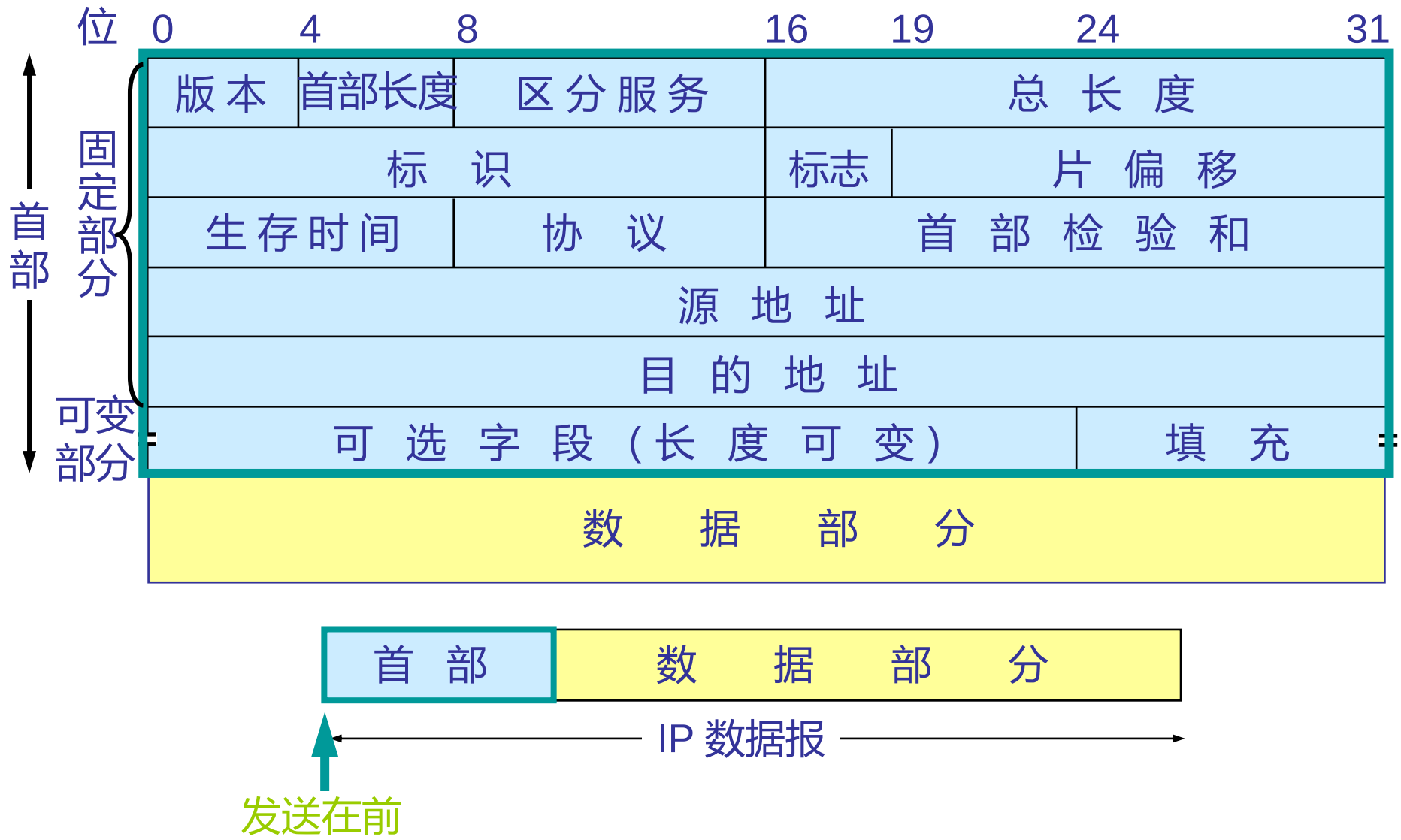
网络程序设计1

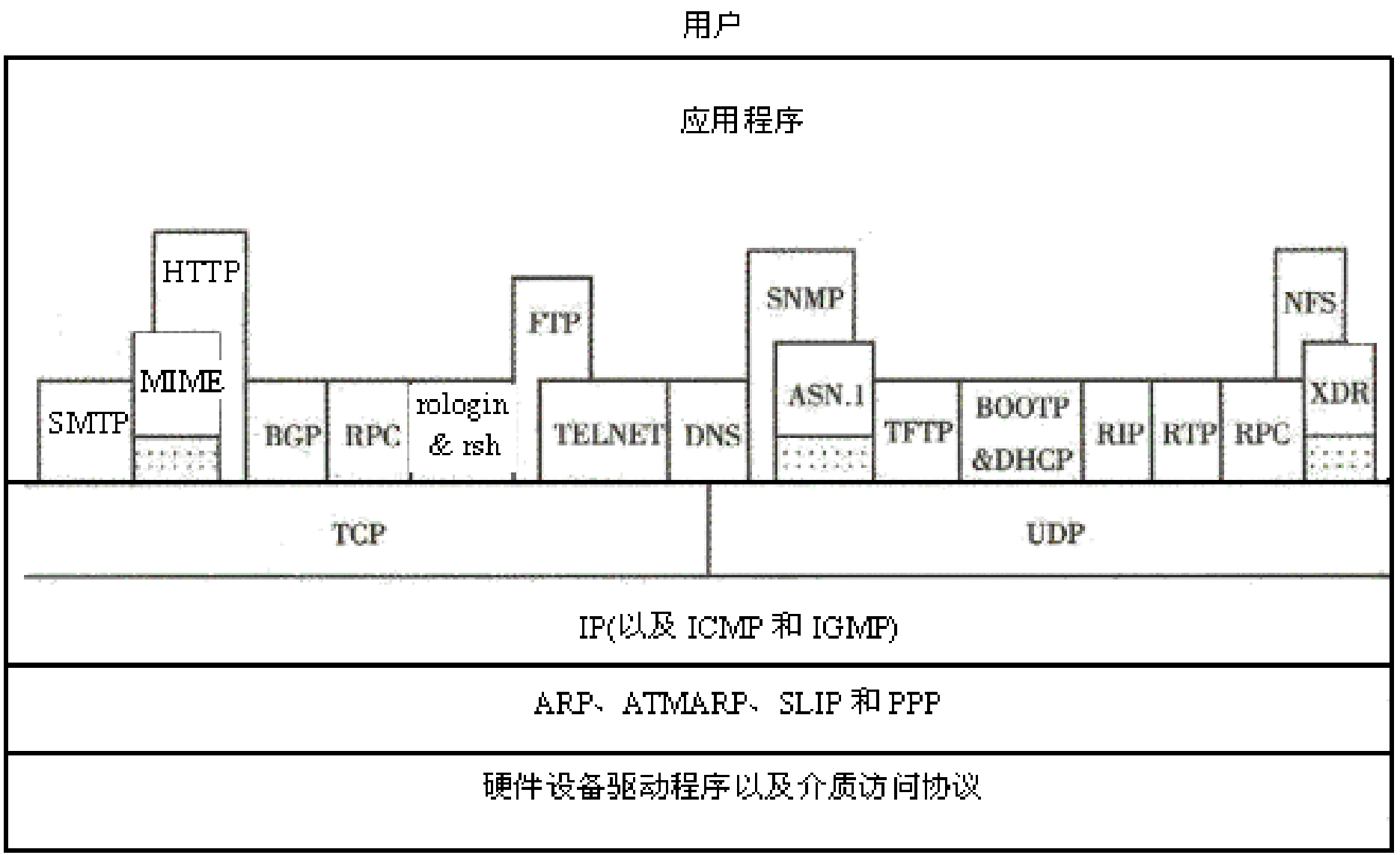
网络程序设计基础


位
0
4
8
16
19
24
31
首部
固定部分
可变部分
版本
首部长度
区分服务
总长度
标识
标志
片偏移
生存时间
协议
首部检验和
源地址
目的地址
可选字段（长度可变）
填充
数据部分

首部
数据部分
IP 数据报
发送在前

端口号

- （1）公认端口（Well Known Ports）：从0到1023，它们紧密绑定（binding）于一些服务。通常这些端口的通讯明确表明了某种服务的协议。例如：80端口实际上总是HTTP通讯。
- （2）注册端口（Registered Ports）：从1024到49151。它们松散地绑定于一些服务。也就是说有许多服务绑定于这些端口，这些端口同样用于许多其它目的。例如：许多系统处理动态端口从1024左右开始。
- （3）动态和/或私有端口（Dynamic and/or Private Ports）：从49152到65535。理论上，不应为服务分配这些端口。实际上，机器通常从1024起分配动态端口。但也有例外：SUN的RPC端口从32768开始



问题：协议在数据传输中如何体现？

问题：怎样设计一个新的协议？

思考

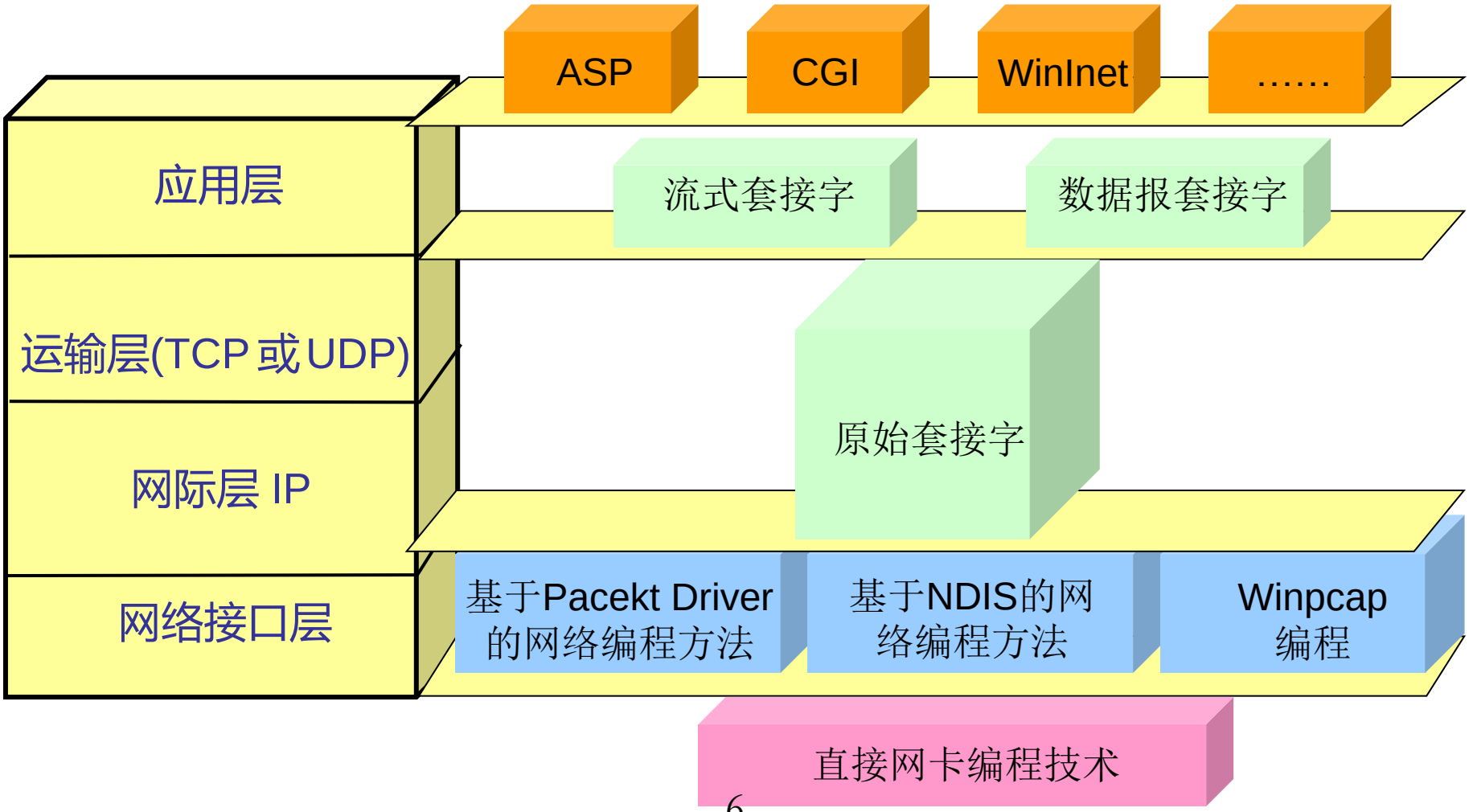
问题1：编写不同层次上的应用程序时，协议封装的对象像各是什么？

问题2：在哪一个层次上封装对编程的影响有哪些？

问题3：在哪一个层次上分用对编程的影响有哪些？

网络程序设计方法纵览

TCP/IP 的体系结构



网络编程方法纵览

- 面向应用的网络编程方法
  - WinInet编程
  - 基于WWW应用的网络编程
  - 面向SOA的Web Service网络编程
- 基于TCP/IP协议栈的网络编程(套接字)
- 面向原始帧的网络编程
  - 直接网卡编程技术
  - 基于Packet Driver的网络编程方法
  - 基于NDIS的网络编程
  - WinPcap编程

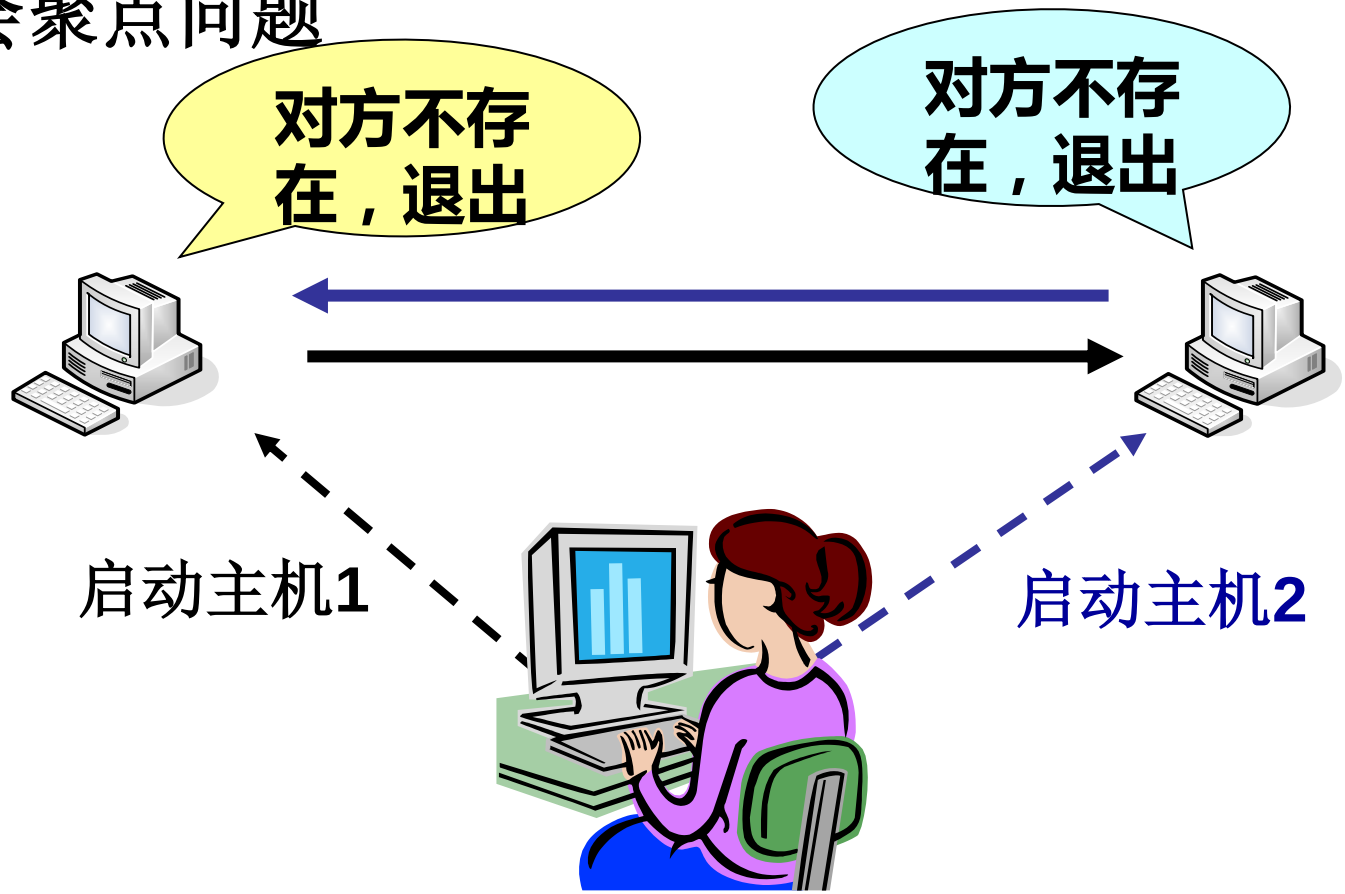
网络程序通信模型

网络应用软件与网络通信之间的关系

- 网络应用程序的三个方面功能
  - 1）实现通信能力
  - 2）处理程序逻辑
  - 3）提供用户交互界面
- 网络通信为网络应用软件提供强大的通信功能，应用软件为网络提供灵活方面的操作平台。

会聚点问题

- 会聚点问题



会聚点问题

- 网络协议只是规定了应用程序在通信时所必须遵循的约定，还有很多关于通信功能和通信实体的组织协调策略尚没考虑，主要包括：
  - 1）确定通信双方的角色，用以部署每个通信实体的具体功能。
  - 2）确定通信双方的通信次序，用以安排不同角色的通信实体的启动和停止时机以及交互顺序。
  - 3）确定通信的传送形式，用以指导应用程序对底层传输服务的选择。

- 会聚点问题的解决方案

  1）要求下层通信协议在请求报文到达后自动创建运行程序

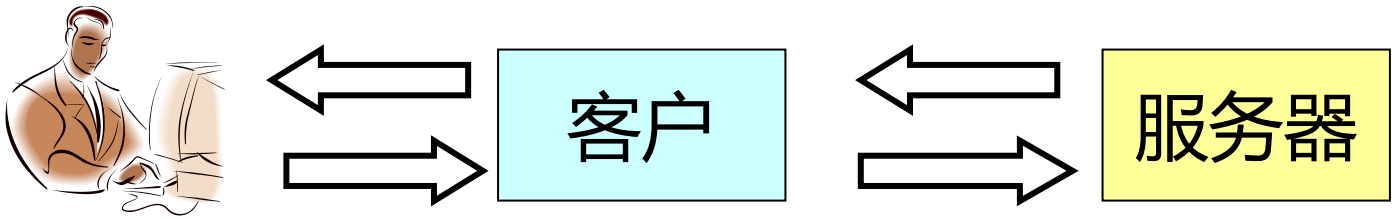
  ——底层通讯程序太复杂！

  2）要求在任何一对进行通信的应用进程中，有一方必须在启动执行后（无限期）等待对方与其联系。

  ——客户－服务器模型的解决方案

客户/服务器模型

- 根据通信发起的方向对基于客户－服务器模型的应用程序进行分类：
- 客户：发起对等通信的应用程序
- 服务器：等待接收客户通信请求的应用程序



客户/服务器交互的一般过程

- 服务器端首先启动，并根据请求提供相应的服务。

① 向操作系统发送请求，请求其打开某个周知公认的端口，以便在该端口上提供服务；

② 等待客户请求达到该端口；

③ 接收服务请求，处理该请求并发送应答；

④ 重复②，等待其它用户请求；

⑤ 关闭服务器端口，释放资源；

客户/服务器交互的一般过程

- 客户方主动请求服务器的服务：

① 向操作系统发送请求，请求其打开一个端口，利用该端口请求服务器服务；

② 向服务器发服务请求报文，等待并接收应答；

③ 重复②；

④ 请求结束后关闭端口，释放资源。

- 说明：

① 客户端与服务器的作用是非对称的，程序编码不同；

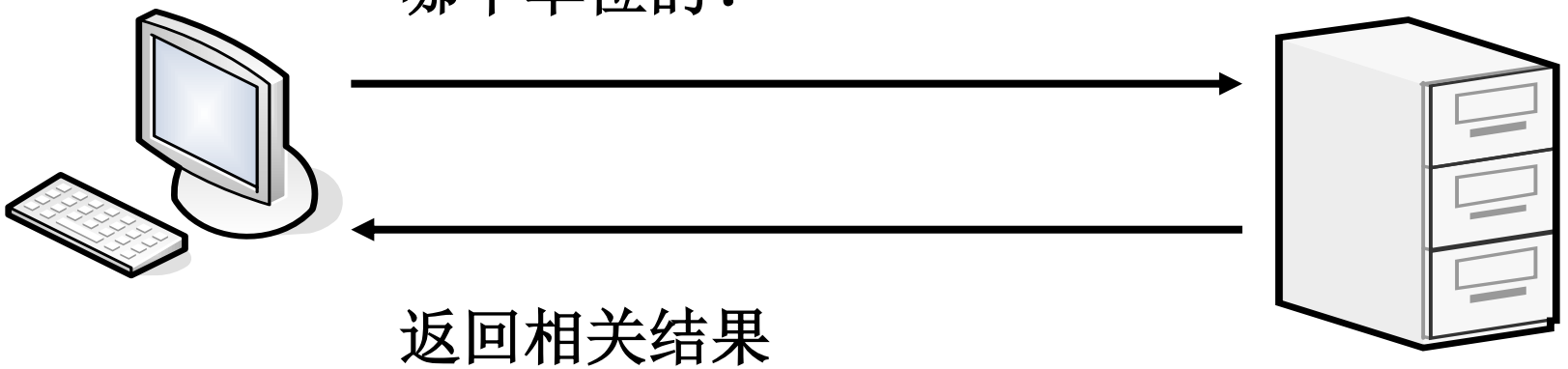
② 服务器进程一般是先于客户请求而启动的。只要系统运行，该服务进程一直存在，直到正常终止或强迫终止。

Whois服务器访问过程举例

客户机

Whois服务器

请问202.196.64.1的地址是
哪个单位的？



返回相关结果

Whois服务器访问过程举例

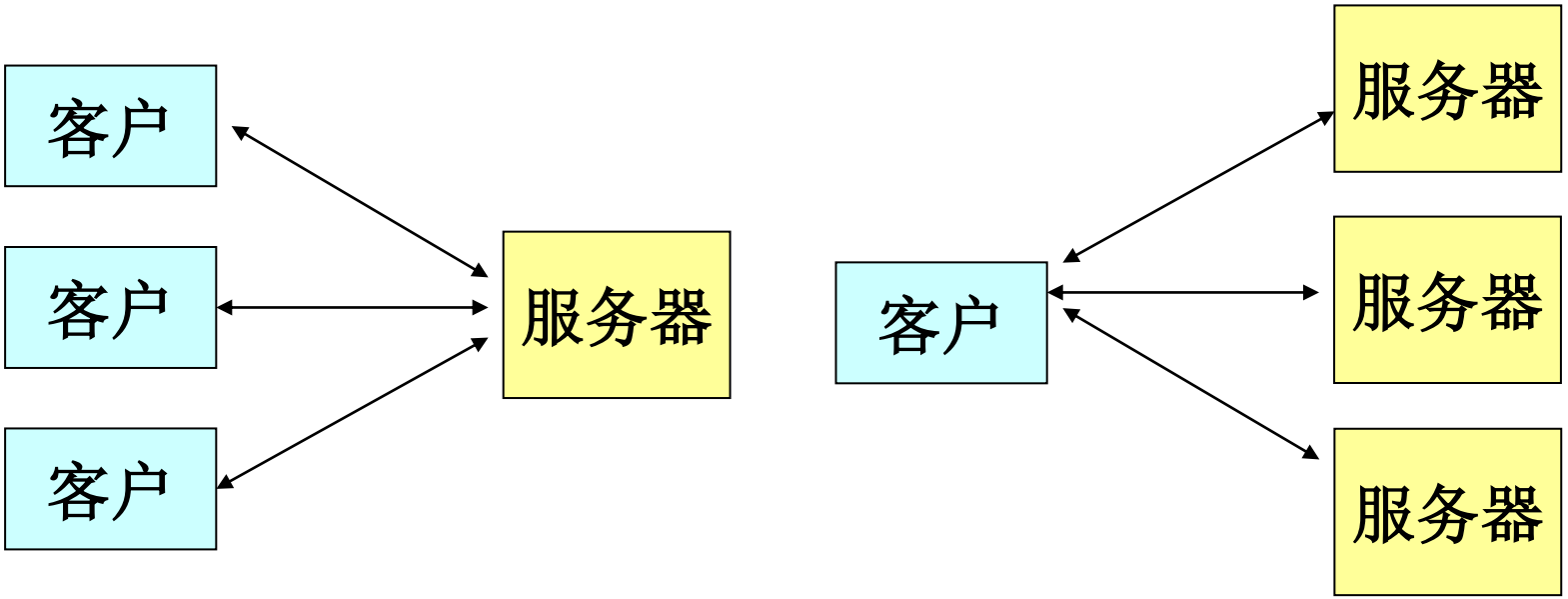
向“whois.nic.gov”递交“whitehouse.gov”的查询请求，查询结果如下：

- *Executive Office of the President (WHITEHOUSE-DOM)*
- *Office of Administration*
- *Washington, DC 20503*
- *Domain Name: WHITEHOUSE.GOV*
- *Status: ACTIVE*
- *Domain Type: Federal*
- *Technical Contact, Administrative Contact:*
- *Reynolds, William D. (WDR)*
- *(202) 395-6975*
- *WILLIAM_D._REYNOLDS@OA.EOP.GOV*
- *Domain servers in listed order:*
- *DNSAUTH1.SYS.GTEI.NET 4.2.49.2*
- *DNSAUTH2.SYS.GTEI.NET 4.2.49.3*
- *DNSAUTH3.SYS.GTEI.NET 4.2.49.4*
- *Record last updated on 28-Aug-00.*

客户/服务器关系

- C/S结构

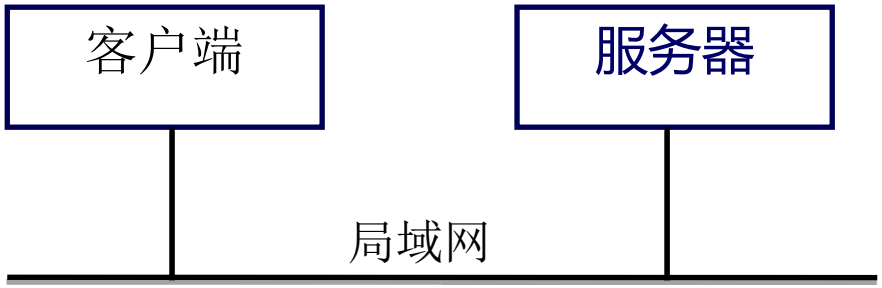
- 客户与服务器间的关系(数量关系)



客户/服务器关系

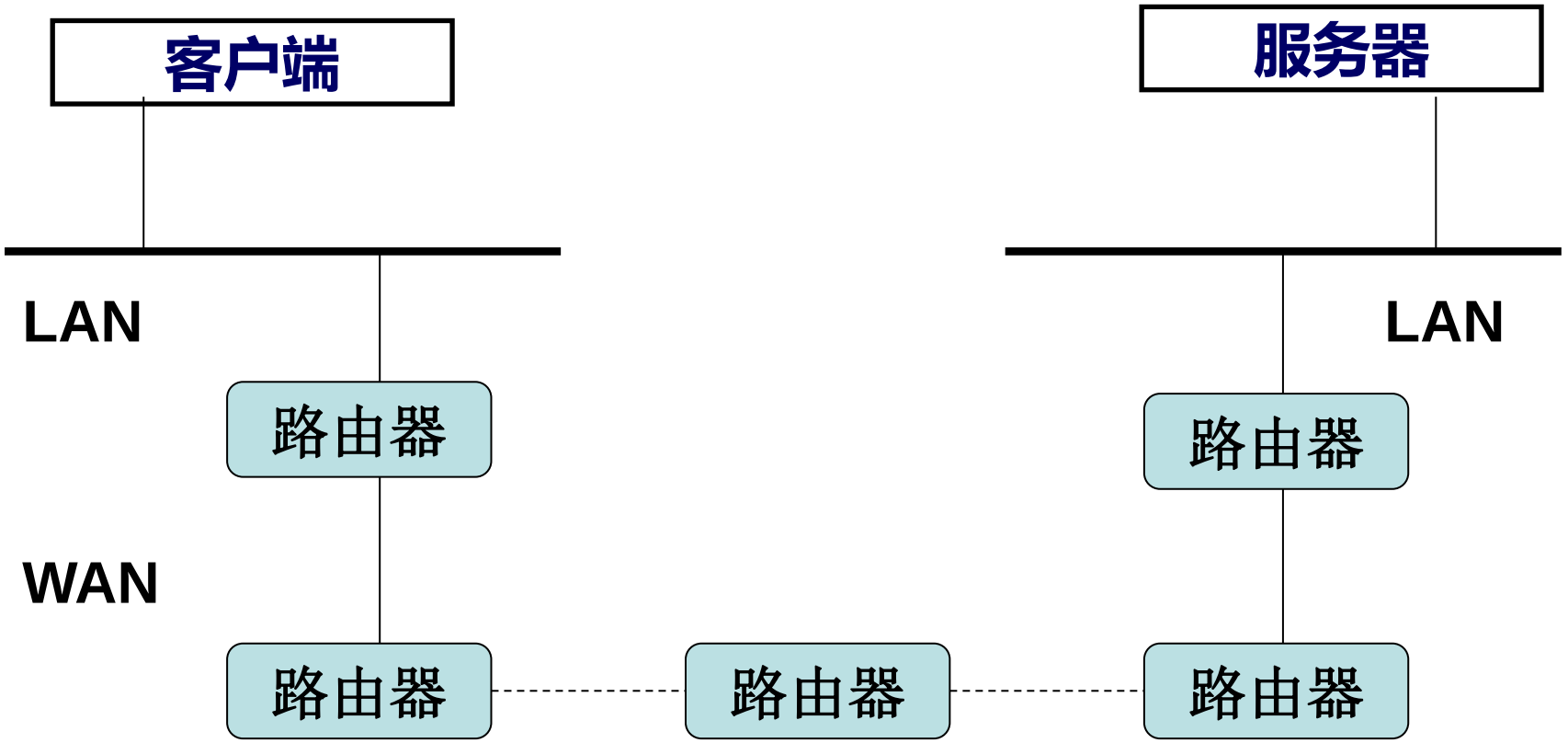
- 客户端与服务器间的位置关系

（A）客户端和服务器在同一主机上



（B）客户端和服务器在同一局域网上

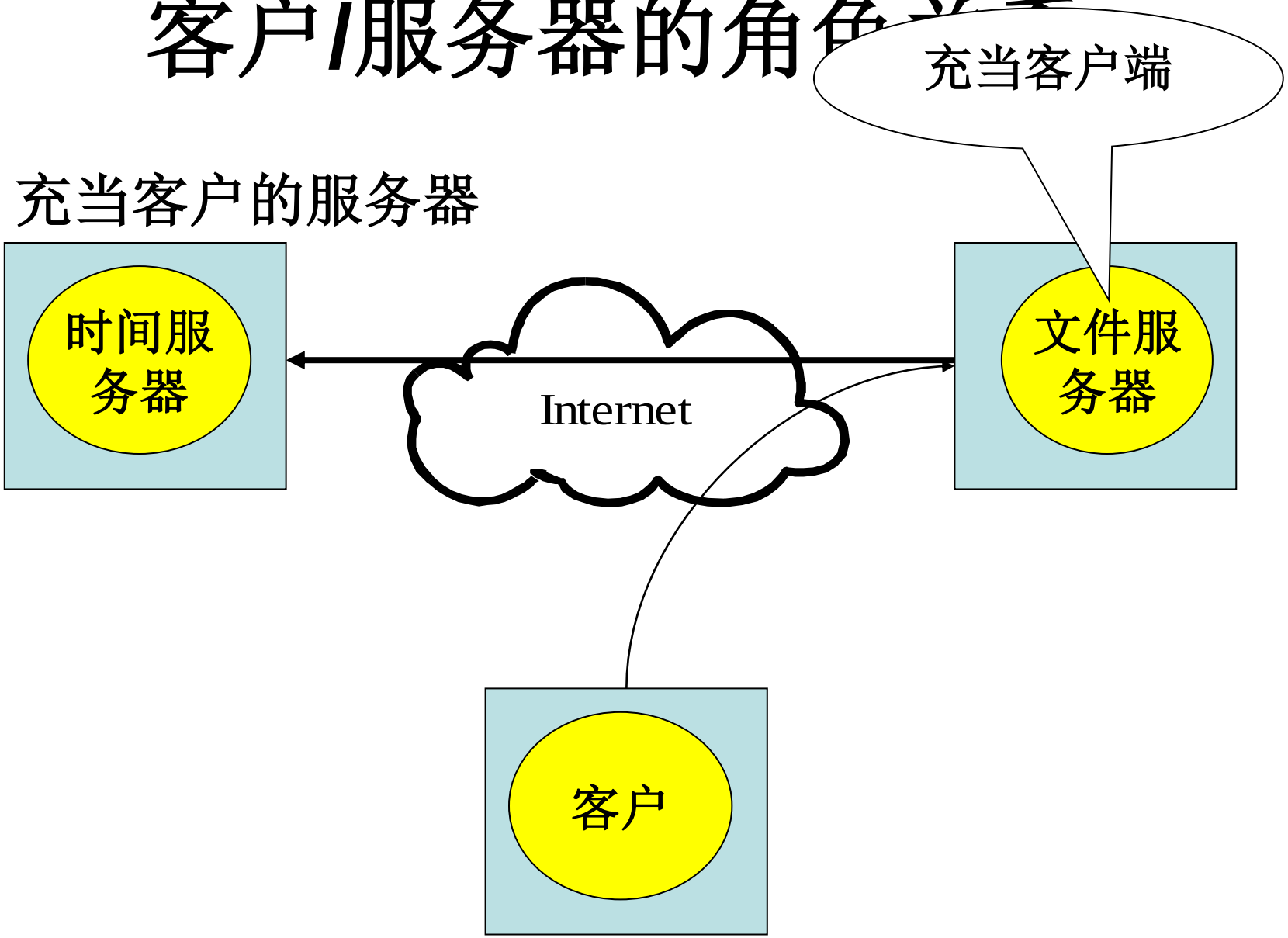
客户/服务器关系



（C）客户端和服务器处于不同的局域网

客户/服务器的角色关系

- 充当客户的服务器



客户端软件

- 客户端软件的参数化
  - 目的：提高客户软件的通用性
  - 举例：telnet远程终端服务
  - 定义：全参数化的客户
  - 建议：在设计客户应用软件时，最好让它包含一些允许用户全部指明机器和目的协议端口号的参数
  - 意义：方便配置和测试

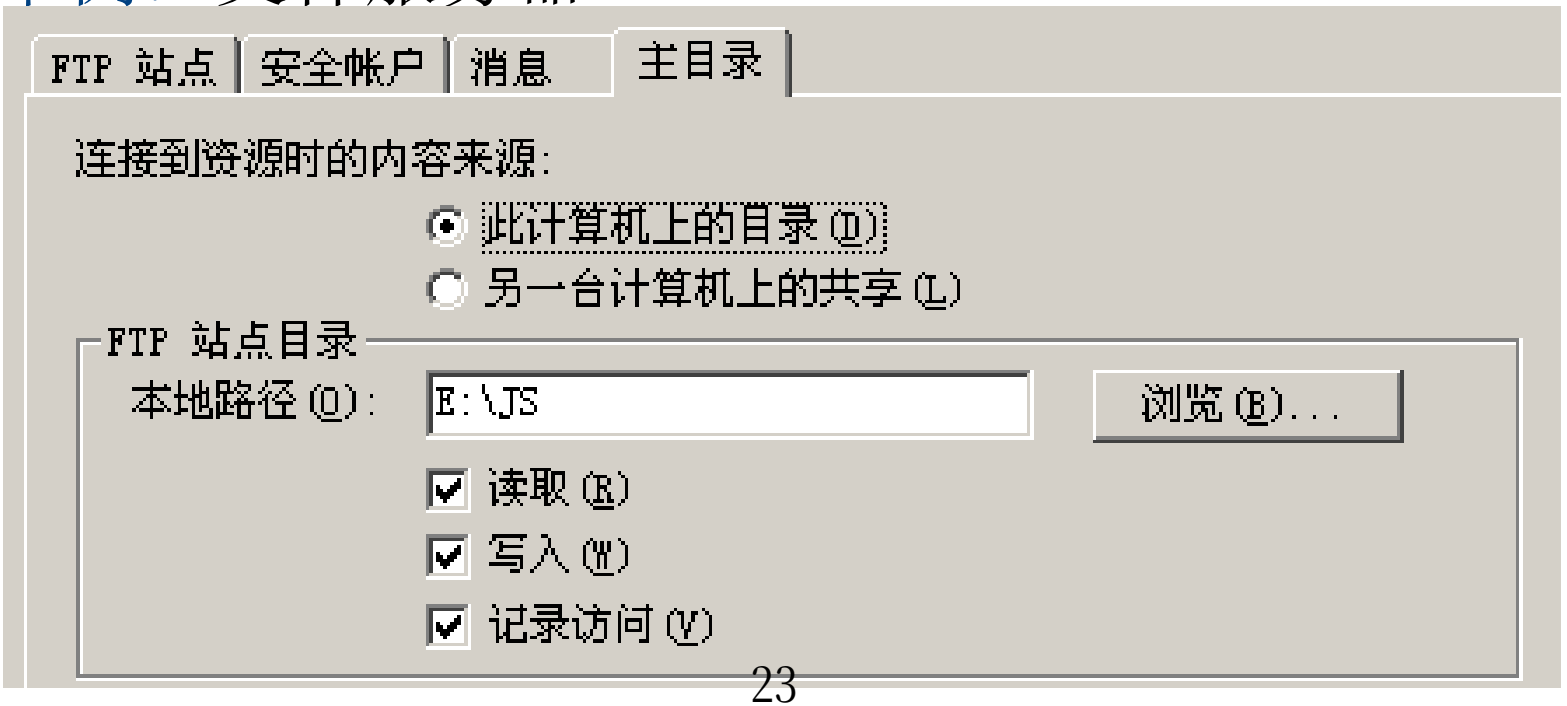
服务器软件的特点与分类

1）服务器的特权和复杂性

特权：访问受操作系统保护的对象（如文件、数据库、设备或协议端口）

考虑：不能将特权传递给使用服务的客户

举例：文件服务器



服务器软件的特点与分类

- 代码复杂性——安全问题：
  - 鉴别——验证客户身份
  - 授权——判断某个客户是否被允许访问服务器所提供的服务
  - 数据安全——确保数据不被无意泄漏或损坏
  - 保密——防止未经授权访问信息
  - 保护——确保网络应用程序不能滥用系统资源
- 代码的复杂性——效率问题
  - 并发处理能力

服务器软件的特点与分类

2）无连接的和面向连接的服务器

在程序员设计客户—服务器软件时，必须在两种类型的交互中做出选择。

①无连接 UDP

②面向连接 TCP

意义：

①决定了客户和服务器交互所采用的算法；

②决定了下层系统所提供的可靠性等级；

服务器软件的特点与分类

- 面向连接的服务器
  - 优点：易于编程
  - 缺点：资源消耗问题
- 无连接的服务器
  - 优点：无资源消耗问题
  - 缺点：编程复杂

问题：如何选择？

服务器软件特点与分类

- 选择思路
  - 考虑
    - 客户和服务器间通信的频度
    - 客户和服务器间交换的数据量
  - UDP
    - 对广播或多播应用程序必须使用UDP；
    - 在可靠的本地环境中运行，不需要额外的可靠性处理；
    - 应用协议指明必须使用UDP；

服务器软件特点与分类

3）无状态和有状态服务器

- 什么是状态信息？
  - 服务器所维护的与客户交互活动的信息。
- 有状态服务器
  - 定义：保存状态信息的服务器
  - 优势：
    - 减少客户和服务器交换报文的大小
    - 允许服务器快速相应请求
- 无状态服务器
  - 定义：不保存任何状态信息的服务器
  - 优势：提高协议的可靠性

服务器软件特点与分类

举例：无状态文件服务器

- 客户发送的请求分为两类：
- 请求从某个指定文件中获取数据
  - 请求在指定文件中存储数据
- 请求报文中应包含以下字段

项目	描述
op	**操作（读或写）**
name	**文件名**
Pos	**在文件中的位置**
Size	**要传输的字节数**
Data	**（只出现在写请求中）**

服务器软件特点与分类

举例：有状态文件服务器

- 客户请求字段：
  - 读请求：读取字节数
  - 写请求：写入字节数和写入数据
- 服务器如何管理状态？

客户	文件名	当前位置	上次操作
1	test.program.c	0	read
2	tcp.book.text	456	read
3	tetris.exe	123	write

服务器软件特点与分类

使用状态服务器的两个要点

- 垃圾信息及时删除：
  - 客户：使用后通知
  - 服务器：释放资源
- 效率：考虑底层传输现状

服务器软件特点与分类
状态信息的维护是一个复杂的问题

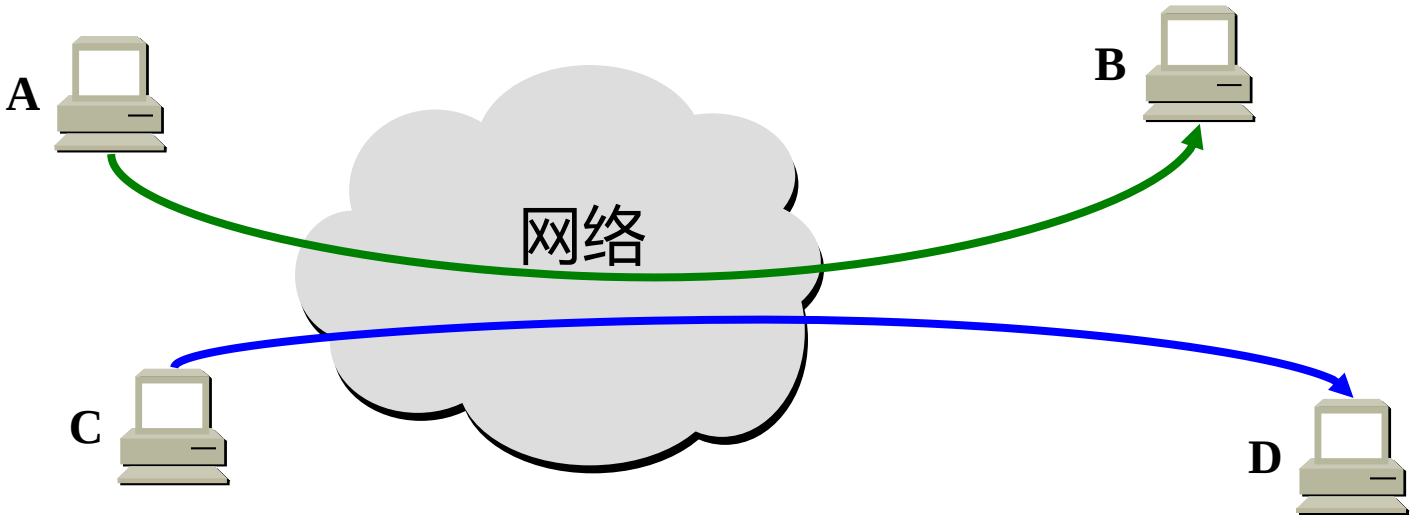
- 状态的维护困难
  - 如：报文重复、延时或交付失序的问题
- 状态的更新复杂
  - 如：客户机重启使得状态不正确
- 状态对客户的标识会混乱
  - 如：使用端点标识的客户重启使得状态混乱

服务器软件特点与分类

- 一个服务器到底是无状态还是有状态，更多取决于应用协议而不是实现。

循环服务器与并发服务器

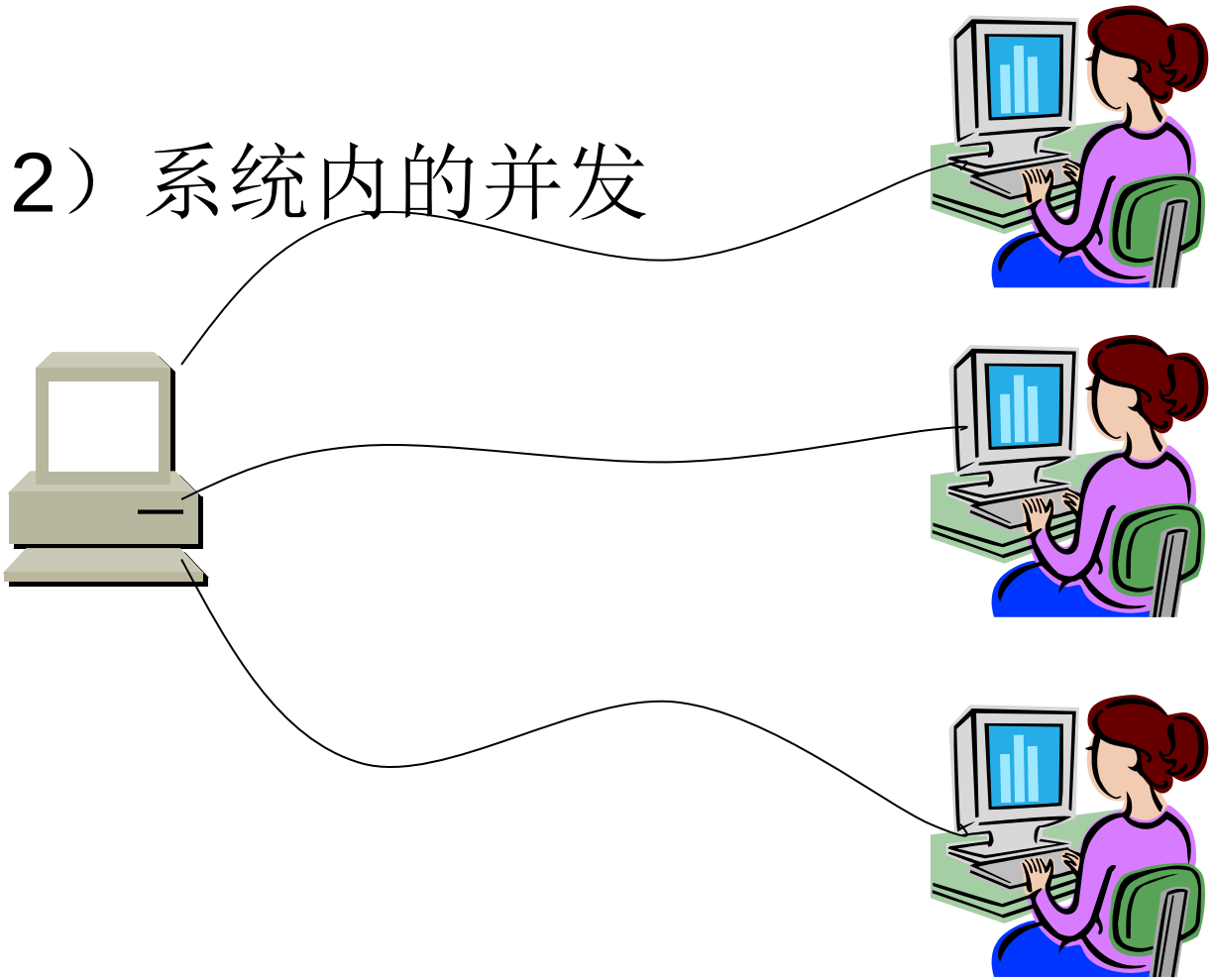
- （1）网络中的并发



从单个机器来看好像各个应用进程都在独占网络资源

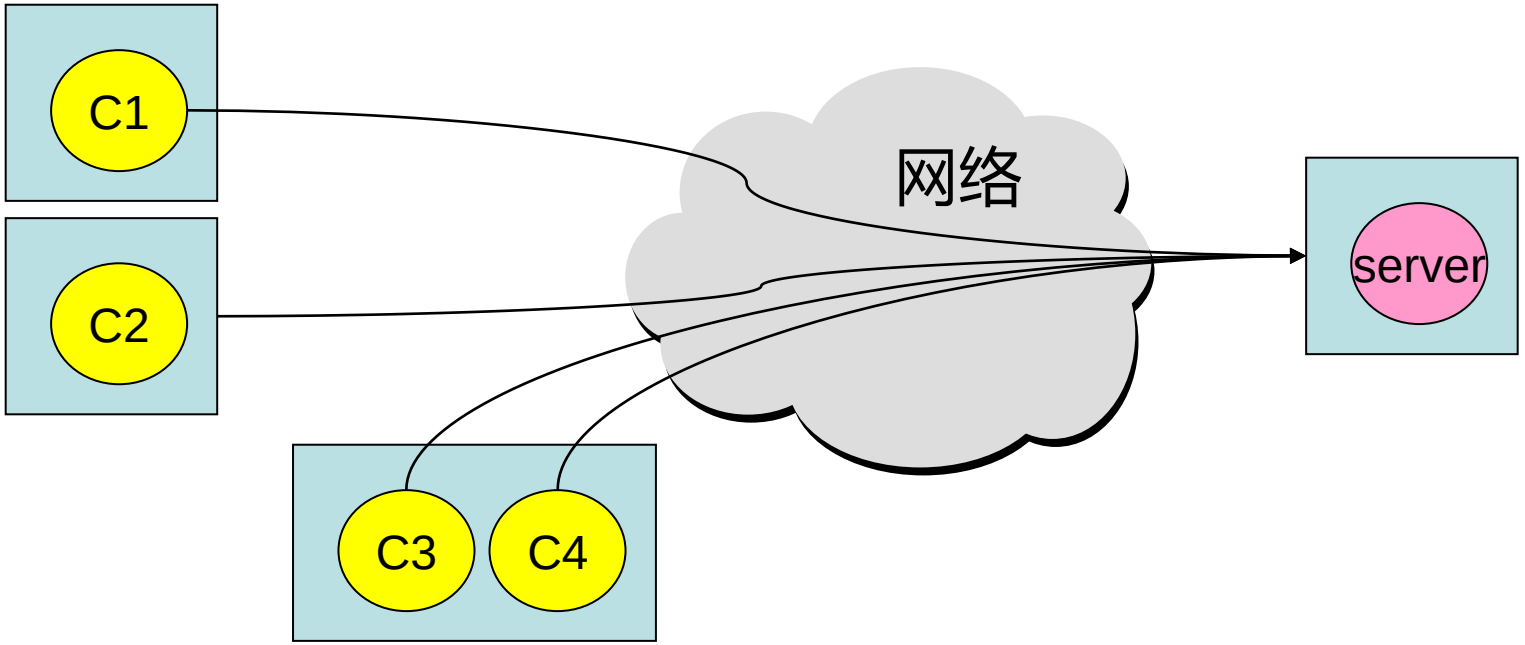
网络硬件执行一些访问规则，这些规则允许每对通信的机器之间相互交换报文，并且能防止某一对应用程序占用了全部的网络带宽而排斥其他应用进程的通信。

- （2）系统内的并发



从用户观点看，好像所有的客户程序都在同时运行

- （3）服务器中的并发



多个客户使用服务器的一个熟知协议端口与服务器联系，服务器软件必须在编程中处理好并发请求。

单个客户程序就像普通的程序那样运行，不明显管理并发。

- 基本概念
  - 并发（concurrency）：真正的或表面呈现的同时计算
  - 分时（time sharing）：使单个处理器在多个计算任务之间足够快地切换，从表面看这些计算是同时进行的。
  - 多处理（multiprocessing）：让多个处理器同时执行多个任务。

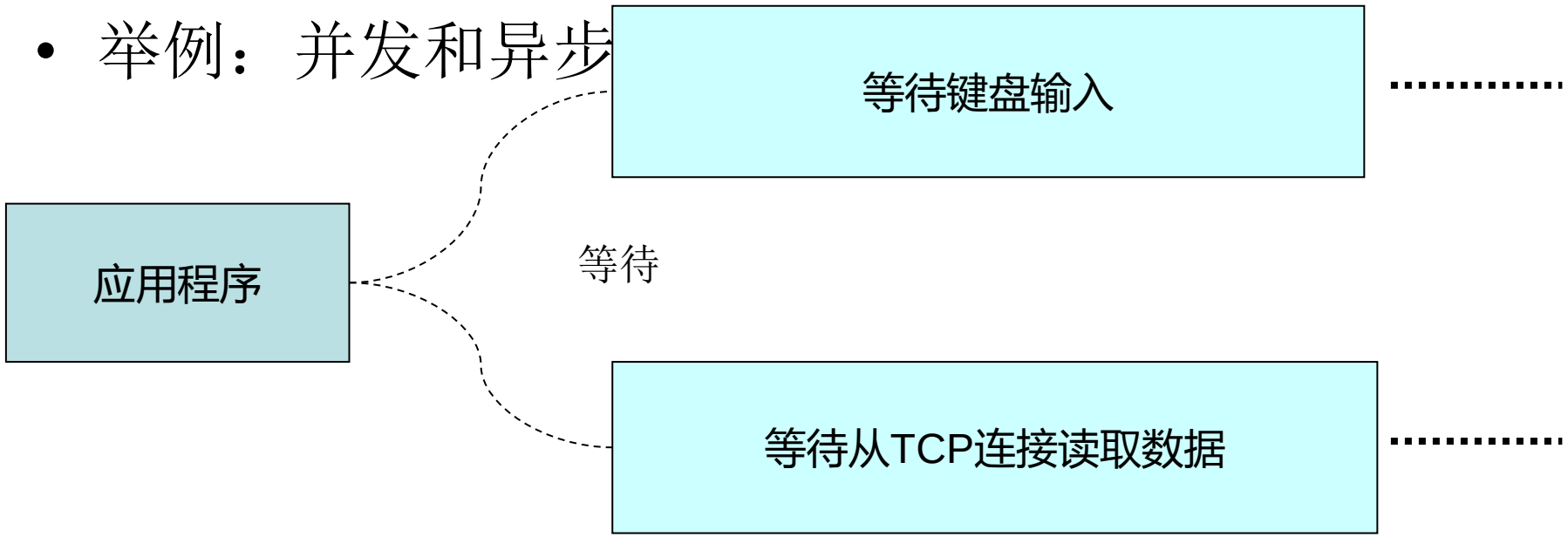
- 基本概念（续）
  - 时间分片（timeslicing）：描述多个并发线程共享可用CPU的系统时间的系统。
    - 目的：试图在所有线程间平均分配可用的处理器资源。
    - 方法：
      - 进程间平均
      - 优先级

并发服务器的设计代价

- 上下文切换（context switch）
  - 当操作系统暂时停止执行某个线程而切换到另一个线程时，会发生上下文切换
  - 代价：使用CPU，在CPU忙于切换时，任何应用线程都不能得到任何服务。
  - 协议软件设计：减少上下文切换次数

引入并发处理的好处 > 上下文切换的开销

- 举例：并发和异步



问题：程序不知道输入的数据是先从键盘来还是先从TCP连接来！

……

单线程的并发设计

多线程的并发设计

串行设计

1. B/S模型

- B/S（Browse/Server）模型也叫B/S模式，它是一种基于Web的通信模型，使用HTTP（Hypertext Transfer Protocol，超文本传送协议）通信。B/S是一种特殊的C/S模型，特殊之处就在于这种模型的客户端一般是某种流行的浏览器，例如，微软的Internet Explorer（也叫IE浏览器）等

- 优点:单台计算机可以访问任何一个Web服务器，不需要针对不同的服务器分别提供专用的客户端软件。

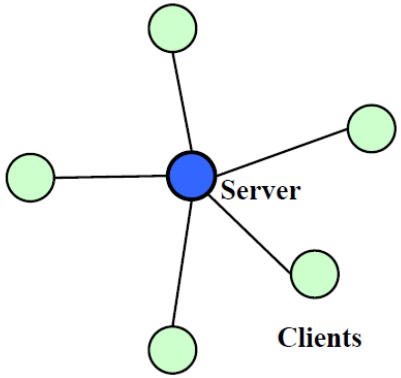
2. P2P模型

- **Server集中式的服务方式具有的弊端：**
  - **资源无法得到充分利用**
  - **全球互联的局限**
- **P2P（Peer-to-Peer，对等互联）是近年来比较流行的通信模型之一。在Peer-to-Peer环境中，每个联网的计算机同时运行一个应用程序的Client部分和Sever部分。**

  一个应用程序既起Server的作用，又起Client的作用。

- P2P是什么？
  - 一种思想
  - 一类应用

2.1 一种思想



- 计算机网络 → 因特网 Internet
- 网络的基础结构：

1、集中式：C/S = Client/Server

-- 好：管理简单，控制有效

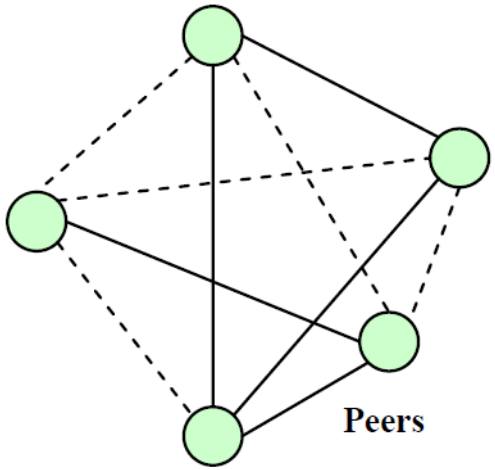
-- 坏：Server瓶颈

2、分布式：Distributed

-- 好：无瓶颈，资源充分利用

-- 坏：管理松散，难于控制

自由
平等
互联



→ P2P = 分布式的极端 （since 1956年）

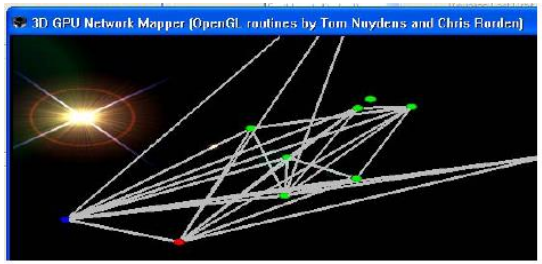
2.2 一类应用

- 文件共享

- 媒体播放

- 数据存储

- 分布计算等



2.3 P2P评价

- 优点:

  配置容易，通信方便，成本低；

- 缺点:

  知识产权的侵害
  网络病毒传播
  管理复杂
  垃圾信息