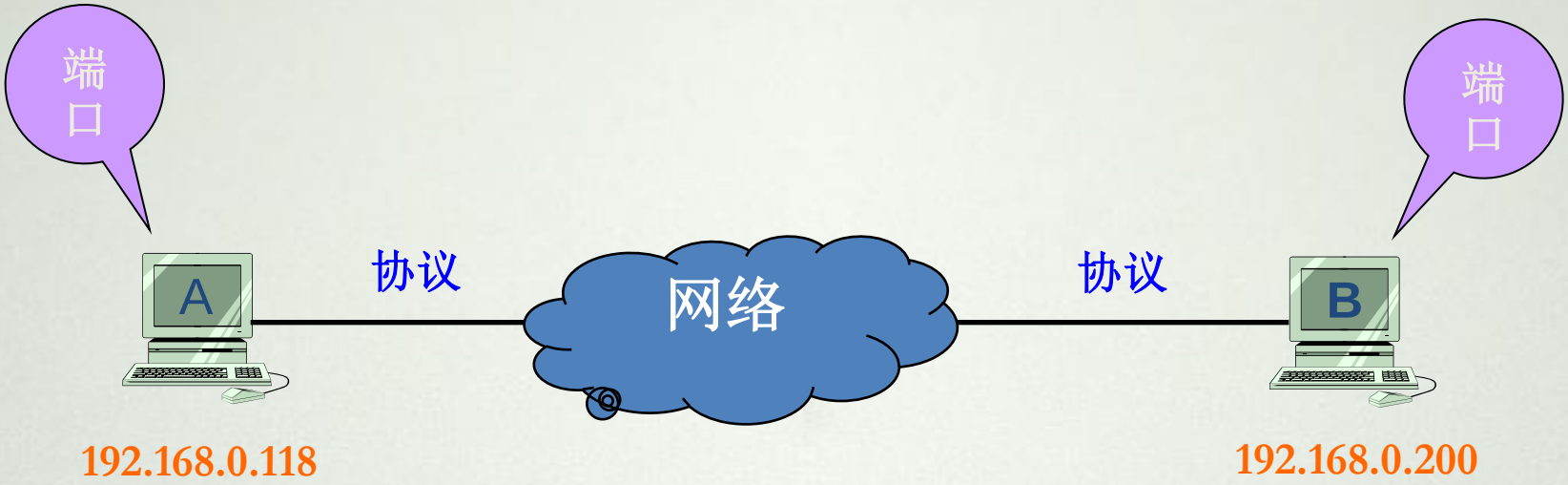
网络基础知识

- 网络编程的目的就是指直接或间接地通过网络协议与其他计算机进行通讯。网络编程中有两个主要的问题，一个是如何准确的定位网络上一台或多台主机，另一个就是找到主机后如何可靠高效的进行数据传输。在TCP/IP协议中IP层主要负责网络主机的定位，数据传输的路由，由IP地址可以唯一地确定Internet上的一台主机。而TCP层则提供面向应用的可靠的或非可靠的数据传输机制，这是网络编程的主要对象，一般不需要关心IP层是如何处理数据的

网络基础知识

- 目前较为流行的网络编程模型是客户机/服务器（C/S）结构。即通信双方一方作为服务器等待客户提出请求并予以响应。客户则在需要服务时向服务器提出申请。服务器始终运行，监听网络端口，一旦有客户请求，就会启动一个服务线程来响应该客户，同时自己继续监听服务端口，使后来的客户也能及时得到服务

两台计算机通过网络进行通信



IP地址

- IP网络中每台主机都必须有一个惟一的IP地址；
- IP地址是一个逻辑地址；
- 因特网上的IP地址具有全球唯一性；
- 32位，4个字节，常用点分十进制的格式表示，例如：192.168.0.200

协议

- 为进行网络中的数据交换（通信）而建立的规则、标准或约定。(=语义+语法+规则)
- 不同层具有各自不同的协议。

网络的状况

- 多种通信媒介——有线、无线.....
- 不同种类的设备——通用、专用.....
- 不同的操作系统——Unix、Windows .....
- 不同的应用环境——固定、移动.....
- 不同业务种类——分时、交互、实时.....
- 宝贵的投资和积累——有形、无形.....
- 用户业务的延续性——不允许出现大的跌宕起伏。
- 它们互相交织，形成了非常复杂的系统应用环境。

网络异质性问题的解决

- 网络体系结构就是使这些用不同媒介连接起来的不同设备和网络系统在不同的应用环境下实现互操作性，并满足各种业务需求的一种粘合剂，它营造了一种“生存空间” —— 任何厂商的任何产品、以及任何技术只要遵守这个空间的行为规则，就能够在其中生存并发展。
- 网络体系结构解决异质性问题采用的是分层方法 —— 把复杂的网络互联问题划分为若干个较小的、单一的问题，在不同层上予以解决。
- **就像我们在编程时把问题分解为很多小的模块来解决一样**。

ISO/OSI七层参考模型

- **OSI(Open System Interconnection)参考模型将网络的不同功能划分为7层**

处理网络应用

数据表示

主机间通信

端到端的连接

寻址和最短路径

介质访问(接入)

二进制传输

ISO/OSI七层参考模型

- 通信实体的对等层之间不允许直接通信。
- 各层之间是严格单向依赖。
- 上层使用下层提供的服务 — Service user；
- 下层向上层提供服务 — Service provider

对等层通信的实质

- 对等层实体之间虚拟通信。
- 下层向上层提供服务，实际通信在最底层完成

OSI各层所使用的协议

- 应用层：远程登录协议Telnet、文件传输协议FTP、 超文本传输协议HTTP、域名服务DNS、简单邮件传输协议SMTP、邮局协议POP3等。
- 传输层：传输控制协议TCP、用户数据报协议UDP。

  TCP：面向连接的可靠的传输协议。

  UDP：是无连接的，不可靠的传输协议。
- 网络层：网际协议IP、Internet互联网控制报文协议ICMP、Internet组管理协议IGMP

两类传输协议：TCP,UDP

- TCP是Transfer Control Protocol的简称，是一种面向连接的保证可靠传输的协议。通过TCP协议传输，得到的是一个顺序的无差错的数据流。发送方和接收方的成对的两个socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个socket（通常都是server socket）等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作

两类传输协议：TCP,UDP

- TCP是一个基于连接的协议，它能够提供两台计算机之间的可靠的数据流．HTTP、FTP、Telnet等应用都需要这种可靠的通信通道

两类传输协议：TCP,UDP

- UDP是User Datagram Protocol的简称，是一种无连接的协议，每个数据报都是一个独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此能否到达目的地，到达目的地的时间以及内容的正确性都是不能被保证的

两类传输协议：TCP,UDP

- UDP是从一台计算机向另一台计算机发送称为数据报的独立数据包的协议，该协议并不保证数据报是否能正确地到达目的地。它是一个非面向连接的协议

两类传输协议：TCP,UDP

- 下面我们对这两种协议做简单比较
  - 使用UDP时，每个数据报中都给出了完整的地址信息，因此无需要建立发送方和接收方的连接。
  - 对于TCP协议，由于它是一个面向连接的协议，在socket之间进行数据传输之前必然要建立连接，所以在TCP中多了一个连接建立的时间

两类传输协议：TCP,UDP

- 使用UDP传输数据时是有大小限制的，每个被传输的数据报必须限定在64KB之内。
- TCP没有这方面的限制，一旦连接建立起来，双方的socket就可以按统一的格式传输大量的数据。
- UDP是一个不可靠的协议，发送方所发送的数据报并不一定以相同的次序到达接收方。
- TCP是一个可靠的协议，它确保接收方完全正确地获取发送方所发送的全部数据

两类传输协议：TCP,UDP

- TCP在网络通信上有极强的生命力，例如远程连接（Telnet）和文件传输（FTP）都需要不定长度的数据被可靠地传输。
- 相比之下UDP操作简单，而且仅需要较少的监护，因此通常用于局域网高可靠性的分散系统中client/server应用程序

两类传输协议：TCP,UDP

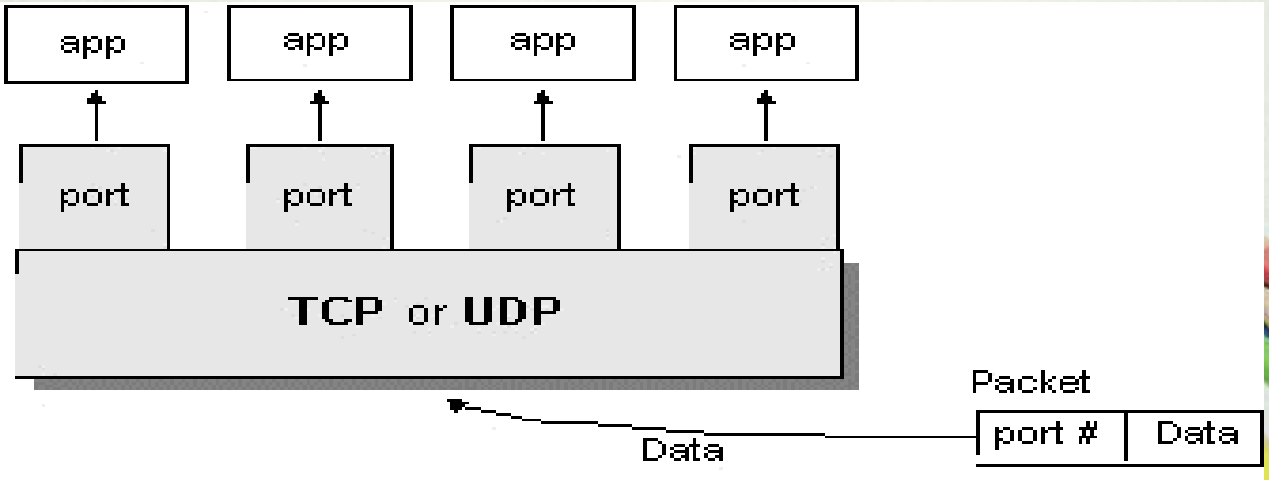
- 既然有了保证可靠传输的TCP协议，为什么还要非可靠传输的UDP协议呢？

两类传输协议：TCP,UDP

- 主要的原因有两个。
  - 一是可靠的传输是要付出代价的，对数据内容正确性的检验必然占用计算机的处理时间和网络的带宽，因此TCP传输的效率不如UDP高
  - 二是在许多应用中并不需要保证严格的传输可靠性，比如视频会议系统，并不要求音频视频数据绝对的正确，只要保证连贯性就可以了，这种情况下显然使用UDP会更合理一些

端口

- 在互联网上传输的数据都包含有用来识别目的地的IP地址和端口号。IP地址用来标识网络上的计算机，而端口号用来指明该计算机上的应用程序

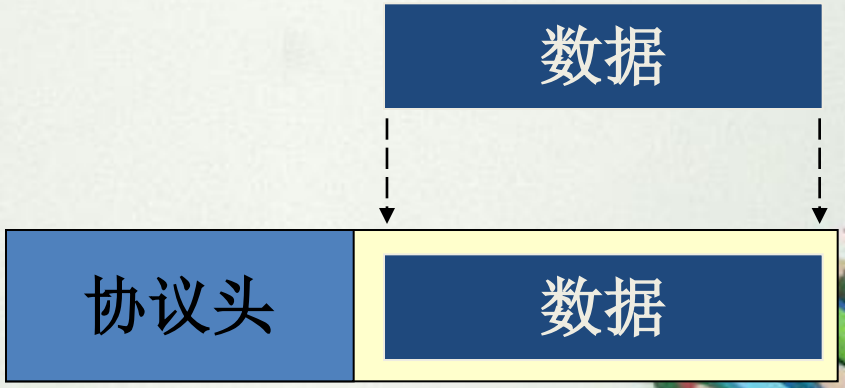


端口

- 端口是一种抽象的软件结构（包括一些数据结构和I/O缓冲区）。应用程序通过系统调用与某端口建立连接（binding）后，传输层传给该端口的数据都被相应的进程所接收，相应进程发给传输层的数据都通过该端口输出。
- 端口用一个整数型标识符来表示，即端口号。端口号跟协议相关，TCP/IP传输层的两个协议TCP和UDP是完全独立的两个软件模块，因此各自的端口号也相互独立，端口通常称为协议端口(protocol port)，简称端口。
- 端口使用一个16位的数字来表示，它的范围是0~65535，1024以下的端口号保留给预定义的服务。例如：http使用80端口。

数据封装

- 一台计算机要发送数据到另一台计算机，数据首先必须打包，打包的过程称为封装。
- 封装就是在数据前面加上特定的协议头部。



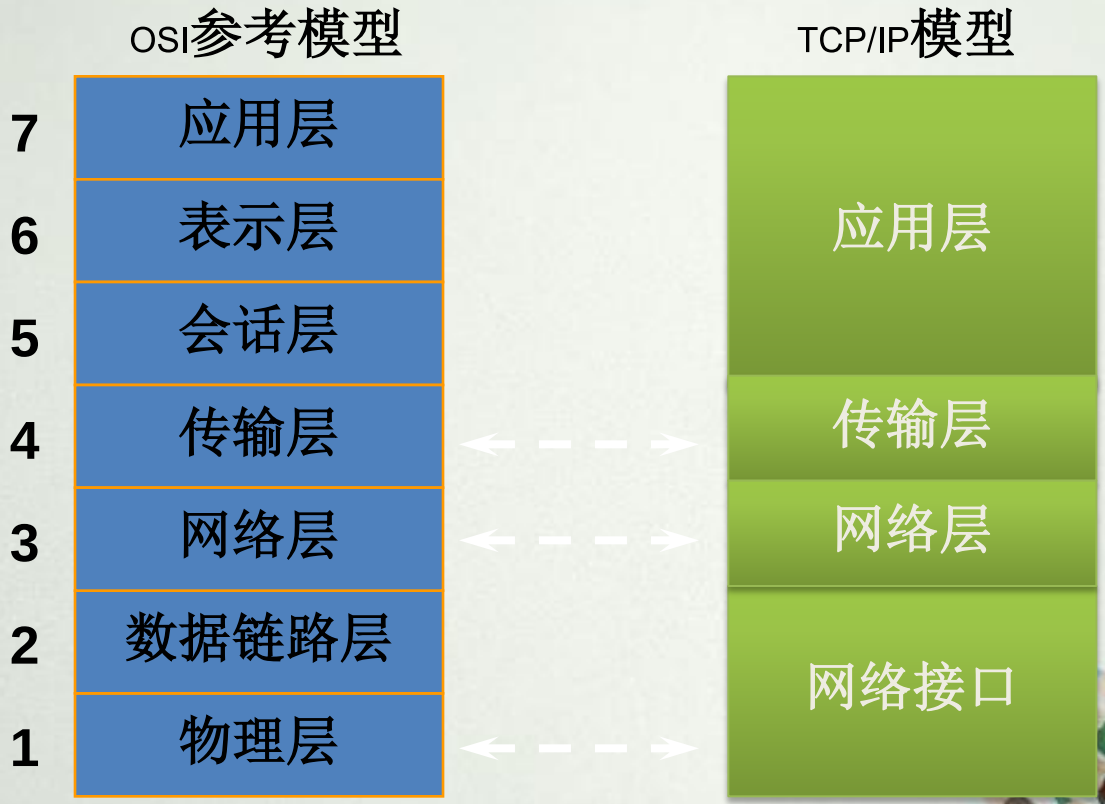
数据封装

- **OSI**参考模型中，对等层协议之间交换的信息单元统称为协议数据单元**(PDU，Protocol Data Unit)**。
- **OSI**参考模型中每一层都要依靠下一层提供的服务。
- 为了提供服务，下层把上层的**PDU**作为本层的数据封装，然后加入本层的头部（和尾部）。头部中含有完成数据传输所需的控制信息。
- 这样，数据自上而下递交的过程实际上就是不断封装的过程。到达目的地后自下而上递交的过程就是不断拆封的过程。由此可知，在物理线路上传输的数据，其外面实际上被包封了多层“信封”。
- 但是，某一层只能识别由对等层封装的“信封”，而对于被封装在“信封”内部的数据仅仅是拆封后将其提交给上层，本层不作任何处理。

TCP/IP模型

- **TCP/IP**起源于美国国防部高级研究规划署**(DARPA)**的一项研究计划——实现若干台主机的相互通信。
- 现在**TCP/IP**已成为**Internet**上通信的工业标准。
- **TCP/IP**模型包括**4**个层次：
  - 应用层
  - 传输层
  - 网络层
  - 网络接口

TCP/IP与OSI参考模型的对应关系



JDK中的网络类

- 通过java.net包中的类，java程序能够使用TCP或UDP协议在互联网上进行通讯
- Java 通过扩展已有的流式输入/输出接口和增加在网络上建立输入/输出对象特性这两个方法支持TCP/IP。
- Java支持TCP和UDP协议族。TCP用于网络的可靠的流式输入/输出。UDP支持更简单的、快速的、点对点的数据报模式

创建和使用URL访问网上资源

- URL(Uniform Resource Locator)是统一资源定位符的简称，它表示Internet上某一资源的地址。通过URL我们可以访问Internet上的各种网络资源，比如最常见的WWW，FTP站点。浏览器通过解析给定的URL可以在网络上查找相应的文件或其他资源。

创建和使用URL访问网上资源

- URL是最为直观的一种网络定位方法。使用URL符合人们的语言习惯，容易记忆，所以应用十分广泛。而且在目前使用最为广泛的TCP/IP中对于URL中主机名的解析也是协议的一个标准，即所谓的域名解析服务。使用URL进行网络编程，不需要对协议本身有太多的了解，功能也比较弱，相对而言是比较简单的。

创建和使用URL访问网上资源

- 一个URL 包括两个主要部分：
  - 协议标识符：HTTP，FTP，File等
  - 资源名字：主机名，文件名，端口号，引用
- 例如：
- http://java.sun.com:80/docs/books/tutorial/index.html#DOWN

创建和使用URL访问网上资源

- 创建URL
  - 在Java程序中，可以创建表示URL地址的URL对象。URL对象表示一个绝对URL地址，但URL对象可用绝对URL、相对URL和部分URL构建

创建和使用URL访问网上资源

- 创建URL
  - 例如：http://www.gamelan.com/pages/index..html
  - new URL("http://www.gamelan.com/pages/index.html");
  - URL gamelan = new URL("http://www.gamelan.com/pages/");
  - URL gamelanGames = new URL(gamelan, "game.html");
  - new URL("http", "www.gamelan.com", "/pages/index.html");
  - new URL("http", "www.gamelan.com", 80, "pages/index.network.html");

创建和使用URL访问网上资源

- 创建URL
  - 如果创建失败：
  - try
  - {
  - URL myURL = new URL(. . .)
  - } catch (MalformedURLException e)
  - {
  - . . .
  - // exception handler code here
  - . . .
  - }

创建和使用URL访问网上资源

- 参见程序 Url1.java
  - 获得URL对象的各个属性
  - getProtocol
  - getHost
  - getPort
  - getFile
  - getRef

创建和使用URL访问网上资源

- 为获得URL的实际比特或内容信息，用它的openConnection( )方法从它创建一个URLConnection对象，如下：
- url.openConnection()
- openConnection( ) 有下面的常用形式：
- URLConnection openConnection( )
- 与调用URL对象相关，它返回一个URLConnection对象。它可能引发IOException异常

URLConnection

- URLConnection是访问远程资源属性的一般用途的类。如果你建立了与远程服务器之间的连接，你可以在传输它到本地之前用URLConnection来检察远程对象的属性。这些属性由HTTP协议规范定义并且仅对用HTTP协议的URL对象有意义

URLConnection

- 参见程序 UrlConnection1.java
  - URL和URLConnection类对于希望建立与HTTP服务器的连接来获取信息的简单程序来说是非常好的
- 参见程序UrlConnection2.java
- 参见程序UrlConnection3.java
  - 比较以上两个程序的细微差别
- 参见程序UrlConnection4.java

InetAddress类

- 无论你是在打电话、发送邮件或建立与Internet的连接，地址是基础。InetAddress类用来封装我们前面讨论的数字式的IP地址和该地址的域名。你通过一个IP主机名与这个类发生作用，IP主机名比它的IP地址用起来更简便更容易理解。
- InetAddress 类内部隐藏了地址数字。

InetAddress类

- 工厂方法
  - InetAddress 类没有明显的构造函数。为生成一个InetAddress对象，必须运用一个可用的工厂方法。
  - 工厂方法（factory method）仅是一个类中静态方法返回一个该类实例的约定。对于InetAddress，三个方法 getLocalHost( )、getByName( )以及getAllByName( )可以用来创建InetAddress的实例

InetAddress类

- static InetAddress getLocalHost( )
- throws UnknownHostException
- static InetAddress getByName(String hostName)
- throws UnknownHostException
- static InetAddress[ ] getAllByName(String hostName)
- throws UnknownHostException

InetAddress类

- getLocalHost( )仅返回象征本地主机的InetAddress对象。
- getByName( )方法返回一个传给它的主机名的InetAddress。
- 如果这些方法不能解析主机名，它们引发一个UnknownHostException异常。
- 在Internet上，用一个名称来代表多个机器是常有的事。getAllByName()工厂方法返回代表由一个特殊名称分解的所有地址的InetAddresses类数组。在不能把名称分解成至少一个地址时，它将引发一个Unknown HostException异常。

InetAddress类

- 参见程序 InetAddress1.java

使用TCP/IP的套接字(Socket) 进行通信

- 什么是Socket?
- 使用Socket进行网络通信的过程
- 使用ServerSocket和Socket实现服务器端和客户端的 Socket通信
- 实现服务器支持多个客户

套接字(socket)的引入

- 为了能够方便的开发网络应用软件，由美国伯克利大学在Unix上推出了一种应用程序访问通信协议的操作系统调用socket(套接字)。socket的出现，使程序员可以很方便地访问TCP/IP，从而开发各种网络应用的程序。
- 随着Unix的应用推广，套接字在编写网络软件中得到了极大的普及。后来，套接字又被引进了Windows等操作系统中。Java语言也引入了套接字编程模型。

套接字(socket)的引入

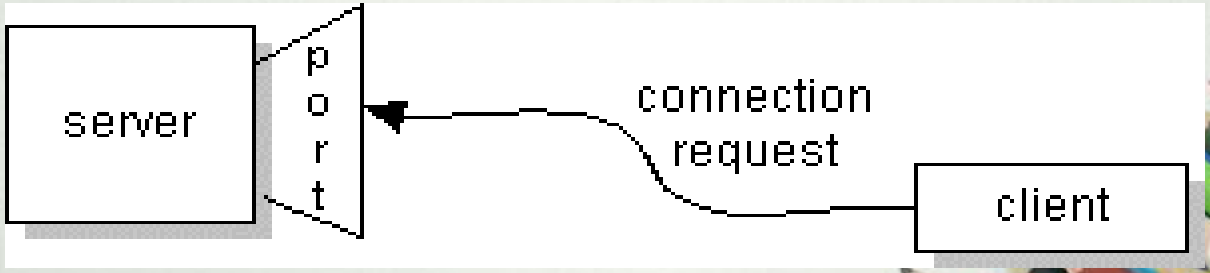
- 什么是Socket?

  Socket是连接运行在网络上的两个程序间的双向通讯的端点

使用TCP/IP的套接字(Socket) 进行通信

使用Socket进行网络通信的过程

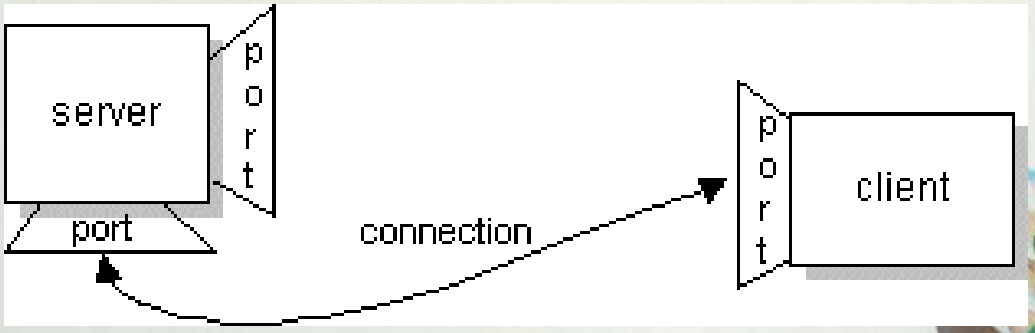
- 服务器程序将一个套接字绑定到一个特定的端口，并通过此套接字等待和监听客户的连接请求。
- 客户程序根据服务器程序所在的主机名和端口号发出连接请求。



使用TCP/IP的套接字(Socket) 进行通信

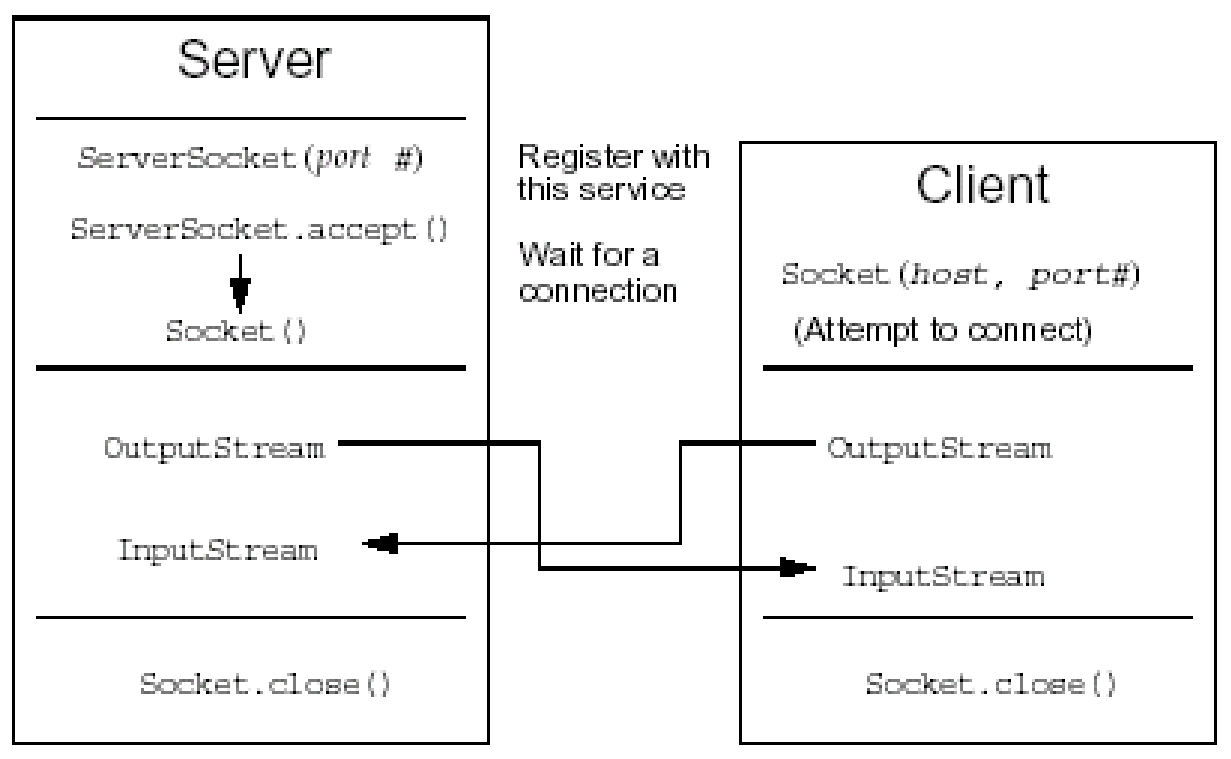
使用Socket进行网络通信的过程

- 如果一切正常，服务器接受连接请求。并获得一个新的绑定到不同端口地址的套接字。
- 客户和服务器通过读、写套接字进行通讯。



使用TCP/IP的套接字(Socket) 进行通信

- 使用ServerSocket和Socket实现服务器端和客户端的 Socket通信



使用TCP/IP的套接字(Socket) 进行通信

- 使用ServerSocket和Socket实现服务器端和客户端的 Socket通信

总结：

1） 建立Socket连接

2） 获得输入/输出流

3）读/写数据

4） 关闭输入/输出流

5） 关闭Socket

基于TCP的socket编程

- 参见程序 Tcp1.java
- 参见程序 EchoServer.java
- 参见程序 EchoClient.java

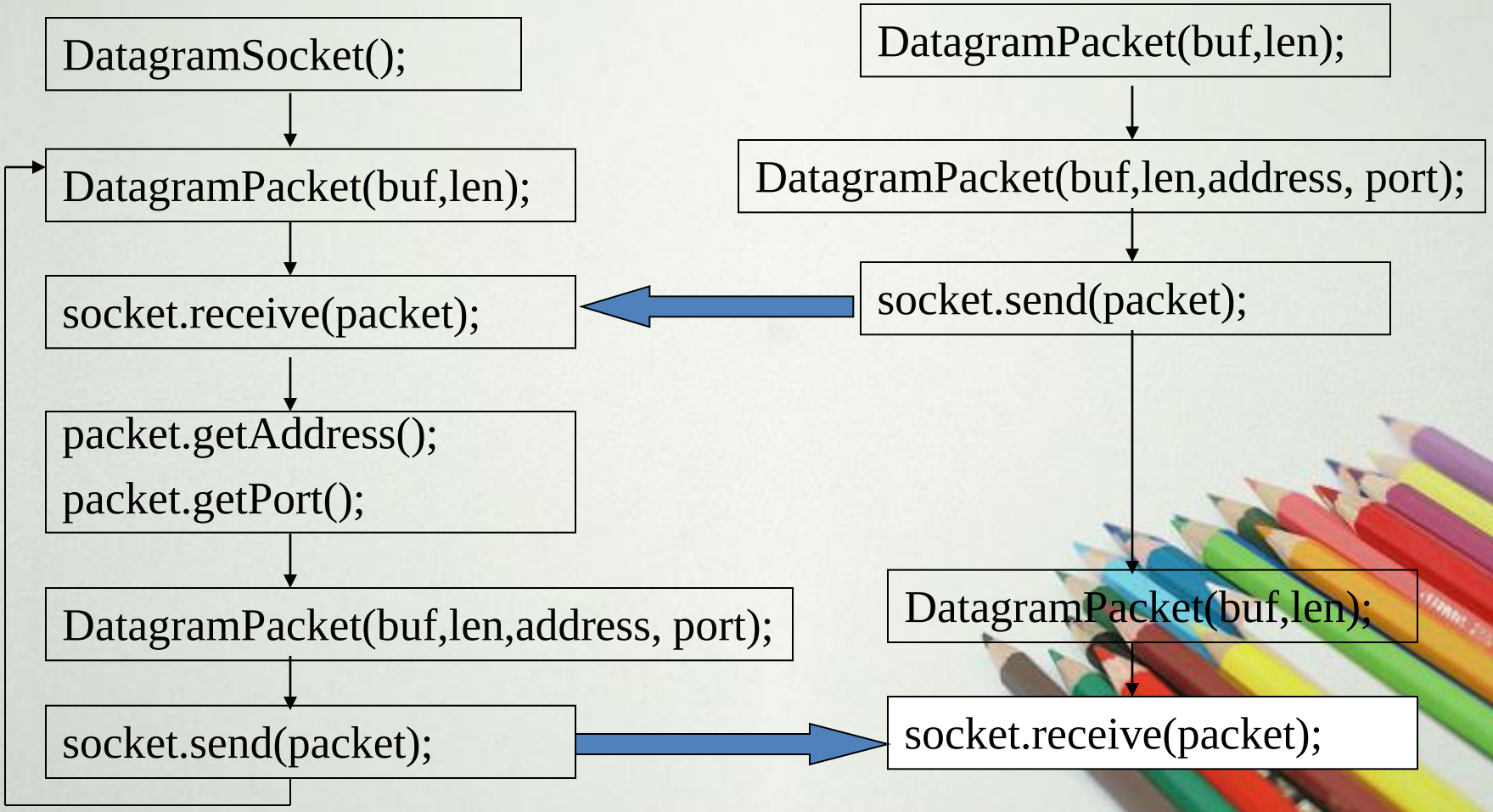
基于TCP的socket编程

- 实现服务器支持多客户机通信
- 服务器端的程序需要为每一个与客户机连接的socket建立一个线程，来解决同时通信的问题。
- 参见程序 EchoServer1.java
- 参见程序 EchoServerThread.java
- 参见程序 EchoClient.java

使用无连接的数据报(UDP) 进行通信

- 什么是Datagram？
  - 数据报是网上传输的独立数据包 ， 数据报是否能正确地到达目的地，到达的时间，顺序，内容的正确性均没有保障。
- java中使用Datagram与DatagramPacket类
- DatagramSocket类利用UDP协议来实现客户与服务器的Socket.
- send():发送数据报
- receive(): 接收数据报

使用无连接的数据报(UDP) 进行通信



基于UDP的socket编程

- 参见程序 Udp1.java
- 参见程序 QuoteServer.java
- 参见程序 QuoteServerThread.java
- 参见程序 QuoteClient.java