

第 6 章 应用层

第 6 章 应用层

- 6.1 域名系统 DNS
- 6.2 文件传送协议
- 6.3 远程终端协议 TELNET
- 6.4 万维网 WWW
- 6.5 电子邮件
- 6.6 动态主机配置协议 DHCP
- 6.7 简单网络管理协议 SNMP
- 6.8 应用进程跨越网络的通信

应用层协议的特点

- 应用层的具体内容就是**规定应用进程在通信时所遵循的协议**。
- 应用层的许多协议都是基于客户服务器方式。客户(**client**)和服务器(**server**)都是指通信中所涉及的两个应用进程。
- **客户服务器方式**所描述的是进程之间服务和被服务的关系。客户是服务请求方，服务器是服务提供方。

6.1 域名系统 DNS

- 6.1.1 域名系统概述
- 6.1.2 因特网的域名结构
- 6.1.3 域名服务器

6.1.1 域名系统概述

人类: 有很多标识:

- 身份证号, 姓名, 护照等

因特网主机、路由器:

- IP在址(32 bit)
- “名字”, 例如, 人们常用的
- `ww.yahoo.com`

Q: IP地址和“名字”之间
是怎么映射的?

域名系统:

- 由分层的**DNS**服务器实现的**分布式数据库**
- 一个允许主机查询分布式数据库的**应用层协议**
 - 注: 因特网的核心功能, 作为应用层协议运行
 - 将复杂的处理推向网路边缘

6.1.2 因特网的域名结构

- 因特网采用了层次树状结构的命名方法。
- 任何一个连接在因特网上的主机或路由器，都有一个**唯一**的层次结构的**名字**，即**域名**。
- 域名的结构由标号序列组成，各标号之间用**点**隔开：

... . **三级域名** . **二级域名** . **顶级域名**

- 各标号分别代表不同级别的域名。

域名只是个逻辑概念

- 域名只是个逻辑概念，并不代表计算机所在的物理地点。
- 变长的域名和使用有助记忆的字符串，是为了便于人来使用。而 **IP** 地址是定长的 **32** 位二进制数字则非常便于机器进行处理。

顶级域名 TLD

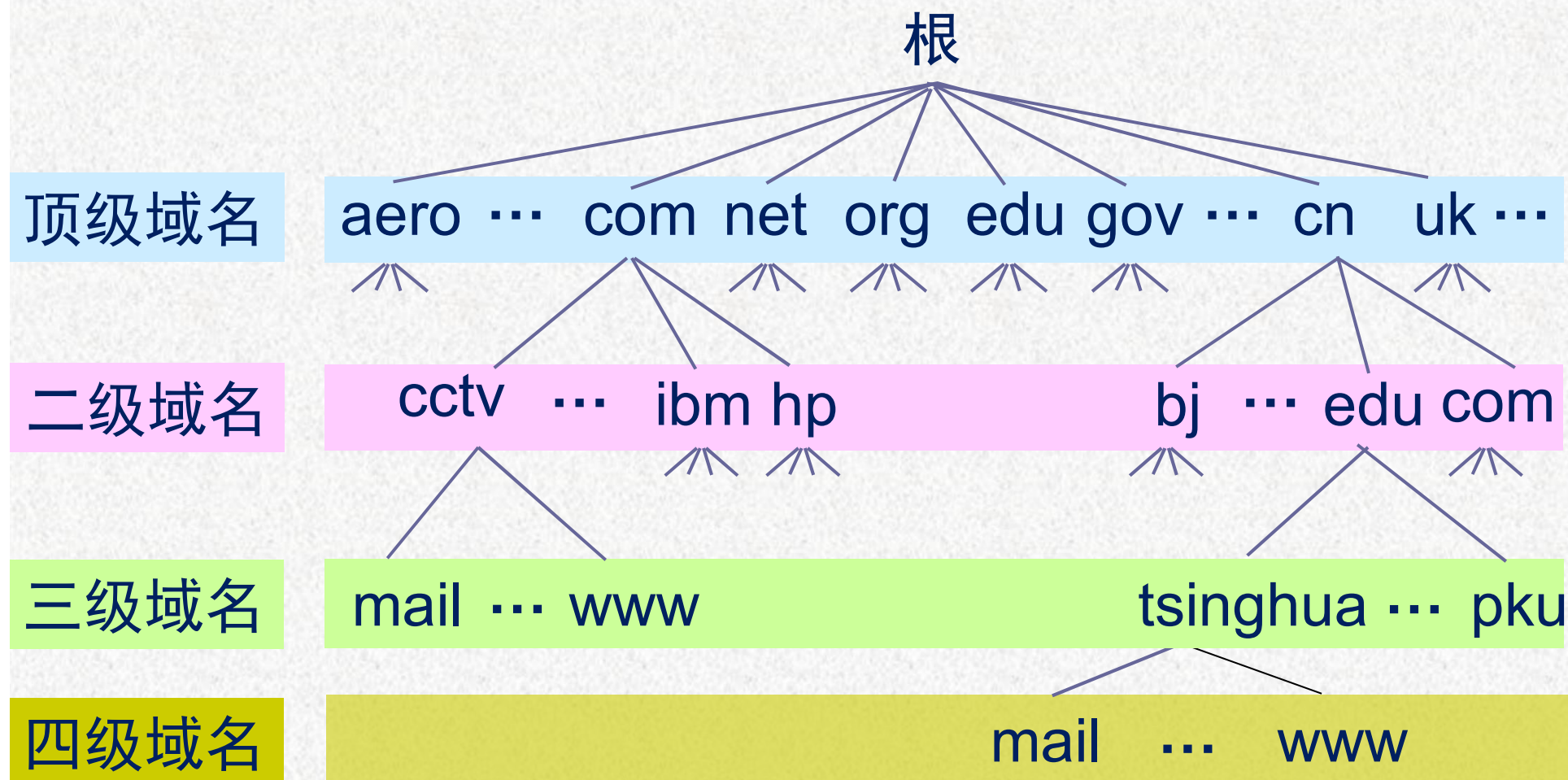
(Top Level Domain)

- (1) 国家顶级域名 nTLD: 如: **.cn** 表示中国, **.us** 表示美国, **.uk** 表示英国, 等等。
- (2) 通用顶级域名 gTLD: 最早的顶级域名是:
- .com** (公司和企业)
 - .net** (网络服务机构)
 - .org** (非赢利性组织)
 - .edu** (美国专用的教育机构 ())
 - .gov** (美国专用的政府部门)
 - .mil** (美国专用的军事部门)
 - .int** (国际组织)

新增加了下列的通用顶级域名

- **.aero** （航空运输企业）
- **.biz** （公司和企业）
- **.cat** （加泰隆人的语言和文化团体）
- **.coop** （合作团体）
- **.info** （各种情况）
- **.jobs** （人力资源管理者）
- **.mobi** （移动产品与服务的用户和提供者）
- **.museum** （博物馆）
- **.name** （个人）
- **.pro** （有证书的专业人员）
- **.travel** （旅游业）

因特网的域名空间



6.1.3 域名服务器

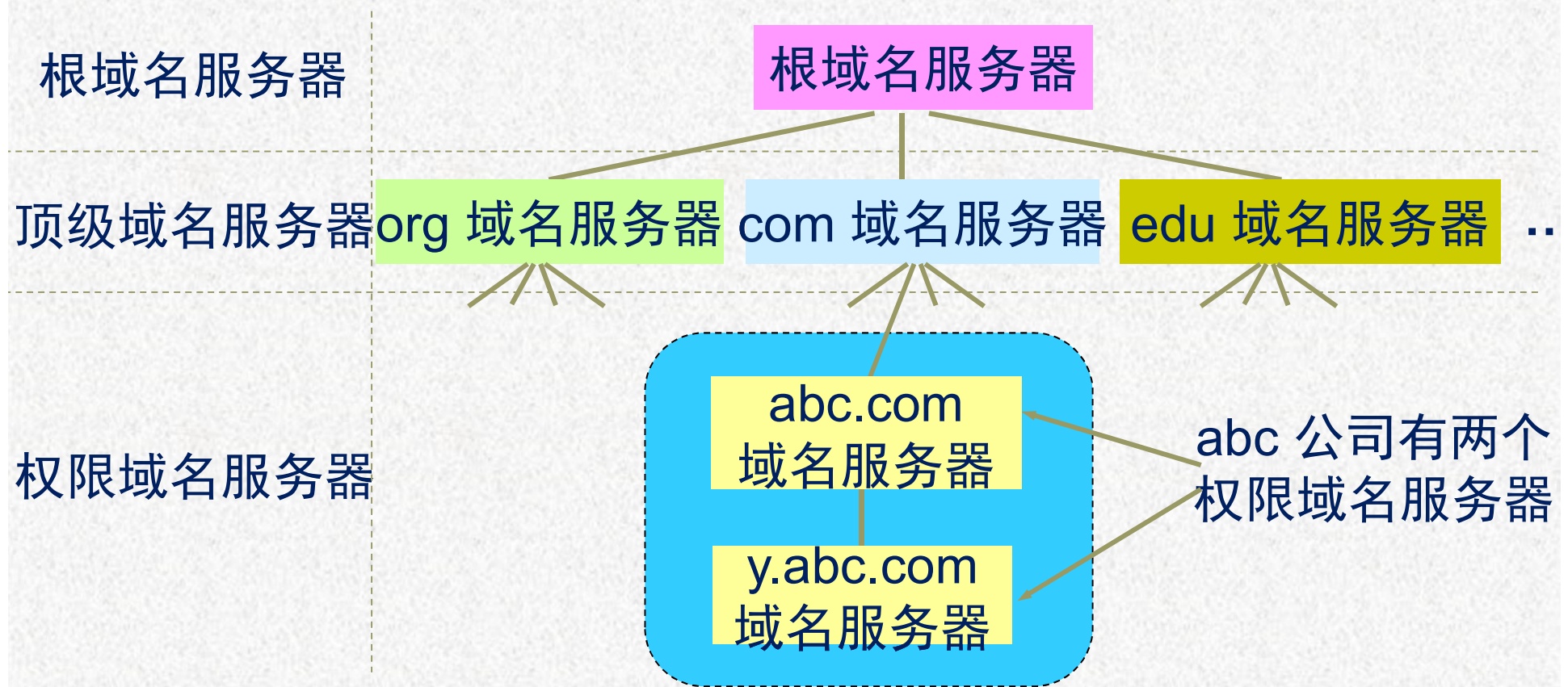
域名数据库与DNS协议

- **分布式DNS服务器**：层次结构的**分布式数据库**
- **DNS协议**：允许主机查询分布式数据库的**应用层协议**
- 将复杂的处理推向网路边缘

DNS为什么不采用集中结构？

- 存在单点失效问题；
- 存在性能可扩展问题，大量的请求会淹没服务器；
- 集中结构使得服务器与客户机的距离远，延迟大；
- 单个数据库容量会很大，频繁的更新操作带来管理上的麻烦。

树状结构的 DNS 域名服务器



域名服务器有以下四种类型

- 根域名服务器
- 顶级域名服务器
- 权限域名服务器
- 本地域名服务器

根域名服务器

——最高层次的域名服务器——

- 根域名服务器是最重要的域名服务器。所有的根域名服务器都知道所有的顶级域名服务器的域名和 IP 地址。
- 不管是哪一个本地域名服务器，若要对因特网上任何一个域名进行解析，只要自己无法解析，就首先求助于根域名服务器。
- 在因特网上共有13 个不同 IP 地址的根域名服务器，它们的名字是用一个英文字母命名，从a 一直到 m（前13 个字母）。

根域名服务器共有 13 套装置 (不是 13 个机器)

- 根域名服务器相应的域名分别是
a.rootservers.net
b.rootservers.net
...
m.rootservers.net
- 到 2006 年底全世界已经安装了一百多个根域名服务器机器，分布在世界各地。
- 这样做的目的是为了更方便用户，使世界上大部分 DNS 域名服务器都能就近找到一个根域名服务器。

举例：根域名服务器 f 的地点分布图



- 根域名服务器并不直接把域名直接转换成IP地址。
- 在使用迭代查询时，根域名服务器把下一步应当找的顶级域名服务器的 IP 地址告诉本地域名服务器。

顶级域名服务器

（即 **TLD** 服务器）

- 域名服务器负责管理在该顶级域名服务器注册的所有二级域名。
- 当收到 **DNS** 查询请求时，就给出相应的回答（可能是最后的结果，也可能是下一步应当找的域名服务器的 **IP** 地址）。

权限域名服务器

- 负责一个区的域名服务器。
- 当一个权限域名服务器还不能给出最后的查询回答时，就会告诉发出查询请求的 **DNS** 客户，下一步应当找哪一个权限域名服务器。

本地域名服务器

- 本地域名服务器对域名系统非常重要。
- 当一个主机发出 **DNS** 查询请求时，这个查询请求报文就发送给本地域名服务器。
- 每一个因特网服务提供者 **ISP**，或一个大学，甚至一个大学里的系，都可以拥有一个本地域名服务器，
- 这种域名服务器有时也称为**默认域名服务器**。

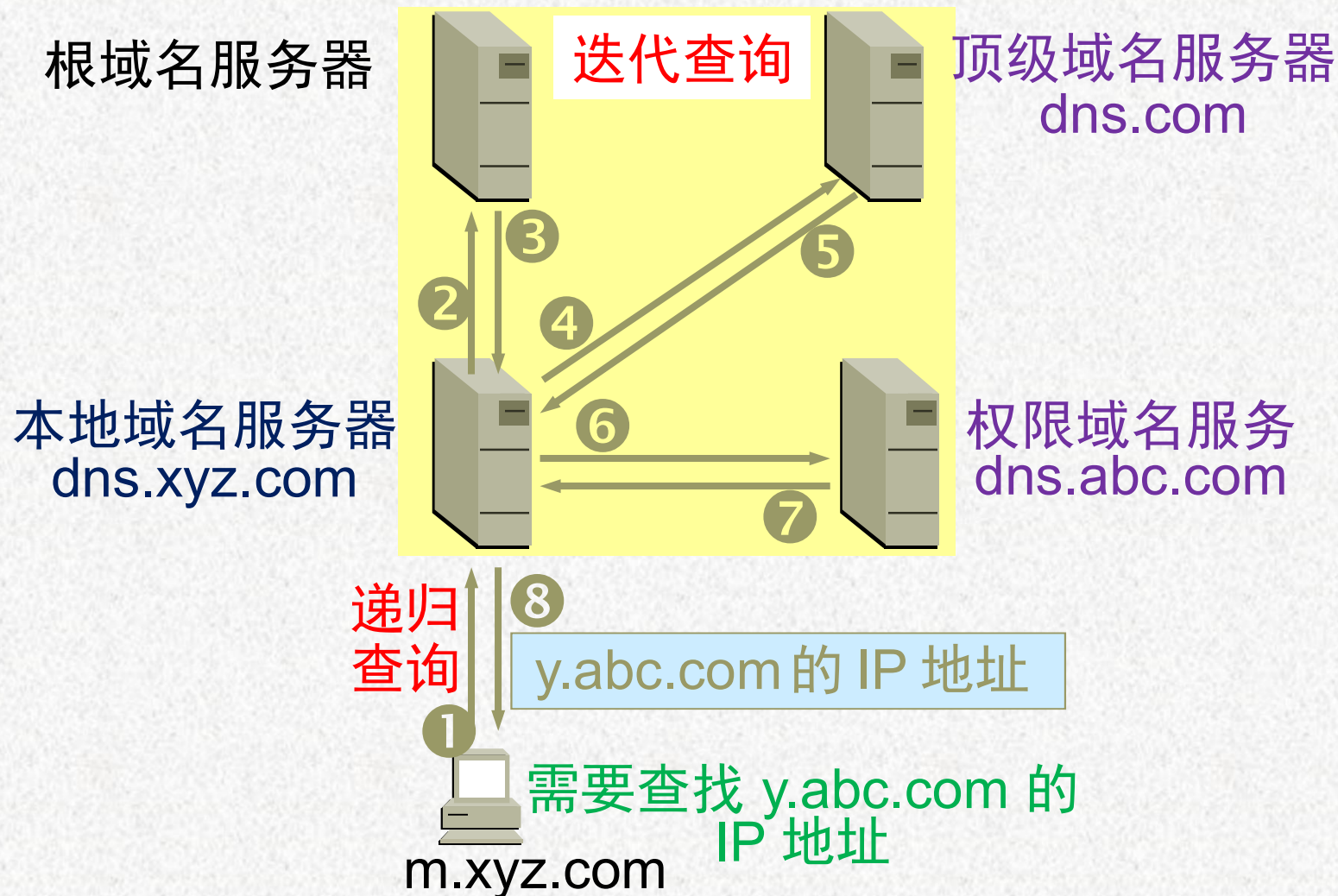
提高域名服务器的可靠性

- **DNS** 域名服务器都把数据复制到几个域名服务器来保存，其中的一个是**主域名服务器**，其他的是**辅助域名服务器**。
- 当主域名服务器出故障时，辅助域名服务器可以保证 **DNS** 的查询工作不会中断。
- 主域名服务器定期把数据复制到辅助域名服务器中，而更改数据只能在主域名服务器中进行。这样就保证了数据的一致性。

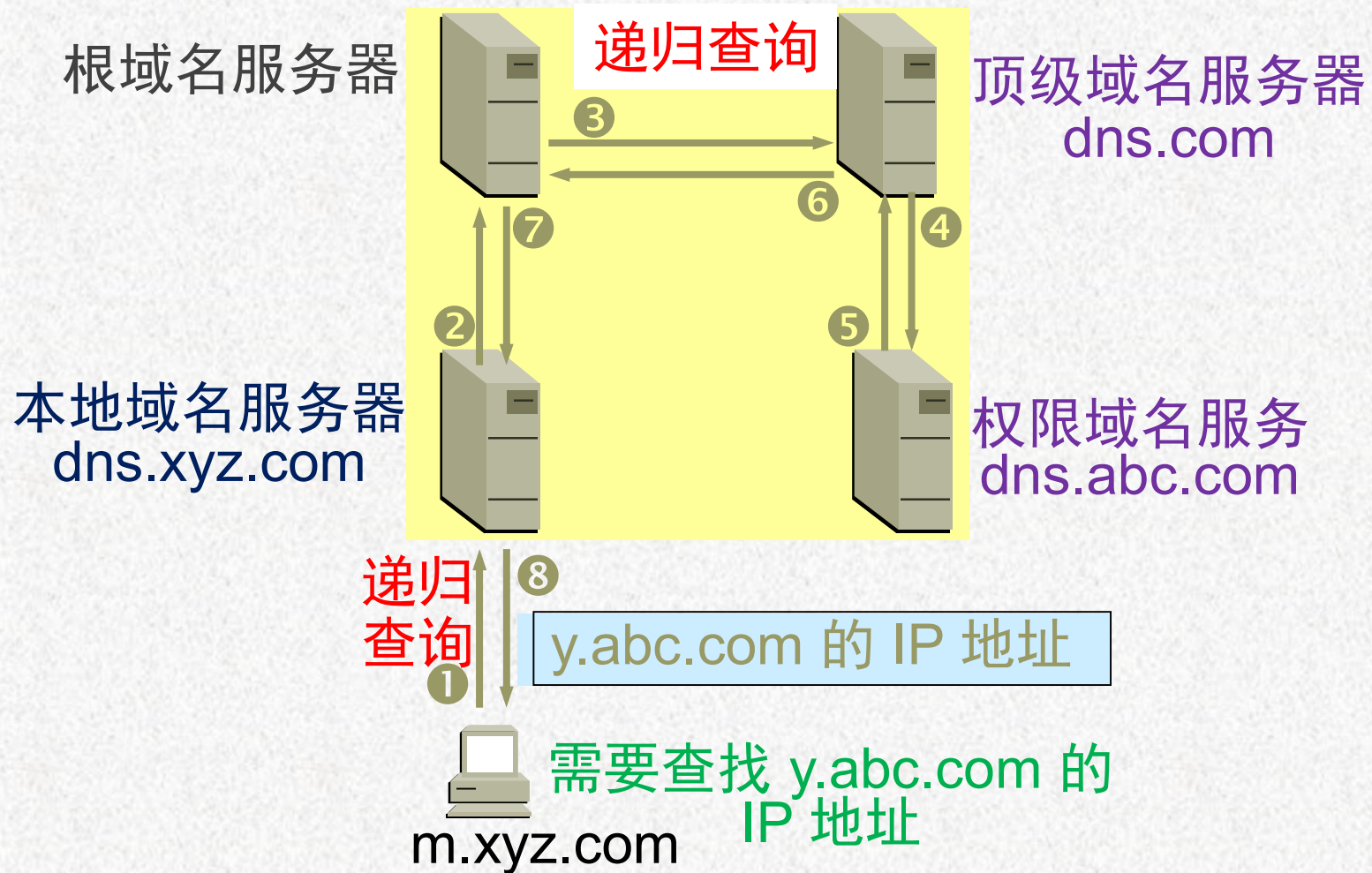
域名的解析过程

- 主机向本地域名服务器的查询一般都是采用**递归查询**。如果主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其他根域名服务器继续发出查询请求报文。
- 本地域名服务器向根域名服务器的查询通常是采用**迭代查询**。当根域名服务器收到本地域名服务器的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地域名服务器进行后续的查询。

本地域名服务器采用迭代查询



本地域名服务器采用递归查询 (比较少用)



名字的高速缓存

- 每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从何处获得名字映射信息的记录。
- 可大大减轻根域名服务器的负荷，使因特网上的 **DNS** 查询请求和回答报文的数量大为减少。
- 为保持高速缓存中的内容正确，域名服务器应为每项内容设置**计时器**，并处理超过**合理时间**的项（例如，每个项目只存放两天）。
- 当权限域名服务器回答一个查询请求时，在响应中都指明绑定有效存在的时间值。增加此时间值可减少网络开销，而减少此时间值可提高域名转换的准确性。

DNS: 名字的高速缓存和记录更新

- 一旦DNS服务器获得一个映射, 它就将其**缓存**起来
 - 缓存记录会过期
 - 顶级DNS服务器的访问记录存放在本地DNS服务器
 - 因此根服务器不会被经常访问
- IETF设计的更新机制
 - 域名系统动态更新 (DNS 更新) RFC 2136

DNS数据库记录

DNS: 资源记录(RR)存放在分布式数据库中

RR 格式: (name, value, type, ttl)

■ Type=A

- name 是主机名
- value 是IP地址

■ Type=NS

- name 是域 (如 foo.com)
- value 该域的权威DNS服务器的主机名

■ Type=CNAME

- name 是主机别名
- value是别名为name的主机对应的规范主机名

■ Type=MX

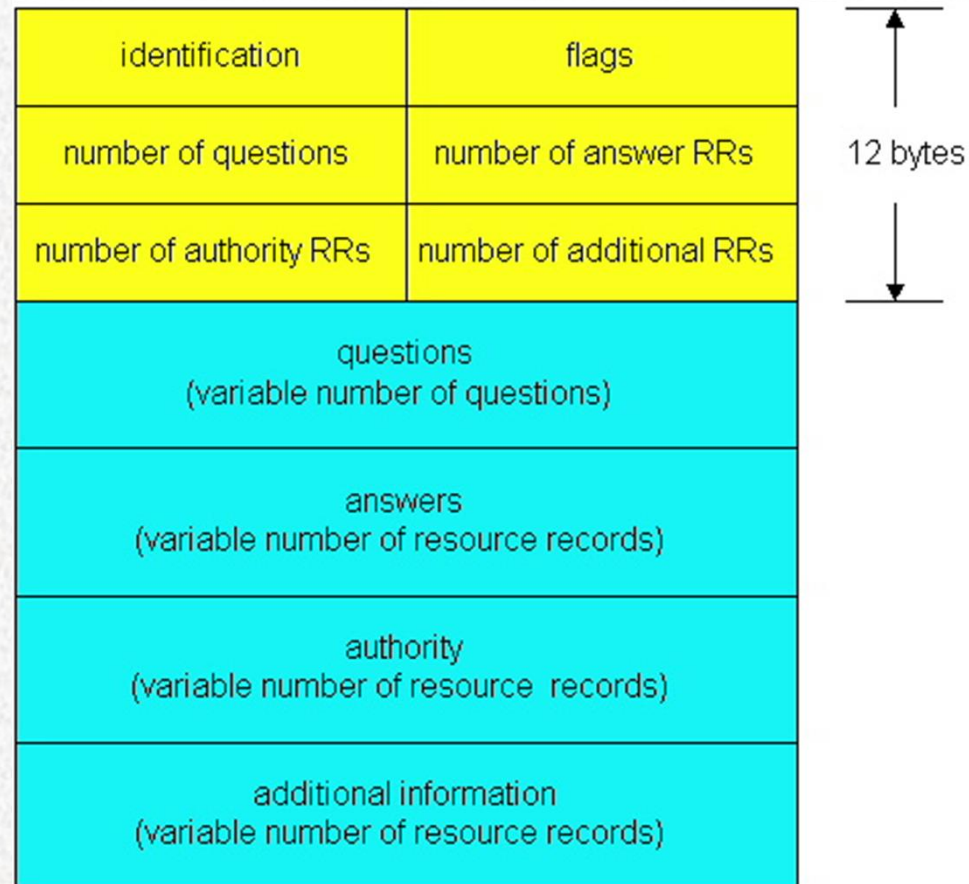
- value 是别名为name的邮件服务器的规范主机名

DNS协议和DNS报文

DNS 协议：查询报文 和 回答报文 具有相同的格式

首部区域

- **标识符**: 16 bit # 这个标识符会被复制到相应的回答报文中 #
- **标志**:
 - “查询/回答” 标志位
 - “希望递归” 标志位
 - “递归可用” 标志位
 - “权威” 标志位



DNS协议DNS协议的消息格式

- query和reply，两者消息的格式相同。
- 头部（12字节）：
 - **identification**：16位。DNS查询由Client发起，Client确定identification, server返回结果时也以相同的identification标识 reply。
 - **flags**：16位。包括：
 - 1-bit的query(0)/reply(1)标志
 - 1-bit的授权DNS服务器,1—是
 - 1-bit的递归查询标志，1表示Host希望利用recursive方法查询
 - 1-bit的可以递归查询的标志(reply消息中设置)
 - **Question部分**：包括一个name域，其值为待查询的域名；type域指示希望查询的类型；
 - **Reply消息中的Answer部分**：查询结果，RR, 可以有多个结果
 - **authority部分**：包含其他授权DNS服务器的信息。

6.2 文件传送协议

6.2.1 FTP 概述

6.2.2 FTP 的基本工作原理

6.2.3 简单文件传送协议 TFTP

6.2.1 FTP概述

- **文件传送协议 FTP (File Transfer Protocol)** 是因特网上使用得最广泛的文件传送协议。
- **FTP** 提供交互式的访问，允许客户指明文件的类型与格式，并允许文件具有存取权限。
- **FTP** 屏蔽了各计算机系统的细节，因而适合于在异构网络中任意计算机之间传送文件。
- **RFC 959** 很早就成为了因特网的正式标准。

文件传送并非很简单的问题

- 网络环境中的一项基本应用就是将文件从一台计算机中复制到另一台可能相距很远的计算机中。
- 初看起来，在两个主机之间传送文件是很简单的事情。
- 其实这往往非常困难。原因是众多的计算机厂商研制出的文件系统多达数百种，且差别很大。

6.2.2 FTP 的基本工作原理

网络环境下复制文件的复杂性：

- (1) 计算机存储数据的格式不同。
- (2) 文件的目录结构和文件命名的规定不同。
- (3) 对于相同的文件存取功能，操作系统使用的命令不同。
- (4) 访问控制方法不同。

FTP 特点

- 文件传送协议 **FTP** 只提供文件传送的一些基本的服务，它使用 **TCP** 可靠的运输服务。
- **FTP** 的主要功能是减少或消除在不同操作系统下处理文件的不兼容性。
- **FTP** 使用**客户服务器方式**。一个 **FTP** 服务器进程可同时为多个客户进程提供服务。
FTP 的服务器进程由两大部分组成：一个**主进程**，负责接受新的请求；另外有若干个**从属进程**，负责处理单个请求。

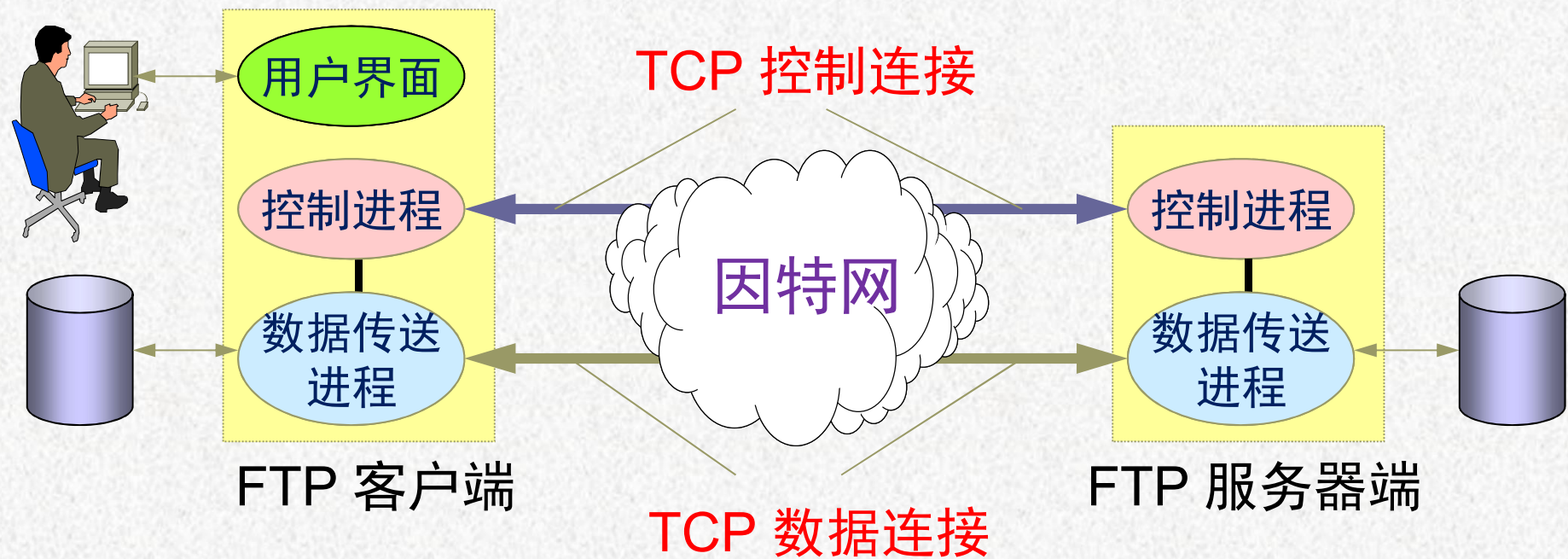
主进程的工作步骤如下

- 打开熟知端口（端口号为 **21**），使客户进程能够连接上。
- 等待客户进程发出连接请求。
- 启动从属进程来处理客户进程发来的请求。从属进程对客户进程的请求处理完毕后即终止，但从属进程在运行期间根据需要还可能创建其他一些子进程。
- 回到等待状态，继续接受其他客户进程发来的请求。主进程与从属进程的处理是并发地进行。

两个连接

- **控制连接**：在整个会话期间一直保持打开，FTP 客户发出的传送请求通过控制连接发送给服务器端的控制进程，但控制连接不用来传送文件。
- **数据连接**：实际用于传输文件的是“**数据连接**”。服务器端的控制进程在接收到 FTP 客户发送来的文件传输请求后就创建“数据传送进程”和“数据连接”，用来连接客户端和服务器的数据传送进程。
- 数据传送进程实际完成文件的传送，在传送完毕后关闭“数据传送连接”并结束运行。

FTP 使用的两个 TCP 连接



两个不同的端口号

- 当客户进程向服务器进程发出建立连接请求时，要寻找连接服务器进程的**熟知端口(21)**，同时还要告诉服务器进程自己的另一个端口号码，用于建立数据传送连接。
- 接着，服务器进程用自己传送数据的**熟知端口(20)**与客户进程所提供的端口号码建立数据传送连接。
- 由于 **FTP** 使用了两个不同的端口号，所以数据连接与控制连接不会发生混乱。

使用两个不同端口号的好处

- 使协议更加简单和更容易实现。
- 在传输文件时还可以利用控制连接（例如，客户发送请求终止传输）。

NFS 采用另一种思路

- **NFS** 允许应用进程打开一个远地文件，并能在该文件的某一个特定的位置上开始读写数据。
- **NFS** 可使用户只复制一个大文件中的一个很小的片段，而不需要复制整个大文件。
- 对于上述例子，计算机 **A** 的 **NFS** 客户软件，把要添加的数据和在文件后面写数据的请求一起发送到远地的计算机 **B** 的 **NFS** 服务器。**NFS** 服务器更新文件后返回应答信息。
- 在网络上传送的只是少量的修改数据。

6.3 远程终端协议 TELNET

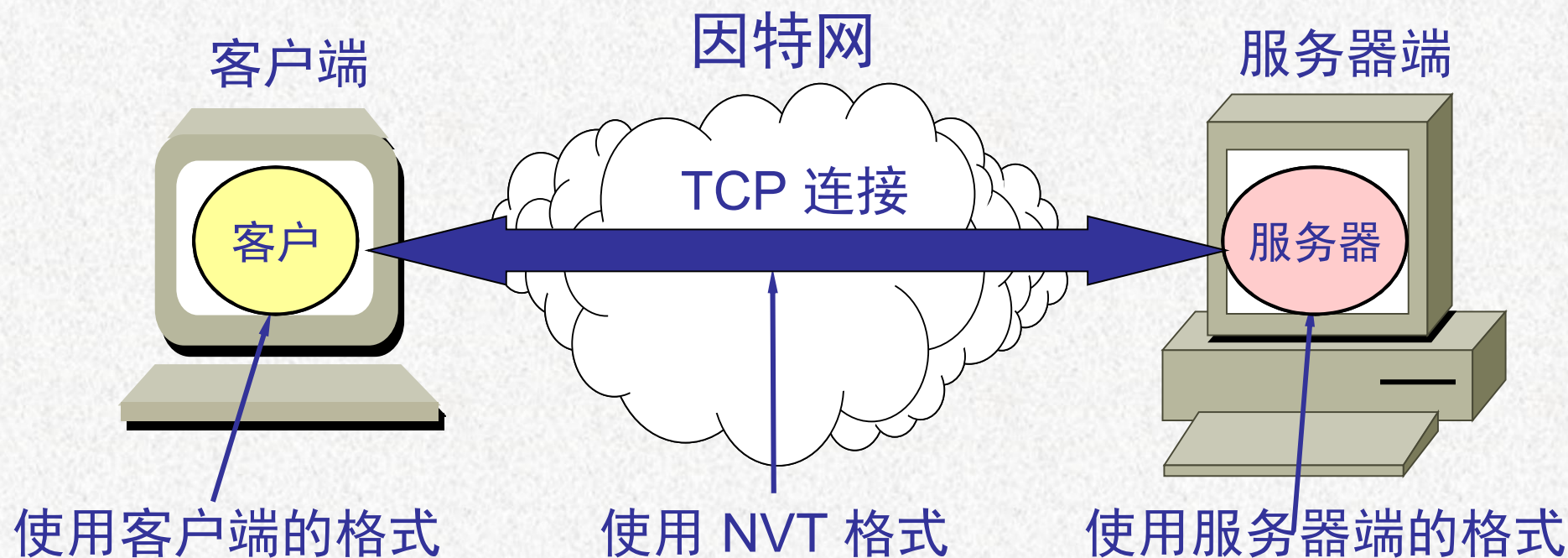
6.3 远程终端协议 TELNET

- **TELNET** 是一个简单的远程终端协议，也是因特网的正式标准。
- 用户用 **TELNET** 就可在其所在地通过 **TCP** 连接注册（即登录）到远地的另一个主机上（使用主机名或 **IP** 地址）。
- **TELNET** 能将用户的**击键**传到远地主机，同时也将远地主机的输出通过 **TCP** 连接返回到用户屏幕。这种服务是透明的，因为用户感觉到好像键盘和显示器是直接连在远地主机上。

客户服务器方式

- 现在由于 PC 机的功能越来越强，用户已较少使用 TELNET 了。
- TELNET 也使用客户服务器方式。在本地系统运行 TELNET 客户进程，而在远地主机则运行 TELNET 服务器进程。
- 和 FTP 的情况相似，服务器中的主进程等待新的请求，并产生从属进程来处理每一个连接。

TELNET 使用 网络虚拟终端 NVT 格式



网络虚拟终端 NVT 格式

- 客户软件把用户的击键和命令转换成 **NVT** 格式，并送交服务器。
- 服务器软件把收到的数据和命令，从 **NVT** 格式转换成远地系统所需的格式。
- 向用户返回数据时，服务器把远地系统的格式转换为 **NVT** 格式，本地客户再从 **NVT** 格式转换到本地系统所需的格式。

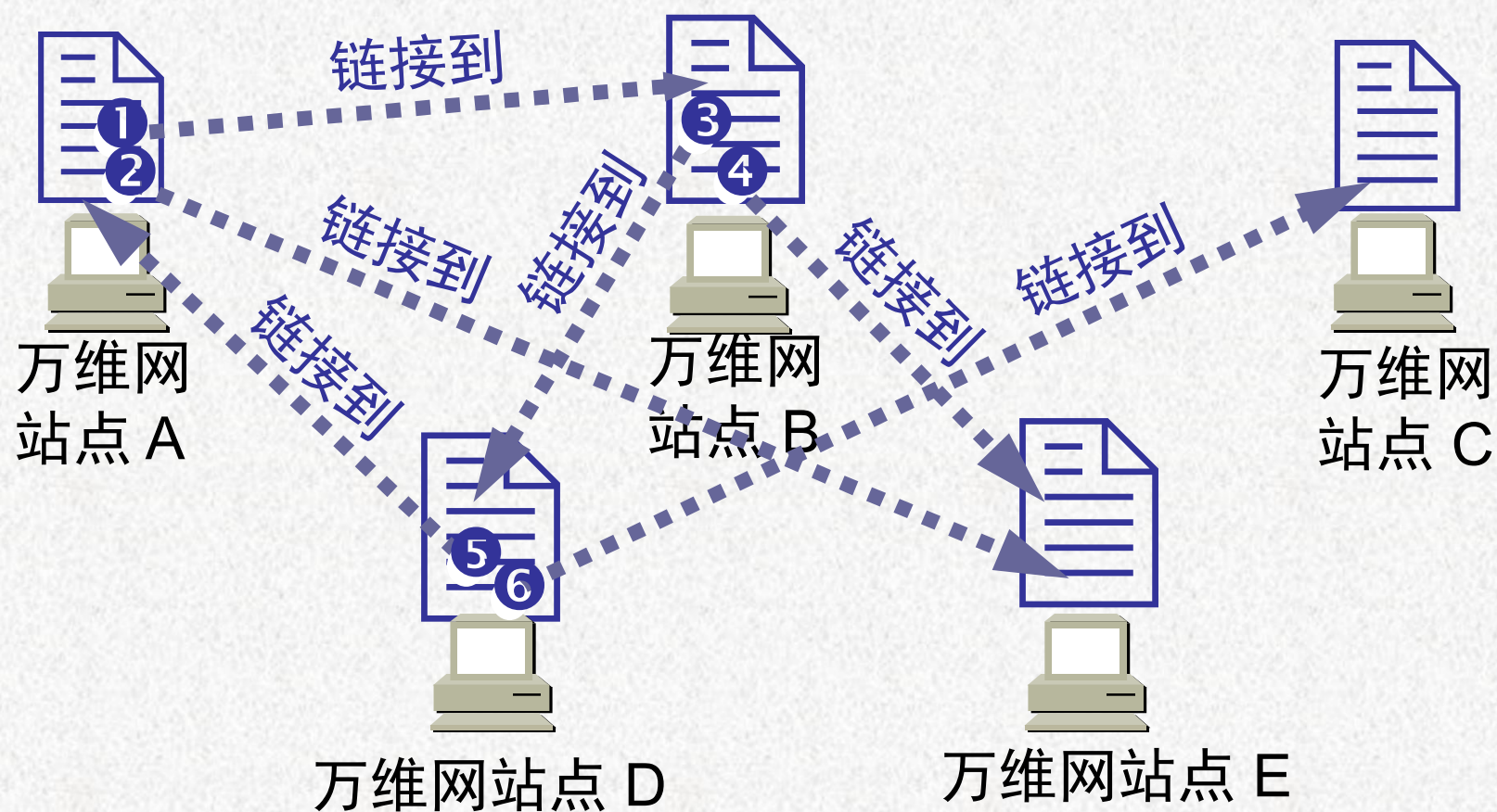
6.4 万维网 WWW

- 6.4.1 概述
- 6.4.2 统一资源定位符 URL
- 6.4.3 超文本传送协议 HTTP
- 6.4.4 万维网的文档
- 6.4.5 万维网的信息检索系统

6.4.1 万维网概述

- **万维网** WWW (World Wide Web)并非某种特殊的计算机网络。
- 万维网是一个大规模的、联机式的信息储藏所。
- 万维网用链接的方法能非常方便地从因特网上的一个站点访问另一个站点，从而主动地按需获取丰富的信息。
- 这种访问方式称为“**链接**”。

万维网提供分布式服务



超媒体与超文本

- 万维网是**分布式超媒体(hypermedia)**系统，它是**超文本(hypertext)**系统的扩充。
- 一个超文本由多个信息源链接成。利用一个链接可使用户找到另一个文档。这些文档可以位于世界上任何一个接在因特网上的超文本系统中。超文本是万维网的基础。
- 超媒体与超文本的区别是文档内容不同。超文本文档仅包含文本信息，而超媒体文档还包含其他表示方式的信息，如图形、图像、声音、动画，甚至活动视频图像。

万维网的工作方式

- 万维网以客户服务器方式工作。
- **浏览器**就是在用户计算机上的万维网**客户程序**。万维网文档所驻留的计算机则运行**服务器程序**，因此这个计算机也称为**万维网服务器**。
- 客户程序向服务器程序发出请求，服务器程序向客户程序送回客户所要的万维网文档。
- 在一个客户程序主窗口上显示出的万维网文档称为**页面**(page)。

万维网必须解决的问题

(1) 怎样标志分布在整个因特网上的万维网文档？

- 使用**统一资源定位符 URL (Uniform Resource Locator)**来标志万维网上的各种文档。
- 使每一个文档在整个因特网的范围内具有唯一的标识符 **URL**。

万维网必须解决的问题

(2) 用何协议实现万维网上各种超链的链接？

- 在万维网客户程序与万维网服务器程序之间进行交互所使用的协议，是**超文本传送协议 HTTP (HyperText Transfer Protocol)**。
- **HTTP** 是一个应用层协议，它使用 **TCP** 连接进行可靠的传送。

万维网必须解决的问题

(3) 怎样使各种万维网文档都能在因特网上的各种计算机上显示出来，同时使用户清楚地知道在什么地方存在着超链？

■ **超文本标记语言 HTML (HyperText Markup Language)** 使得万维网页面的设计者可以很方便地用一个超链从本页面的某处链接到因特网上的任何一个万维网页面，并且能够在自己的计算机屏幕上将这些页面显示出来。

万维网必须解决的问题

(4) 怎样使用户能够很方便地找到所需的信息？

- **为了在万维网上方便地查找信息，用户可使用各种的搜索工具（即搜索引擎）。**

6.4.2 统一资源定位符 URL

1. URL的格式

- 统一资源定位符 URL 是对可以从因特网上得到的资源的位置和访问方法的一种简洁的表示。
- URL 给资源的**位置**提供一种抽象的识别方法，并用这种方法给资源定位。
- 只要能够对资源定位，系统就可以对资源进行各种操作，如存取、更新、替换和查找其属性。
- URL 相当于一个文件名在网络范围的扩展。因此 URL 是与因特网相连的机器上的任何可访问对象的一个指针。

URL 的一般形式

- 由以冒号隔开的两大部分组成，并且在 URL 中的字符对大写或小写没有要求。
- URL 的一般形式是：

<协议>://<主机>:<端口>/<路径>

ftp —— 文件传送协议 FTP

http —— 超文本传送协议 HTTP

News —— USENET 新闻

URL 的一般形式（续）

- 由以冒号隔开的两大部分组成，并且在 URL 中的字符对大写或小写没有要求。
- **URL** 的一般形式是：

<协议>://<主机>:<端口>/<路径>

<主机> 是存放资源的主机
在因特网中的域名

URL 的一般形式（续）

- 由以冒号隔开的两大部分组成，并且在 URL 中的字符对大写或小写没有要求。
- URL 的一般形式是：

<协议>://<主机>:<端口>/<路径>

有时可省略

使用 HTTP 的 URL

- 使用 HTTP 的 URL 的一般形式

http://<主机>:<端口>/<路径>

↑
这表示使用 HTTP 协议

使用 HTTP 的 URL

■ 使用 HTTP 的 URL 的一般形式

http://<主机>:<端口>/<路径>

冒号和两个斜线是规定的格式

使用 HTTP 的 URL

- 使用 HTTP 的 URL 的一般形式

http://<主机>:<端口>/<路径>

这里写主机的域名

使用 HTTP 的 URL

- 使用 HTTP 的 URL 的一般形式

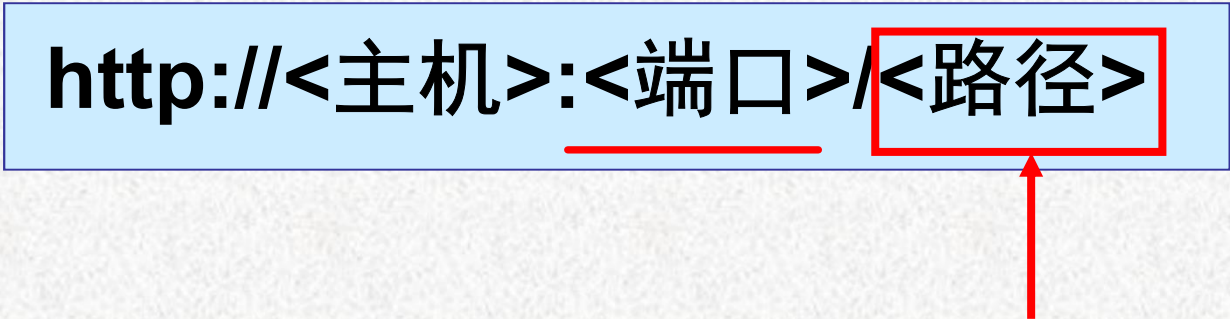
http://<主机>:<端口>/<路径>

↑
HTTP 的默认端口号是 80，通常可省略

使用 HTTP 的 URL

■ 使用 HTTP 的 URL 的一般形式

http://<主机>:<端口>/<路径>

A diagram showing the general form of an HTTP URL: http://<主机>:<端口>/<路径>. The entire URL is enclosed in a light blue rectangular box. The '<端口>' part is underlined with a red line. The '<路径>' part is enclosed in a red rectangular box. A red arrow points upwards from the text below to the red box around '<路径>'.

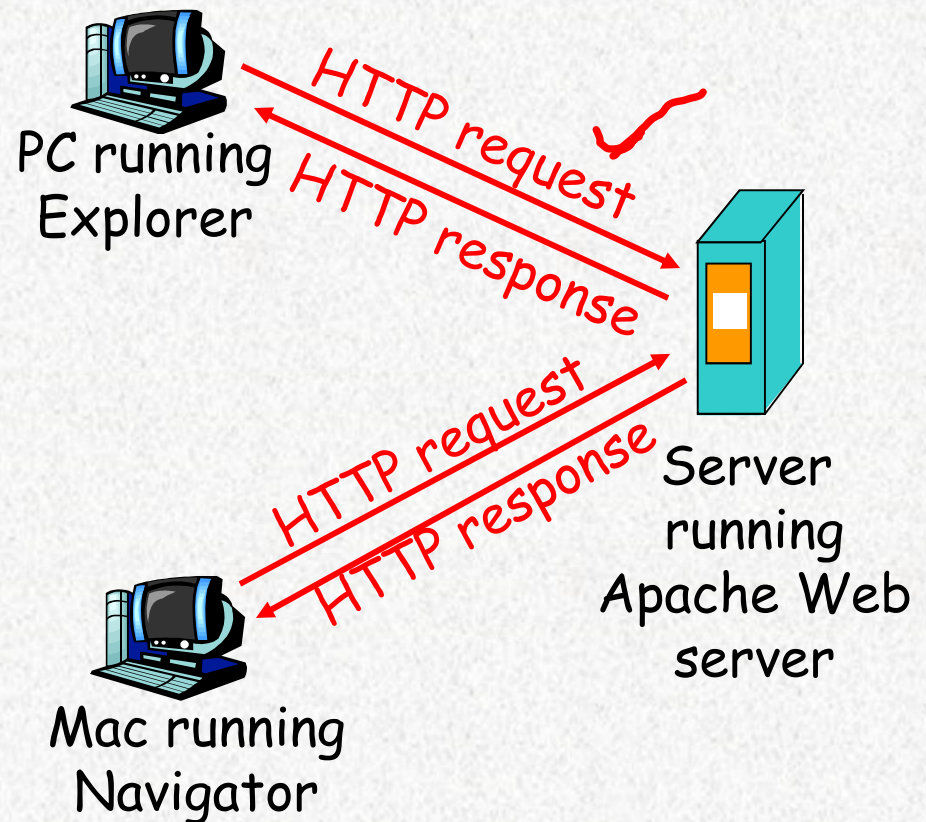
若再省略文件的<路径>项，则 URL 就指到因特网上的某个主页(home page)。

6.4.3 超文本传送协议 HTTP

HTTP 协议概况

HTTP: 超文本传输协议

- Web的应用层协议
- 客户机/服务器结构
 - **客户机**: 发送请求并接收、显示 Web对象的浏览器
 - **服务器**: 响应请求并发送对象的Web服务器
- HTTP 1.0: RFC 1945
- **HTTP 1.1: RFC 2616**



HTTP 协议概况 (续)

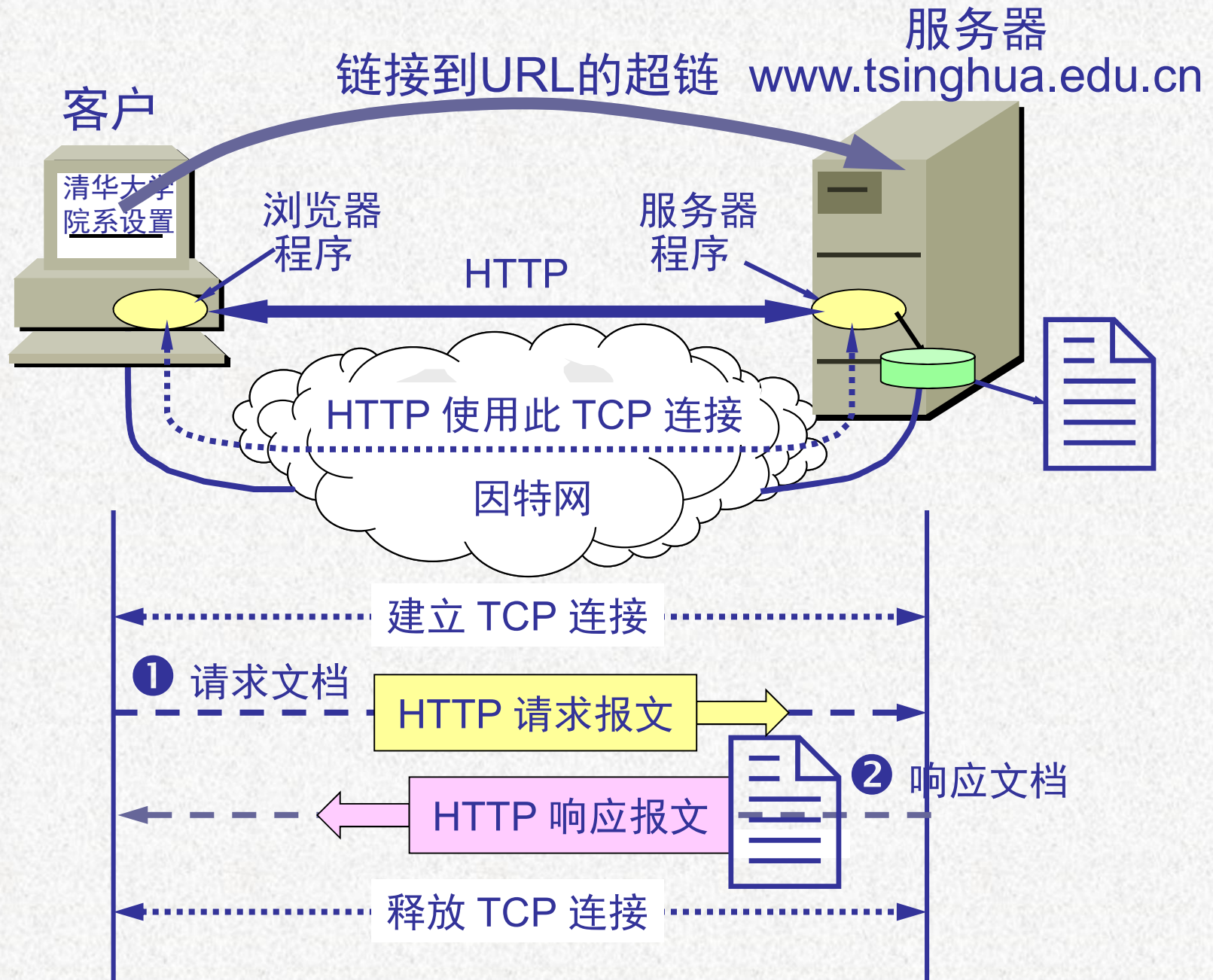
HTTP使用TCP:

- 客户机向服务器80端口发起连接请求
- 服务器同意客户机的连接请求
- 在浏览器和Web服务器之间进行 HTTP 报文交换
- 关闭TCP连接

HTTP 是无状态协议

- server不保存任何client的任何状态信息。
- 如果server在很短的时间内从browser接收到对某个object的两次请求，server就会发送两次response。

万维网的工作过程



用户点击鼠标后所发生的事件

- (1) 浏览器分析超链指向页面的 URL。
- (2) 浏览器向 DNS 请求解析 `www.tsinghua.edu.cn` 的 IP 地址。
- (3) 域名系统 DNS 解析出清华大学服务器的 IP 地址。
- (4) 浏览器与服务器建立 TCP 连接
- (5) 浏览器发出取文件命令：
 `GET /chn/yxsx/index.htm`。
- (6) 服务器给出响应，把文件 `index.htm` 发给浏览器。
- (7) TCP 连接释放。
- (8) 浏览器显示“清华大学院系设置”文件 `index.htm` 中的所有文本。

非持久 HTTP 连接 (contains text, references to 10 jpeg images)

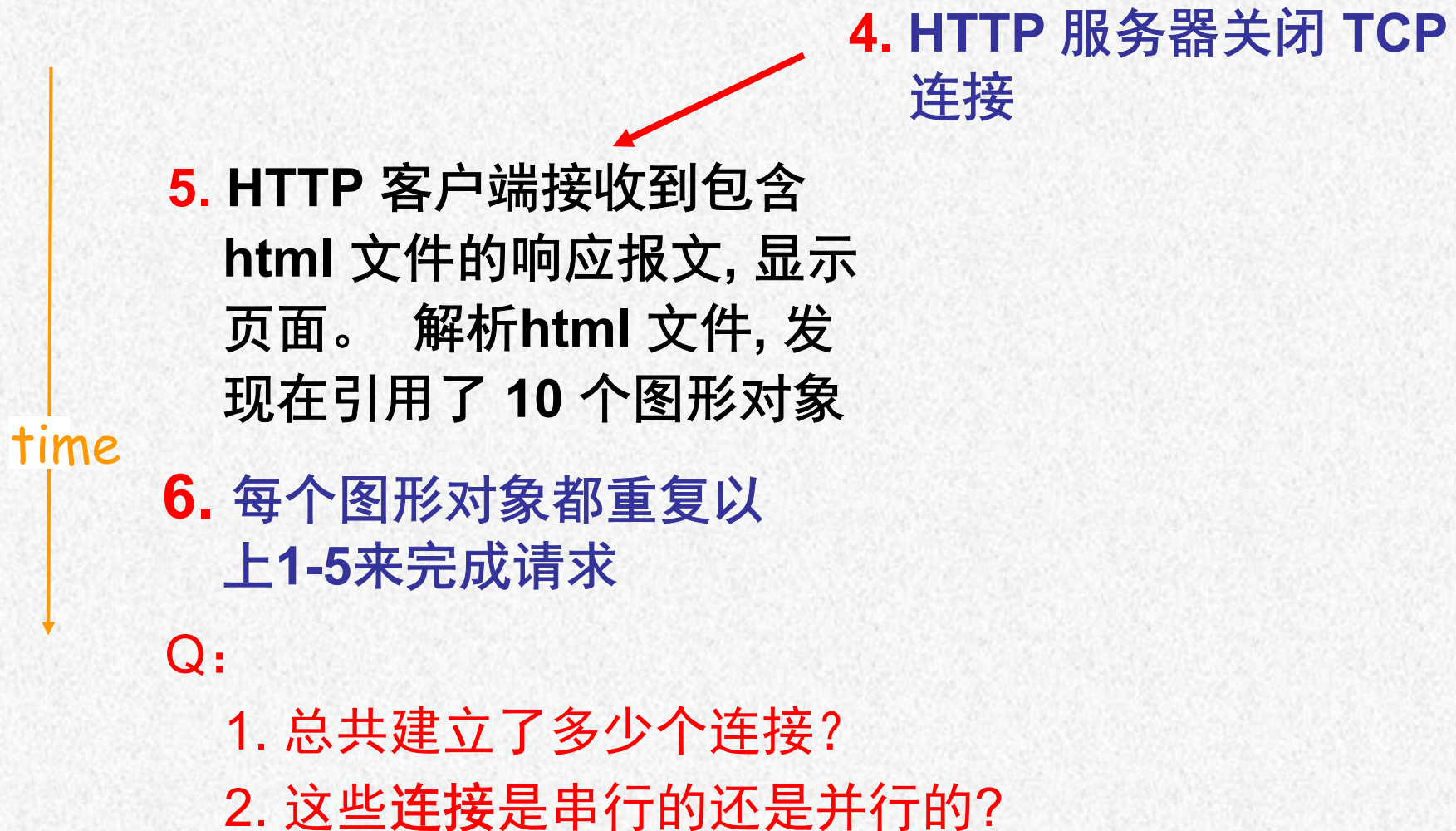
用户输入URL

`www.someSchool.edu/someDepartment/home.index`

-
1. HTTP 客户端向 HTTP 服务器 (www.someSchool.edu) 80端口发连接请求
 2. HTTP 服务器 (www.someSchool.edu) 在80端口等待TCP连接, 并接受连接请求, 通知客户端
 3. HTTP 通过TCP连接发送 HTTP 请求报文, 报文包含客户端想要的对象 `someDepartment/home.index`
 4. HTTP 服务接收到请求报文, 生成包含被请求对象的响应报文, 并将其发送给客户端

time

非持久 HTTP 连接(续)



Web服务响应时间

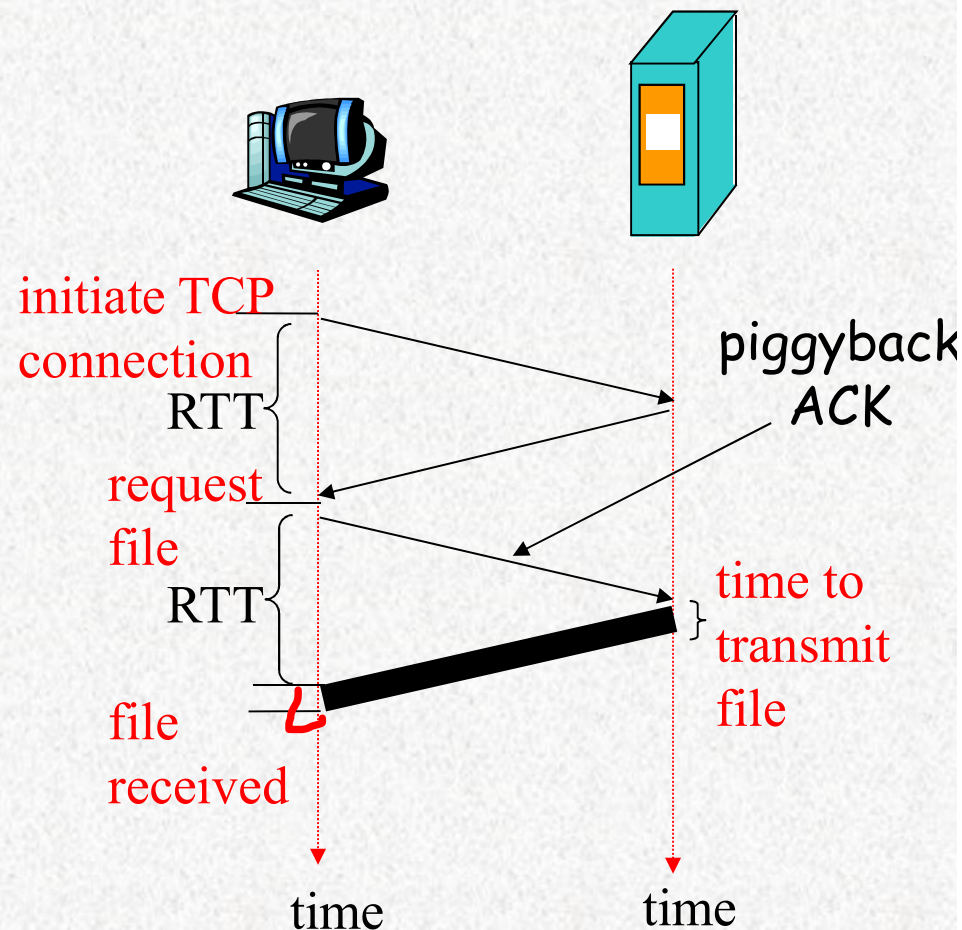
请求一个网页文档所需的时间

往返时间RTT的定义： 一个小分组从客户机到服务器，再回到客户机所花的时间

响应时间：

- 发起TCP连接用时1RTT
- 发送请求报文和收到响应报文用时1RTT 文件传输时间Round Trip Time

响应时间 = 2RTT+文件传输时间



持续连接

(persistent connection)

- **HTTP/1.1 协议使用持续连接。**
- **万维网服务器在发送响应后仍然在一段时间内保持这条连接，使同一个客户（浏览器）和该服务器可以继续在这条连接上传送后续的 HTTP 请求报文和响应报文。**
- **这并不局限于传送同一个页面上链接的文档，而是只要这些文档都在同一个服务器上就行。**
- **目前一些流行的浏览器（例如，IE 6.0）的默认设置就是使用 HTTP/1.1。**

持续HTTP连接

非持续连接的缺点:

- 请求一个对象用时 $2RRT$
- 操作系统必须为每个TCP连接分配资源
- 浏览器经常并行打开很多连接，给服务器带来严重负担

持续连接

- 服务器发送完响应报文后保留连接
- 相同客户机和服务器之间的后续请求和响应报文可通过相同的连接进行传送

持续HTTP连接（续）

非流水线方式持续连接:

- 客户机只有在前一个响应接收到之后才能发出新的请求
- 每个引用对象的请求和接收用时1RRT

流水线方式持续连接:

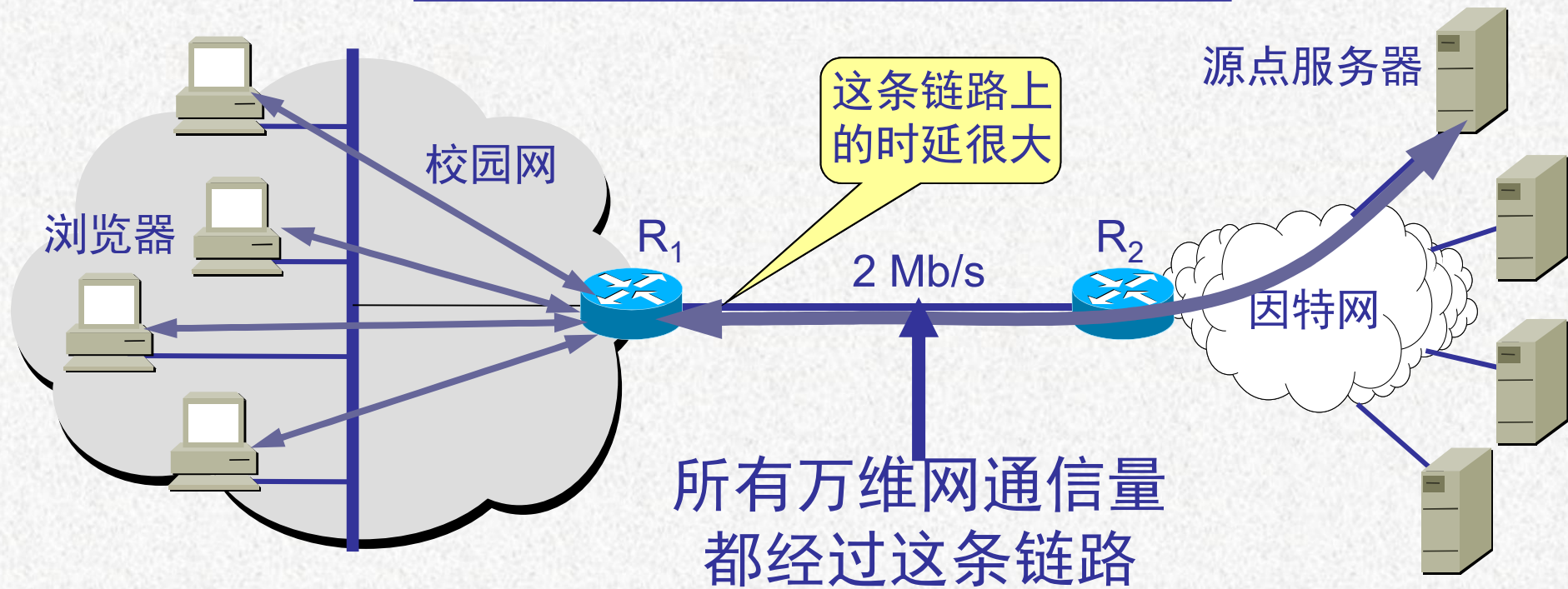
- HTTP/1.1默认模式
- 当browser在分析base HTML文件时，碰到一个referenced object就发出request消息，不管前面发送request消息的respond消息是否到达。
- 所有的引用对象可能只花费1RRT

代理服务器 (proxy server)

- **代理服务器(proxy server)**又称为万维网高速缓存(Web cache)，它代表浏览器发出 HTTP 请求。
- 万维网高速缓存把最近的一些请求和响应暂存在本地磁盘中。
- 当与暂时存放的请求相同的新请求到达时，万维网高速缓存就把暂存的响应发送出去，而不需要按 URL 的地址再去因特网访问该资源。

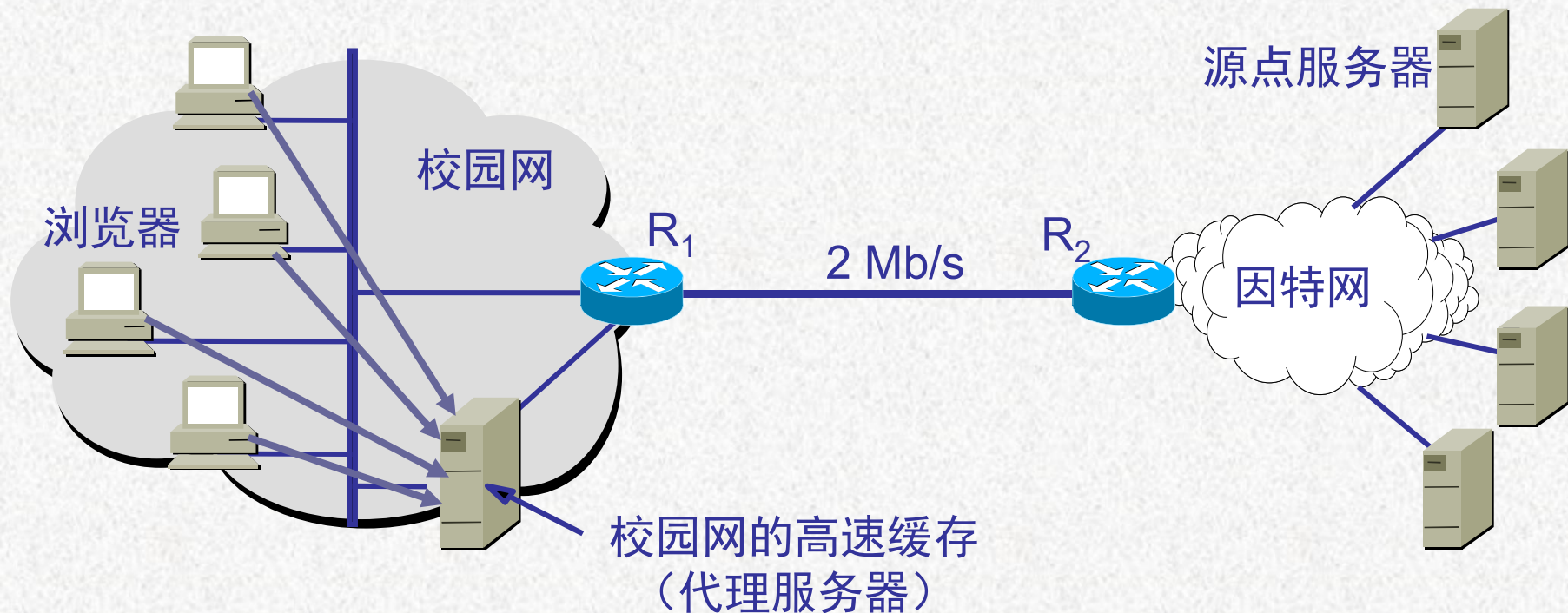
使用高速缓存可减少访问因特网服务器的时延

没有使用高速缓存的情况



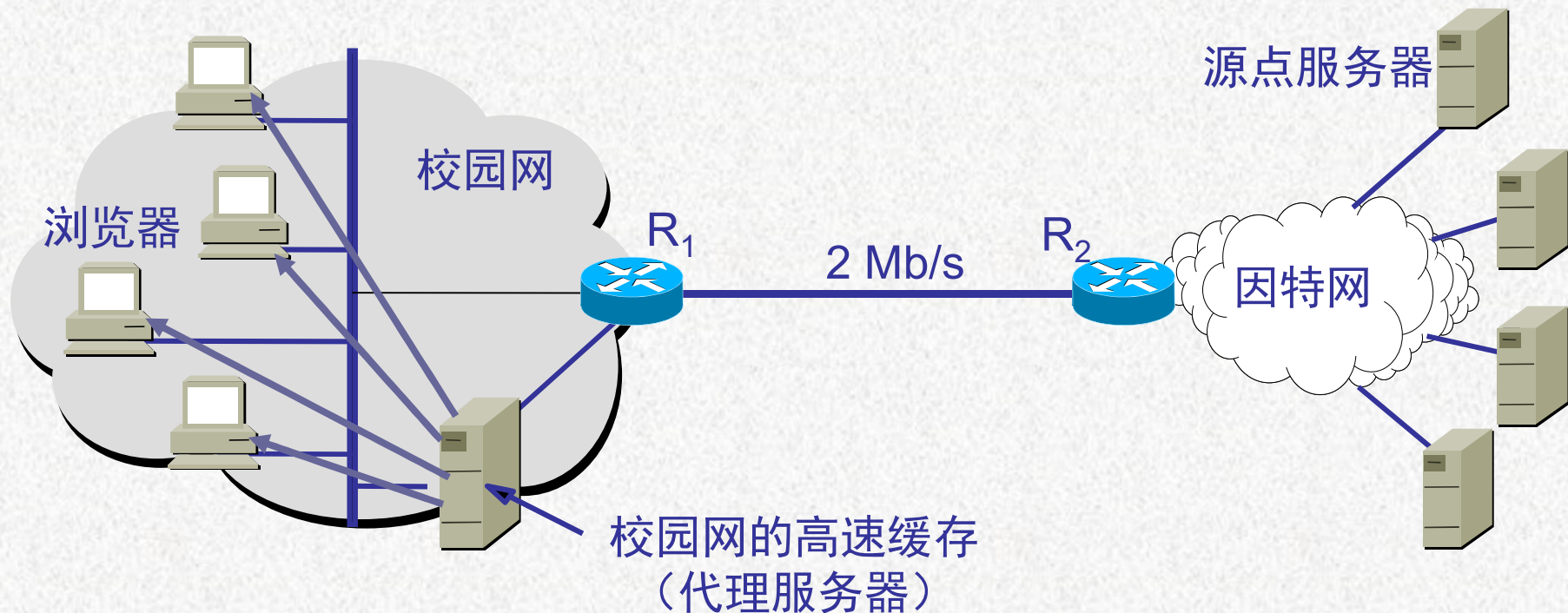
使用高速缓存的情况

(1) 浏览器访问因特网的服务器时，要先与校园网的高速缓存建立 **TCP** 连接，并向高速缓存发出 **HTTP** 请求报文



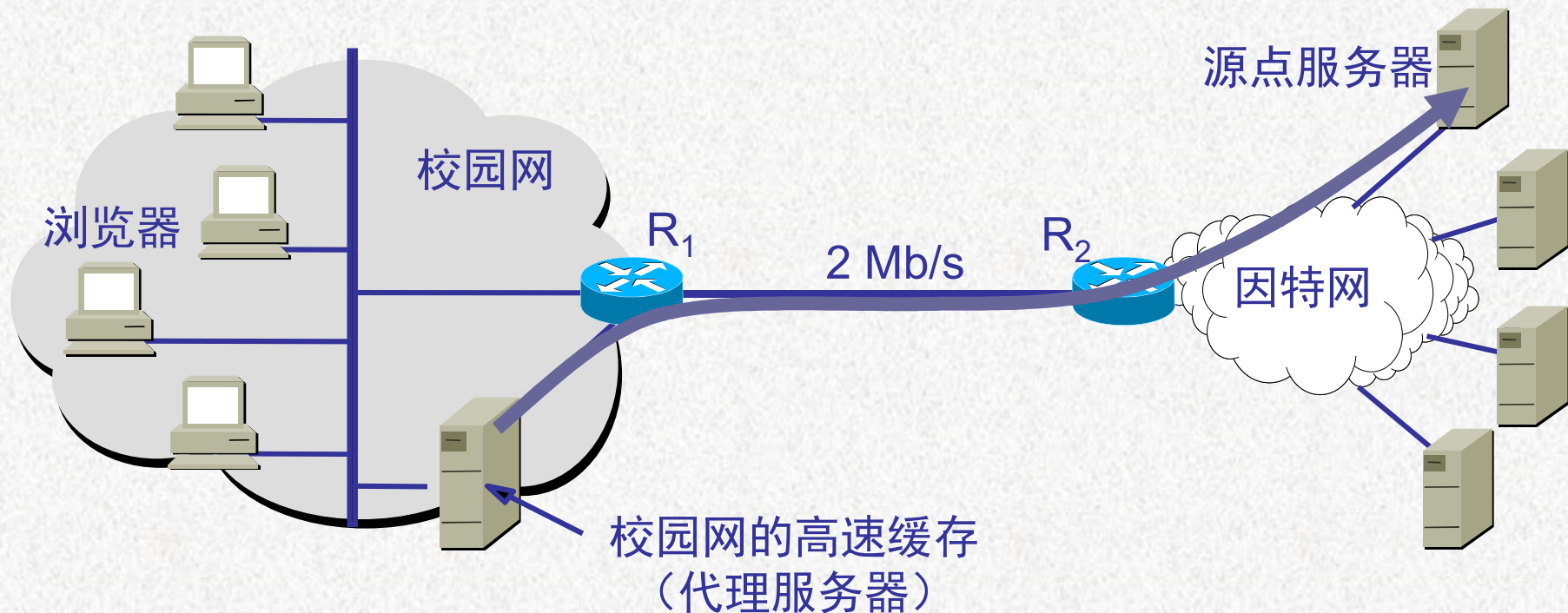
使用高速缓存的情况

(2) 若高速缓存已经存放了所请求的对象，则将此对象放入 **HTTP** 响应报文中返回给浏览器。



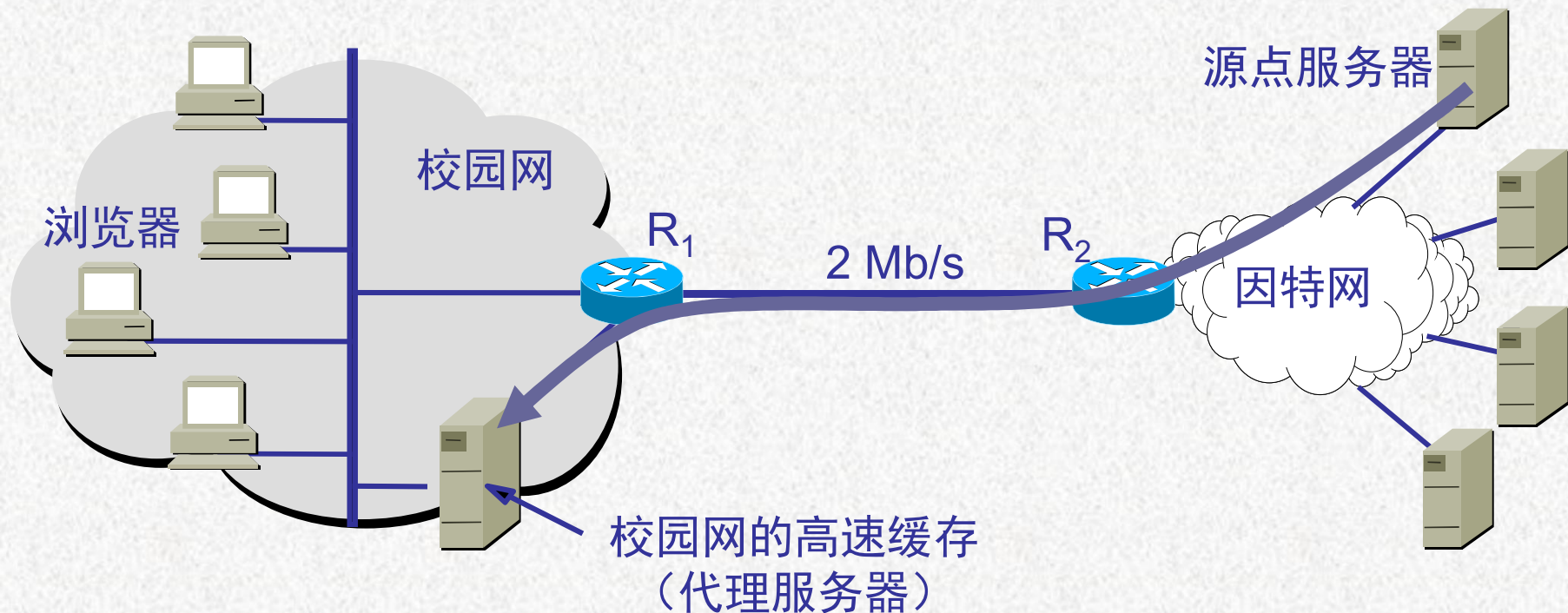
使用高速缓存的情况

(3) 否则，高速缓存就代表发出请求的用户浏览器，与因特网上的源点服务器建立 **TCP** 连接，并发送 **HTTP** 请求报文。



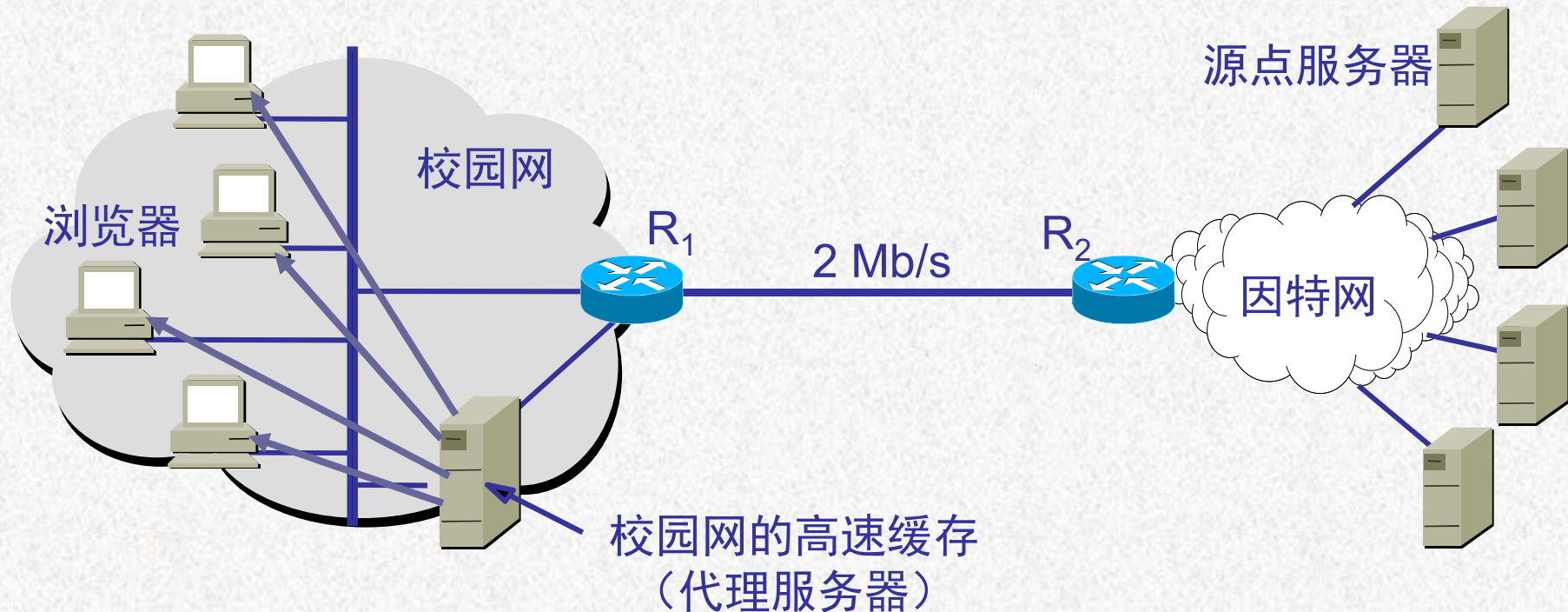
使用高速缓存的情况

(4) 源点服务器将所请求的对象放在 HTTP 响应报文中返回给校园网的高速缓存。



使用高速缓存的情况

(5) 高速缓存收到此对象后，先复制在其本地存储器中（为今后使用），然后再将该对象放在 **HTTP** 响应报文中，通过已建立的 **TCP** 连接，返回给请求该对象的浏览器。

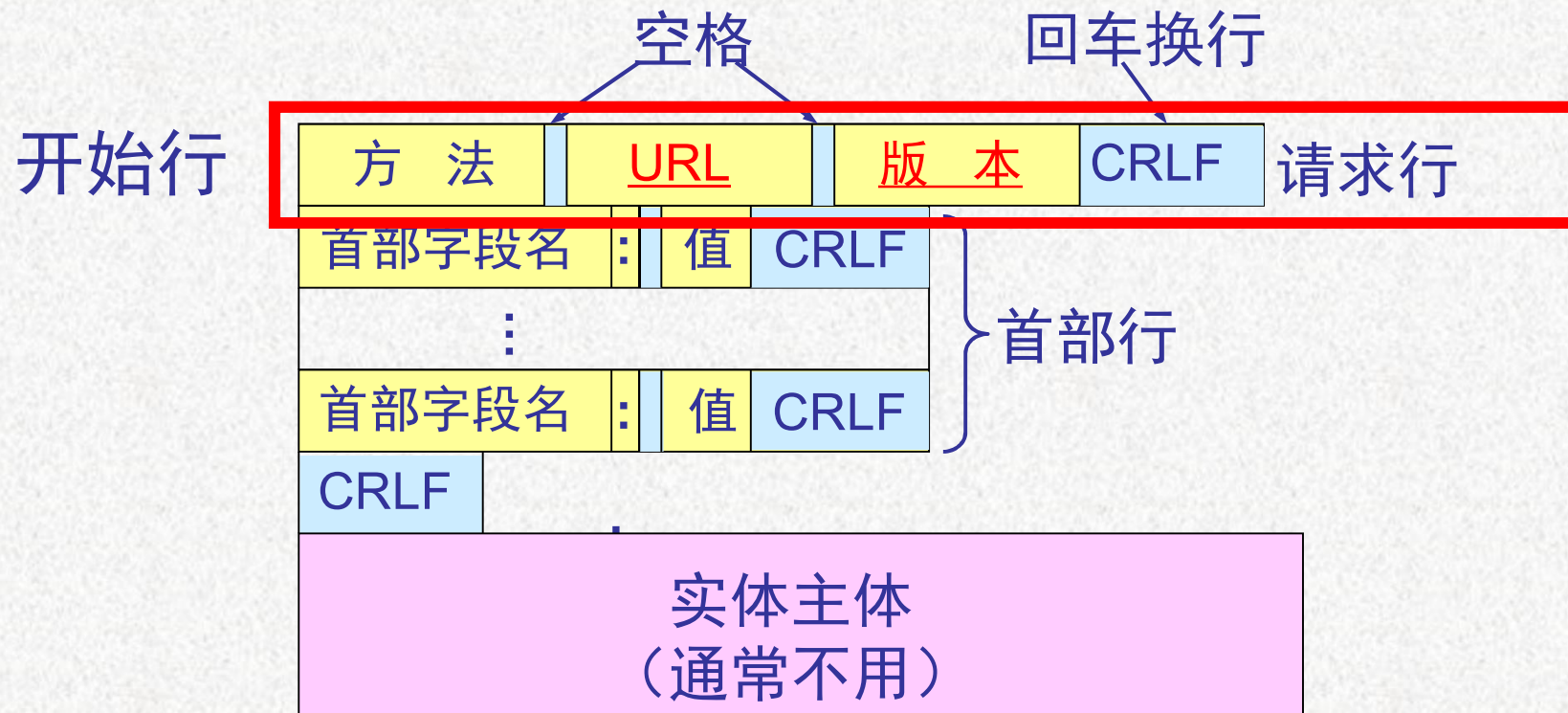


3. HTTP 的报文结构

HTTP 有两类报文：

- 请求报文——从客户向服务器发送请求报文。
- 响应报文——从服务器到客户的回答。
- 由于 HTTP 是面向正文的(text-oriented), 因此在报文中的每一个字段都是一些 ASCII 码串, 因而每个字段的长度都是不确定的。

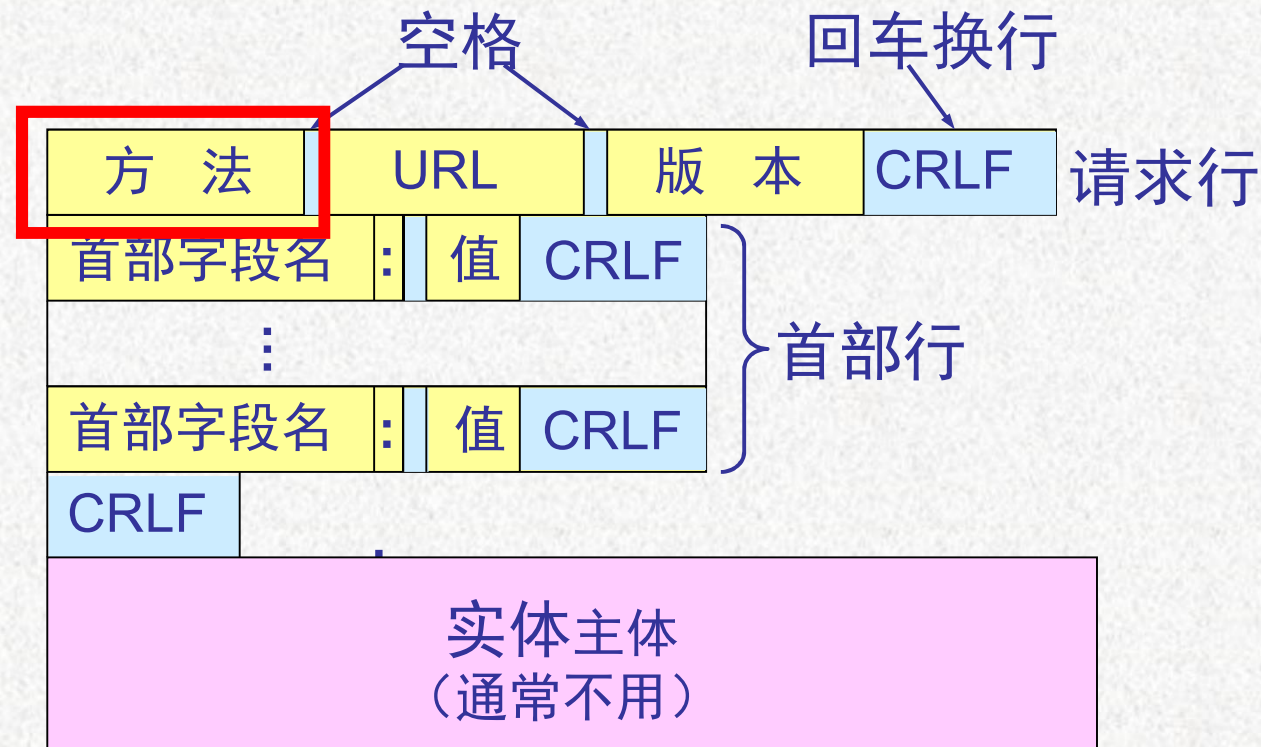
HTTP 的报文结构（请求报文）



报文由三个部分组成，即**开始行**、**首部行**和**实体主体**。

在请求报文中，开始行就是请求行。

HTTP 的报文结构（请求报文）

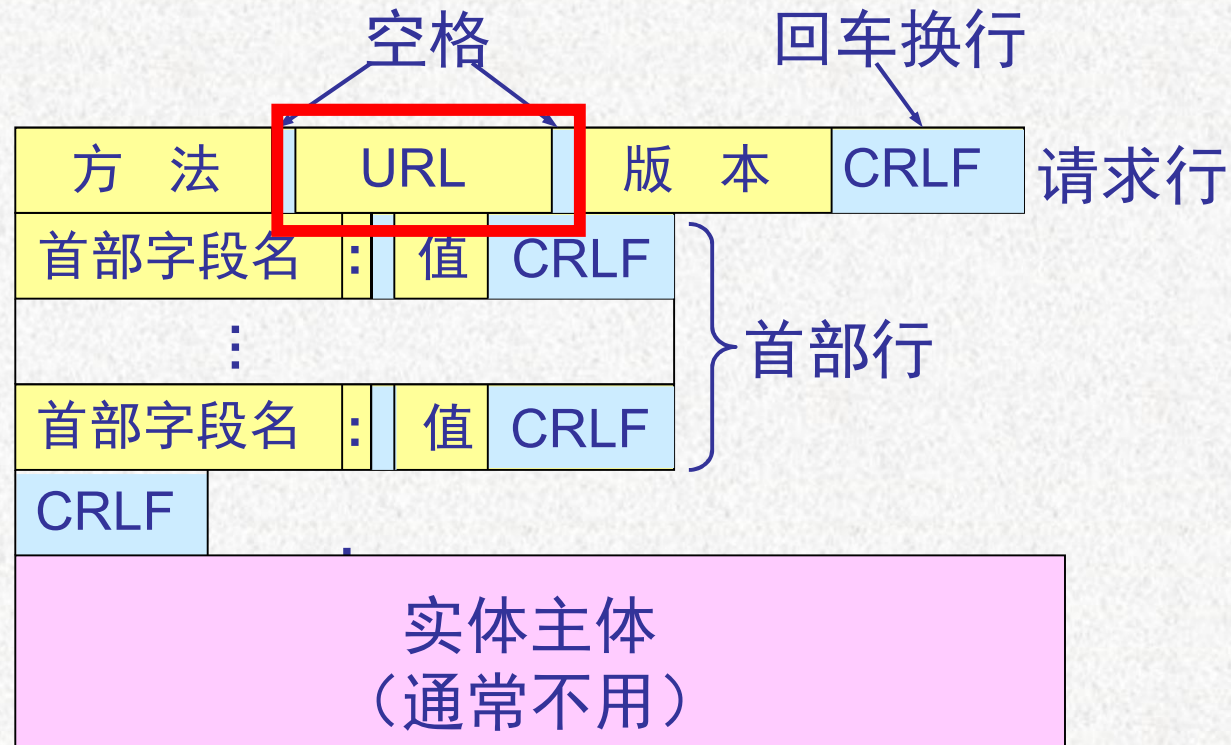


“**方法**”是面向对象技术中使用的专门名词。所谓“方法”就是对**所请求的对象进行的操作**，因此这些方法实际上也就是一些**命令**。因此，请求报文的类型是由它所采用的方法决定的。

HTTP 请求报文的一些方法

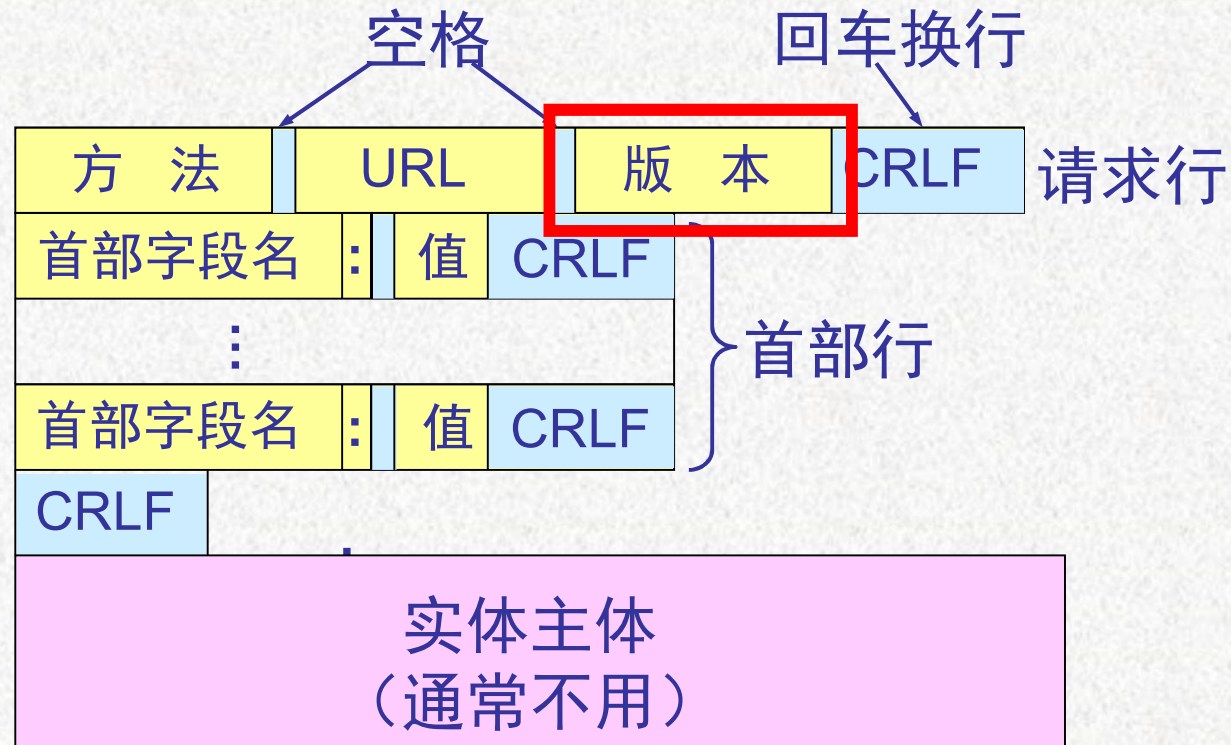
方法（操作）	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL 所标志的信息
HEAD	请求读取由 URL 所标志的信息的首部
POST	给服务器添加信息（例如，注释）
PUT	在指明的 URL 下存储一个文档
DELETE	删除指明的 URL 所标志的资源
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

HTTP 的报文结构（请求报文）



“URL”是所请求的资源的 URL。

HTTP 的报文结构（请求报文）



“版本” 是 HTTP 的版本。

HTTP报文格式：请求报文

■ HTTP 请求报文 (RFC2616) :

■ ASCII 格式

Mozilla= Mosaic +Godzilla

请求消息
(GET, POST,
HEAD commands)
首部行

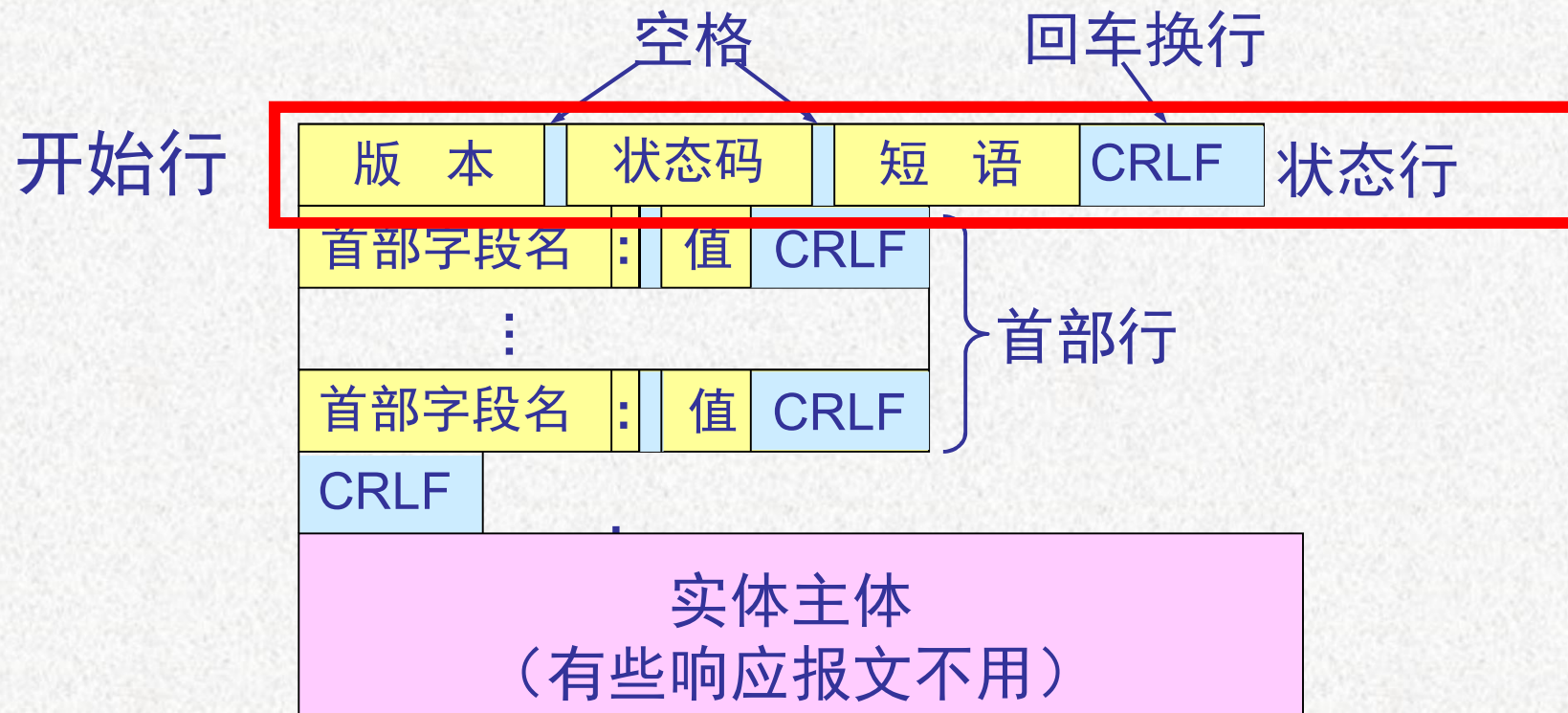
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/2.0
Connection: close
Accept-language: fr
```

回车+换行(0x0D0A)
指示消息头结束

(extra carriage return, line feed)

消息包含多行，行与行用
回车+换行隔开

HTTP 的报文结构（响应报文）



响应报文的开始行是**状态行**。

状态行包括三项内容，即 **HTTP 的版本**，**状态码**，以及解释状态码的**简单短语**。

Response消息包括三部分

- 1、状态行：包括version、status code、corresponding status信息等三个域。
- 2、首部行：
 - 1) Connection行：
 - close指Server在发送完被请求的object后将关闭TCP连接；
 - keep-alive指Server在发送完被请求的Object后保持TCP连接
 - 2) Date行：Server创建respond消息的时间
 - 3) Server行：Server的基本信息
 - 4) Last-Modified：所发送object被创建或最后修改的时间，与Web Cache有关
 - 5) Content-Length：所发送object的长度，单位为byte
 - 6) Content-type：所发送object的类型。
- 3、数据实体：包含被请求的object

HTTP 响应报文

状态行
(协议, 状态码
status code
status phrase)

首部行

HTTP/1.1 200 OK

Connection: close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998

Content-Length: 6821

Content-Type: text/html

数据
请求的
HTML 文档

data data data data data ...

4. 用户与服务器交互: cookies

Server希望能够识别用户身份，并根据用户身份来提供个性化服务，或者对用户的访问加以限制。

利用**Cookie**技术，**Server**可以跟踪用户操作。

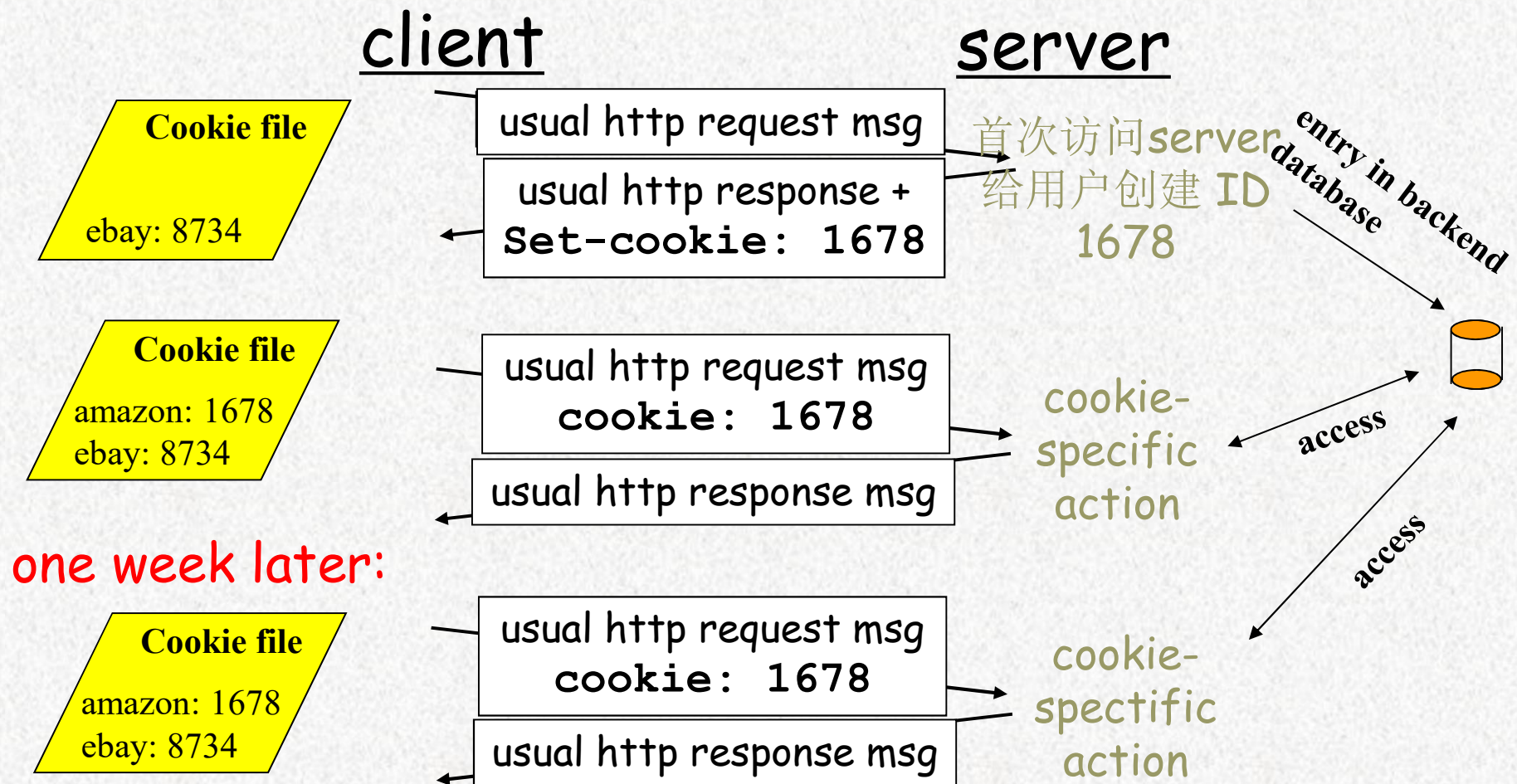
4个组成部分:

- 1) HTTP响应报文中有一个**Cookie**首部行
- 2) HTTP请求报文中有一个**Cookie**首部行
- 3) 用户系统中保留有一个**Cookie**文件，由用户浏览器管理
- 4) Web 站点有一个后台数据库

例:

- 张三经常用同一台计算机访问因特网
- 她第一次访问一个电子商务网站
- 当请求报文到达该Web服务器时，该Web站点将产生一个唯一识别码，度以此作为索引在它的后端数据库中产生一个项

Cookies: 记录 “状态”



Cookies (续)

可以记录些什么:

- 授权、认证
- 购物车信息
- 信息推荐
- 用户会话状态
(Web电子邮件)

Cookies 和隐私:

- **cookies** 可以使站点了解更多信息
- 用户可能会提供姓名和电子邮件给站点
- 搜索引擎用重定向和**Cookie**也可以获得用户信息
- 广告公司可以通过站点获得用户信息

6.5 电子邮件

- 6.5.1 电子邮件概述
- 6.5.2 简单邮件传送协议 **SMTP**
- 6.5.3 电子邮件的信息格式
- 6.5.4 邮件读取协议 **POP3** 和 **IMAP**
- 6.5.5 基于万维网的电子邮件
- 6.5.6 通用因特网邮件扩充 **MIME**

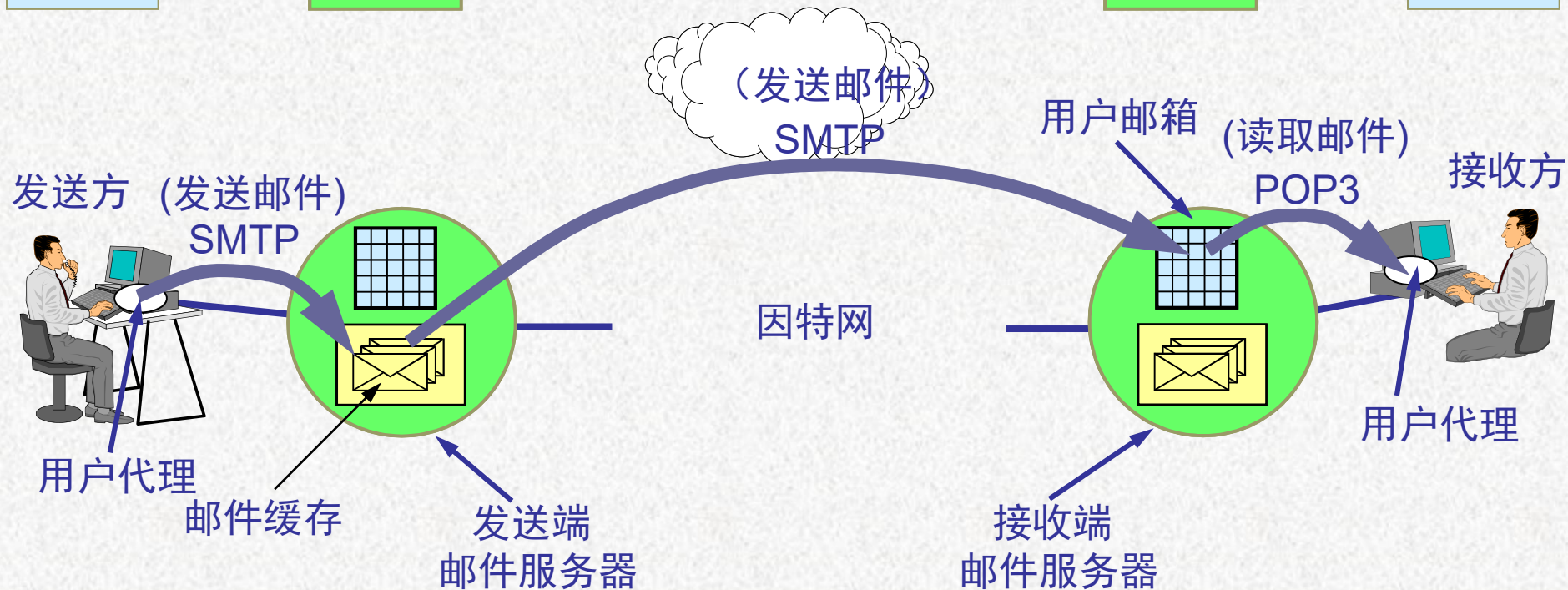
6.5.1 概述

- **电子邮件(e-mail)**是因特网上使用得最多的和最受用户欢迎的一种应用。
- 电子邮件把邮件发送到收件人使用的邮件服务器，并放在其中的收件人邮箱中，收件人可随时上网到自己使用的邮件服务器进行读取。
- 电子邮件不仅使用方便，而且还具有传递迅速和费用低廉的优点。
- 现在电子邮件不仅可传送文字信息，而且还可附上声音和图像。

电子邮件的一些标准

- 发送邮件的协议：**SMTP**
- 读取邮件的协议：**POP3** 和 **IMAP**
- **MIME** 在其邮件首部中说明了邮件的数据类型(如文本、声音、图像、视像等)，使用 **MIME** 可在邮件中同时传送多种类型的数据。

电子邮件的最主要的组成构件



用户代理 UA (User Agent)

- 用户代理 UA 就是用户与电子邮件系统的接口，是电子邮件客户端软件。
- 用户代理的功能：撰写、显示、处理和通信。
- 邮件服务器的功能是发送和接收邮件，同时还要向发信人报告邮件传送的情况（已交付、被拒绝、丢失等）。
- 邮件服务器按照客户服务器方式工作。邮件服务器需要使用发送和读取两个不同的协议。

应当注意

- 一个邮件服务器既可以作为客户，也可以作为服务器。
- 例如，当邮件服务器 **A** 向另一个邮件服务器 **B** 发送邮件时，邮件服务器 **A** 就作为 **SMTP** 客户，而 **B** 是 **SMTP** 服务器。
- 当邮件服务器 **A** 从另一个邮件服务器 **B** 接收邮件时，邮件服务器 **A** 就作为 **SMTP** 服务器，而 **B** 是 **SMTP** 客户。

发送和接收电子邮件的 几个重要步骤

- ❶ 发件人调用 **PC** 机中的用户代理撰写和编辑要发送的邮件。
- ❷ 发件人的用户代理把邮件用 **SMTP** 协议发给发送方邮件服务器，
- ❸ **SMTP** 服务器把邮件临时存放在邮件缓存队列中，等待发送。
- ❹ 发送方邮件服务器的 **SMTP** 客户与接收方邮件服务器的 **SMTP** 服务器建立 **TCP** 连接，然后就把邮件缓存队列中的邮件依次发送出去。

发送和接收电子邮件的 几个重要步骤（续）

- ⑤ 运行在接收方邮件服务器中的**SMTP**服务器进程收到邮件后，把邮件放入收件人的用户邮箱中，等待收件人进行读取。
- ⑥ 收件人在打算收信时，就运行 **PC** 机中的用户代理，使用 **POP3**（或 **IMAP**）协议读取发送给自己的邮件。
- 请注意，**POP3** 服务器和 **POP3** 客户之间的通信是由 **POP3** 客户发起的。

电子邮件的组成

- 电子邮件由**信封(envelope)**和**内容(content)**两部分组成。
- 电子邮件的传输程序根据邮件信封上的信息来传送邮件。用户在从自己的邮箱中读取邮件时才能见到邮件的内容。
- 在邮件的信封上，最重要的就是收件人的地址。

电子邮件地址的格式

- TCP/IP 体系的电子邮件系统规定电子邮件地址的格式如下：

收件人邮箱名@邮箱所在主机的域名 (6-1)

- 符号 “@”读作 “at”，表示 “在” 的意思。
- 例如，电子邮件地址
xiexiren@tsinghua.org.cn

这个用户名在该域名
的范围内是唯一的。

邮箱所在的主机的域名
在全世界必须是唯一的

6.5.2 简单邮件传送协议 SMTP

- **SMTP** 所规定的就是在两个相互通信的 **SMTP** 进程之间应如何交换信息。
- 由于 **SMTP** 使用客户服务器方式，因此负责发送邮件的 **SMTP** 进程就是 **SMTP** 客户，而负责接收邮件的 **SMTP** 进程就是 **SMTP** 服务器。
- **SMTP** 规定了 14 条命令和 21 种应答信息。每条命令用 **4 个字母组成**，而每一种应答信息一般只有一行信息，由一个 3 位数字的代码开始，后面附上（也可不附上）很简单的文字说明。

SMTP 通信的三个阶段

1. 连接建立：连接是在发送主机的 **SMTP** 客户和接收主机的 **SMTP** 服务器之间建立的。**SMTP**不使用中间的邮件服务器。
2. 邮件传送
3. 连接释放：邮件发送完毕后，**SMTP** 应释放 **TCP** 连接。

6.5.3 电子邮件的信息格式

- 一个电子邮件分为信封和内容两大部分。
- RFC 822 只规定了邮件内容中的首部(header)格式，而对邮件的主体(body)部分则让用户自由撰写。
- 用户写好首部后，邮件系统将自动地将信封所需的信息提取出来并写在信封上。所以用户不需要填写电子邮件信封上的信息。
- 邮件内容首部包括一些关键字，后面加上冒号。最重要的关键字是：To 和 Subject。

邮件内容的首部

- “To:”后面填入一个或多个收件人的电子邮件地址。用户只需打开地址簿，点击收件人名字，收件人的电子邮件地址就会自动地填入到合适的位置上。
- “Subject:”是邮件的主题。它反映了邮件的主要内容，便于用户查找邮件。
- 抄送 “Cc:”表示应给某某人发送一个邮件副本。
- “From”和“Date”表示发信人的电子邮件地址和发信日期。“Reply-To”是对方回信所用的地址。

6.5.4 邮件读取协议

POP3 和 IMAP

- 邮局协议 **POP** 是一个非常简单、但功能有限的邮件读取协议，现在使用的是它的第三个版本 **POP3**。
- **POP** 也使用客户服务器的工作方式。
- 在接收邮件的用户 **PC** 机中必须运行 **POP** 客户程序，而在用户所连接的 **ISP** 的邮件服务器中则运行 **POP** 服务器程序。

IMAP 协议

(Internet Message Access Protocol)

- **IMAP** 也是按客户服务器方式工作，现在较新的是版本 4，即 **IMAP4**。
- 用户在自己的 **PC** 机上就可以操纵 **ISP** 的邮件服务器的邮箱，就像在本地操纵一样。
- 因此 **IMAP** 是一个联机协议。当用户 **PC** 机上的 **IMAP** 客户程序打开 **IMAP** 服务器的邮箱时，用户就可看到邮件的首部。若用户需要打开某个邮件，则该邮件才传到用户的计算机上。

IMAP 的特点

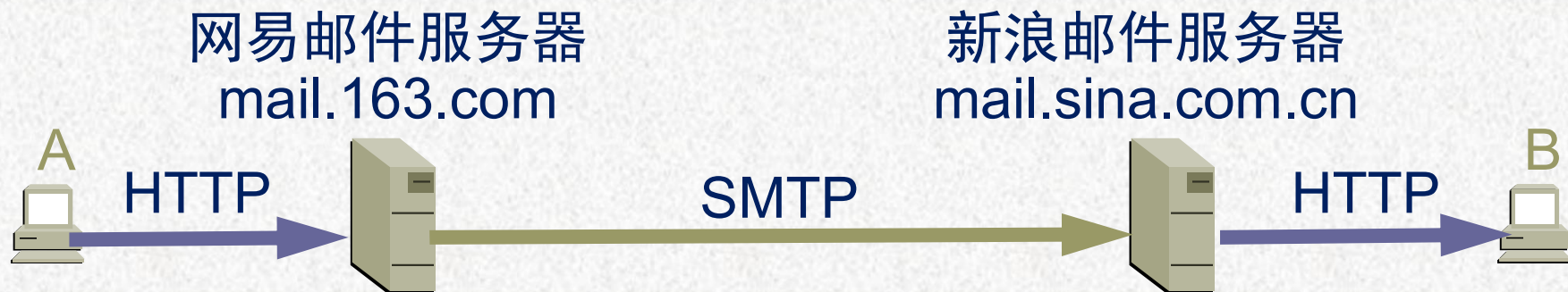
- **IMAP**最大的好处就是用户可以在不同的地方使用不同的计算机随时上网阅读和处理自己的邮件。
- **IMAP** 还允许收件人只读取邮件中的某一个部分。例如，收到了一个带有视像附件（此文件可能很大）的邮件。为了节省时间，可以先下载邮件的正文部分，待以后有时间再读取或下载这个很长的附件。
- **IMAP** 的缺点是如果用户没有将邮件复制到自己的PC机上，则邮件一直是存放在 **IMAP** 服务器上。因此用户需要经常与 **IMAP** 服务器建立连接。

必须注意

- 不要将邮件读取协议 **POP** 或 **IMAP** 与邮件传送协议 **SMTP** 弄混。
- 发信人的用户代理向源邮件服务器发送邮件，以及源邮件服务器向目的邮件服务器发送邮件，都是使用 **SMTP** 协议。
- 而 **POP** 协议或 **IMAP** 协议则是用户从目的邮件服务器上读取邮件所使用的协议。

6.5.5 基于万维网的电子邮件

- 电子邮件从 **A** 发送到网易邮件服务器是使用 **HTTP** 协议。
- 两个邮件服务器之间的传送使用 **SMTP**。
- 邮件从新浪邮件服务器传送到 **B** 是使用 **HTTP** 协议。



6.5.6 通用因特网邮件扩充 MIME

1. MIME 概述

SMTP 有以下缺点：

- **SMTP** 不能传送可执行文件或其他的二进制对象。
- **SMTP** 限于传送 7 位的 **ASCII** 码。许多其他非英语国家的文字（如中文、俄文，甚至带重音符号的法文或德文）就无法传送。
- **SMTP** 服务器会拒绝超过一定长度的邮件。
- 某些 **SMTP** 的实现并没有完全按照[RFC 821]的 **SMTP** 标准。

MIME 的特点

- **MIME** 并没有改动 **SMTP** 或取代它。
- **MIME** 的意图是继续使用目前的[RFC 822]格式，但增加了邮件主体的结构，并定义了传送非 **ASCII** 码的编码规则。

MIME 和 SMTP 的关系



MIME 主要包括三个部分

- 5 个新的邮件首部字段，它们可包含在[RFC 822]首部中。这些字段提供了有关邮件主体的信息。
- 定义了许多邮件内容的格式，对多媒体电子邮件的表示方法进行了标准化。
- 定义了传送编码，可对任何内容格式进行转换，而不会被邮件系统改变。

MIME 增加 5 个 新的邮件首部

- **MIME-Version:** 标志 **MIME** 的版本。现在的版本号是 **1.0**。若无此行，则为英文文本。
- **Content-Description:** 这是可读字符串，说明此邮件是什么。和邮件的主题差不多。
- **Content-Id:** 邮件的唯一标识符。
- **Content-Transfer-Encoding:** 在传送时邮件的主体是如何编码的。
- **Content-Type:** 说明邮件的性质。

2. 内容传送编码 (Content-Transfer-Encoding)

- 最简单的编码就是 7 位 **ASCII** 码，而每行不能超过 1000 个字符。**MIME** 对这种由 **ASCII** 码构成的邮件主体不进行任何转换。
- 另一种编码称为 **quoted-printable**，这种编码方法适用于当所传送的数据中只有少量的非 **ASCII** 码。
- 对于任意的二进制文件，可用 **base64** 编码。

3. 内容类型

- **MIME** 标准规定 **Content-Type** 说明必须含有两个标识符，即内容类型(**type**)和子类型(**subtype**)，中间用 “/” 分开。
- **MIME** 标准定义了 7 个基本内容类型和 15 种子类型。

6.6 动态主机配置协议 DHCP

6.6 动态主机配置协议 DHCP

- 为了将软件协议做成通用的和便于移植，协议软件的编写者把协议软件参数化。这就使得在很多台计算机上使用同一个经过编译的二进制代码成为可能。
- 一台计算机和另一台计算机的区别，都可通过一些不同的参数来体现。
- 在软件协议运行之前，必须给每一个参数赋值。

协议配置

- 在协议软件中给这些参数赋值的动作叫做**协议配置**。
- 一个软件协议在使用之前必须是已正确配置的。
- 具体的配置信息有哪些则取决于协议栈。

协议配置（续）

- 需要配置的项目

- (1) IP 地址

- (2) 子网掩码

- (3) 默认路由器的 IP 地址

- (4) 域名服务器的 IP 地址

- 这些信息通常存储在一个配置文件中，计算机在引导过程中可以对这个文件进行存取。

动态主机配置协议 DHCP

(Dynamic Host Configuration Protocol)

- 动态主机配置协议 DHCP 提供了即插即用连网 (plug-and-play networking) 的机制。
- 允许一台计算机加入新的网络和获取IP地址而不用手工参与。

DHCP 使用客户服务器方式。

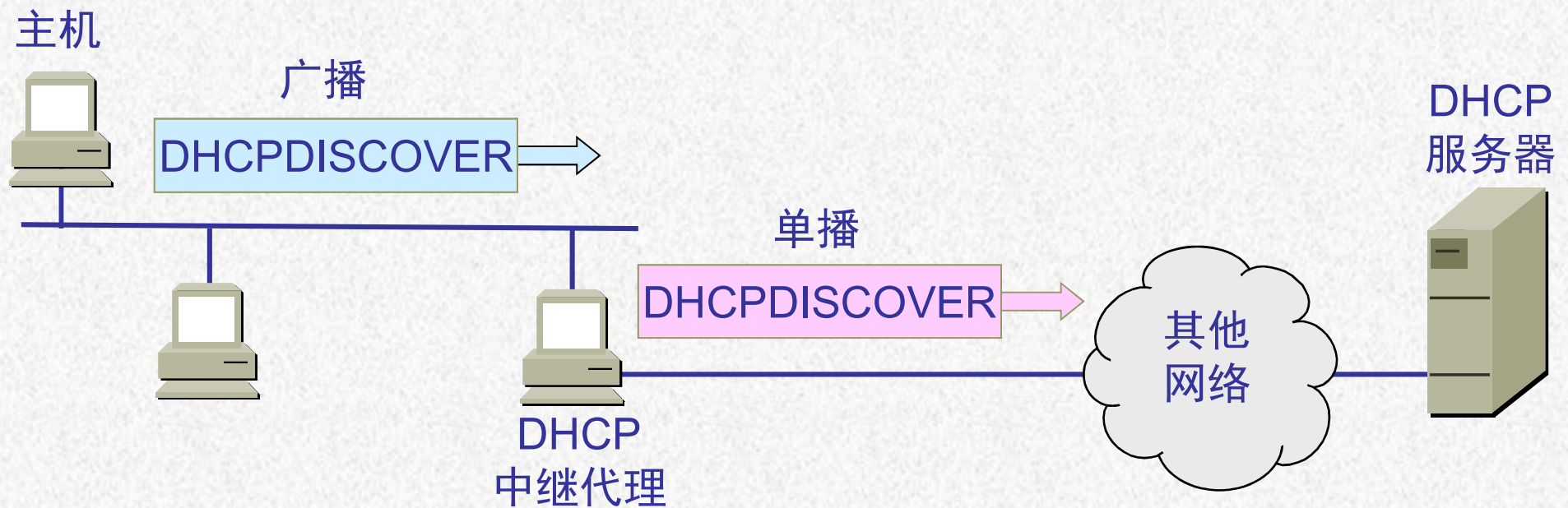
- 需要 IP 地址的主机在启动时就向 **DHCP 服务器广播发送发现报文**（**DHCPDISCOVER**）。
- 本地网络上所有主机都能收到此广播报文，但只有 **DHCP 服务器**才回答此广播报文。
- **DHCP 服务器**先在其数据库中查找该计算机的配置信息。
 - 若找到，则返回找到的信息。
 - 否则，从服务器的 **IP 地址池**分配一个地址给该计算机。
- **DHCP 服务器**的回答报文叫做**提供报文**（**DHCPOFFER**）。

DHCP 中继代理(relay agent)

- 并不是每个网络上都有 DHCP 服务器，这样会使 DHCP 服务器的数量太多。现在是每一个网络至少有一个 DHCP 中继代理，它配置了 DHCP 服务器的 IP 地址信息。
- 当 DHCP 中继代理收到主机发送的发现报文后，就以单播方式向 DHCP 服务器转发此报文，并等待其回答。收到 DHCP 服务器回答的提供报文后，DHCP 中继代理再将此提供报文发回给主机。

DHCP 中继代理

以单播方式转发发现报文

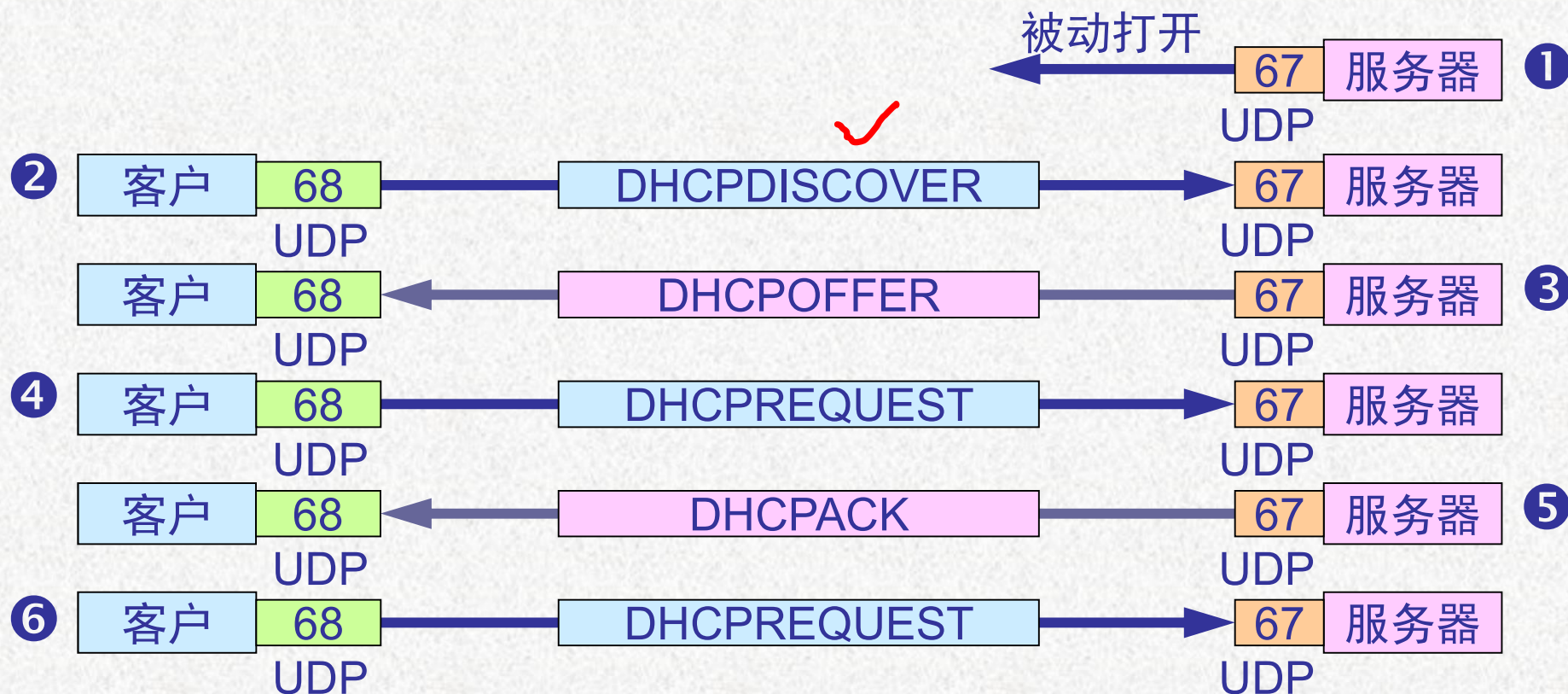


注意：DHCP 报文只是 UDP 用户数据报中的数据。

租用期(lease period)

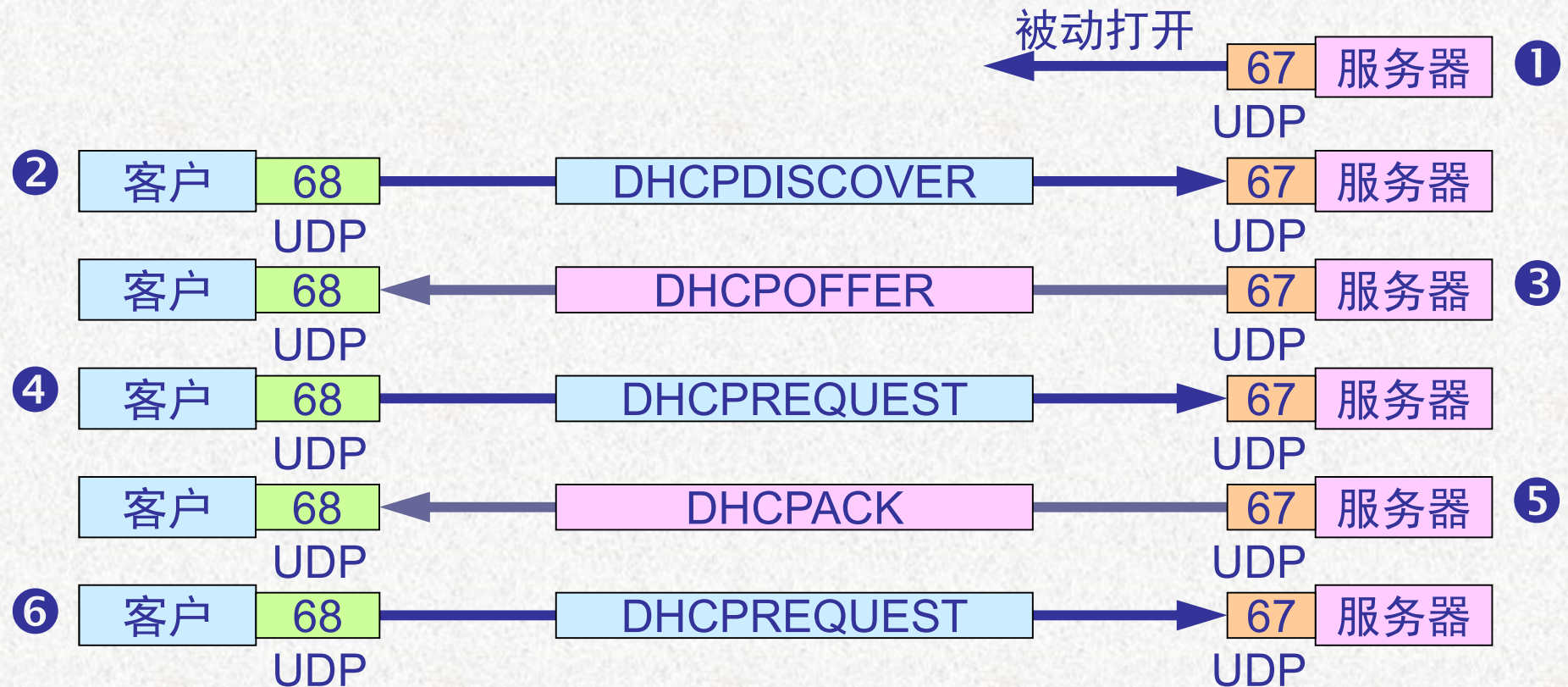
- **租用期**：DHCP 服务器分配给 DHCP 客户的 IP 地址是临时的，因此 DHCP 客户只能在**一段有限的时间**内使用这个分配到的 IP 地址。
- 租用期的数值应由 DHCP 服务器自己决定。
- DHCP 客户也可在自己发送的报文中（例如，发现报文）**提出对租用期的要求**。

DHCP 协议的工作过程



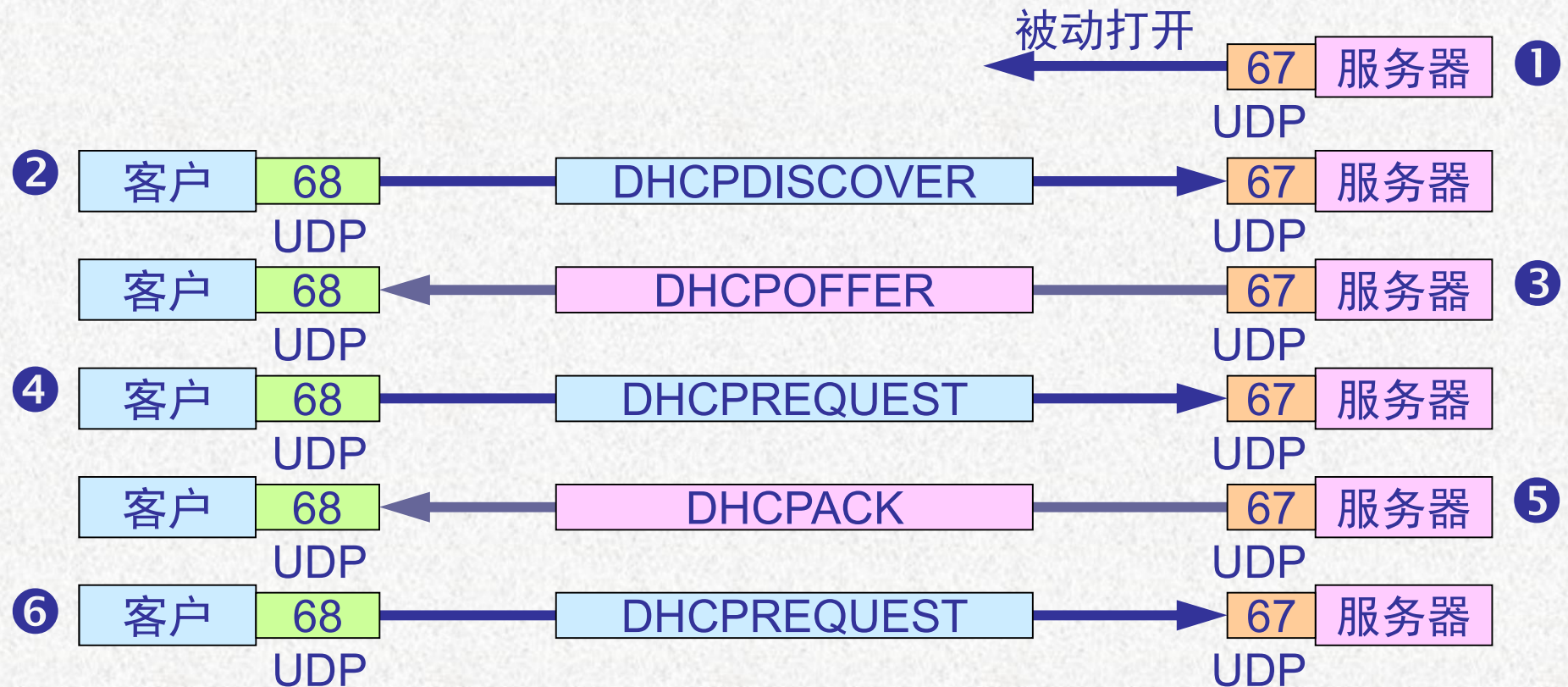
①：DHCP 服务器被动打开 UDP 端口 67，等待客户端发来的报文。

DHCP 协议的工作过程



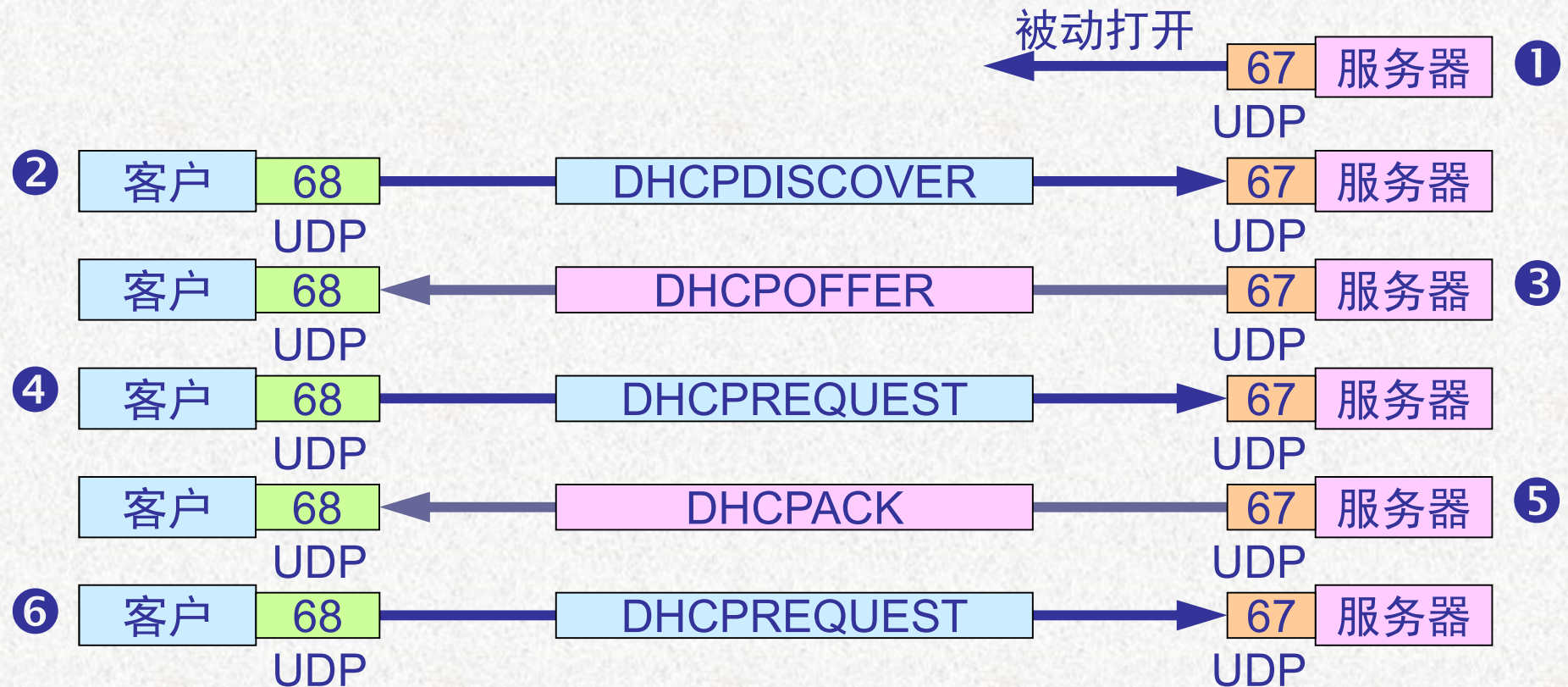
②: DHCP 客户从 UDP 端口 68 发送 DHCP 发现报文。

DHCP 协议的工作过程



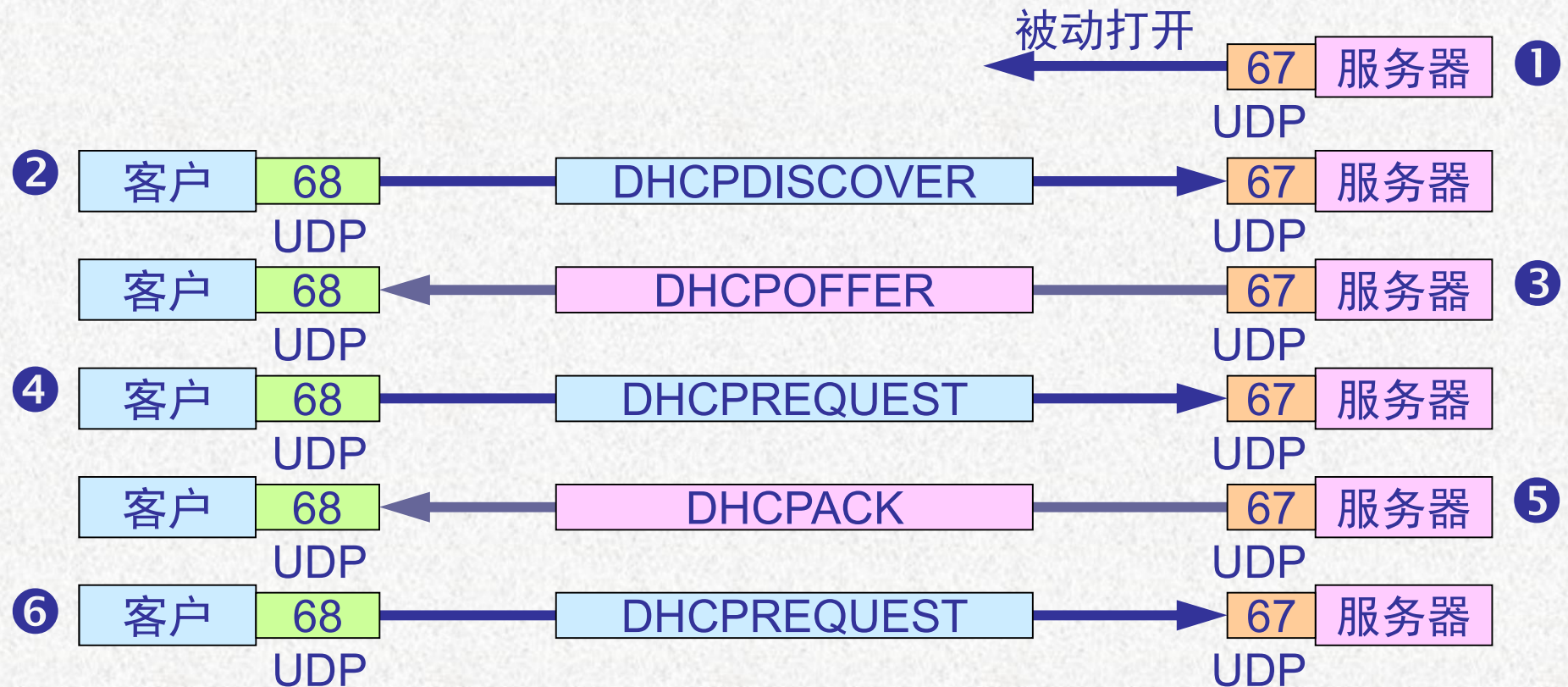
③： 凡收到 DHCP 发现报文的 DHCP 服务器都发出 DHCP 提供报文，因此 DHCP 客户可能收到多个 DHCP 提供报文。

DHCP 协议的工作过程



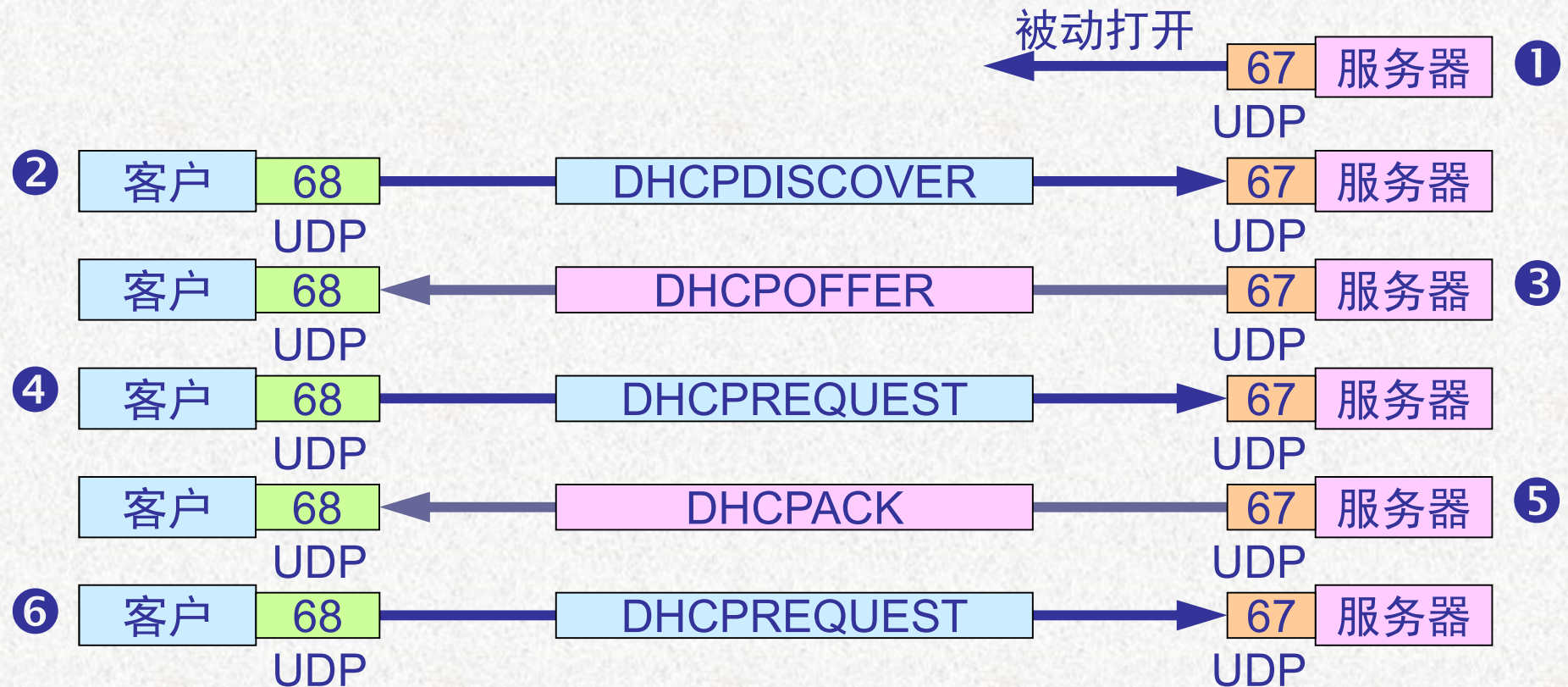
④： DHCP 客户从几个 DHCP 服务器中选择其中的一个，并向所选择的 DHCP 服务器发送 DHCP 请求报文。

DHCP 协议的工作过程



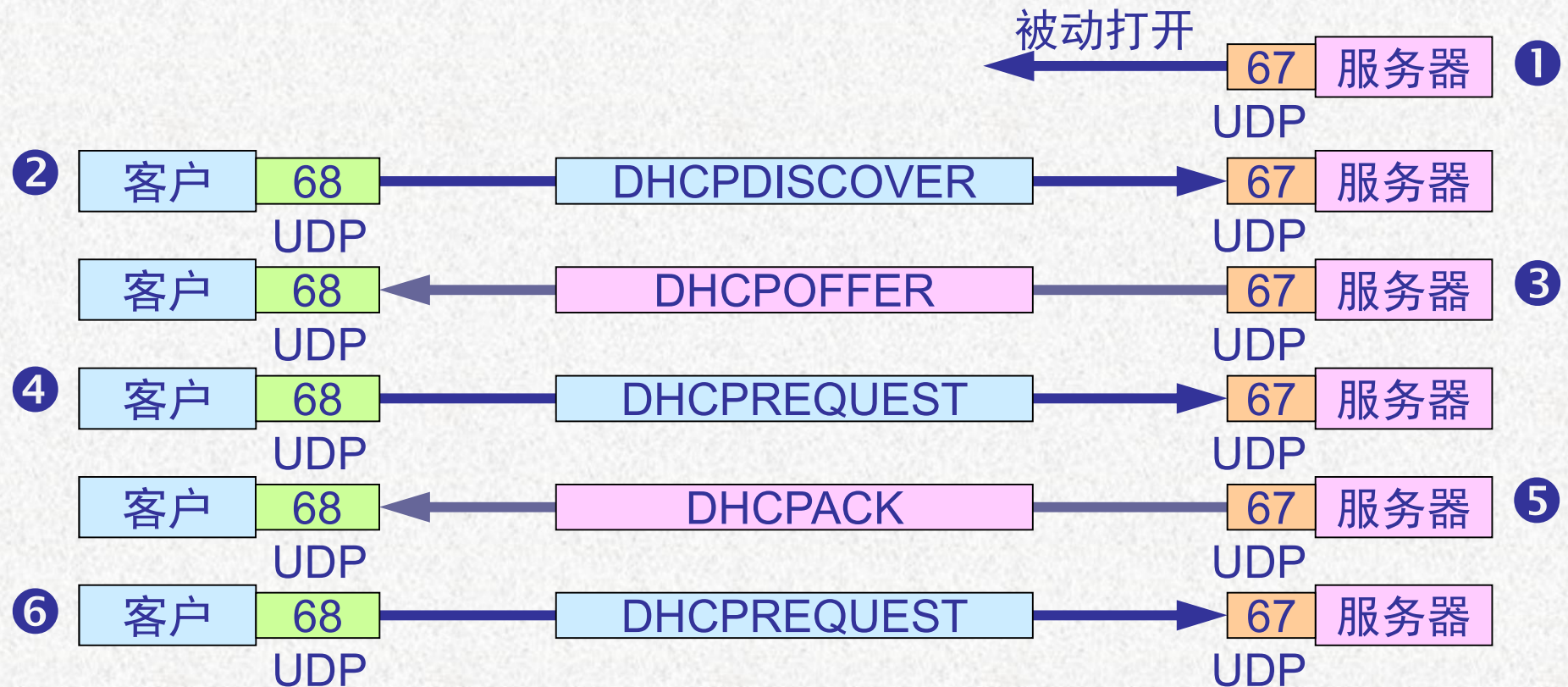
⑤： 被选择的 DHCP 服务器发送确认报文 DHCPACK，进入已绑定状态，并可开始使用得到的临时 IP 地址了。

DHCP 协议的工作过程



DHCP 客户现在要根据服务器提供的租用期 T 设置两个计时器 T_1 和 T_2 ，它们的超时时间分别是 $0.5T$ 和 $0.875T$ 。当超时时间到就要请求更新租用期。

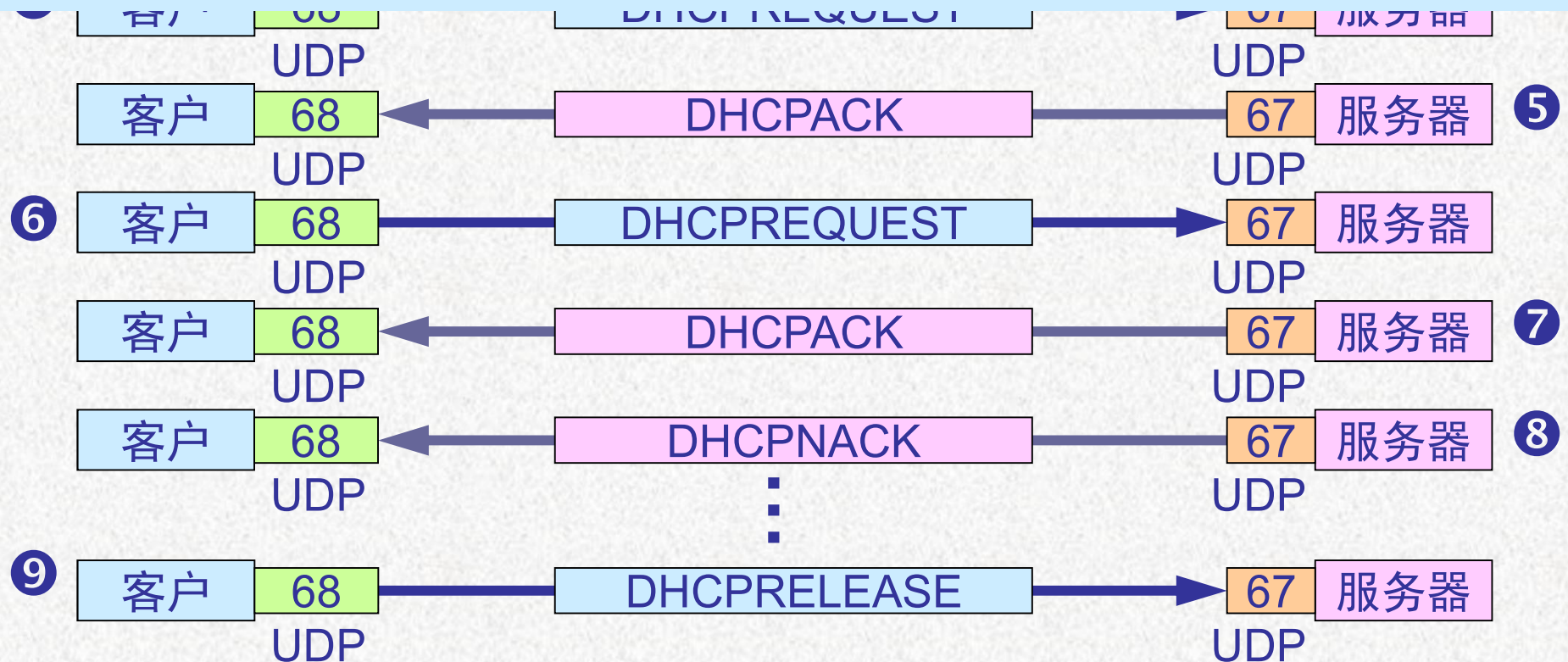
DHCP 协议的工作过程



⑥： 租用期过了一半（T1 时间到），DHCP 发送请求报文 **DHCPREQUEST** 要求更新租用期。

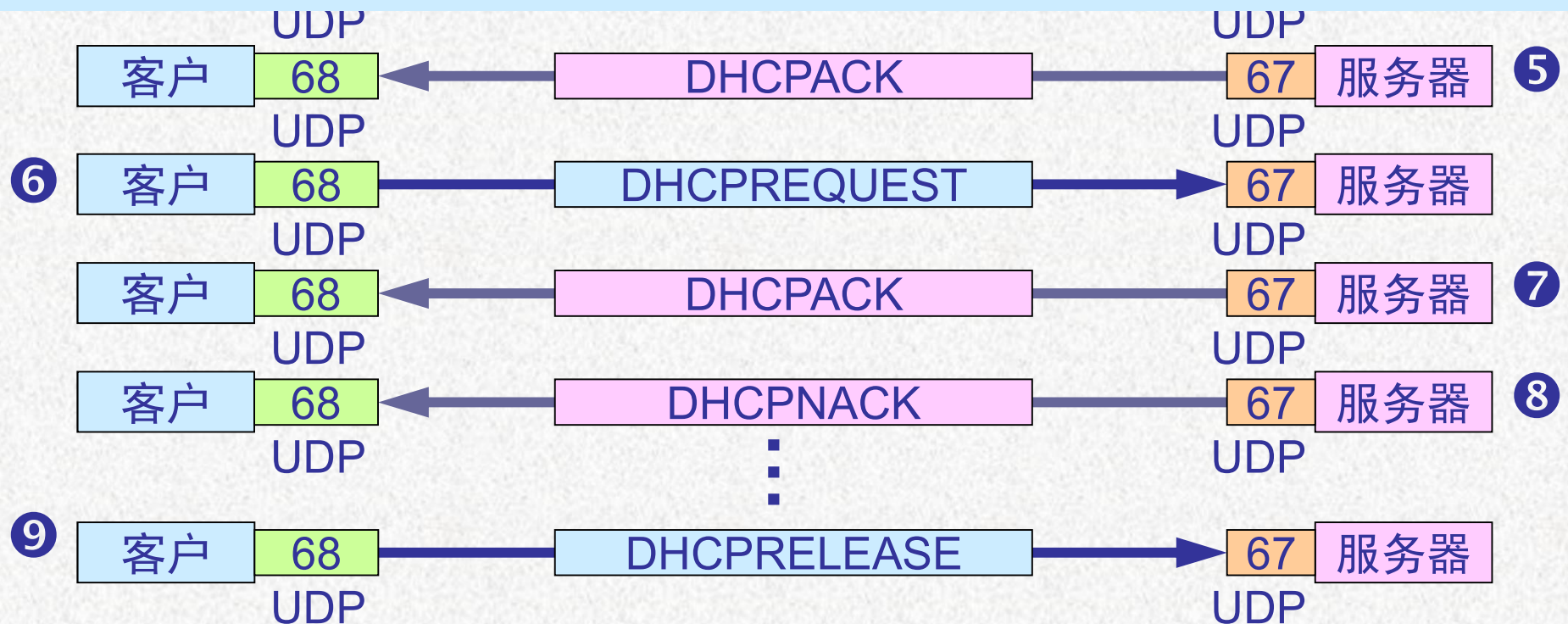
DHCP 协议的工作过程

⑦： DHCP 服务器若同意，则发回确认报文 DHCPACK。DHCP 客户得到了新的租用期，重新设置计时器。



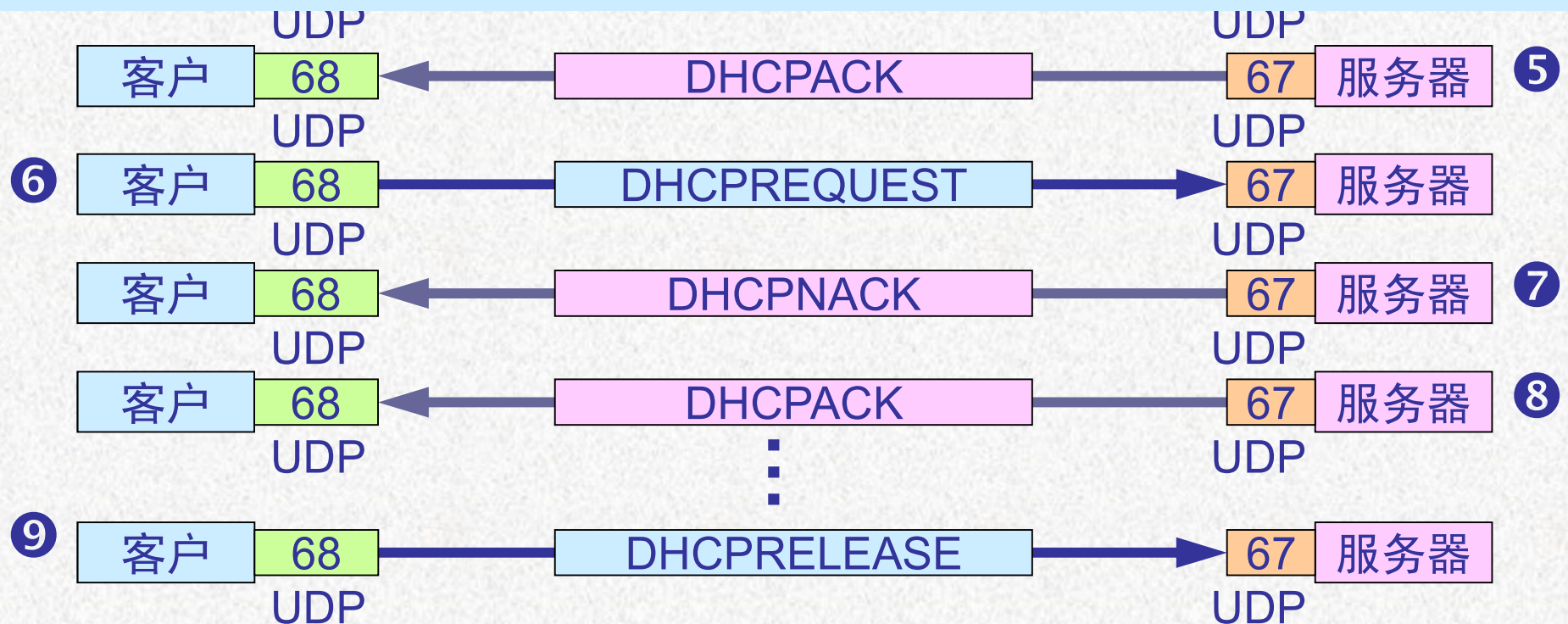
DHCP 协议的工作过程

⑧：DHCP 服务器若不同意，则发回否认报文 DHCPNACK。这时 DHCP 客户必须立即停止使用原来的 IP 地址，而必须重新申请 IP 地址（回到步骤**②**）。



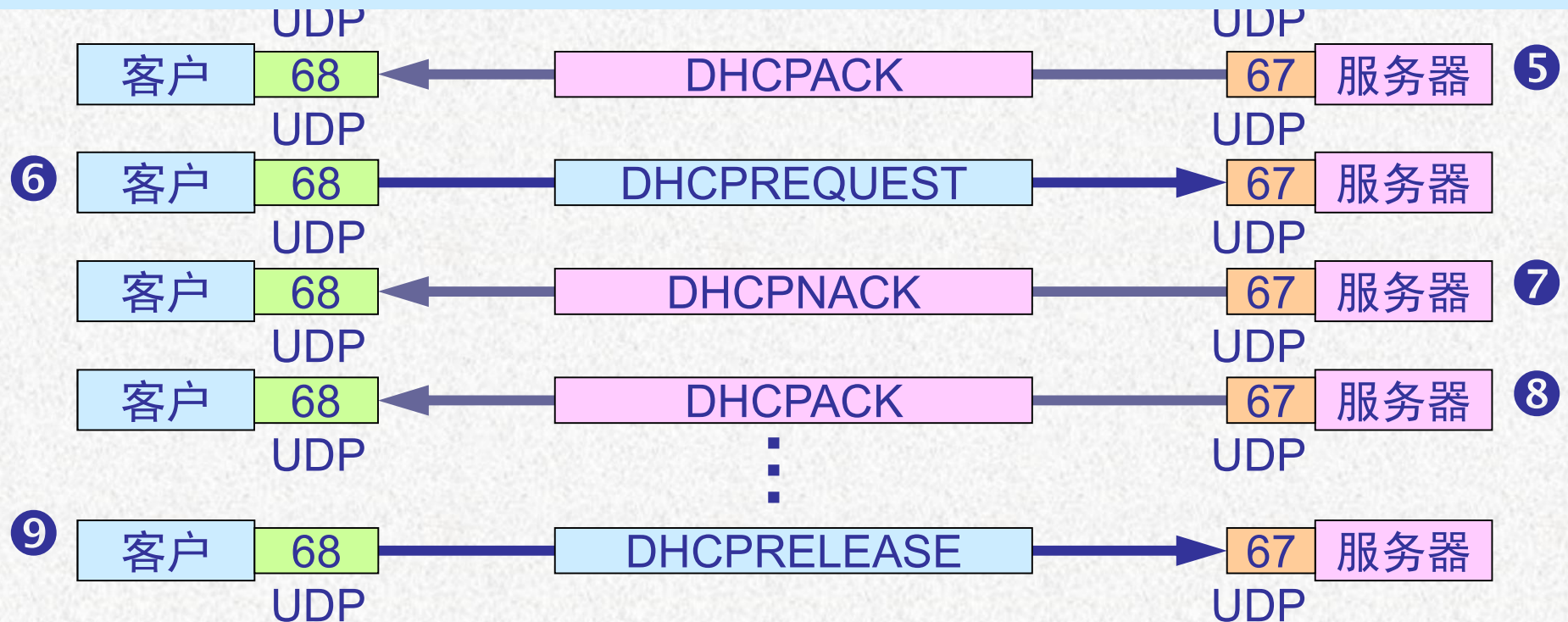
DHCP 协议的工作过程

若 DHCP 服务器不响应步骤 ⑥ 的请求报文 DHCPREQUEST，则在租用期过了 87.5% 时，DHCP 客户必须重新发送请求报文 DHCPREQUEST（重复步骤 ⑥），然后又继续后面的步骤。



DHCP 协议的工作过程

⑨：DHCP 客户可随时提前终止服务器所提供的租用期，这时只需向 DHCP 服务器发送释放报文 DHCPRELEASE 即可。



6.7 简单网络管理协议 SNMP

- 6.7.1 网络管理的基本概念
- 6.7.2 管理信息结构 SMI
- 6.7.3 管理信息库 MIB
- 6.7.4 SNMP 的协议数据单元和报文

6.7.1 网络管理的基本概念

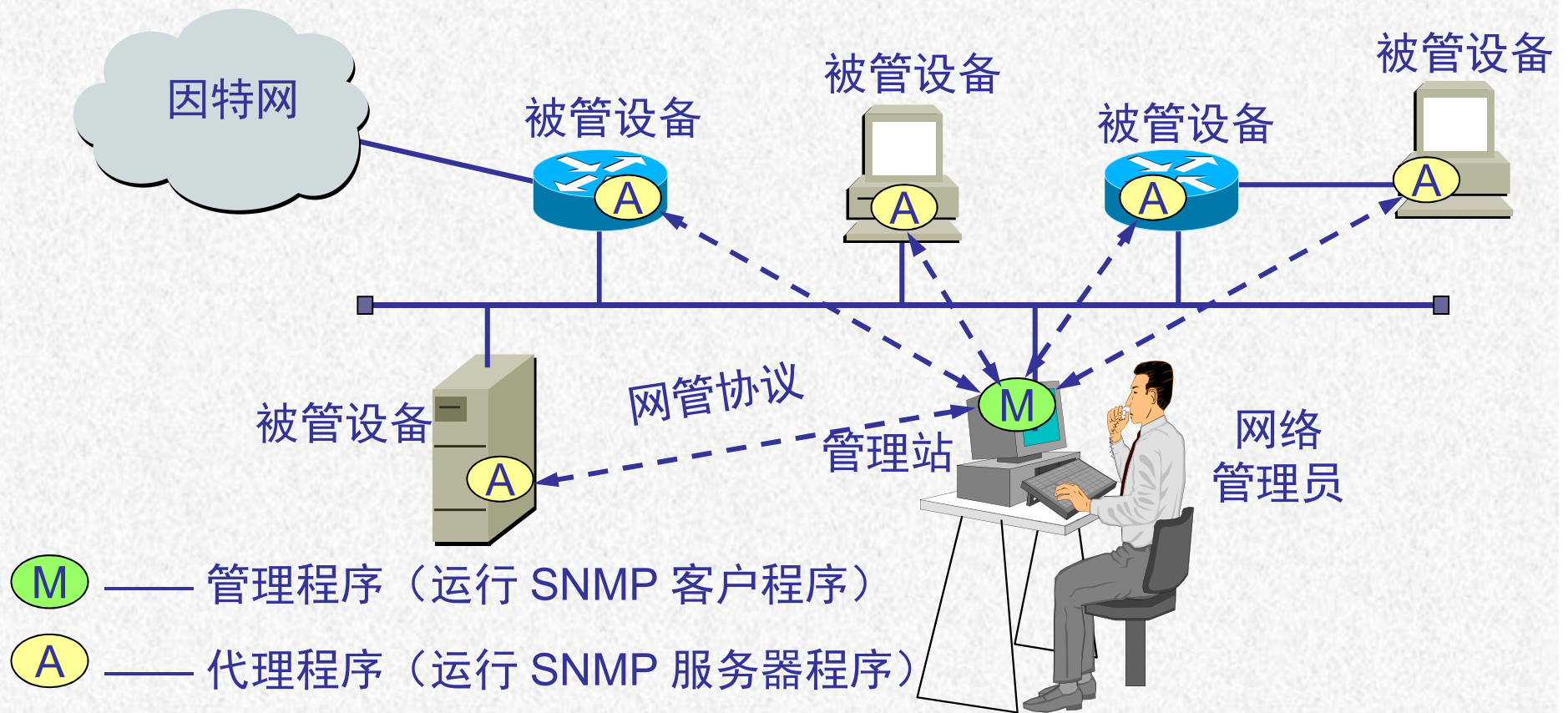
■ 网络管理

- 对网络资源进行监视、测试、配置、分析、评价和控制
- 满足网络的一些需求，如实时运行性能，服务质量等。

■ 网络管理常简称为网管。

- 网络管理并不是指对网络进行行政上的管理。

网络管理的一般模型



网络管理模型中的主要构件

- **管理站**也常称为**网络运行中心 NOC** (Network Operations Center)，是网络管理系统的核心。
- **管理程序**在运行时就成为**管理进程**。
- 管理站（硬件）或管理程序（软件）都可称为**管理者(manager)**。
- **网络管理员(administrator)** 指的是人。
 - 大型网络往往实行多级管理，因而有多个管理者，而一个管理者一般只管理本地网络的设备。

被管对象(Managed Object)。

- 网络的每一个被管设备中可能有多个被管对象。
 - 被管设备有时可称为网络元素或网元。
 - 在被管设备中也会有一些不能被管的对象。

代理(agent)

- 在每一个被管设备中都要运行一个程序以便和管理站中的管理程序进行通信。这些运行着的程序叫做**网络管理代理程序**，或简称为**代理**。
- 代理程序在管理程序的命令和控制下在被管设备上采取本地的行动。

网络管理协议

- **网络管理协议**，简称为**网管协议**。
 - 网管协议就是管理程序和代理程序之间进行通信的规则。
 - 网络管理员利用网管协议通过管理站对网络中的被管设备进行**管理**。

客户服务器方式

- 管理程序和代理程序按**客户服务器方式**工作。
- 管理程序运行 **SNMP 客户程序**，向某个代理程序发出请求（或命令），代理程序运行 **SNMP 服务器程序**，返回响应（或执行某个动作）。
- 在网管系统中往往是一个（或少数几个）客户程序与很多的服务器程序进行交互。

6.7.2 简单网络管理协议 SNMP 概述

- 网络管理的基本原理：

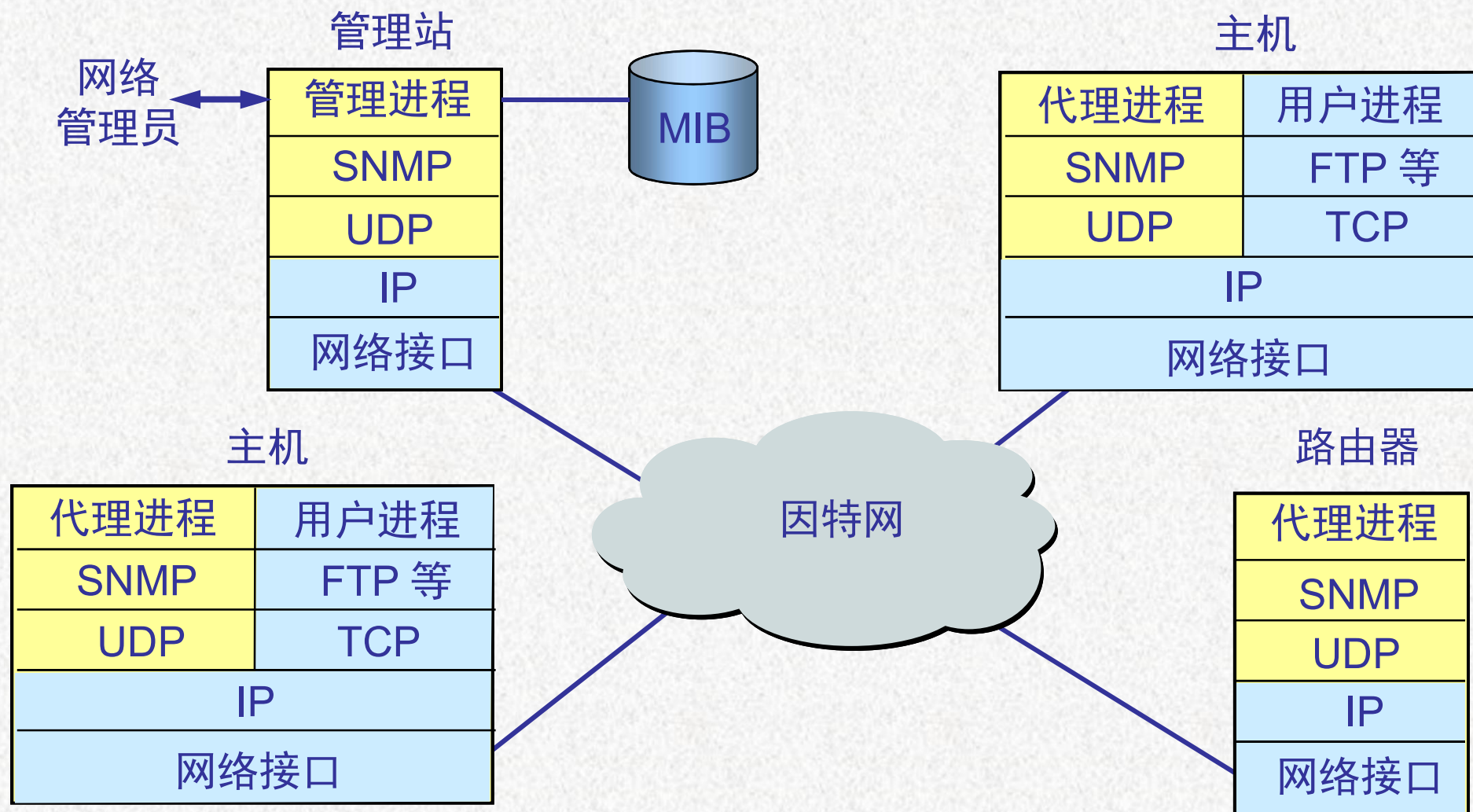
若要管理某个对象，就必然会给该对象添加一些软件或硬件，但这种“添加”必须对原有对象的影响尽量小些。

- **SNMP** 发布于 1988 年。IETF 在 1990 年制订的网管标准 **SNMP** 是因特网的正式标准。
- 以后有了新版本 **SNMPv2** 和 **SNMPv3**，因此原来的 **SNMP** 又称为 **SNMPv1**。

SNMP 的指导思想

- **SNMP** 最重要的指导思想就是要尽可能简单。
- **SNMP** 的基本功能包括监视网络性能、检测分析网络差错和配置网络设备等。
- 在网络正常工作时，**SNMP** 可实现统计、配置、和测试等功能。当网络出故障时，可实现各种差错检测和恢复功能。
- 虽然 **SNMP** 是在 **TCP/IP** 基础上的网络管理协议，但也可扩展到其他类型的网络设备上。

SNMP 的典型配置



SNMP 的管理站和委托代理

- 整个系统必须有一个**管理站**。
 - 管理进程和代理进程利用 **SNMP** 报文进行通信，而 **SNMP** 报文又使用 **UDP** 来传送。
- 若网络元素使用的不是 **SNMP** 而是另一种网络管理协议，**SNMP** 协议就无法控制该网络元素。
 - 这时可使用**委托代理**(proxy agent)。委托代理能提供如协议转换和过滤操作等功能对被管对象进行管理。

SNMP 的网络管理

由三个部分组成

- **SNMP 本身**
- **管理信息结构 SMI**
(Structure of Management Information)
- **管理信息库 MIB**
(Management Information Base)。

SNMP

- **SNMP** 定义了管理站和代理之间所交换的分组格式。
 - 所交换的分组包含各代理中的对象（变量）名及其状态（值）。
 - **SNMP** 负责读取和改变这些数值。

SMI(Structure of Management Information)

- **SMI** 定义了命名对象和定义对象类型（包括范围和长度）的通用规则，以及把对象和对象的值进行编码的规则。
 - 这样做是为了确保网络管理数据的语法和语义的无二义性。但从 **SMI** 的名称并不能看出它的功能。
 - **SMI** 并不定义一个实体应管理的对象数目，也不定义被管对象名以及对象名及其值之间的关联。

MIB (Management Information Base)

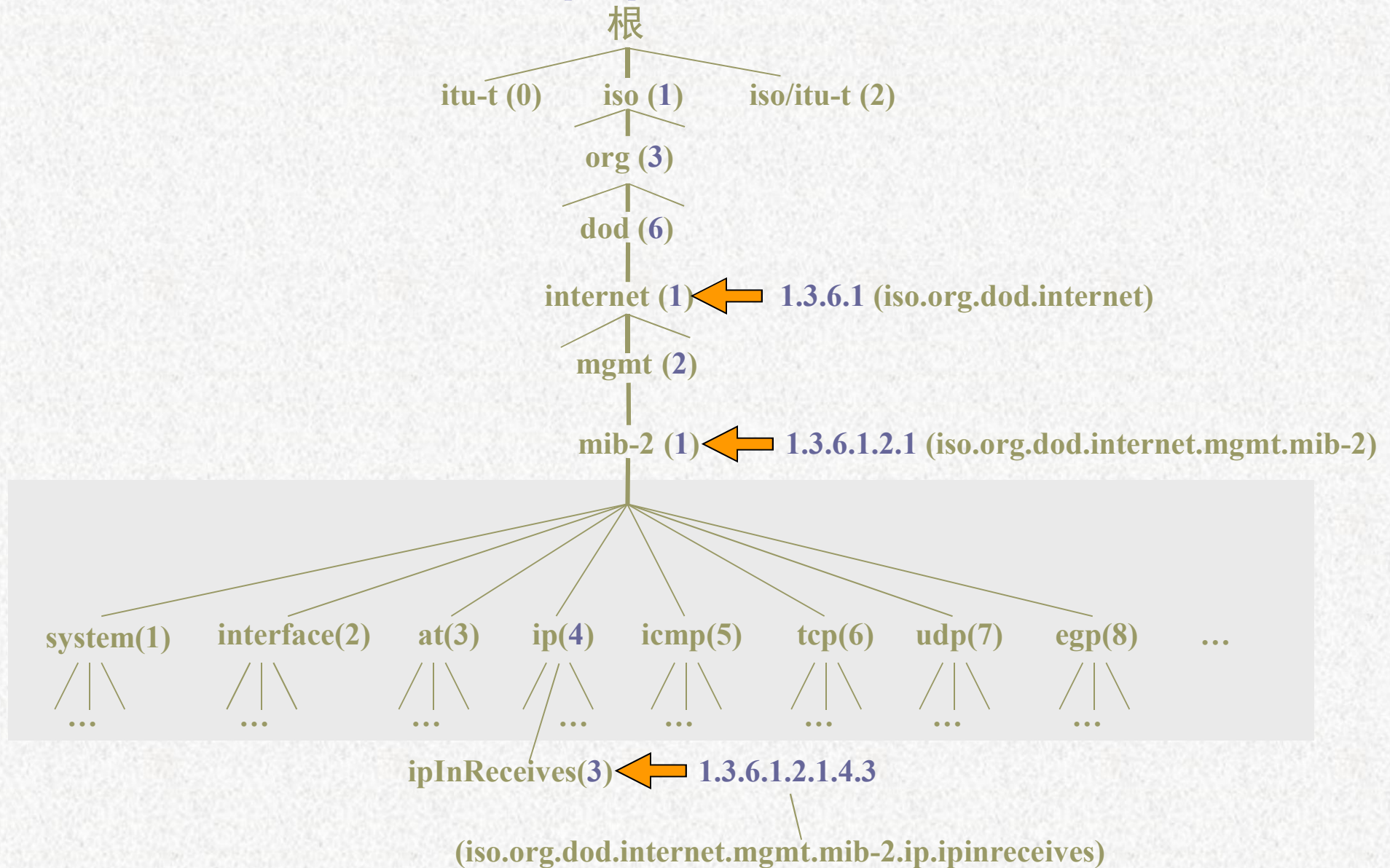
- **MIB** 在被管理的实体中创建了命名对象，并规定了其类型。
 - **SMI**建立规则
 - **MIB**变量说明
 - **SNMP**完成网管动作

6.7.2 管理信息结构 SMI

■ SMI 的功能，制定以下相关规则：

- ✓ (1) 对象命名规则：被管对象应怎样命名；
- ✓ (2) 对象类型描述：用来存储被管对象的数据类型有哪些种；
- ✓ (3) 数据编码规则：在网络上传送的管理数据应如何编码。

SMI 规定所有被管对象必须在命名树上



SMI 使用 ASN.1

- **SMI 标准指明了所有的 MIB 变量必须使用抽象语法记法 1（ASN.1）来定义。**
 - **ASN.1 是一种数据类型描述语言，具有类似于面向对象程序设计语言中所提供的类型机制。**
 - **ASN.1 可定义任意复杂结构的数据类型，而不同的数据类型之间还可以有继承关系。**
 - **抽象语法只描述数据的结构形式且与具体的编码格式无关，同时也不涉及这些数据结构在计算机内如何存放。**
- **SMI 把数据类型分为两大类：简单类型和结构化类型。**

基本编码规则 BER

(Basic Encoding Rule)

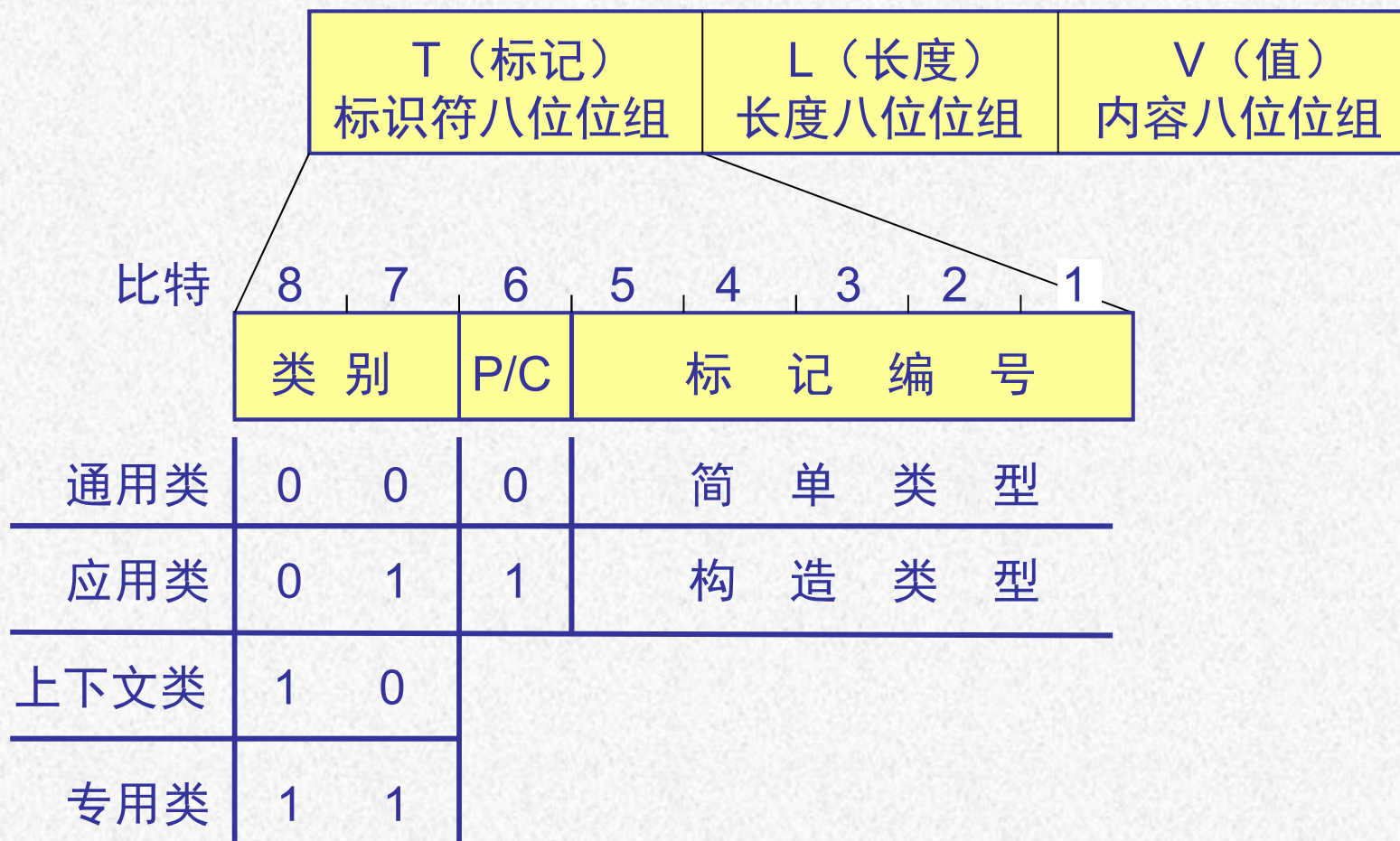
- **ISO** 在制订 **ASN.1** 语言的同时也为它定义了一种标准的编码方案，即**基本编码规则 BER**。
 - **BER** 指明了每种数据类型中每个数据的值的表示。
 - 发送端用 **BER** 编码，可将用 **ASN.1** 所表述的报文转换成唯一的比特序列。
 - 接收端用 **BER** 进行解码，得到该比特序列所表示的 **ASN.1** 报文。

用 TLV 方法进行编码

- 把各种数据元素表示为以下三个字段组成的八位位组序列：

- (1) T 字段，即标识符八位位组(identifier octet)，用于标识**标记**。
- (2) L 字段，即长度用八位位组(length octet)，用于标识后面 V 字段的**长度**。
- (3) V 字段，即内容八位位组(content octet)，用于标识数据元素的**值**。

用 TLV 方法进行编码



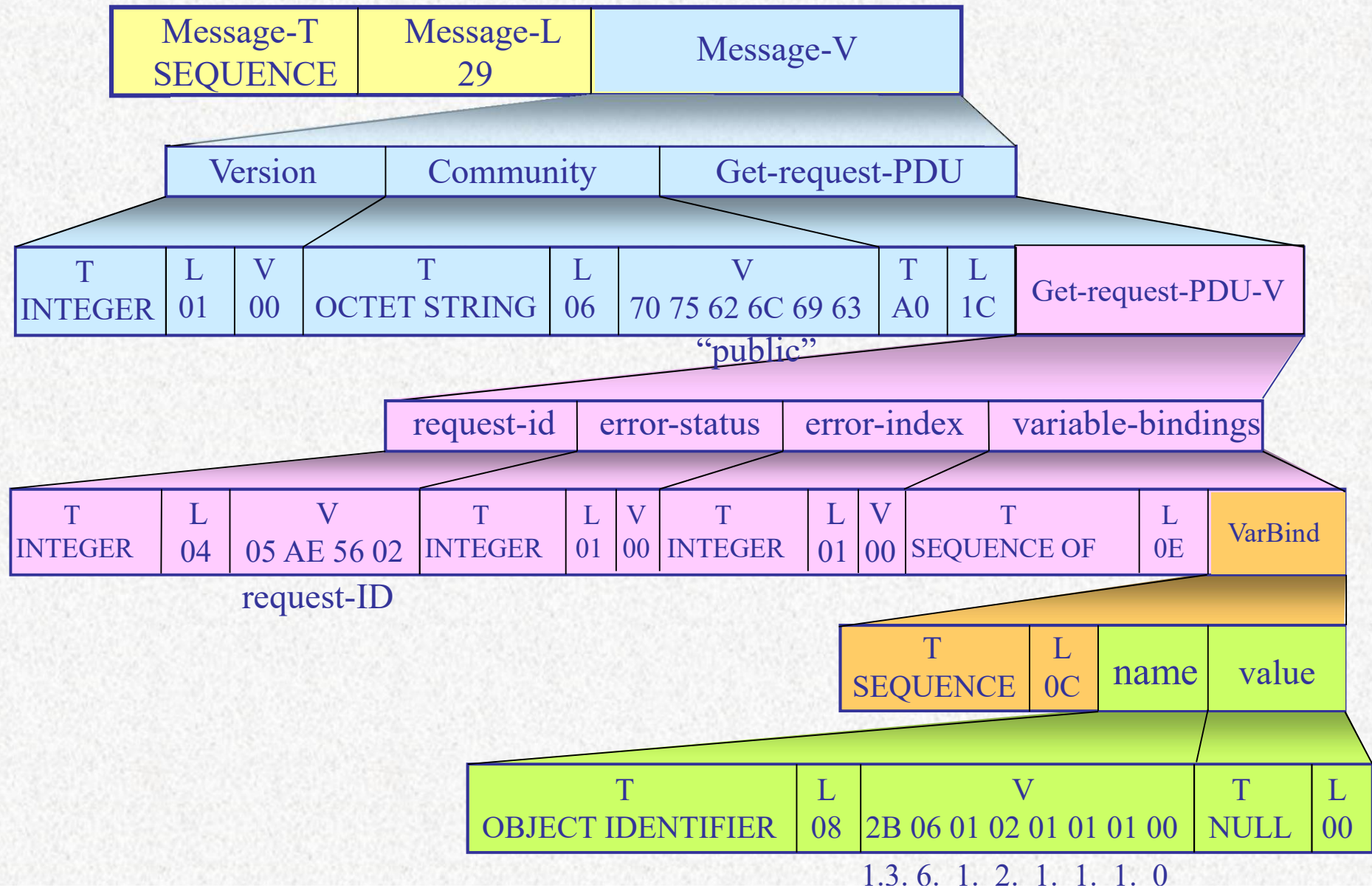
TLV 中的 V 字段

定义数据的值

- 例如，INTEGER 15，其 T 字段是**02**，INTEGER 类型要用 4 字节编码。最后得出 TLV 编码为 **02 04 00 00 00 0F**。
- 又如 IPAddress 192.1.2.3，其 T 字段是 40，V 字段需要 4 字节表示，因此得出 IPAddress 192.1.2.3 的 TLV 编码是 40 **04 C0 01 02 03**。

SNMP 的 Get-request 报文

ASN.1 编码



6.7.3 管理信息库 MIB

(Management Information Base)

- 被管对象必须维持可供管理程序读写的若干控制和状态信息。这些信息总称为管理信息库 MIB 。
- 管理程序使用 MIB 中这些信息的值对网络进行管理（如读取或重新设置这些值）。

6.7.4 SNMP 的 协议数据单元和报文

- **SNMP**的操作只有两种基本的管理功能，即：
 - “**读**”操作，用 **get** 报文来检测各被管对象的状况；
 - “**写**”操作，用 **set** 报文来改变各被管对象的状况。

SNMP 的探测操作

- **探测操作**——SNMP 管理进程定时向被管理设备周期性地发送探测信息。
- 探测的好处是：
 - 可使系统相对简单。
 - 能限制通过网络所产生的管理信息的通信量。
 - 但探测所能管理的设备数目不能太多。
 - 如探测频繁而并未得到有用的报告，则通信线路和计算机的 CPU 周期就被浪费了。

陷阱(trap)

- **SNMP** 不是完全的探询协议，它允许不经过询问就能发送某些信息。这种信息称为**陷阱**，表示它能够捕捉“事件”。
- 当被管对象的代理检测到有事件发生时，就检查其门限值。
- 代理只向管理进程报告达到某些门限值的事件（即**过滤**）。
 - 仅在严重事件发生时才发送陷阱；
 - 陷阱信息很简单且所需字节数很少。

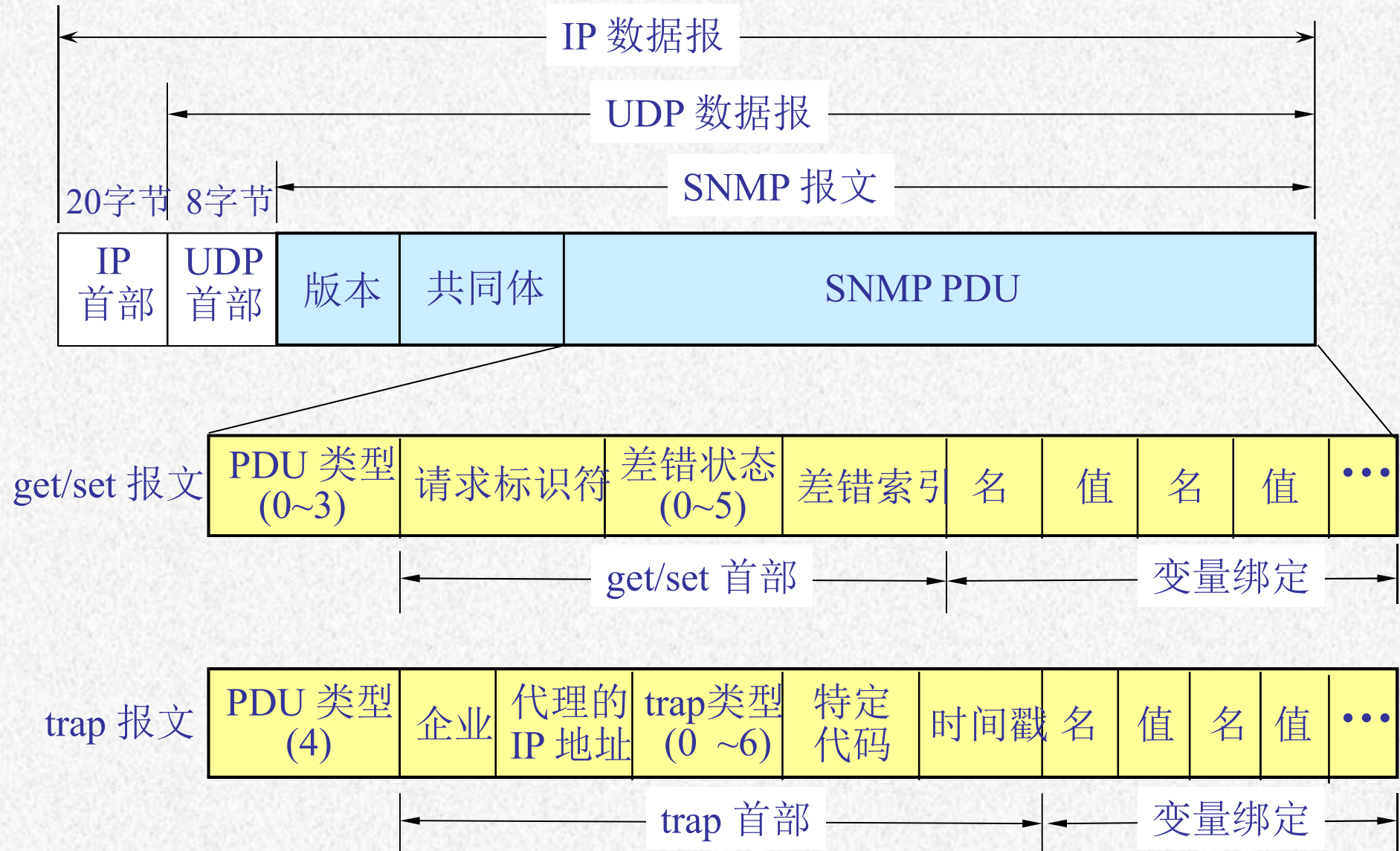
SNMP 使用无连接的 UDP

- **SNMP 使用无连接的 UDP**，因此在网络上传送 **SNMP** 报文的开销较小。但 **UDP** 不保证可靠交付。
 - 在运行代理程序的**服务器端**用熟知端口 **161** 来接收 **get** 或 **set** 报文和发送响应报文（与熟知端口通信的客户端使用临时端口）。
 - 运行管理程序的**客户端**则使用熟知端口 **162** 来接收来自各代理的 **trap** 报文。

SNMPv1 定义的 协议数据单元类型

PDU 编号	PDU名称	用途
0	GetRequest	用来查询一个或一组变量的值
1	GetNextRequest	允许在 MIB 树上读取下一个变量， 此操作可反复进行
2	Reponse	代理向管理者或管理者向管理者发送 响应
3	SetRequest	对一个或多个变量值进行设置
5	GetBulkRequest	管理者从代理读取大数据块的值
6	InformRequest	管理者从另一管理者读取代理的变量
7	SNMPv2Trap	代理向管理者报告异常事件
8	<u>Report</u> t	管理者之间报告某些差错

SNMPv1 的报文格式



SNMP 报文的组成

- 版本
- 共同体(community)
- **SNMP PDU**——由三个部分组成
 - PDU 类型
 - get/set 首部或 trap 首部
 - 变量绑定(variable-bindings)（变量绑定指明一个或多个变量的名和对应的值）。

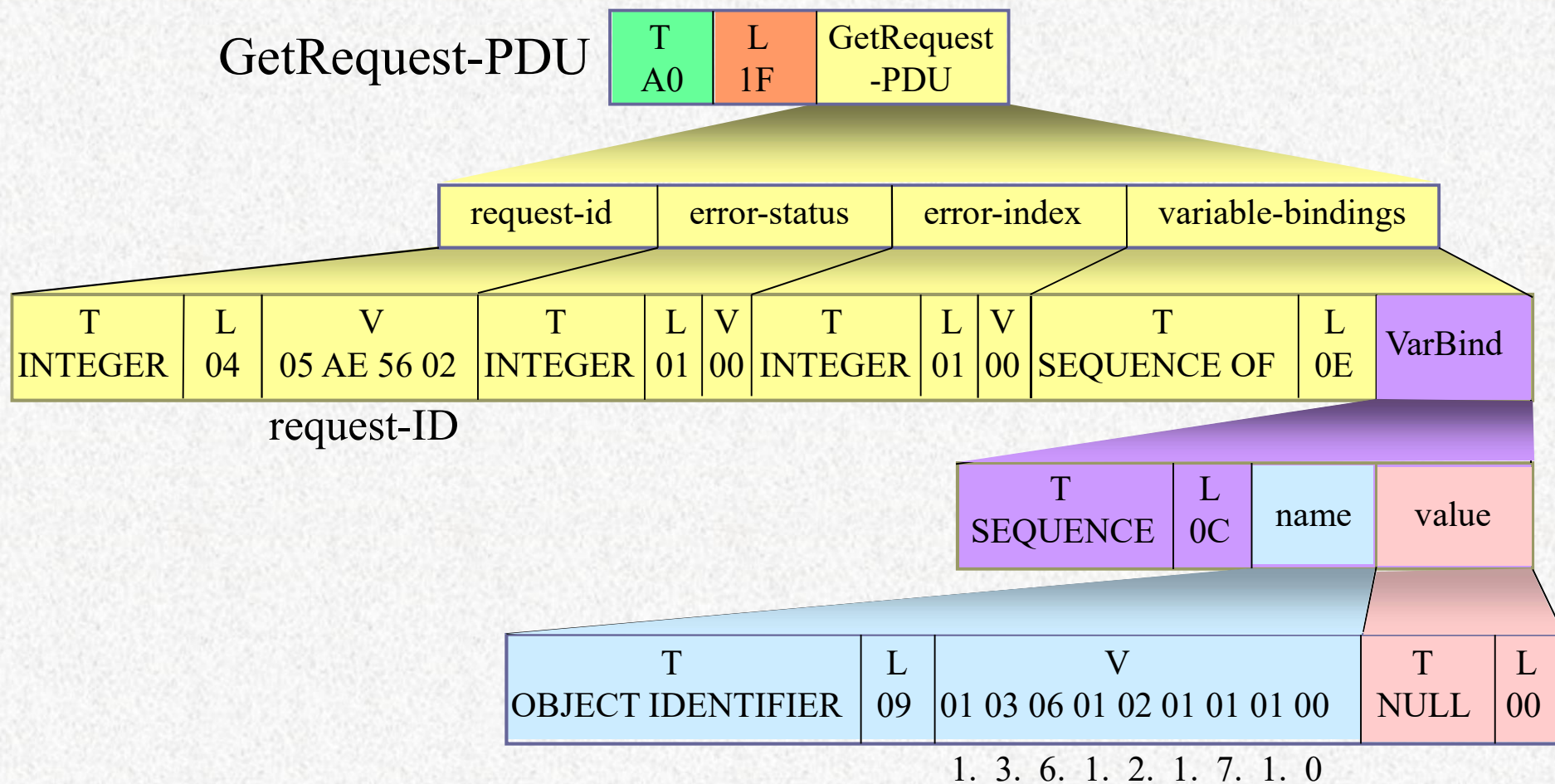
get/set 首部的字段

- 请求标识符(request ID)
- 差错状态(error status)
- 差错索引(error index)

trap 首部的字段

- 企业(enterprise)
- 陷阱类型
- 特定代码(specific-code)
- 时间戳(timestamp)

Get-request 报文 ASN.1 编码



6.7.6 SNMPv2 和 SNMPv3

SNMPv1 的主要缺点是：

- (1) 不能有效地传送大块的数据**
- (2) 不能将网络管理的功能分散化**
- (3) 安全性不够好**

SNMPv2

- 1996 年发布 IETF 发布了 8 个 SNMPv2 文档 [RFC 1901~1908]。但 SNMPv2 在安全方面的设计过分复杂，使得有些人不愿意接受它。
- SNMPv2 增加了 **get-bulk-request** 命令，可一次从路由器的路由表中读取许多行的信息。
- SNMPv2 的 **get** 命令允许返回部分的变量值，这就提高了效率，减少了网络上的通信量。
- SNMPv2 采用了分散化的管理方法。在一个网络中可以有多多个顶级管理站，叫做管理服务器。

SNMPv2

- 增加了一个 **inform** 命令和一个管理进程到管理进程的 **MIB (manager-to-manager MIB)**。
- 使用这种 **inform** 命令可以使管理进程之间互相传送有关的事件信息而不需要经过请求。这样的信息则定义在管理进程到管理进程的 **MIB** 中。

SNMPv3

- 1998 年 1 月 IETF 发表了 SNMPv3 的有关文档[RFC 2271-2275]。
- 仅隔 15 个月后就更新为[RFC 2571-2575]。
- SNMPv3 最大的改进就是安全特性。也就是说，只有被授权的人员才有资格执行网络管理的功能（如关闭某一条链路）和读取有关网络管理的信息（如读取一个配置文件的内容）。

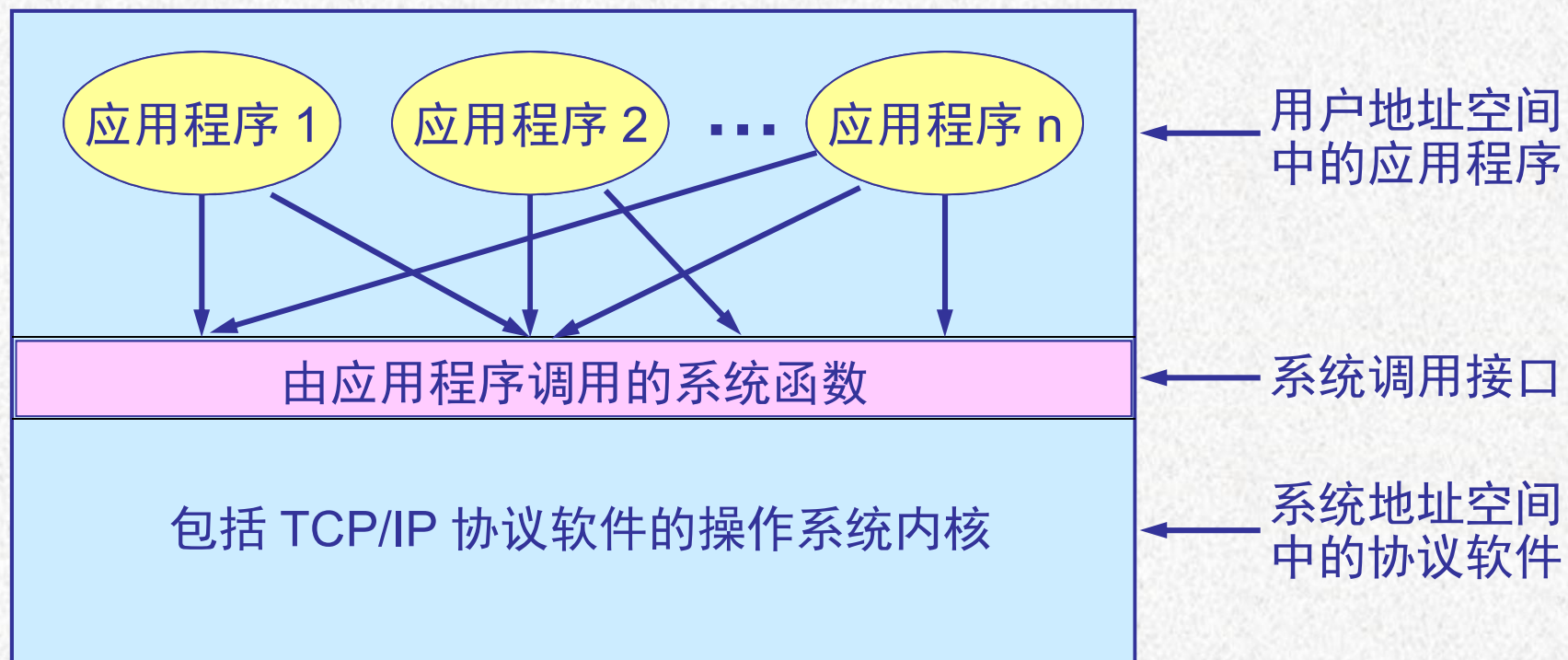
6.8 应用进程跨越网络的通信

- 6.8.1 系统调用和应用编程接口
- 6.8.2 几种常用的系统调用

6.8.1 系统调用和应用编程接口

- 大多数操作系统使用**系统调用**(system call)的机制在应用程序和操作系统之间传递控制权。
- 对程序员来说，每一个系统调用和一般程序设计中的函数调用非常相似，只是系统调用是将控制权传递给了操作系统。

多个应用进程 使用系统调用的机制



应用编程接口 API

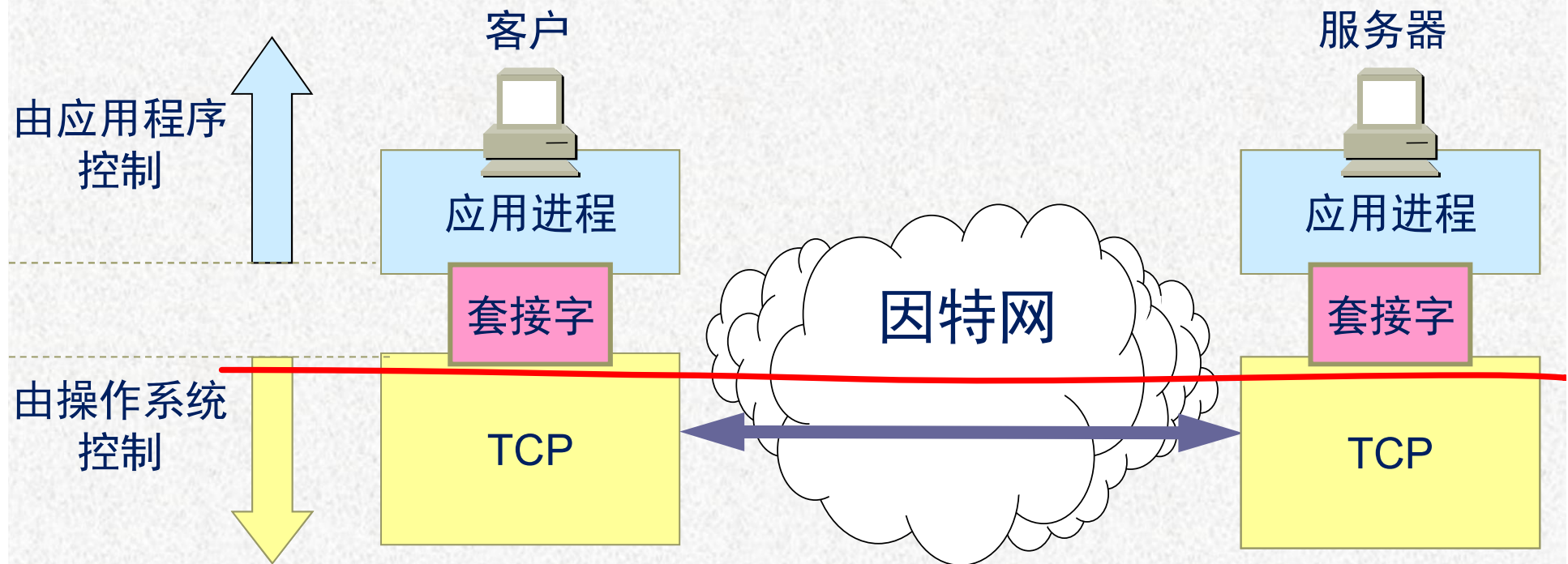
(Application Programming Interface)

- 当某个应用进程启动系统调用时，控制权就从应用进程传递给了**系统调用接口**。
- 此接口再将控制权传递给计算机的操作系统。操作系统将此调用转给某个内部过程，并执行所请求的操作。
- 内部过程一旦执行完毕，控制权就又通过系统调用接口返回给应用进程。
- 系统调用接口实际上就是应用进程的控制权和操作系统的控制权进行转换的一个接口，即**应用编程接口 API**。

几种应用编程接口 API

- **Berkeley UNIX** 操作系统定义了一种 **API**，它又称为套接字接口(socket interface)。
- 微软公司在其操作系统中采用了套接字接口 **API**，形成了一个稍有不同的 **API**，并称之为 **Windows Socket**。
- **AT&T** 为其 **UNIX** 系统 **V** 定义了一种 **API**，简写为 **TLI** (Transport Layer Interface)。

应用进程通过套接字接入到网络



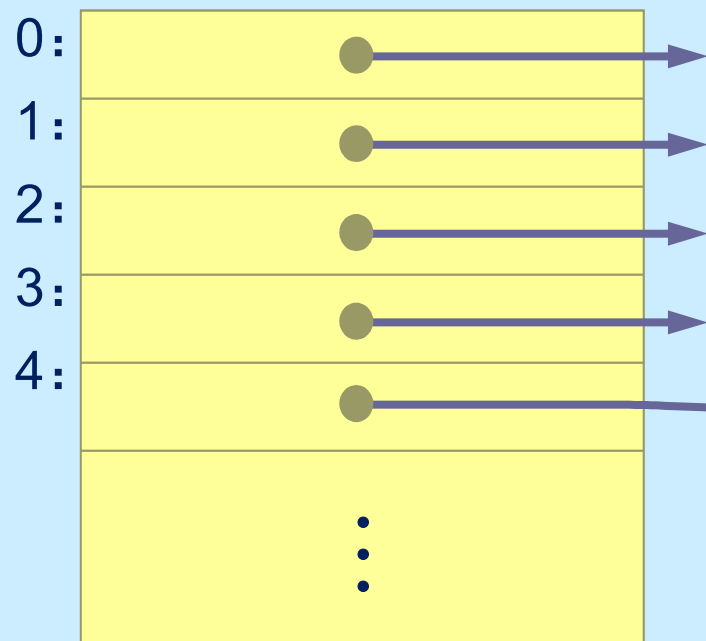
套接字的作用

- 当应用进程需要使用网络进行通信时就发出系统调用，请求操作系统为其创建“套接字”，以便把网络通信所需要的系统资源分配给该应用进程。
- 操作系统为这些资源的总和用一个叫做套接字描述符的号码来表示，并把此号码返回给应用进程。应用进程所进行的网络操作都必须使用这个号码。
- 通信完毕后，应用进程通过一个关闭套接字的系统调用通知操作系统回收与该“号码”相关的所有资源。

调用 socket 创建套接字

操作系统

套接字描述符表
(每一个进程一个描述符)



套接字的数据结构

协议族: PF_INET
服务: SOCK_STREAM
本地 IP 地址:
远地 IP 地址:
本地端口:
远地端口:
⋮

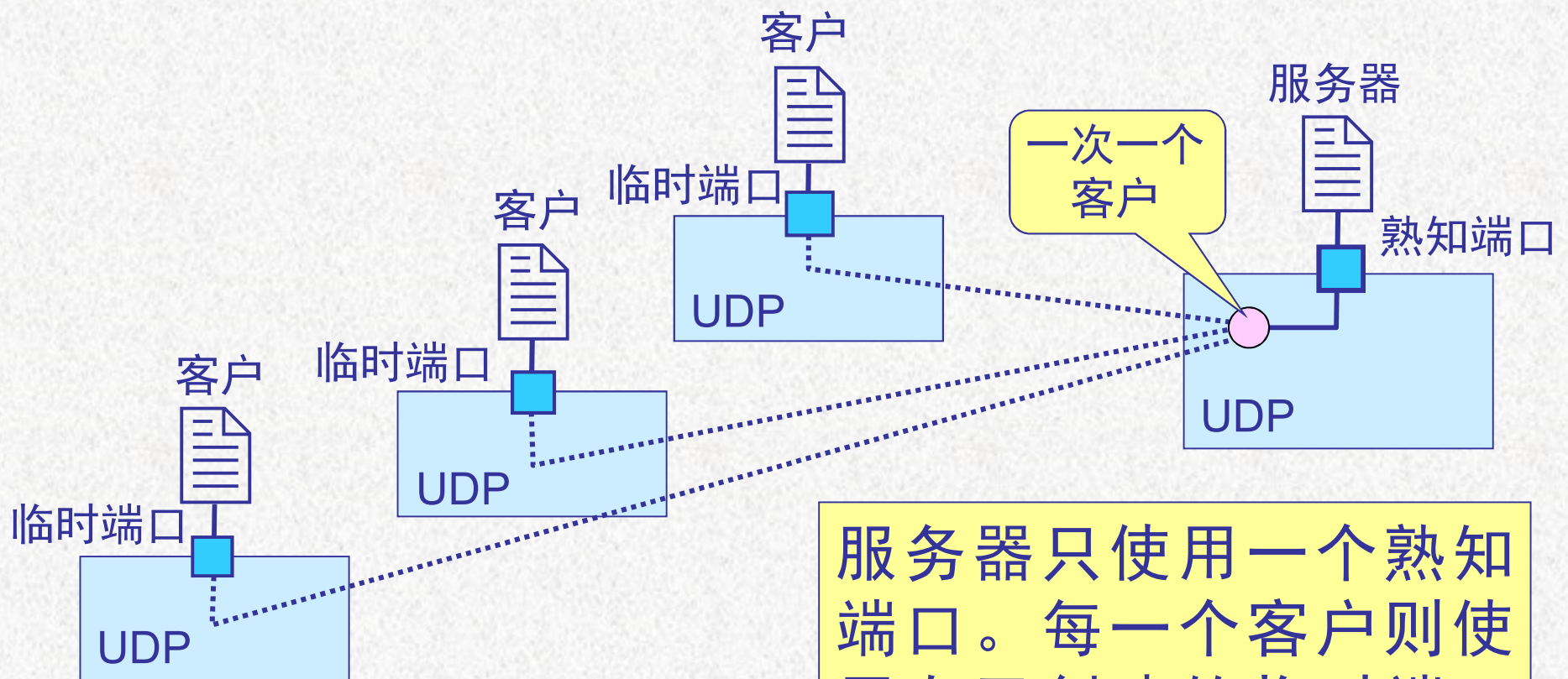
6.8.2 服务器的两种工作方式

- 服务器可工作在两种不同的方式：
- **循环方式(iterative mode)**——在计算机中一次只运行一个服务器进程。当有多个客户进程请求服务时，服务器进程就按请求的先后顺序**依次做出响应**。
- **并发方式(concurrent)**——在计算机中同时运行多个服务器进程，而每一个服务器进程都对某个特定的客户进程做出响应。

1. 无连接循环服务器

- 无连接 **UDP** 的服务器通常都工作在循环方式——一个服务器在同一时间只能向一个客户提供服务。
- 服务器收到客户的请求后，就发送 **UDP** 用户数据报响应该客户。但对其他客户发来的请求则暂时不予理睬，这些请求都在服务器端的队列中排队等候服务器的处理。
- 当服务器进程处理完毕一个请求时，就从队列中读取来自下一个客户的请求，然后继续处理。

无连接循环服务器的特点

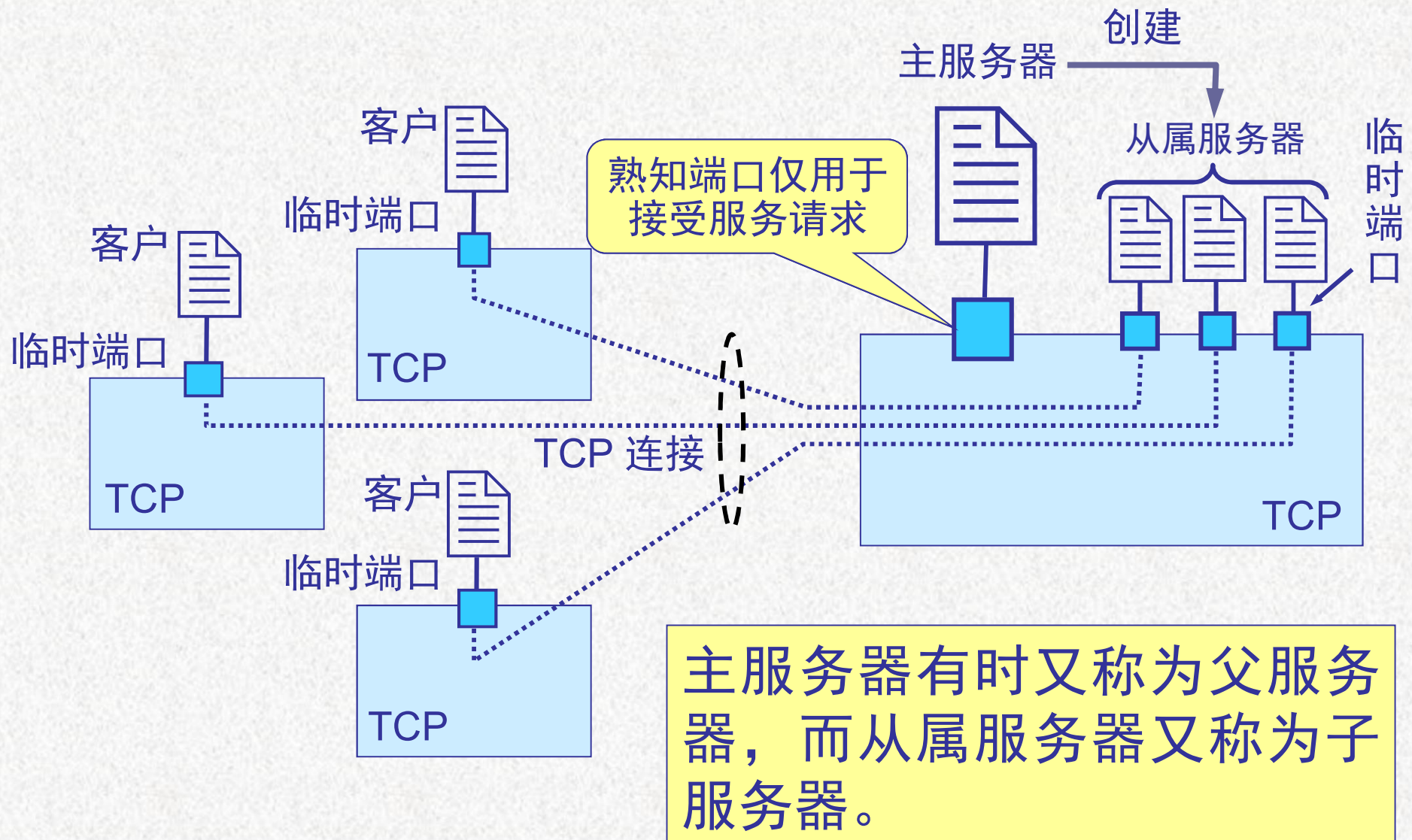


服务器只使用一个熟知端口。每一个客户则使用自己创建的临时端口（端口号自己设定）。

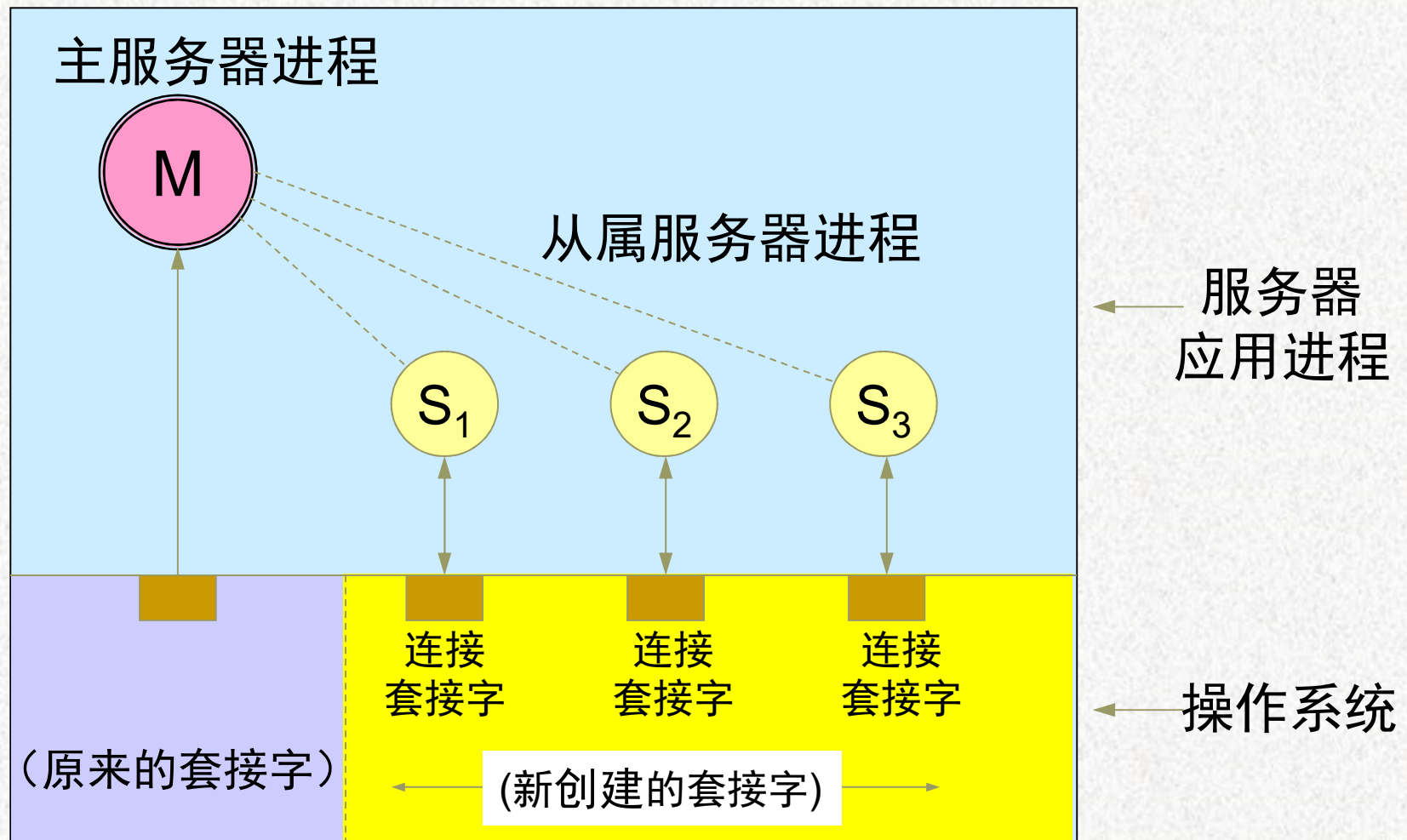
2. 面向连接并发服务器

- 服务器在同一时间可向多个客户提供服务。
- **TCP** 是面向连接的，服务器和多个客户之间必须建立多条 **TCP** 连接，而每一条 **TCP** 连接要在其数据传送完毕后才能释放。
- 使用 **TCP** 的服务器只能有一个熟知端口。主服务器在熟知端口等待客户发出的请求。
- 一旦收到客户的请求，立即创建一个**从属服务器**，并指明从属服务器使用临时端口和该客户建立 **TCP** 连接
- 然后主服务器继续在原来的熟知端口等待向其他客户提供服务。

面向连接并发服务器

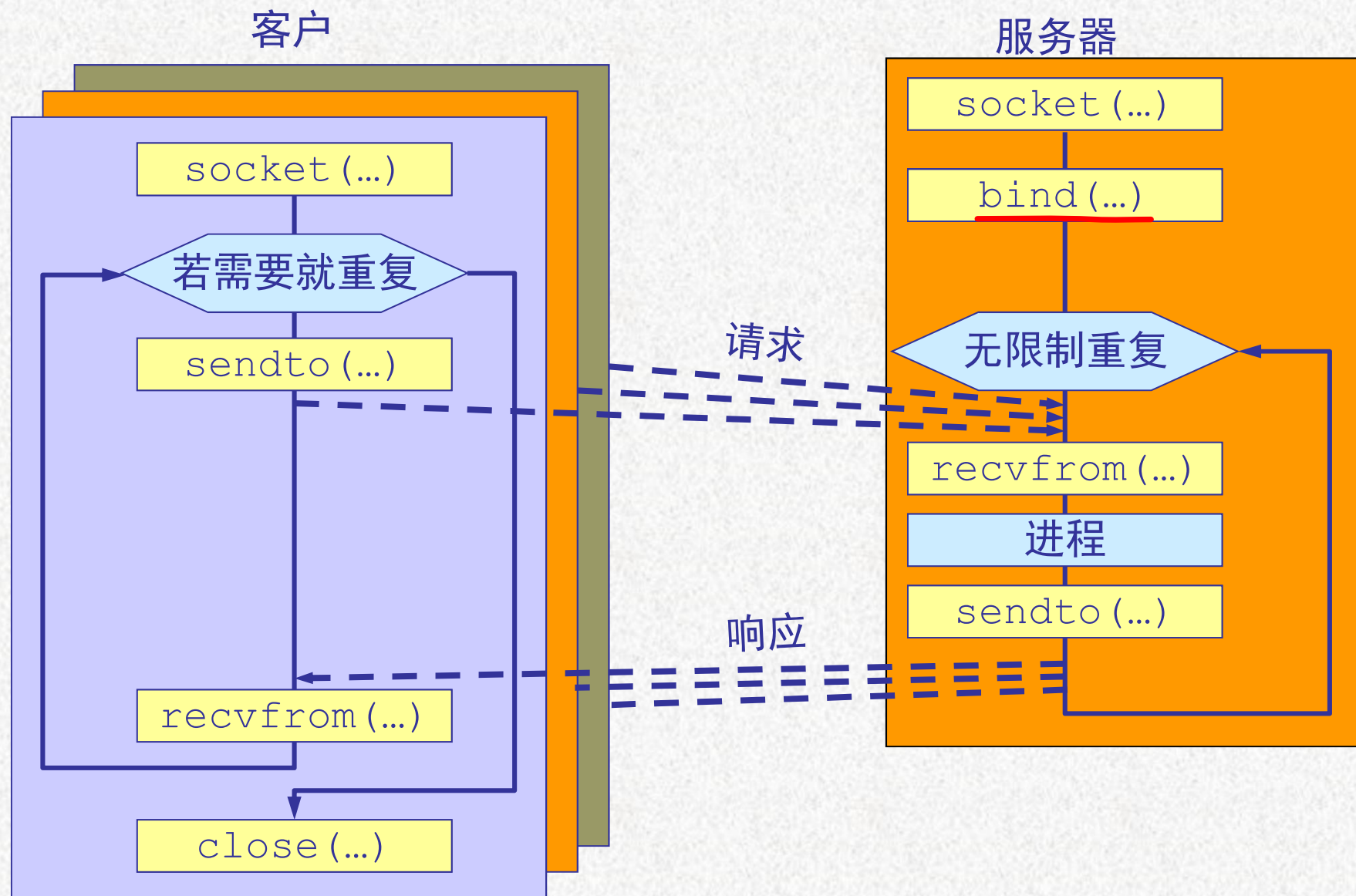


并发方式工作的服务器



6.8.3 进程通过系统调用接口进行通信的过程

1. 无连接循环服务器



2. 面向连接并发服务器

