

第2章 应用层

© 电子科技大学信息与软件工程学院
计算机网络课程组, 2023



网络应用及应用 层协议的概念、实现

传输层服务模型
客户服务器模式
对等模式peer-to-peer



通过对主流应用层协 议分析来掌握应用层 协议

HTTP

FTP

SMTP / POP3 / IMAP

DNS

网络应用编程

socket API



目录 CONTENT

01 应用层协议原理

02 Web应用和 HTTP协议

03 文件传输协议：FTP

04 因特网中的电子邮件
SMTP, POP3, IMAP

05 DNS：因特网的目录服务

06 P2P 文件共享

07 TCP套接字编程

08 UDP套接字编程

09 构造一个简单的Web服务器

01 应用层协议原理



一些网络应用

电子邮件

Web

即时讯息

远程登陆

P2P文件共享

多用户网络游戏

流式存储视频片段

因特网电话

实时视频会议

在两台计算机之间的两个
帐户之间的文件传输

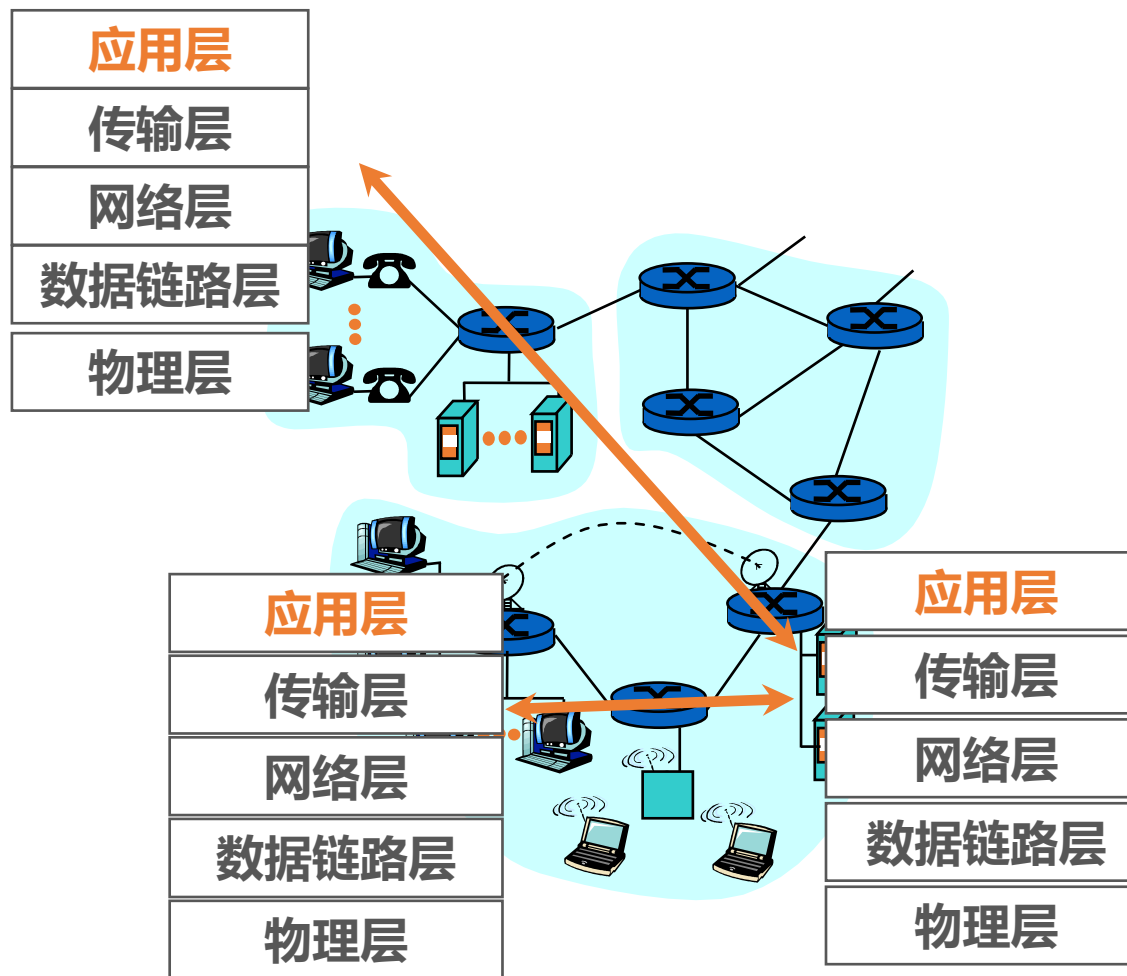
研发网络应用程序

研发网络应用程序的核心

- 写出能够运行在不同的端系统并通过网络彼此通信的程序
- 例如，Web：Web服务器软件和浏览器软件通信

没有应用程序软件 运行在网络核心设备上

- 网络核心设备不在应用层起作用
- 这种设计方法促进了应用程序的研发



网络应用程序体系结构

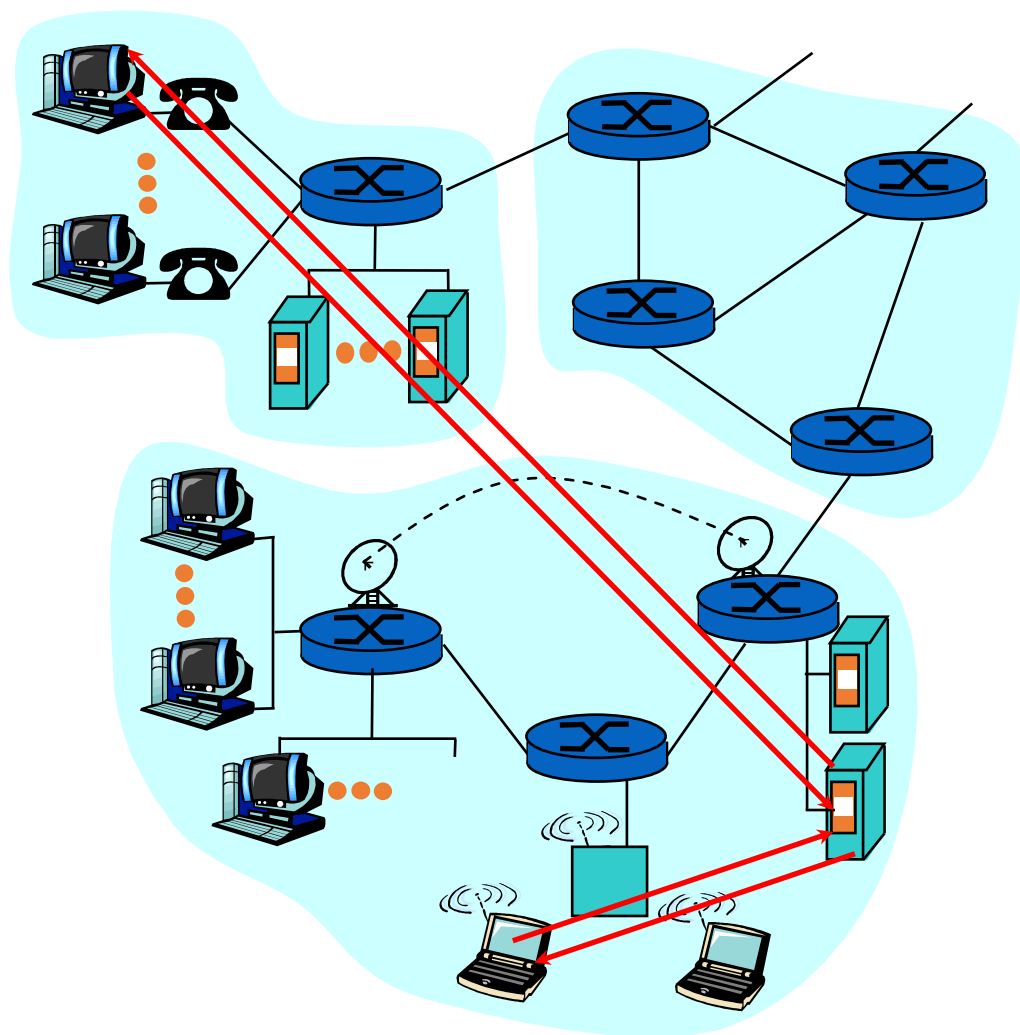


客户机/服务器体系结构

P2P体系结构

客户机/服务器和P2P混合的
体系结构

客户机/服务器体系结构



服务器

- 总是打开的主机
- 具有固定的、众所周知的IP地址
- 主机群集常被用于创建强大的虚拟服务器

客户机

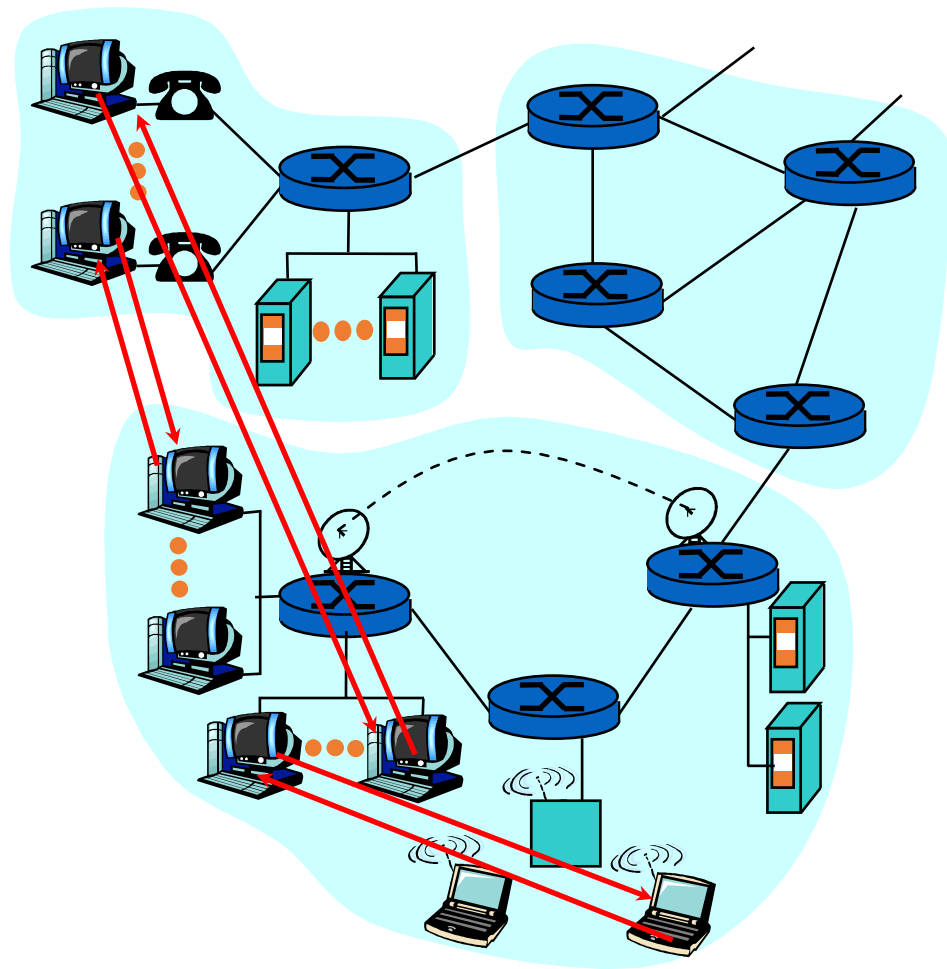
- 同服务器端通信
- 可以间断的同服务器连接
- 可以拥有动态IP地址
- 客户机相互之间不直接通信

纯P2P体系结构

- 没有总是打开的服务器
- 任意一对主机直接相互通信
- 对等方间歇连接并且可以改变IP地址
- 例如：Gnutella

优点：自扩展性

缺点：难以管理



客户机/服务器和P2P混合的体系结构

Napster

文件直接在对等方之间交换
文件搜索通过服务器
中心服务器记录对等方内容
对等方查询中心服务器来决定要求的文件位置



即时讯息

两个聊天用户之间是P2P
注册、查询通过服务器
用户上线时要在中心服务器上注册
用户与中心服务器联系以找出在线伙伴



区块链&比特币&.....



进程通信

- 进程：运行在端系统中的程序
- 同一主机上的两个进程通过内部进程通信机制进行通信
- 不同主机上的进程通过交换报文相互通信

客户进程：

发起通信的进程

服务器进程：

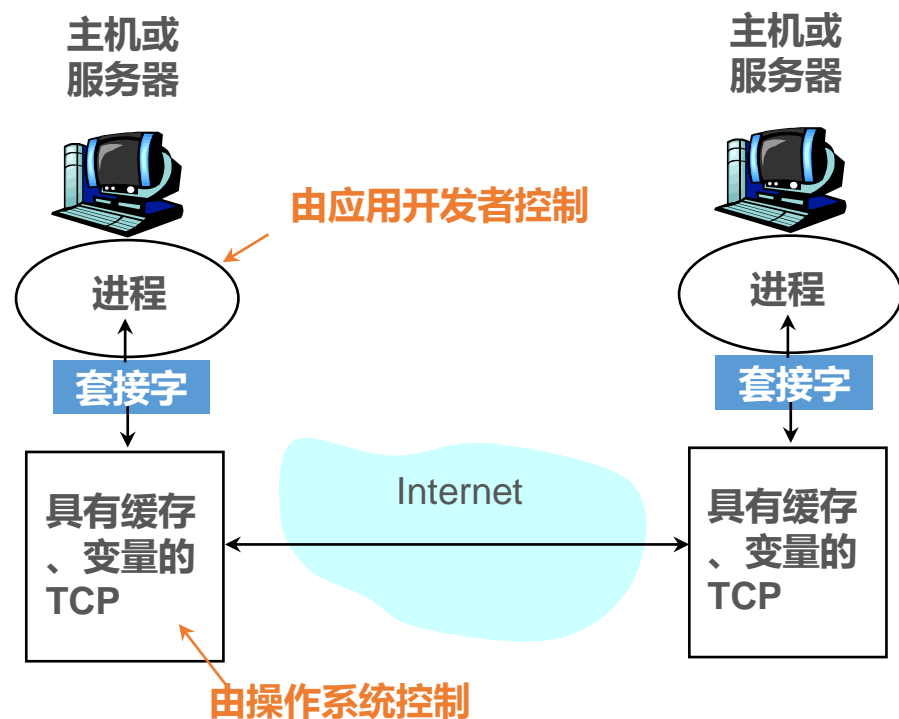
等待联系的进程

- 注意：具有P2P体系结构的应用程序既有客户进程和服务进程。

进程与计算机网络的接口-套接字

- ◆ 进程通过它的套接字在网络上发送和接收报文
- ◆ 套接字类比于门户
 - 发送进程把报文推出门户
 - 发送进程假定门户到另外一侧之间有运输设施，该设施可以传送报文到接收进程

- ◆ 套接字又叫做应用程序编程接口API
- ◆ 用户通过API对传输层的控制仅限于：
 - (1) 选择传输协议;
 - (2) 能设定几个参数



进程寻址

- 为了一个进程能接收报文，它需要一个标识
- 主机有唯一的32位IP地址
- **问：**主机的IP地址足够标识进程吗？
- **答：**不能。因为一台主机上能够运行许多进程。

- 主机上的进程标识包括IP地址和**端口号**
- 常用应用程序的端口号：
 - Web服务：80
 - 邮件服务：25
- 将在第三章详细分析端口号

应用层协议

- 交换的报文类型，如请求报文和应答报文
- 报文类型的语法：报文中的各个字段及其详细描述
- 字段的语义，即包含在字段中的信息的含义
- 进程何时、如何发送报文及对报文进行响应

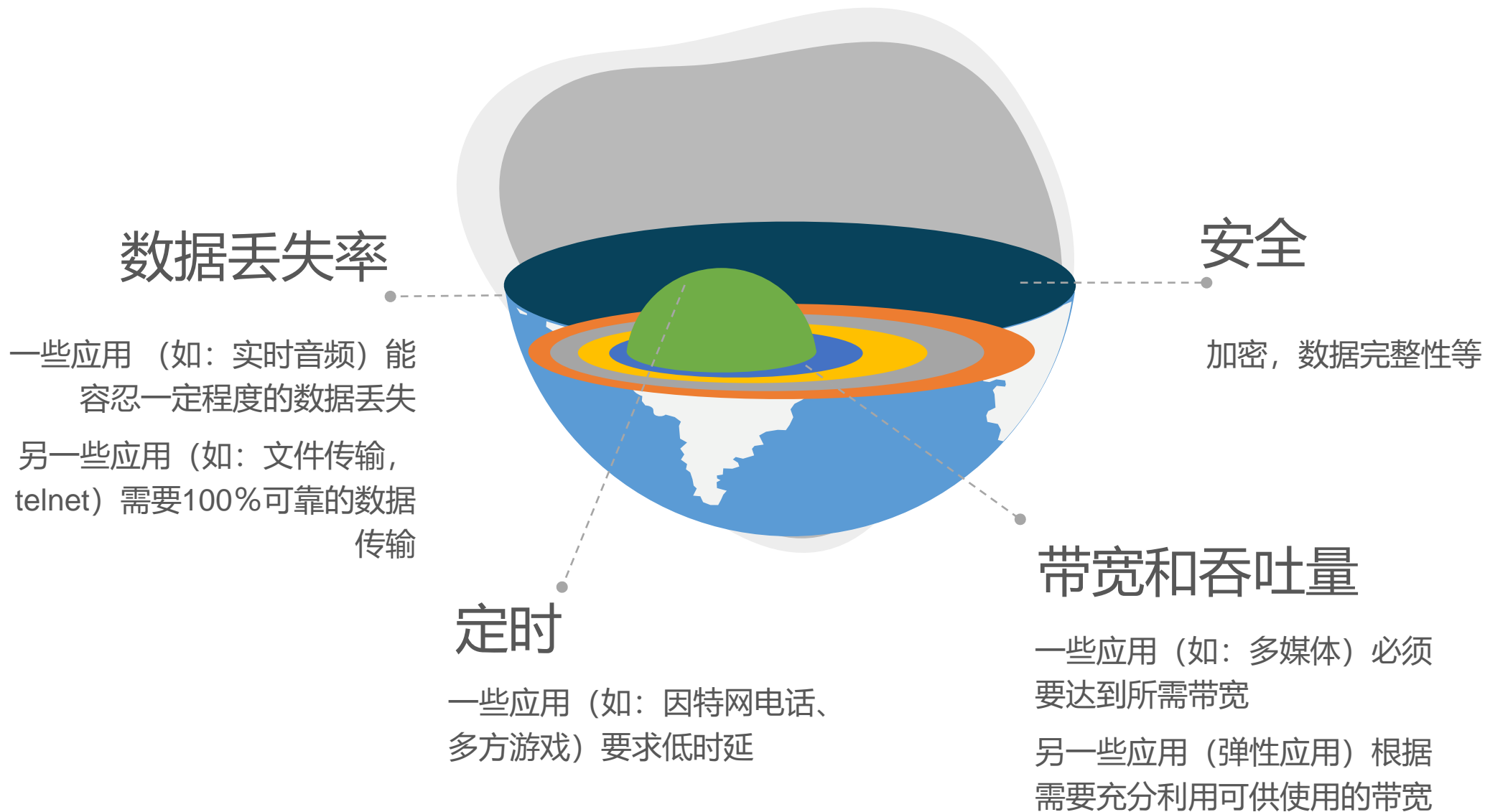
公共领域协议

- 由RFC文档定义
- 可供大家使用
- 例如：HTTP, SMTP

专用协议

- 例如：KaZaA, Skype等

应用需要什么样的服务？



常见应用的传输服务需求

应用	数据丢失	带宽	时间敏感
文件传输	不能丢失	弹性	不
电子邮件	不能丢失	弹性	不
Web	不能丢失	弹性	不
实时音频/视频	容忍丢失	音频: 几kbps-1Mbps	是, 100 msec
存储音频/视频	容忍丢失	视频: 10kbps-5Mbps	是, 几秒
交互式游戏	容忍丢失	视频: 10kbps-5Mbps	是, 100 msec
即时讯息	不能丢失	几 kbps 以上 弹性	是和不是

因特网运输协议提供的服务

TCP服务

- 面向连接的服务：在客户机程序和服务器程序之间必须建立连接
- 可靠的传输服务：接收和发送进程间
- 流量控制：发送方不会淹没接收方
- 拥塞控制：网络出现拥塞时抑制发送进程
- 没有提供：时延保证，最小带宽保证

UDP 服务

- 不可靠数据传输
- 没有提供：建立连接，可靠性，流量控制，拥塞控制，时延和带宽保证

因特网应用：应用层协议，传输协议

应用	应用层协议	下面的传输协议
电子邮件	SMTP [RFC 2821]	TCP
远程终端访问	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流媒体	通常专用 (e.g. RealNetworks)	TCP or UDP
因特网电话	通常专用 (e.g., Skype)	典型用 UDP

安全TCP

- 没有加密
- 网络明文传输，如用户名和口令信息等

TCP和UDP

- 提供加密的TCP连接
- 保证数据完整性
- 端点认证

SSL
(Security Socket Layer)

- 应用使用SSL库调用TCP服务接口
- SSL提供套接字API

SSL在应用层

02 Web应用和HTTP协议



Web和HTTP

常用术语

- 网页（Web页，或称文档）由许多对象组成。
- 对象就是文件，可以是HTML文件, JPEG图像, Java applet, 音频文件...
- 多数网页由单个基本HTML文件和若干个所引用的对象构成
- 每个对象被一个URL(Uniform Resource Locator统一资源定位符)寻址
- 举例URL:

http://www.someschool.edu/someDept/pic.gif

↑
协议

主机名

路径名

Web浏览器 (browser)

- 浏览器的发展史
- 常见浏览器
 - PC端: IE, Chrome, Safari, Firefox, Opera ; 各种马甲浏览器(Edge, QQ, 360, 遨游, 猎豹, 搜狗, Vivaldi, Yandex ,)
 - 手机端: 各手机自带浏览器, UC, ...
- 浏览器内核(Rendering Engine)
 - Trident
 - Gecko
 - Webkit
 - Blink



搜狗浏览器



QQ浏览器



UC浏览器



360浏览器



百度浏览器



猎豹浏览器



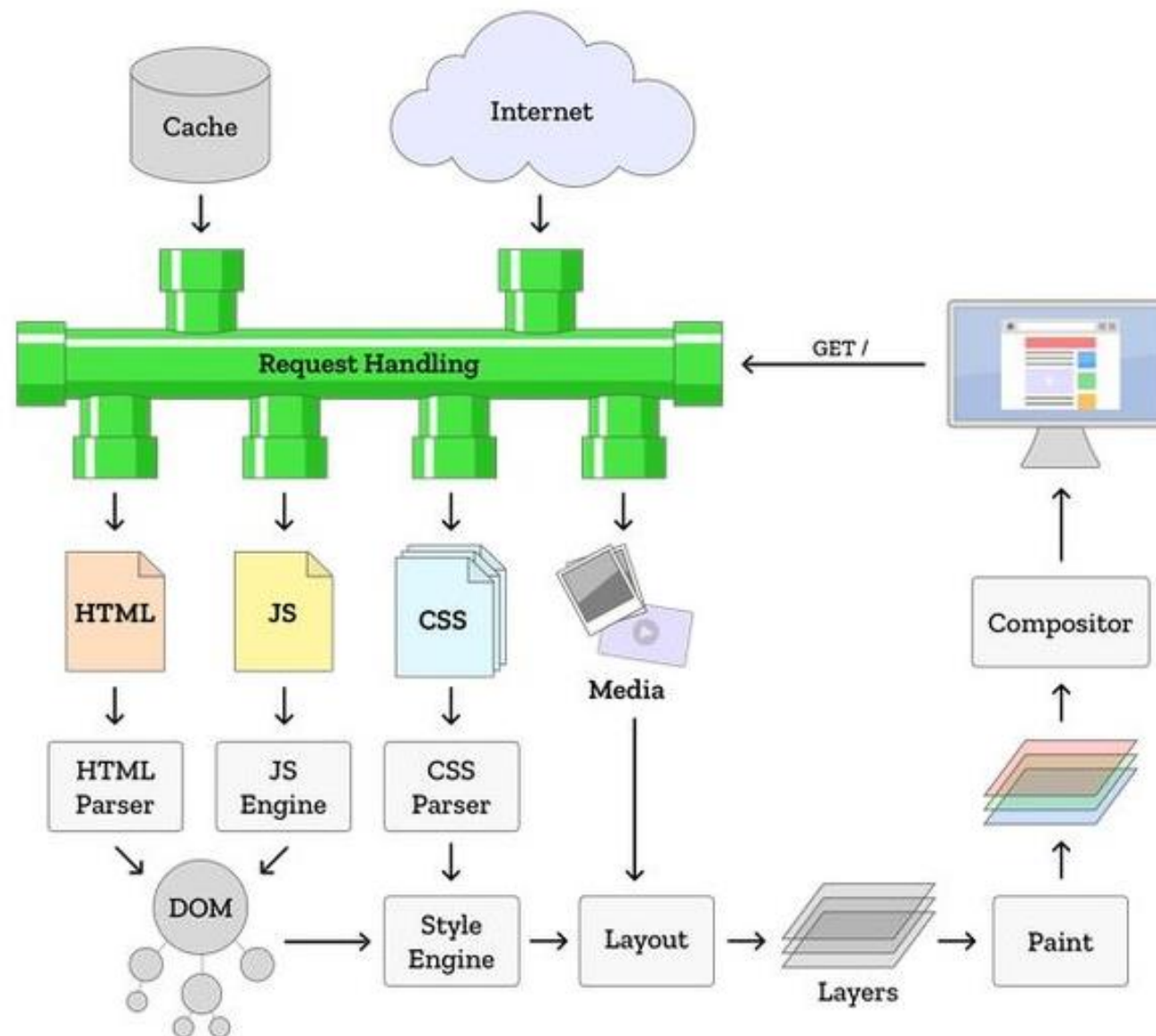
世界之窗浏览器



遨游浏览器

Web浏览器 (browser)

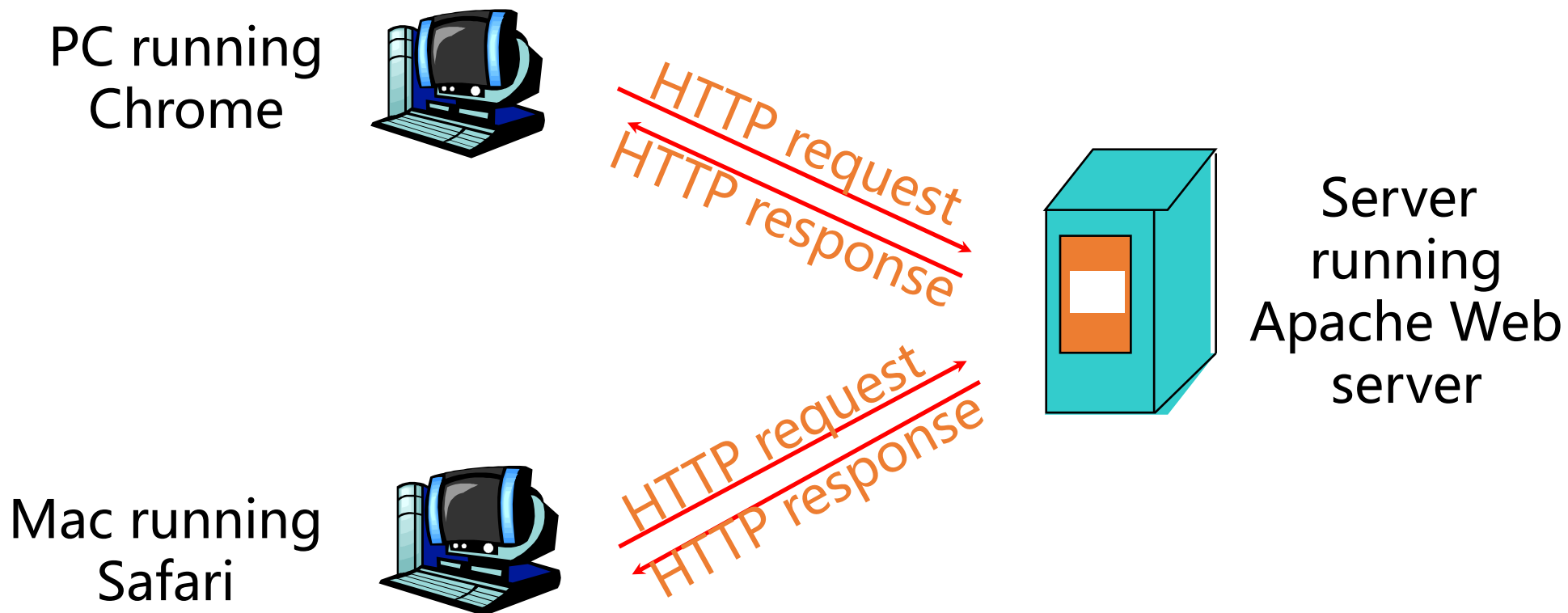
浏览器工作原理



HTTP概述

- HTTP: 超文本传输协议 (HyperText Transfer Protocol)
 - Web的应用层协议
 - client/server模式
 - **client**: 浏览器browser请求, 接收, “解释显示” Web对象
 - **server**: Web服务器响应请求, 发送 Web对象
 - HTTP 1.0: RFC 1945
 - HTTP 1.1: RFC 2616

HTTP概述



HTTP概述 (续)

使用TCP:

- 客户初始化一个与HTTP服务器80端口的TCP连接 (创建套接字)
- HTTP服务器接受来自客户的TCP连接请求, 建立连接
- Browser (HTTP client)和Web服务器 (HTTP server) 交换HTTP消息(应用层协议消息)包括HTTP请求和响应消息
- 最后结束(或叫关闭)TCP连接

HTTP是无状态协议

- HTTP服务器不维护客户先前的状态信息

补充

维护状态的协议非常复杂!

- ❑ 必须维护过去历史 (状态信息)
- ❑ 如果server/client崩溃, 它们各自的状态视图可能不一致, 因此必须保持协调一致。

HTTP连接

非持久HTTP连接

- 每个TCP连接上只传送一个对象，下载多个对象需要建立多个TCP连接
- HTTP/1.0使用非持久HTTP连接

持久HTTP连接

- 一个TCP连接上可以传送多个对象
- HTTP/1.1默认使用持久HTTP连接

非持久HTTP连接(HTTP1.0)

网页由1个HTML文件,
和10个jpeg图像构成

假设用户输入URL `http://www.someSchool.edu/someDepartment/home.index`

1a. HTTP客户初始化1个与服务
器主机

`www.someSchool.edu`中
HTTP服务器的TCP连接

1b. `www.someSchool.edu`服务器
主机中的HTTP服务器在80端口监
听来自HTTP客户的TCP连接请求.
收到连接请求, 接受, 建立连接, 通
知客户.

time



非持久HTTP连接(HTTP1.0)

2. HTTP客户发送1个HTTP请求消息 (*request message*) 包含URL到TCP连接套接字. 消息指出客户要Web对象 -
someDepartment/home.index

3. HTTP服务器接收请求消息, 产生1个响应消息 *response message* 包含被请求对象, 并发送这个消息到自身TCP连接套接字

time



非持久HTTP连接(HTTP1.0)

4. HTTP服务器结束TCP连接

5. HTTP 客户接收包含html文件的响应消息, 显示html. 解析html文件, 找出10个引用jpeg对象

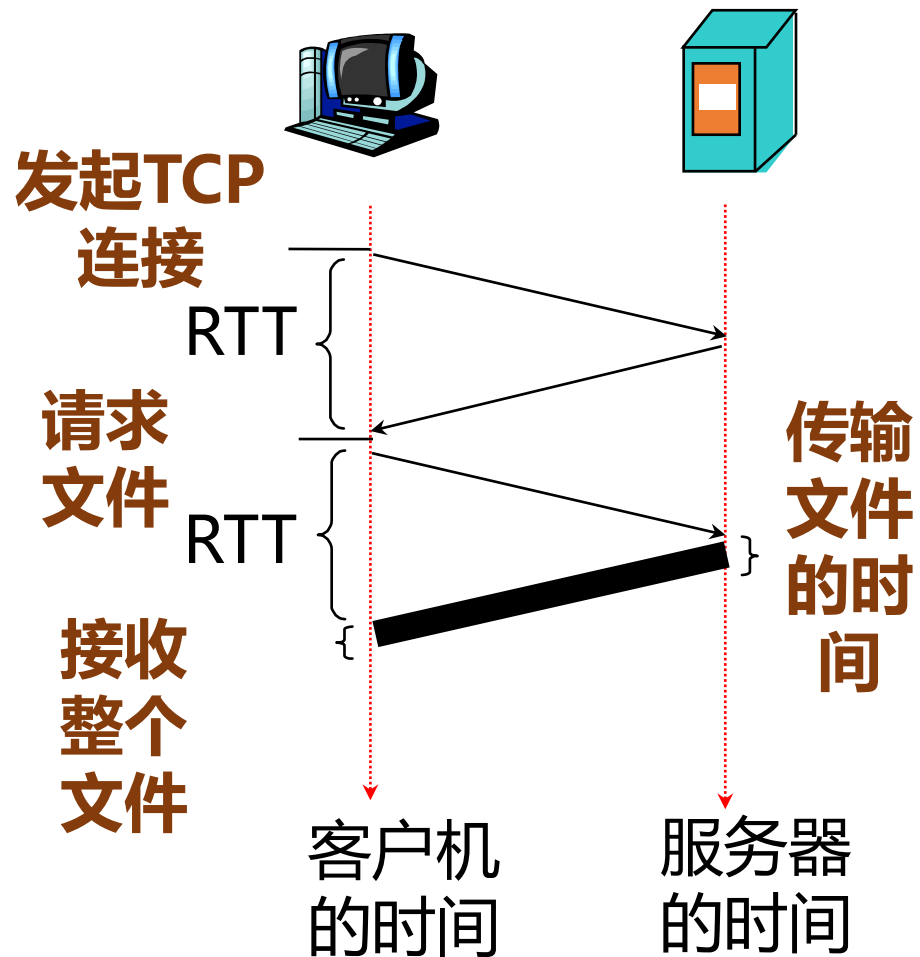
6. 对10个引用jpeg对象的每一个重复步骤1-5

time

响应时间模型

- 定义往返时间RTT(Round-Trip Time):
1个小分组从客户主机到服务器再到客户主机所花费的时间
- 响应时间:
 - 1个RTT用于建立TCP连接
 - 1个RTT用于HTTP请求/响应消息的交互
 - html文件传输时间
- $\text{total} = 2\text{RTT} + \text{transmit time}$

响应时间模型



持久HTTP连接(HTTP1.1)

非持久HTTP连接的问题

每个对象需要2个RTT
OS必须为每个TCP连接分配主机资源
大量客户的并发TCP连接形成服务器的严重负担

持久HTTP连接

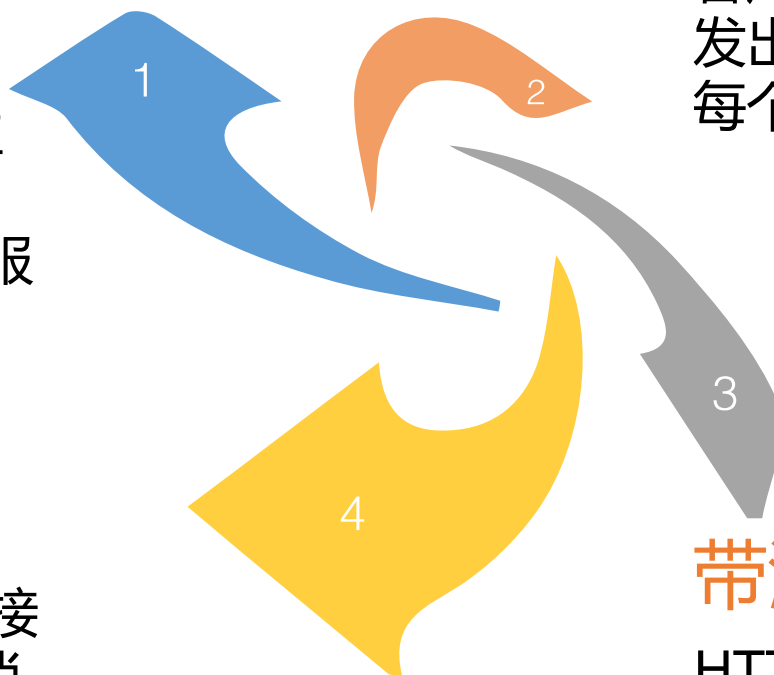
服务器发送响应消息后保持连接
同1客户/服务器的后续HTTP消息继续在该连接上传送

不带流水线的持久HTTP连接

客户先前响应消息收到,才发出新的请求消息
每个引用对象经历1个RTT

带流水线的持久HTTP连接

HTTP/1.1默认使用
客户遇到1个引用对象就发送请求消息
所有引用对象只经历1个RTT



HTTP/2

- 关键目标：减少多对象HTTP请求的延迟
 - HTTP1.1：在单个TCP连接上引入了多个流水线GET
 - 服务器按顺序响应GET请求（FCFS: first-come-first-served scheduling）
 - 对于FCFS，小对象可能必须在大对象后面等待传输（head-of-line (HOL) blocking, 线头阻塞HOL）
 - 丢失恢复（重新传输丢失的TCP段）对对象传输时间的影响

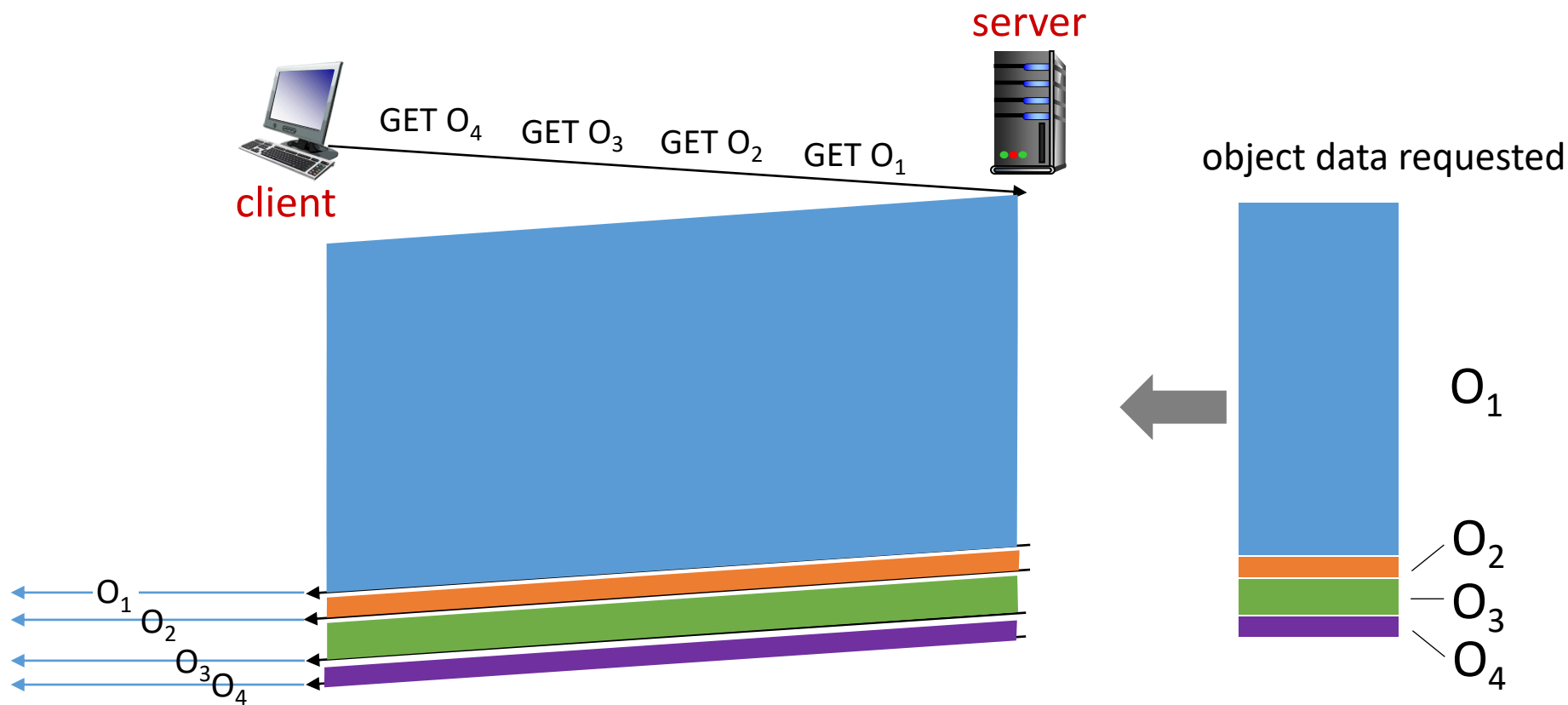
HTTP/2

●关键目标：减少多对象HTTP请求的延迟

- HTTP/2:[RFC 7540, 2015]增加了服务器向客户端发送对象的灵活性
- 方法、状态代码、大多数头字段与HTTP1.1相比没有变化
- 基于客户端指定的对象优先级的请求对象的传输顺序（不一定是FCFS）
- 将未请求的对象推送到客户端
- 将对象划分为框架，安排框架以减少HOL阻塞

HTTP/2: 减缓HOL阻塞

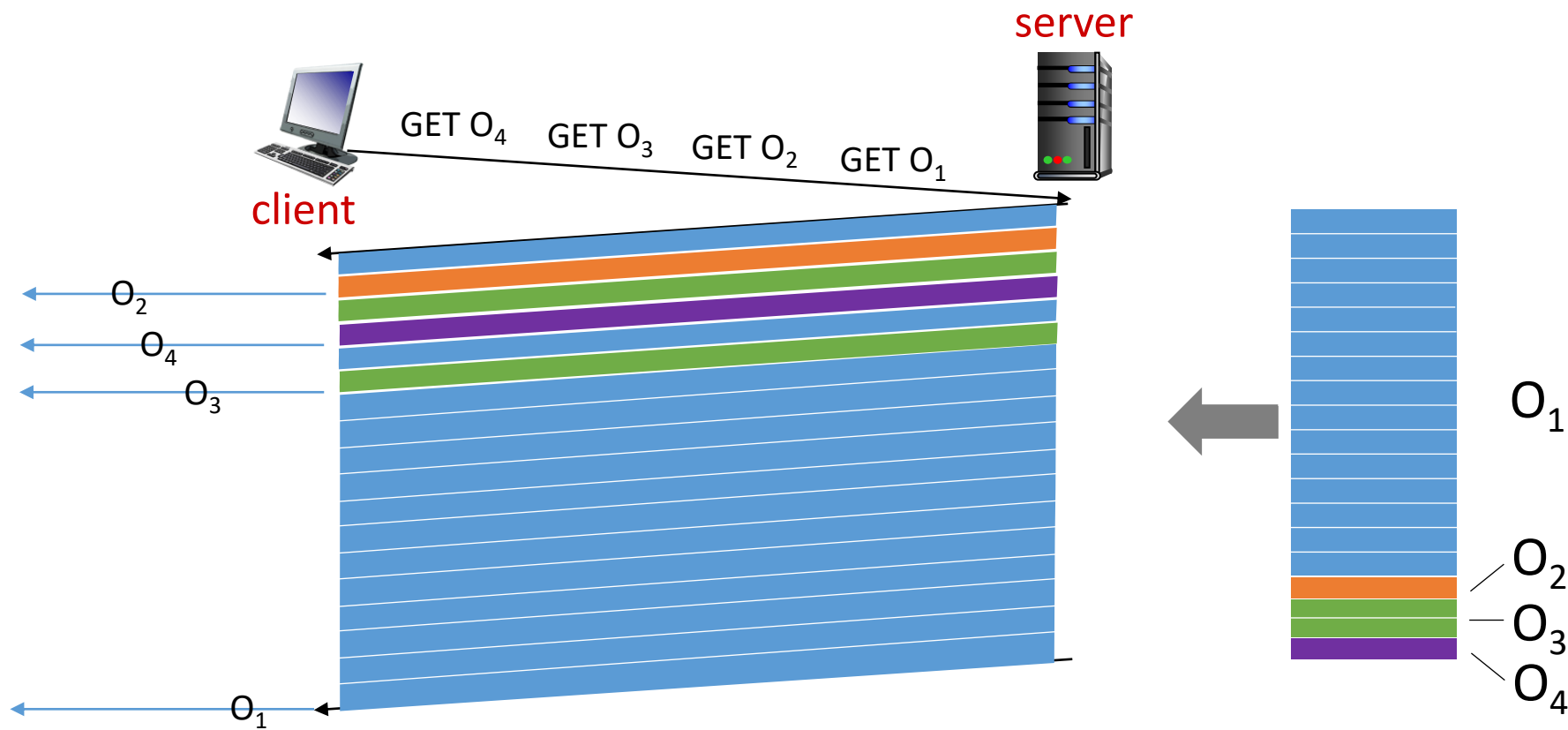
HTTP 1.1:客户端请求1个大对象（例如视频文件）和3个小对象



按要求的顺序交付: O_2, O_3, O_4 O_1 在后面等待

HTTP/2: 减缓HOL阻塞

HTTP/2: 对象被划分为帧(frame), 帧交错传输



O₂, O₃, O₄ 被快速交付, 而O₁ 只被稍微延迟

从 HTTP2 到 HTTP/3

HTTP/2通过单个TCP连接意味着：

- 从数据包丢失中恢复仍然暂停所有对象传输
 - 与HTTP1.1一样，浏览器有动机打开多个并行TCP连接，以减少停滞，提高整体吞吐量
- 普通TCP连接没有安全性
- HTTP/3：通过UDP增加了安全性、每个对象的错误和拥塞控制（更多的流水线操作）
 - 关于传输层支持HTTP/3的更多信息: **QUIC协议** (Quick UDP Internet Connections)

HTTP报文格式

- 2类HTTP报文:请求报文 *request*, 响应报文 *response*
- HTTP请求报文:
 - ASCII文本 (易于人读格式)

请求行
(GET, POST,
HEAD)

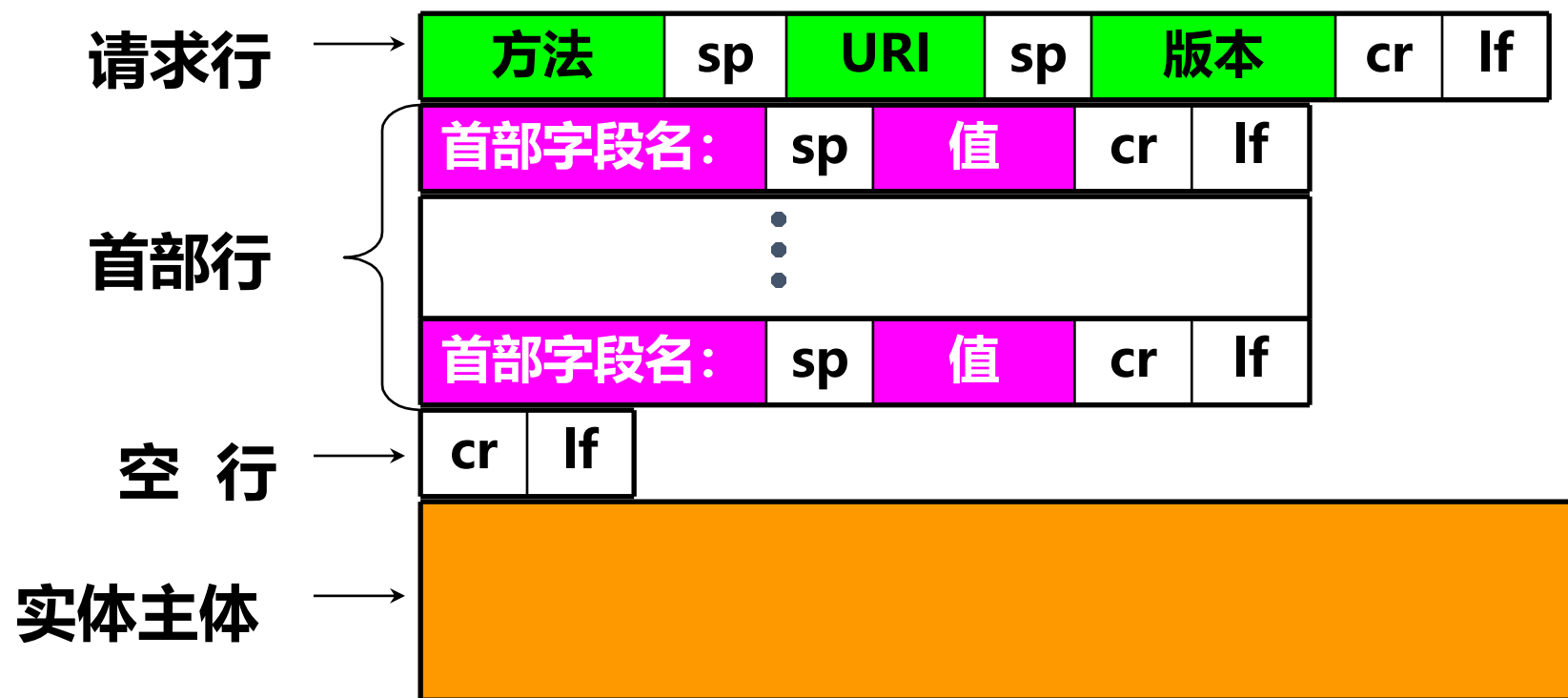
头部行

回车换行
指示结束

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0 //该代理类型的对象版本
Connection: Close //不使用持久连接
Accept-language:zh-cn //中文版本

(额外的 回车换行)
```


HTTP报文格式



注: URI=Uniform Resource Identifiers

方法类型

HTTP/1.0

- GET
- POST
- HEAD
 - 服务器收到请求时，用HTTP报文进行响应，但不返回请求对象

HTTP/1.1

- GET, POST, HEAD
- PUT
 - 文件在实体主体中被上载到URL字段指定的路径
- DELETE
 - 删除URL字段指定的文件

上载表单（各字段）输入值

Post方法:

- 网页时常包含表单输入
- 输入值在请求报文的实体主体中被上载到服务器

URL方法:

- 使用GET方法
- 表单(各字段)输入值被上载,以URL请求行的字段:
`www.somesite.com/animalsearch?monkeys&banana`

HTTP响应消息

状态行
(版本、状态编
码、短语)

首部行

数据, e.g.,
被请求的
HTML文件

HTTP/1.1 200 OK
Connection: close
Date: Sat, 06 Aug 2011 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Thu, 22 Jun 2011
Content-Length: 6821
Content-Type: text/html

data data data data data ...

HTTP 响应的状态码

- 位于服务器响应客户的响应消息的第一行
- 几个常见的样本状态码:

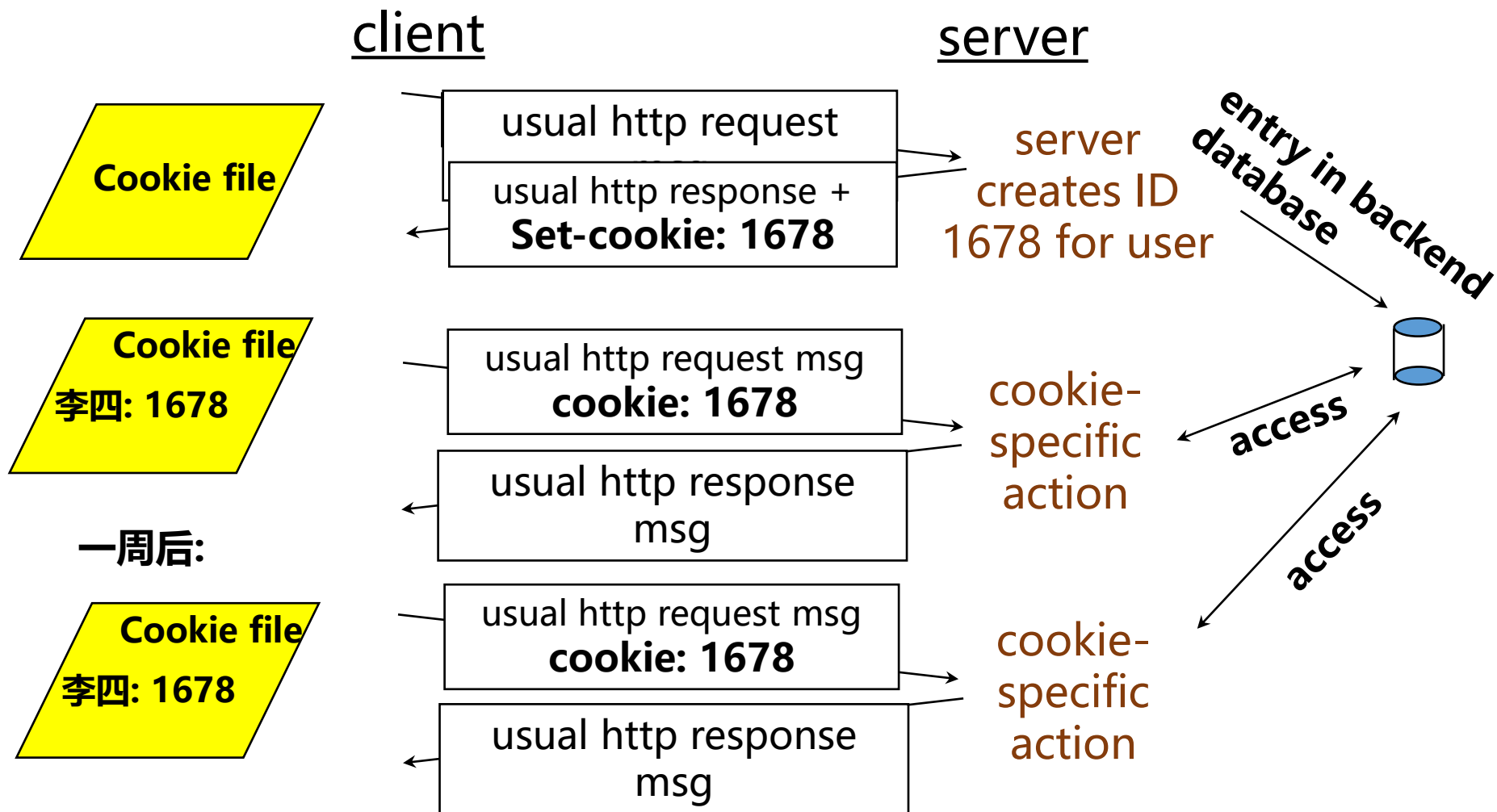


Cookies: 跟踪用户

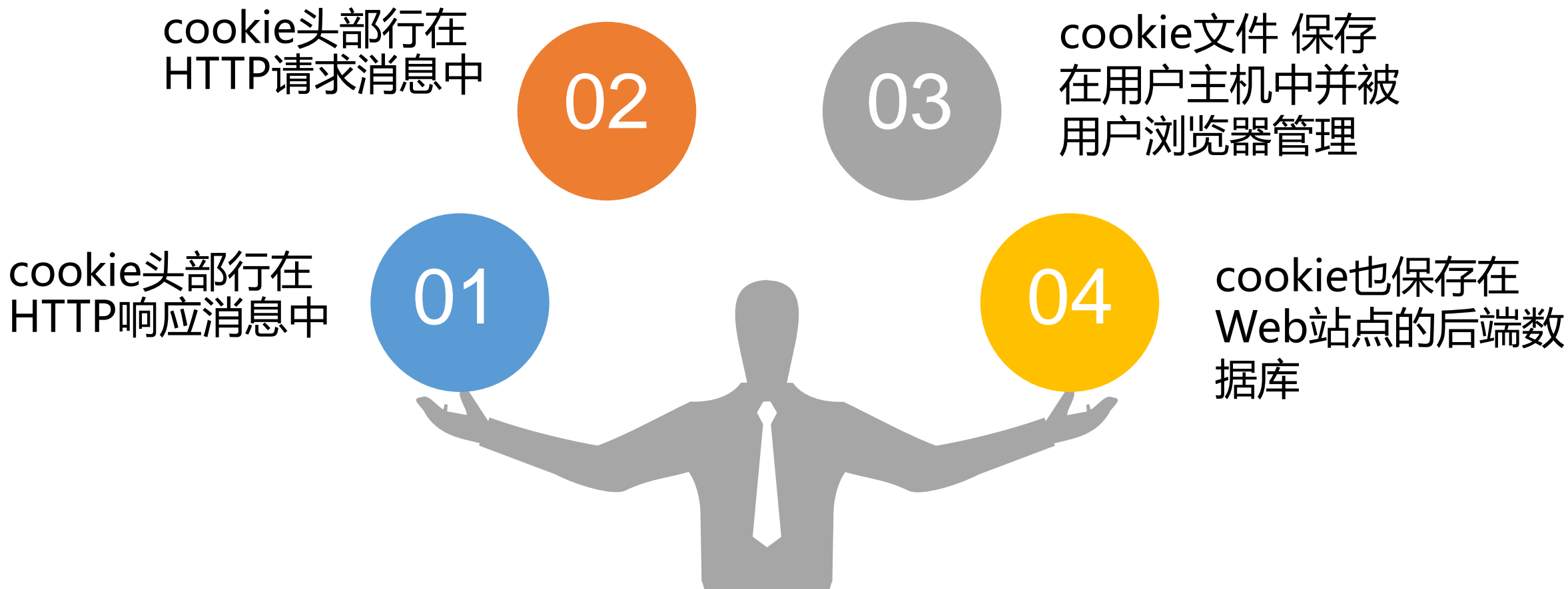
举例:

- 李四总是使用同台PC访问Internet
- 他首次访问1个电子商务网站
- 当他最初发出HTTP请求访问该站点时,该站点创建一个唯一的 ID,并在后端数据库创建一个对应于该ID的表项

Cookies: 跟踪用户(续.)



Cookies的 4 个重要方面



Cookies: 跟踪用户(续.)

Cookies可以带来什么？



Cookies: 跟踪用户(续.)

补充

Cookies 和隐私:

- ❑ cookies允许网站更加了解你
- ❑ 你可以提供名字和e-mail给网站
- ❑ 广告公司通过网站获得信息
- ❑ Cookies不适合游动用户

Cookies: 跟踪用户(续.)

- Cookie可用于：
 - 跟踪用户在给定网站上的行为（第一方cookie）
 - 在多个网站上跟踪用户行为（第三方cookie），而无需用户选择访问跟踪器网站！
 - 跟踪可能对用户不可见：可能是一个不可见的链接
- 通过Cookie进行的第三方跟踪：在Firefox、Safari浏览器中默认禁用，将于2023年在Chrome浏览器中禁用

GDPR (EU General Data Protection Regulation, 欧盟通用数据保护条例) and cookies

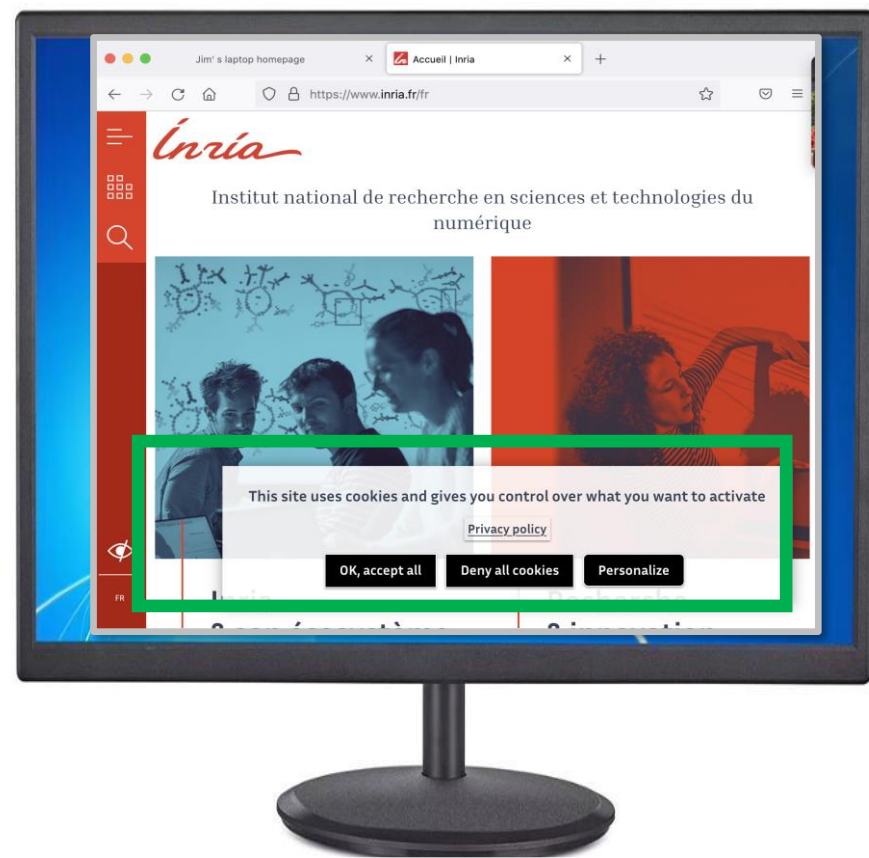
“Natural persons may be associated with online identifiers [...] such as internet protocol addresses, cookie identifiers or other identifiers [...].

This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them.”

GDPR, recital 30 (May 2018)



当cookie可以识别个人时，cookie被视为个人数据，受GDPR个人数据法规的约束



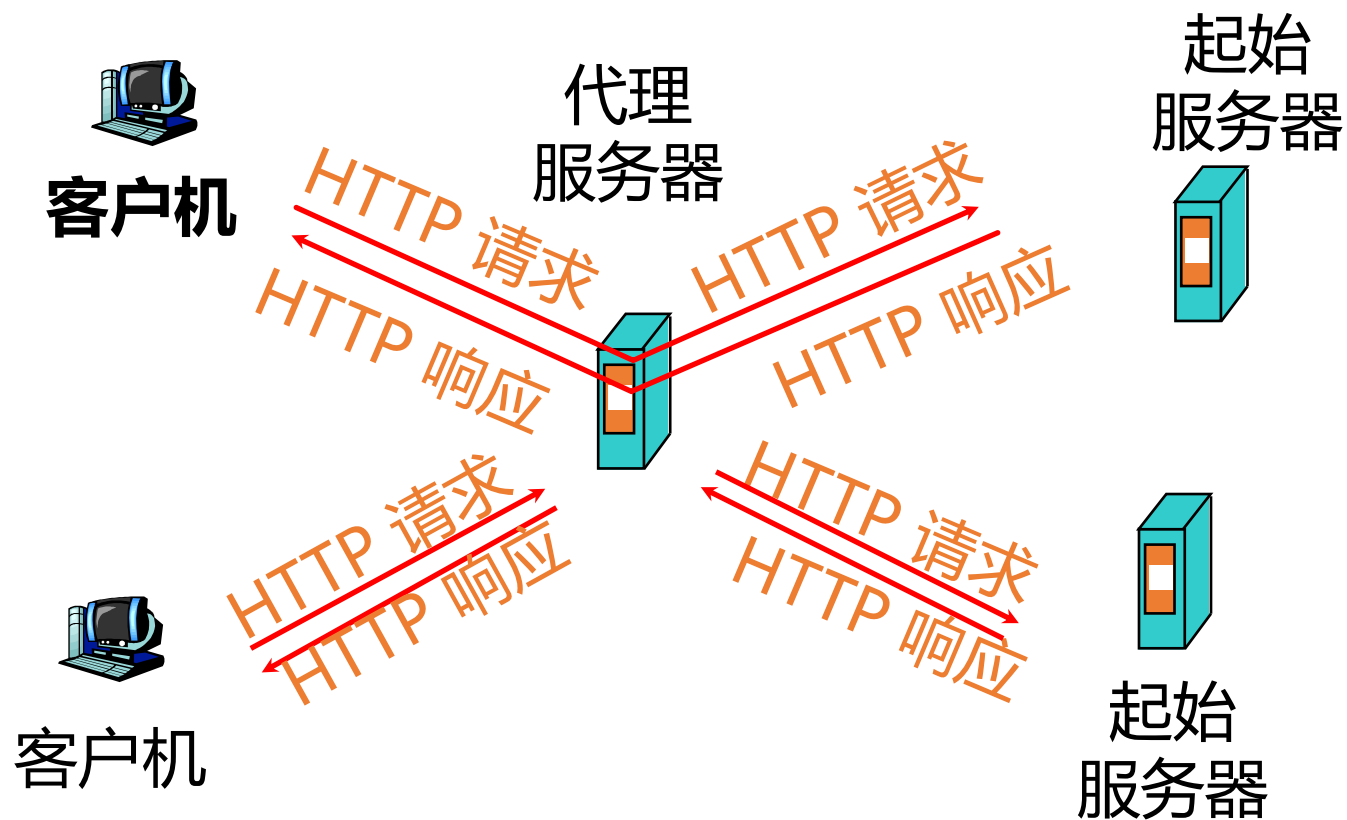
用户可以明确控制是否允许cookie

Web 缓存 (代理服务器)

- **目标:** 代表起始服务器满足HTTP请求
- 用户配置浏览器: Web 访问经由缓存
- 所有HTTP请求指向缓存
 - 对象在缓存中: 缓存器返回对象
 - 否则缓存器向起始服务器发出请求, 接收对象后转发给客户机

Web 缓存 (代理服务器)

目标: 代表起始服务器满足HTTP请求。



Web缓存（续）

- ❑ 缓存器既是服务器又是客户机
- ❑ 一般的，Web缓存器既是服务器又是客户机
- ❑ 典型的缓存器被ISP提供（大学、公司或居民ISP）

为什么需要Web缓存器？

- 减少对客户机请求的响应时间
- 减少内部网络与接入链路上的通信量
- 能从整体上大大降低因特网上的Web流量

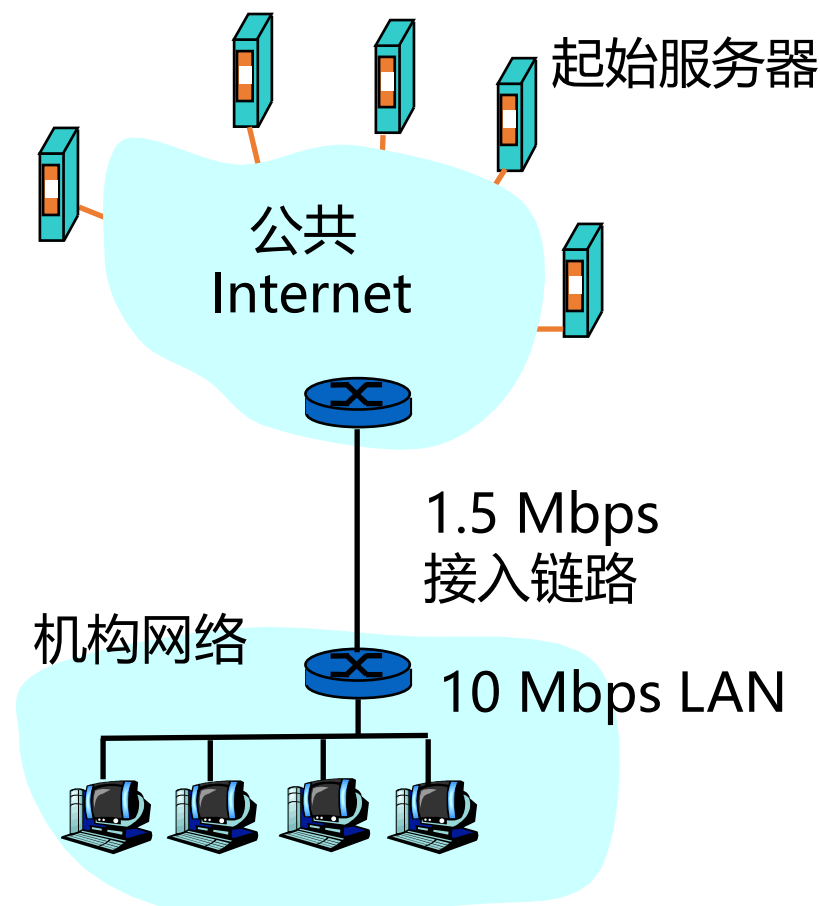
缓存器举例

假设

- ❑ 对象平均长度 = 100,000 bits
- ❑ 浏览器对对象的平均访问速率 = 15/sec
- ❑ 因特网时延 = 2 sec

结论

- ❑ 局域网上的流量强度 = 0.15
- ❑ 链路上的流量强度 = 1
- ❑ 总延时 = 因特网时延 + 接入时延 + 局域网时延
= 2 秒 + 数分钟 + 数毫秒



缓存器举例（续）

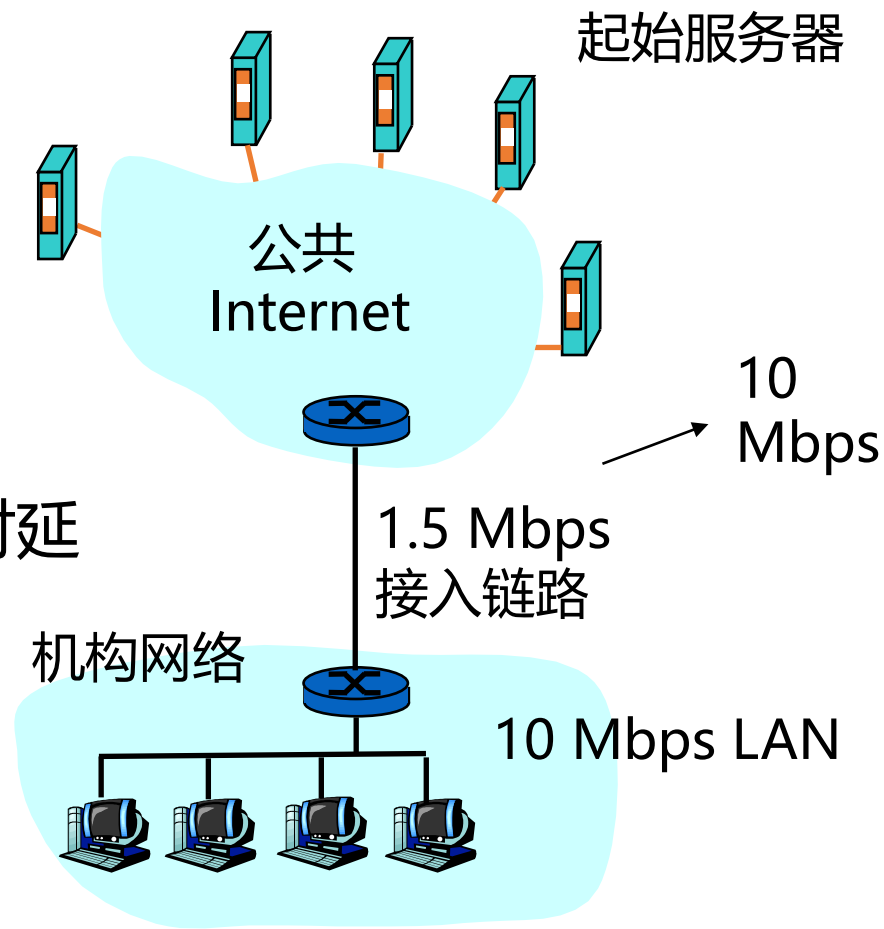
可能的解决办法——

◆ 方法1：增加接入链路的速率

□ 分析：

- 局域网上的流量强度 = 15%
- 链路上的流量强度 = 15%
- 总时延 = 因特网时延 + 接入时延 + 局域网时延
= 2 秒 + 数毫秒 + 数毫秒

□ 这种方案需要较大的投资



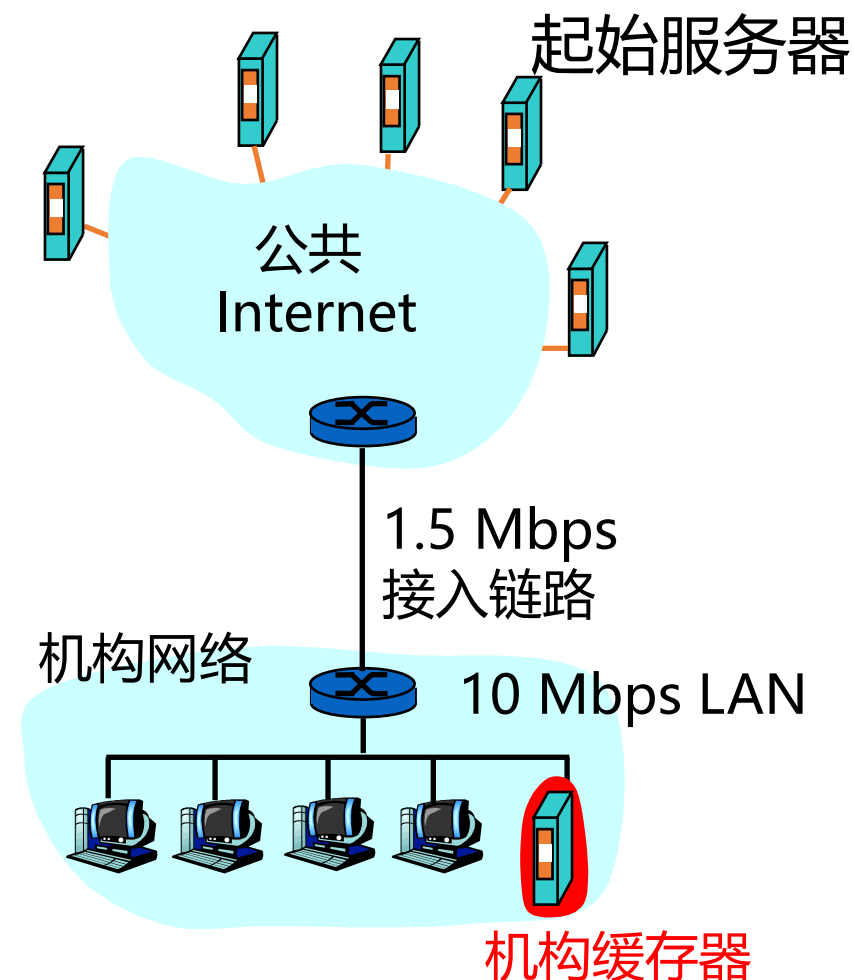
缓存器举例 (续)

◆ 方法2：安装缓存器

□ 分析：

- 假设缓存器命中率为0.4
- 40%的请求立即会得到响应
- 60 %的请求通过访问起始服务器满足
- 链路上的流量强度减为0.6，可以忽略不计
- 总的平均延时 = 因特网延时 + 接入延时 +
局域网延时 = $0.6 * 2 \text{ 秒} + \text{数毫秒} < 1.4 \text{ 秒}$

□ 这种方案投资较少



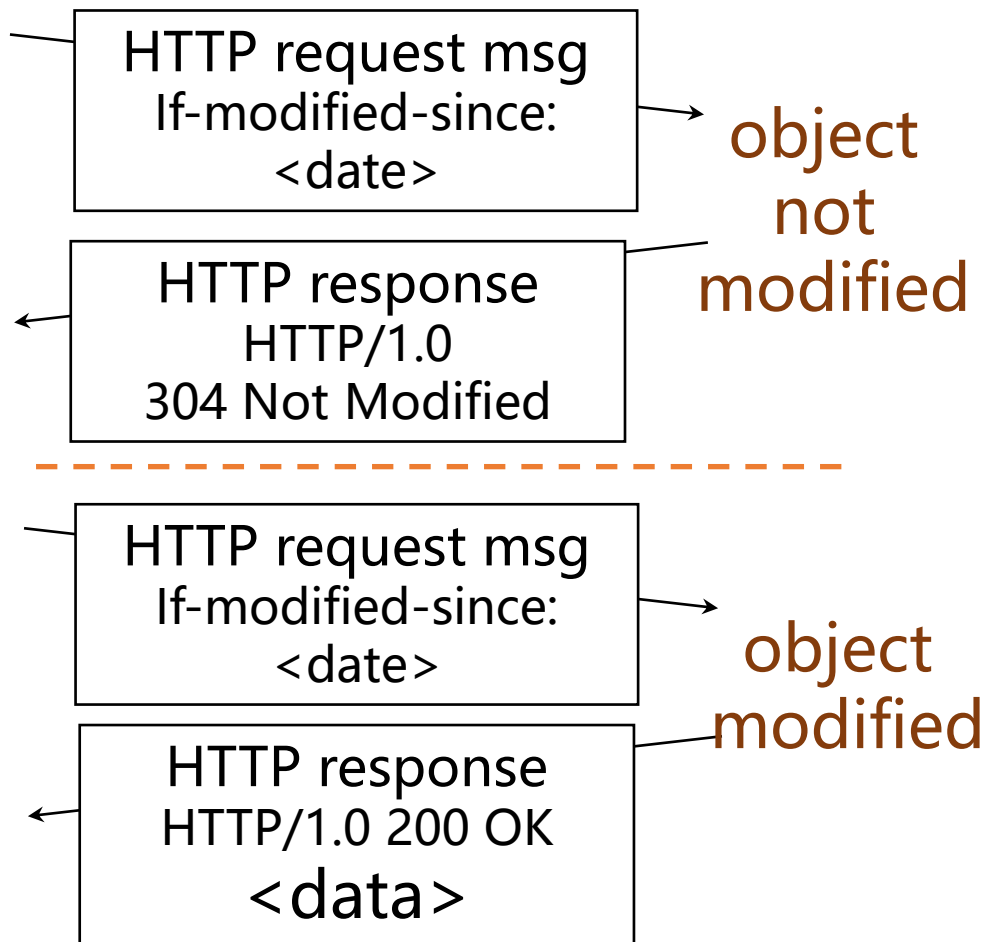
条件GET方法

- ❑ 目的: 证实缓存器中的对象是否为最新
- ❑ 缓存器: 在请求报文中包含对象最后修改时间
If-modified-since: <date>
- ❑ 服务器: 如果对象是最新的则响应报文中不包含对象:
HTTP/1.0 304 Not Modified

条件GET方法

cache

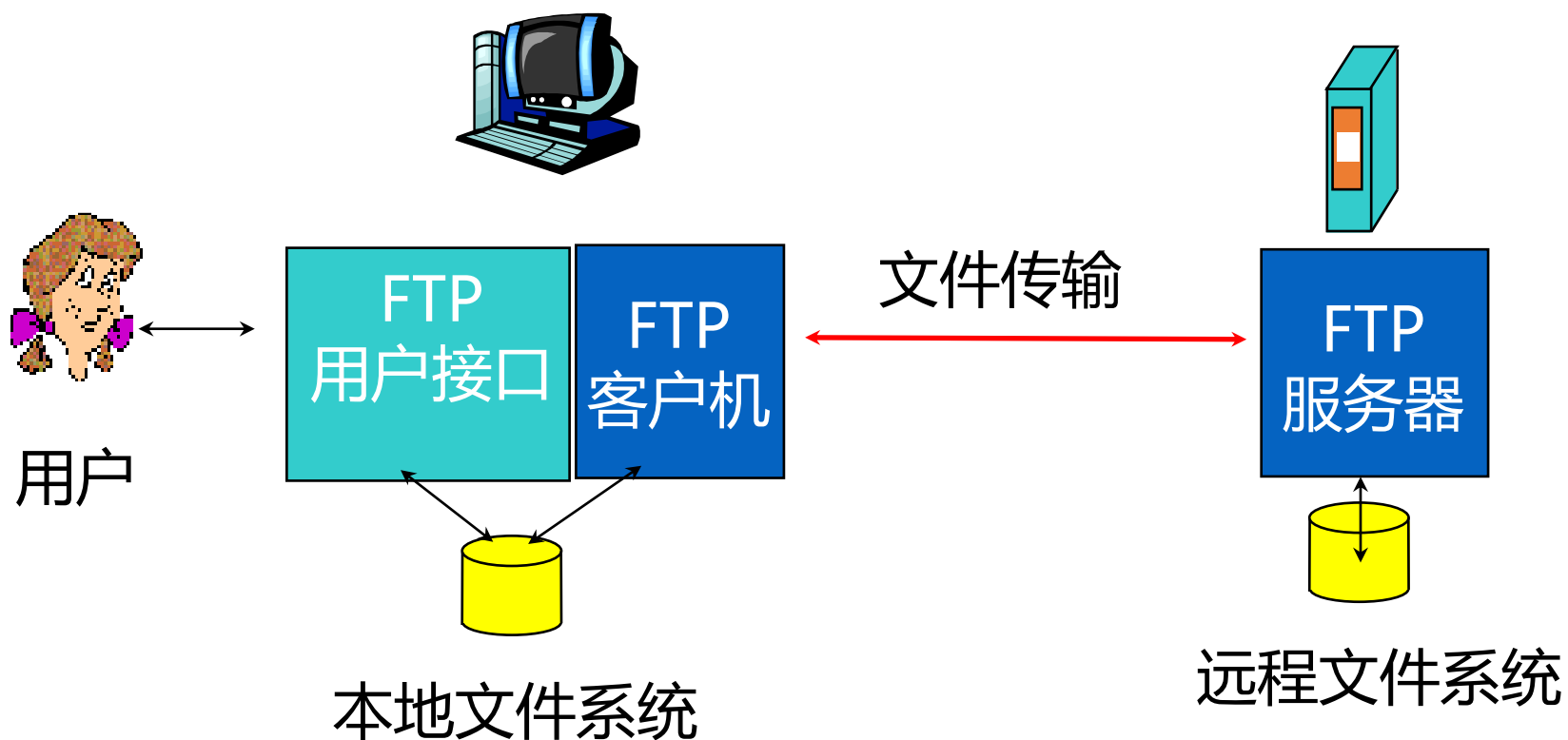
server



03 文件传输协议:FTP*



FTP: 文件传输协议 File Transfer Protocol



FTP: 文件传输协议 File Transfer Protocol

- 传输文件到远程主机/从远程主机下载文件
- client/server模式
 - *client*: 发起传输的一方
 - *server*: 远程主机
- ftp: RFC 959
- ftp服务器: 端口号 21

FTP: 文件传输协议 File Transfer Protocol

- FTP客户首先发起建立1个与FTP服务器端口号21之间的TCP控制连接, 指定TCP作为传输层协议
- 客户在建立的控制连接上获得身份认证
- 客户在建立的控制连接上发送命令来浏览远程主机的目录.
- 当服务器接收到1个文件传输命令时, 在服务器端口号20创建1个与客户的TCP数据连接
- 1个文件传输后,服务器结束这个TCP数据连接.

FTP: 独立的控制连接, 数据连接



- ❑ 服务器创建第2个TCP与客户的数据连接来传输下一个文件.
- ❑ 控制连接: 带外发送控制信息
- ❑ FTP 服务器要维护用户状态信息: 当前目录, 先前的身份认证

FTP: 文件传输协议

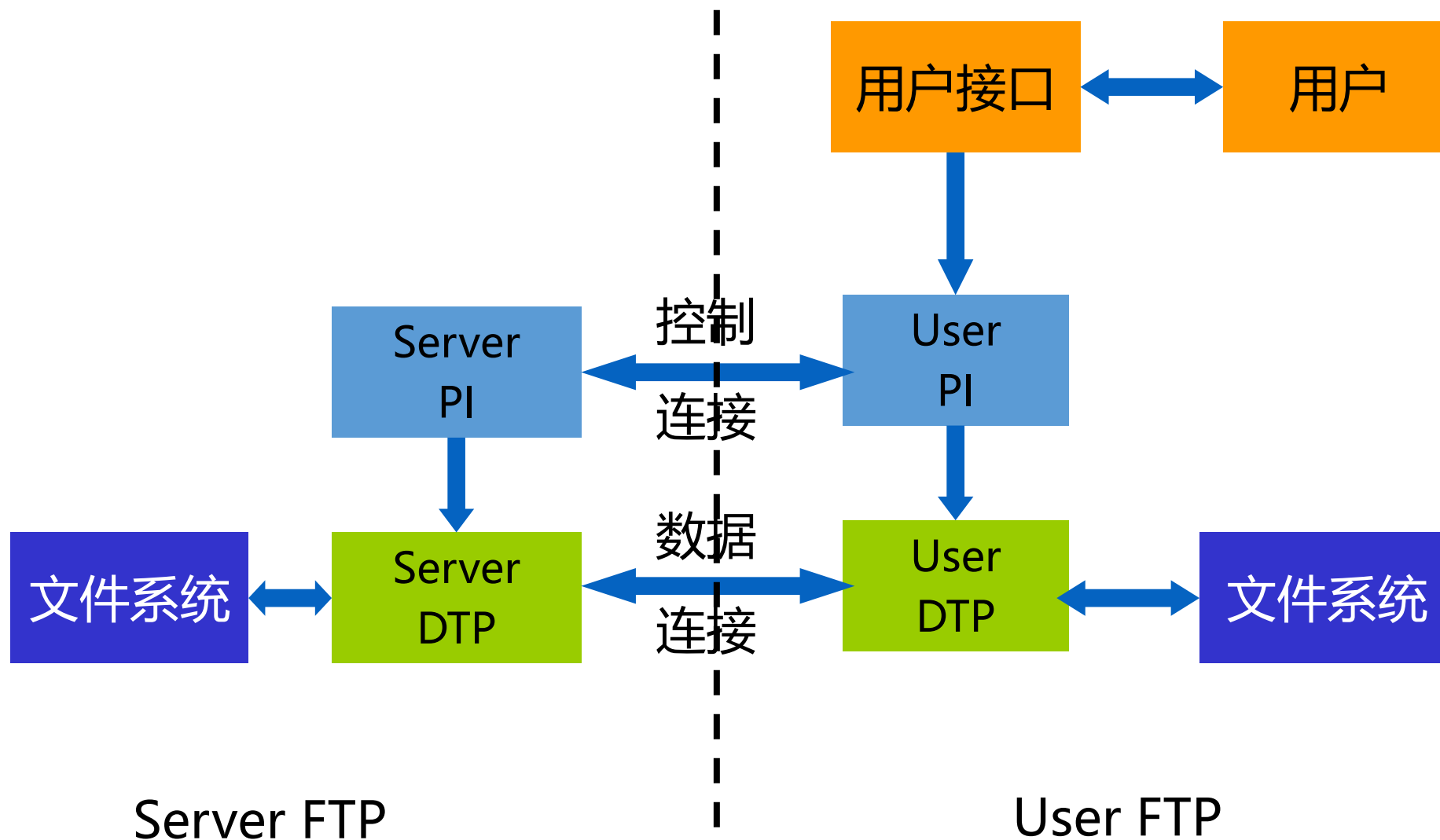
控制连接:

- USER-PI(protocol interpreter): 用户协议解释器
- SERVER-PI: 服务器协议解释器

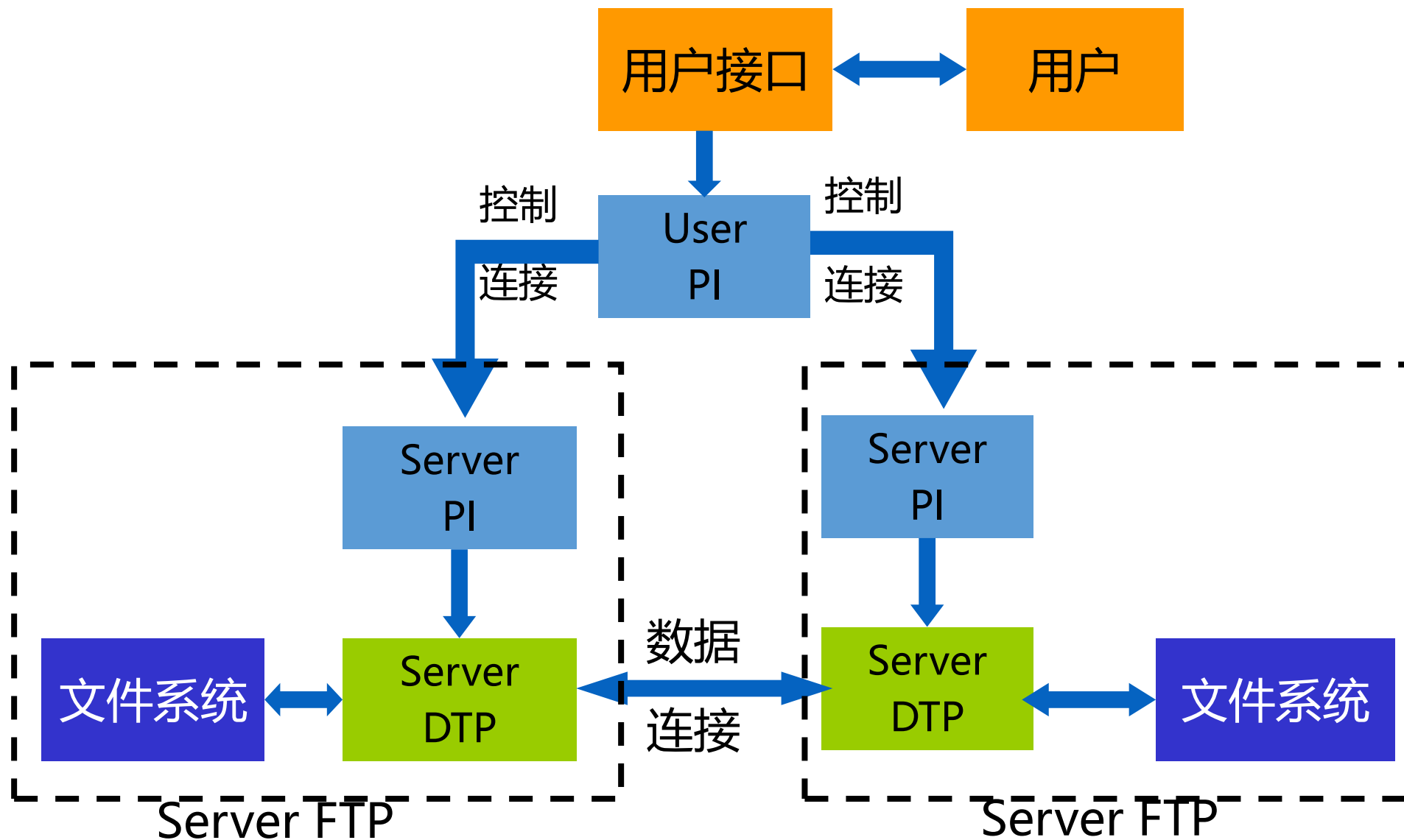
数据连接:

- user-DTP(Data Transfer Process): 用户数据传输进程
- server-DTP: 服务器数据传输进程

FTP: 文件传输系统模型一



FTP: 文件传输系统模型二



FTP数据连接建立方式

主动模式:

- 客户端发送PORT命令

PORT h1,h2,h3,h4,p1,p2

(h1-h4是IP地址, p1-p2是端口号)

- 服务器根据PORT命令指定的客户端地址和端口号发起数据连接

被动模式:

- 客户端发送PASV命令
- 服务器返回监听的地址和端口号
- 客户端发起数据连接

FTP命令和应答

常见命令:

- 在控制连接上发送ASCII文本
- USER *username*
- PASS *password*
- LIST: 返回当前远程目录的文件列表
- RETR filename: 获取远程主机当前目录下的1个文件(get)
- STOR filename: 存放1个文件到远程主机当前目录下(put)

常见应答:

- 状态码及其相应短语 (类同 HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file

04 因特网中的电子邮件 SMTP, POP3, IMAP



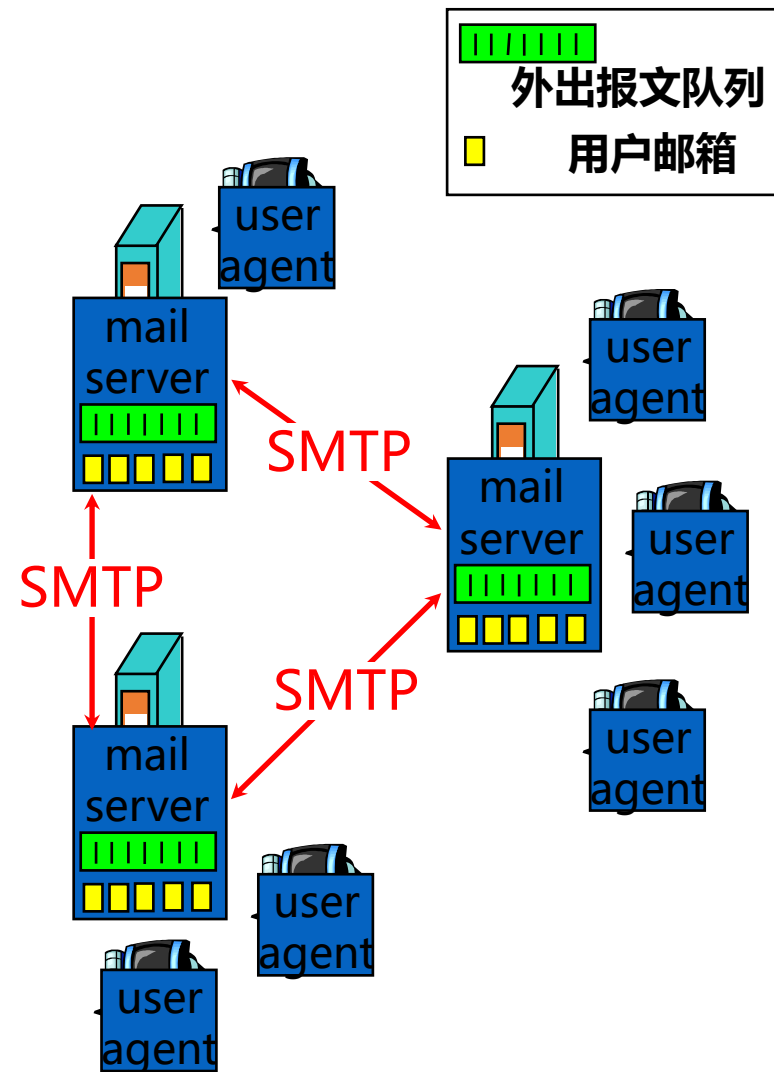
电子邮件

3个主要组成部分:

- 用户代理user agents
- 邮件服务器mail servers
- 简单邮件传送协议和邮件接收协议

用户代理:

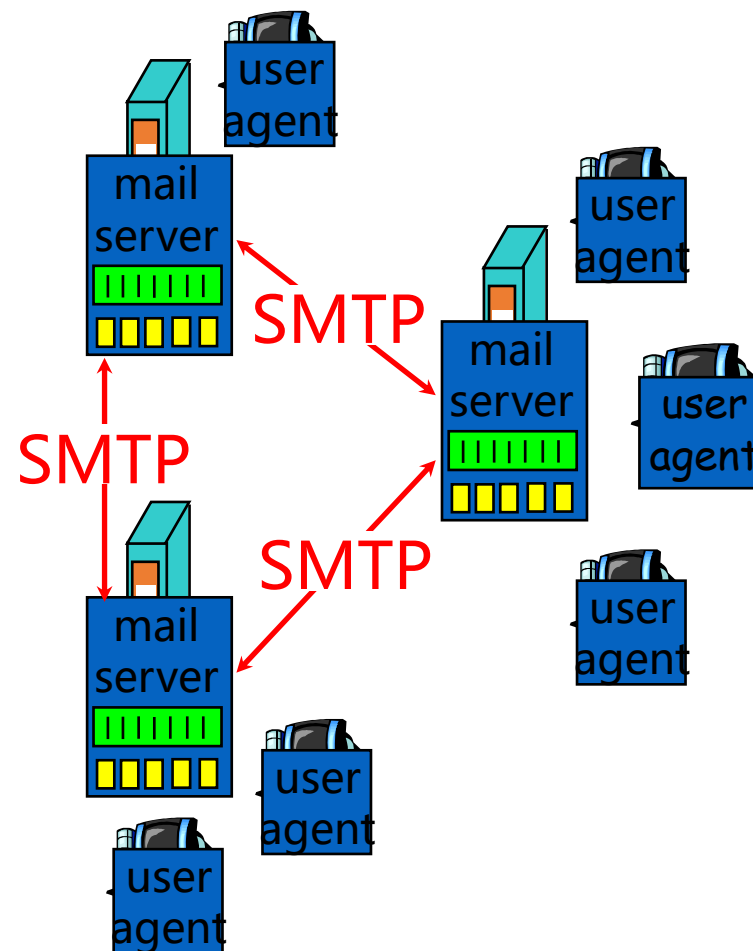
- 允许用户阅读,回复,转发,保存,编辑邮件消息
- 例如: Outlook, foxmail等
- 发送, 接收邮件消息到/从服务器
- 运行邮件协议



电子邮件: 邮件服务器

邮件服务器

- 邮箱mailbox 存放用户接收的邮件消息
- 外出报文队列outgoing message queue
- 运行邮件协议



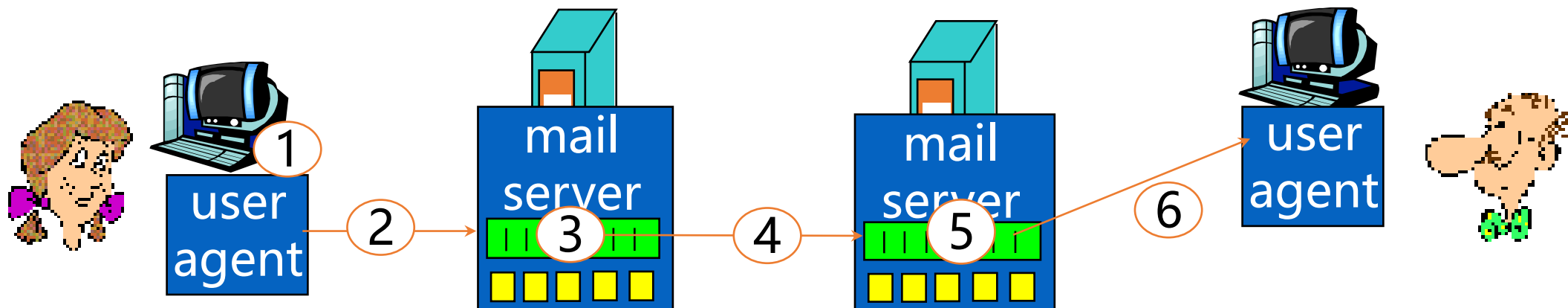
电子邮件: SMTP [RFC 5321]

Simple Mail Transfer Protocol

- 客户使用TCP来可靠传输邮件消息到服务器端口号25
- 直接传送: 发送服务器到接收服务器
- 传输的3个阶段
 - 握手 (问候)
 - 邮件消息的传输
 - 结束
- 命令/应答的交互
 - 命令: ASCII文本格式
 - 应答: 状态码及其短语
- 邮件消息必须是7-bit ASCII

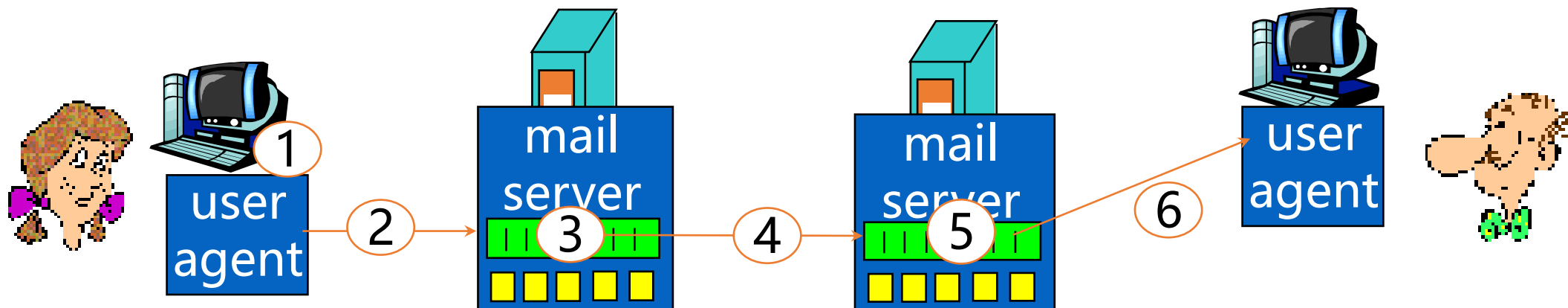
例子: Alice发送邮件消息到Bob

- 1) Alice使用用户代理编写邮件消息(给bob@some school.edu的)
- 2) Alice的用户代理发送邮件消息 到她的邮件服务器;邮件消息存放在邮件消息队列
- 3) Alice邮件服务器的SMTP客户端发起建立一个到Bob的邮件服务器的SMTP服务器端的TCP连接,经过应用层握手.



例子: Alice发送邮件消息到Bob

- 4) SMTP客户在这个TCP连接上发送Alice的邮件消息
- 5) Bob服务器存放邮件消息存到 Bob的邮箱
- 6) Bob调用他的用户代理读邮件消息



SMTP客户和服务器的命令交互

S: 220 hamburger.edu

C: HELO crepes.fr

S: 250 Hello crepes.fr, pleased to meet you

C: MAIL FROM: <alice@crepes.fr>

S: 250 alice@crepes.fr... Sender ok

C: RCPT TO: <bob@hamburger.edu>

S: 250 bob@hamburger.edu ... Recipient ok

SMTP客户和服务器的命令交互

S: 354 Enter mail, end with "." on a line by itself

C: Do you like ketchup?

C: How about pickles?

C: .

S: 250 Message accepted for delivery

C: QUIT

S: 221 hamburger.edu closing connection

SMTP: 总结

与HTTP的比较:

- SMTP使用持久连接
- SMTP 要求邮件消息 (header & body)必须是7-bit ASCII
- SMTP服务器使用 CRLF.CRLF 来判断邮件消息的结束
- HTTP: 拉协议
- SMTP: 推协议
- 都有ASCII 命令/应答交互, 状态码
- HTTP: 每个对象封装在它各自的 HTTP响应消息中发送
- SMTP: 一个邮件内各个对象置于同一个邮件消息的多目部分发送

邮件消息的格式

SMTP: 用来交换邮件消息的协议

RFC 822: 文本邮件消息格式标准:

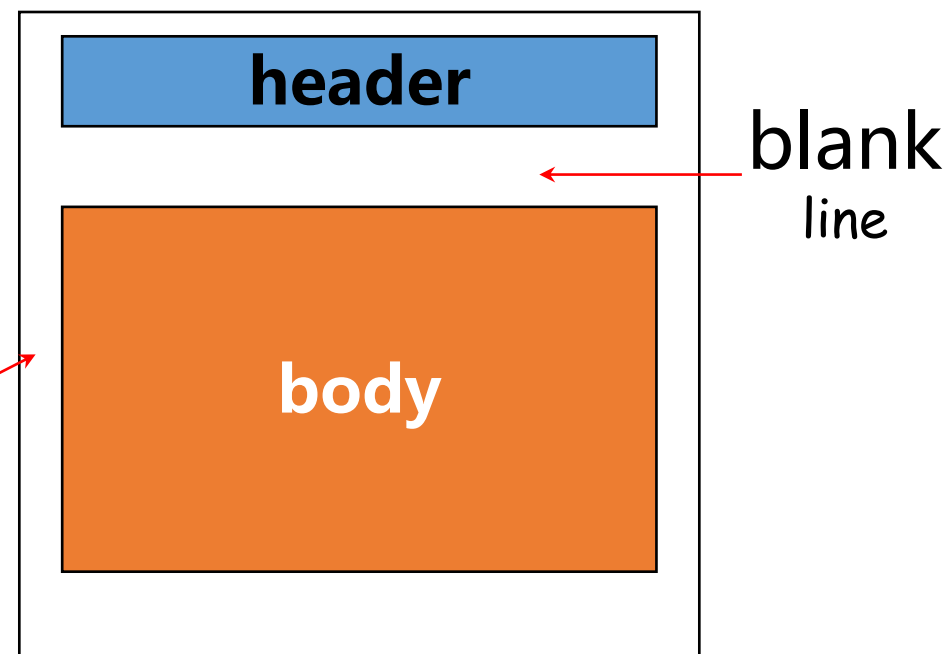
- 信头 - 头部行。如:

- To:
- From:
- Subject:

这些头部不同于SMTP命令!

- 信体

- 邮件消息也必须是ASCII字符



邮件消息的格式: 多媒体扩展

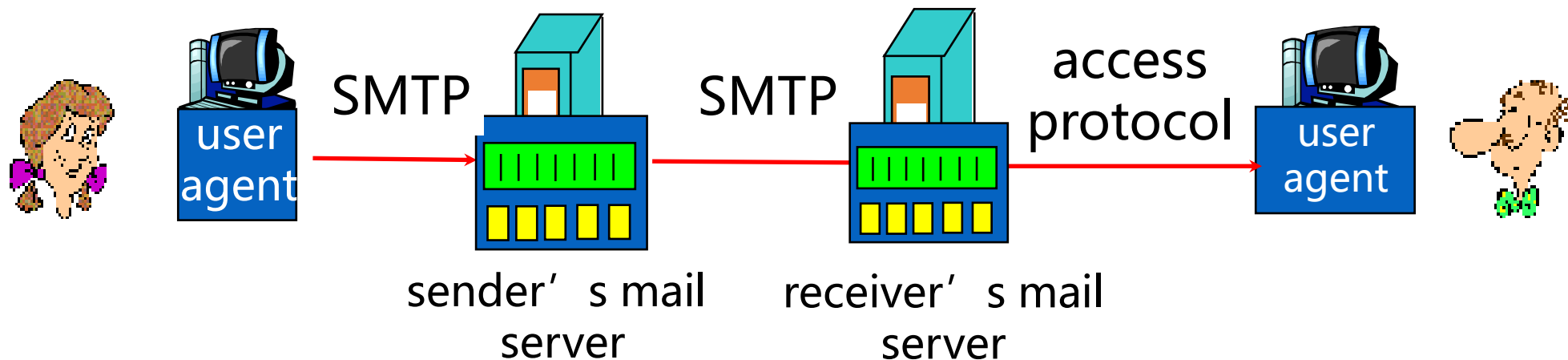
- MIME: Multipurpose Internet mail Extensions
多用途因特网邮件扩展, RFC 2045, 2046
- 增添额外的信头头部声明MIME content-type

MIME版本
用来编码数据
的方法
多媒体数据
类型名, 子类型名,
参数声明
编码后的数据

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

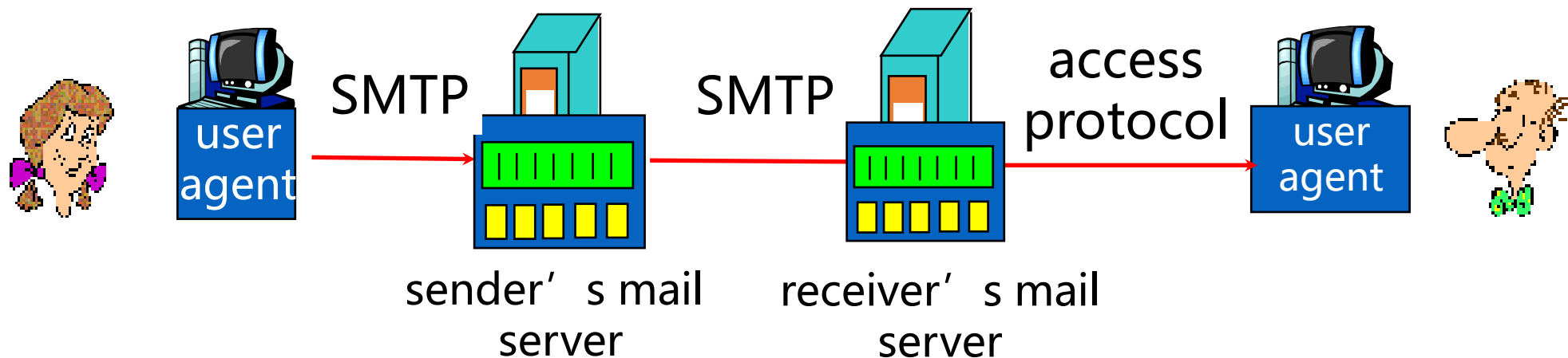
base64 encoded data .....
.....base64 encoded data
```

邮件访问协议



- SMTP: 递送/存储邮件消息到接收者邮件服务器
- 邮件访问协议: 从服务器获取邮件消息
 - POP: Post Office Protocol 邮局协议[RFC 1939]110端口号
 - 身份认证 (代理 <-->服务器) 并 下载邮件消息

邮件访问协议



- IMAP: Internet Message Access Protocol [RFC 3501] 143端口
 - 更多功能特征 (更复杂!)
 - 允许用户像对待本地邮箱那样操纵远程邮箱的邮件
- HTTP: Hotmail , Yahoo! Mail, etc.

SMTP客户和服务器的命令交互

S: 220 hamburger.edu

C: HELO crepes.fr

S: 250 Hello crepes.fr, pleased to meet you

C: MAIL FROM: <alice@crepes.fr>

S: 250 alice@crepes.fr... Sender ok

C: RCPT TO: <bob@hamburger.edu>

S: 250 bob@hamburger.edu ... Recipient ok

SMTP客户和服务器的命令交互

C: DATA

S: 354 Enter mail, end with "." on a line by itself

C: Do you like ketchup?

C: How about pickles?

C: .

S: 250 Message accepted for delivery

C: QUIT

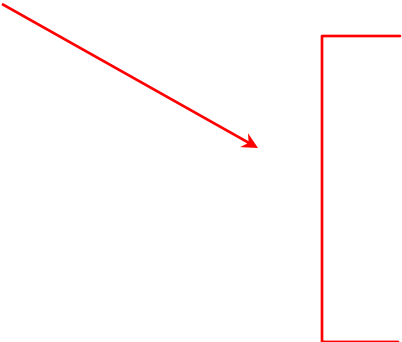
S: 221 hamburger.edu closing connection

POP3协议

身份认证阶段

authorization phase

- 客户命令:
 - user username
 - pass password
- 服务器响应
 - +OK
 - -ERR



S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully
logged on

POP3协议

传输阶段

transaction phase, client: →

- list: 列出邮件编号
- retr: 按编号取邮件
- dele: 删除
- quit

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 和 IMAP

POP3的更多细节

- 先前例子使用 “Download-and-delete”.
- Bob换客户端后不能再读邮件
- “Download-and-keep”模式: 在不同客户机上的邮件拷贝
- POP3的会话是无状态的

IMAP

- 保存所有邮件消息在一个位置: 服务器
- 允许用户在服务器的各文件夹中管理邮件消息
- IMAP跟踪用户会话的状态信息:
 - 文件夹和邮件消息IDs与文件夹名字的映射

05

DNS:英特网的目录服务



DNS: 域名系统Domain Name System

人: 很多标识符:

- ID, name, passport #

Internet主机, 路由器:

- IP address (32 bit) – 用于分组寻址
- “主机名”, e.g.,
gaia.cs.umass.edu – 用于人记忆识别

Q: 可以在IP地址和主机名之间建立映射吗?

DNS服务器提供的功能:

- 主机名到IP地址的转换
- 主机别名
 - 一个主机可以有一个规范主机名和多个主机别名
- 邮件服务器别名
- 负载分配
 - DNS实现冗余服务器: 一个IP地址集合可以对应于同一个规范主机名。

DNS

Domain Name System:

域名系统

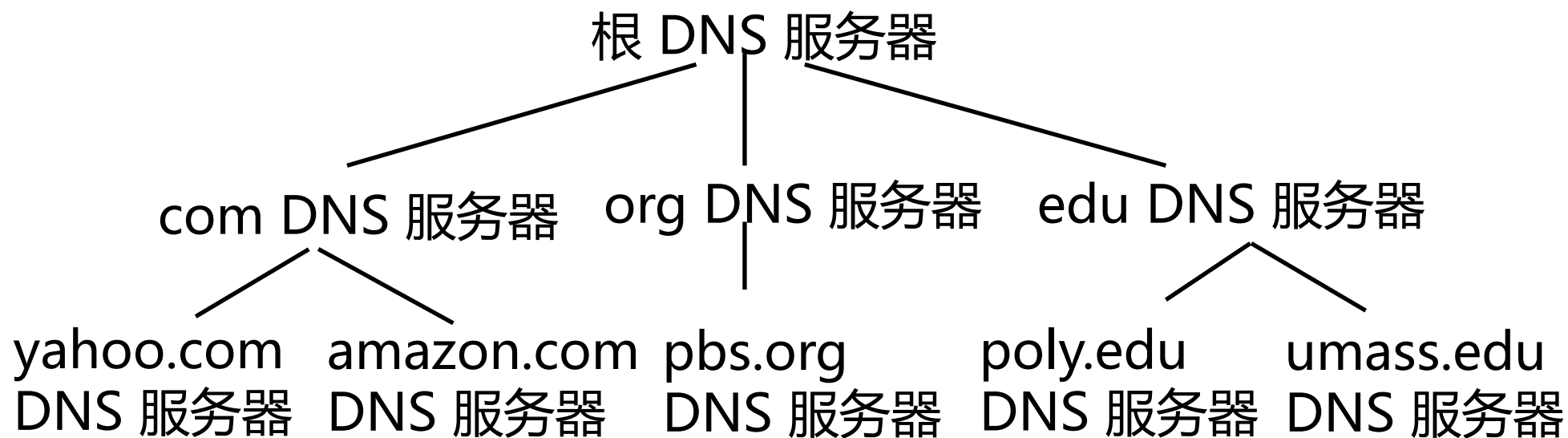
- ❑ **分布式数据库**: 一个由分层DNS服务器实现的分布式数据库
- ❑ **应用层协议**: DNS服务器实现域名转换 (域名/地址转换)

为什么不集中式DNS?

- ❑ 单点故障
- ❑ 巨大访问量
- ❑ 远距离集中式数据库
- ❑ 维护

不可扩展!

分布式、层次数据库



分布式、层次数据库

客户机怎样决定主机名www.amazon.com的IP地址?

- ❑ 客户机查询根服务器得到com DNS服务器
- ❑ 客户机查询com DNS服务器得到amazon.comDNS服务器
- ❑ 客户机查询amazon.comDNS服务器得到www.amazon.com的IP地址

DNS: 根名字服务器Root name servers

<http://www.root-servers.org/>

- 根名字服务器负责记录顶级域名服务器的信息

13 root name servers

a.root-servers.net

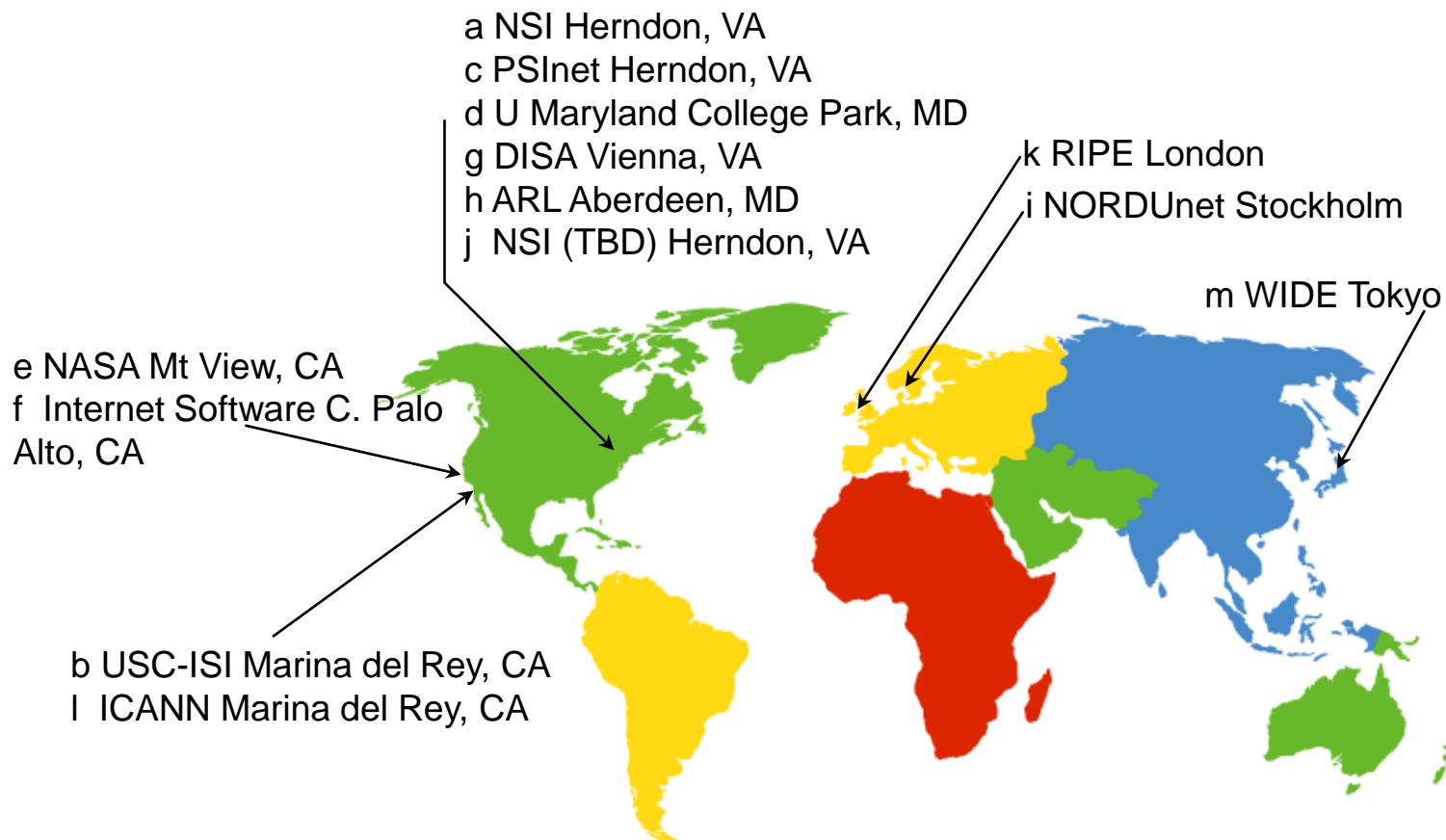
...

m.root-servers.net

10个美国, 1个英国, 1个瑞典, 1个日本
共1092台服务器 (到2020年3月12日为止)

DNS: 根名字服务器Root name servers

<http://www.root-servers.org/>



顶级域服务器和权威DNS服务器

- **顶级域服务器 (top-level domain servers)** : 负责顶级域名 com, org, net, edu, etc, 和所有国家的顶级域名 uk, fr, ca, jp.
 - Network solutions 公司维护com顶级域的TLD服务器
 - Educause 公司维护edu顶级域的 TLD服务器
- **权威DNS服务器(authoritative DNS servers)**: 在因特网上具有公共可访问主机（如Web服务器和邮件服务器）的每个组织机构必须提供公共可访问的DNS记录，这些记录将这些主机的名字映射为IP地址。组织机构的权威DNS服务器负责保存这些DNS记录。
 - 多数大学和公司维护它们的基本权威DNS服务器

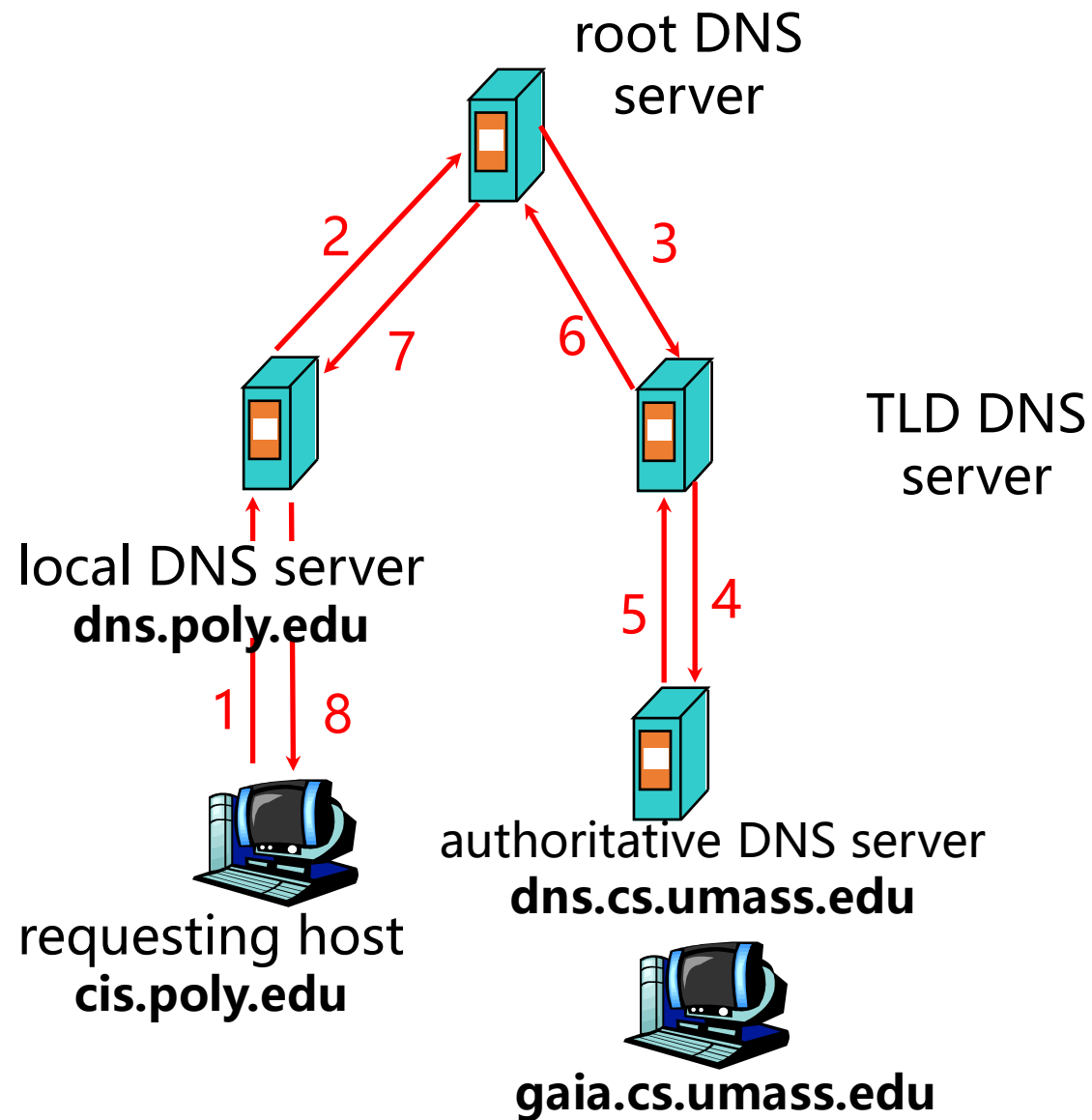
本地DNS服务器(Local DNS name server)

- ❑ 严格来说不属于该服务器的层次结构
- ❑ 每个ISP（如居民区ISP、公司、大学）都有一个本地DNS
 - 也叫默认服务器
- ❑ 当主机发出DNS请求时，该请求被发往本地DNS服务器。
 - 起着代理的作用，转发请求到层次结构中。

DNS查询方法一

递归查询(recursive query):

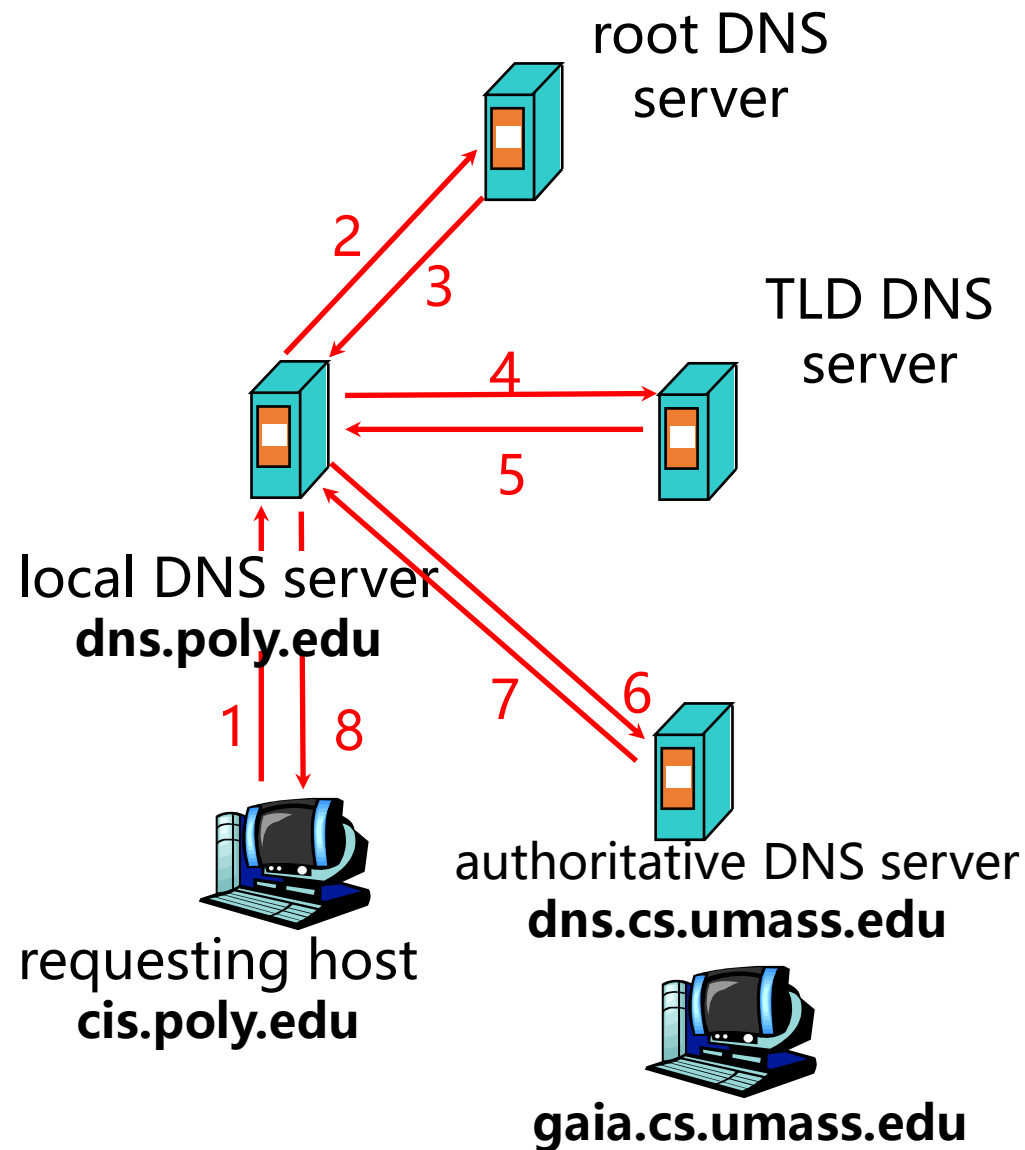
- ❑ 名字解析的负担交给被查询的名字服务器
- ❑ 被查询的名字服务器负载重?



DNS查询方法二

迭代查询(iterated query):

- ❑ 被查询的名字服务器 回复可以被查询的名字服务器的IP地址
- ❑ “我不知道它的名字，但是你可以问服务器xx”



DNS缓存和权威DNS记录更新

- 一旦名字服务器获得DNS映射, 它将缓存该映射到局部内存
 - 服务器在一定时间后将丢弃缓存的信息
 - 本地DNS服务器可以缓存TLD服务器的IP地址
 - 因此根DNS服务器不会被经常访问
- 权威DNS服务器记录更新: IETF动态更新/通报机制
 - RFC 2136

DNS记录

DNS: 存储资源记录(RR, Resource Records)的分布式数据库

RR 格式: (name, value, type,ttl)

□ **Type=A (Address)**

- name = 主机名
- value = IP地址

□ **Type=CNAME (canonical)**

- name = 主机别名
**www.ibm.com的真名为
servereast.backup2.ibm.com**
- value = 真实的规范主机名

DNS记录

DNS: 存储资源记录(RR, Resource Records)的分布式数据库

RR 格式: (name, value, type,ttl)

- Type=NS
(name server)
 - name = 域名 (如foo.com)
 - value = 该域权威名字服务器的主机名

- Type=MX (mail exchange)
 - name = 邮件服务器的主机别名
 - value = 邮件服务器的真实规范主机名

DNS协议, 消息

DNS协议：查询报文与应答报文，但具有同样的报文格式

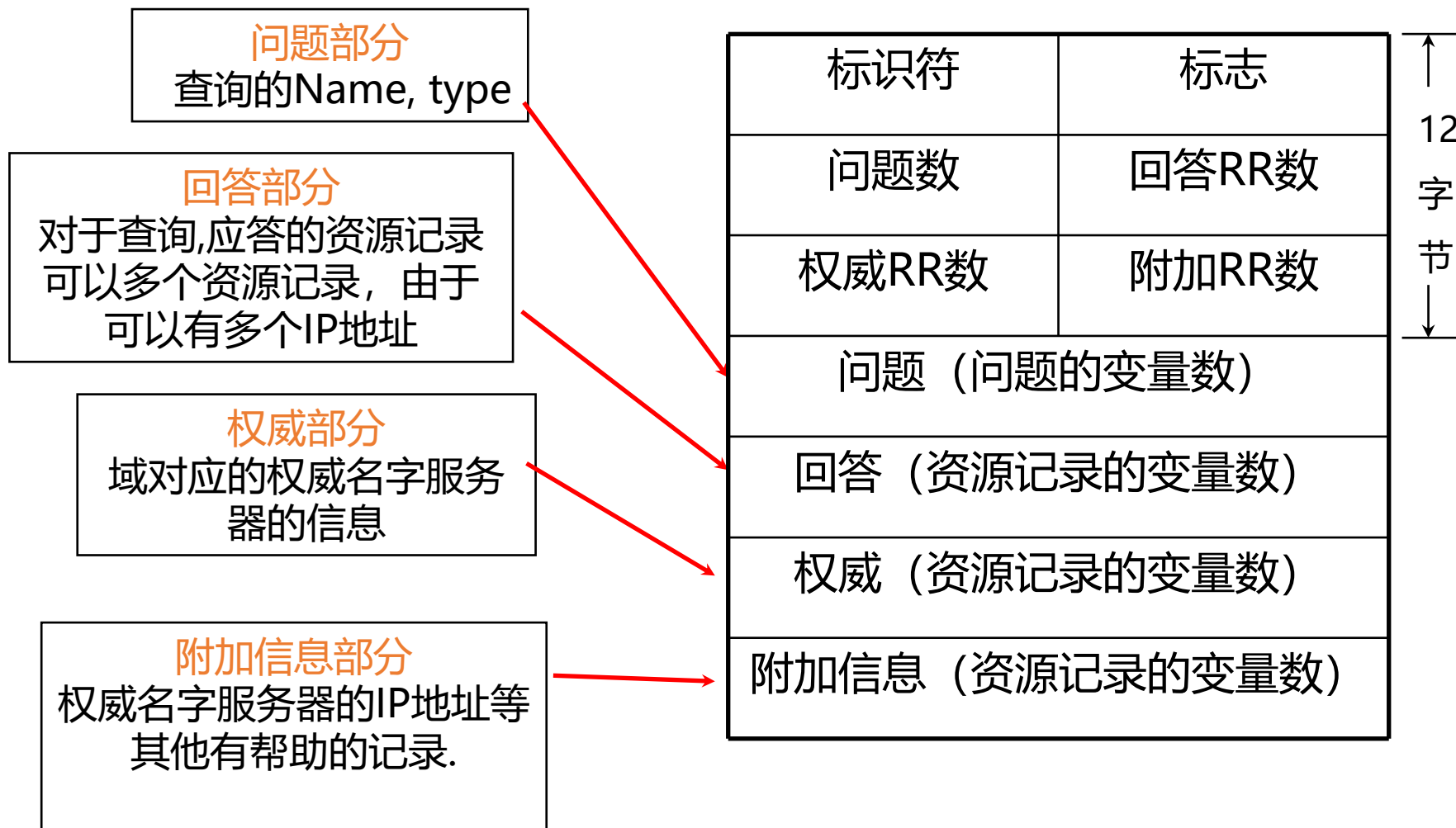
报文头部

- ❑ 标识符: 16位, 查询和应答报文使用相同的标识符
- ❑ 标志: 有若干个标志构成, 分别标识不同的功能
 - 查询/应答 - 0/ 1
 - 查询希望是/非递归查询 - 1/0
 - 应答可/否获得(支持)递归查询 - 1/0
 - 应答是/否来自权威名字服务器 - 1/ 0

标识符	标志
问题数	回答RR数
权威RR数	附加RR数
问题 (问题的变量数)	
回答 (资源记录的变量数)	
权威 (资源记录的变量数)	
附加信息 (资源记录的变量数)	

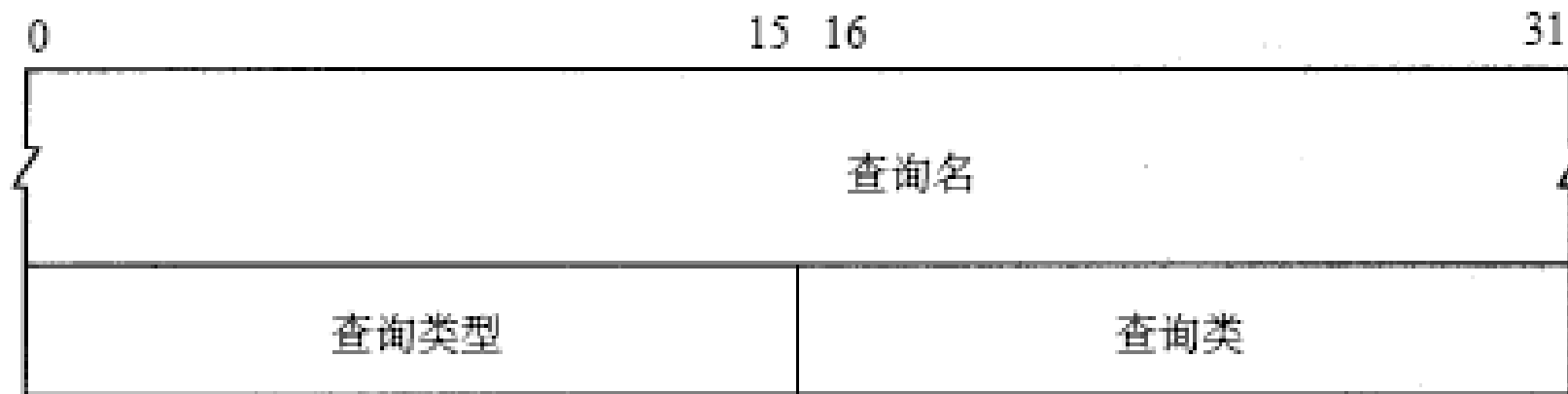
↑
12
字
节
↓

DNS协议, 消息



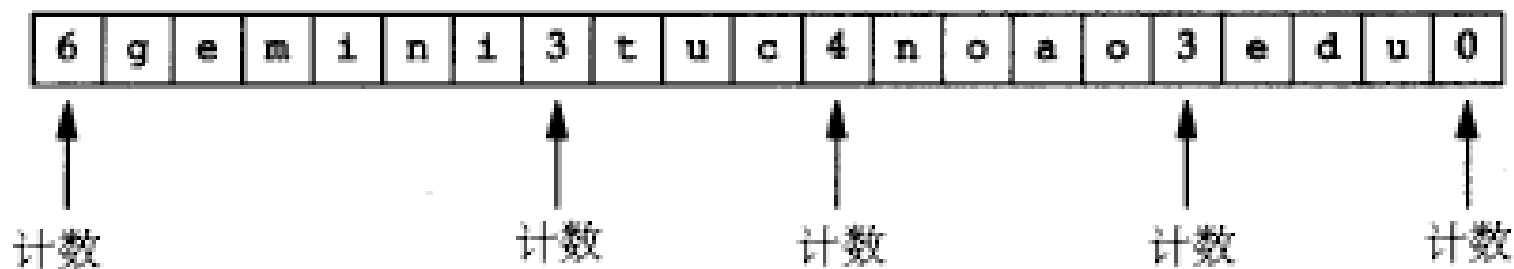
DNS协议, 消息*

DNS消息中问题部分的格式



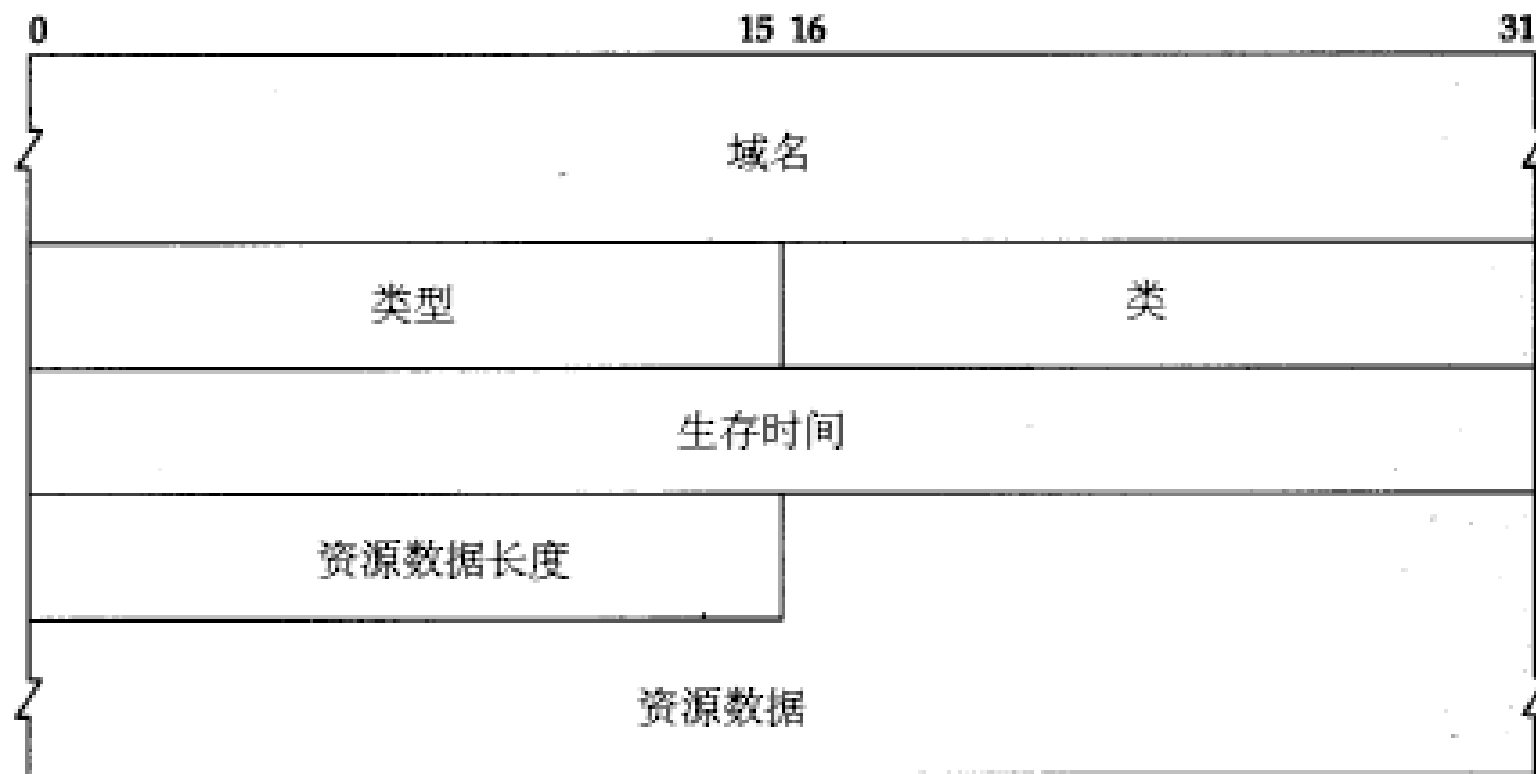
查询名示例:

例如: 查询名为 `gemini.tuc.noao.edu`



DNS协议, 消息*

DNS消息中回答、权威、附加部分的格式 (类同DNS数据库记录中的RR)



DNS协议, 消息*

DNS消息中回答、权威、附加部分的格式示例（用wireshark抓包得到）：

```
Domain Name System (response)
  [Request In: 739]
  [Time: 0.003302000 seconds]
  Transaction ID: 0x8759
  + Flags: 0x8180 (standard query response, No error)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 0
  - Queries
    + reg.eol.cn: type A, class IN
  - Answers
    - reg.eol.cn: type A, class IN, addr 121.194.3.182
      Name: reg.eol.cn
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 5 minutes
      Data length: 4
      Addr: 121.194.3.182
  - Authoritative nameservers
    - eol.cn: type NS, class IN, ns dns1.cernet.cn
      Name: eol.cn
      Type: NS (Authoritative name server)
      Class: IN (0x0001)
      Time to live: 42 minutes, 20 seconds
      Data length: 14
      Name server: dns1.cernet.cn
    + eol.cn: type NS, class IN, ns dns2.cernet.cn
```

在DNS数据库中插入记录

例子：刚刚创建一个“网络乌托邦”公司

- 如果你想在注册登记机构注册你的域名network.com，则
 - 需要提供你自己的基本权威DNS服务器和辅助权威DNS服务器的名字和IP地址
 - 该注册登记机构将下列两条资源记录插入注册机构的DNS系统中：

(network.com, dns1.network.com, NS)

(dns1.network.com, 212.212.212.1, A)

在DNS数据库中插入记录

- 如果你想建立一个网站，则可以将网址www.network.com以类型A的方式记录到你的权威DNS服务器dns1.network.com中。
- 如果你想建一个邮件服务器，则可以将mail.network.com以类型MX的方式记录到你的权威DNS服务器dns1.network.com中。
- 人们怎样得到你的Web站点的IP地址呢？

演示：nslookup程序的使用

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

-t TYPE      - list records of the given type (e.g. A,CNAME,MX,NS,PTR etc.)
view FILE    - sort an 'ls' output file and view it with pg
exit         - exit the program

> server 8.8.8.8
Default Server:  google-public-dns-a.google.com
Address:  8.8.8.8

> www.facebook.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

*** No address (A) records available for www.facebook.com

> www.twitter.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

*** No address (A) records available for www.twitter.com

> www.cctv.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
Name:      a411.b.akamai.net
Addresses:  60.254.142.33, 60.254.142.24
Aliases:   www.cctv.com, g1.cctvcdn.net, www.cctvgeo.akadns.net
           www.cctv.com.edgesuite.net

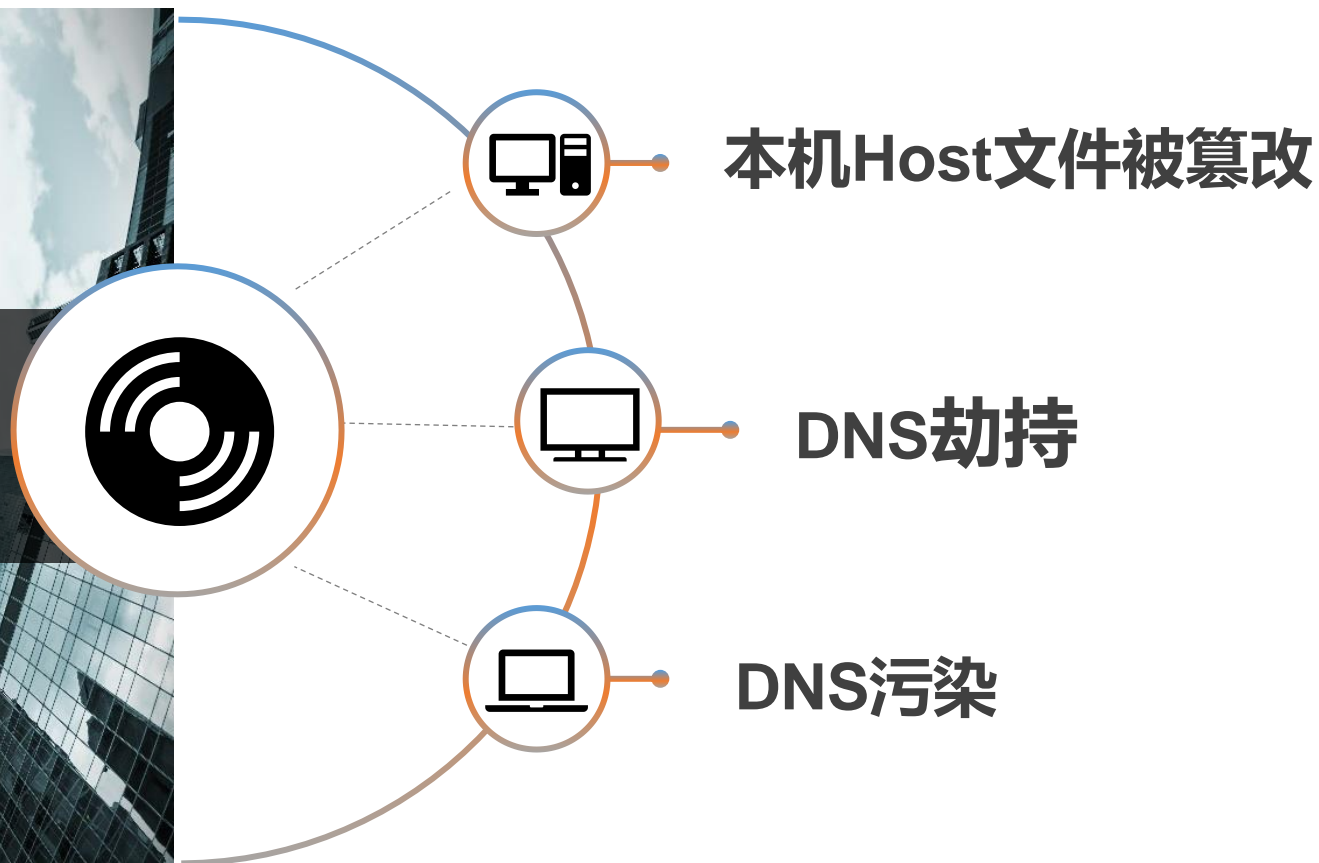
>
```

DNS攻击

- DDoS攻击：对根域名服务器或顶级域名服务器发起拒绝服务攻击
- 重定向攻击：中间人攻击、DNS中毒攻击（发送欺骗的域名解析结果给DNS服务器）
- 利用DNS实现DDoS攻击：DNS反弹式拒绝服务攻击(DNS reflector attacks, 又称DNS amplification attacks)。伪造客户地址向大量的dns服务器发出请求，导致客户无法访问dns服务器进行域名解析。

DNS攻击

常见攻击方法



DNS与国家安全

● 域名解析控制权现状

- 全球根域名服务器多数在美国
- 域名服务器的管理者ICANN是非盈利性机构，且已经从美国政府脱离，但是它仍然是受美国法律约束的组织，在事实上，它对根域名服务器的管理都要向美国政府申报

● 域名解析由某国家控制的潜在问题

- 一旦与某国发生冲突，可以在技术上阻碍对该国域名的解析。伊拉克战争期间，美国终止了伊拉克国家顶级域名.IQ的解析；
- 在塔利班政权统治阿富汗时期，美国将阿富汗国家顶级域名.AF的管理权授予前流亡政府；
- 2004年4月，由于对顶级域名管理权问题发生分歧，导致利比亚国家顶级域名.LY瘫痪，利比亚从互联网上“消失”了3天

DNS与国家安全

提高我国域名系统安全性的建议

- 加强国家网络空间安全的战略谋划
 - 加强网络空间安全筹划和顶层设计
 - 积极参与国际互联网治理
- 增强国家域名系统的安全防护能力
 - 建立互联网安全应急替代机制
- 以网络技术发展为契机，抢占先机
 - 充分利用全球下一代网络发展契机，建立新框架（IPV6,物联网等）
 - 充分利用新型网络应用的发展，降低对现有域名体系的依赖

06

P2P技术*

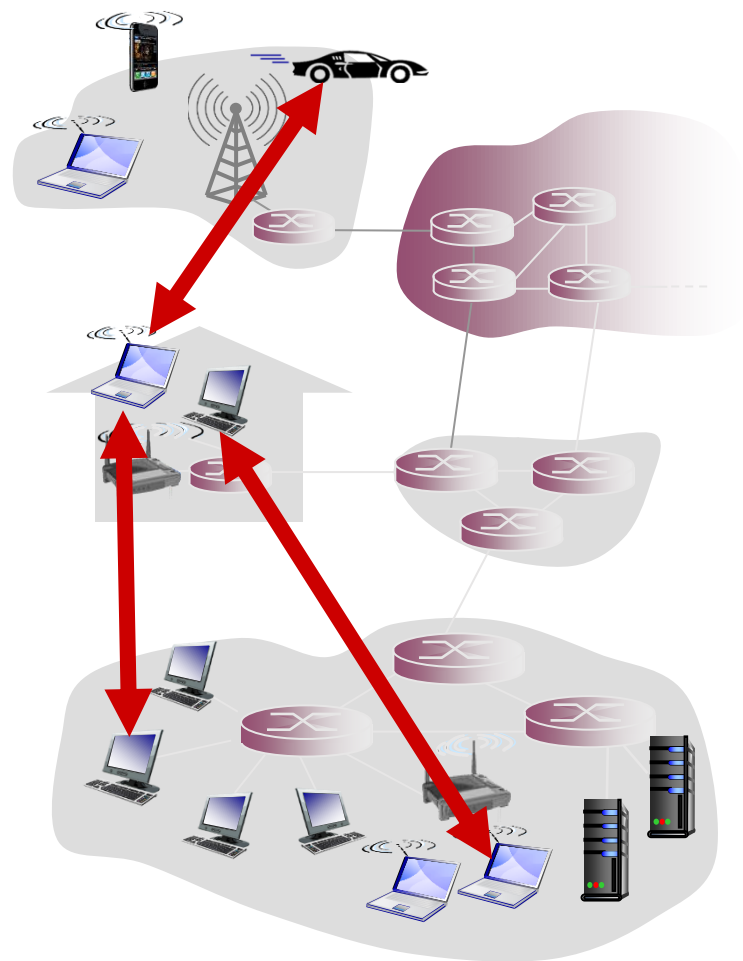


纯P2P架构

- 没有总是在线的服务器
- 任意端系统之间直接通信
- 对等方之间可以间断连接并可以改变IP地址

例子：

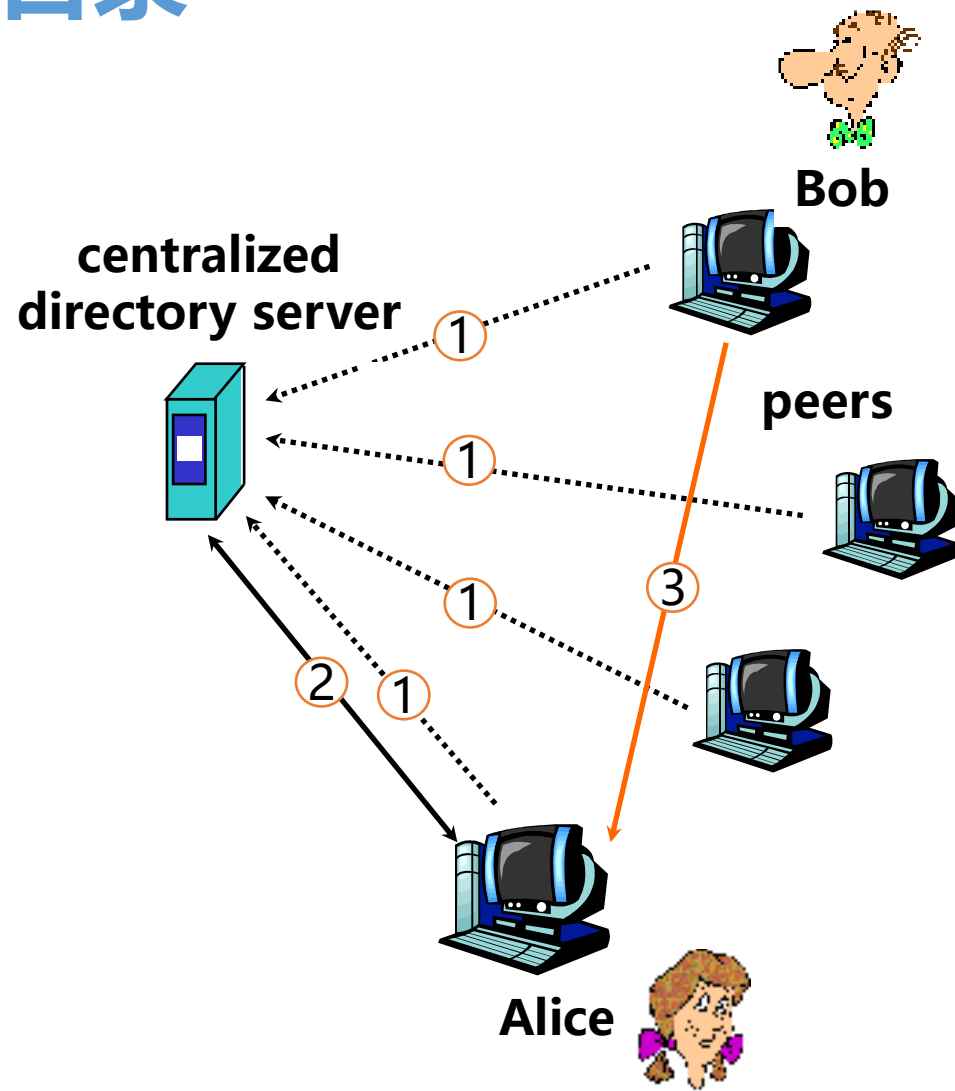
- 文件分发
- 流媒体
- VoIP



P2P: 集中式目录

“Napster” 公司的设计

- 1) 当对等方启动时，它通知目录服务器以下信息
 - IP地址
 - 可供共享的对象名称
- 2) Alice查询文件 “Hey Jude”
- 3) Alice 向Bob请求文件



P2P: 集中式目录的问题

- 单点故障
- 性能瓶颈
- 侵犯版权

文件传输是分散的,
但是定位内容的过程
是高度集中的



查询洪泛: Gnutella

- 全分布
 - 没有集中式服务器
- 公共域协议
- 许多Gnutella客户机实现Gnutella协议

覆盖网络: graph

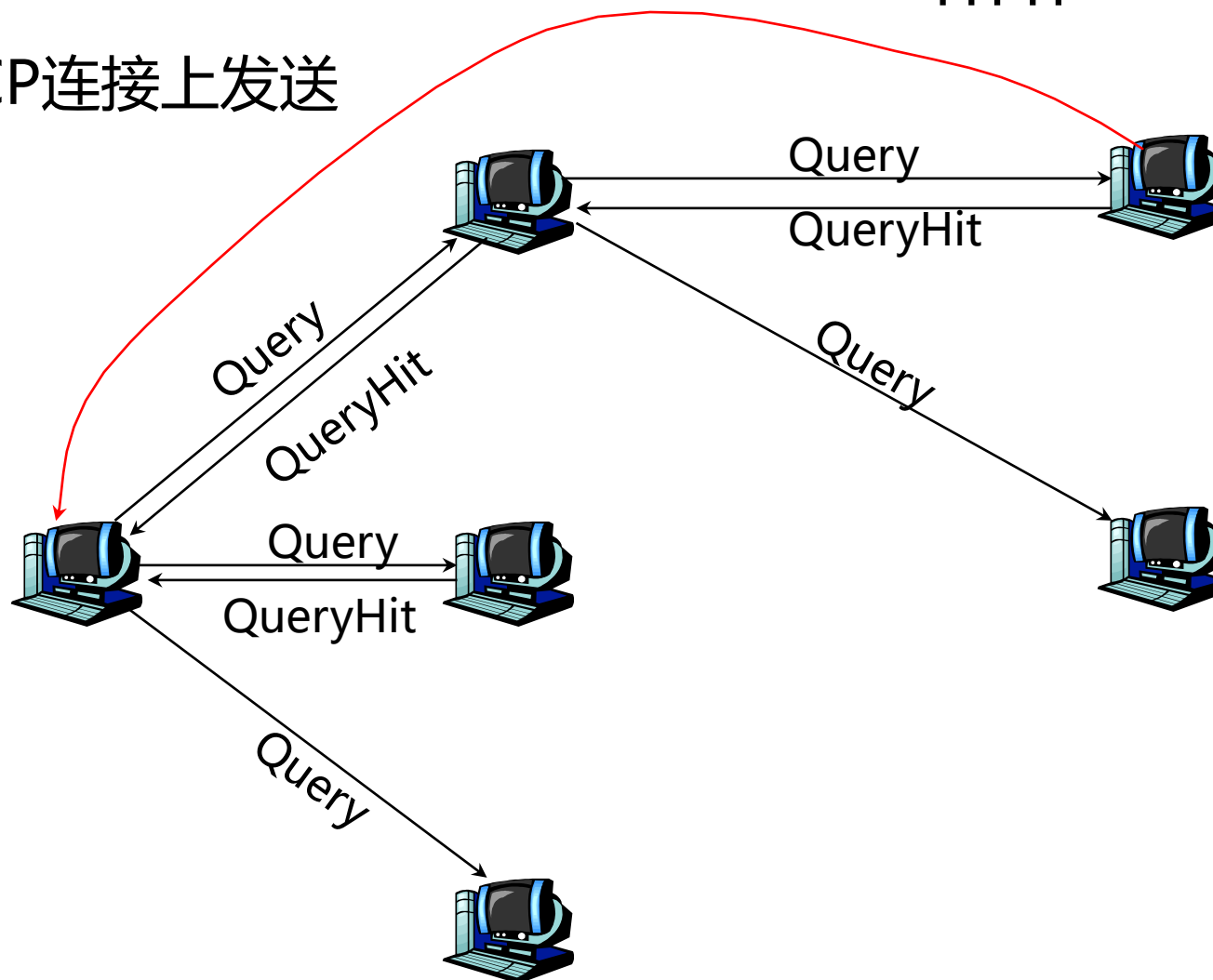
- 如果对等方X和Y维护了一条TCP连接, 则说X和Y之间有一条边
- 所有活跃的对等方和边组成覆盖网络
- 边不是物理通信链路
- 给定对等方连接的覆盖网络路径中的节点少于10个, 即TTL小于10

Gnutella: 协议

File transfer:
HTTP

- ❑ 查询报文在已有的TCP连接上发送
- ❑ 对等方转发报文
- ❑ QueryHit 报文按反向路径传送

可扩展性:
限范围泛洪



Gnutella: 加入对等方

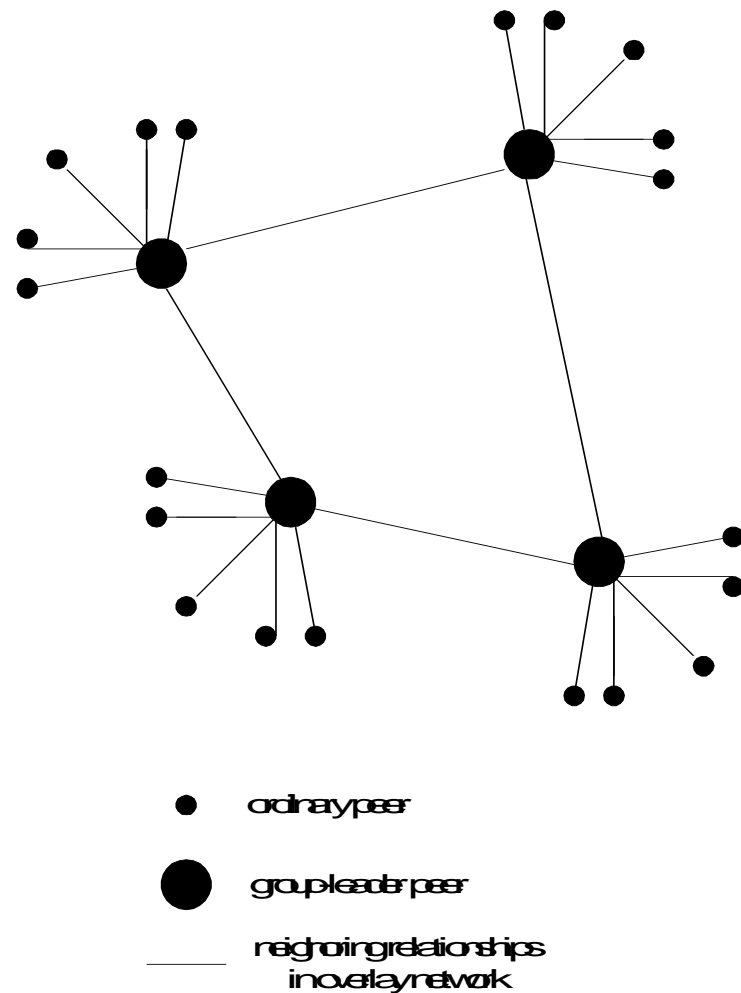
- ❑ 加入对等方X必须发现在Gnutella网络中的其他对等方：使用对等方列表。
- ❑ X试图与该列表上的对等方建立一条TCP连接，直到与Y创建一条连接。
- ❑ X向Y发送一个Ping报文；Y转发该Ping报文。
- ❑ 所有的对等方接收Ping报文并响应一个Pong报文。
- ❑ X接收到许多Pong报文。然后能同某些其他对等方建立TCP连接。

Gnutella: 对等方离开

- ❑ 主动离开:离开接点的所有对等方都会刷新自身的激活对等方列表,并开始与列表中的新的对等方建立连接
- ❑ 断网:发送信息的时候对等方没有响应,则表明对等方离开,节点刷新自身的激活对等方列表,并开始与列表中的新的对等方建立连接

KaZaA

- 每个对等方要不被指派为组长，要不被指派给一个组长
 - 对等方和组长之间建立TCP连接
 - 组长之间建立TCP连接
- 组长维护它的子对等方共享的内容



KaZaA

- 每个文件有文件的散列码标识
- 客户机送向组长发送关键词的查询
- 组长响应匹配
 - 逐项匹配：元数据，散列值，IP地址
- 如果组长转发查询给其他组长则其他组长响应匹配
- 客户端选择要下载的文件

KaZaA

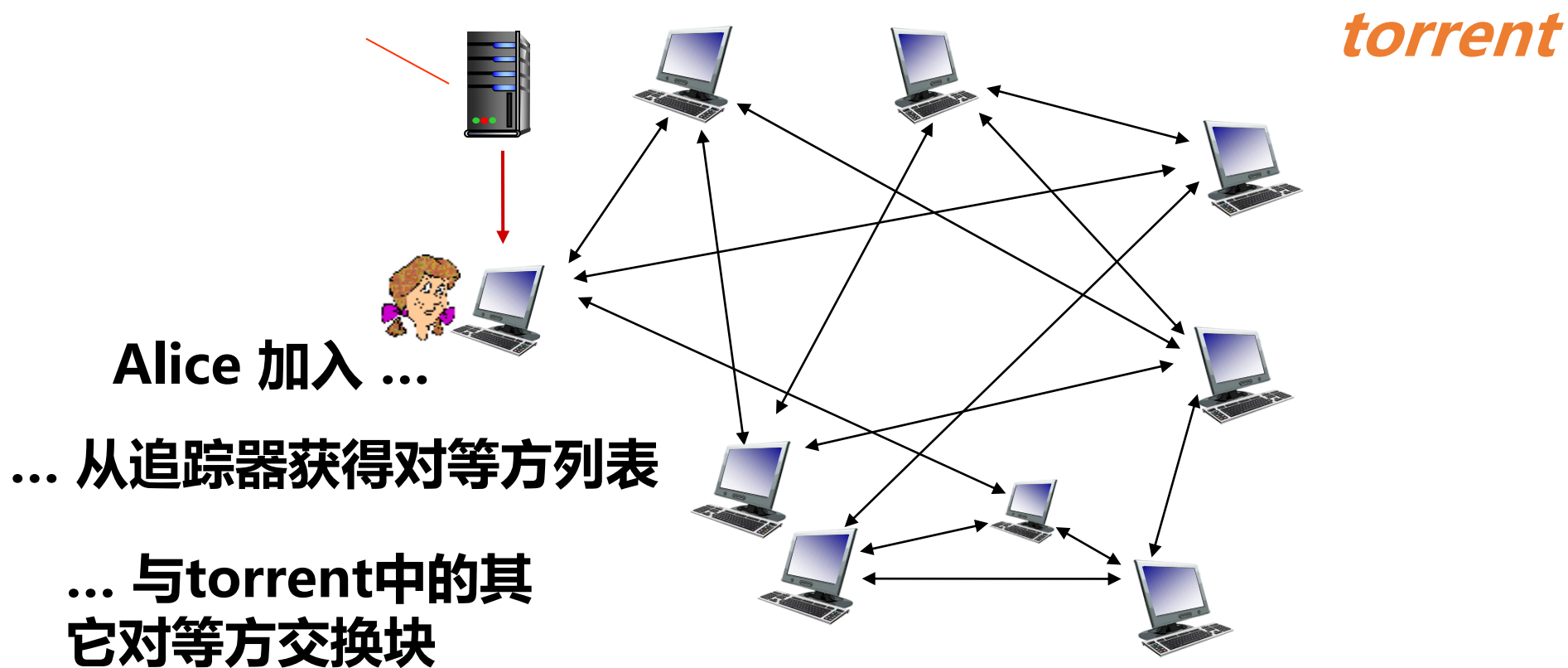
- 请求排队：限制对等方并行上载数量，新的请求进行排队。
- 激励优先权：根据不同的上载下载比例优先服务贡献大者。
- 并行下载：将一个文件分成若干段，从多个对等方并行下载。

P2P文件分发-BitTorrent

- BitTorrent是一种用于文件分发的流行P2P协议。
- 参与一个特定文件分发的所有对等方的集合被称为一个洪流 (torrent) 。
- 一个洪流中的对等方彼此下载等长度的文件块 (chunk) , 典型块长度为256KB。

P2P文件分发-BitTorrent

追踪器tracker: 跟踪参与洪流的所有对等方



P2P文件分发-BitTorrent

- 对等方加入 torrent:
 - 没有文件块，但会随着时间流逝从其它对等方处累积文件块
 - 在tracker处注册，取得对等方列表，连到所有对等方的一个子集（邻居）
- ❖ 在下载的同时给其它对等方上传文件块
- ❖ 对等方可能改变和其交换文件块的对象
- ❖ 对等方会不断进入或者离开
- ❖ 一旦某对等方下载完了整个文件，它可以离开（自私）或者继续留在torrent系统里（无私）

BitTorrent: 请求、发送

□ 请求文件块

- ❖ 在任何给定的时刻，不同的对等方拥有不同的文件块子集
- ❖ 每个对等方会周期性的询问其它每个它连接的对等方当前所拥有的文件块列表
- ❖ 对等方将请求下载最稀缺的文件块

□ 发送文件块: tit-for-tat(一报还一报)

- ❖ Alice发送文件块的对象是所有邻居中向自己发送速率最快的4个
 - 其它邻居被阻塞
 - 每10秒重新计算速率
- ❖ 每30秒，随机选择一个其他邻居，发送文件块

DHT(分布式Hash表)

- ❖ DHT: 一个分布式的P2P 数据库
- ❖ 数据库由许多(key,value) ((键, 值)) 对构成。例如:
 - ❑ key: 社保号; value: 人名
 - ❑ key: 电影名称; value: IP地址
- ❖ 所有(key, value) 对被分发到成千上万的对等方用户群中
- ❖ 一个对等方利用key来查询DHT
 - ❑ DHT返回与之匹配的value
- ❖ 对等方还可以插入(key, value)对

Q: 怎样把键值分配给对等方?

- 核心问题:
 - 分配 (key, value) 对给各对等方
- 基本思想:
 - 把每个key转化成整数
 - 给每个对等方分配一个整数标识符
 - 把 (key,value) 对分配给标识符离key最近的
那个对等方

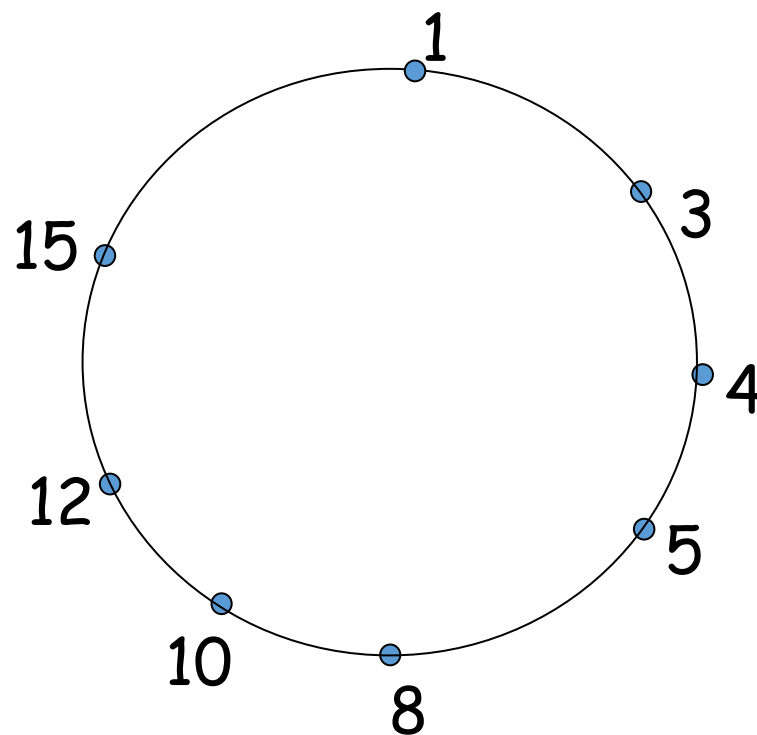
DHT 标识符

- 给每个对等方分配一个 $[0, 2^n - 1]$ 之间的整数标识符, n 为某给定值.
 - 每个标识符由 n 比特构成.
- 需要每个key也在同样的范围内
- 为得到整数key, 将原key做hash
 - 例如, $\text{key} = \text{hash}(\text{"Led Zeppelin IV"})$
 - 这就是为什么叫做分布式hash表的原因

将key分配给对等方

- 规则：把key分配给具有最邻近ID的对等方.
- 为方便起见: 最临近被定义为该key的直接后继 (immediate successor)
- 例如:
 - $n=4$; peers: 1,3,4,5,8,10,12,14;
 - key = 13, then successor peer = 14
 - key = 15, then successor peer = 1

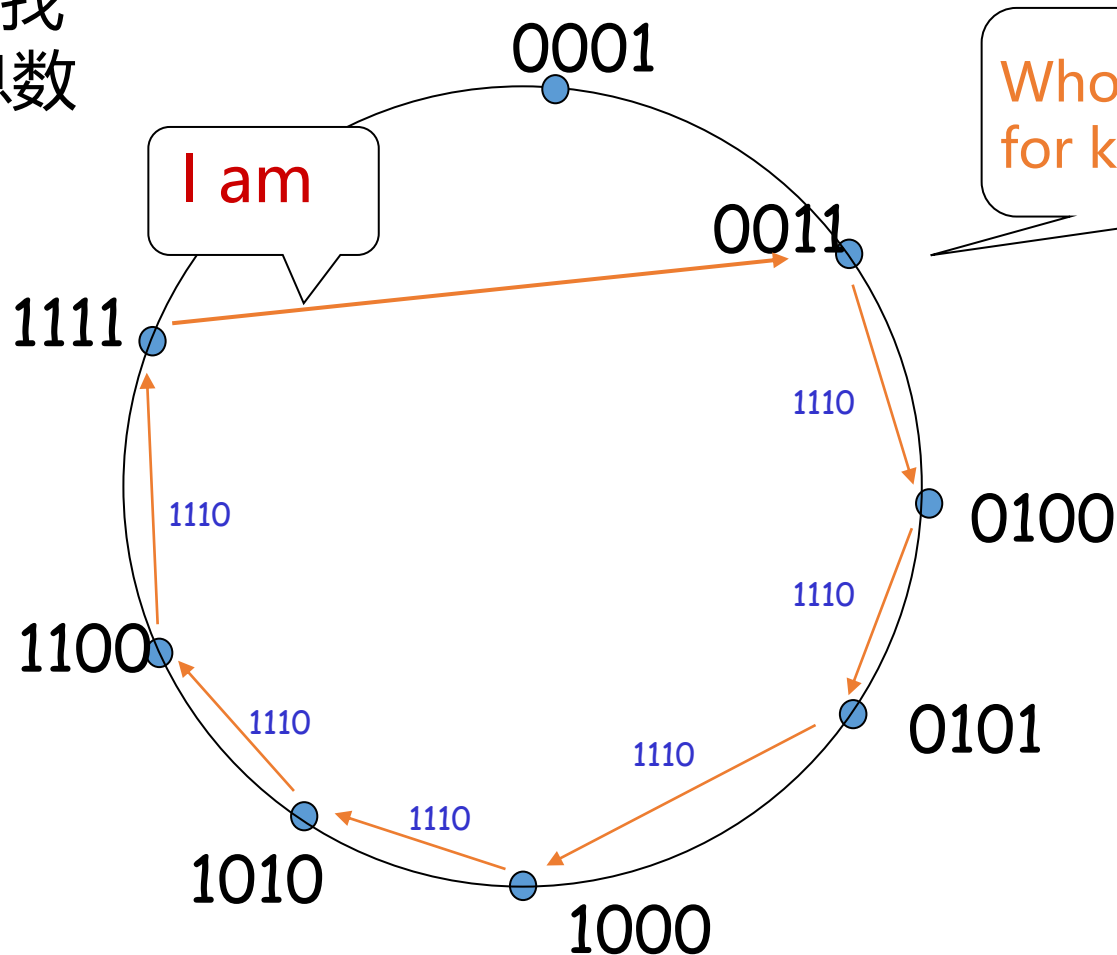
环形 DHT



- 每个对等方仅和其直接后继和直接前任(predecessor)联系.
- “覆盖网络”

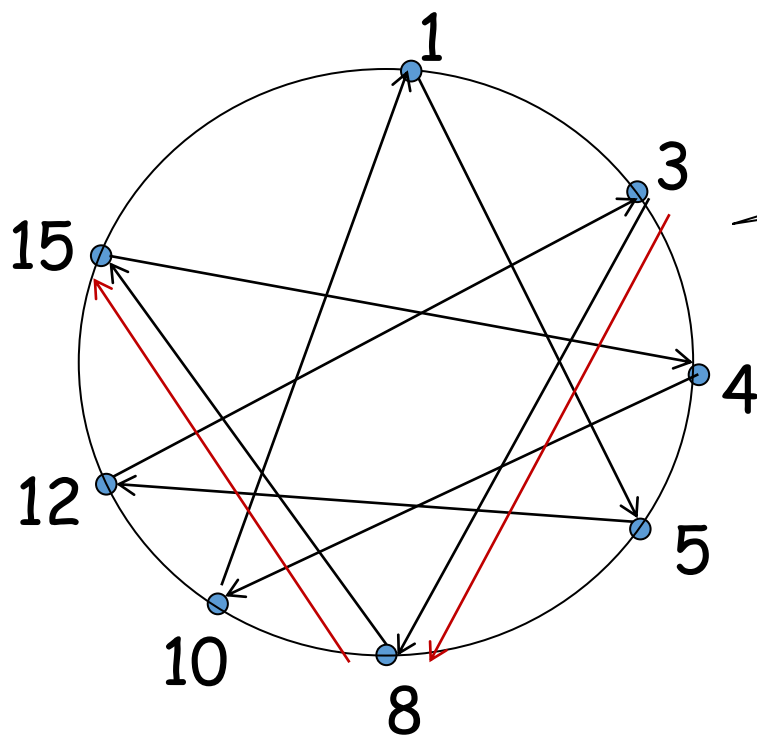
环形 DHT

当有 N 个对等方时，为找到负责的键，发送消息数量的负责度是 $O(N)$



将最临近定义为最近的后继者

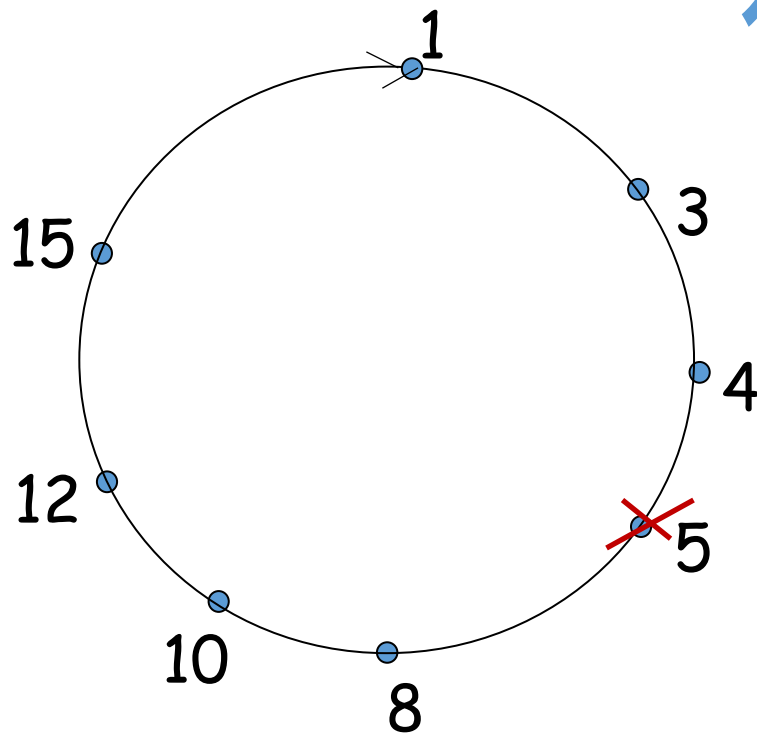
带捷径的环形DHT



Who's responsible for key 1110?

- 每个对等方知晓直接前任、后继以及捷径方的IP
- 本例中，将消息数从6减至2
- DHT可以设计为每个对等方的邻居和每个请求的报文数均为 $O(\log N)$

对等方扰动



- ❖ 对等方可能进入或者离去
- ❖ 对等方需要知晓它后面两个后继的地址
- ❖ 每个对等方周期性的ping这两个后继以检查它们的存活性
- ❖ 如果直接后继离开了，则选择它的下一个后继作为直接后继

例如：peer 5离开

- peer 4检测到5的离开，将8当作其直接后继，并且问 8 它的直接后继是谁（10），然后将10当作其第二后继。
- 如果编号为13的peer要加入怎么办？

07

内容分发网络(CDN)



内容分发网络(Content distribution networks, CDN)

- 挑战: 如何从海量的视频中, 挑选出某些内容, 采用流的方式发送给成千上万的用户?
- 选项1: 使用单点, 庞大的服务器
 - 单点故障
 - 单点网络拥塞
 - 远距离用户的访问路径很长
 - 输出链路上发送同一视频的多份重复拷贝

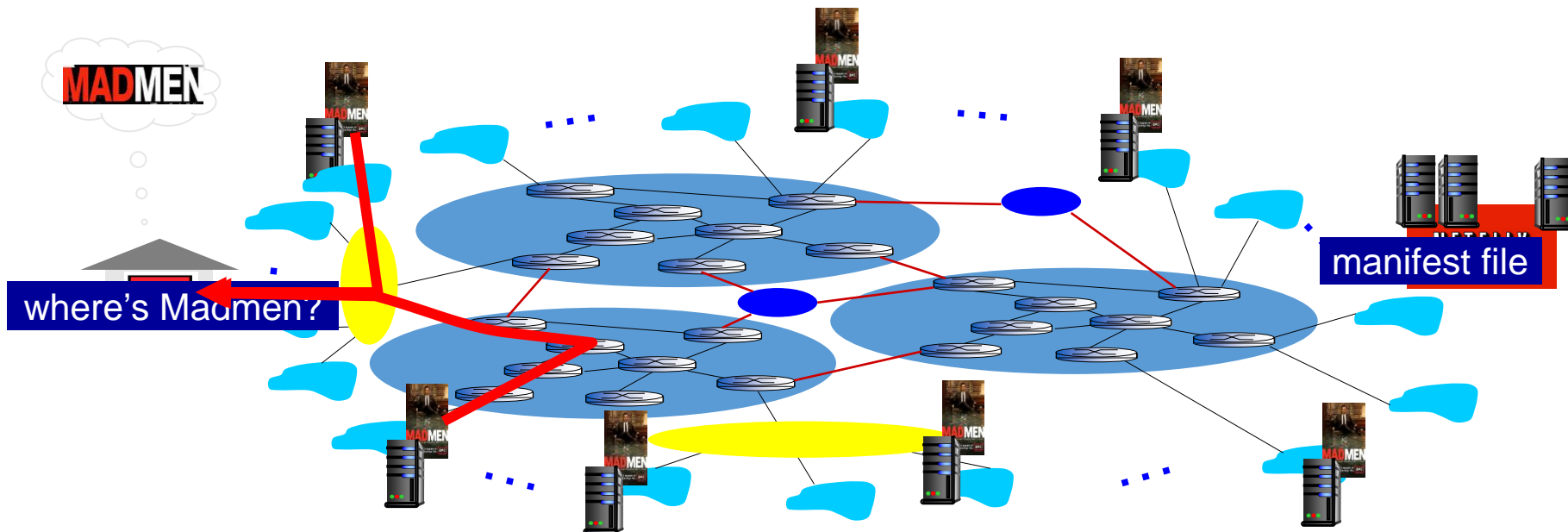
....很显然: 这种方案不具有可扩展性

内容分发网络(Content distribution networks, CDN)

- 挑战: 如何从海量的视频中, 挑选出某些内容, 采用流的方式发送给成千上万的用户?
- 选项2: 将多份拷贝存储在地理上分散的不同站点来提供服务(CDN)
 - 深入: 将CDN服务器部署在众多的接入网络中
 - 靠近用户
 - Akamai首创, 使用了1700多个位置
 - 邀请做客: 在少量(例如10个)靠近接入网的关键位置(例如IXP), 建造大集群, 邀请到ISP做客
 - Limelight等使用

内容分发网络(Content distribution networks, CDN)

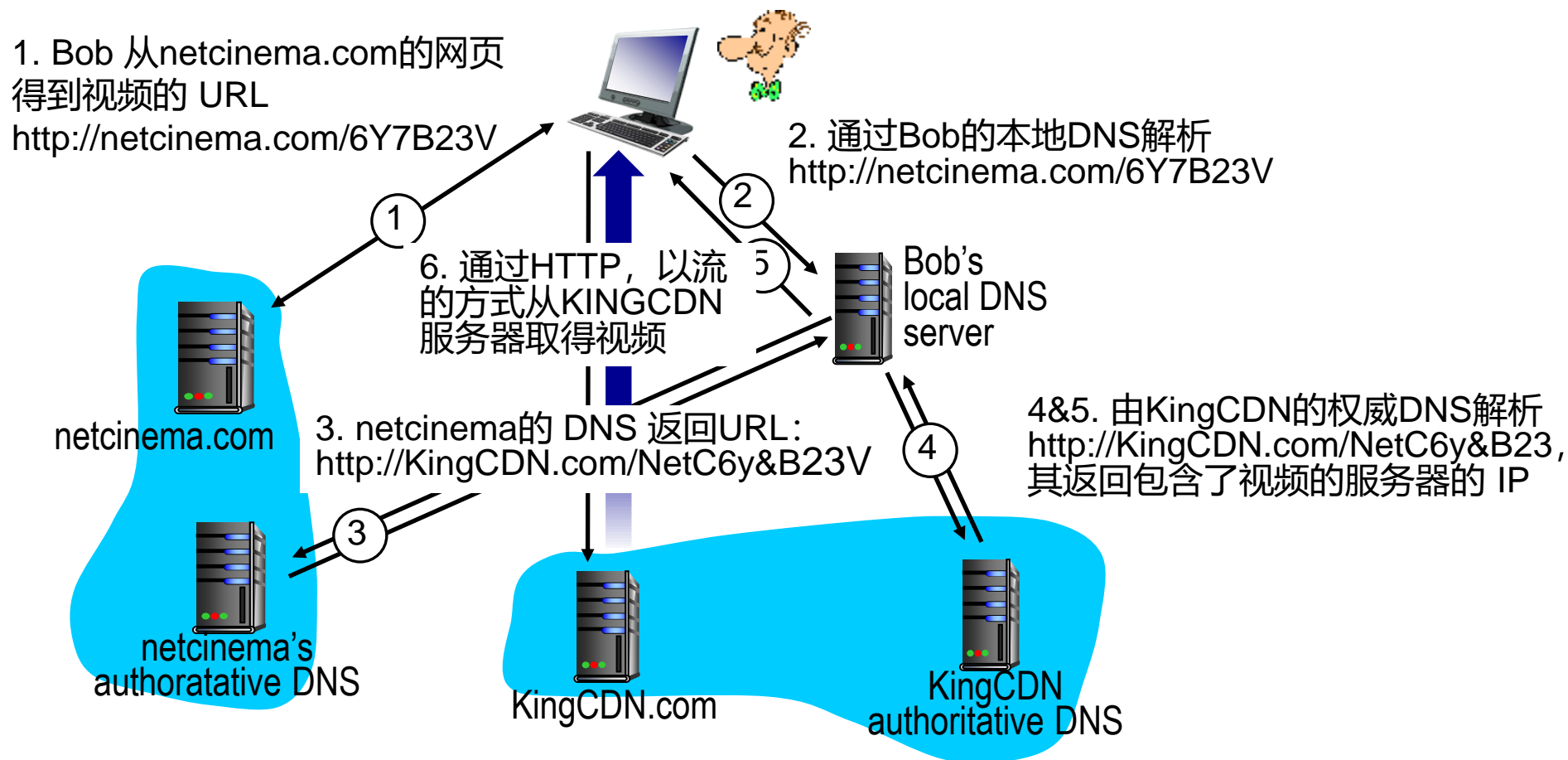
- CDN: 将内容的拷贝存储在CDN节点中
- 用户向CDN请求内容
 - 被定向到附近的拷贝, 取得内容
 - 如果网络路径拥塞, 则可能选择其他的拷贝



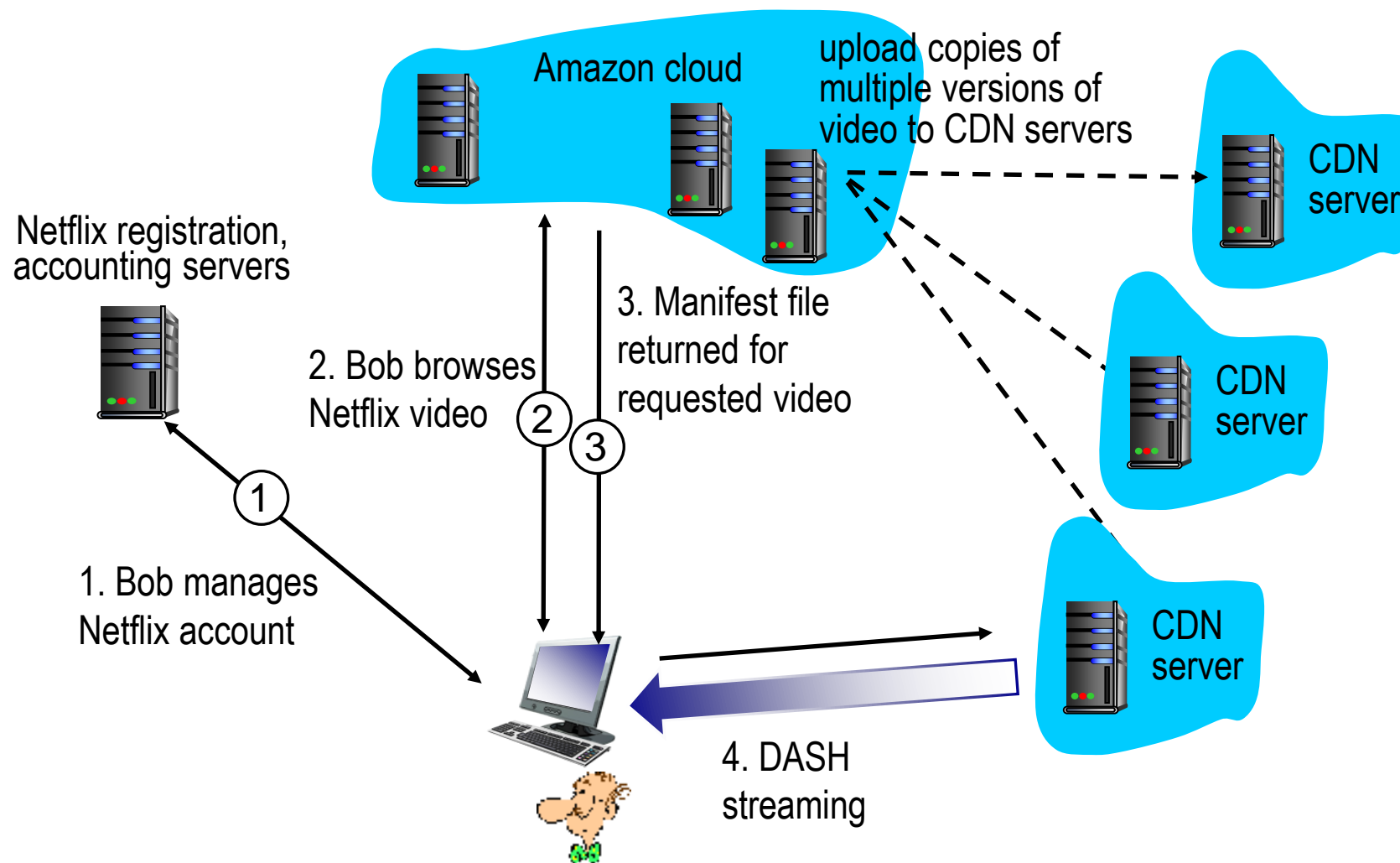
CDN 内容访问

Bob (client) 请求视频: <http://netcinema.com/6Y7B23V>

- 视频存储在CDN上, 地址为<http://KingCDN.com/NetC6y&B23V>



案例研究: Netflix *



08

构造一个简单的Web 服务器*



构造一个简单的Web服务器

- 仅处理一个HTTP请求
- 接收请求
- 解析头部
- 从服务器文件系统中得到所请求的文件
- 生成HTTP响应报文:
 - 首部 + 文件
- 向客户机发送响应报文
- 安装服务器后可以在其他主机上使用浏览器（如IE），向服务器请求一个文件

具体需求

服务器基本流程：

- 创建套接字
- 获取HTTP请求，并解析HTTP请求报文
- 显示请求报文各字段的字段名和值，并进行说明
- 根据HTTP请求报文获得对象文件路径名
- 根据路径名打开本地文件
- 封装本地文件到HTTP响应报文
- 使用套接字发送HTTP相应报文

小 结

- 应用程序体系结构
 - 客户机/服务器
 - P2P
 - 混合
- 应用服务器需要的服务
 - 可靠, 带宽, 延时
- 网络运输层协议
 - 面向连接, 可靠: TCP
 - 不可靠, 数据报: UDP
- 通用协议:
 - HTTP
 - FTP
 - SMTP, POP, IMAP
 - DNS

小 结

重点: 掌握应用层协议

- 典型的请求/应答消息交换:
 - 客户请求信息或服务
 - 服务器以数据或状态码应答
- 消息格式:
 - 头部
 - 数据
- 控制报文 vs. 数据报文
 - 带内, 带外
- 集中 vs. 分散
- 可靠 vs. 不可靠报文传输
- “网络边缘的复杂性”

第二章：复习大纲

- 网络应用程序体系结构
- Web应用和HTTP协议
 - 基本术语（网页、URL等）
 - HTTP的特性及其区别(无状态、非持久和持久等)
 - 请求和响应报文
 - COOKIE技术
 - Web缓存
- 文件传输协议FTP：两种连接*
- 电子邮件：组成及其使用的协议
- DNS的功能和实现
- P2P文件共享原理和实现技术*
- CDN
- 构造简单的web服务器*