面向对象与多线程综合实验

结合华为云平台的基于JAVASE数据挖掘系统设计与实现

主要内容

一
- 实验目标和要求

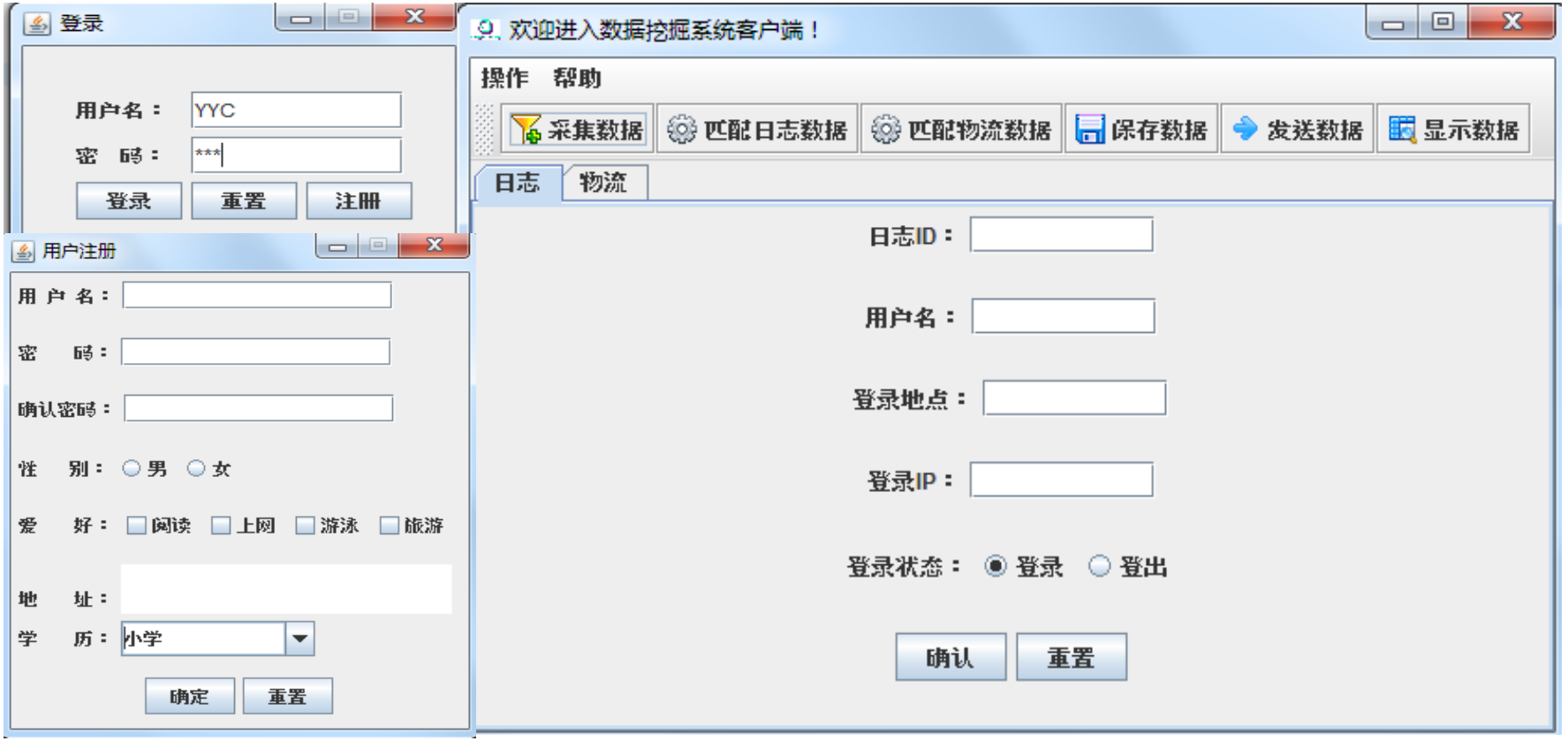
二
- 实验内容

三
- 知识要点

系统简介

- 基于Java SE 的数据挖掘系统

基于客户端服务器端（Client-Server，C-S）模式，实现日志与物流数据信息的采集、匹配、保存、显示等功能，为数据分析挖掘提供基础支撑。



需求分析

- 系统包括客户端应用程序、服务器端应用程序---C/S模式
- 用户和数据信息的保存---JDBC数据库保存和查询
- 用户能够进行注册和登录，授权后使用系统---GUI登录和注册界面设计与功能实现
- 能够实现日志和物流信息的数据采集（录入），登录登出对匹配、信息保存和数据显示等功能---GUI主界面设计与功能实现
- 系统能够进行数据自动刷新功能，与数据库保持同步---线程
- 客户端与服务器端交互，客户端能够将数据发送到服务器端，服务器端接收客户端发送的日志和物流信息，进行保存和处理---Socket通信
- 系统优化---JAVA高级应用与新特性

任务分配

课时分配	实验任务	任务列表	主要知识点分解
第1-2次课	基于控制台的系统数据**采集、匹配、显示**和**记录**功能实现	✓ 注册华为软开云账户 ✓ 搭建数据挖掘系统**框架** ✓ 实现日志和物流数据信息的**采集**、**匹配**和**显示**功能 ✓ 实现匹配的物流数据信息的文件保存和读取**记录**功能	继承与多态/异常处理/集合/文件存储及IO流/华为软开云
第3次课	基于**JDBC**的控制台系统基本功能实现	✓ 创建项目所需的数据库表，并搭建数据访问基础环境 ✓ 实现并测试匹配的日志、物流信息的数据库保存和查询功能	JAVAJDBC/MySQL
第4次课	基于SwingGUI的系统**注册**和**登录界面**设计实现	✓ 创建用户数据库表、用户实体类和用户业务逻辑类 ✓ 创建用户注册窗口，并将用户注册信息保存到数据库 ✓ 创建用户登录窗口，登录成功则进入系统主界面	SwingGUI/事件驱动/WinBuilder
第5次课	基于SwingGUI系统**主界面**设计实现和系统**优化**	✓ 实现主界面中的菜单和工具栏 ✓ 实现主界面中的日志和物流**数据采集**、**匹配**、**保存**和**显示**功能 ✓ GUI系统优化	高级UI组件
第6次课	系统信息**自动刷新**功能实现	✓ 使用线程实现每隔**1分钟(或自定义)**日志和物流显示数据表格**自动刷新**功能，以便与数据库保持同步	线程
第7次课	系统客户端/服务器端数据**发送(交互)**功能实现	✓ 客户端应用程序：修改主界面**发送数据**页面响应，即使用Socket实现数据由客户端发送到服务器 ✓ 服务器端应用程序：使用Server Socket实现接收客户端发送的日志和物流数据信息，并将信息保存到数据库	Socket网络编程
第8次课	系统验收	✓ 数据挖掘系统的基本功能实现与演示 ✓ **在华为软开云平台完成系统需求分析和设计反推**	PPT演示、系统运行
拓展	Java 高级应用以及Java8新特性	◆ 使用注解重新迭代升级系统代码 ◆ 使用格式化将输出的日期进行格式化输出 ◆ 使用Lambda表达式迭代升级主窗口中“帮助”菜单的事件处理 ◆ 使用Lambda表达式实现查找指定的匹配信息并显示	增加注解和格式化以及Lambda优化和查询

功能要求

系统基于C/S模式，包括客户端和服务器端应用程序

- 1. 用户登录和注册功能:用户验证口令通过后登录系统，新用户进行注册，并将注册信息保存数据库
- 2. 日志、物流数据的采集功能：对日志和物流数据进行采集，并保存到Mysql数据库；
- 3. 日志、物流数据的筛选匹配功能，以日志信息为例：
  - 根据日志的登录、登出状态，对日志进行分类，分别存放到**登录**日志集合（ArrayList<LogRec> logIns）和**登出**日志集合（ArrayList<LogRec> logOuts）中。
  - 在登录日志和登出日志中，根据**用户名**和IP地址进行匹配：如果存在相同的用户名和IP地址，则日志信息匹配成功，将匹配的日志数据封装到MatchedLogRec对象，并保存到匹配日志集合（ArrayList<MatchedLogRec> matchLogs）中。
- 4. 日志、物流数据的数据保存功能：在系统主界面中点击“保存数据”按钮时，将匹配的日志数据和物流数据保存到本地文件和数据库中。
- 5. 客户端服务器端交互功能：
  - ✓ 客户端的数据发送功能：在客户端通过Socket技术向服务器端发送匹配的日志数据和物流数据；当数据发送成功后，清空客户端暂时存放数据的集合，然后弹出信息提醒；
  - ✓ 服务器数据查询功能：当点击客户端显示数据功能时，服务器端从数据库中查找符合条件的数据，并发送到客户端；在客户端以表格的形式将数据显示。

实验目标和要求

实验目标

1. 理解与掌握知识点：
   - 掌握基于TCP的Java网络编程Socket类与ServerSocket类的方法及使用
   - 了解Java网络编程相关API
   - 了解网络基础的相关概念
   - 进一步复习线程相关知识
2. 熟悉工具与平台：Windowbuilder插件完成GUI界面开发
3. 开发程序：实现系统GUI图形界面的数据发送功能

实验要求与交付

要求：

1. 客户端主界面窗口数据发送功能的实现，可以将匹配后的日志物流数据发送到服务器端；

2. 服务器端应用程序，实现接收所有客户端发送的日志和物流信息，并将信息保存到数据库；

3. 程序运行时，先选定一台服务器为网络服务器，并修改属性文件中的服务器IP与选定的机器IP一致，在服务器上运行服务器应用程序，然后在客户机运行客户端程序。

✱注意：本次实验简化服务器与客户端都为本机。

✱交付：

1. 代码与运行结果

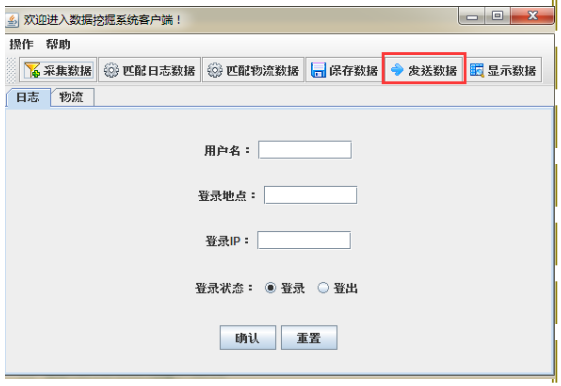
2. 开发过程中遇到问题截图和解决办法

本次实验内容

实验内容

1. 做什么

◆ 实现系统GUI图形界面的数据发送功能



2. 怎么做

1. 更新客户端应用程序，将匹配后的日志和物流数据发送到服务器应用程序

2. 新建服务器端应用程序，实现接收所有客户端发送的日志和物流信息，并将信息保存到数据库

3. 依次运行服务器端与客户端应用程序

3. 对应项

➢ 客户端应用程序

Socket类

更改《发送数据》监听事件处理代码

SendDataListener

更新《保存数据》监听事件处理代码

SaveDataListener

➢ 服务器端应用程序

新建net包，DmsNetServer.java：

1. 创建并开启接收日志和物流数据的线程类
2. 将客户端发送的数据保存到数据库

➢ 运行程序

1. 开启服务器端应用程序
2. 运行客户端应用程序

实验内容系统框架


q_dms_chapter07Socket
src
com.qst.dms.db
com.qst.dms.dos
com.qst.dms.entity
com.qst.dms.exception
com.qst.dms.gather
com.qst.dms.net
DmsNetServer.java
com.qst.dms.service
LogRecService.java
TransportService.java
UserService.java
com.qst.dms.ui
LoginFrame.java
MainFrame.java
RegistFrame.java
com.qst.dms.util
q_dms_mysql.sql
q_dms.sql
引用的库
JRE 系统库 [jre1.8.0_191]
config
images
libs
MatchedTransports.txt
MatchLogs.txt
数据访问工具
控制台程序
数据实体
服务器应用程序
登录界面
主窗口界面
注册界面

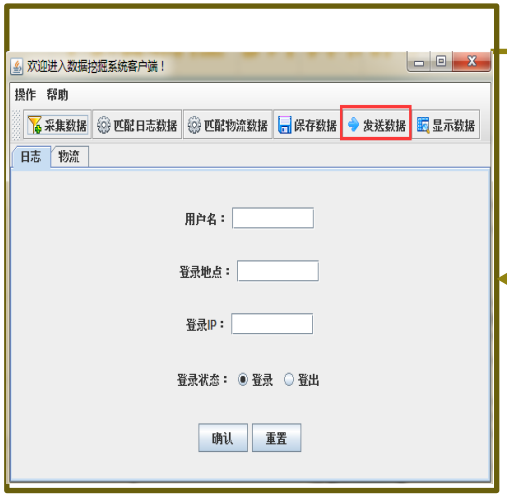
本实验课网络编程内容

分别编写数据挖掘客户端应用程序和服务器端应用程序

客户端应用程序

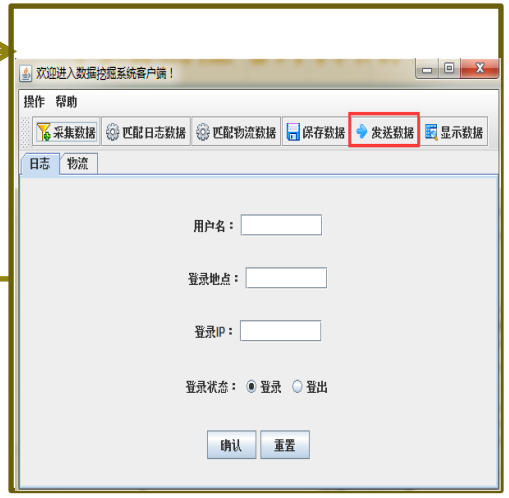
因特网Internet

服务器端应用程序



客户端程序把匹配后的日志和物流数据发送到服务器

服务器端程序接收传来的数据并保存到数据库



客户端应用程序

配置服务器端的IP地址

- 将服务器的IP地址写在config目录下.properties文件中，便于后期服务器移植与维护
- MainFrame声明成员变量serverIP,并在构造方法中获取serverIP的值

mysql.properties

```
driver =com.mysql.jdbc.Driver
url = jdbc:mysql://127.0.0.1:3306/q_dms?useUnicode=true&characterEncoding=UTF-8&useSSL=false
user = root
password = 123456
serverIP=127.0.0.1
```

MainFrame.java

```
// 服务器IP地址
private String serverIP;

// 构造方法
public MainFrame() {
    ...

// 开启更新表格数据的线程
new UpdateTableThread().start();

// 从配置文件中获取网络通信服务器的IP地址
serverIP = Config.getValue("serverIP");
```

客户端应用程序

将匹配后的日志和物流数据发送到服务器

- MainFrame类中增加发送数据监听类SendDataListener，将匹配物流信息发送到服务器
- 并为数据发送菜单注册监听，以下为日志信息发送到服务器为例

```
// 发送数据监听类
private class SendDataListener implements ActionListener { ...
    // 发送数据的事件处理方法
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            // 判断匹配的日志信息列表是否不为空
            if (matchedLogs != null && matchedLogs.size() > 0) {
                // 创建Socket发送日志信息，日志信息发送到服务器的6666端口
                Socket logSocket = new Socket(serverIP, 6666);
                // 创建用于序列化匹配日志信息对象的输出流
                ObjectOutputStream logOutputStream = new ObjectOutputStream(
                        logSocket.getOutputStream());
                // 向流中写入匹配的日志信息
                logOutputStream.writeObject(matchedLogs);
                logOutputStream.flush();
                logOutputStream.close();
                // 因匹配的日志信息已发送到服务器，因此清空日志列表
                matchedLogs.clear();
                logList.clear();
                // 显示对话框
                JOptionPane.showMessageDialog(null, "匹配的日志数据已发送到服务器！",
                        "提示", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "没有匹配的日志数据！", "提示",
                        JOptionPane.WARNING_MESSAGE);
            }
            if (matchedTrans != null && matchedTrans.size() > 0) {
                //补充代码，创建Socket发送物流信息，日志信息发送到服务器的6668端口
```

客户端应用程序

将客户端保存数据到数据库方法注释！

- MainFrame类中更改保存数据监听类SaveDataListener，将原来数据保存到数据库代码注释掉，即客户端的匹配数据需发送到服务器端，由服务器端接收后再保存到数据库，实现客户端和服务器端的业务分离

```
// 保存数据监听类
private class SaveDataListener implements ActionListener {
    // 数据保存的事件处理方法
    @Override
    public void actionPerformed(ActionEvent e) {
        if (matchedLogs != null && matchedLogs.size() > 0) {
            // 保存匹配的日志信息
            logRecService.saveMatchLog(matchedLogs);
            //logRecService.saveMatchLogToDB(matchedLogs);
            //数据保存，清空所采集的数据
            //20191211注释该行，这里先清空的话，集合为空，服务器端无法把集合保存到数据库
            //matchedLogs.clear();
            logList.clear();
            // 显示对话框
            JOptionPane.showMessageDialog(null, "匹配的日志数据已保存到文件中！", "提示",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "没有匹配的日志数据！", "提示",
                    JOptionPane.WARNING_MESSAGE);
        }
        if (matchedTrans != null && matchedTrans.size() > 0) {
            // 保存匹配的物流信息
            transportService.saveMatchedTransport(matchedTrans);
            // transportService.saveMatchTransportToDB(matchedTrans);
            // 显示对话框
            JOptionPane.showMessageDialog(null, "匹配的物流数据已保存到文件中！", "提示",
                    JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "没有匹配的物流数据！", "提示",
                    JOptionPane.WARNING_MESSAGE);
        }
    }
}
```

服务器端应用程序

- 新建net包，DmsNetServer类，接收所有客户端发送的日志和物流数据，并将信息保存到数据库
- 开启接收日志和物流的线程类监听客户端连接请求并实现数据接收和
- 以日志数据线程类AcceptLogThread为参考，完成物流数据的服务器端数据接收和保存到数据库

```java
package com.qst.dms.net;

import java.io.IOException;

// 服务器端应用程序，接收客户端发送来的数据保存到数据库中
public class DmsNetServer {

    public DmsNetServer() {
        // 通过使用不同的端口区分接收不同数据：6666端口是日志，6668端口是物流
        // 开启监听6666端口的线程，接收日志数据
        new AcceptLogThread(6666).start();
        // 开启监听6668端口的线程，接收物流数据
        new AcceptTranThread(6668).start();
        System.out.println("网络服务器已开启...");
    }
```

```java
// 接收日志数据的线程类
private class AcceptLogThread extends Thread {
    private ServerSocket serverSocket;
    private Socket socket;
    private LogRecService logRecService;
    private ObjectInputStream ois;

    public AcceptLogThread(int port) {
        logRecService = new LogRecService();
        try {
            serverSocket = new ServerSocket(port);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // 重写run()方法，将日志数据保存到数据库中
    @Override
    public void run() {
        while (this.isAlive()) {
            try {
                // 接受客户端发送过来的套接字
                socket = serverSocket.accept();
                if (socket != null) {
                    ois = new ObjectInputStream(socket.getInputStream());
                    // 反序列化，得到匹配日志列表
                    ArrayList<MatchedLogRec> matchedLogs = (ArrayList<MatchedLogRec>) ois
                            .readObject();
                    // 将客户端发送来的匹配日志信息保存到数据库
                    logRecService.saveMatchLogToDB(matchedLogs);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        try {
            ois.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

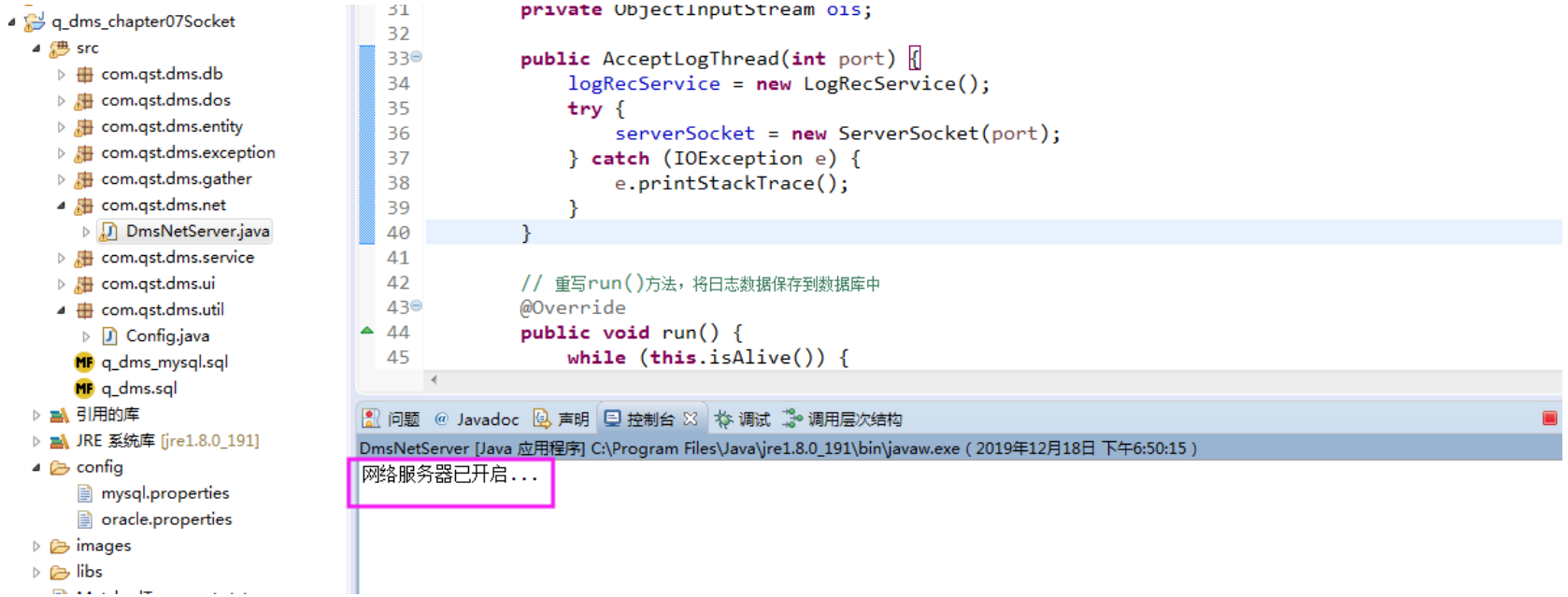
服务器端应用程序

- 开启接收物流数据的线程类AcceptTranThread，监听客户端连接请求并实现物流数据接收

```
    // 接收物流数据的线程类

    private class AcceptTranThread extends Thread {
        //...需补充接收物流数据的线程类

        // 重写run()方法，将数据保存到数据库中

        @Override
        public void run() {
                    }
        }

    // 主程序
    public static void main(String[] args) {
        new DmsNetServer();
    }
}
```

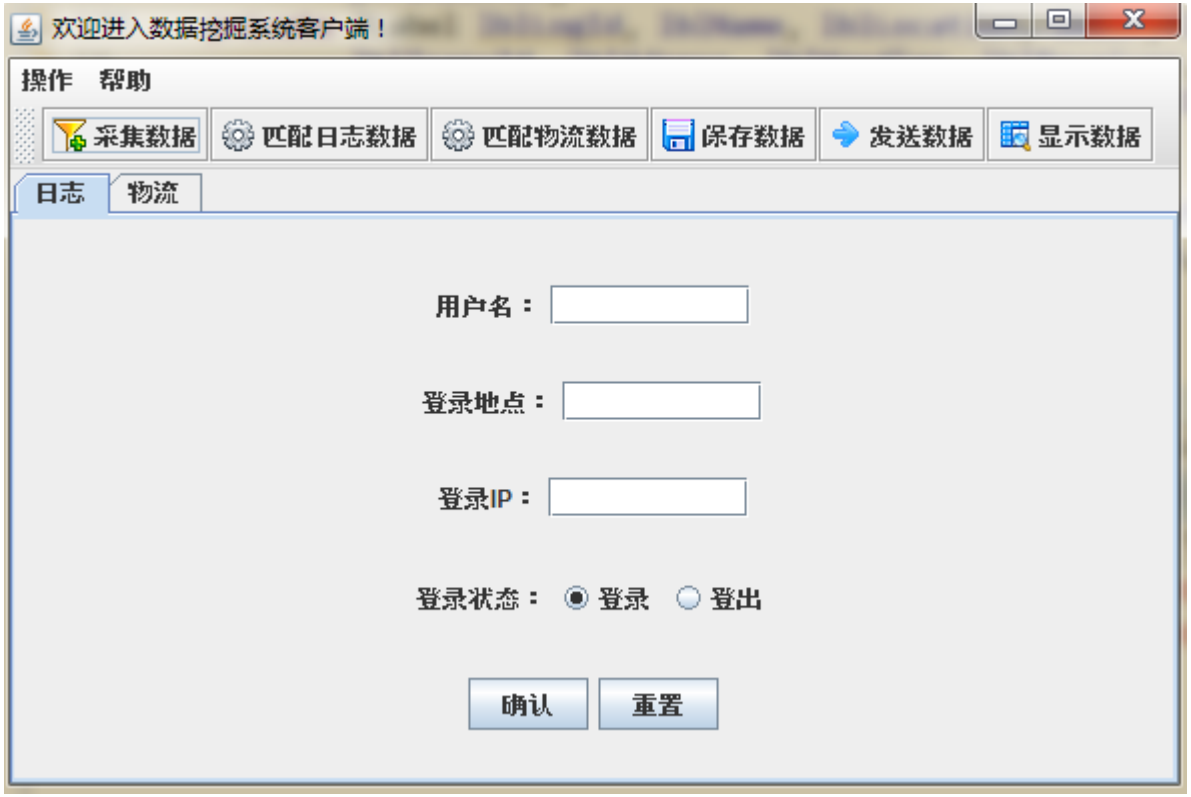
启动服务器程序

- 开启服务器端应用程序DmsNetServer.java，监听客户端连接请求



运行客户端程序请求服务

- 运行客户端应用程序MainFrame.java，实现与服务器端应用程序的通信



知识点

- 计算机网络基础概念
- 网络编程

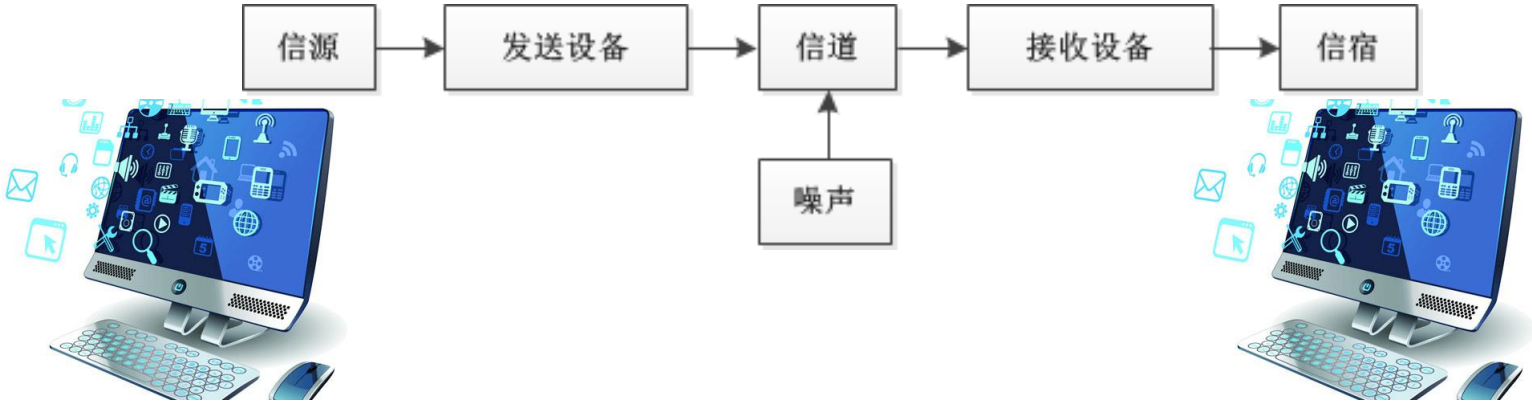
一、网络基础概念

- 计算机网络
- 协议
- TCP/IP四层模型
- IP和端口
- 域名DNS和网络服务

计算机网络

- 计算机网络是通信技术与计算机技术紧密结合的产物

- 通信系统模型：



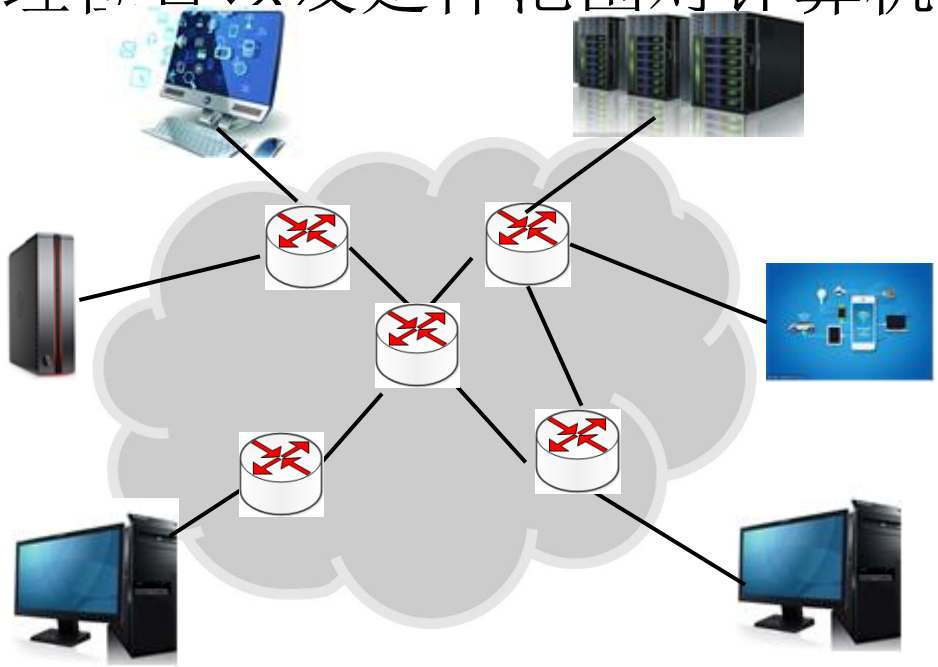
- 计算机网络就是一种通信网络

计算机网络=通信技术+计算机技术

计算机网络

计算机网络就是一种**通信网络**

- 定义----计算机网络是互联的、自治的计算机集合
- 分类----根据规模大小、地理位置以及延伸范围对计算机网络进行分类：
  - ✓ 局域网（LAN）
  - ✓ 城域网（MAN）
  - ✓ 广域网（WAN）



计算机网络=通过交换网络互连

计算机网络Internet

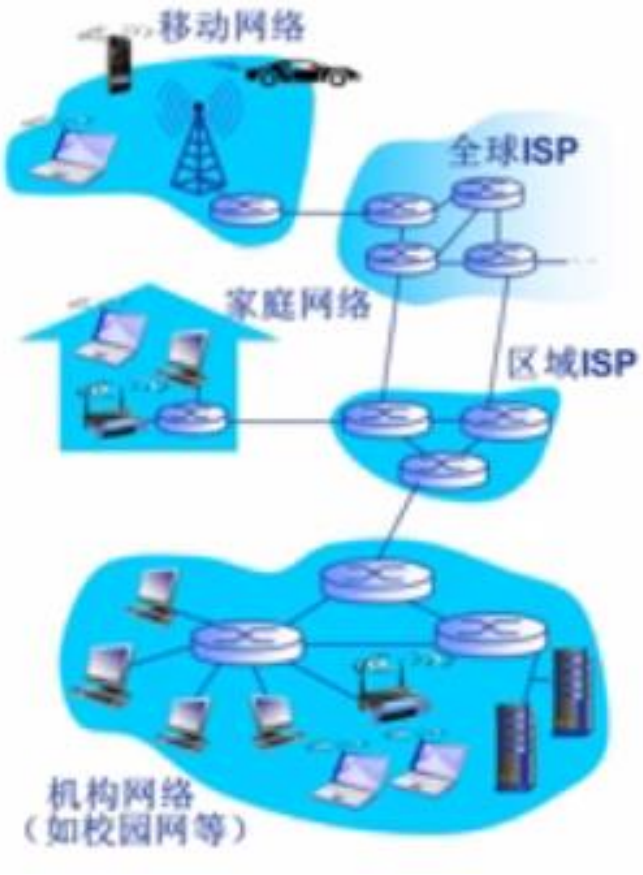
世界上最大的具有代表性的网络

计算机网络===Internet 代名词

Internet 是全球最大的互联网络，是网络互连的“网络之网络”，是大量的互连的计算设备集合，并为网络应用提供通信服务

Wiki定义：

- The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is **a network of networks** that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.----组成
- The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony, and file sharing.----服务



计算机网络协议

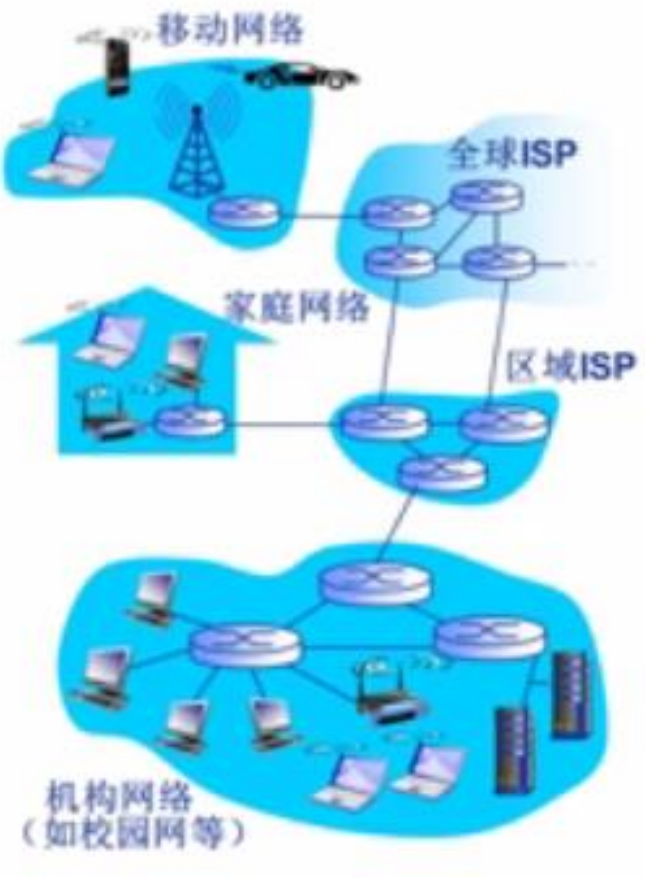
- 仅仅有硬件（主机、链路、路由器、交换机...）互连
- 能够连接上Internet，并保证数据有序传输？
- 还需要协议！

- 计算机硬件（主机、链路、路由器、交换机...）是计算机网络的**基础**
- 计算机网络中数据交换必须遵守实现**约定好的规则**

如交通系统

国道，高速，铁路，飞机

交通规则===协议

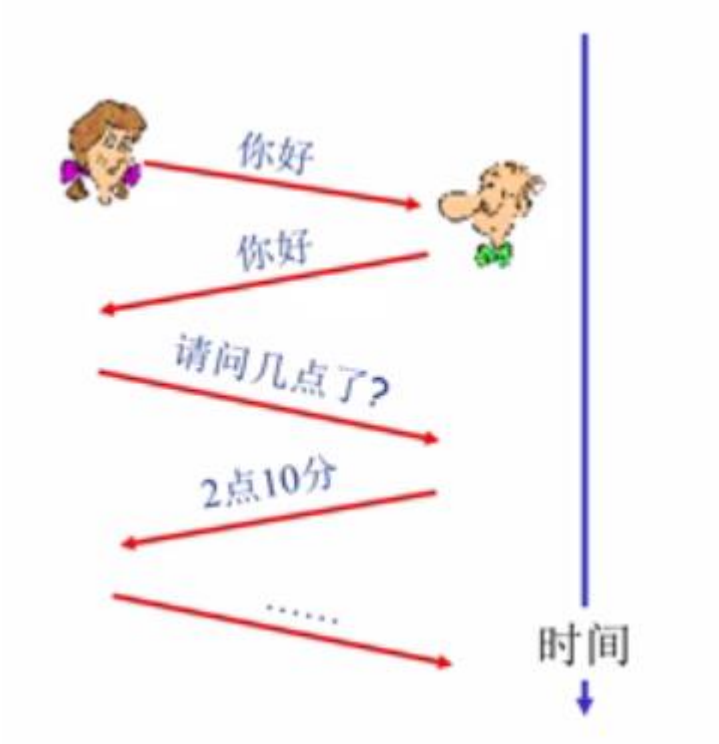


计算机网络协议

任何通信或信息交换过程都需要规则！

人类通信

- 发送特定消息
- 对消息进行响应



计算机网络协议

- 在计算机网络中实现通信必须遵守一些约定，即通信协议，协议规范网络中所有信息发送和接收过程
- 通信协议规定了通信的内容、方式和通信时间，其核心要素有:
  - 语义
  - 语法
  - 时序
- 常见的通信协议包括：
  - TCP/IP协议
  - IPX/SPX协议
  - NetBEUI协议
  - RS-232-C协议、V.35等

计算机网络协议

- Internet 网络中的协议

国际互联网协会制定一系列互连网规范和标准，以RFC文件的形式发布

互联网协议两个重要的协议：

TCP Transmission Control Protocol传输控制协议

IP Internet Protocl 网际协议

TCP/IP协议由网络层的IP协议和传输层的TCP协议组成。连入Internet 中的软硬件必须遵从TCP/IP协议族。

计算机协议

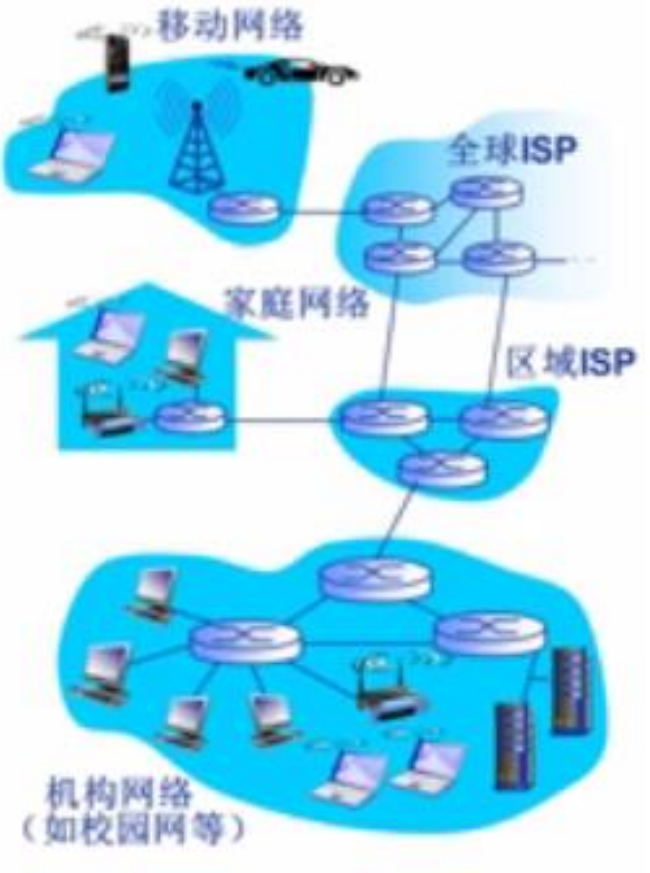
- Internet 网络十分复杂的系统，组成

主机、路由器、链路、应用、协议、软件硬件等等。

是否存在一种**系统结构/模型**能否有效**描述网络**？ 分层结构

每层遵循某些网络协议完成本层功能

TCP/IP网络模型



TCP/IP网络四层模型

传输层协议：**一组关于客户端应用程序与服务器应用程序之间的互相通信的规范和标准**

传输层通信方式：

- ✓ TCP：通信双方先建立**连接connection**，再进行双向数据传输。例如打电话
- ✓ Transmission Control Protocol，TCP传输控制协议（关于有连接通信的规范和标准）
  UDP：直接将数据单向传输给对方，不需要事先建立连接，事后也不需要对方回复。例如　视频电话/电报。 User Datagram Protocol，UDP用户数据报协议

TCP 为应用层提供了可靠的数据传输方式

UDP不是百分百可靠，占用带宽少

网络通信的本质就是网络应用程序之间的数据传输

端到端之间的进程通信 ：传输形式，每次传输数据都会有一个发送方和一个接收方。

TCP/IP网络四层模型

程序员编程Web服务等通用网络服务程序，首先应该了解TCP/IP网络中相关的应用层协议，按照协议要求编程。
例如，想编写新的浏览器程序，应学习HTTP协议和HTML语法

编写自己专有的网络服务程序，可以指定自己的应用层协议，明确网络服务的内容和流程，按照要求分别编程客户端和服务器程序。

端口

1.多个网络应用程序共用一条物理链路

...	端口21	端口80	...	65536

TCP或UDP：不同程序监听不同的端口

程序A Host1:21　　程序B Host1:80

网络传输层基于同一物理链路划分多个端口Port，会根据端口号将其所接收的网络数据分发给相应的应用程序。用于区分不同应用程序的通信。
端口：16位整数编号，0-65535
网络应用程序会通过某个端口来检查并接收数据，这被称作监听端口
不同程序监听不同的端口。
这就实现多个程序共用一条物理链路

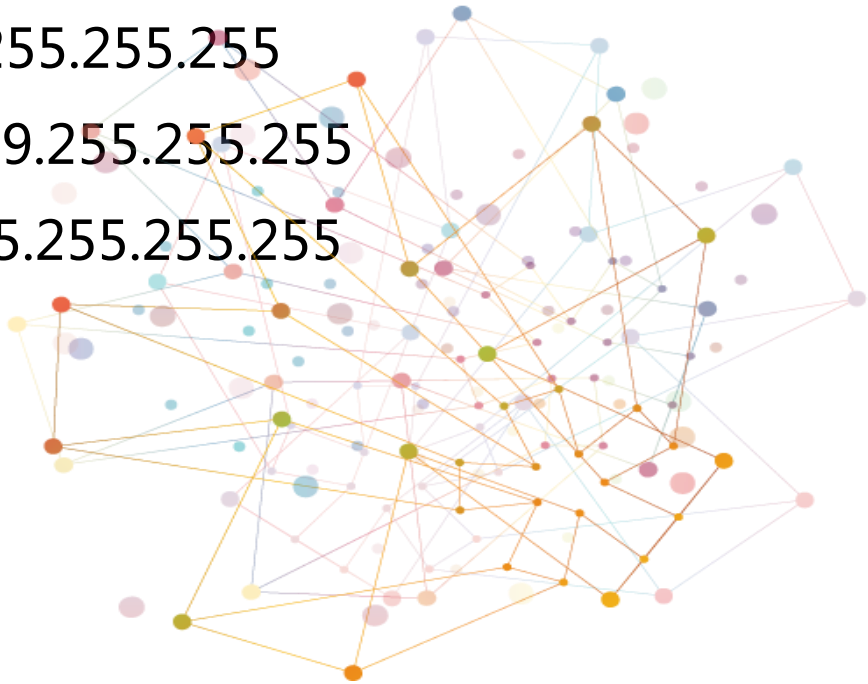
2.发送方如何指定接收方程序

主机名：端口号
IP：端口号

www.whut.edu.cn:80

IP地址和端口

- 在计算机网络中实现通信必须遵守一些约定，即通信协议。IP地址用于唯一地标识网络中的一个通信实体，IP地址被分成五类：
- A类地址：范围1.0.0.0~127.255.255.255
- B类地址：范围128.0.0.0~191.255.255.255
- C类地址：范围192.0.0.0~223.255.255.255
- D类地址：范围从224.0.0.0到239.255.255.255
- E类地址：范围从240.0.0.0到255.255.255.255



IP地址和端口

A、B、C三类地址是由Internet NIC在全球范围内统一分配，其最大网络数及范围如表所示：

类别	最大网络数	IP地址范围	最大主机数	私有IP地址范围
A	126（2^7-2)	0.0.0.0~127.255.255.255	16777214	10.0.0.0~10.255.255.255
B	16384(2^14)	128.0.0.0~191.255.255.255	65534	172.16.0.0~172.31.255.255
C	2097152(2^21)	192.0.0.0~223.255.255.255	254	192.168.0.0~192.168.255.255

IP地址和端口

一些特殊的网址：
每一个字节都为0的地址（0.0.0.0）对应于当前主机

每个字节都为1的地址（255.255.255.255）是当前子网的广播地址

以“11110”开头的E类IP地址都保留用于将来和实验使用

地址中数字127.0.0.1到127.255.255.255用于回路测试

网络ID的第一个8位组也不能全置为“0”，全“0”表示本地网络

端口

端口号是一个整数型标识符来表示，用于表示该数据帧应当交给哪个应用程序来处理，是应用程序与外界交流的出入口，用于表示数据交给哪个通信程序进行处理，通常将端口分为三类：

- 公认端口（Well Known Ports）：从0到1023
- 注册端口（Registered Ports）：从1024到49151
- 动态和/或私有端口（Dynamic and/or PrivatePorts）：从49152到65535

端口

端口是通过端口号来标记的，常用端口及其对应服务

端口号	服务
7	Echo服务端口
21	FTP服务端口
23	Telnet服务端口
25	SMTP服务端口
80	HTTP服务端口

自己编写的应用程序尽量避免使用表中的公认端口值。

域名与DNS

端口域名是由一串用点分隔的字符串所组成的Internet上某一台计算机或计算机组的名称。

域名由两个或两个以上的词构成，中间由“.”号分隔开。

通常由特定字符集、英文字母、数字及“-”任意组合而成，但开头和结尾均不能含有“-”符号。

域名中的字母不分大小写，最长可达67个字节。域名不仅便于记忆，而且即使在IP地址发生变化的情况下，通过改变解析对应关系，域名仍可保持不变。目前也有一些其他语言的域名，如中文域名。

域名

按照级别，可以将域名分为：

- 顶级域名
  - 国际顶级域名（iTDs）
  - 国家顶级域名（nTLDs）
- 二级域名
- 三级域名

DNS

- DNS（Domain Name System，域名系统）是域名与IP地址之间相互映射的一个分布式数据库。

- 通过DNS可以将用户输入的域名解析为与之相关IP地址。

DNS

网络上的一台计算机被称为一个主机HOST 每个主机都有一个主机名，便于记忆并经权威机构登记的名字，域名==对外提供服务的主机名。

www.baidu.com

www.whut.edu.cn

localhost:表示用户当前使用的计算机，本机。127.0.0.1

域名便于记忆，但是计算机网络内部使用的是IP地址。

DNS域名系统是关于域名的应用层协议，规范了域名的格式、域名与IP地址之间的映射。

客户端和服务器

服务器Server：

- 提供网络Service
- 本质计算机 硬件+软件，更强的计算和存储能力GPU/内存 /WinServer/Linux/Unix
- 多种网络服务：

  WWW World Wide Web 网站服务

  email电子邮件服务 收发电子邮件

  FTP（File Transfer Protocol）文件传输服务

网络服务是通过服务器应用程序来提供的，不同的网络服务需要不同的服务器应用程序。

服务器与普通计算机的最大区别：应用程序

客户端Client

- 使用网络服务的计算机系统就是客户端
- PC机、智能手机
- 使用网络服务是通过客户端应用程序

  使用WWW网站服务需要Web客户端程序，如浏览器

  智能手机App Application

网络服务

- 浏览器一个常用的客户端应用程序

www.whut.edu.cn

浏览器如何获得网页信息的呢？

浏览器获取服务器信息的过程

客户端应用程序与服务器应用程序的通信的过程

1. 客户端应用程序发起，发送服务请求
2. 服务器接收请求，将请求的网站主页发送回客户端应用程序

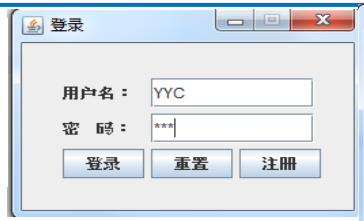
程序之间如何通信？ 通过计算机网络来完成

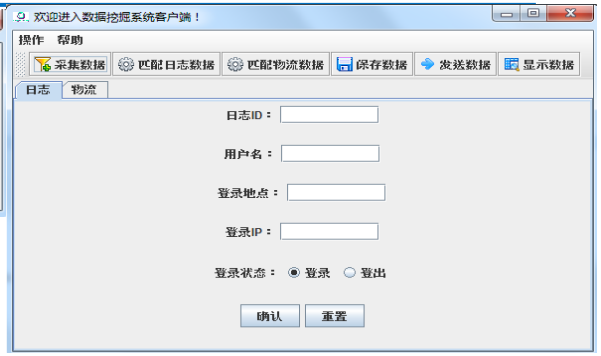
网络编程

- 网络编程要解决的问题
- TCP/IP网络模型
- JAVA 基于TCP的网络编程与Socket类
- JAVA网络编程其他有关的类

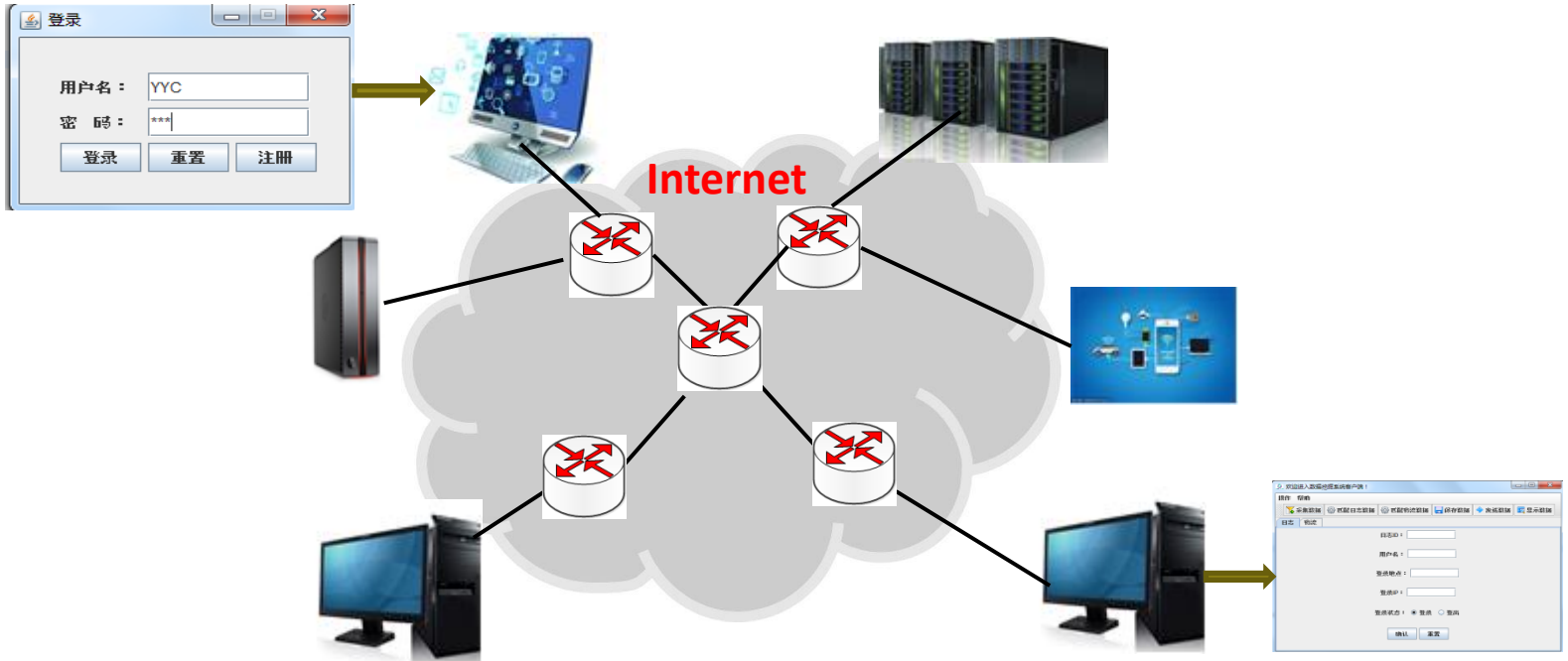
网络编程

1. 应用程序运行在本机，没有与外界通信



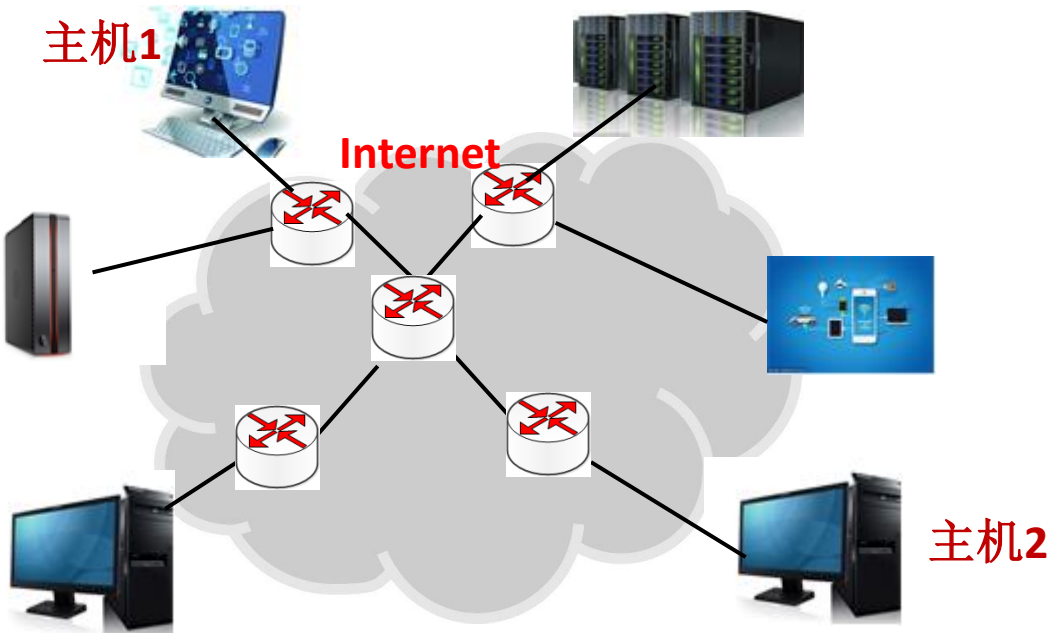


2. 如何能使网络上两台主机应用程序之间进行通信
—网络编程



网络编程需要解决的问题

1. 能够找到网络上这两台主机
2. 能够识别主机上运行的某个应用程序
3. 实现应用程序之间的数据可靠传输

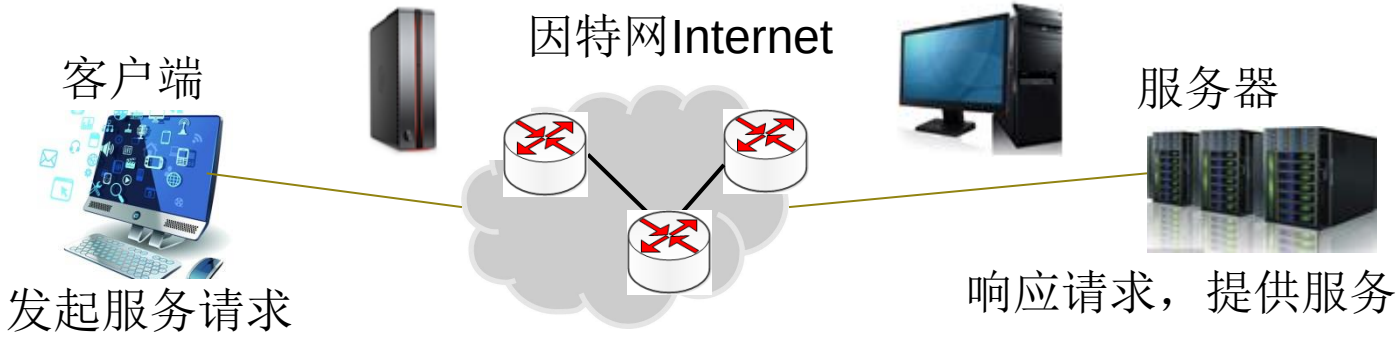


1.IP地址

2.端口号

3.TCP/IP网络分层及协议

TCP/IP网络四层模型



分层思想：

高内聚低耦合

上层屏蔽下层实现细节，只使用其提供的服务

应用层	负责处理应用程序的逻辑	ping、http、DNS
传输层	为两台主机上的应用程序提供端到端的通信	TCP、UDP
网络/际层	实现数据包的路由转发（IP地址），找到主机	ICMP、IP
网络接口层	实现信号在物理媒介上传输（MAC地址）	ARP、RARP、网卡接口…

信息

数据包

信号

客户端

服务器

因特网Internet

信息

发起服务请求

响应请求，提供服务

数据包

信号

客户端应用程序	应用层 规范网络服务的内容与流程 如HTTP FTP DNS	服务器应用程序
通信插槽Socket Java Socket类	传输层 规范信息传输的方式 如TCP/UDP	通信插槽Socket Java Socket类
操作系统	网络层 规范网络地址格式和路由选择算法 如 RIP	操作系统
网卡及驱动、交换机基站等	网络接口层 信息与信号的转换与传输 如以太网、WIFI、5G	网卡及驱动、交换机基站等

信息

数据包

信号

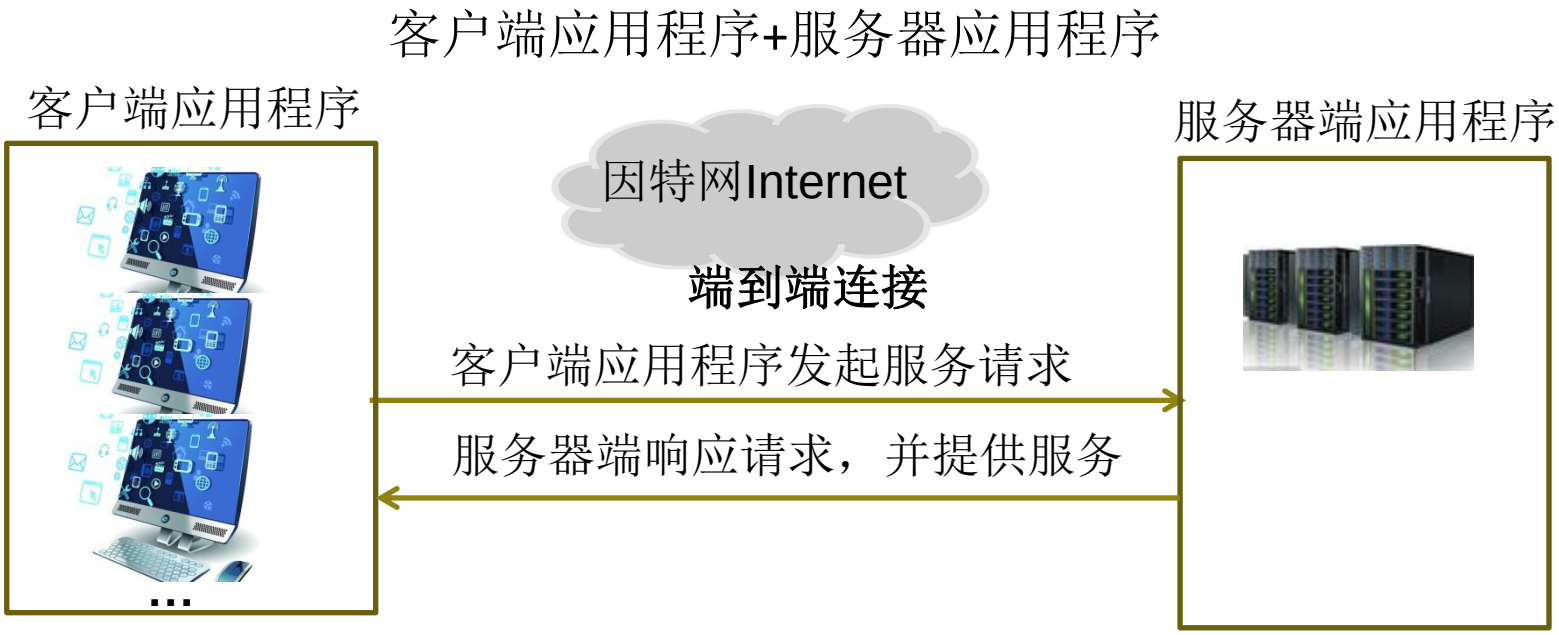
客户端与服务器端具体通信过程？

关于客户端应用程序与服务器应用程序之间的互相通信的规范和标准

传输层就是负责源-目的也就是端到端之间进程之间的完整报文传输。

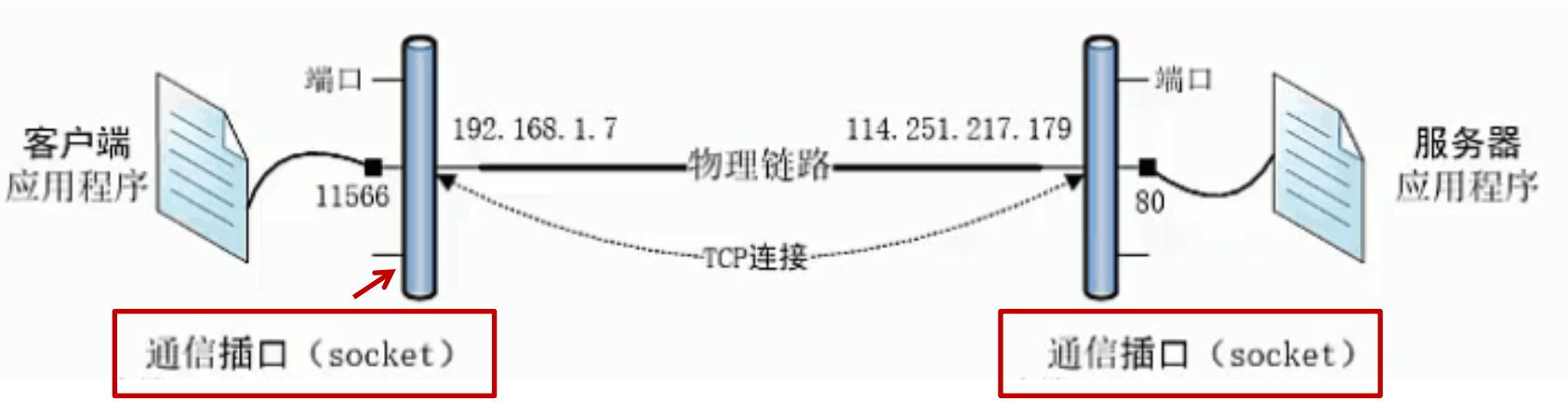
基于TCP协议的网络编程

- TCP（Transmission Control Protocol,传输控制协议）是一种有连接传输层协议，为不同主机的应用程序之间提供一个可靠的像管道一样的连接，保证数据传输的可靠
- 一个完整的**Client/Server**架构网络服务：



基于TCP协议的网络编程 Socket

- 网络上的两个应用程序使用TCP/IP协议进行通信时，会在通信的两端建立一个双向的通信连接管道，这个连接的一端称为一个Socket（套接字）
- Socket可看成网络通信中两台主机之间逻辑连接的端点-通信插口。
- TCP/IP协议在一条物理链路上划分了65536个端口，不同程序使用不同的端口进行通信，这样同一个主机上的多个应用程序可以共用一条物理链路进行通信。
- Socket允许应用程序将网络连接当成一个**IO流**



JAVA基于TCP协议的网络编程

■java.net包中提供了网络编程所需的类，Socket套接字对象封装了TCP通信两端的通信端口，主要使用两种Socket类

1. ServerSocket：是服务器套接字，用于服务器
2. Socket：建立网络连接时使用

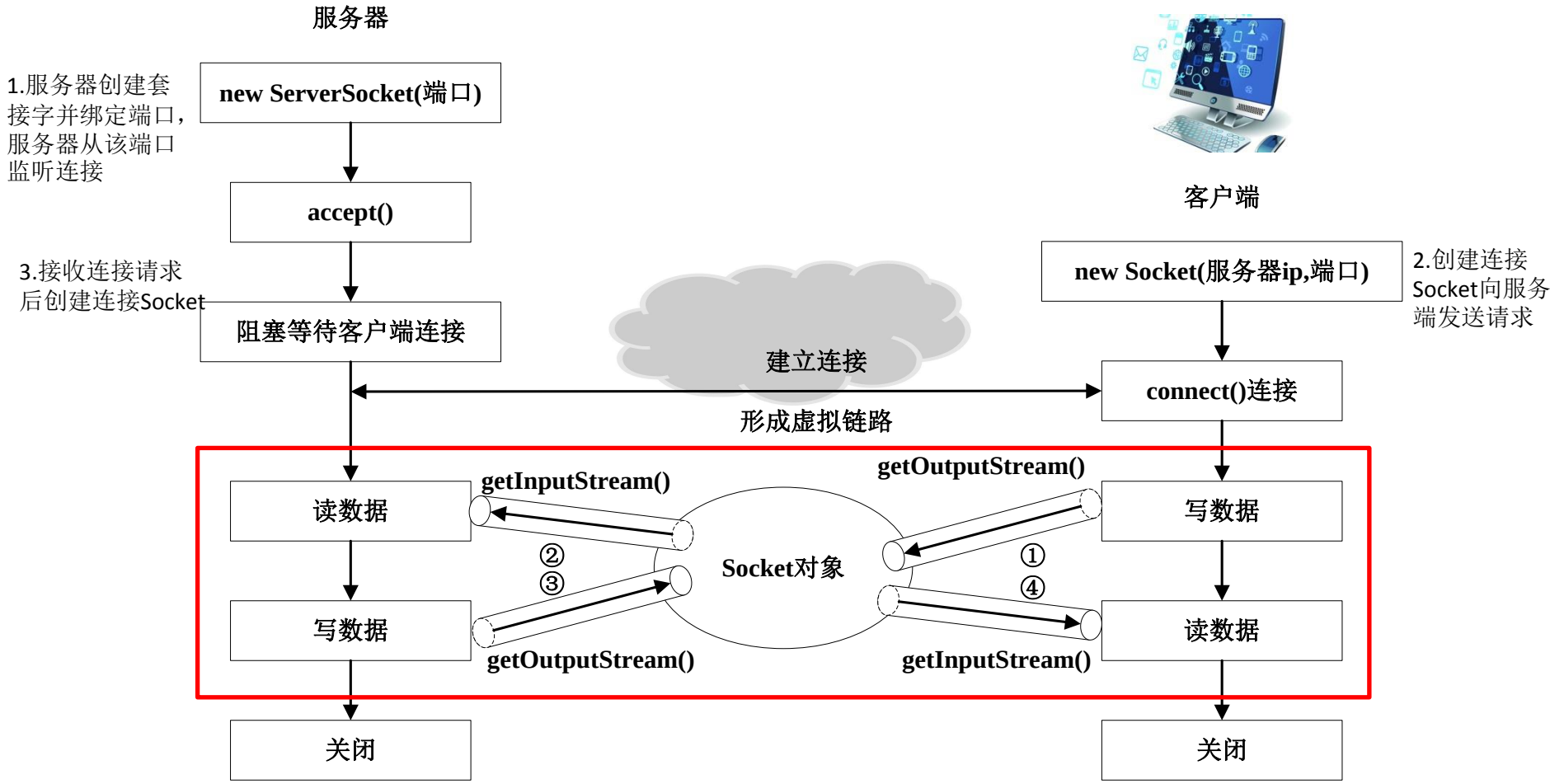
- TCP连接成功后，客户端与服务器应用程序各有一个套接字Socket对象，分别表示TCP连接两端的通信插口。
- 将应用程序从通信插口（Socket对象）**接收**数据抽象成**输入流**，向通信插口**发送**数据抽象成**输出**流，这样**网络通信**问题转换成**输入输出问题**。
- Socket套接字类中的两个重要方法

---getInputStream()获得套接字的字节型输入流对象，**从输入流对象读取数据，就是接收对方发来的消息**

---getOutputStream()获得套接字的字节型输出流对象，向**输出流对象写入数据，就是向对方发送信息**

---还可以根据需要将字节型输入输出流对象包装成其他的流对象，如字符型，带缓冲区的流对象等。

Socket通信模型

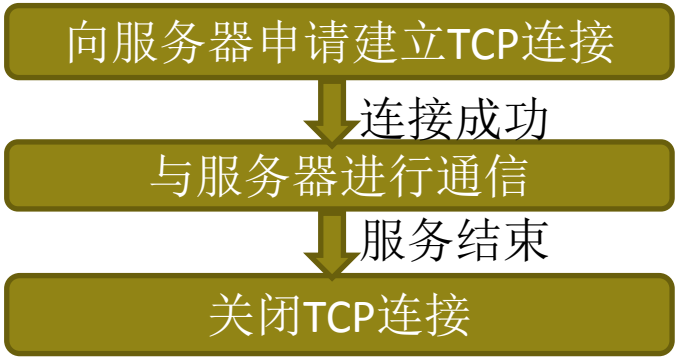


以输入输出的形式实现了网络通信！

基于TCP编写C/S架构代码框架

客户端应用程序
(使用网络服务)

- 算法流程



发送业务请求
接收服务响应

- 代码结构

```
try{//处理可能出现的异常
   //创建套接字对象，向服务器申请建立TCP连接
   Socket sc=new Socket(服务器域名或IP地址，服务端口);
   …//连接成功后与服务器通信，发送服务请求，接收服务响应
   sc.close();//服务结束后断开TCP连接
}
catch(IOExceptione){System.out.println("IOException!");}
```

基于TCP编写C/S架构代码框架

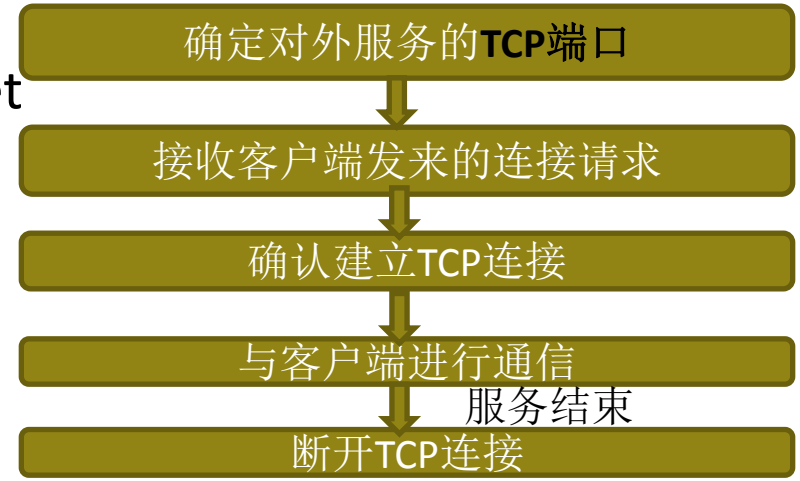
服务器应用程序(提供网络服务)

- 算法流程

服务器套接字类Server Socket

创建服务器套接字类就是来监听某个端口

accept 用于监听服务请求，有请求则建立TCP连接返回套接字对象



- 代码结构

```
try{//处理可能出现的异常
   //创建服务器套接字对象，给出服务端口，
   //后面服务器会一直监听这个端口
       ServerSocket ss=new ServerSocket(服务端口);
       while(true) { //服务器应保持运行状态，随时接受客户端的连接请求
       Socket s=ss.accept();//如果有TCP连接请求，则确认建立连接，返回套接字对象
       …//连接成功后与客户端进行通信，接受服务请求，发送服务响应
        s.close();//服务结束后断开TCP连接
        // 继续循环，准备接受下一个连接请求，可接收不同客户端的连接请求
          }
        ss.close();//服务结束后断开TCP连接
        }
        catch(IOExceptione){System.out.println("IOException!");}
```

基于TCP编写C/S架构代码框架

套接字对象的输入输出流

- TCP连接成功后，客户端与服务器应用程序各有一个套接字Socket对象，分别表示TCP连接两端的通信插口。
- Java API将应用程序从通信插口（Socket对象）**接收**数据抽象成**输入流**，向通信插口**发送**数据抽象成**输出**流，这样**网络通信**问题转换成**输入输出问题**。
- Socket套接字类中的两个重要方法

---getInputStream()获得套接字的字节型输入流对象，**从输入流对象读取数据，就是接收对方发来的消息**

---getOutputStream()获得套接字的字节型输出流对象，向**输出流对象写入数据，就是向对方发送信息**

---可以根据需要将字节型输入输出流对象包装成其他的流对象，如字符型，带缓冲区的流对象等。

Socket类

- 通常客户端使用Socket来连接指定的服务器，常用的构造方法：
- Socket(InetAddress |String host,int port)
- Socket(InetAddress|String host,int port,InetAddress localAddr,int localPort)
- Socket类常用方法：

方法	功能描述
public InetAddress getInetAddress()	返回连接到远程主机的地址，如果连接失败则返回以前连接的主机
public int getPort()	返回Socket连接到远程主机的端口号
public int getLocalPort()	返回本地连接终端的端口号
public InputStream getInputStream()	返回一个输入流，从Socket读取数据
public OutputStream getOutputStream()	返回一个输出流，往Socket中写数据
public synchronized void close()	关闭当前Socket连接

Socket类

- 使用Socket进行网络通信的具体步骤：

1.根据指定IP地址和端口号创建一个Socket对象；

2.调用getInputStream()方法或getOutputStream()方法打开连接到Socket的输入/出流；

3.客户端与服务器根据协议进行交互，直到关闭连接；

4.关闭客户端的Socket。

ServerSocket类

- ServerSocket是服务器套接字，运行在服务器端，通过指定端口主动监听来自客户端的Socket连接

- ServerSocket类常用的构造方法：
  - ✓ ServerSocket(int port)
  - ✓ ServerSocket(int port,int backlog)
  - ✓ ServerSocket(int port,int backlog,InetAddress localAddr)

ServerSocket类的构造方法都声明抛出IOException异常，因此在创建ServerSocket对象必须捕获或抛出异常。另外，在选择端口号时，建议选择注册端口（范围是1024~49151的数），通常应用程序使用这个范围内的端口，以防止发生冲突。

ServerSocket类

- ServerSocket常用的方法：

方法名	功能说明
public Socket accept()	接收客户端Socket连接请求，并返回一个与客户端Socket对应的Socket实例；该方法是一个阻塞方法，如果没有接收到客户端发送的Socket，则一直处于等待状态，线程也会被阻塞用于产生"阻塞"，直到接受到一个连接，并且返回一个客户端的Socket对象实例。 "阻塞"是一个术语，它使程序运行暂时"停留"在这个地方，直到一个会话产生，然后程序继续；通常"阻塞"是由循环产生的。
public InetAddress getInetAddress()	返回当前ServerSocket实例的地址信息
public int getLocalPort()	返回当前ServerSocket实例的服务端口
public void close()	关闭当前ServerSocket实例

ServerSocket类

- 使用ServerSocket进行网络通信的具体步骤：

1.根据指定的端口号来实例化一个ServerSocket对象

2.调用ServerSocket对象的accept()方法接收客户端发送的Socket对象

3.调用Socket对象的getInputStream()/getOutputStream()方法来建立与客户端进行交互的IO流

4.服务器与客户端根据一定的协议交互，直到关闭连接

5.关闭服务器端的Socket

回到第2步，继续监听下一次客户端发送的Socket请求连接

ServerSocket类

- 使用Socket进行基于C/S架构的网络通信程序设计的过程：

  - 服务器端通过某个端口监听是否有客户端发送Socket连接请求;
  - 客户端向服务器端发出一个Socket连接请求;
  - 服务器端调用accept()接收客户端Socket并建立连接;
  - 通过调用Socket对象的getInputStream()/getOutStream()方法进行IO流操作，服务器与客户端之间进行信息交互;
  - 关闭服务器端和客户端的Socket。

Java网络编程API

- Java中有关网络方面的功能都定义在java.net包中
- URL和URLConnection等类提供了以程序的方式来访问Web服务
- URLDecoder和URLEncoder提供了字符串相互转换的静态方法

Java网络编程API

InetAddress类

Java提供InetAddress类来封装IP地址或域名，该类无构造方法，常用方法如下：

方法	功能描述
public static InetAddress getLocalHost()	获得本机对应的InetAddress对象
public static InetAddress getByName (String host)	根据主机获得对应的InetAddress对象，参数host可以是IP地址或域名
public static InetAddress[] getAllByName(String host)	根据主机获得具有相同名字的一组InetAddress对象
public static InetAddress getByAddress(byte[] addr)	获取addr所封装的IP地址对应的InetAddress对象
public String getCanonicalHostName()	获取此IP地址的全限定域名
public bytes[] getHostAddress()	获得该InetAddress对象对应的IP地址字符串
public String getHostName()	获得该InetAddress对象的主机名称
public boolean isReachable(int timeout)	判断是否可以到达该地址

Java网络编程API

- URL类
  - ✓ URL（Uniform Resource Locator，统一资源定位器）表示互联网上某一资源的地址
  - ✓ URL可以由协议名、主机、端口和资源四个部分组成，其语法：

```
protocol://host:port/resourceName
```

  - protocol是协议名
  - host是主机名
  - port是端口
  - resourceName是资源名

Java网络编程API

URL类

- ✓ Java将URL封装成URL类，通过URL对象记录下完整的URL信息。
- ✓ URL类常用方法：

方法	功能描述
public URL(String spec)	构造方法，根据指定的字符串来创建一个URL对象
Public URL(String protocol,String host,int port,String file)	构造方法，根据指定的协议、主机名、端口号和文件资源来创建一个URL对象
public URL(String protocol, String host, String file)	构造方法，根据指定的协议、主机名、和文件资源来创建URL对象
public String getProtocol()	返回协议名
public String getHost()	返回主机名
public int getPort()	返回端口号，如果没有设置端口，则返回-1
public String getFile()	返回文件名
public String getRef()	返回URL的锚
public String getQuery()	返回URL的查询信息
public String getPath()	返回URL的路径
public URLConnection openConnection()	返回一个URLConnection对象
public final InputStream openStream()	返回一个用于读取该URL资源的InputStream流

Java网络编程API

- URL类
  - ✓ URLConnection代表与URL指定的数据源的动态连接
  - ✓ 允许使用POST或PUT和其他HTTP请求方法将数据送回服务器
  - ✓ URLConnection常用方法：

方法	功能描述
public int getContentLength()	获得文件的长度
public String getContentType()	获得文件的类型
public long getDate()	获得文件创建的时间
public long getLastModified()	获得文件最后修改的时间
public InputStream getInputStream()	获得输入流，以便读取文件的数据
public OutputStream getOutputStream()	获得输出流，以便输出数据
public void setRequestProperty(String key,String value)	设置请求属性值

Java网络编程API

- URLDecoder和URLEncoder两个工具类
  - ✓ 在编程过程中涉及到普通字符串和application/x-www-form-urlencoded MIME字符串之间相互转换时
  - ✓ 需要使用URLDecoder和URLEncoder两个工具类
    - URLDecoder类提供了decode(String s,String enc)静态方法
    - URLEncoder类提供了encode(String s,String enc)静态方法

总结

- 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。如何准确的定位网络上的主机；如何可靠高效的进行数据传输。
- 网络上两台计算机之间的通信，本质上实两个网络应用程序之间的通信，网络应用程序与普通应用程序之间最大的区别在于，网络应用程序具有通信功能，能与网络上的其他程序互相通信，协同工作。
- TCP/IP网络传输层为网络应用程序提供了两种不同的通信方式：有连接通信TCP和无连接通信UDP。
- 程序之间的网络通信==利用JAVA API中与网络通信相关的类来编写网络应用程序。
- Socket和ServerSocket分别用来表示双向连接的客户端和服务端，在创建Socket或ServerSocket时必须捕获或声明异常，在Socket对象使用完毕时，要将其关闭，并且遵循一定的关闭次序。

网络编程与计算机网络之间的关系