



# 计算机网络（第 6 版）

---

## 第 6 章 应用层



# 第 6 章 应用层

---

## 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

## 6.2 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



# 第 6 章 应用层 ( 续 )

---

6.3 远程终端协议 TELNET

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

6.4.6 博客、微博和轻博



# 第 6 章 应用层 ( 续 )

---

## 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



# 第 6 章 应用层 ( 续 )

---

6.6 动态主机配置协议 DHCP

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

6.8 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



# 应用层协议的特点

---

- 每个应用层协议都是为了解决某一类应用问题，而问题的解决又往往是通过位于不同主机中的多个应用进程之间的通信和协同工作来完成的。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

- 许多应用层软件经常直接使用**域名系统** DNS (Domain Name System)，但计算机的用户只是间接而不是直接使用域名系统。
- 因特网采用层次结构的命名树作为主机的名字，并使用**分布式的**域名系统 DNS。
- 名字到 IP 地址的解析是由若干个域名服务器程序完成的。域名服务器程序在专设的结点上运行，运行该程序的机器称为**域名服务器**。



## 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。
- ( mail.ustb.edu.cn )





# 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。
- 域名中的“点”和点分十进制 IP 地址中的“点”并无一一对应的关系。点分十进制 IP 地址中一定是包含三个“点”，但每一个域名中“点”的数目则不一定正好是三个。



# 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 )



## 顶级域名 TLD （续）

---

(3) 基础结构域名 (infrastructure domain) :  
这种顶级域名只有一个，即 arpa，用于  
反向域名解析（IP 地址到域名的转换），  
因此又称为反向域名。

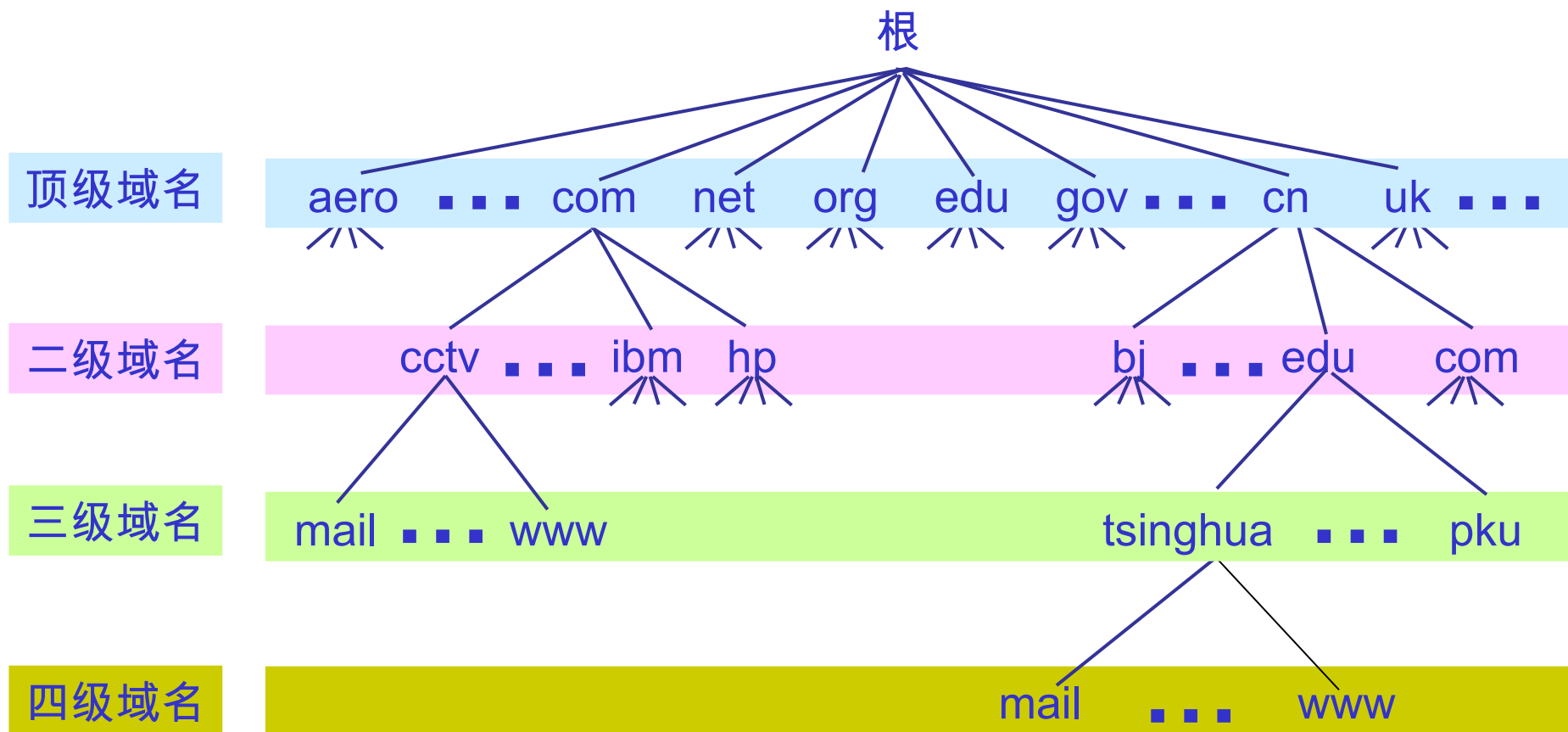


# 新增加了下列的通用顶级域名

---

- .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

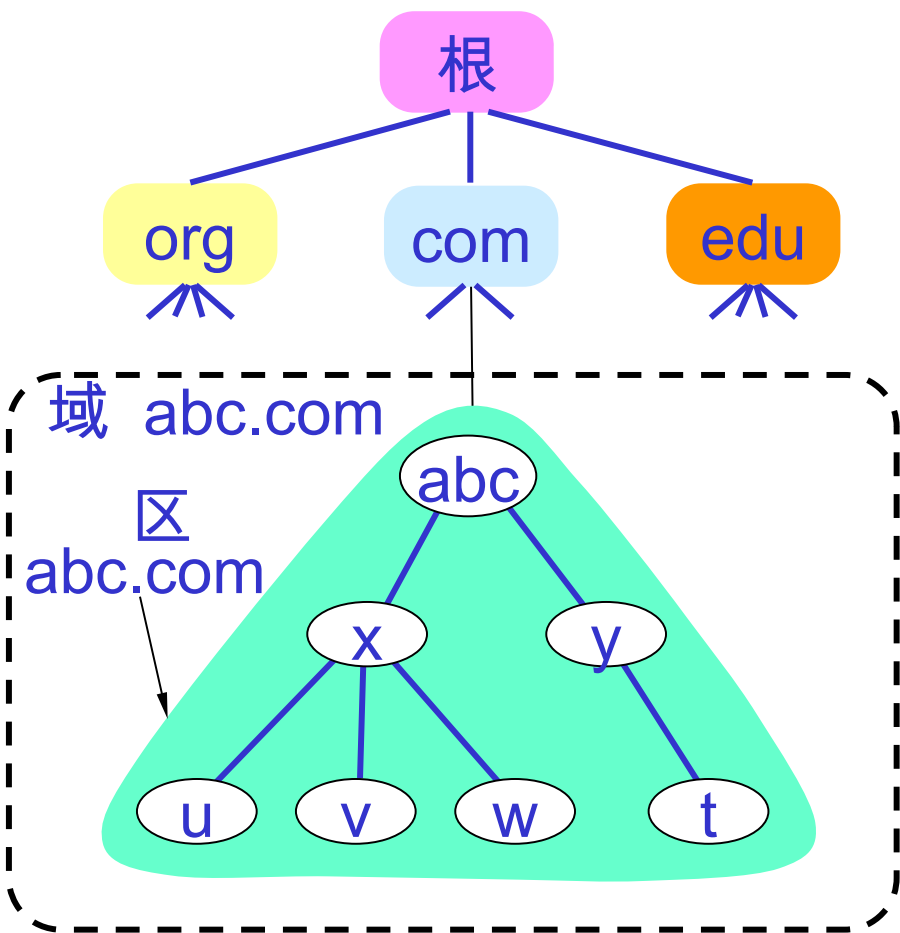




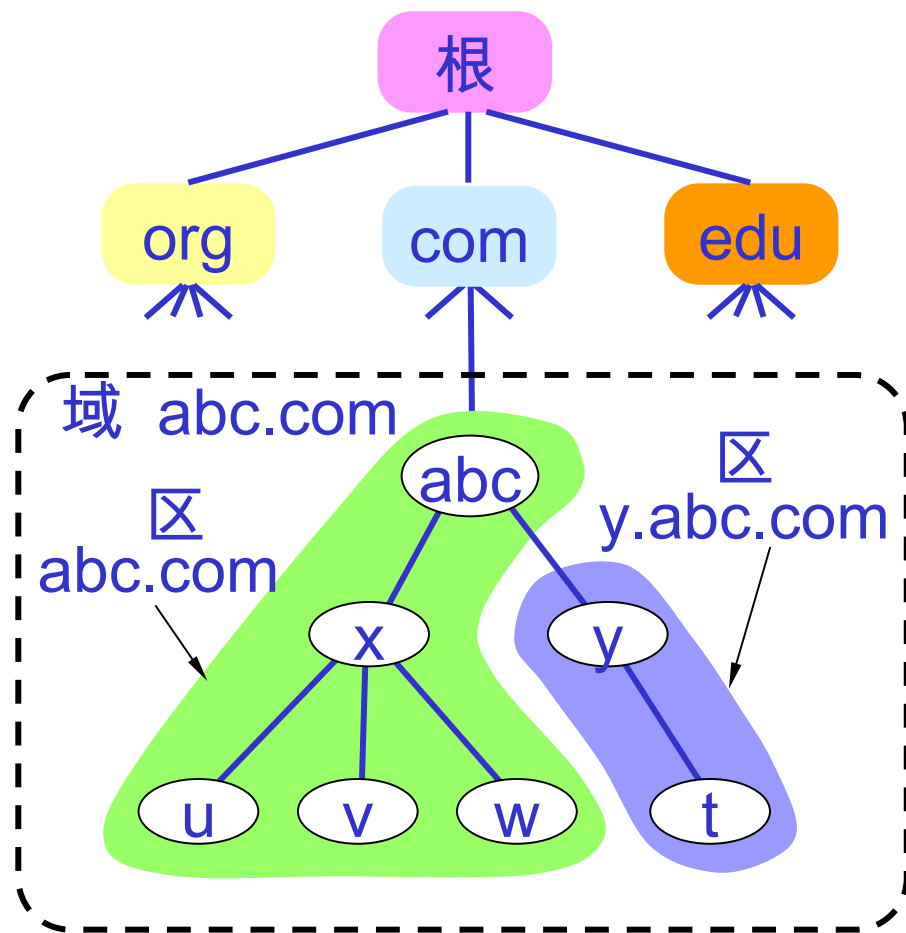
## 6.1.3 域名服务器

- 一个服务器所负责管辖的（或有权限的）范围叫做**区** (zone)。
- 各单位根据具体情况来划分自己管辖范围的区。但在一个区中的所有节点必须是能够连通的。
- 每一个区设置相应的**权限域名服务器**，用来保存该区中的所有主机的域名到 IP 地址的映射。
- DNS 服务器的管辖范围不是以“域”为单位，而是以“区”为单位。

# 区的不同划分方法举例



(a) 区 = 域



(b) 区 < 域

# 树状结构的 DNS 域名服务器

根域名服务器

根域名服务器

顶级域名服务器

org 域名服务器

com 域名服务器

edu 域名服务器 ...

权限域名服务器

abc.com  
域名服务器

y.abc.com  
域名服务器

abc 公司有两个  
权限域名服务器





# 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）。

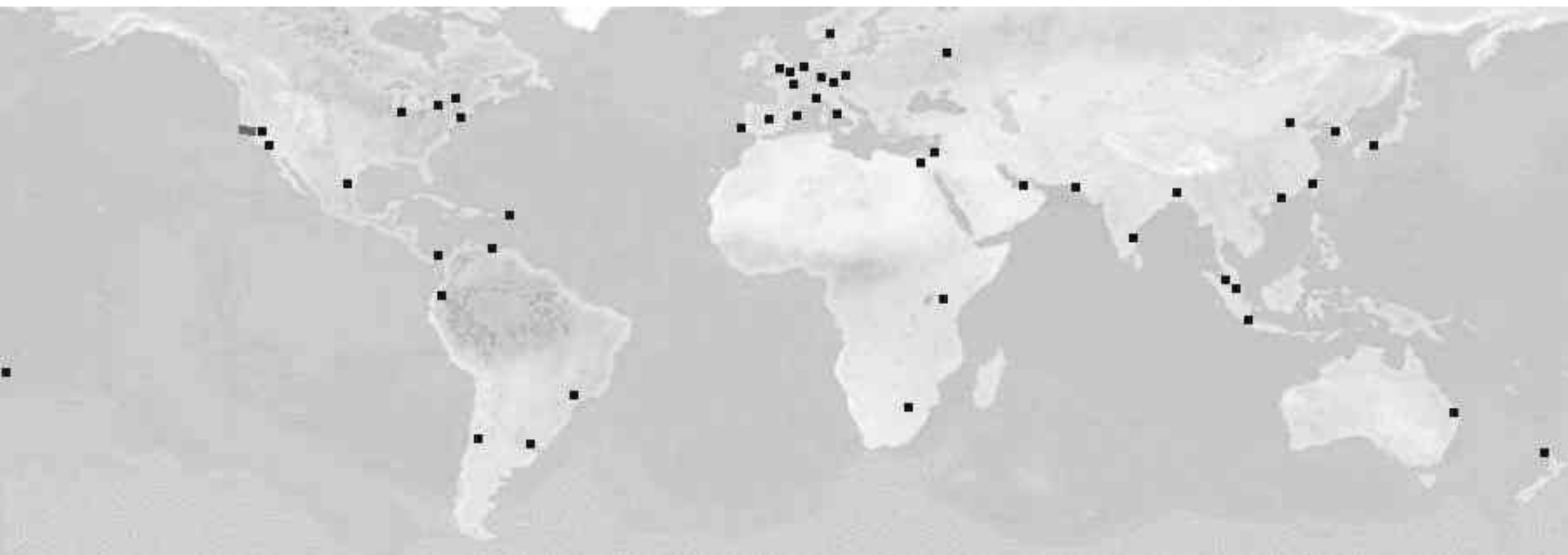


# 根域名服务器共有 13 套装置 ( 不是 13 个机器 )

---

- 这些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 的地點分布图 ( 2012 年 5 月 )



根域名服务器 f 共有 49 个机器

- 根域名服务器并不直接把域名转换成 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 )

- 这些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

---

- 这就是前面已经讲过的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。



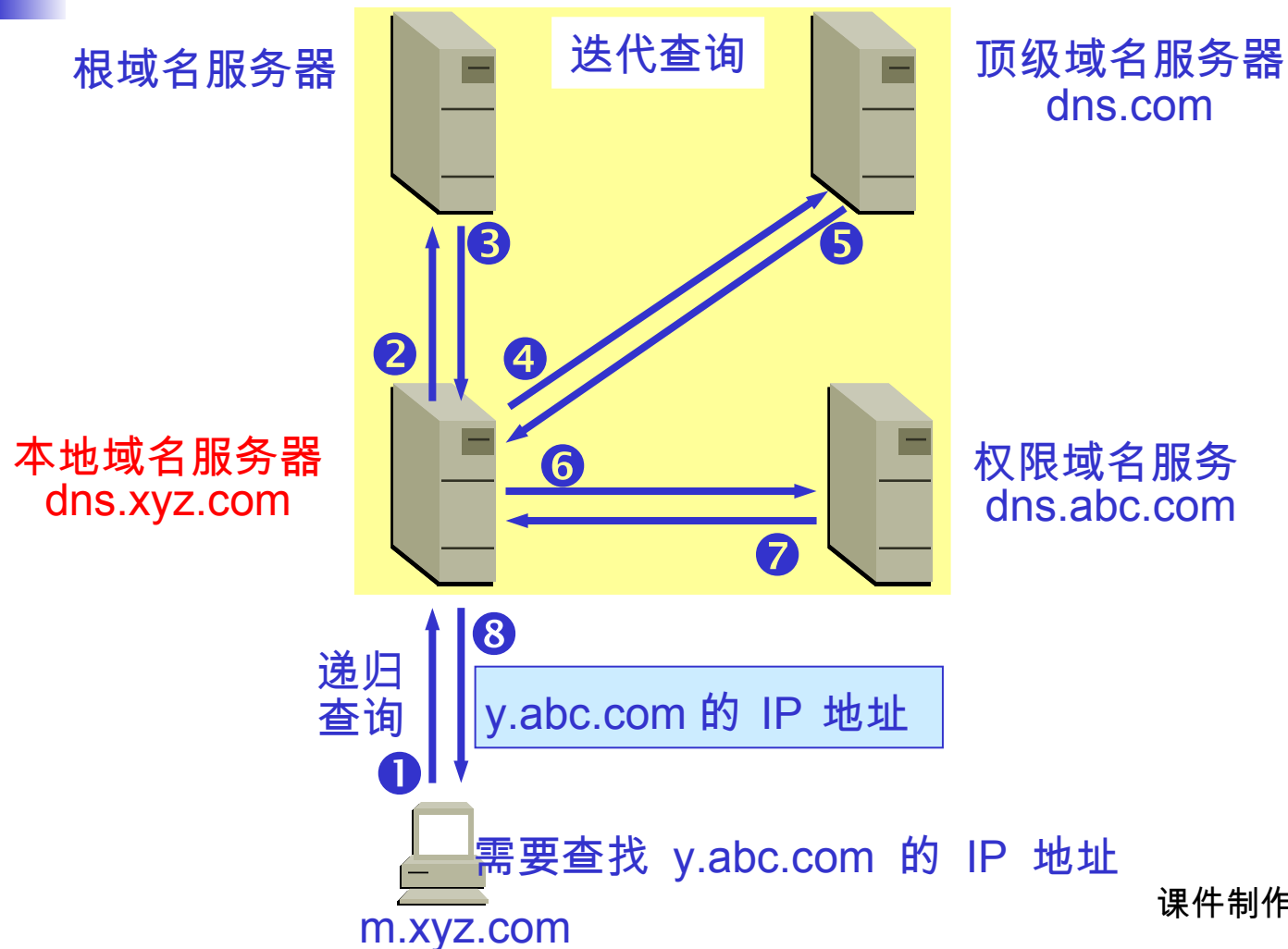


# 域名的解析过程

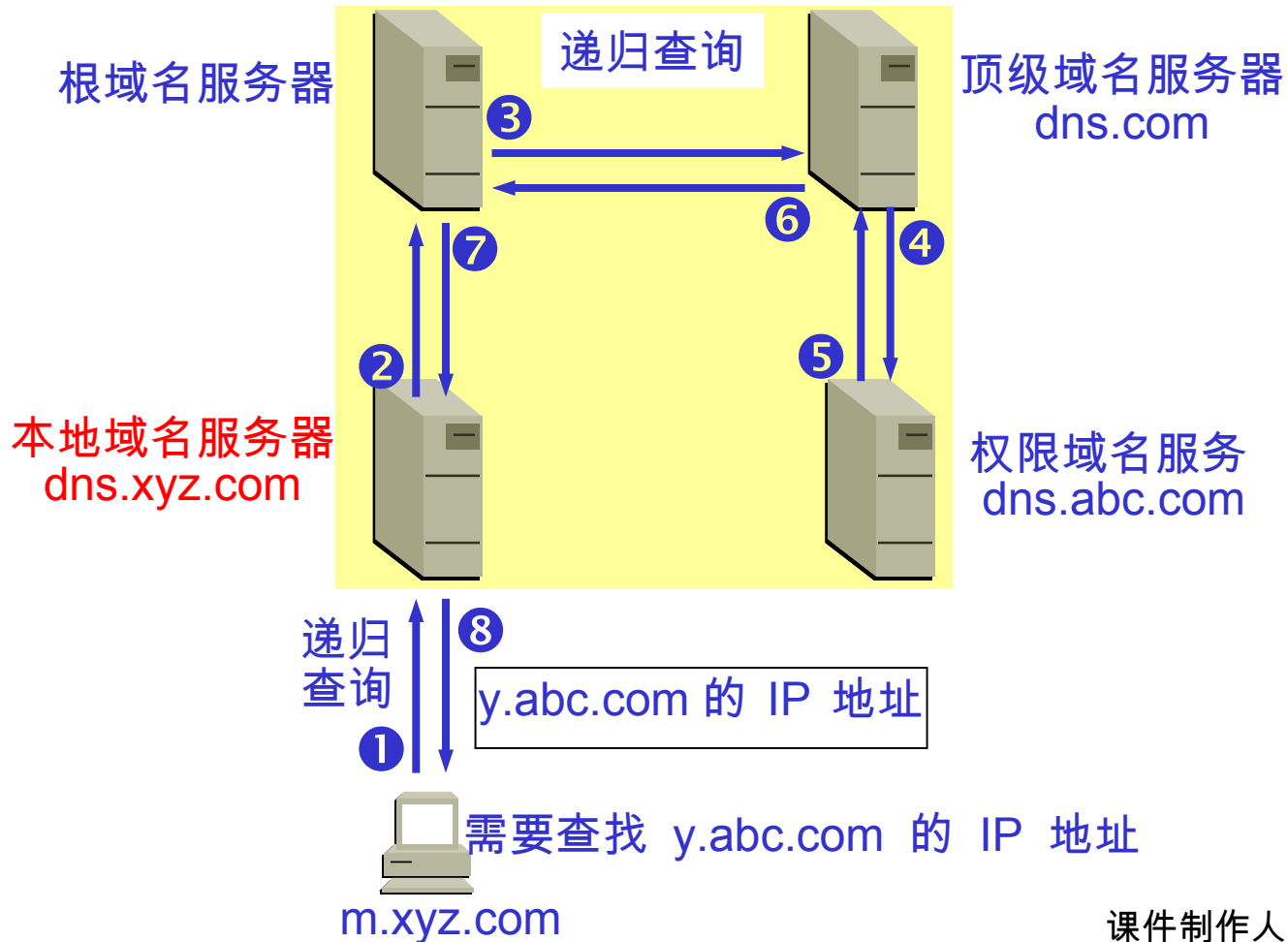
---

- 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

---

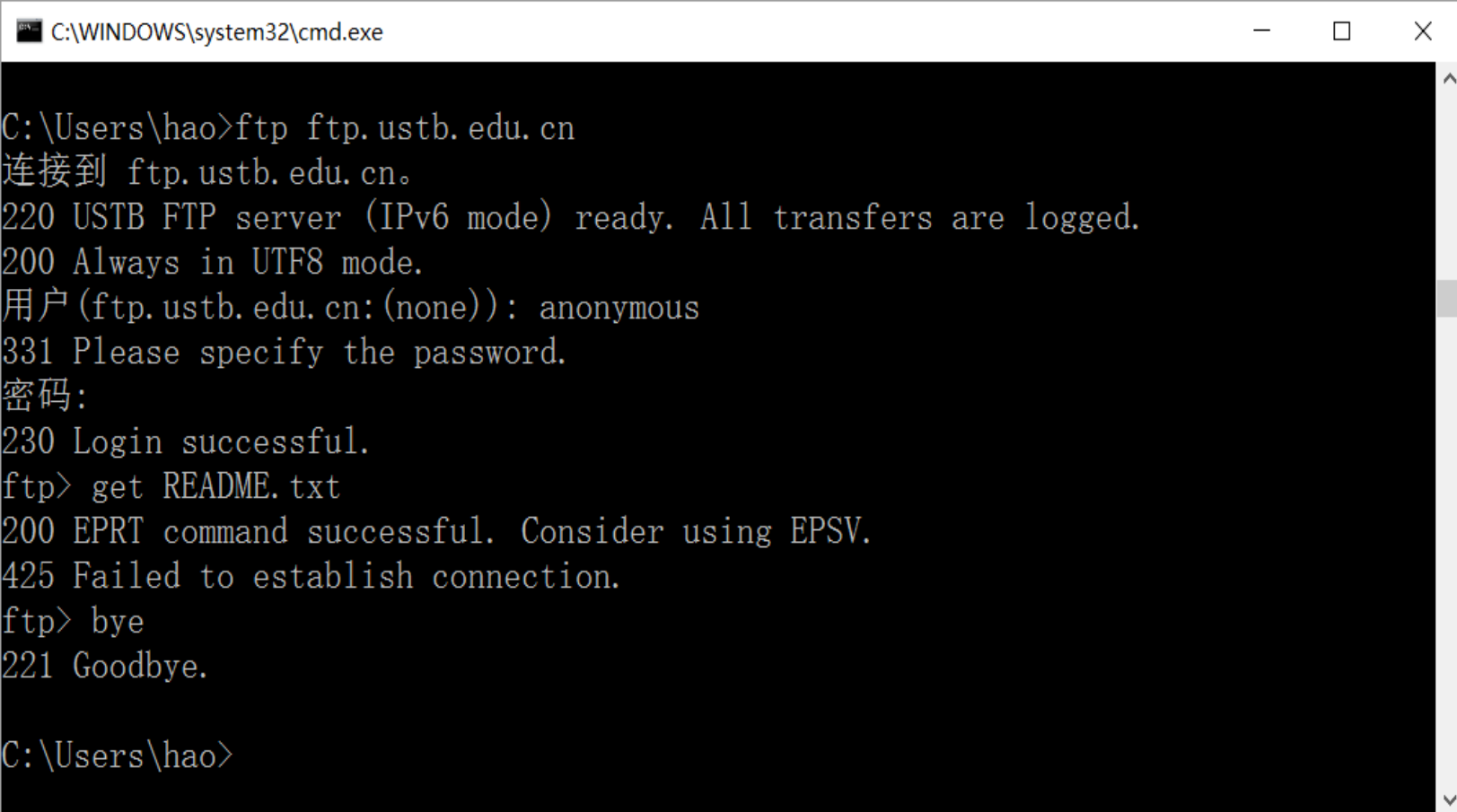
- 每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从何处获得名字映射信息的记录。
- 可大大减轻根域名服务器的负荷，使因特网上的 DNS 查询请求和回答报文的数量大为减少。
- 为保持高速缓存中的内容正确，域名服务器应为每项内容设置计时器，并处理超过合理时间的项（例如，每个项目只存放两天）。
- 当权限域名服务器回答一个查询请求时，在响应中都指明绑定有效存在的时间值。增加此时间值可减少网络开销，而减少此时间值可提高域名转换的准确性。

## 6.2 文件传送协议

### 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。

# FTP 命令举例



```
C:\WINDOWS\system32\cmd.exe

C:\Users\hao>ftp ftp.ustb.edu.cn
连接到 ftp.ustb.edu.cn。
220 USTB FTP server (IPv6 mode) ready. All transfers are logged.
200 Always in UTF8 mode.
用户(ftp.ustb.edu.cn:(none)): anonymous
331 Please specify the password.
密码:
230 Login successful.
ftp> get README.txt
200 EPRT command successful. Consider using EPSV.
425 Failed to establish connection.
ftp> bye
221 Goodbye.

C:\Users\hao>
```

# 基于 Web 的 FTP





# 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

---

- 1. 打开熟知端口（端口号为 21），使客户进程能够连接上。
- 2. 等待客户进程发出连接请求。
- 3. 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。
- 4. 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。

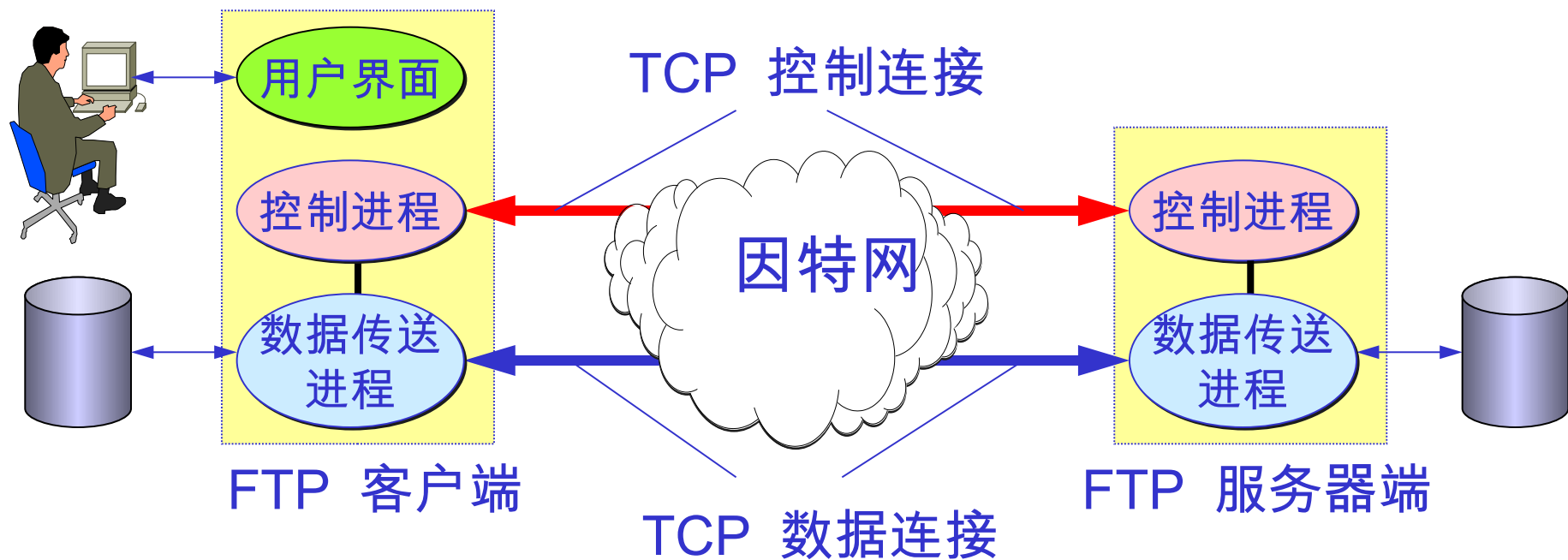


# 两个连接

---

- **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。
- 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。



## 6.2.3 简单文件传送协议 TFTP

### (Trivial File Transfer Protocol)

---

- TFTP 是一个很小的文件传送协议。
- TFTP 使用客户服务器方式和使用 **UDP 数据报**，因此 TFTP 需要有自己的差错改正措施。
- TFTP 只支持文件传输而**不支持交互**。
- TFTP 没有一个庞大的命令集，没有列目录的功能，**也不能对用户进行身份鉴别**。



# TFTP 的主要特点是

---

- (1) 每次传送的数据 PDU 中有 512 字节的数据，但最后一次可不足 512 字节。
- (2) 数据 PDU 也称为文件块 (block)，每个块按序编号，从 1 开始。
- (3) 支持 ASCII 码或二进制传送。
- (4) 可对文件进行读或写。
- (5) 使用很简单的首部。



# TFTP 的工作很像停止等待协议

---

- 发送完一个文件块后就等待对方的确认，确认时应指明所确认的块编号。
- 发完数据后在规定时间内**收不到确认**就要重发数据 PDU。
- 发送确认 PDU 的一方若在规定时间内**收不到下一个文件块**，也要重发确认 PDU。这样就可保证文件的传送不致因某一个数据报的丢失而告失败。



# TFTP 的工作很像停止等待协议

- 在一开始工作时。TFTP 客户进程发送一个读请求 PDU 或写请求 PDU 给 TFTP 服务器进程，其熟知端口号码为 69。
- TFTP 服务器进程要选择一个新的端口和 TFTP 客户进程进行通信。
- 若文件长度恰好为 512 字节的整数倍，则在文件传送完毕后，还必须在最后发送一个只含首部而无数据的数据 PDU。
- 若文件长度不是 512 字节的整数倍，则最后传送数据 PDU 的数据字段一定不满 512 字节，这正好可作为文件结束的标志。

# ftp\_tftp\_nfs 三种文件传输协议的区别

- 1. FTP 的基本应用就是将文件从一台计算机复制到另一台计算机中。它要存取一个文件，就必须先获得一个本地文件的副本，如果修改文件，也只能对文件的副本进行修改，然后再将修改后的文件副本传回到原节点。对 FTP 要记住几个关键词：**交互式、存取权限和副本。**

# ftp\_tftp\_nfs 三种文件传输协议的区别

- 2. 简单文件传送协议 TFTP(Trivial File Transfer Protocol) 是一个小而易于实现的文件传送协议。TFTP 是基于 UDP 数据报，需要有自己的差错改正措施。TFTP 只支持文件传输，不支持交互，没有庞大的命令集。也没有目录列表功能，以及不能对用户进行身份鉴别。但它的代码所占内存较小，适合用于升级路由器系统版本。
- TFTP 和 FTP 一个主要的区别就是它没有交互式，且不进行身份验证

# ftp\_tftp\_nfs 三种文件传输协议的区别

- 3. 在 NFS 的应用中，本地 NFS 的客户端应用可以透明地读写位于远端 NFS 服务器上的文件，就像访问本地文件一样。
- 远端 NFS 服务器上的文件目录可以挂载 ( mount ) 到本地 NFS 客户端机器上，适用于 windows\unix\linux 等不同操作系统。

# ftp\_tftp\_nfs 三种文件传输协议的区别

- 4.FTP 在修改数据文件时是需要首先获得一个文件的副本，如果计算机 A 上运行的应用程序要在远地计算机 B 的一个很大的文件中添加一行信息。那么就需要将此文件从计算机 B 传送到计算机 A，添加好信息后再回传到计算机 B。来回传输这样大的文件很花费时间，而这种传送是不必要的。
- 而 NFS 可使用户只复制一个大文件中的一个很小的片段，在网络上传送的只是少量的修改数据。
- 即 **NFS 和 FTP 的区别是无副本**





## 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，同时也能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**。



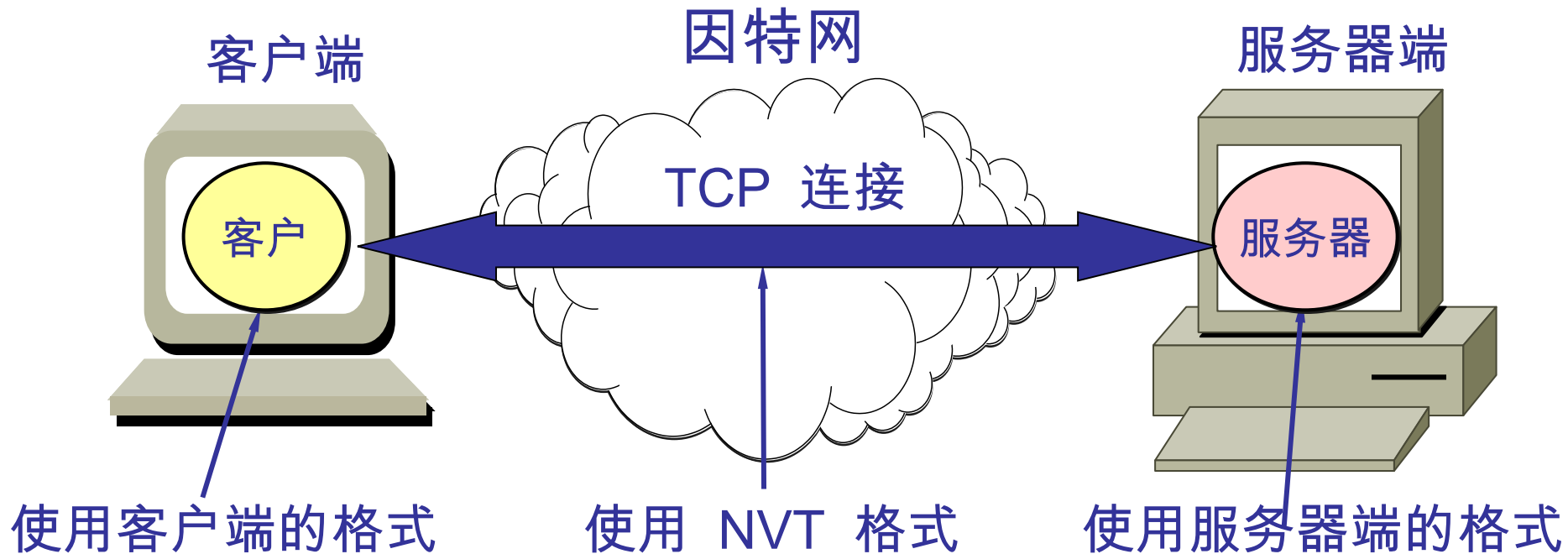
# 客户-服务器方式

---

- 现在由于 PC 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

---

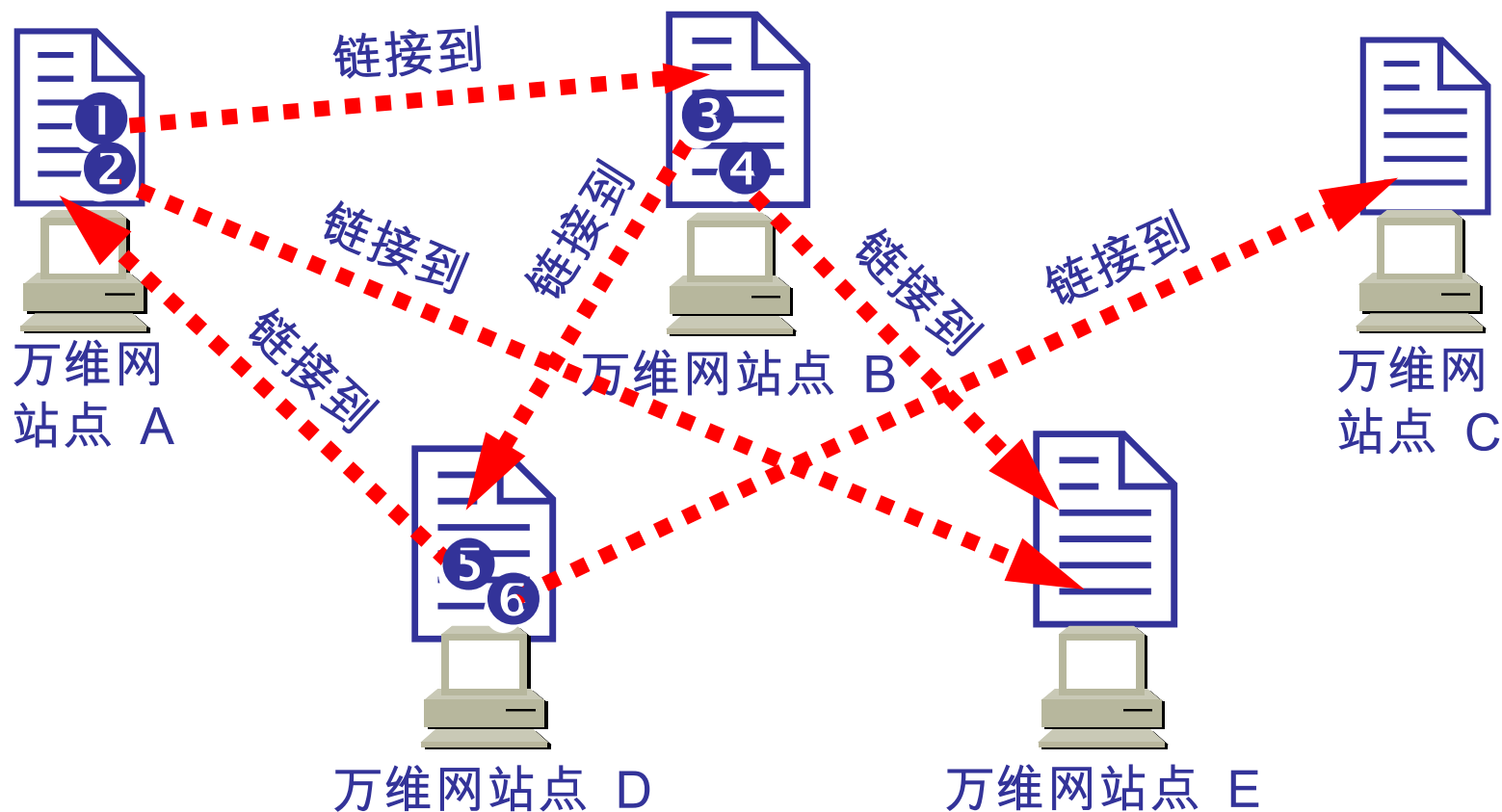
- 客户软件把用户的击键和命令转换成 NVT 格式，并送交服务器。
- 服务器软件把收到的数据和命令，从 NVT 格式转换成远地系统所需的格式。
- 向用户返回数据时，服务器把远地系统的格式转换为 NVT 格式，本地客户再从 NVT 格式转换到本地系统所需的格式。

# 6.4 万维网 WWW

## 6.4.1 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 >

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)。



# Web 网站的工作原理

---

## ■ URL

- 统一资源定位符 ( Uniform Resource Locator )
- 是一种系统的地址符号，用简洁的字符串来表示通过 Internet 传输的信息资源。例如，URL 可用来表示 Web 网站地址和 FTP 站点地址。
- URL 是一个字符的序列，其书写方式如下：< 协议 > : //< 域名地址或 IP 地址 > / 文件路径和文件名，例如，
  - <http://www.javvin.com/lab1/index.htm>
  - <ftp://ftp.ustb.edu.cn/public>

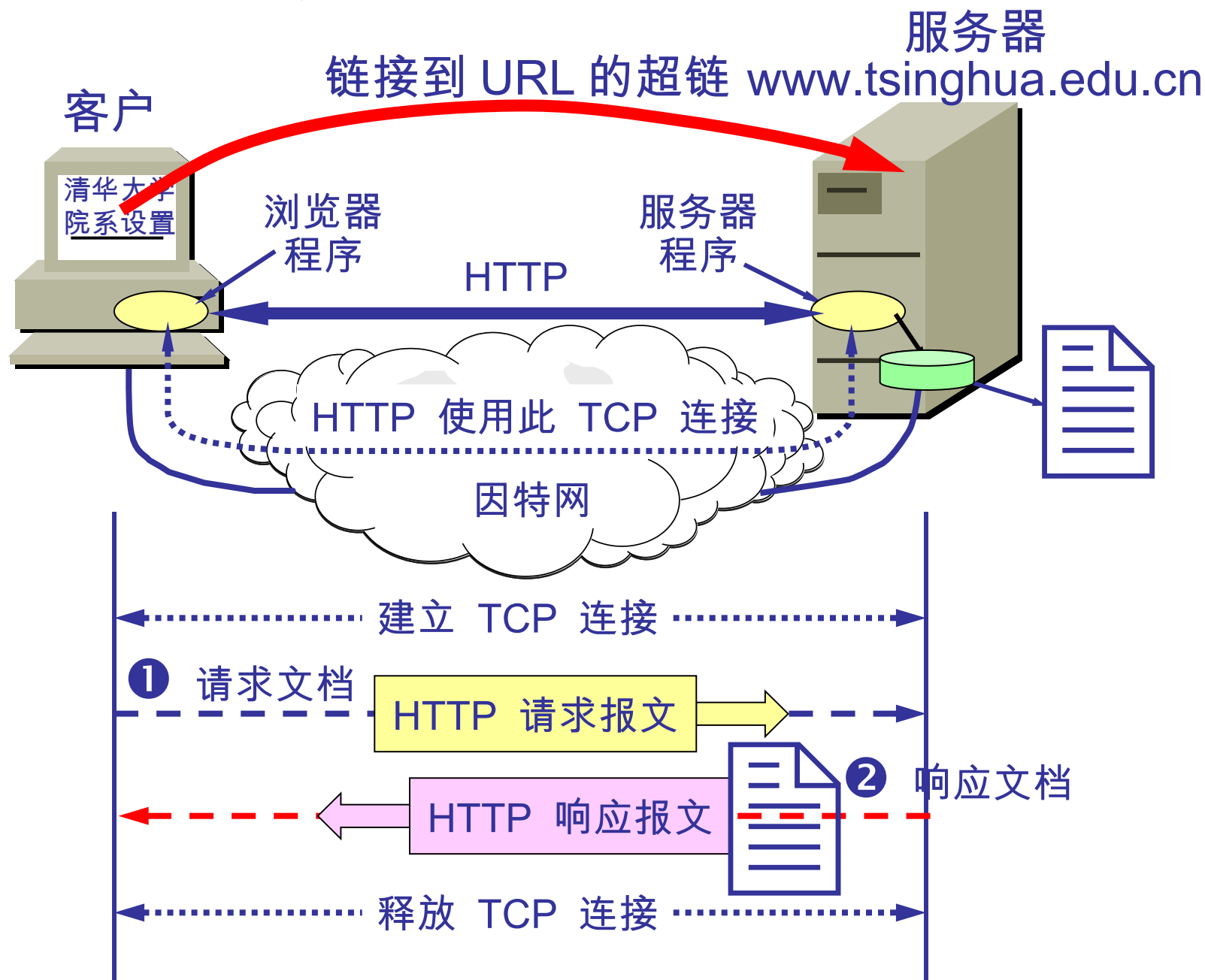
## 6.4.3 超文本传送协议 HTTP

### 1. HTTP 的操作过程

---

- 为了使超文本的链接能够高效率地完成，需要用 HTTP 协议来传送一切必须的信息。
- 从层次的角度看，HTTP 是面向事务的 (transaction-oriented) 应用层协议，它是万维网上能够可靠地交换文件（包括文本、声音、图像等各种多媒体文件）的重要基础。

# 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。

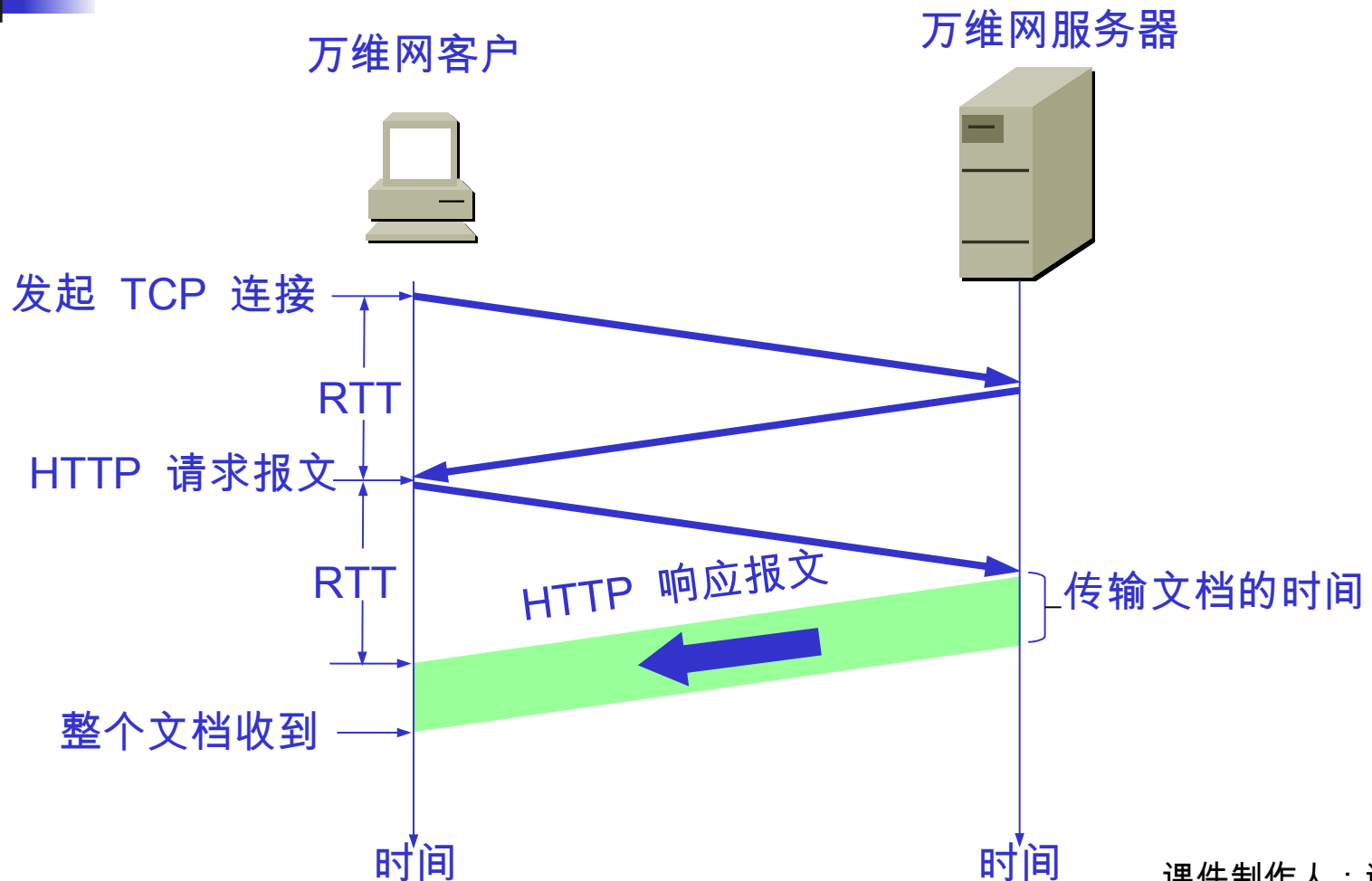


# HTTP 的主要特点

---

- HTTP 是面向事务的客户服务器协议。
- HTTP 1.0 协议是**无状态的** (stateless)。
  -
- HTTP 协议本身也是无连接的，虽然它使用了面向连接的 TCP 向上提供的服务。

# 请求一个万维网文档所需的时间





# 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。



# 持续连接的两种工作方式

- 非流水线方式：客户在收到前一个响应后才能发出下一个请求。这比非持续连接的两倍 RTT 的开销节省了建立 TCP 连接所需的一个 RTT 时间（每个对象一个 RTT 时间）。但服务器在发送完一个对象后，其 TCP 连接就处于空闲状态，浪费了服务器资源。
- 流水线方式：客户在收到 HTTP 的响应报文之前就能够接着发送新的请求报文。一个接一个的请求报文到达服务器后，服务器就可连续发回响应报文。使用流水线方式时，客户访问所有的对象只需花费一个 RTT 时间，使 TCP 连接中的空闲时间减少，提高了下载文档效率。



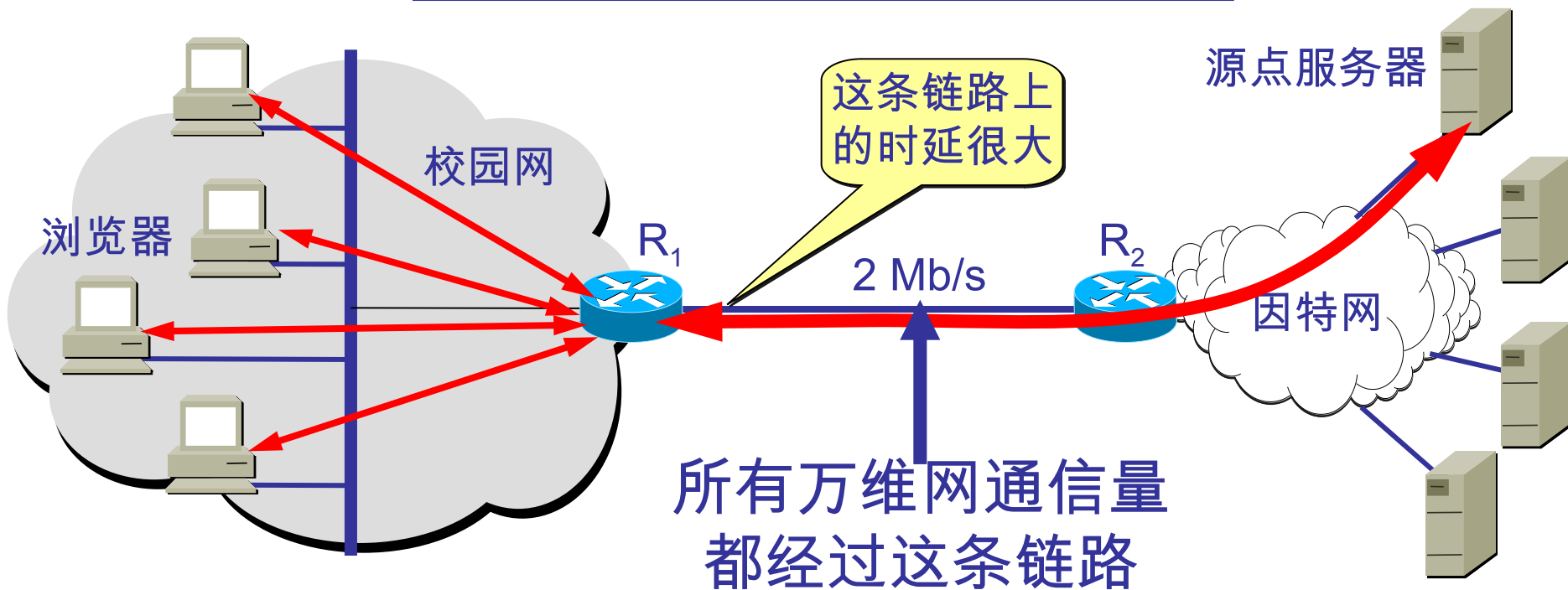
# 代理服务器 (proxy server)

---

- **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。

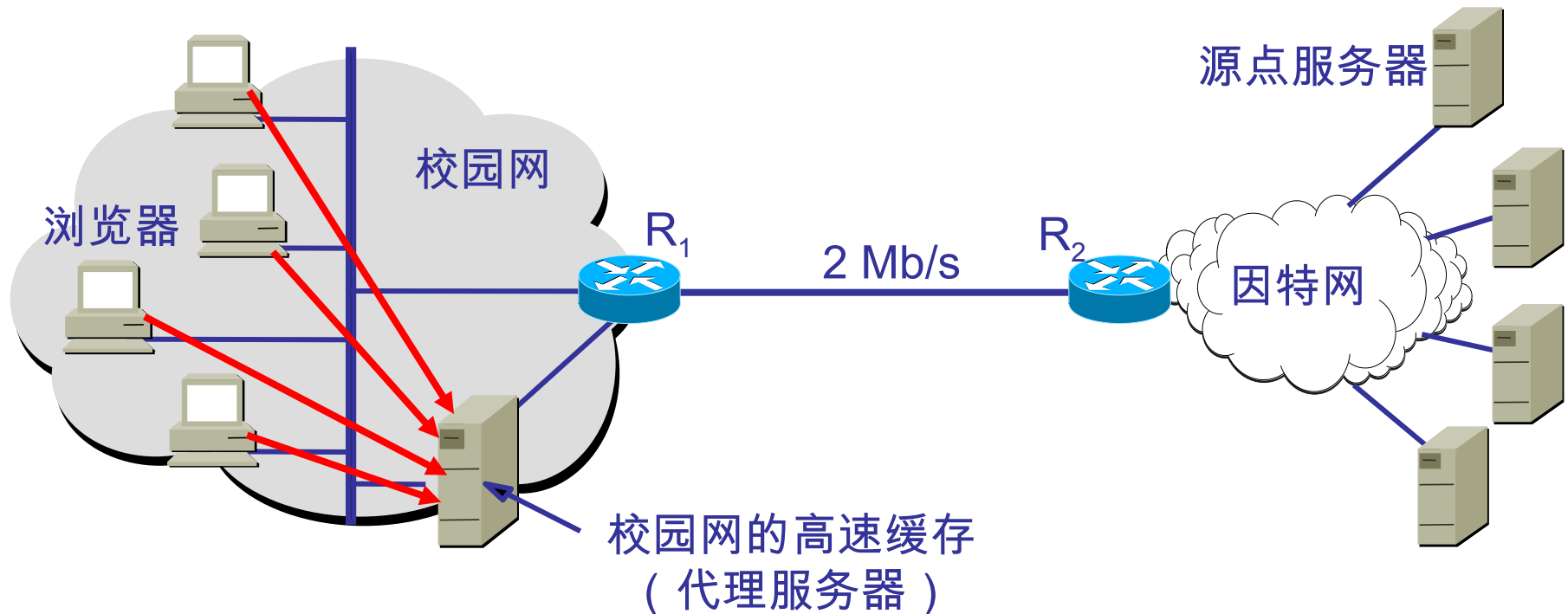
# 使用高速缓存可减少 访问因特网服务器的时延

没有使用高速缓存的情况



# 使用高速缓存的情况

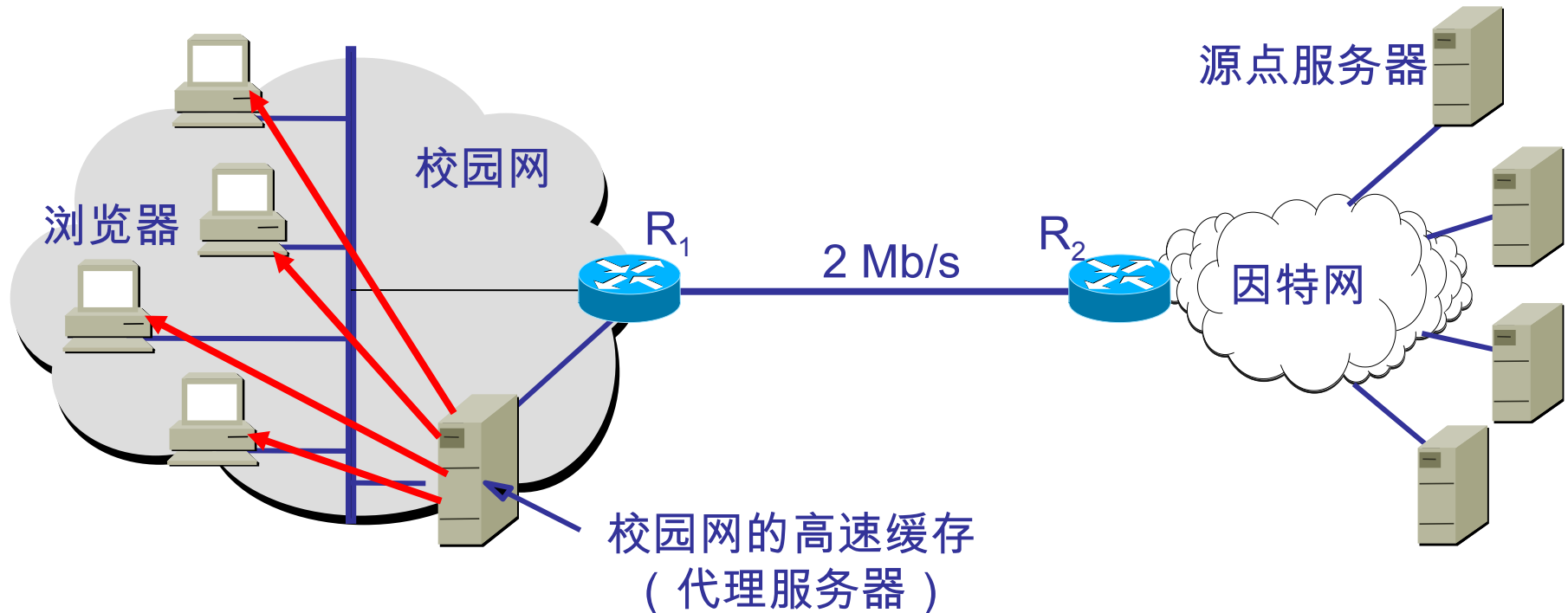
(1) 浏览器访问因特网的服务器时，要先与校园网的高速缓存建立 TCP 连接，并向高速缓存发出 HTTP 请求报文





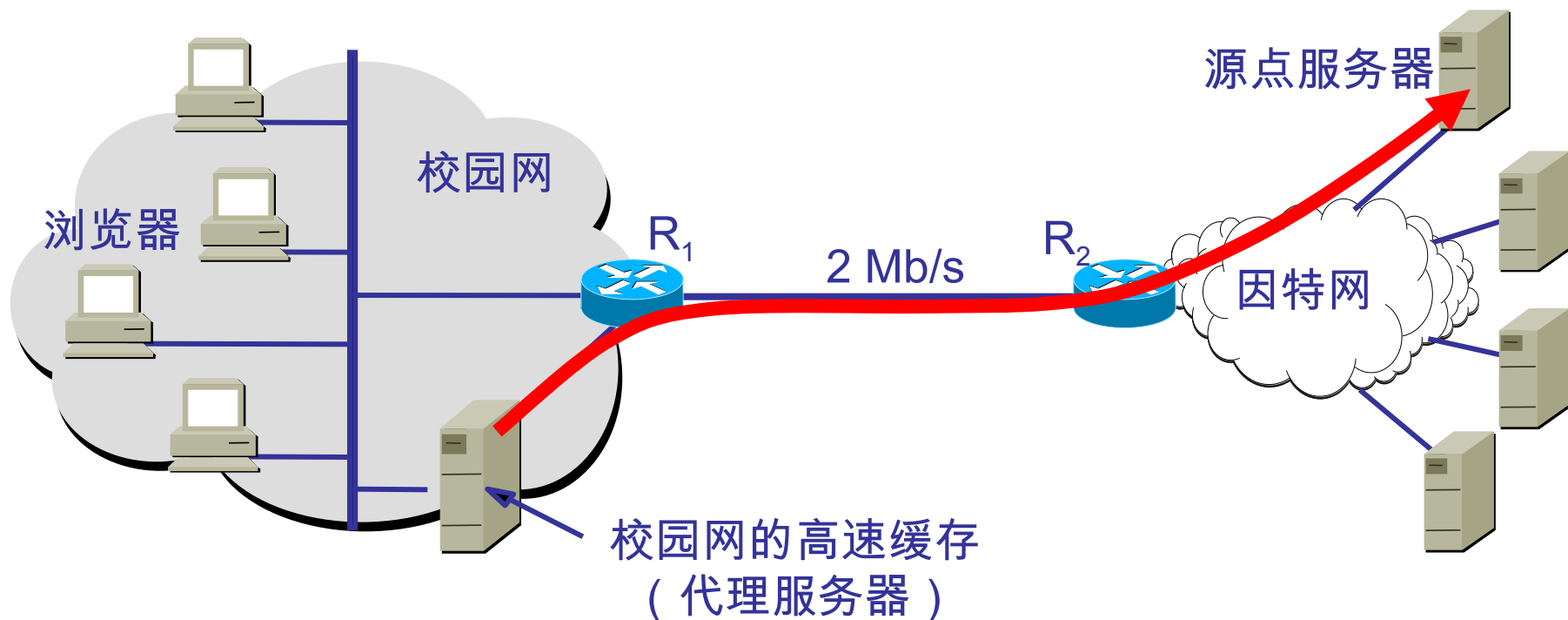
# 使用高速缓存的情况

(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。



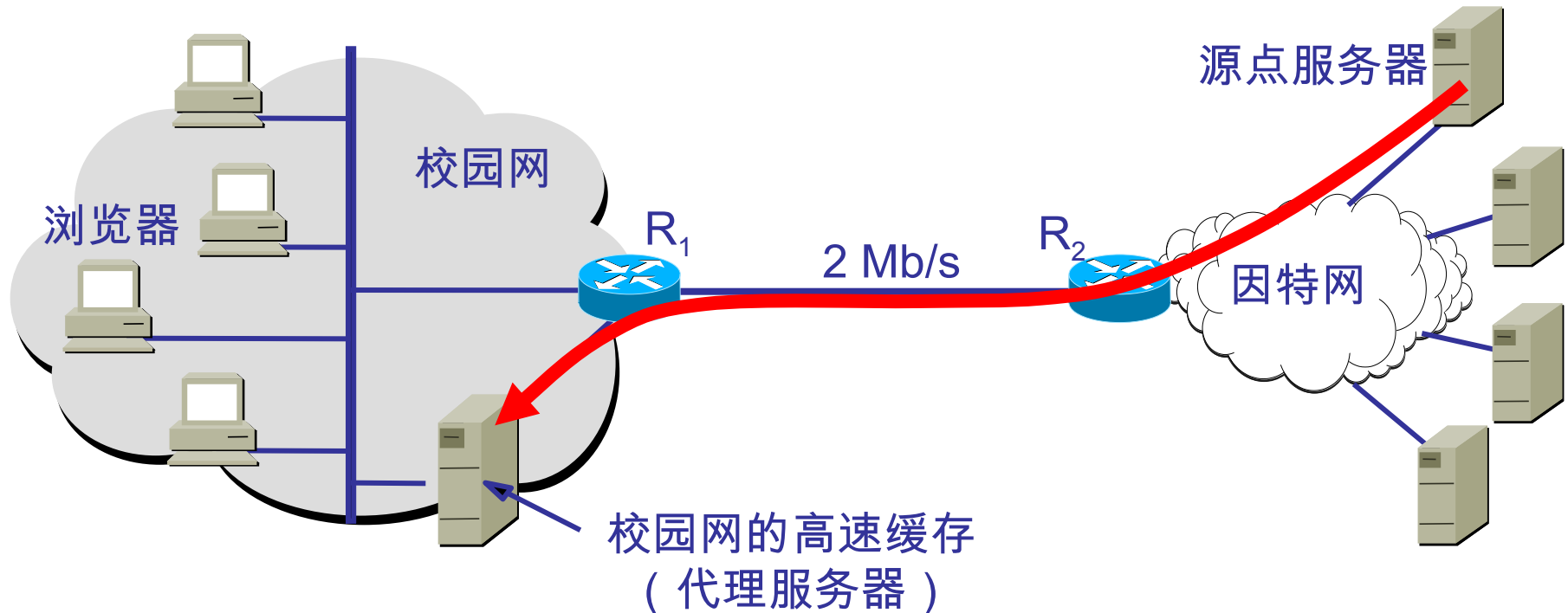
# 使用高速缓存的情况

(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。



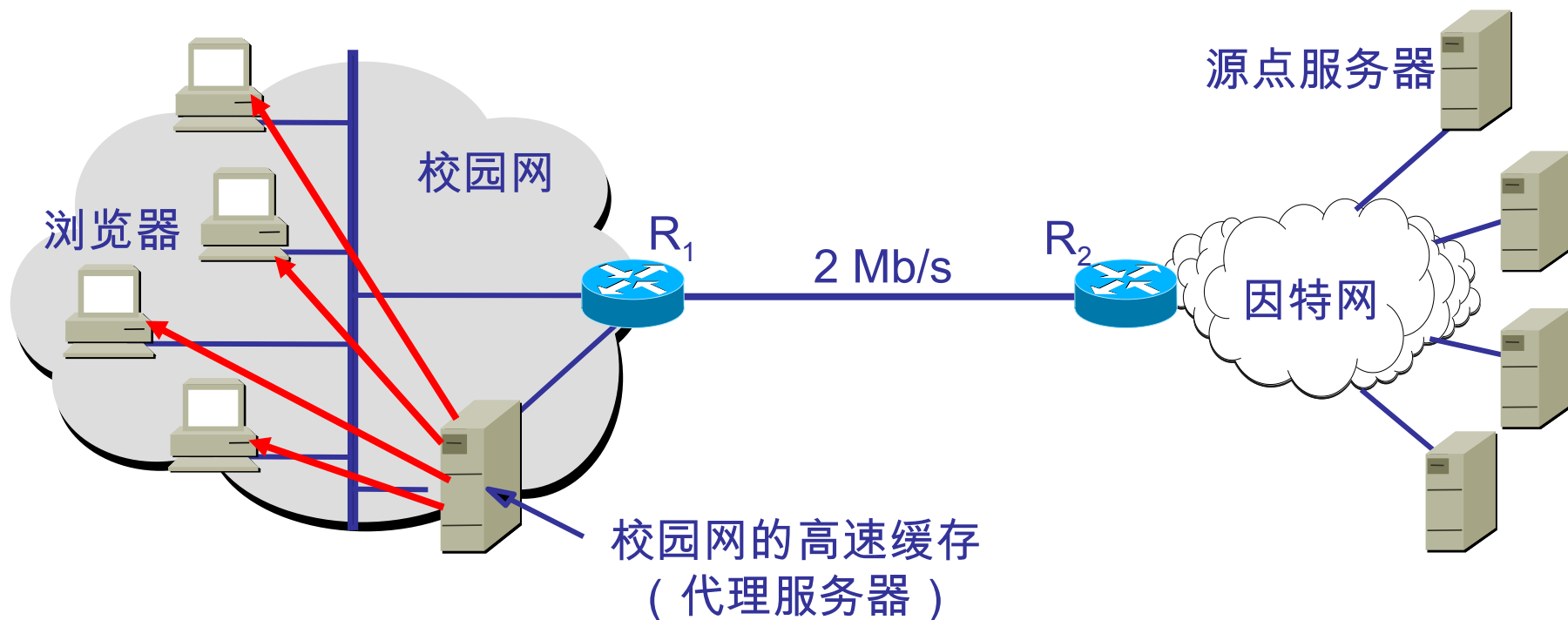
# 使用高速缓存的情况

(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。





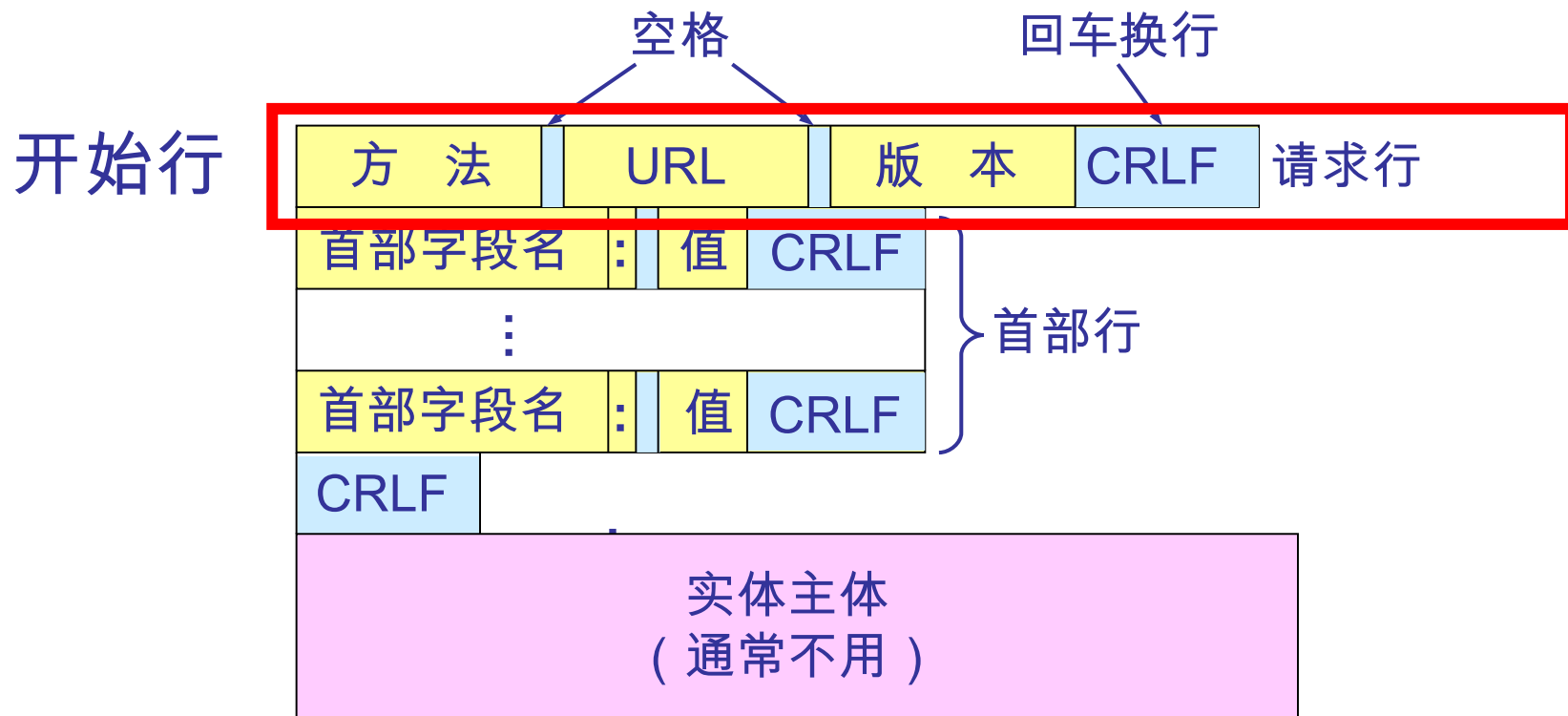
### 3. HTTP 的报文结构

---

HTTP 有两类报文：

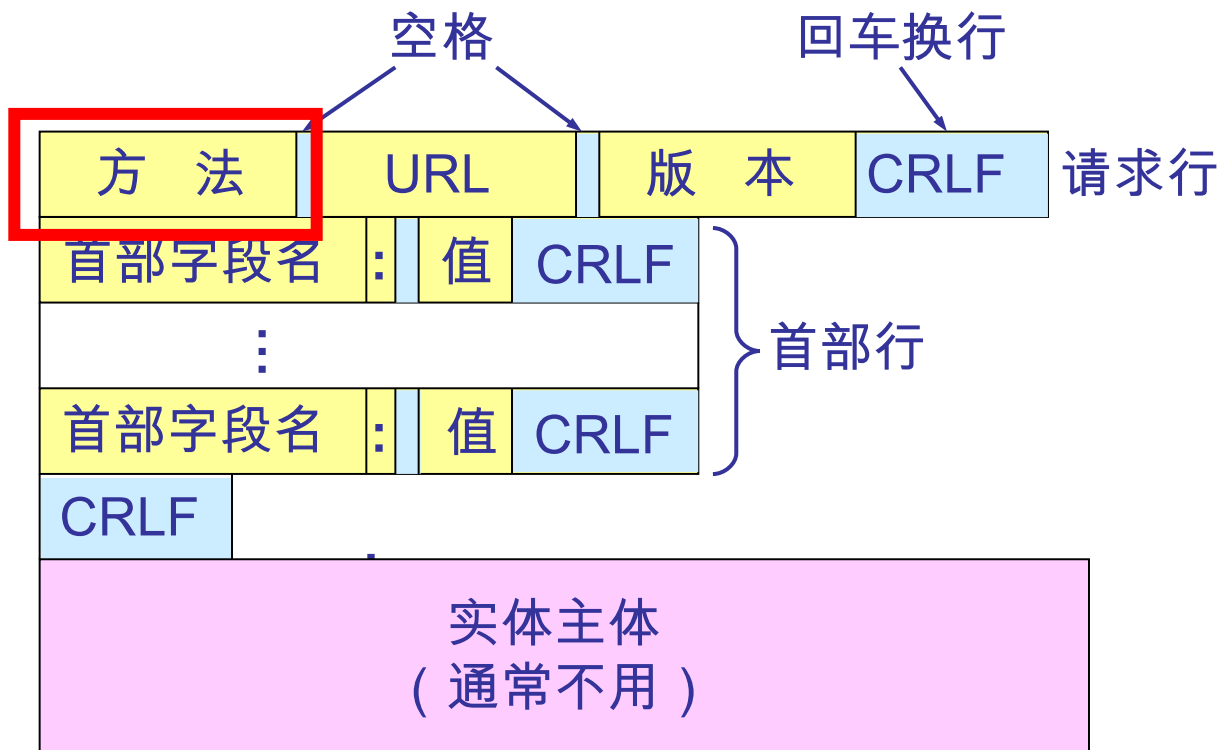
- 请求报文——从客户向服务器发送请求报文。
- 响应报文——从服务器到客户的回答。
- 由于 HTTP 是面向正文的 (text-oriented) ，因此在报文中的每一个字段都是一些 ASCII 码串，因而每个字段的长度都是不确定的。

# 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。

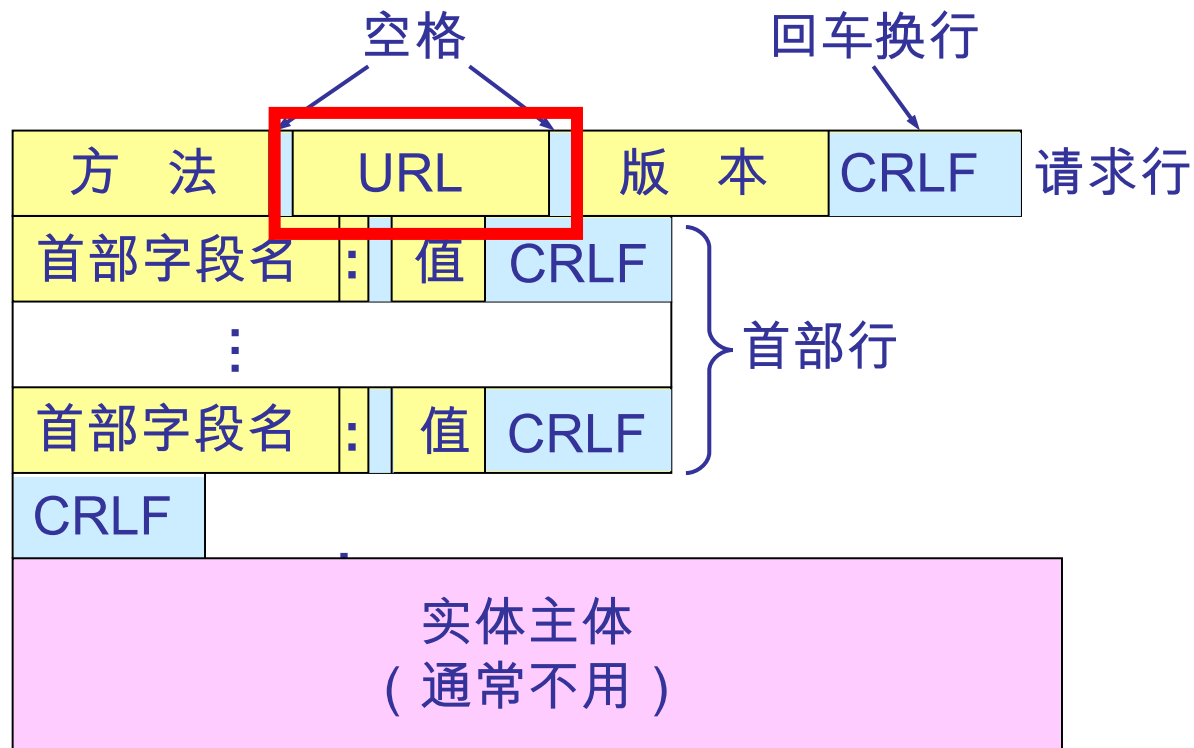


# HTTP 请求报文的一些方法

方法 ( 操作 )	意义
OPTION	请求一些选项的信息
GET	请求 <b>读取</b> 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	<b>给服务器添加信息</b> ( 例如 , 注释 )
PUT	在指明的 URL 下 <b>存储一个文档</b>
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

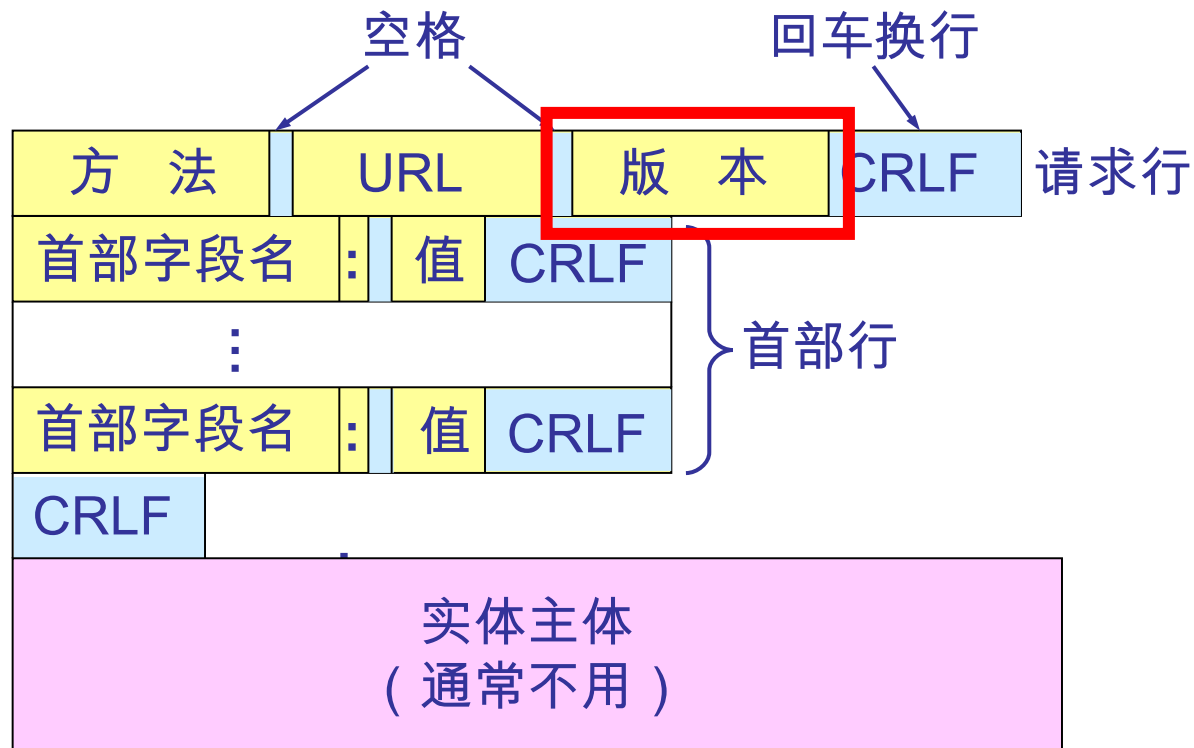


# HTTP 的报文结构（请求报文）



“URL” 是所请求的资源的 URL。

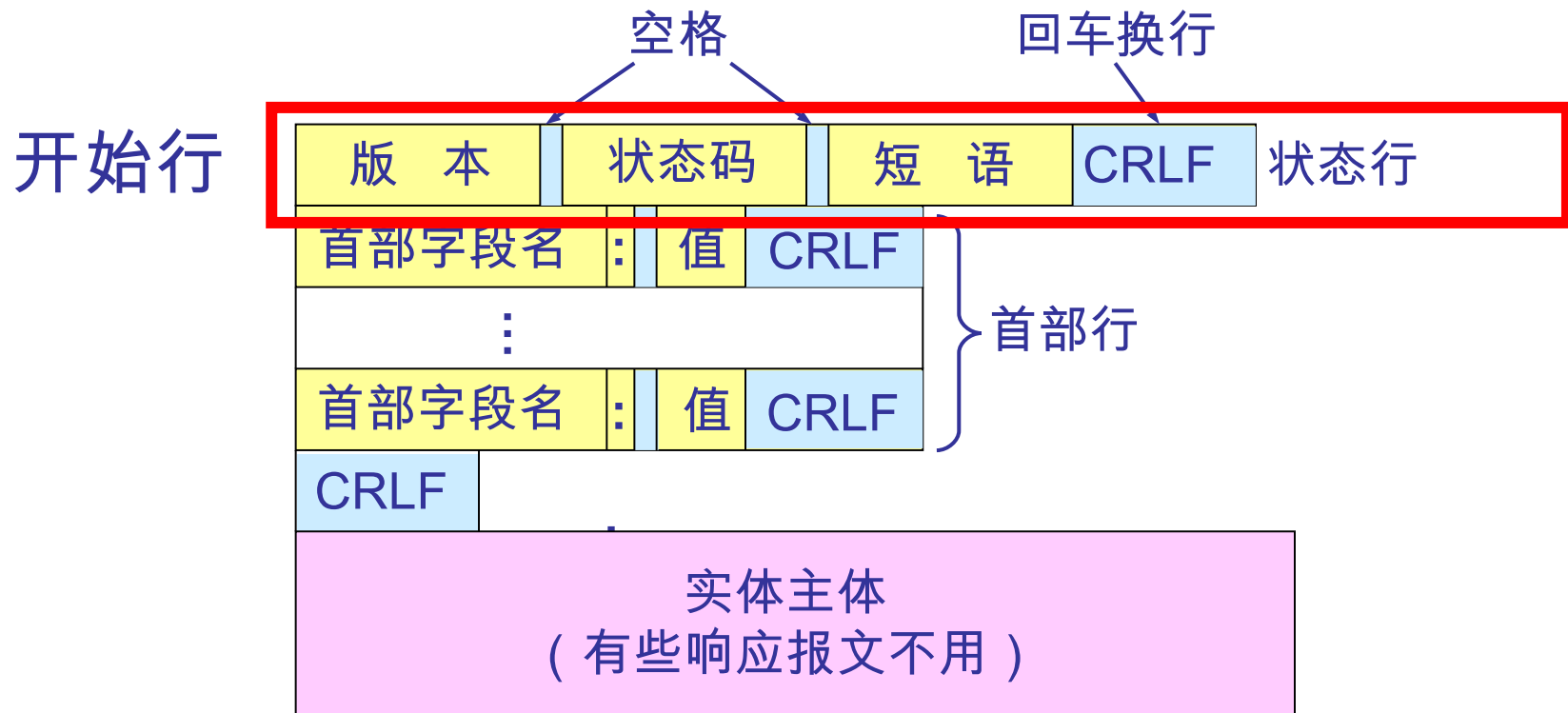
# HTTP 的报文结构（请求报文）



“版本”是 HTTP 的版本

。

# HTTP 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**。



# 状态码都是三位数字

---

- 1xx 表示通知信息的，如请求收到了或正在进行处理。
- 2xx 表示成功，如接受或知道了。
- 3xx 表示重定向，表示要完成请求还必须采取进一步的行动。
- 4xx 表示客户的差错，如请求中有错误的语法或不能完成。
- 5xx 表示服务器的差错，如服务器失效无法完成请求。



# 例题

---

P256



## 4. 在服务器上存放用户的信息

---

- 万维网站点使用 Cookie 来跟踪用户。
- Cookie 表示在 HTTP 服务器和客户之间传递的状态信息。
- 使用 Cookie 的网站服务器为用户产生一个唯一的识别码。利用此识别码，网站就能够跟踪该用户在该网站的活动。



# Cookie 可以用来提升用户体验

---

- 比如网站可以使用 Cookie 来记录用户的 **登录状态**，用户只要登录一次就可以不用登录了，
- 购物网站通过 Cookie 来 **保存购物车中的商品**等。
- 同时很多的 **网站分析**都是依靠 Cookie 来完成的

# 第一方 Cookie 和第三方 Cookie

- 
- 比如，访问 `www.a.com` 这个网站，这个网站设置了一个 Cookie，这个 Cookie 也只能被 `www.a.com` 这个域下的网页读取，这就是第一方 Cookie。
  - 如果还是访问 `www.a.com` 这个网站，网页里有用到 `www.b.com` 网站的一张图片，浏览器在 `www.b.com` 请求图片的时候，`www.b.com` 设置了一个 Cookie，那这个 Cookie 只能被 `www.b.com` 这个域访问，反而不能被 `www.a.com` 这个域访问，因为对我们来说，我们实际是在访问 `www.a.com` 这个网站被设置了一个 `www.b.com` 这个域下的 Cookie，所以叫**第三方 Cookie**。





# 第三方 Cookie 的优势和应用

- 第三方 Cookie 在某些特定情况下可以实现第一方 Cookie 无法实现的功能。
- 比如，当我们有多个域名的网站需要跟踪，我们希望了解到用户点击某个广告到达域名 A 下的网页，然后可能浏览了不论那个域名下的页面，最后在域名 B 下的网页完成注册的情况。
- 广告可以在域名 A 下的网页被跟踪到，而注册可以在域名 B 下的网页跟踪到。
- 如果我们使用第一方 Cookie，会为域名 A 建立一个 Cookie，为域名 B 再建立一个 Cookie，他们可以关联各自域名下网页上的行为，但是无法关联起来。
- 而使用第三方 Cookie，那么无论多少个域，都只有一个 Cookie，一个属于第三方域的 Cookie，网站下所有域都能共享这个 Cookie，那么所有的行为都能被关联起来分析。

## 6.4.4 万维网的文档

### 1. 超文本标记语言 HTML

---

- 超文本**标记**语言 HTML 中的 Markup 的意思就是“设置标记”。
- HTML 定义了许多用于排版的命令（即标签）。
- HTML 把各种标签嵌入到万维网的页面中。这样就构成了所谓的 HTML 文档。HTML 文档是一种可以用任何文本编辑器创建的 **ASCII 码文件**。



# HTML 文档

---

- 仅当 HTML 文档是以 **.html** 或 **.htm** 为后缀时，浏览器才对此文档的各种标签进行解释。
- 如 HTML 文档改换以 **.txt** 为其后缀，则 HTML 解释程序就不对标签进行解释，而浏览器只能看见原来的文本文件。
- 当浏览器从服务器读取 HTML 文档后，就按照 HTML 文档中的各种标签，根据浏览器所使用的显示器的尺寸和分辨率大小，重新进行排版并恢复出所读取的页面。

# HTML 文档中标签的用法

<HTML>

HTML 文档开始



<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

首部开始

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

标题

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

首部结束

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

主体开始



<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>



# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

1 级标题

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P>这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P>这是第二个段落。 </P>

</BODY>

</HTML>

第一个段落



# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

第二个段落

</BODY>

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

主体结束

</HTML>

# HTML 文档中标签的用法

<HTML>

<HEAD>

<TITLE> 一个 HTML 的例子 </TITLE>

</HEAD>

<BODY>

<H1>HTML 很容易掌握 </H1>

<P> 这是第一个段落。虽然很短，但它仍是一个段落。 </P>

<P> 这是第二个段落。 </P>

</BODY>

</HTML>

HTML 文档结束





# 两种不同的链接

---

- 远程链接：超链的终点是其他网点上的页面。
- 本地链接：超链指向本计算机中的某个文件。



## 2. 动态万维网文档

---

- **静态文档**是指该文档创作完毕后就存放在万维网服务器中，在被用户浏览的过程中，内容不会改变。
- **动态文档**是指文档的内容是在浏览器访问万维网服务器时才由应用程序动态创建。
- 动态文档和静态文档之间的主要差别体现在**服务器**一端。这主要是文档内容的生成方法不同。而从浏览器的角度看，这两种文档并没有区别。



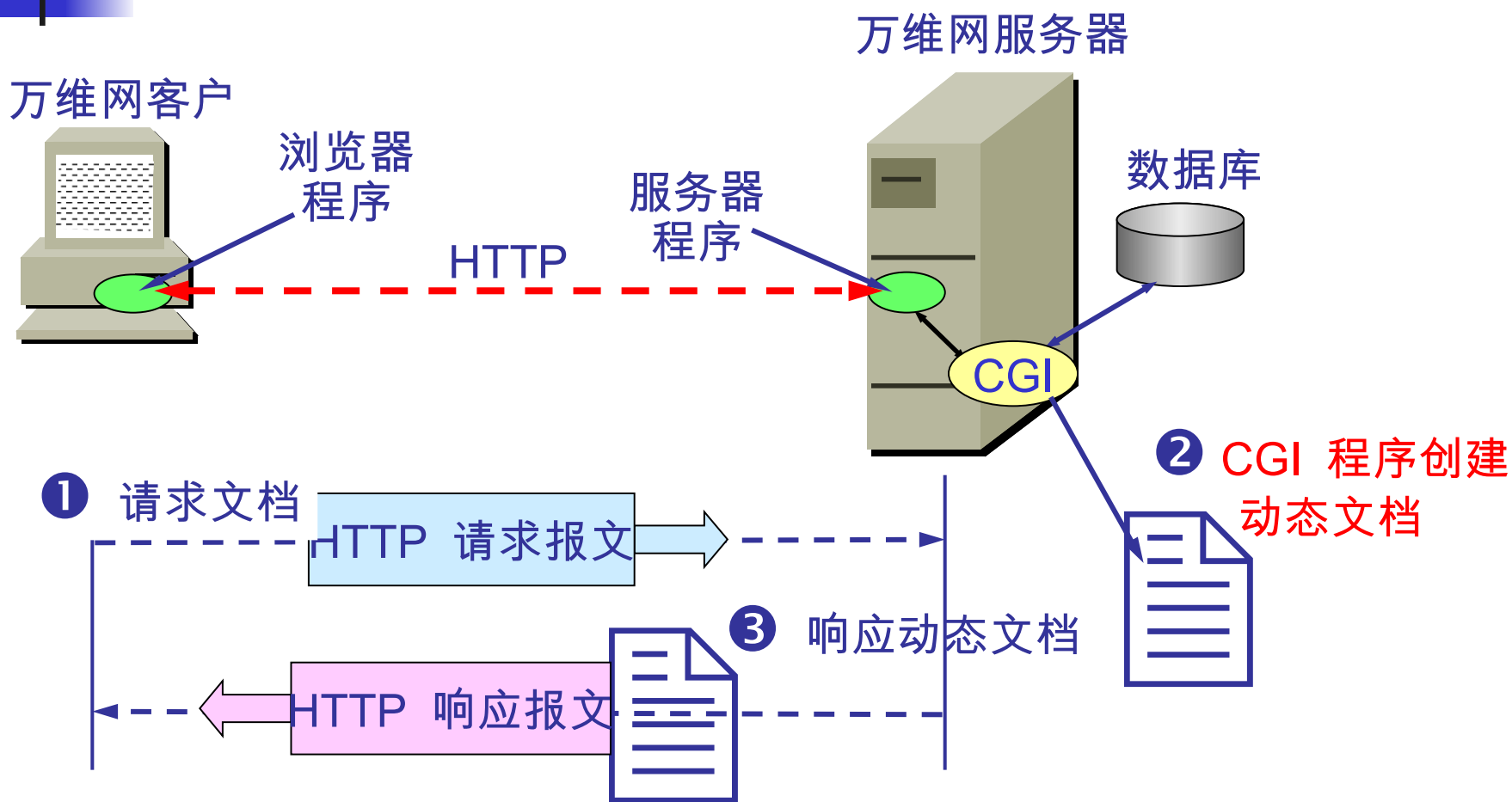
# 万维网服务器功能的扩充

---

- (1) 应增加另一个应用程序，用来处理浏览器发来的数据，并创建动态文档。
- (2) 应增加一个机制，用来使万维网服务器把浏览器发来的数据传送给这个应用程序，然后万维网服务器能够解释这个应用程序的输出，并向浏览器返回HTML 文档。



# 扩充了功能的万维网服务器



# 通用网关接口 CGI

(Common Gateway Interface)

- CGI 是一种标准，它定义了动态文档应如何创建，输入数据应如何提供给应用程序，以及输出结果应如何使用。
- 万维网服务器与 CGI 的通信遵循 CGI 标准。
- “通用”：CGI 标准所定义的规则对其他任何语言都是通用的。
- “网关”：CGI 程序的作用像网关。
- “接口”：有一些已定义好的变量和调用等可供其他 CGI 程序使用。



# CGI 程序

---

- CGI 程序的正式名字是 CGI 脚本 (script)。
- “脚本”指的是一个程序，它被另一个程序（解释程序）而不是计算机的处理机来解释或执行。
- 脚本运行起来要比一般的编译程序要慢，因为它的每一条指令先要被另一个程序来处理（这就要一些附加的指令），而不是直接被指令处理器来处理。



# CGI 标准

---

■ CGI 是 Common Gateway Interface 的缩写，是用于连接网页和应用程序的接口。众所周知，HTML 语言的功能是比较贫乏的，难以完成诸如访问数据库等一类的操作，而实际的情况则是经常需要先对数据库进行操作（比如文件检索系统），然后把访问的结果动态地显示在网页上。诸如此类的需求只用 HTML 是无法做到的，所以 CGI 便应运而生。



# CGI 程序

- CGI 程序是在 WebServer 端运行的一个可执行程序，由主页的一个热链接激活进行调用，并对该程序的返回结果进行处理，显示在主页上。简而言之，CGI 就是为了扩展主页的功能而设立的。比如 JavaScript、VBScript 等。
- 随后，诸如 ASP、ASP.net、JSP 等技术也发展起来了，它们的目的是相同的，只是编写起来更容易、功能更丰富。

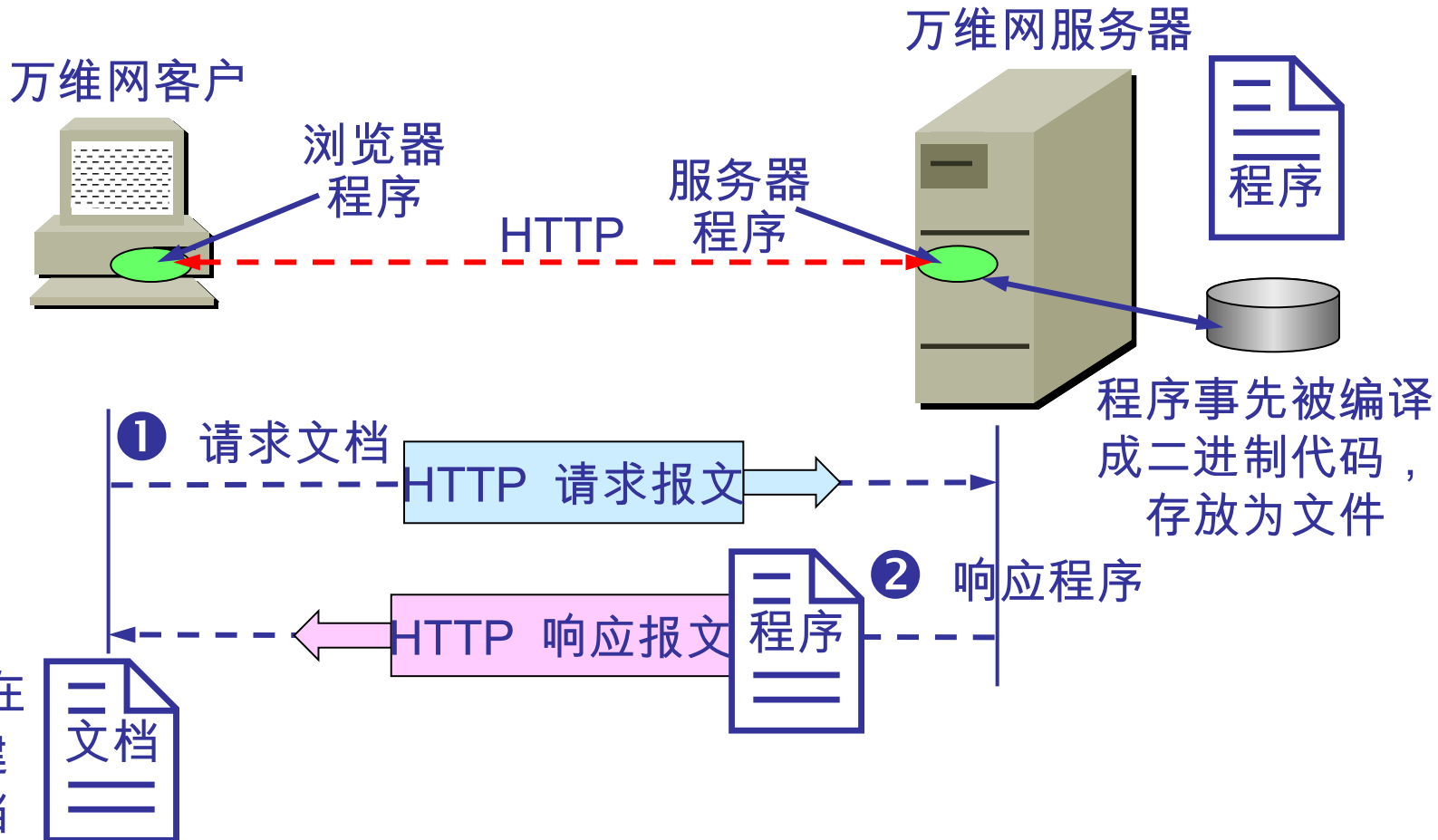


### 3. 活动万维网文档

---

- **活动文档** (active document) 技术把所有的工作都转移给浏览器端。
- 每当浏览器请求一个活动文档时，服务器就返回一段程序副本在浏览器端运行。
- 活动文档程序可与用户直接交互，并可连续地改变屏幕的显示。
- 由于活动文档技术不需要服务器的连续更新传送，对网络带宽的要求也不会太高。

# 活动文档在客户端创建





# 用 Java 技术创建活动文档

---

- 由美国 Sun 公司开发的 **Java** 语言是一项用于创建和运行活动文档的技术。
- 在 Java 技术中使用“**小应用程序**”(applet) 来描述活动文档程序。
- 用户从万维网服务器下载嵌入了 Java 小应用程序的 HTML 文档后，可在浏览器的屏幕上点击某个图像，就可看到动画效果，或在下拉式菜单中点击某个项目，就可看到计算结果。
- Java 技术是活动文档技术的一部分。



# Java 技术共三个主要组成部分

- (1) 程序设计语言。Java 包含一个新的程序设计语言，用来编写传统的计算机程序和 Java 小应用程序。
- (2) 运行 (runtime) 环境。这是运行 Java 程序所必须的运行环境，其中包括 Java 虚拟机（简称为 JVM），该软件定义了 Java 二进制代码的执行模型。
- (3) 类库 (class library)。为了更容易编写 Java 小应用程序，Java 提供了强大的类库支持。



# Java

---

- Java 是一种面向对象的高级语言，从 C++ 派生出来的，它省略了 C++ 很多复杂的、很少用的语言特点。
- Java 的每一个数据项都有一个确定的类型。对数据的操作严格按照该数据的类型来进行。
- Java 的编译程序将源程序转换成 Java 字节码 (bytecode)，这是一种与机器无关的二进制代码。计算机程序调用解释程序读取字节码，并解释执行。



# 计算机硬件无关

---

- Java 语言、字节码以及 Java 运行系统都被设计成与计算机硬件无关。一旦形成了字节码，就可可在任何计算机上运行并产生相同的输出。
- Java 小应用程序与机器无关可在任何计算机上运行的浏览器程序能够下载并运行活动文档。
  -
- 可保证活动文档在所有的浏览器上产生同样的正确输出。
- 可大大地降低活动文档的创建和测试费用，因为不必为每一种计算机都制作一个副本。



# Java 解释程序

---

- 运行 Java 的浏览器需要有 HTML 解释程序和 Java 小应用程序解释程序。
- 解释程序的核心是一个模仿计算机的简单循环。
- 解释程序维持一个指令指针，在初始化时指在小应用程序的开始处。
- 在每一次循环操作时，解释程序在指令指针指向的地址读取字节码。然后解释程序对字节码进行解码，并完成指明的操作。

## 6.4.5 万维网的信息检索系统

### 1. 全文检索搜索和分类目录搜索

- 在万维网中用来进行搜索的程序叫做**搜索引擎**。
- **全文检索搜索引擎**是一种纯技术型的检索工具。它的工作原理是通过搜索软件到因特网上的各网站收集信息，找到一个网站后可以从这个网站再链接到另一个网站。然后按照一定的规则建立一个很大的在线数据库供用户查询。
- 用户在查询时只要输入关键词，就从已经建立的索引数据库上进行查询（并不是实时地在因特网上检索到的信息）。



# 分类目录搜索

- **分类目录搜索引擎**并不采集网站的任何信息，而是利用各网站向搜索引擎提交的网站信息时填写的关键词和网站描述等信息，经过人工审核编辑后，如果认为符合网站登录的条件，则输入到分类目录的数据库中，供网上用户查询。
- 分类目录搜索也叫做**分类网站搜索**。



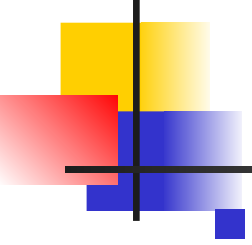
# 一些著名的搜索引擎

---

- 最著名的全文检索搜索引擎：
  - Google ( 谷歌 ) ([www.google.com](http://www.google.com))
  - 百度 ([www.baidu.com](http://www.baidu.com))
- 最著名的分类目录搜索引擎：
  - 雅虎 ([www.yahoo.com](http://www.yahoo.com))
  - 雅虎中国 ([cn.yahoo.com](http://cn.yahoo.com))
  - 新浪 ([www.sina.com](http://www.sina.com))
  - 搜狐 ([www.sohu.com](http://www.sohu.com))
  - 网易 ([www.163.com](http://www.163.com))

# 垂直搜索引擎

(Vertical Search Engine)



针对某一特定领域、特定人群或某一特定需求提供搜索服务。垂直搜索也是提供关键字来进行搜索的，但被放到了一个行业知识的上下文中，返回的结果更倾向于信息、消息、条目等。



## 6.4.6 博客、微博和轻博

### 1. 博客

- 博客是万维网日志 (web log) 的简称。也有人把 blog 进行音译，译为“部落格”，或“部落阁”。还有人用“博文”来表示“博客文章”。
- Weblog 这个新词是 Jorn Barger 于 1997 年创造的。
- 简写的 blog（这是今天最常用的术语）则是 Peter Merholz 于 1999 年创造的。
- 有人把 blog 既当作名词，也当作动词，表示编辑博客或写博客。



# 1. 博客（续）

---

- 博客已经极大地扩充了因特网的应用和影响。
- 在博客出现以前，网民是因特网上内容的消费者。
- 但博客改变了这种情况，网民不仅是因特网上内容的消费者，而且还是因特网上内容的生产者。



## 2. 微博

- 微博就是微型博客 (microblog) ，又称为微博客。
- 微博不同于一般的博客。微博只记录片段、碎语，三言两语，现场记录，发发感慨，晒晒心情，永远只针对一个问题进行回答。
- 用户可以通过网页、 WAP 网、手机短信彩信、手机客户端等多种方式更新自己的微博。
- 每条微博字数限制为 140 字，提供插入单张图片、视频地址、音乐功能。



## 3. 轻博

---

- 轻博就是轻博客 (light blogging) 。
- 轻博客是一种介于博客和微博之间的网络服务，同样为用户提供生成内容表达自己的平台。
- 轻博可以发送博文，没有字数限制。
- 轻博发表后，其界面会好看些。
- 在轻博中，推荐与发现的内容比较丰富。

# 6.5 电子邮件

## 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。

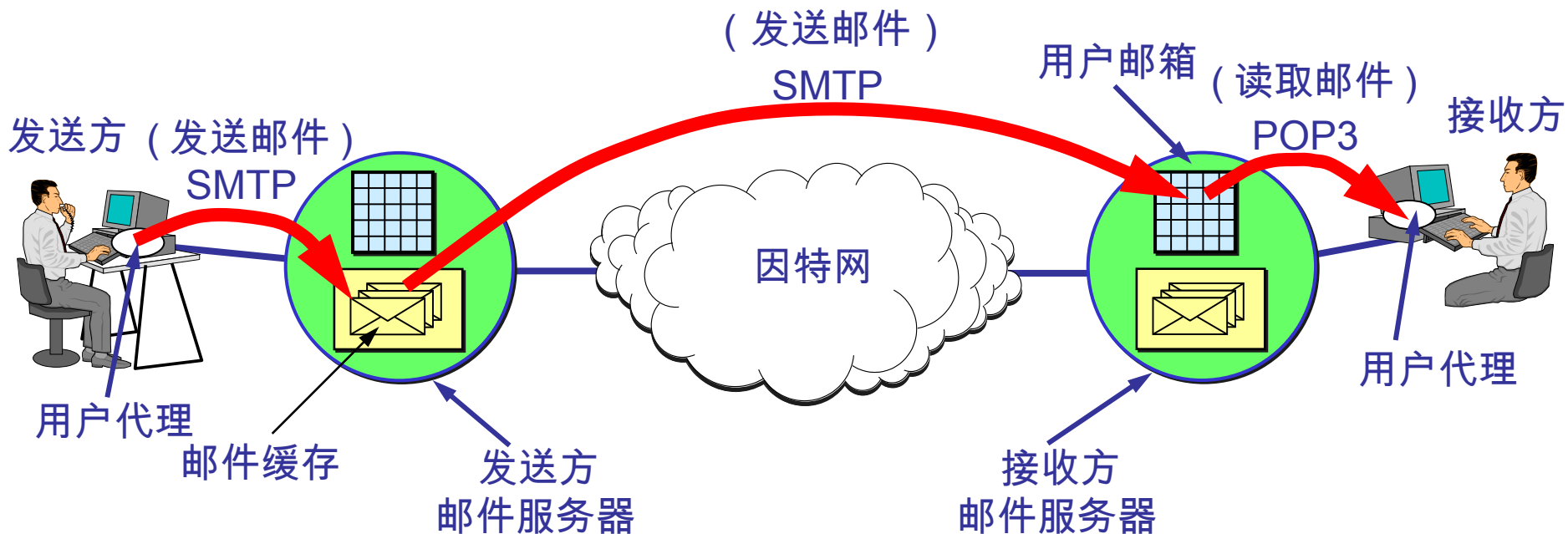


# 电子邮件的一些标准

---

- 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。



# 发送和接收电子邮件的几个重要步骤

---

- ① 发件人调用 PC 中的用户代理撰写和编辑要发送的邮件。
- ② 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，
- ③ SMTP 服务器把邮件临时存放在邮件缓存队列中，等待发送。
- ④ 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 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。



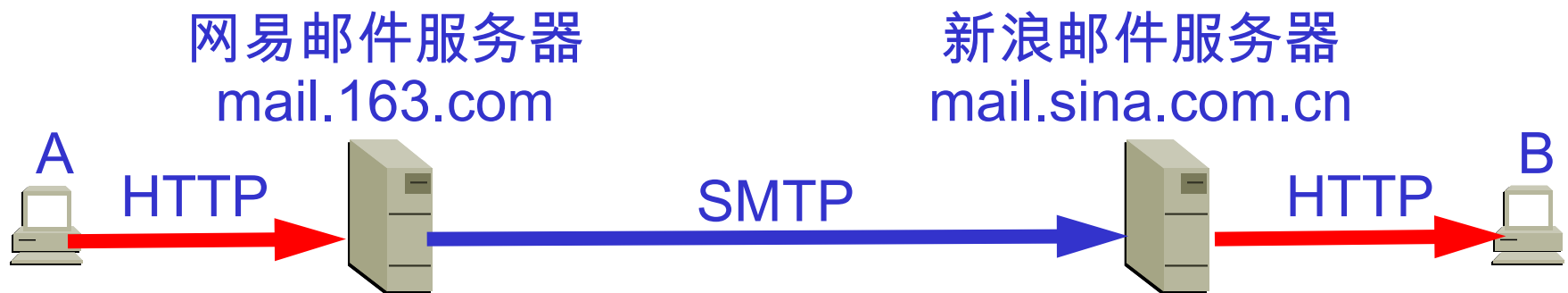
# 必须注意

---

- 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。



# base64 编码表

数值	字符	数值	字符	数值	字符	数值	字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

# ASCII 码表

ASCII表																										
( American Standard Code for Information Interchange 美国标准信息交换代码 )																										
高四位   低四位		ASCII控制字符												ASCII打印字符												
		0000						0001						0010		0011		0100		0101		0110		0111		
		0						1						2		3		4		5		6		7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	►	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◄	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☹	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	!!	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	—	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↕	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	▣	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◉	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	└	^_	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	🎵	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	🎵	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	␣	^Backspace 代码: DEL
注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。																										
课件制作人																										

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。

课件制作人：谢希仁



# 例题

---

- P277

- 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。





# 例题

---

P278



# 作业

---

- P296
- 6-04
- 6-10
- 6-18
- 6-22
- 6-27



## 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。
- 这种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 ) , 这时该主机就成为 DHCP 客户。
  - 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。若找不到, 则从服务器的 IP 地址池 (address pool) 中取一个地址分配给该计算机。DHCP 服务器的回答报文叫做提供报文 ( DHCPOFFER ) 。



# DHCP 中继代理 (relay agent)

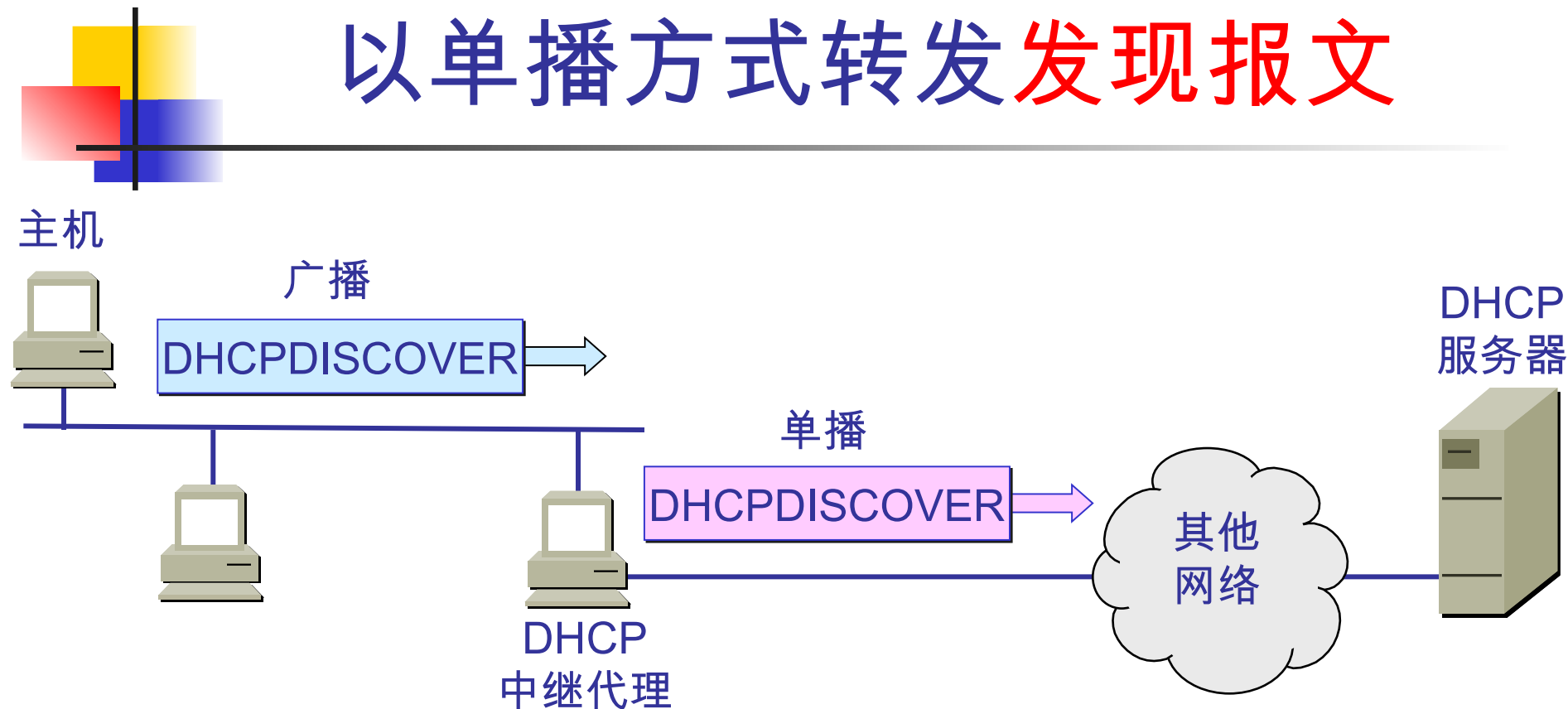
---

- 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。

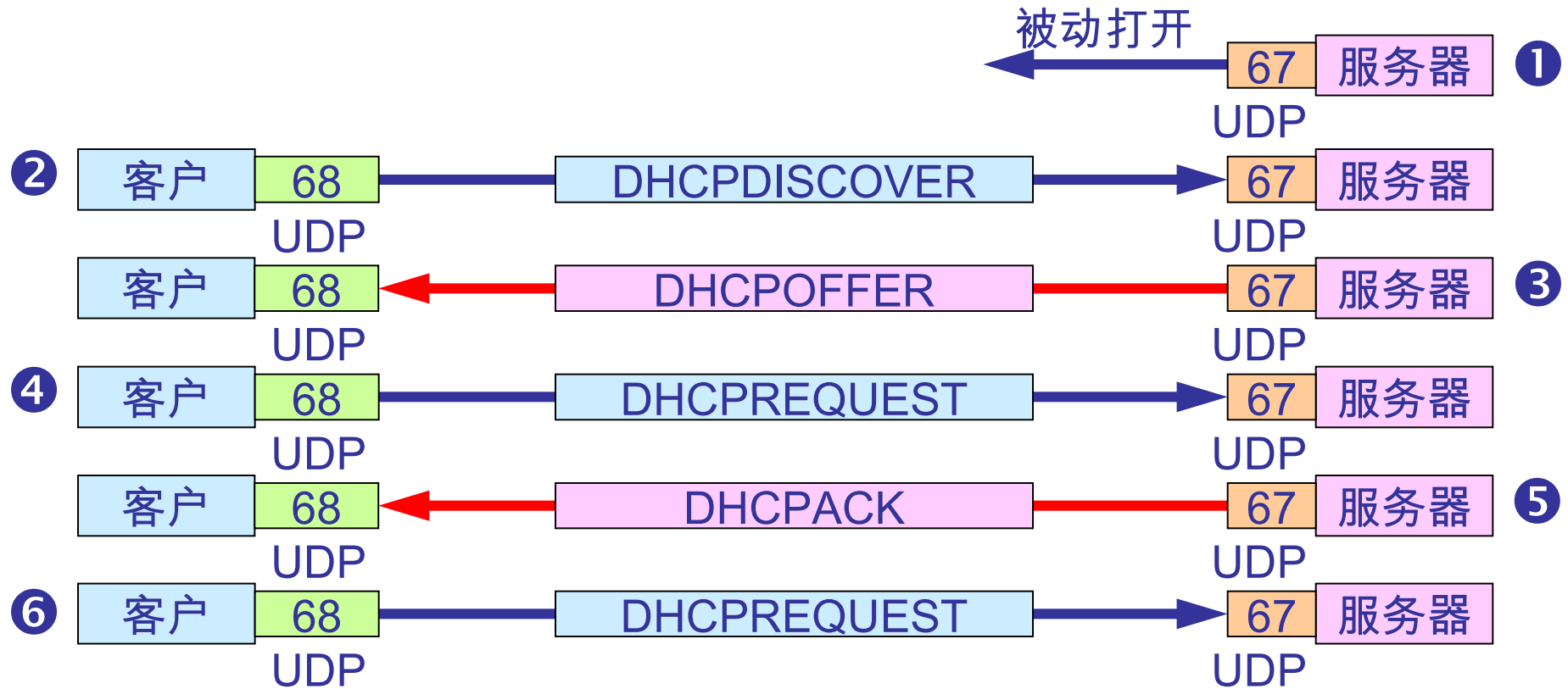


# 租用期 (lease period)

---

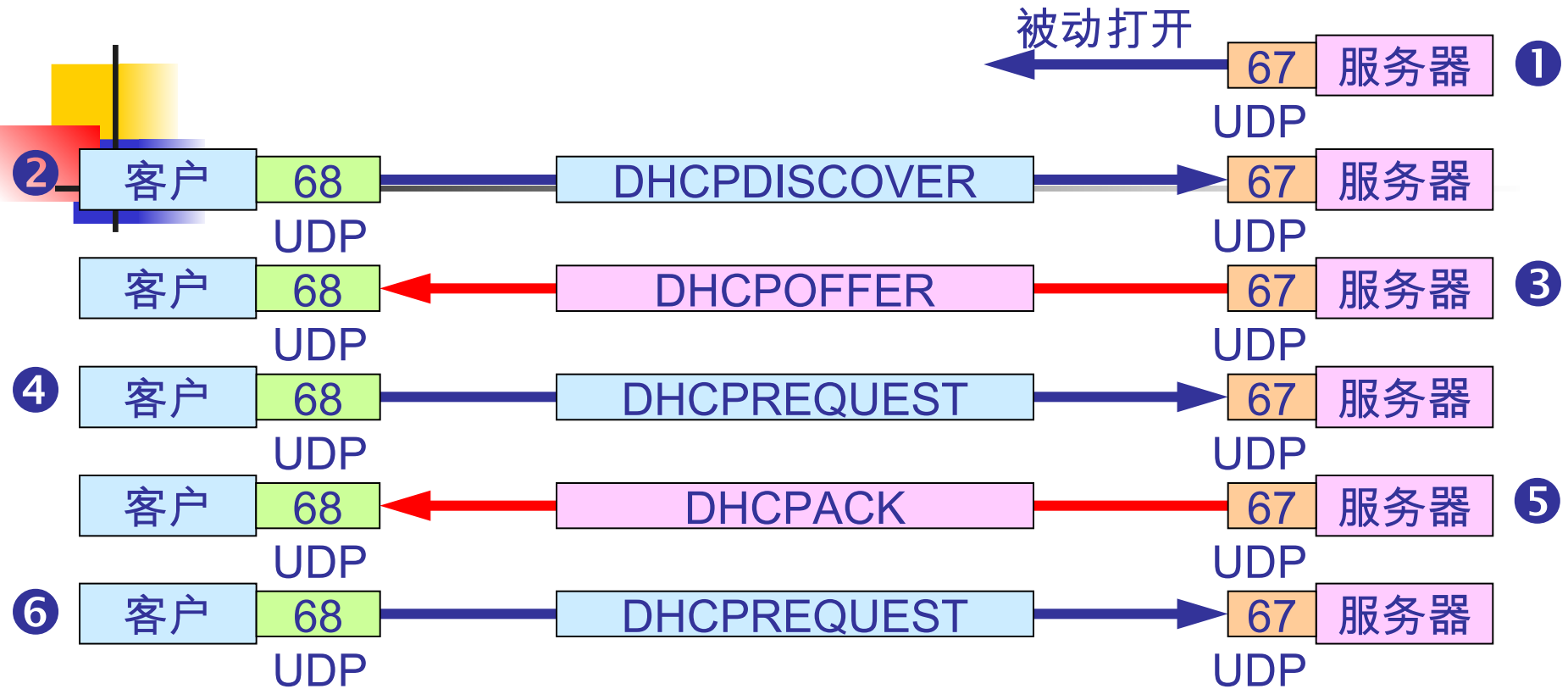
- DHCP 服务器分配给 DHCP 客户的 IP 地址是临时的，因此 DHCP 客户只能在一段有限的时间内使用这个分配到的 IP 地址。DHCP 协议称这段时间为**租用期**。
- 租用期的数值应由 DHCP 服务器自己决定。
- DHCP 客户也可在自己发送的报文中（例如，发现报文）提出对租用期的要求。

# DHCP 协议的工作过程



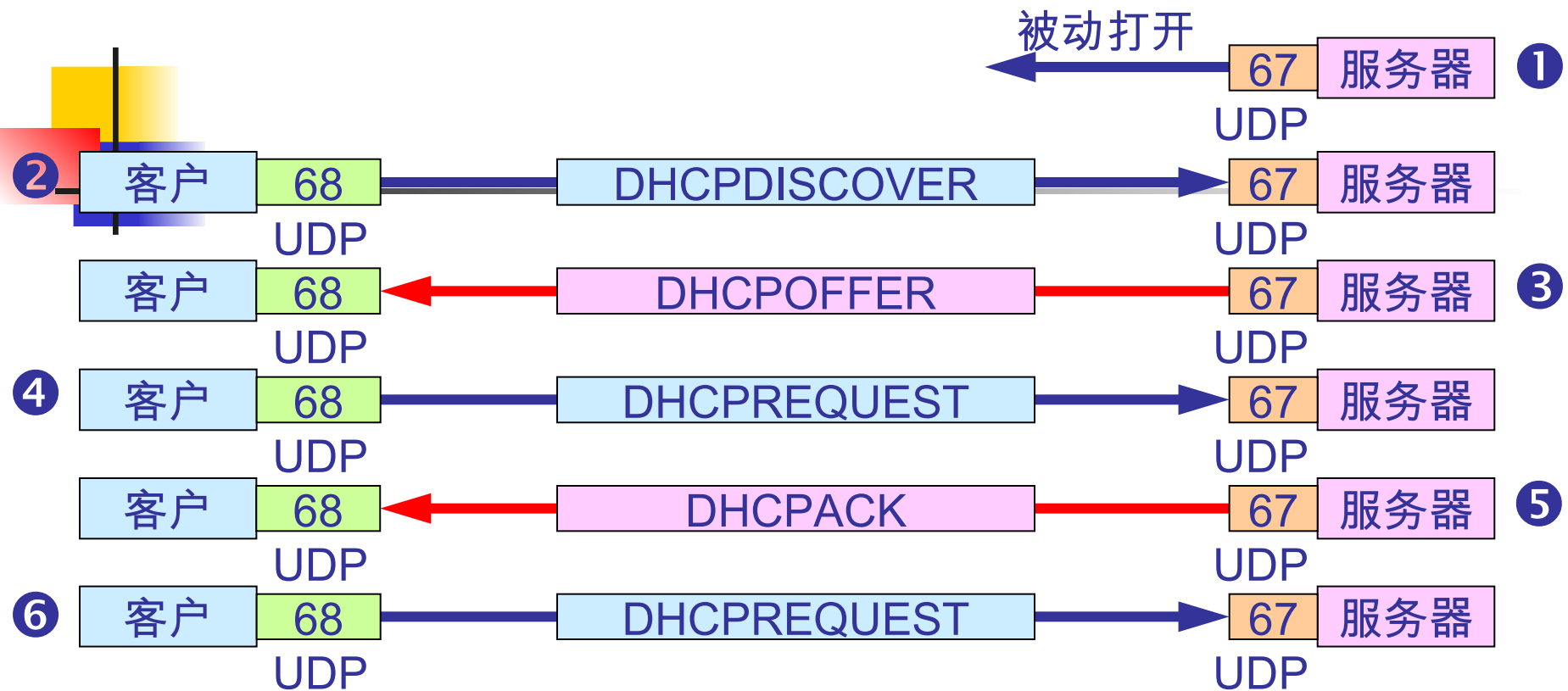
**①** : DHCP 服务器被动打开 UDP 端口 67 , 等待客户端发来的报文。

# DHCP 协议的工作过程



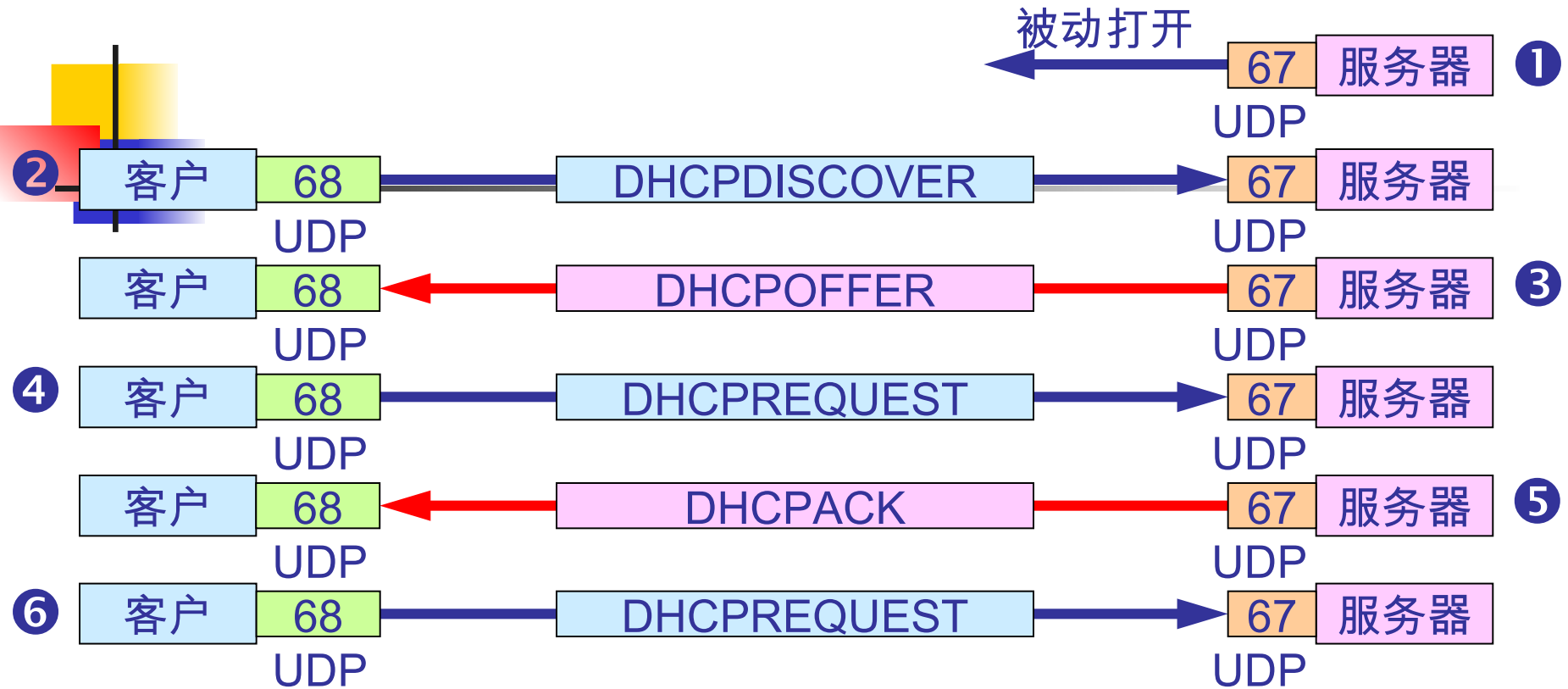
**②** : DHCP 客户从 UDP 端口 68 发送 DHCP 发现报文。

# DHCP 协议的工作过程



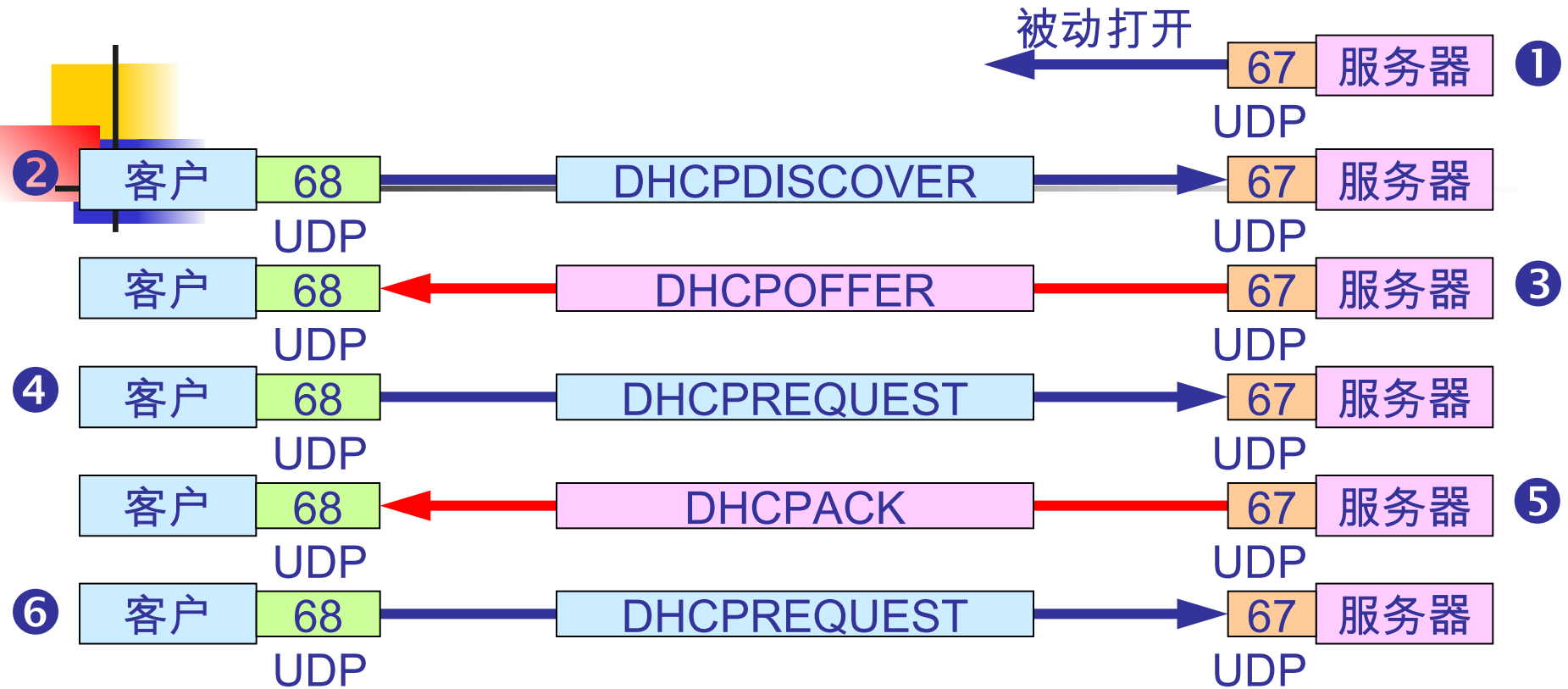
**③** : 凡收到 DHCP 发现报文的 DHCP 服务器都发出 DHCP 提供报文，因此 DHCP 客户可能收到多个 DHCP 提供报文。

# DHCP 协议的工作过程



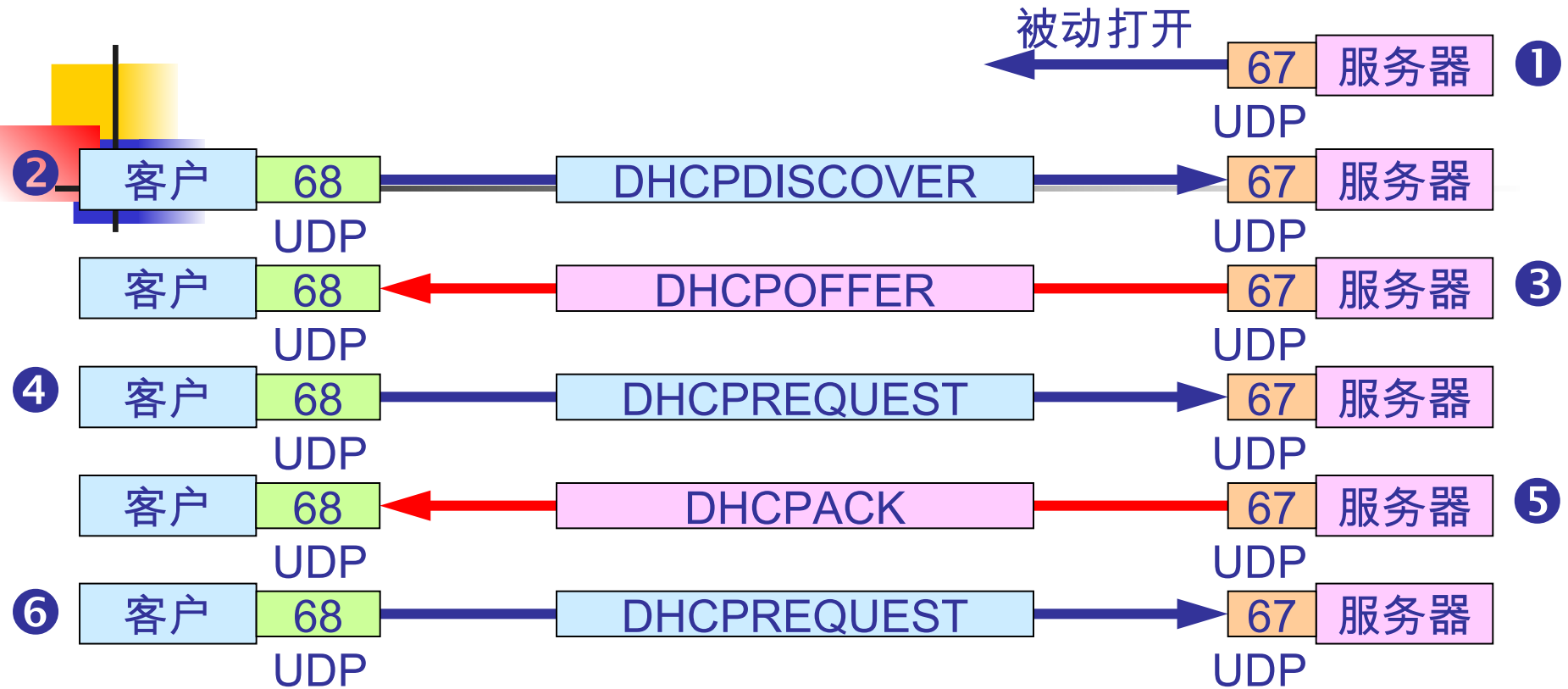
**④** : DHCP 客户从几个 DHCP 服务器中选择其中的一个，并向所选择的 DHCP 服务器发送 DHCP 请求报文。

# 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。

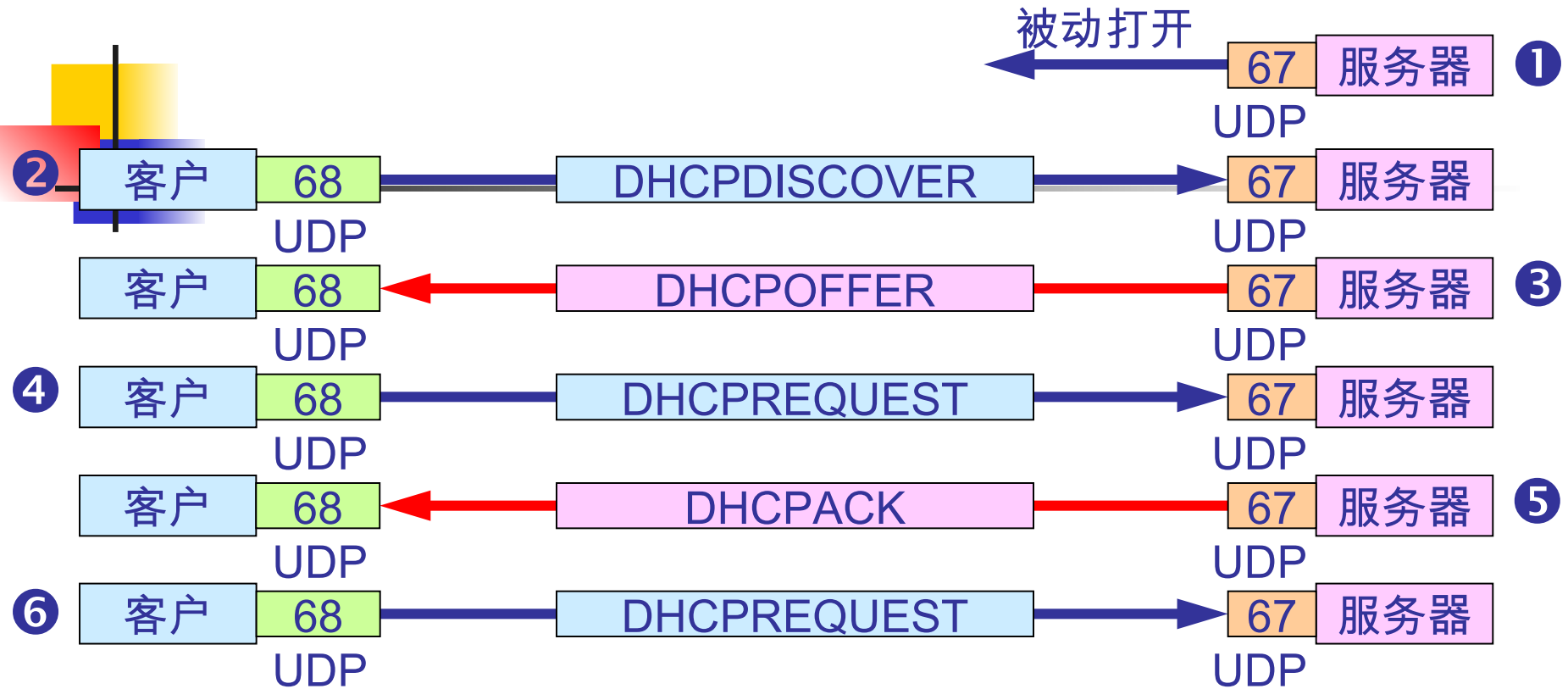
# DHCP 协议的工作过程



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。



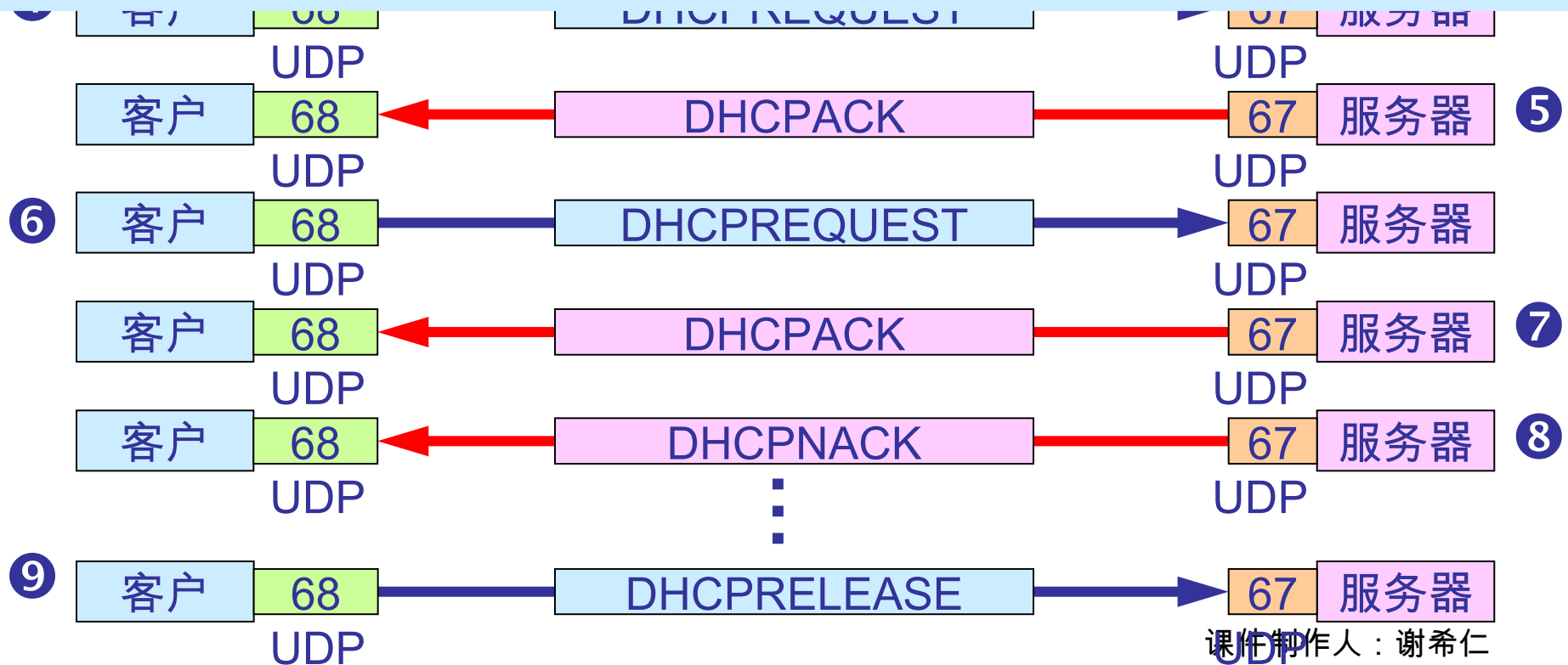
# DHCP 协议的工作过程



**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。

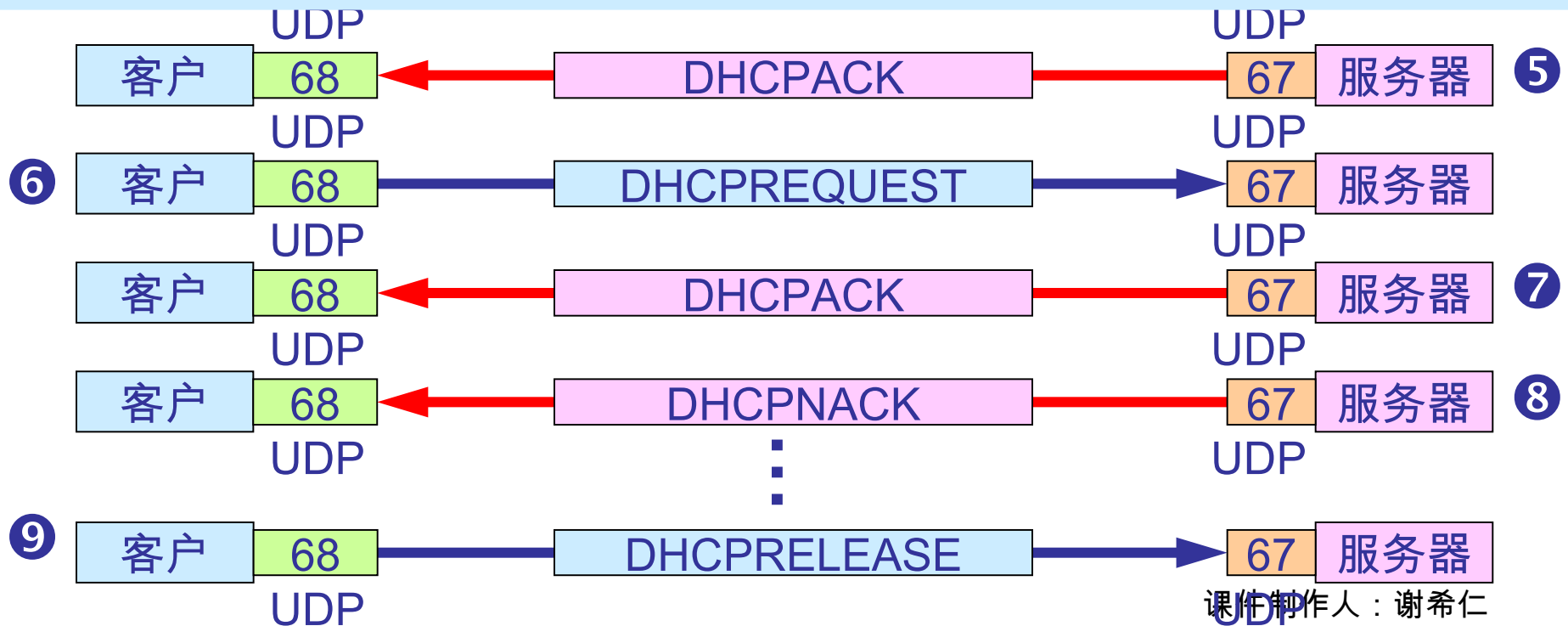
# DHCP 协议的工作过程

**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。



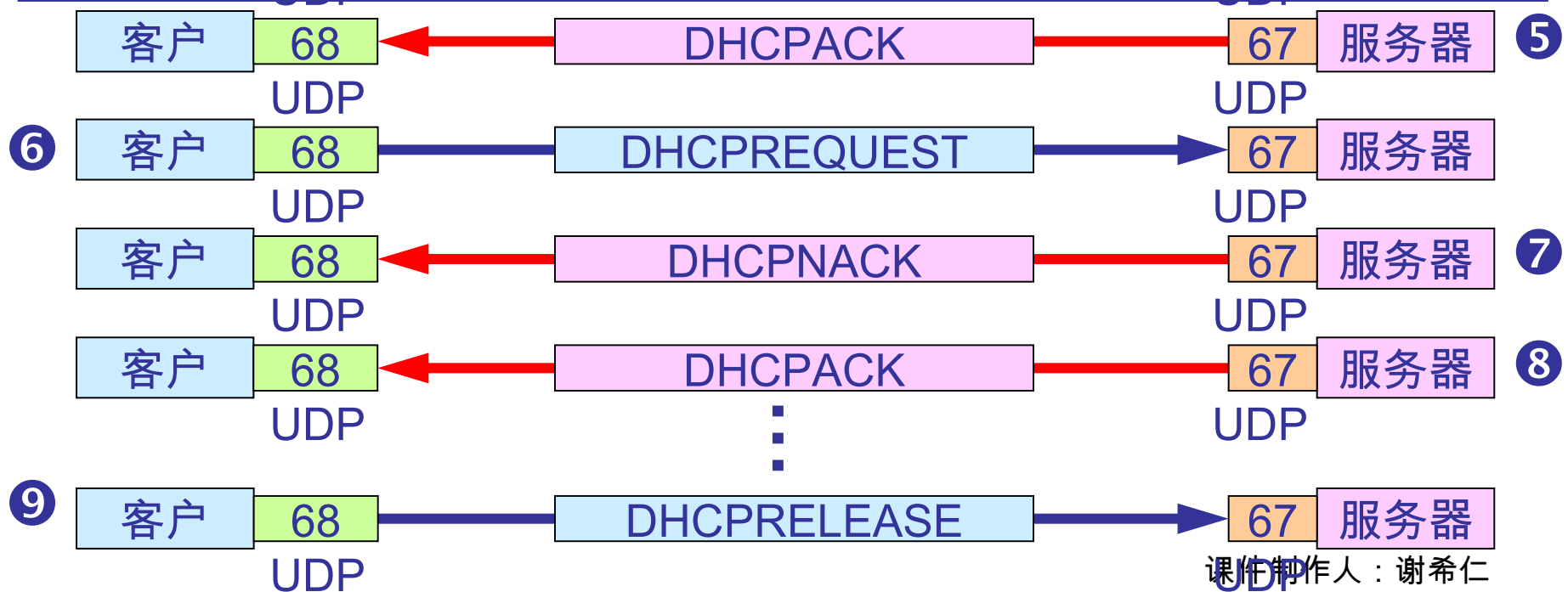
# DHCP 协议的工作过程

**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**)。



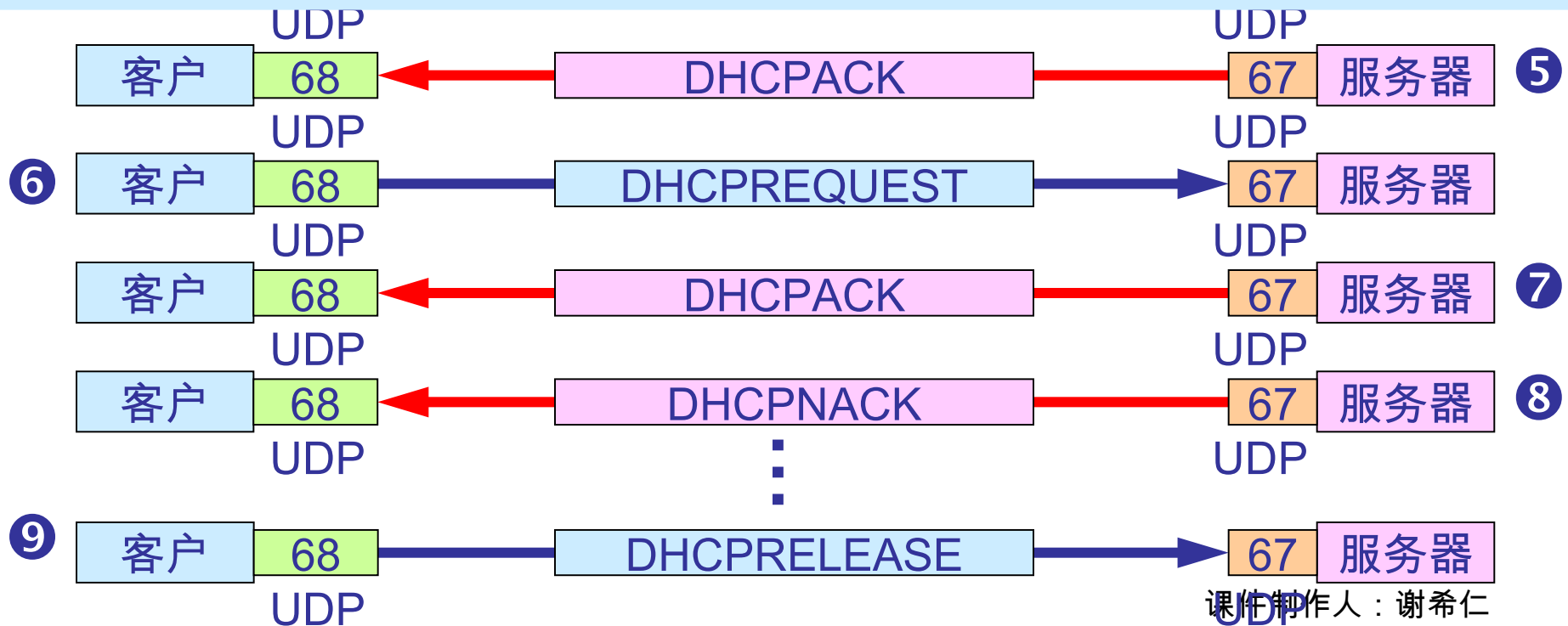
# DHCP 协议的工作过程

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。

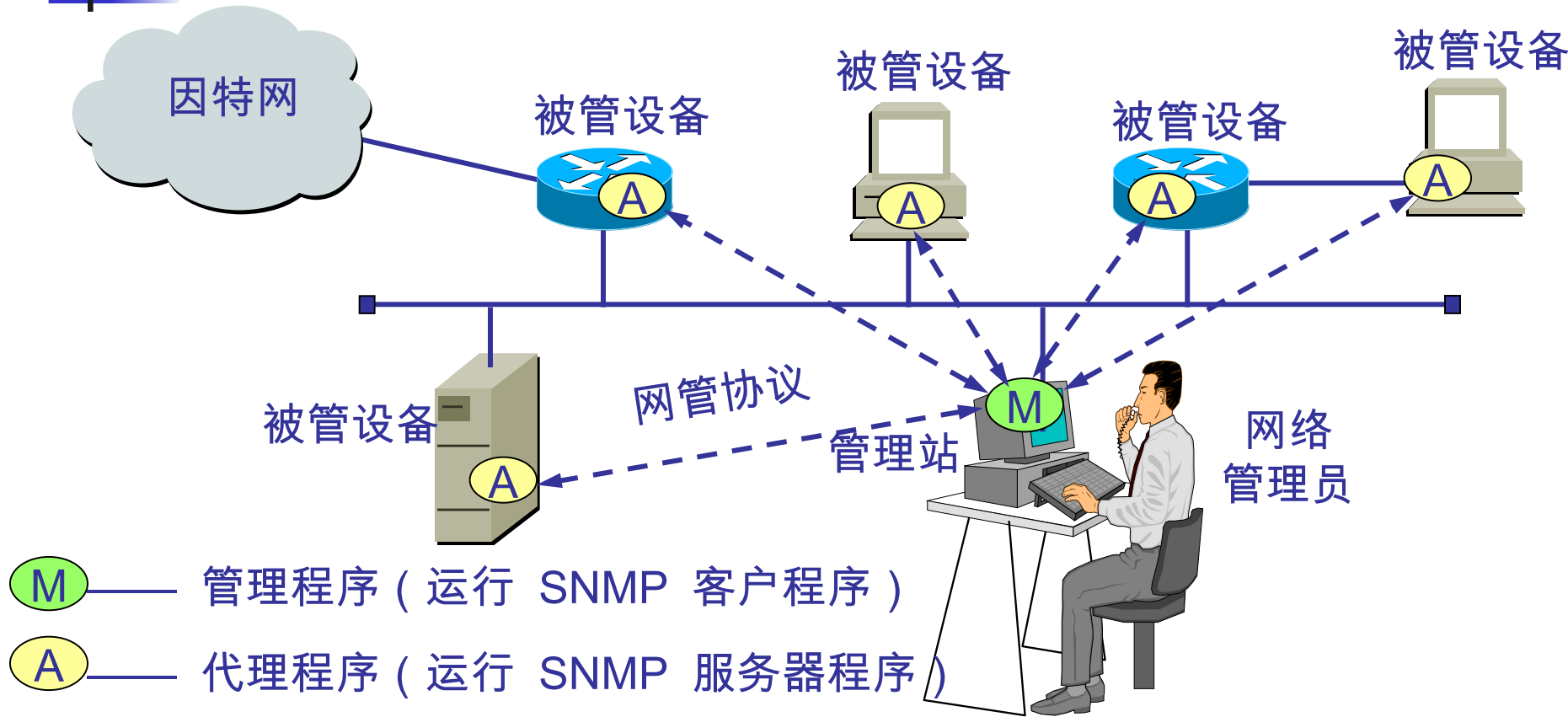


# 6.7 简单网络管理协议 SNMP

## 6.7.1 网络管理的基本概念

- **网络管理**包括对硬件、软件和人力的使用、综合与协调，以便对网络资源进行监视、测试、配置、分析、评价和控制，这样就能以合理的价格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)。
- Manager 不是指人而是指机器或软件。
- 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。
- 需要注意的是，并不是网管协议本身来管理网络。网管协议就是管理程序（ manager ）和代理程序 (agent) 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的管理。



# 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。



# 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（比如 agent），但这种“添加”必须对原有对象的影响尽量小些。



# 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

- 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

---

- MIB 在被管理的实体中创建了命名对象，并规定了其类型。

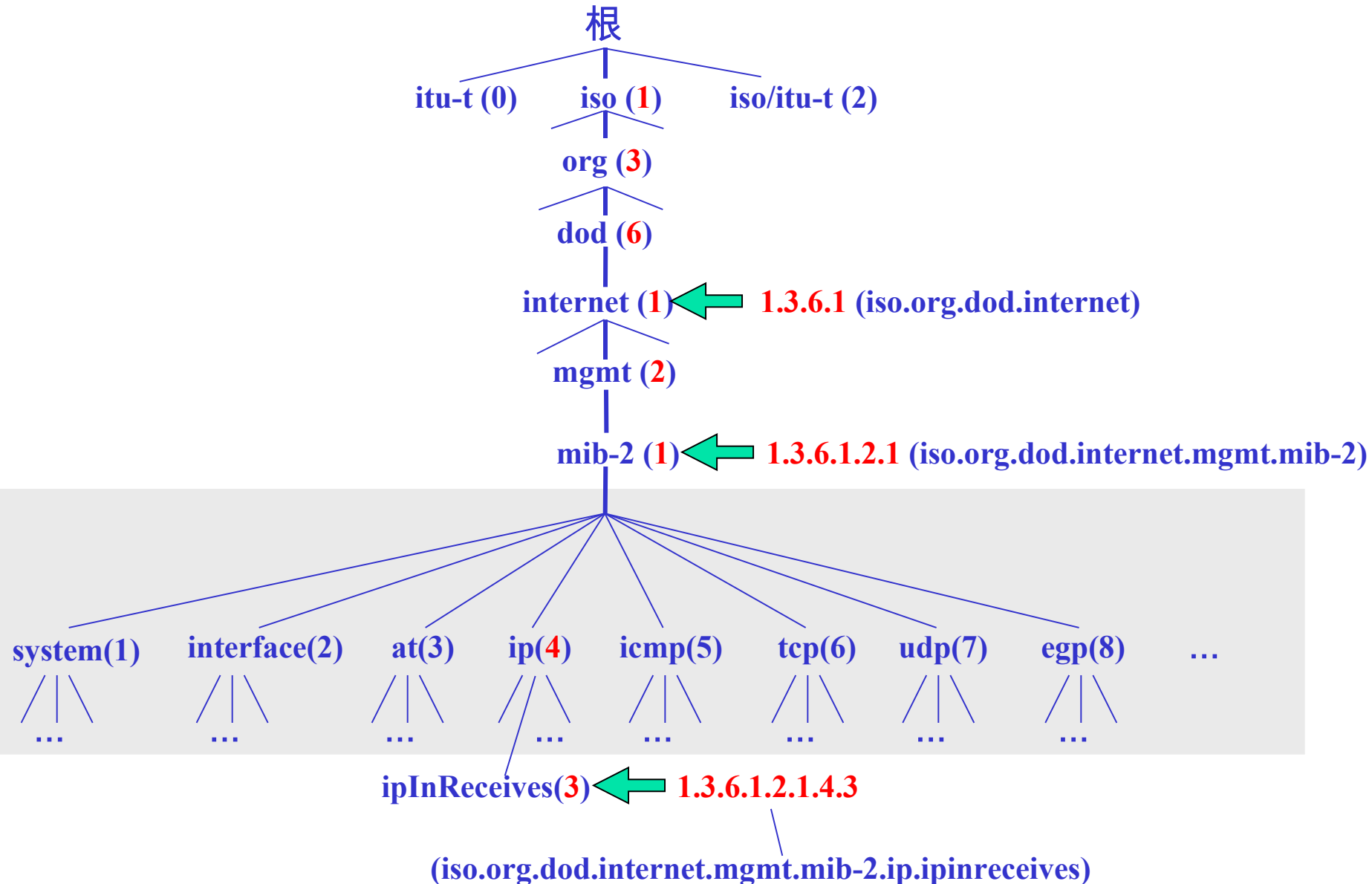


## 6.7.2 管理信息结构 SMI

---

- SMI 的功能：
  - (1) 被管对象应怎样命名；
  - (2) 用来存储被管对象的数据类型有哪些种；
  - (3) 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。
- SMI 既是 ASN.1 的子集，又是 ASN.1 的超集。
- ASN.1 的**记法很严格**，它使得数据的含义不存在任何可能的**二义性**。
- SMI 把**数据类型**分为两大类：简单类型和结构化类型。



# 例题

---

- P285
- 简单类型和结构化类型
- 表 6-3



# 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。





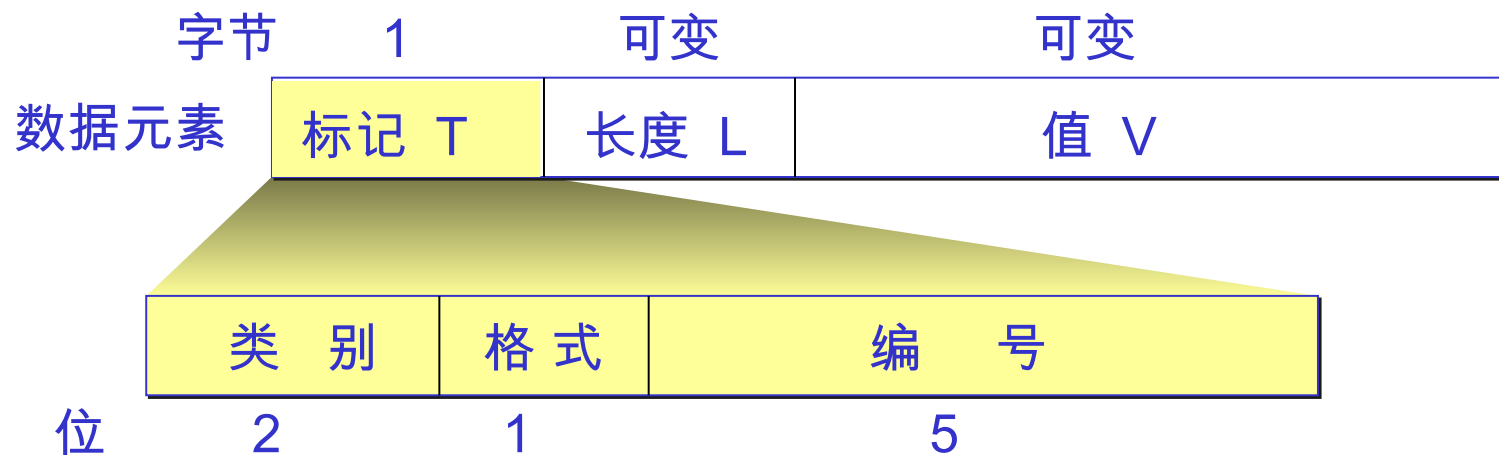
# 用 TLV 方法进行编码

---

- 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 ( Tag )**。
  - (2) L 字段，即长度八位位组 (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**。

# TLV 中的 T 字段

## 定义数据的类型





# 例题

---

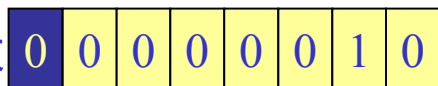
- P286
- 表 6-4

# TLV 中的 L 字段

## 定义 V 字段的长度

指出 V 字段长度 = 2 字节

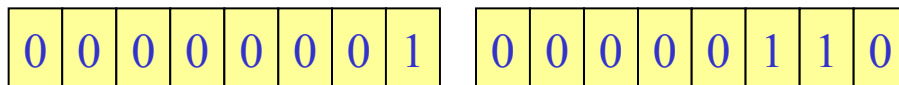
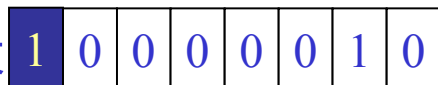
单字节的 L 字段



指出后续字节数 = 2

指出 V 字段长度 = 262 字节

多字节的 L 字段



← 后续字节数 = 2 →

# 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。



## 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）。



# 例题

---

- P287

- 表 6-5 : 信息类别
- 表 6-6 : MIB 变量



## 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。
- 但探测管理协议不够灵活，而且所能管理的设备数目不能太多。探测系统的开销也较大。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”。
- 这种陷阱信息的参数是受限制的。
- 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**）。过滤的好处是：
  - 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 是有效的网络管理协议

---

- 使用探询（至少是周期性地）以维持对网络资源的实时监视，同时也采用陷阱机制报告特殊事件，使得 SNMP 成为一种有效的网络管理协议。



# 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 ( 无编号 4 )



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	Response	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	Report	管理者之间报告某些差错

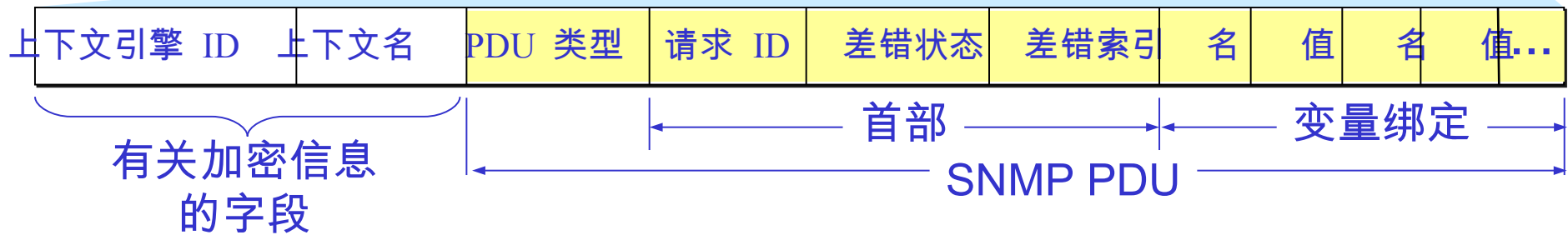
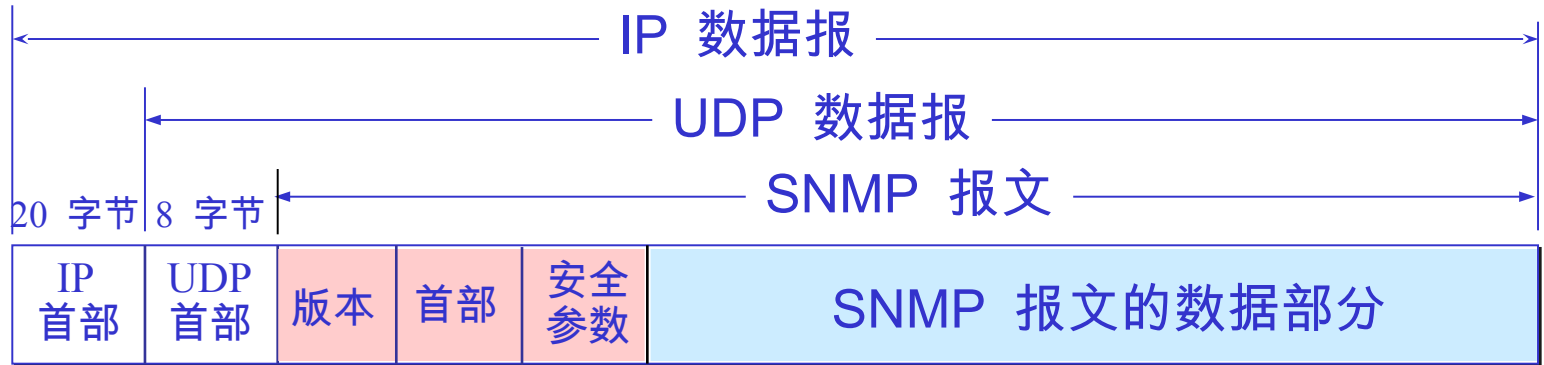


# 例题

---

- P288-289
- T 字段 (A0) 的 A(1010) 的含义
- 10 : 上下文类
- 1 : 结构化数据类型
- 0 : 编号字段的第一个比特

# SNMP 的报文格式



# Get-request 报文 ASN.1 编码

GetRequest-PDU

T A0	L 1D	GetRequest -PDU
---------	---------	--------------------

request-id	error-status	error-index	variable-bindings
------------	--------------	-------------	-------------------

T INTEGER	L 04	V 05 AE 56 02	T INTEGER	L 01	V 00	T INTEGER	L 01	V 00	T SEQUENCE OF	L 0F	VarBind
--------------	---------	------------------	--------------	---------	---------	--------------	---------	---------	------------------	---------	---------

request-ID

T SEQUENCE	L 0D	name	value
---------------	---------	------	-------

T OBJECT IDENTIFIER	L 09	V 01 03 06 01 02 01 01 01 00	T NULL	L 00
------------------------	---------	---------------------------------	-----------	---------

1. 3. 6. 1. 2. 1. 7. 1. 0





# 作业

---

- P298
- 6-41
- 注：设请求 ID 的十六进制值为
- ( 00010614 )<sub>16</sub>



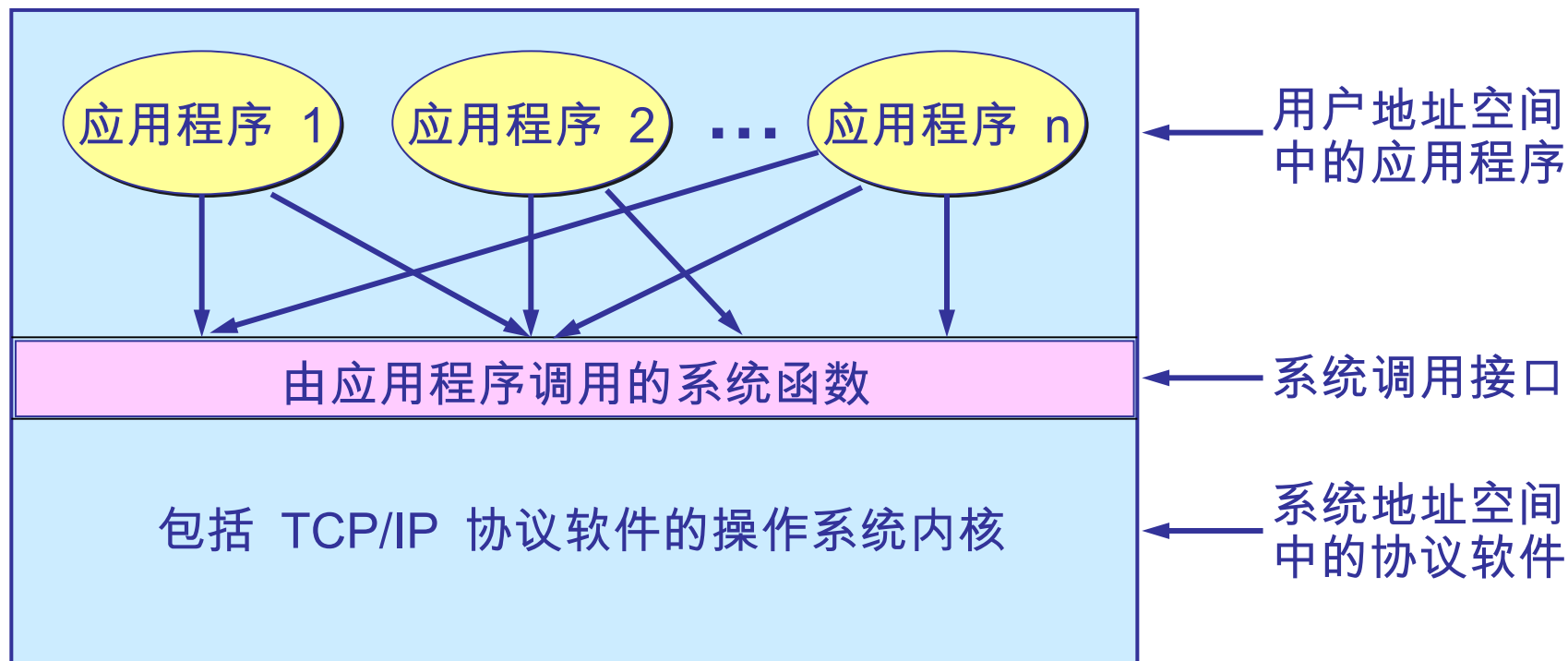
## 6.8 应用进程跨越网络的通信

### 6.8.1 系统调用和应用编程接口

---

- 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。

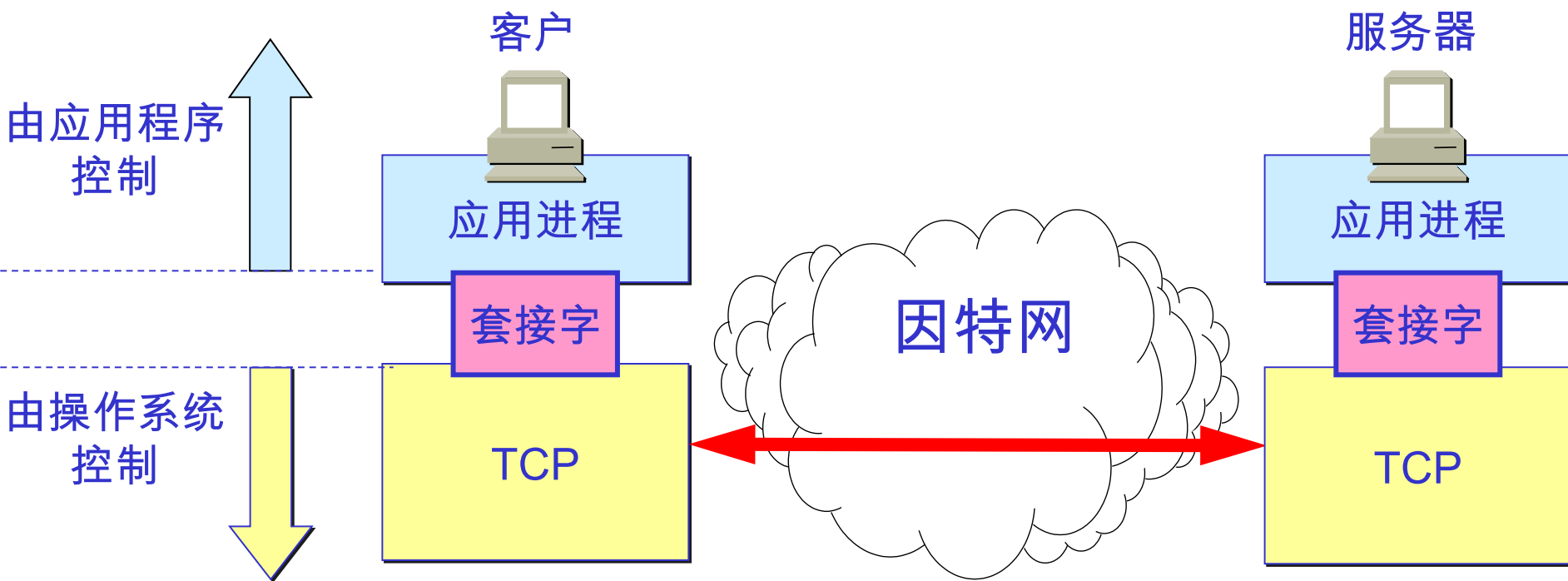


# 几种应用编程接口 API

---

- 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，简称为 TLI (Transport Layer Interface)。

# 应用进程通过套接字接入到网络





# 套接字的作用

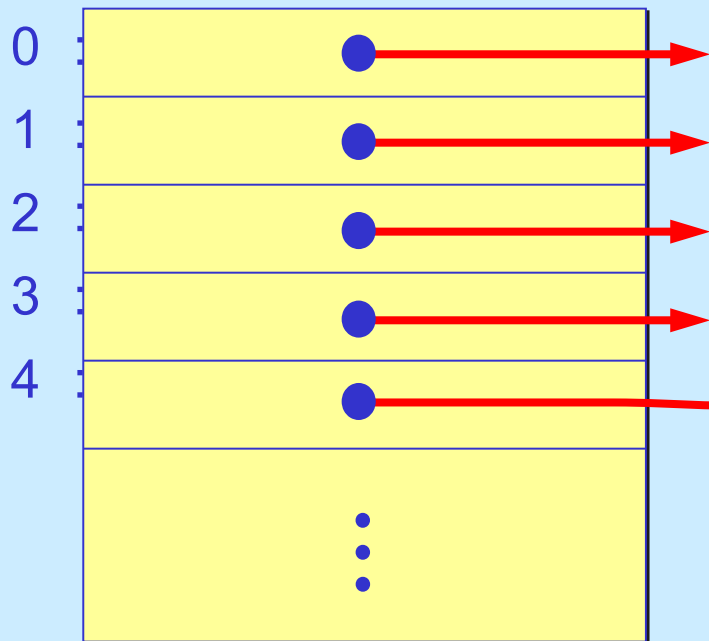
---

- 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：
⋮



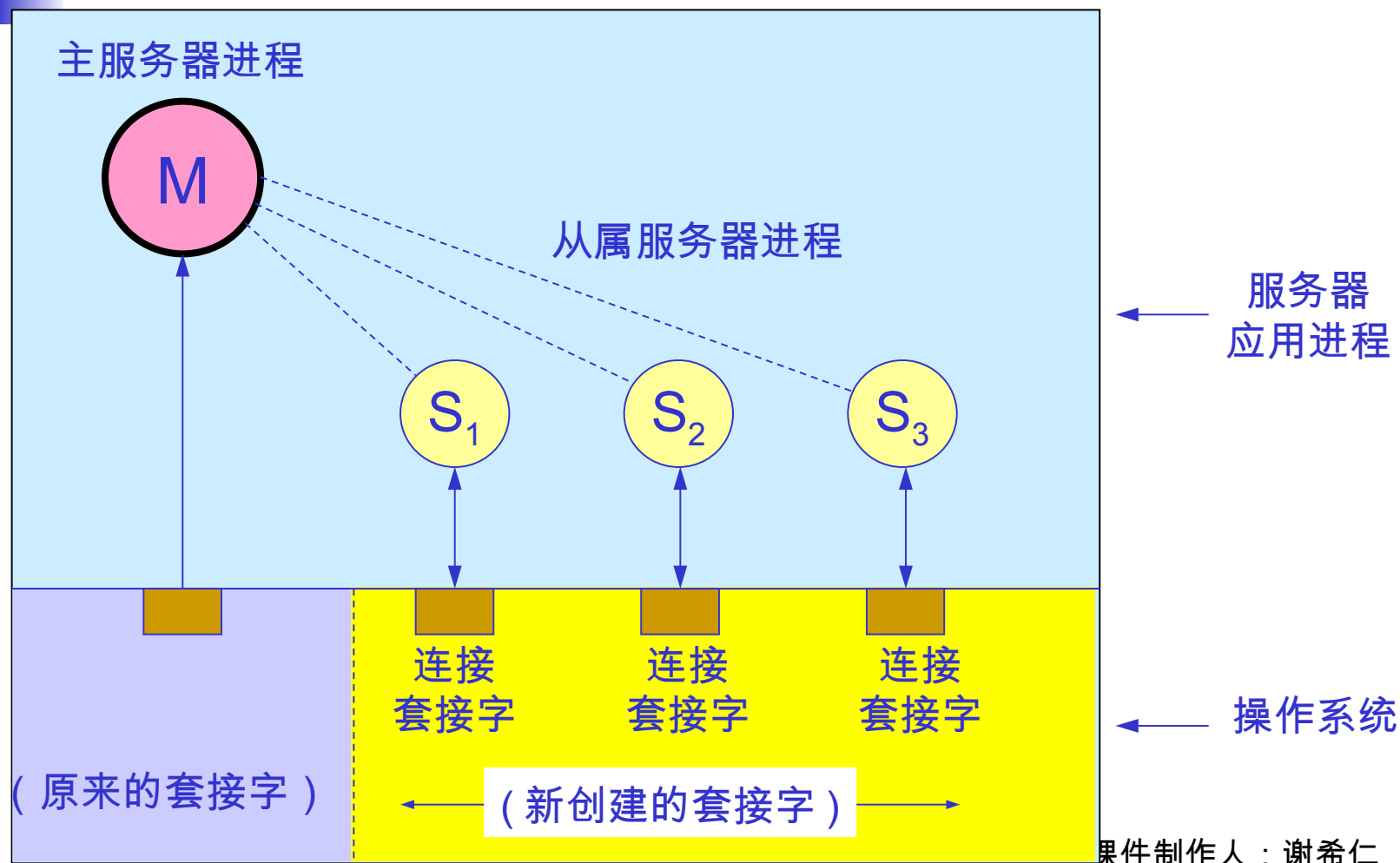
## 6.8.2 几种常用的系统调用

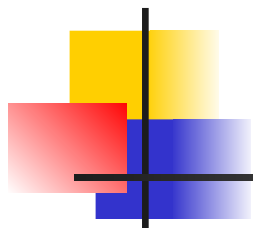
### 1. 连接建立阶段

- 当套接字被创建后，它的端口号和 IP 地址都是空的，因此应用进程要调用 **bind**（**绑定**）来指明套接字的本地地址。在服务器端调用 **bind** 时就是把熟知端口号和本地 IP 地址填写到已创建的套接字中。这就叫做把**本地地址绑定到套接字**。
- 服务器在调用 **bind** 后，还必须调用 **listen**（**收听**）把套接字设置为被动方式，以便随时接受客户的服务请求。UDP 服务器由于只提供无连接服务，不使用 **listen** 系统调用。
- 服务器紧接着就调用 **accept**（**接受**），以便把远地客户进程发来的连接请求提取出来。系统调用 **accept** 的一个变量就是要指明从哪一个套接字发起的连接。

## 6.8.2 几种常用的系统调用

### 并发方式工作的服务器





# 系统调用使用顺序的例子

