

华中科技大学电信学院 2016

网络应用

计算机网络
计算

学习目标

- 了解基于HTTP的Web服务；
- 了解基于DNS的互联网域名管理系统；
- 了解面向内容分发的互联网新技术，包括P2P、CDN等。





引言

核心问题：应用需要自己的协议

- 传统应用
- 多媒体应用
- 基础设施服务
- 覆盖网络
- 总结

网络应用实例

- E-mail (Gmail)
- Web应用 (Renren, Tudou)
- 即时通信 (QQ)
- 远程登录 (ssh, telnet)
- P2P 文件共享 (Gnutella, eMule, BitTorrent)
- 多用户的网络游戏 (CGA)
- 流媒体视频播放 (迅雷)
- IP电话 (Skype)
- 实时视频会议
- 网格计算 (Grid)
- ...



编写一个网络应用程序



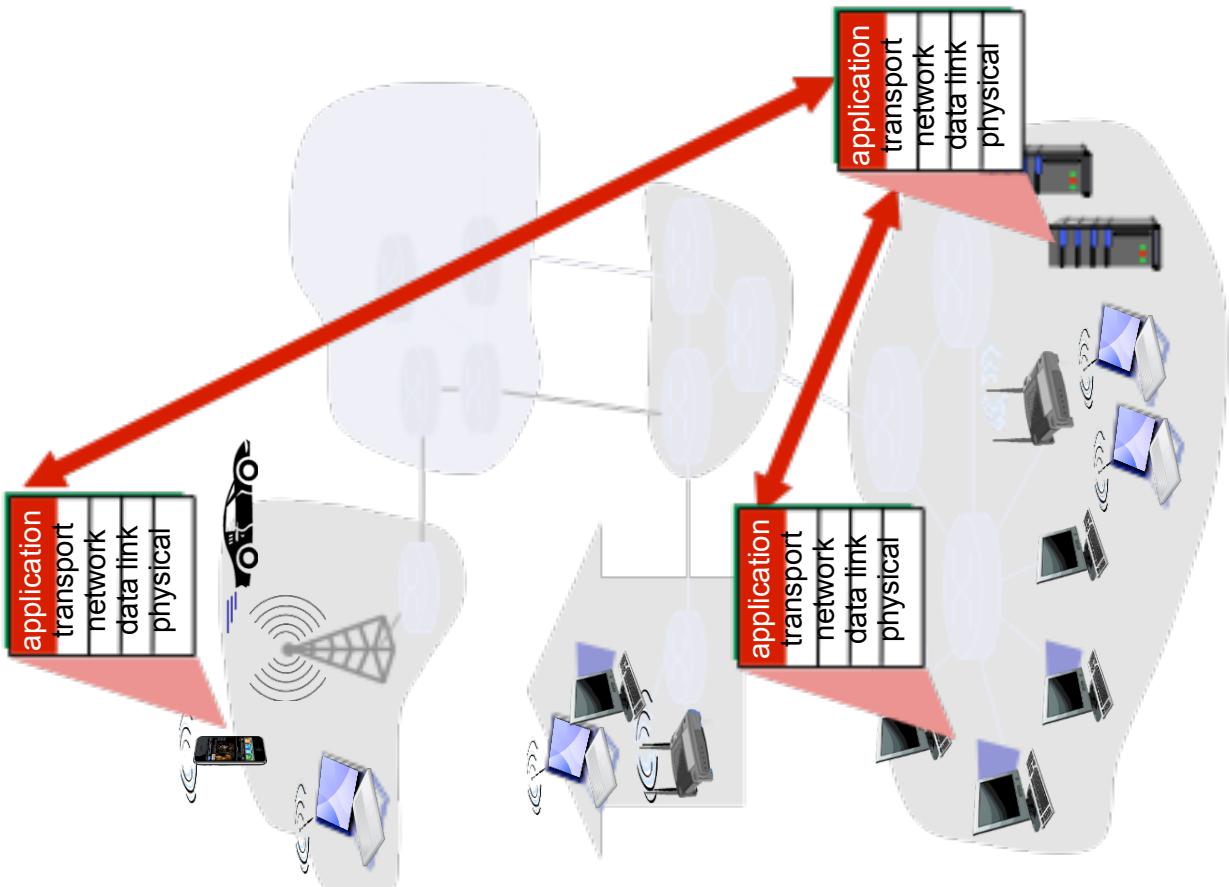
程序满足以下条件：

- 运行在不同的端结点
- 通过网络通信
- 例如：Web服务器程序与浏览器程序的通信

不需要为网络核心设备编写程序

网络核心设备不运行用户程序

端结点的应用程序能够迅速的开发和推广



应用层协议的定义



- 消息互换的类型
 - 例如，请求，响应
 - 消息格式：
 - 消息中有哪些区域，这些区域如何描述
 - 消息语义
 - 在不同区域中的信息有什么含义
 - 当进程发送和响应消息时应遵循的规则
- 公共协议：
 - 在 RFC 里面定义的
 - 允许共同使用
 - 例如，HTTP, SMTP
- 私有协议：
 - 例如，Skype



应用程序需要怎样的传输服务？

数据丢失

- 某些程序(例如音频方面的)
可以忍受某些数据丢失
- 其他的程序(例如文件传输、
远程登录)要求100%可靠的
数据传输

吞吐量

- 某些应用(例如流媒体)
要求最低吞吐量至少达到某个
值
- 其他弹性应用，无论怎样
的安全性

即时性

- 某些程序(例如网络游戏等)要求
低延时的实时性

加密，数据完整，...

网络应用常见传输服务需求



应用	数据丢失 容错	吞吐量	时间敏感性
文件传输	无要求	无	无
e-mail	不能丢失	无	无
Web文档	不能丢失	无	无
实时视频/音频	不能丢失	音频: 5kbps-1Mbps 视频: 10kbps-5Mbps	yes, 100's msec
存储视频/音频	容忍丢失	同上	几秒以内
交互式游戏	容忍丢失	很小的吞吐量	yes, 100's msec
即时通信	容忍丢失	无要求	yes and no
	不能丢失		

互联网传输协议所提供的服务

TCP 服务:

- 建立连接: 客户机和服务器必须建立连接
- 发送和接收进程间的可靠传输
- 流量控制: 发送端不会发送太快而导致接收端无法接收
- 拥塞控制: 当网络过载时降低发送的速率
- 不提供: 实时性, 最小吞吐量, 安全性

UDP 服务:

- 发送和接收进程间的不可靠数据传输
- 不提供:
 - 建立连接
 - 可靠性
 - 流量控制
 - 拥塞控制
 - 即时性
 - 吞吐量
 - 安全性

Q: 为什么还需要 UDP?





互联网应用：应用层协议以及传输协议

应用	应用层协议	下层的传输协议
e-mail	SMTP [RFC 2821]	TCP
远程登录	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流媒体	HTTP (例如 YouTube), RTP [RFC 1889]	TCP or UDP
网络电话	SIP, RTP, 私有协议 (例如, Skype)	一般为 UDP



核心問題

應用需要自己的协议



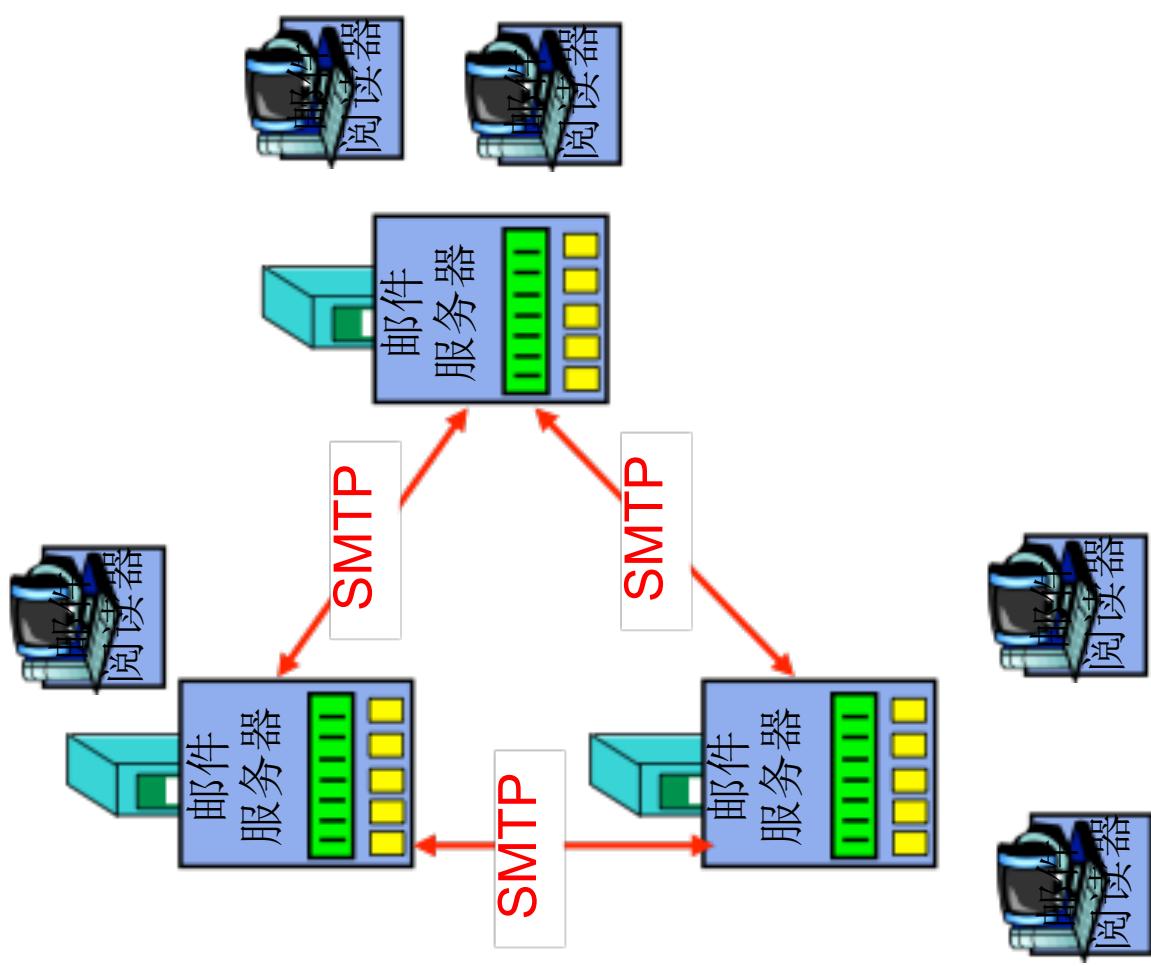
- 引言
- 核心问题：应用需要自己的协议
- 传统应用
- 电子邮件(SMTP, MIME, IMAP)
- 互维网(HTTP)
- Web服务
- 多媒体应用
- 基础设施服务
- 覆盖网络
- 总结

电子邮件：邮件服务器

邮件服务器

- 邮箱包含了用户接收到的消息，将要发送的邮件消息的消息队列

- **SMTP 协议** 邮件服务器之间发送邮件消息，客户端：发送邮件服务器“**服务器**”：接收邮件服务器



电子邮件: SMTP [RFC 2821]

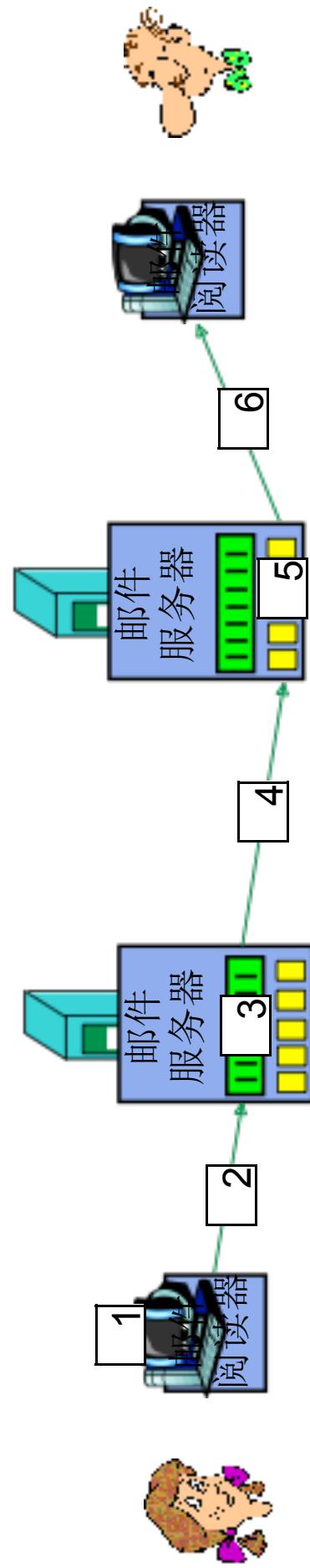
- 使用TCP协议从客户端向服务器可靠的发送电子邮件消息，端口号25
- 直接传输：发送服务器到接收服务器
- 传输的三个阶段
 - 握手（问候）
 - 消息发送
 - 终止
- 命令/响应的交互
 - 命令：ASCII文本信息
 - 响应：状态码与短语
- 消息必须是7位ASCII码



场景: Alice 向 Bob 发送消息



- 1) Alice 使用邮件阅读器撰写消息，然后发送到 bob@some school.edu
- 2) Alice 的邮件阅读器将消息发送给她自己的邮件服务器；消息进入待发送的消息队列
- 3) SMTP 的客户端与 Bob 的邮件服务器建立一个 TCP 连接
- 4) SMTP 客户端通过 TCP 连接将 Alice 的消息发送出去
- 5) Bob 的邮件服务器将消息发送到 Bob 的邮箱里
- 6) Bob 使用他的邮件服务器来阅读消息



SMTP交互示例



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
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

自己尝试使用SMTP协议进行交互

- telnet servername 25
- 观察从服务器返回的220状态码
- 输入 HELO, MAIL FROM, RCPT TO, DATA, QUIT 命令
- 以上的操作可让你不使用Email客户端就能发送邮件



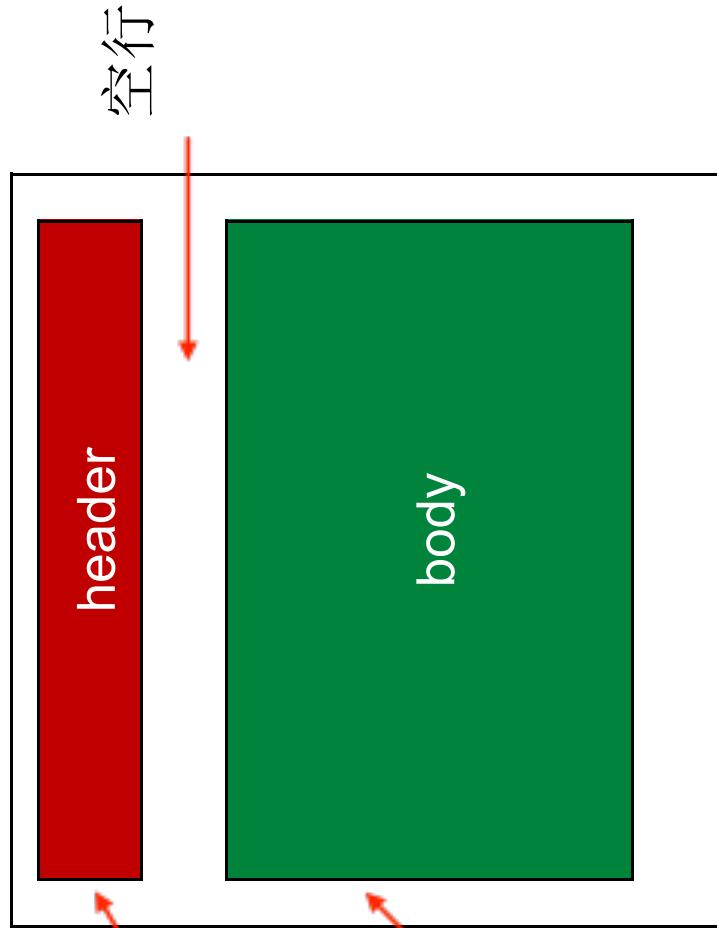
邮件消息格式

SMTP: 用来交换电子邮件消息的协议

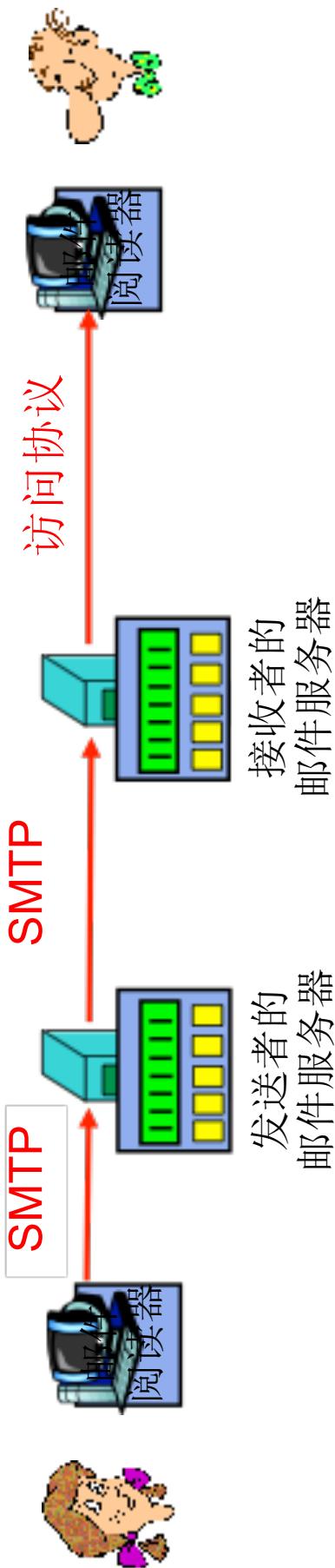
RFC 822: 文本消息格式的标准:

头信息行, 例如:

- To:
 - From:
 - Subject:
- 与SMTP协议的命令不同
- 正文
- 消息内容, 只含有ASCII码字母



邮件访问协议



- SMTP: 在接收者的服务器中发送/存储
- 邮件访问协议: 从服务器取得
- POP: 邮局协议 [RFC 1939]
 - 授权 (阅读器 <--> 服务器) 与下载
- IMAP: 互联网邮件访问协议 [RFC 1730]
 - 更多功能 (更复杂)
 - 处理在服务器上存储的消息
- HTTP: GMail, Hotmail, Yahoo! Mail, 等等.



- 引言
- 核心问题: 应用需要自己的协议
- 传统应用
- 电子邮件(SMTP, MIME, IMAP)
- 万维网(HTTP)
- Web服务
- 多媒体应用
- 基础设施服务
- 覆盖网络
- 总结

Web 和 HTTP

首先介绍一些术语

- 网页由某些**对象**组成
- 对象可以是HTML文件，JPEG图片，Java Applet，音频文件，……
- 网页由包含以上某些对象的基本**HTML文件**构成
- 每个对象都由一个**URL**定位
- URL实例：

eic.hust.edu.cn/someDept/pic.gif

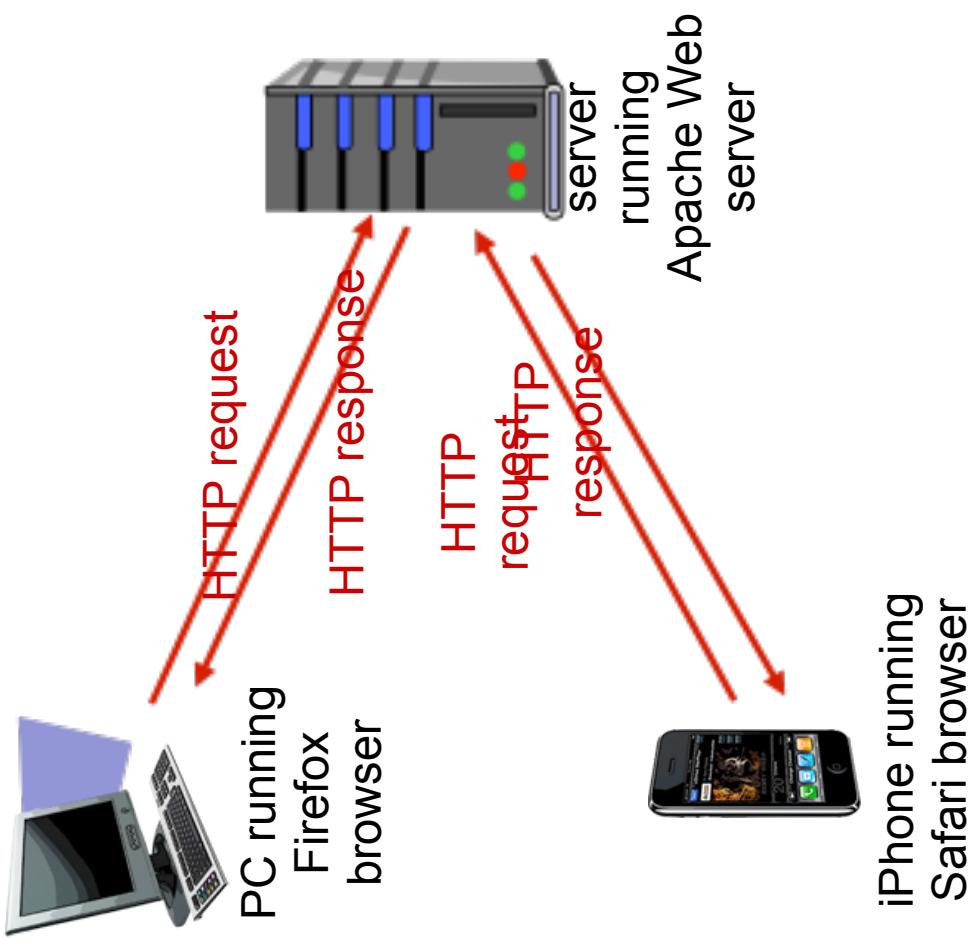
主机名 路径名



HTTP 概要

HTTP: 超文本传输协议

- 网络的应用层协议
- 客户机/服务器模型
 - **客户机:** 浏览器请求和接收消息，显示Web对象
 - **服务器:** Web服务器发送对象给浏览器，作为对请求的响应
-
-



HTTP 概要 (续)

使用 TCP 协议：

- 客户端发送连接请求（产生socket）给服务器的80端口
- 服务器接收从客户端发送来的TCP连接请求
- 浏览器（HTTP客户端）和Web服务器（HTTP服务端）之间交换HTTP消息（应用层协议消息）
- 关闭TCP连接

HTTP 是“无状态”协议

- 服务器不保存客户端之前请求的信息记录

注意

状态化协议是很复杂的！
过去的历史数据（状态）必须维持
如果服务器/客户端崩溃了，必
他们的状态可能不一致，必
须进行协调统一

HTTP 连接

非持久连接的 HTTP

- 每个TCP连接最多传输一个对象

持久连接的 HTTP

- 通过客户机和服务器之间的一个TCP连接，可以传输多个对象





非持久连接的 HTTP

假设用户输入 URL

www.someSchool.edu / someDepartment/home.index
(包含文本和 10 个图像的引用)

- 1a. HTTP 客户端向位于 www.someSchool.edu 端口号 80 的 HTTP 服务器（进程）发送 TCP 连接请求
- 1b. 位于 www.someSchool.edu 的 HTTP 服务器在 80 端口等待 TCP 连接。接受连接请求，通知客户端
2. HTTP 客户端向连接的 socket 发送 HTTP 请求信息（包含 URL）。该消息表示了客户端想要获得对象 someDepartment/home.index
3. HTTP 服务器接收到请求信息，然后构造一个响应信息，包含了被请求的对象文件，然后将其发送给它的 socket

时间





非持久连接的 HTTP (续)

4. HTTP 服务器关闭TCP连接.



5. HTTP 客户机接收到包含html文件的响应信息，显示html。分析html文件，找到某个引用了的jpeg对象

6. 重复1-5步，只到得到全部的10个jpeg对象

时间
↓

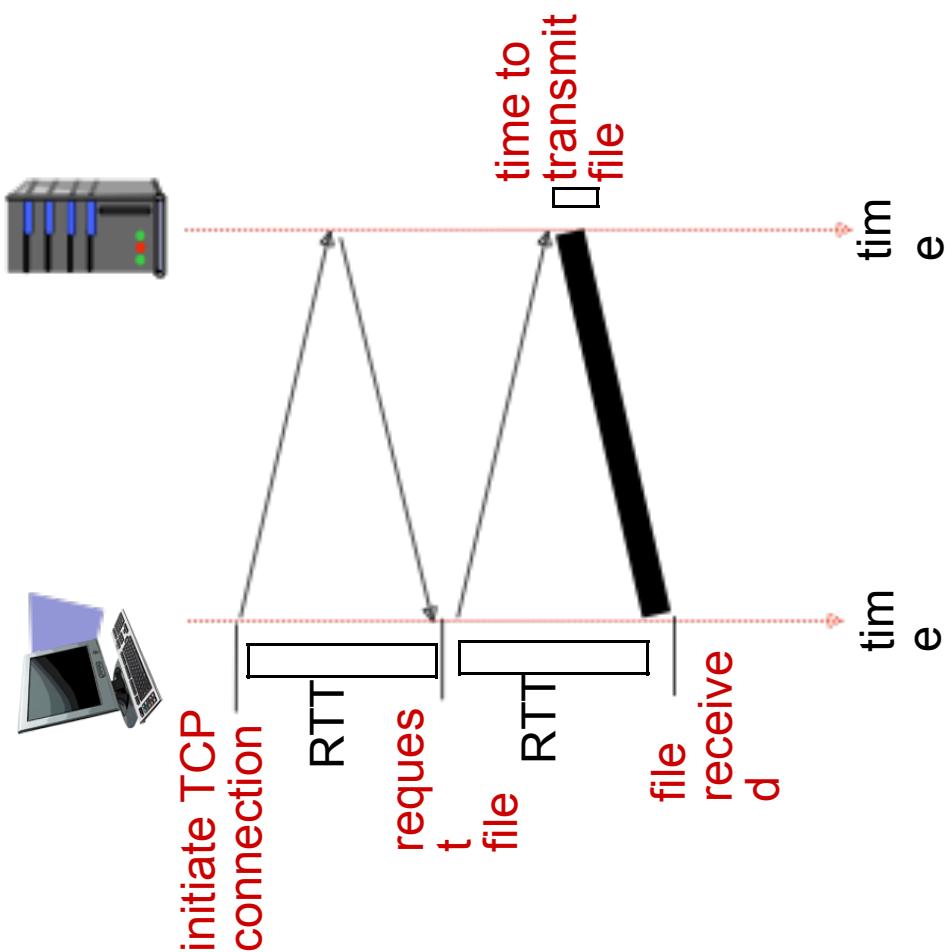


非持久连接的HTTP: 响应时间

RTT的定义: 一个数据包从客户端到服务器，再从服务器回到客户端的时间

响应时间:

- 一个 RTT时间来建立TCP连接
- 一个 RTT时间用来发送HTTP请求以及开始几个字节的HTTP响应
- 文件传输时间
- **总计响应时间 = 2RTT+文件传输时间**



持久连接的 HTTP

非持久连接的 HTTP 缺点：

- 每个对象的传输都需要2个RTT时间
- 每个TCP连接都要增加系统开销
- 浏览器通常会同时建立多个TCP连接来获得引用的对象
- 服务器在发送响应消息之后保持连接该客户机和服务器之间随后的HTTP消息就在这一个连接上传送
- 客户端每遇到一个引用的对象就向服务器发送请求
- 所有引用的对象在一个RTT时间里就可以传送完毕

持久连接的HTTP



HTTP 请求消息

- 两类HTTP消息：请求，响应
- HTTP请求消息：
 - ASCII (易读的格式)

请求行
(GET, POST,
HEAD命令)

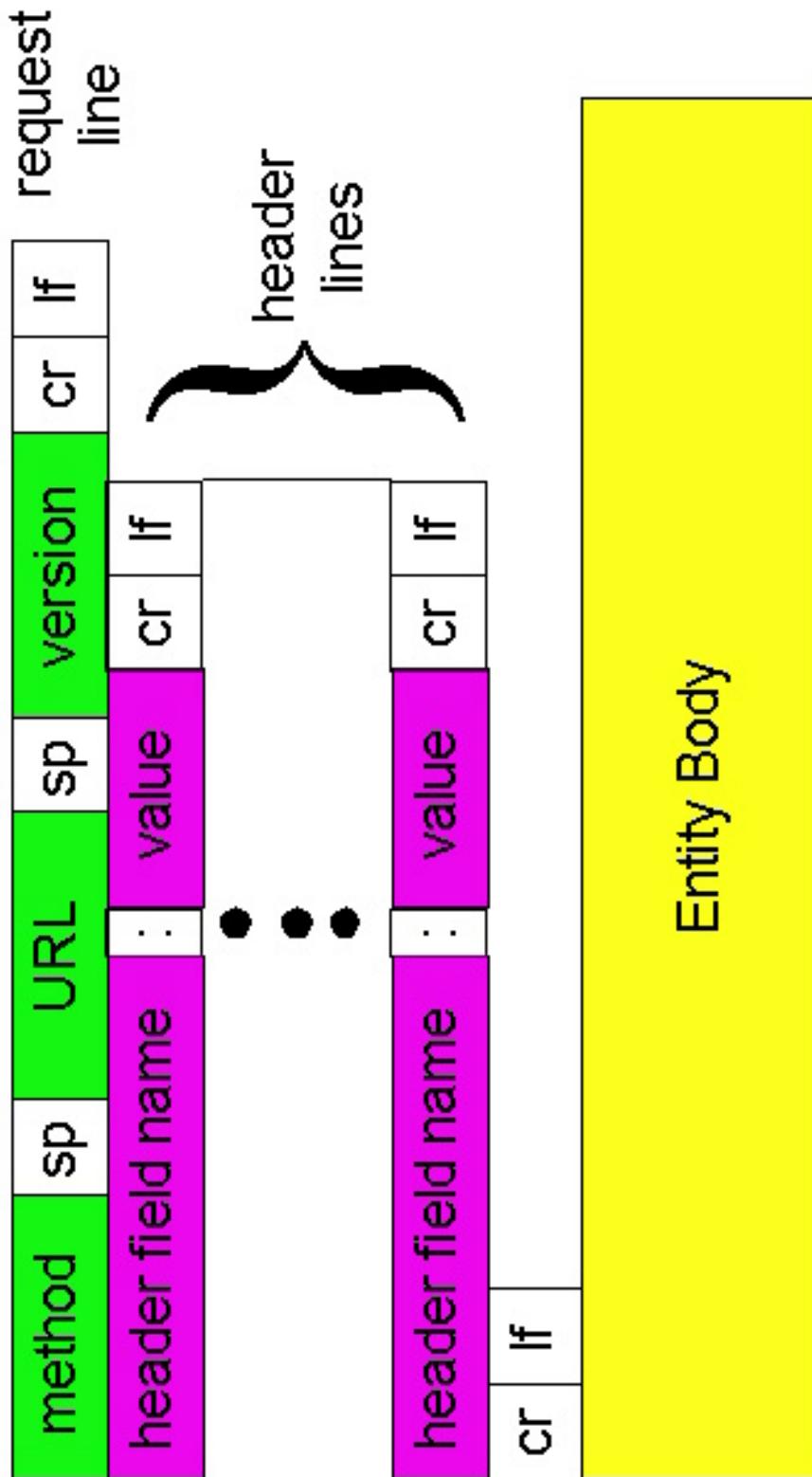
头信息行

```
GET /somedir/page.html
HTTP/1.1
Host: www.someschool.edu
User-Agent: Mozilla/4.0
Connection: close
Accept-Language: fr
```

回车，换行
表明该信息的结尾
(额外的回车，换行)



HTTP请求消息：一般格式



上传文本框的输入

Post方法:

- 网页通常包含一些文本框的输入内容
- 输入通过entity body上传到服务器

URL方法:

- 使用GET method
- 输入通过请求行中的URL域值上传:

www.somesite.com/animalsearch?
monkeys&banana



方法类型

HTTP/1.0

- GET
 - POST
 - HEAD
- 请求服务器不要将请求的对象放在响应信息里

HTTP/1.1

- GET, POST, HEAD
- PUT
 - 在指定的URL路径上传文件
- DELETE
 - 在指定的URL路径删除文件



HTTP 响应信息



状态行
(协议代码
协议状态)

HTTP/1.1 200 OK

Connection close

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

Server: Apache/1.3.0 (Unix)

头信息
行

Last-Modified: Mon, 22 Jun 1998 ...

Content-Length: 6821

Content-Type: text/html

data data data data data ...

数据，例如
请求的
HTML文件



HTTP 响应状态代码

服务器发送给客户端响应信息的第一行
一些代码示例：

200 OK

- 请求成功，请求对象在该信息的尾部
- **301 Moved Permanently**
 - 请求对象有移动，新位置将在该信息后面指出(位置:)
- **400 Bad Request**
 - 服务器不明白请求信息
- **404 Not Found**
 - 服务器没有找到请求的文档
- **505 HTTP Version Not Supported**

自己尝试编写 HTTP(客户端)



1. 登录你最喜爱的Web服务器：

telnet eic.hust.edu.cn 80 在eic.hust.edu.cn 的80端口（默认的HTTP服务器端口）打开TCP连接。
发送给eic.hust.edu.cn 80端口的所有信息都将显示出来

2. 键入一个 GET 的 HTTP 请求：

**GET /index.html HTTP/1.1
Host: eic.hust.edu.cn**
输入这些之后（按两次回车）
你将给HTTP服务器发送最小的（但是完整的）GET请求

3. 查看HTTP服务器的响应信息！

用户-服务器状态: cookies



许多主流的Web网站都使用Cookies

四个构成要素:

- 1) HTTP响应消息的cookie头信息行
- 2) 在HTTP请求信息中的cookie头信息行
- 3) Cookie文件保存在用户的主机上, 由用户的浏览器进行管理
- 4) Web网站的后端数据库

示例:

Susan经常使用个人电脑登陆互

联网

第一次访问某个电子商务网站

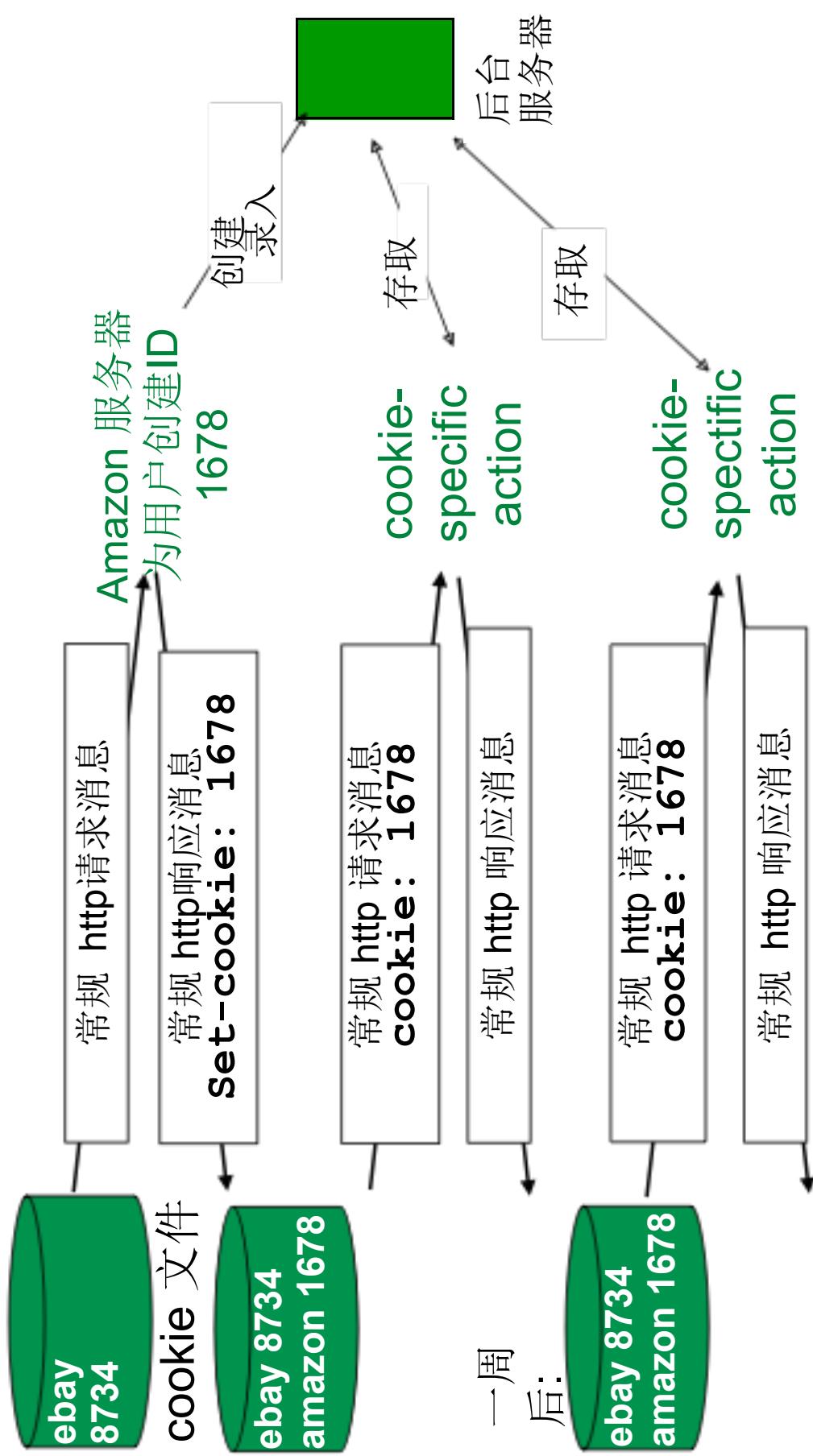
当初始的HTTP请求到达网站服务器后, 服务器创建了:

唯一的ID号
在后台的数据库中录入该ID

Cookies: 保持“状态”(续.)

客户机

服务器



Cookies (续)



Cookies能做什么：

- 授权
- 购物车
- 推荐
- 用户活动状态
(Web e-mail)

Cookies与隐私：

Cookies允许网站得到很多你的个人消息，你将向网站提供你的名字和e-mail

注意

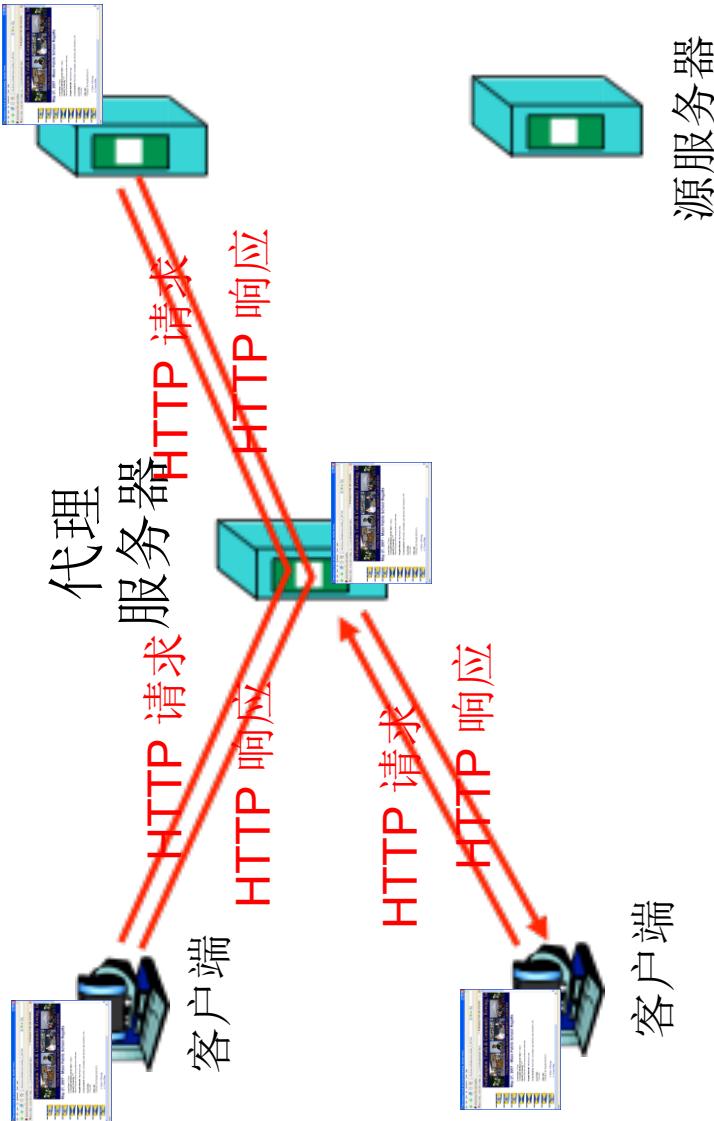
如何保持“状态”：

- 协议端点：在多个会话中在发送端/接收端保持状态
- cookies: http消息中携带了状态信息

Web 缓存(代理服务器)

目标: 不涉及源服务器就能满足客户端的请求

- 用户设置浏览器: 通过代理服务器向代理服务器发送所有的HTTP请求消息
- 如果请求的对象在代理服务器中: 代理服务器将该对象给浏览器返回
- 反之, 代理服务器向源服务器请求对象, 然后将结果发给客户端



关于Web代理



- 代理服务器同时扮演了服务器和客户机的角色
- 通常，代理服务器由ISP安置(学校,公司,住宅区的ISP)

为什么要使用Web代理?

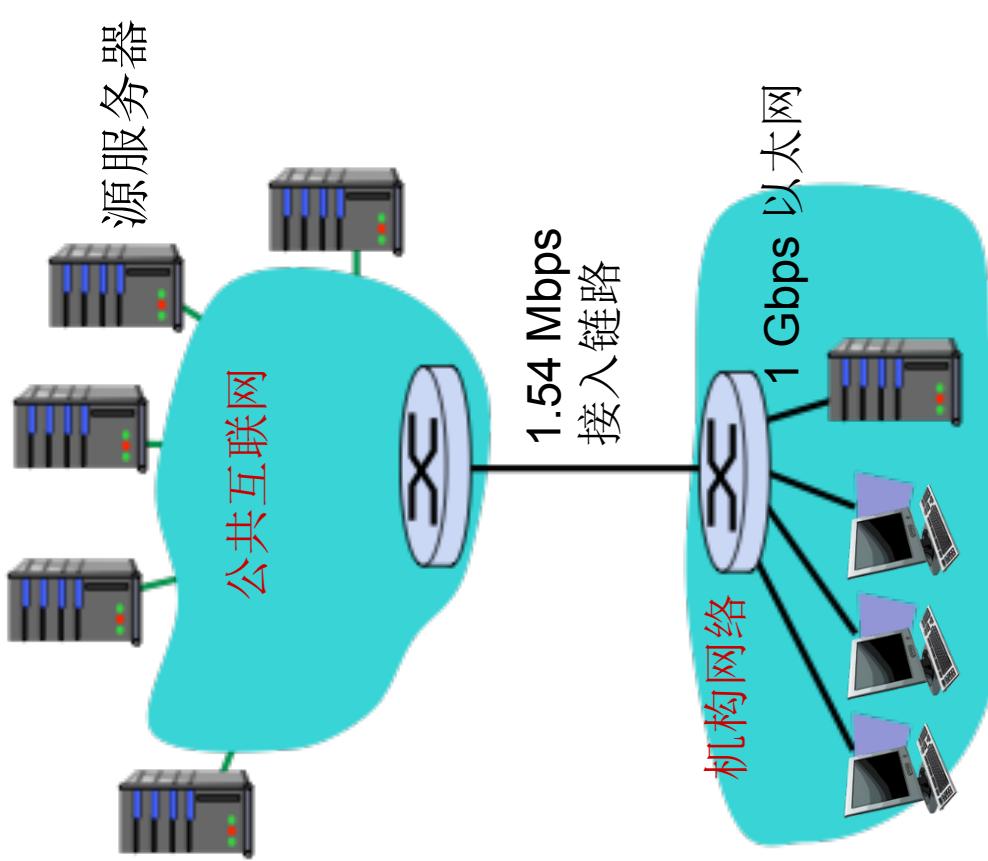
- 减少客户端请求的响应时间
- 减少某个机构接入链路的吞吐量。

- Internet里有很多代理服务器：使性能较差的内容发布者能够有效的内容发布(P2P文件共享技术也能做到这一点)

缓存案例

假设：

- 文件对象的平均大小: 100K bits
- 浏览器平均数据率: 1.50 Mbps
- 浏览器平均数路到源服务器的往返时延: 2 sec
- 接入网络链路带宽: 1.54 Mbps



结果：

- 局域网利用率: 0.15% = 99%
接人链路 = 互联网时延 + 接入时延 + 局域网时延
- = 2 sec + minutes + usecs

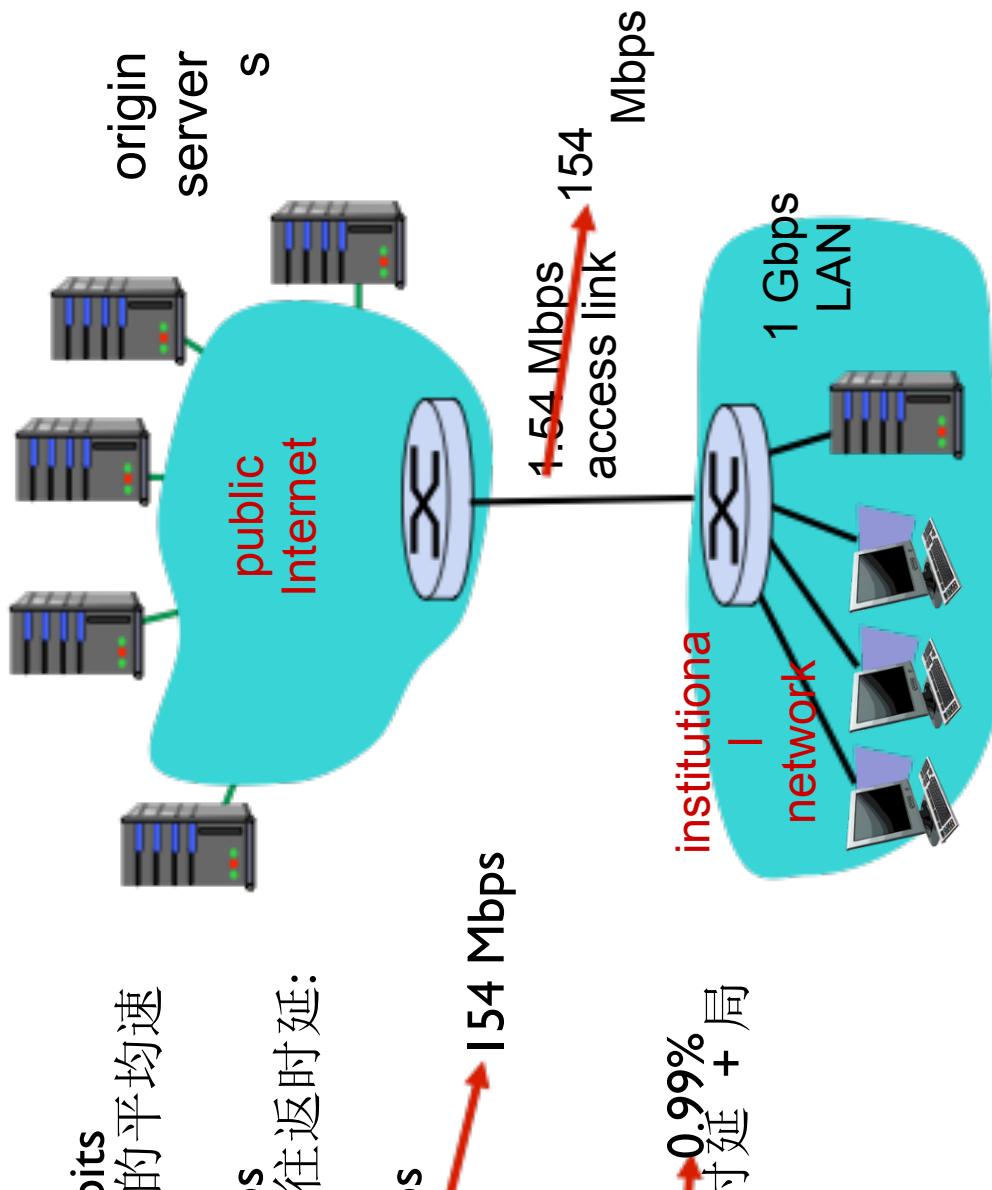
问题!

缓存案例：更快的接入链路



假设：

- 文件对象的平均大小: 100K bits
- 浏览器平均发送请求的平均速率: **15/sec**
- 浏览器平均数据率: **1.50 Mbps**
- 该机构路由器到源服务器的往返时延: **2 sec**
- 接入网络链路带宽: **1.54 Mbps**



结果：

- 局域网利用率: **0.15%**
- 接入链路利用率 = **99%** → **0.99% 总时延 = 互联网时延 + 接入时延 + 局域网时延**
- = **2 sec + minutes + usecs**
- msecs

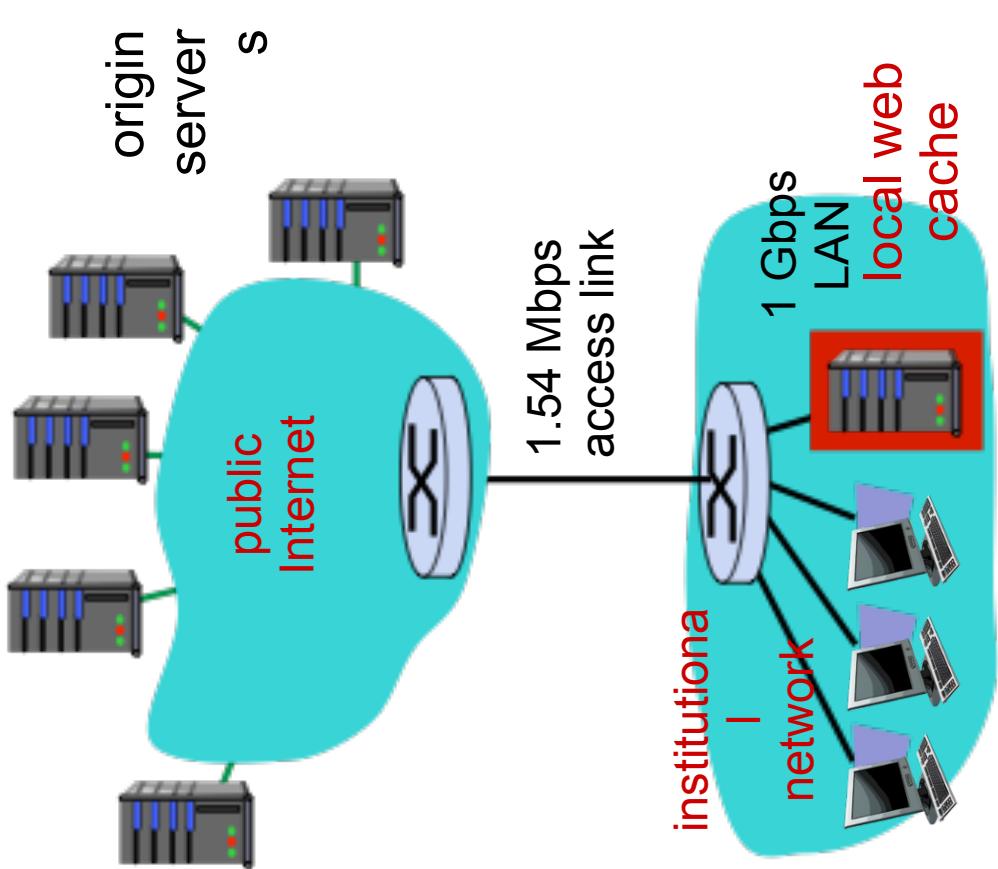
代价: 增加外网链路带宽(**昂贵!**)

缓存案例：增加本地缓存



假设：

- 文件对象的平均大小: 100K bits
- 浏览器平均数据率: 1.50 Mbps
- 该机构路由器到源服务器的往返时延: 2 sec
- 接入网络链路带宽: 1.54 Mbps



结果：

- 局域网利用率: 0.15%
- 接入链路利用率 - ?
- 总时延 - ?

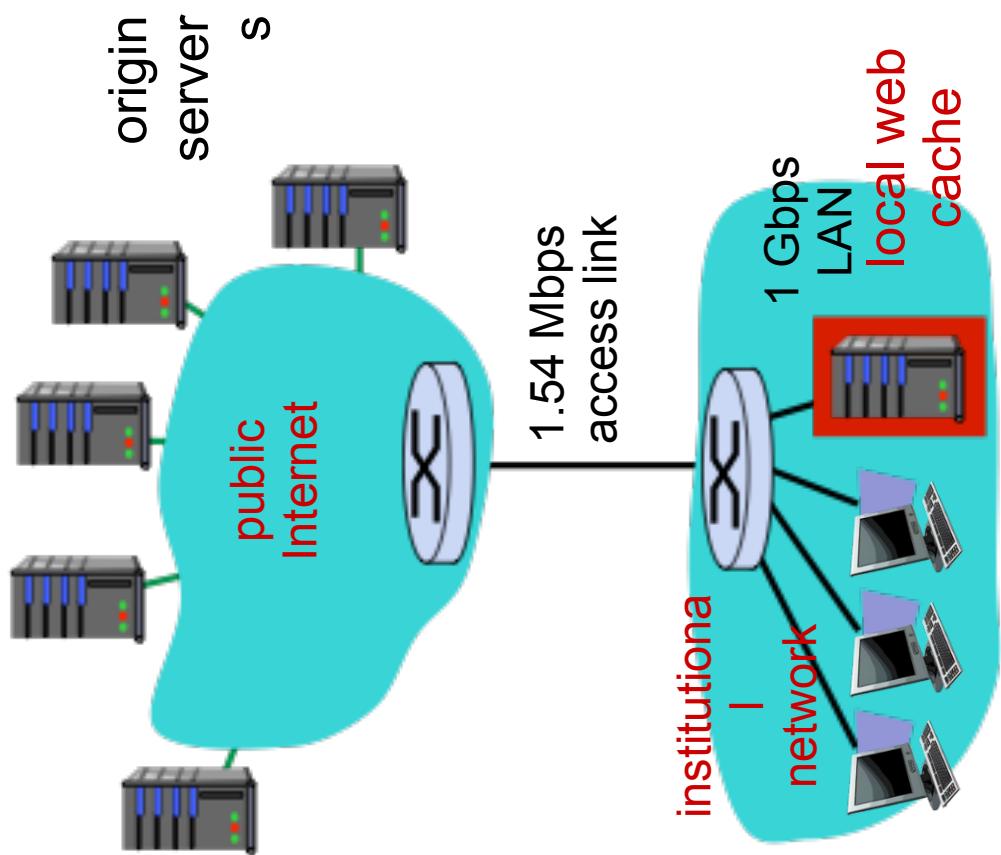
如何计算链路利用率和时延?

代价: web缓存(便宜!)

缓存案例：增加本地缓存

考虑本地缓存，计算接入链路利用率：

- 假设缓存命中率为0.4
- 40% 的请求可以在缓存服务器找到备份，
60%的请求仍然发到原服务器
- 接入链路利用率为 $0.9 / 1.54 = .58$
- 通过接入链路的数据率 = $0.6 * 1.50 \text{ Mbps} = .9$
- Mbps 利用率 = $0.9 / 1.54 = .58$
- 总时延 = $0.6 * (\text{从原服务器下载的时延}) + 0.4 * (\text{从缓存服务器下载的时延})$
 $= 0.6(2.0) + 0.4(\sim msecs)$
 $= \sim 1.2 \text{ secs}$
- 时延小于使用 154 Mbps 链路的时延
(而且更便宜!)



有条件的 GET



目标: 如果代理服务器已经缓存了更新后的版本，则源服务器不发送任何对象文件

代理服务器：在HTTP请求中指明已缓存副本的日期

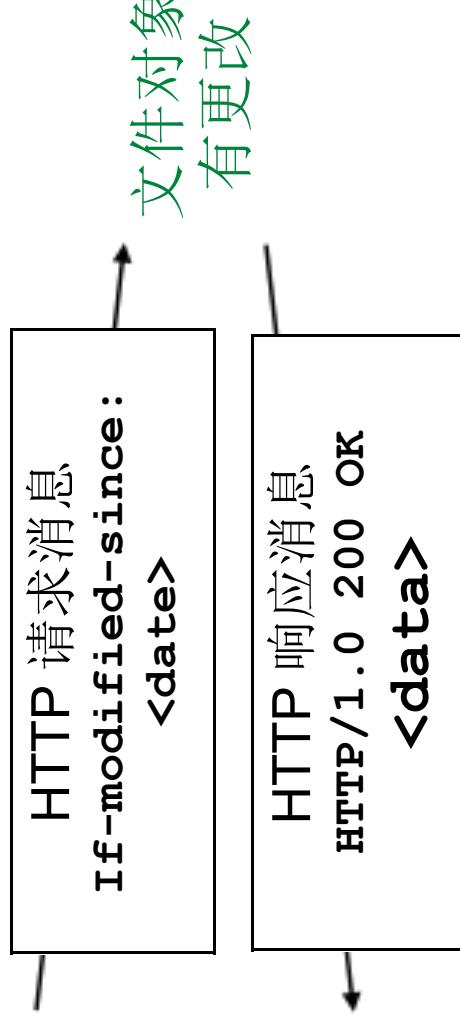
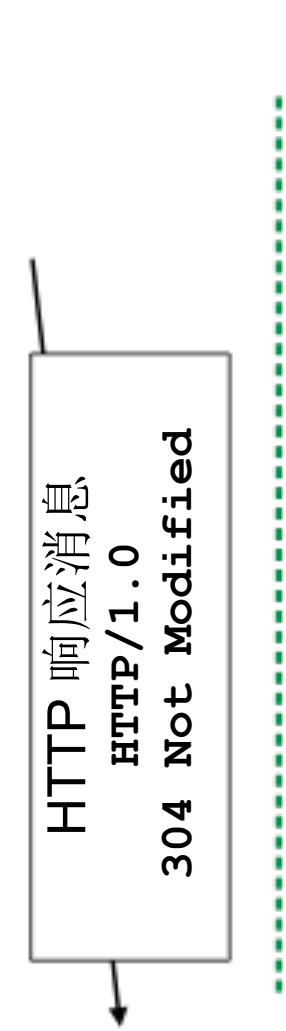
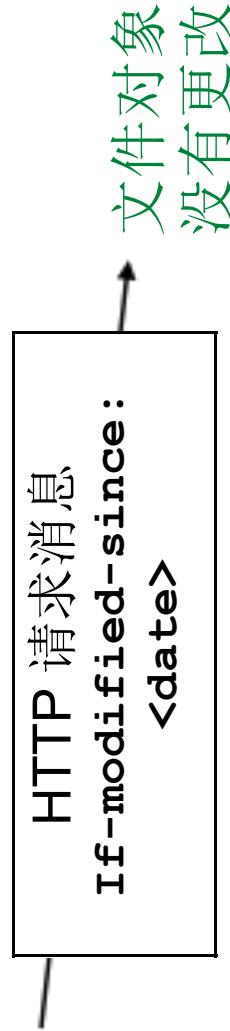
If-modified-since: <date>

服务器：如果缓存的副本已经更新，则响应消息中不发送对象文件：

HTTP/1.0 304 Not Modified

cache

server



讨论：SMTP vs HTTP



- SMTP 协议使用持续的连接
- SMTP 要求消息（包括消息头和消息体）都是7位ASCII码
 - SMTP服务器使用CRLF。CRLF（回车换行）来表示消息的结束
- 都有ASCII码的命令和响应交互，状态码
 - HTTP: 拉取信息(pull)
 - SMTP: 推送信息(push)
- HTTP: 每个对象文件都封装在各自的应用消息中
- SMTP: 多个对象文件在消息的多个部分发送（multiple objects sent in multipart msg）



- 引言
- 核心问题：应用需要自己的协议
- 传统应用
- 电子邮件(SMTP, MIME, IMAP)
- 万维网(HTTP)
Web服务
- 多媒体应用
- 基础设施服务
- 覆盖网络
- 总结

Web服务



- 应用与应用的直接通信主要来源于商业应用场景
- 历史上，企业、商业机构或者其他组织之间需要通过人工交互，例如填写订购单或者查询某个产品是否有货
- 即使在同一一个企业，软件系统也可能无法直接交互
- 因为这些系统是独立开发，需要人工交互
- 越来越多的人工交互正被应用与应用之间的交互所代替
- 例如，A企业的订购应用可以直接向B企业的订单系统发送一个订购消息，马上收到回复，是否有货
- 如果B企业无货，A的订购应用可以马上向另一个供货商发出订单，或者向一群供货商发送竞购信息
- Web服务类别： SOAP vs REST

Web 服务架构: SOAP

- 为不同的网络应用构建定制协议
- SOAP提供了协议规范框架，并根据协议可通过软件工具自动实现协议，而且模块化的部分规范也可以跨协议重用。



Web 服务架构: REST



- REST将不同的Web服务视为Web资源，可由URI定义，并可通过HTTP协议交互
- REST是一种Web架构
- REST优点包括稳定性和网络可扩展性

提纲



- 引言
- 核心问题: 应用需要自己的协议
- 传统应用
- 多媒体应用
- 多媒体应用的资源分配
- 基础设施服务
- 覆盖网络
- 总结

服务质量保证

- 资源预留
 - 呼叫建立, 信令 (RSVP)
 - 流量, 服务质量声明
 - 接入控制
 - 面向服务质量的调度
(e.g., WFQ)
-



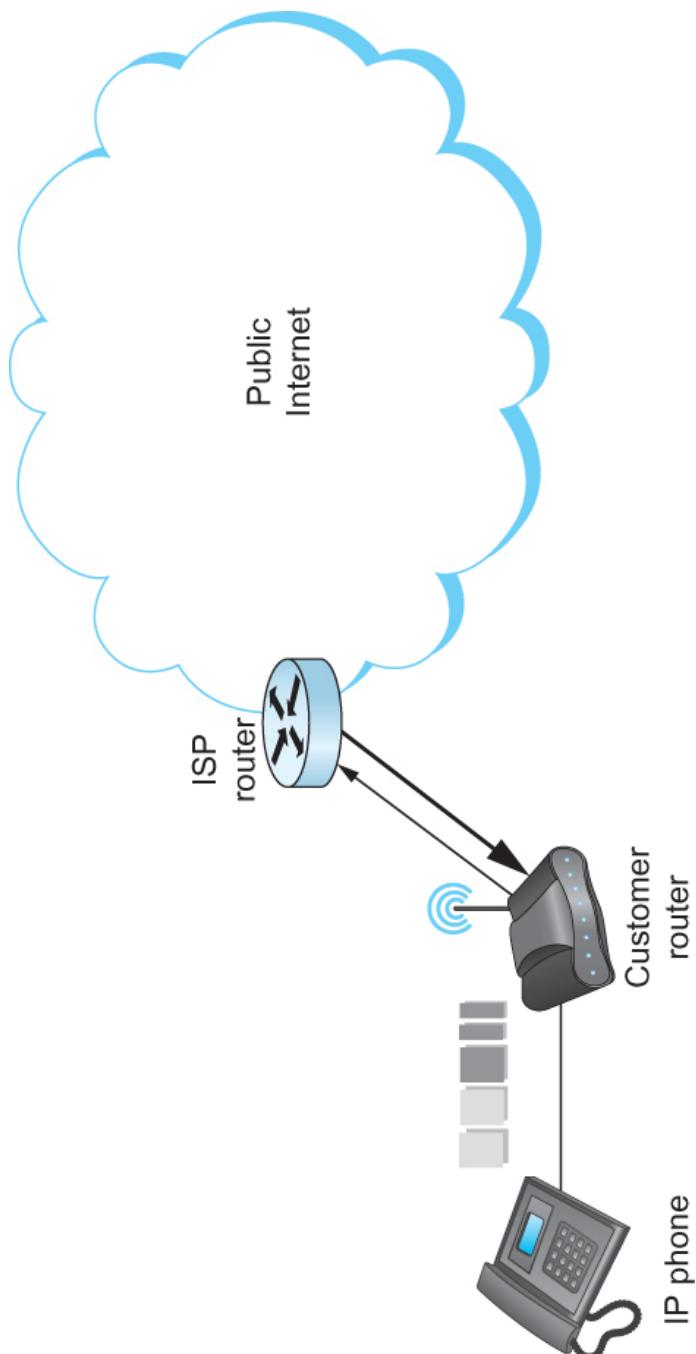


多媒体应用的资源分配

- 会话控制协议(例如 SIP和H.323)可用于初始化和控制多媒体应用的通信，而RTP用于传输应用的数据
- 多媒体应用需要网络进行合理的资源分配，从而保障其服务质量
- 区分服务可分为应用提供比较简单但可扩展的资源分配
- 一个多媒体应用可以通过设置IP报头的DSCP(区分服务代码指针)，使媒体数据分组获得合适的服务质量

案例：区分服务资源分配

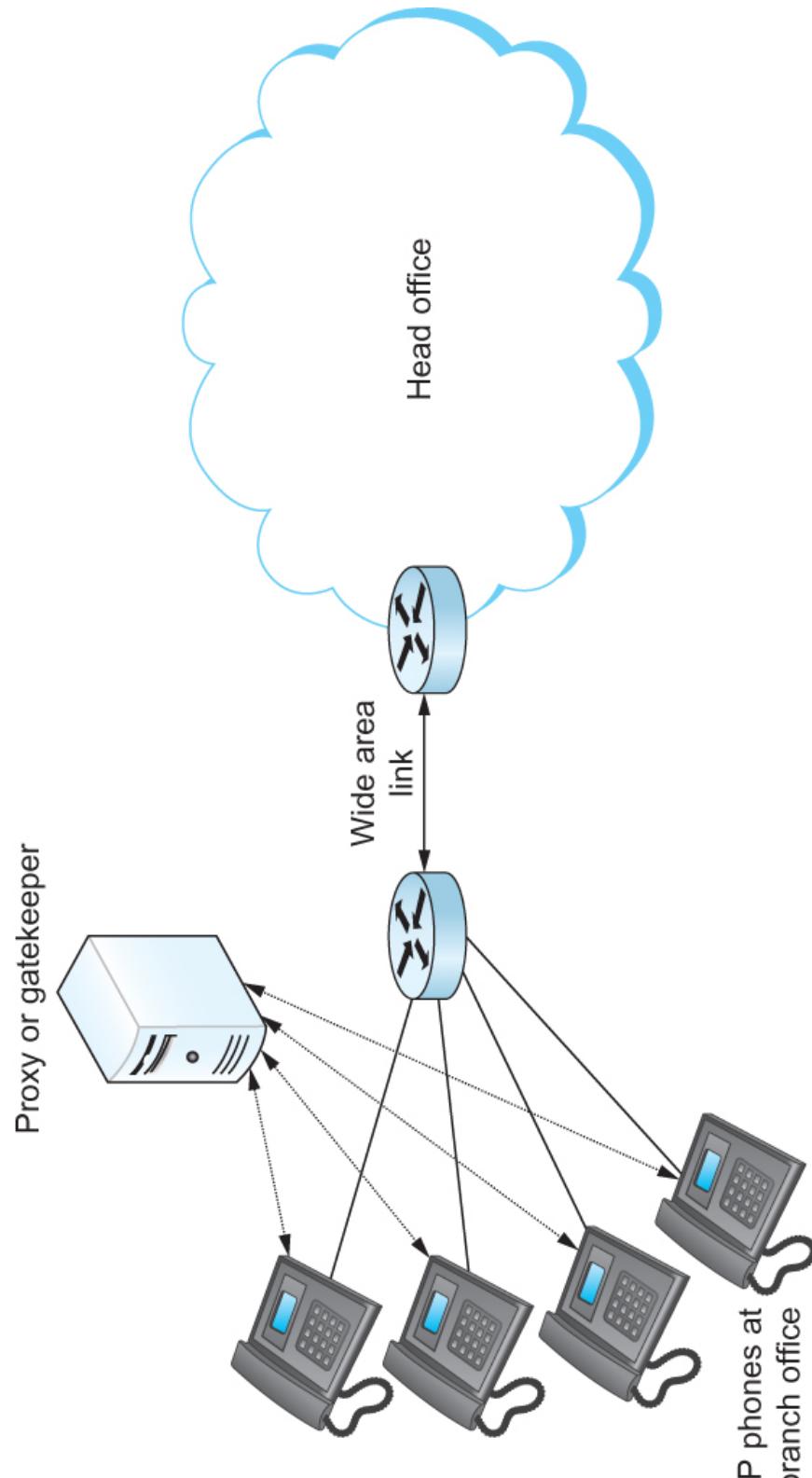
- 网络电话应用使用区分服务，提高服务质量
- 区分服务排队仅用于从客户路由器到ISP的上行链路





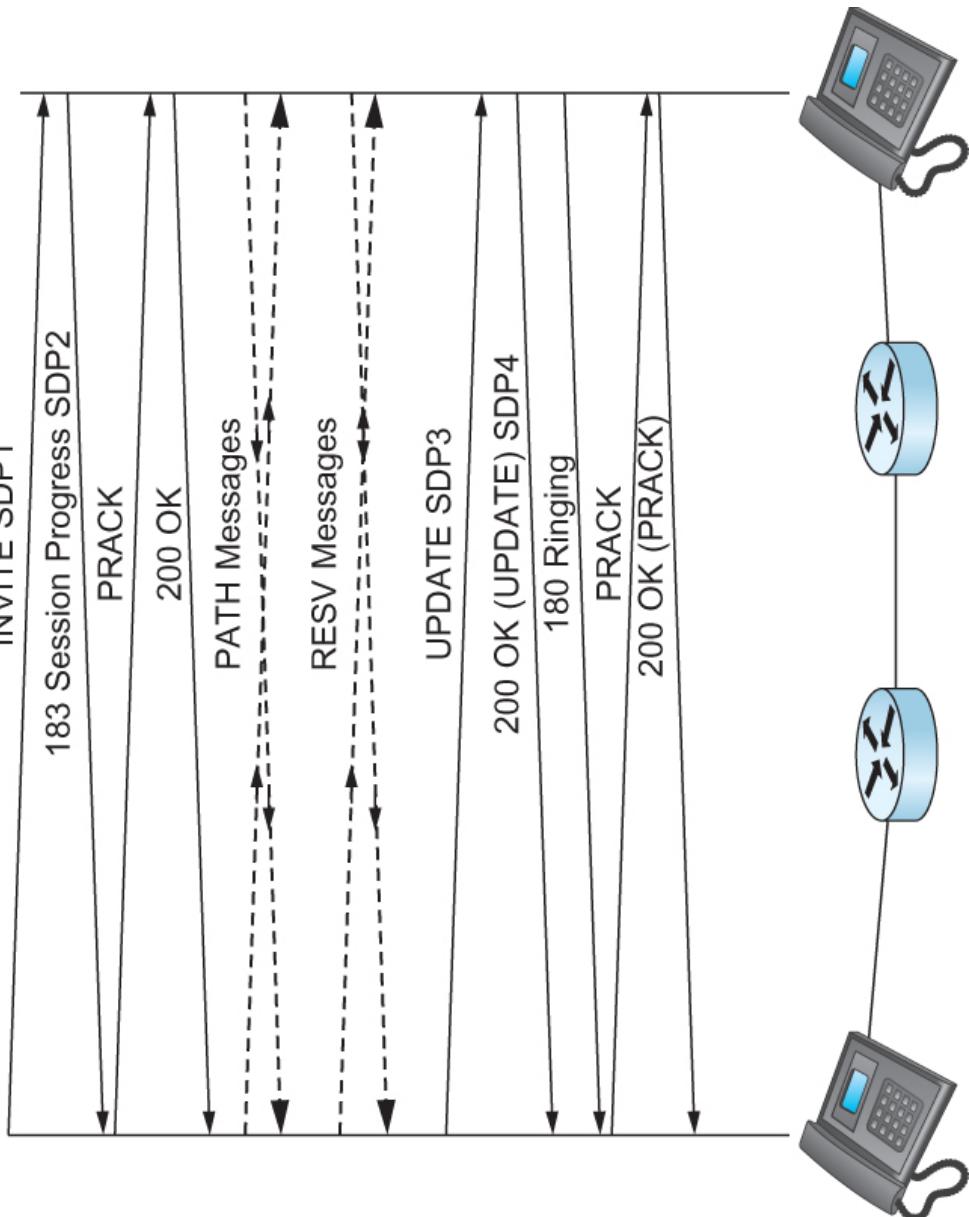
案例：通过准入控制进行资源分配

- 会话控制实现准入控制



案例：准入控制协议与会话控制协议交互

- SIP协议用于会话控制
- RSVP用于准入控制决策
- **会话控制和资源分配任务的交叉**



提纲

- 引言
- 核心问题：应用需要自己的协议
- 传统应用
- 多媒体应用
- 基础设施服务
 - 域名解析(DNS)
 - 网络管理(SNMP)
- 覆盖网络
- 总结



域名系统: DNS

每个人都有很多识别身份的代号:

- 身份证号, 名字, 护照号码
 - **互联网的主机, 路由器:**
 - IP地址(32位) - 用于数据包的寻址
 - “名称”, 例如, www.yahoo.com - 方便人们记忆
- 问题:** IP地址与主机名称之间
的映射?

域名系统:

- 许多**域名服务器**有层次的组成了一个**分布式数据库**
- 应用层协议的主机**, 路由器, 域名服务器之间互相通信来解析域名, (地址与名称间的互相转换)
- 注意: 互联网的核心功能, 都由应用层协议实现
- 复杂性置于网络“边缘”



DNS

DNS 服务

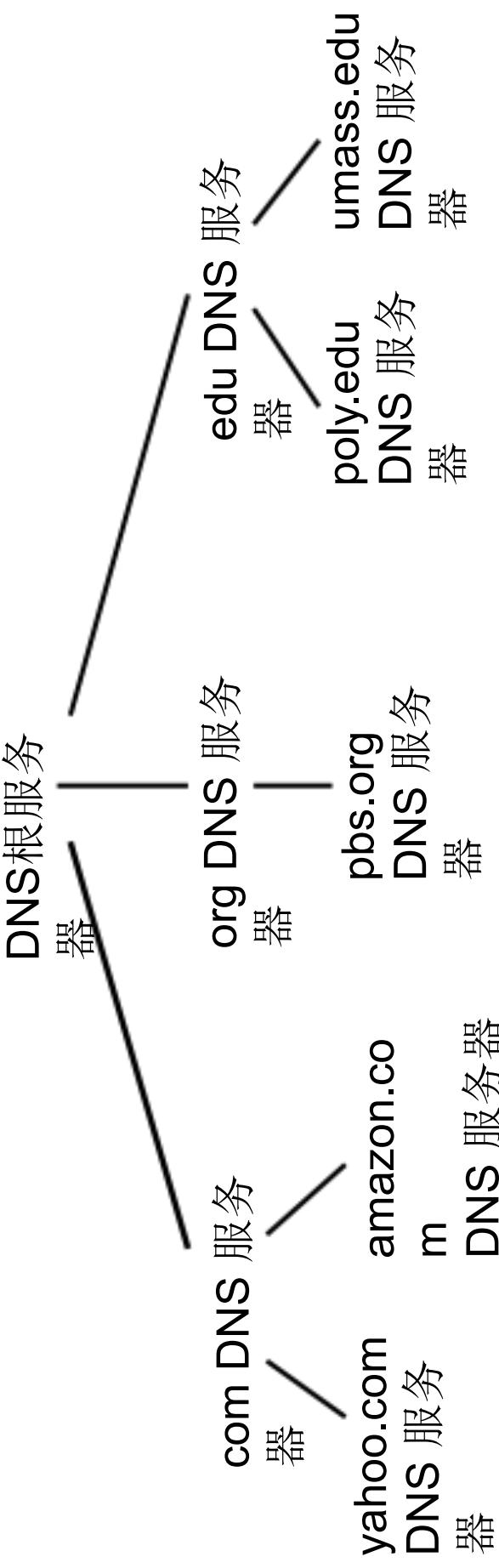
- 主机名到IP地址的映射
 - 主机别名
 - 正规主机名, 别名
 - 邮件服务器别名
 - 负载分配
 - 单点故障
 - 通信量
 - 中心数据库过远
 - 维护
- Web 服务器的镜像：一系列的IP地址对应一个正规的主机名
难以扩展！

为何不用集中式的DNS?





分布式, 分层数据库



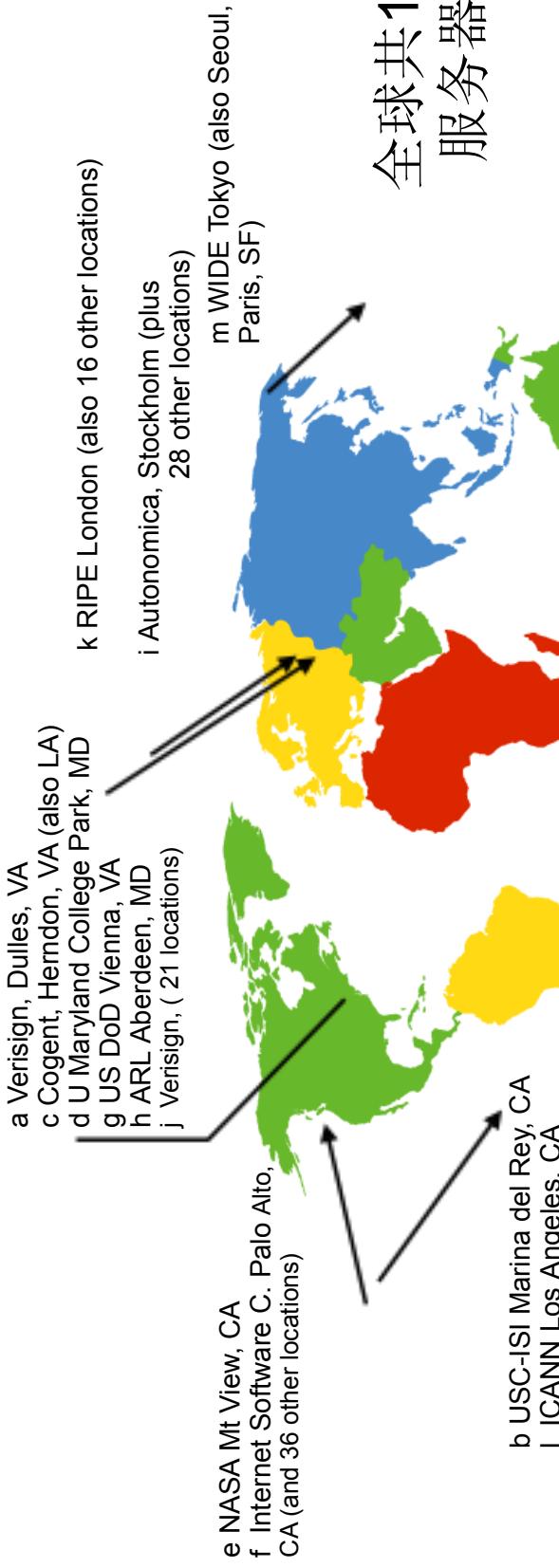
用户需要www.amazon.com的IP地址; 1st approx:

- 用户向某个根服务器查询com DNS服务器
- 用户向com DNS服务器查询amazon.com DNS服务器
- 用户向amazon.com DNS服务器查询www.amazon.com的IP地址

DNS: 根域名服务器



- 由本地域名服务器联系，如果其不能完成域名解析
- 根域名服务器：
 - 如果域名映射未知，则联系权威的域名服务器获得映射
 - 将该映射返回给本地域名服务器



顶级域名(TLD)与权威服务器



- **顶级域名(TLD)服务器:**
 - 负责com, org, net, edu, 等等的域名，以及所有的顶级国家域名，uk, fr, ca, jp.
 - Network Solutions公司为顶级域名com维护服务器
 - Educause公司为顶级域名edu提供维护服务
- **权威DNS服务器:**
 - 某机构的DNS服务器，为该机构的服务器（例如，Web, Mail）提供权威的域名到IP地址的映射。
 - 可以由某机构或者服务提供商维护

本地域名服务器



- 不直接的属于某个层次结构
- 每个ISP(住宅区ISP, 公司, 学校)都有一个本地域名服务器.
- 也称为“默认识名服务器”
- 当主机产生DNS询问时，该询问被送到它本地的DNS服务器那里
- 类似于一个代理服务器，它将询问消息转发给有层次结构的域名服务器那里

DNS 域名解析示例



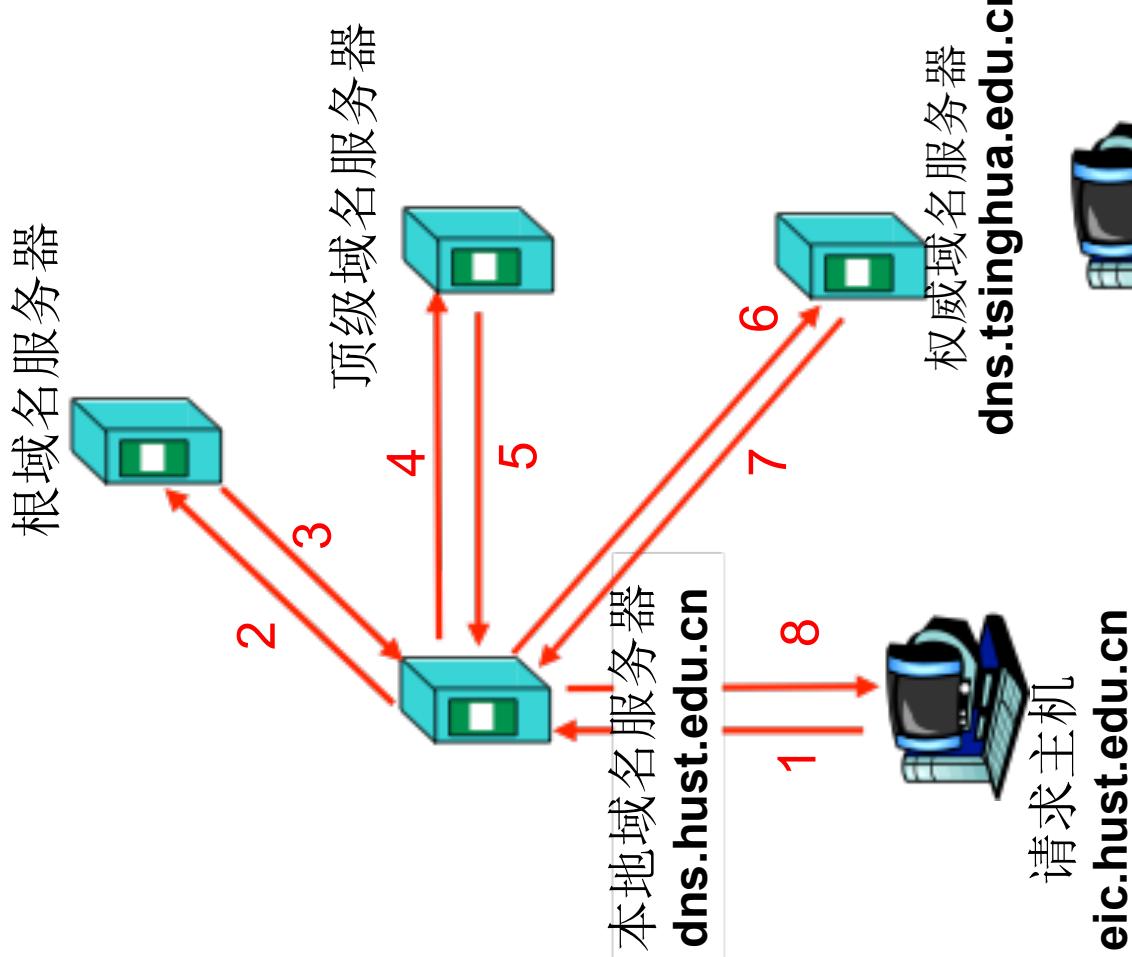
- eic.hust.edu.cn 主机想
要查询

ee.tsinghua.edu.cn 的 IP
地址

迭代查询:

被询问的服务器将可询
问服务器的名称作为答
复

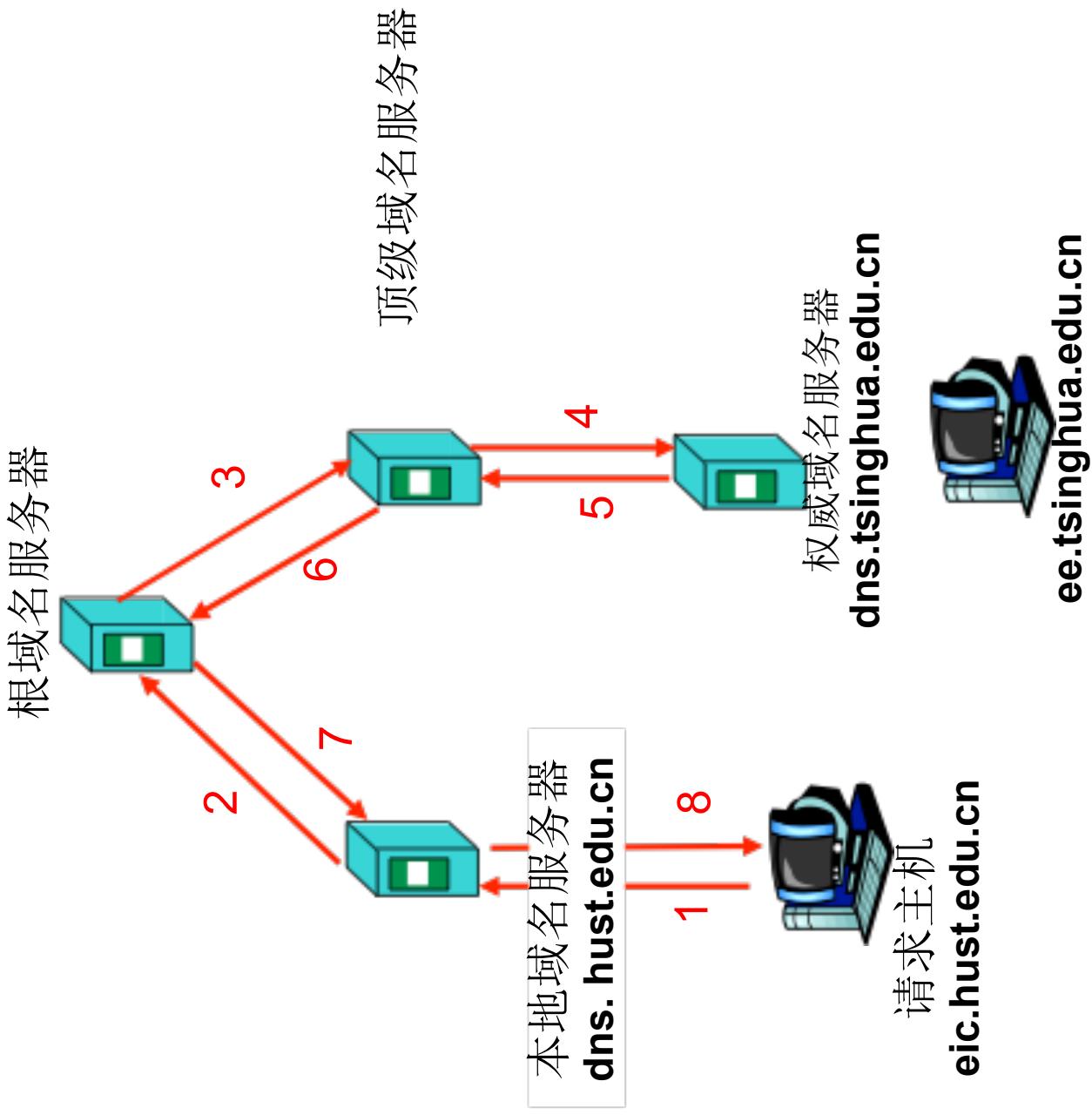
“我不知道这个域名，但
是可以询问这个服务器”



ee.tsinghua.edu.cn

DNS 域名解析示例

- r 递归查询:
- r 将查询域名的责任交给被联系的域名服务器，负载过大？



DNS: 缓存和更新记录



- 一旦(任何)域名服务器得到了某个映射, 它将缓存这个映射
 - 缓存的映射将在某一段时间之后过期(消失)
 - 顶级域名服务器通常都被本地域名服务器缓存
 - 因此根域名服务器不是经常被访问
- 更新/通告机制正在由IETF设计
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

DNS 记录

DNS: 分布式数据库存储资源记录 (RR)

RR 格式: (**name**, **value**, **type**, **ttl**)

- r Type=A
 - **name** 是主机名
 - **value** 是IP地址
- r Type=CNAME
 - **name** 是某些正规（真正）的域名的别名
- www.ibm.com的正规名为
servereast.backup2.ibm.com
- r Type=NS
 - **name** 是域名 (例如foo.com)
 - **value** 是该域的权威域名服务器名称
- r Type=MX
 - **Value** 与**name**有关的邮件服务器的名称



DNS 协议，消息



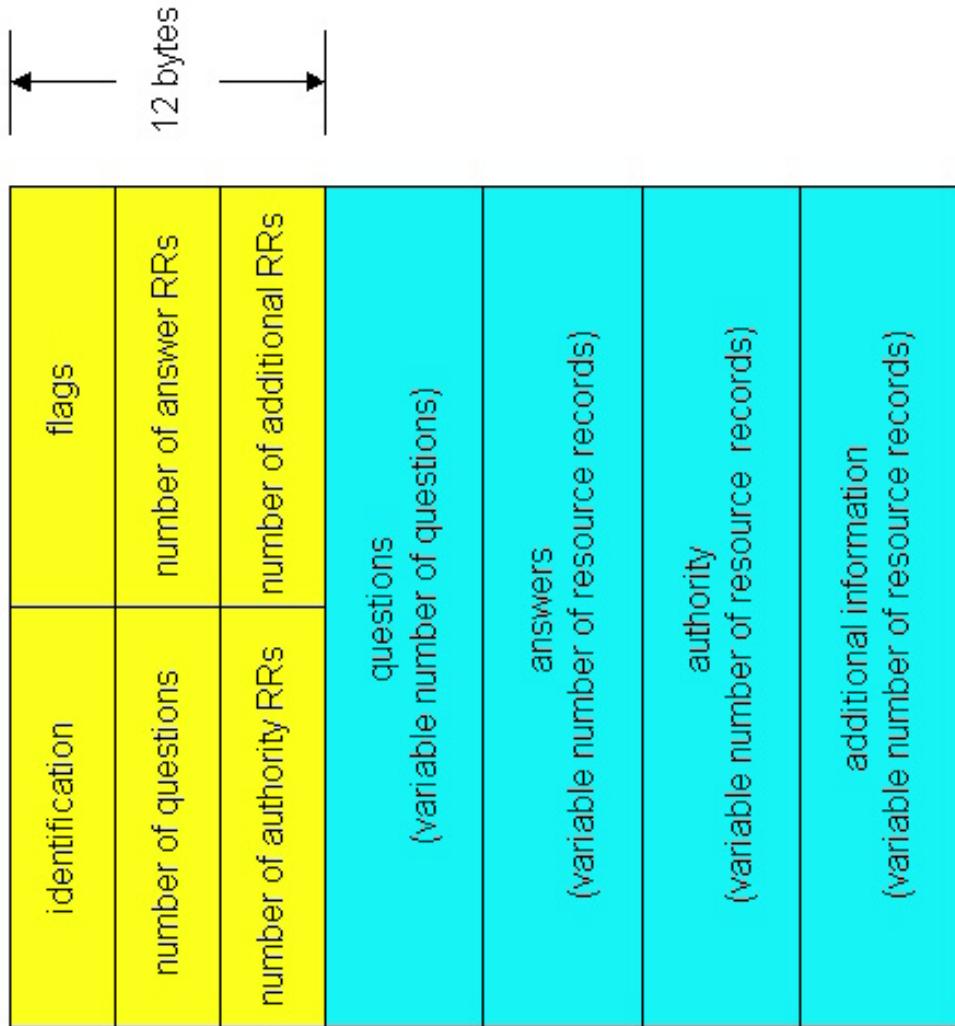
DNS 协议：查询与回复消息，都使用相同的消息格式

消息头部

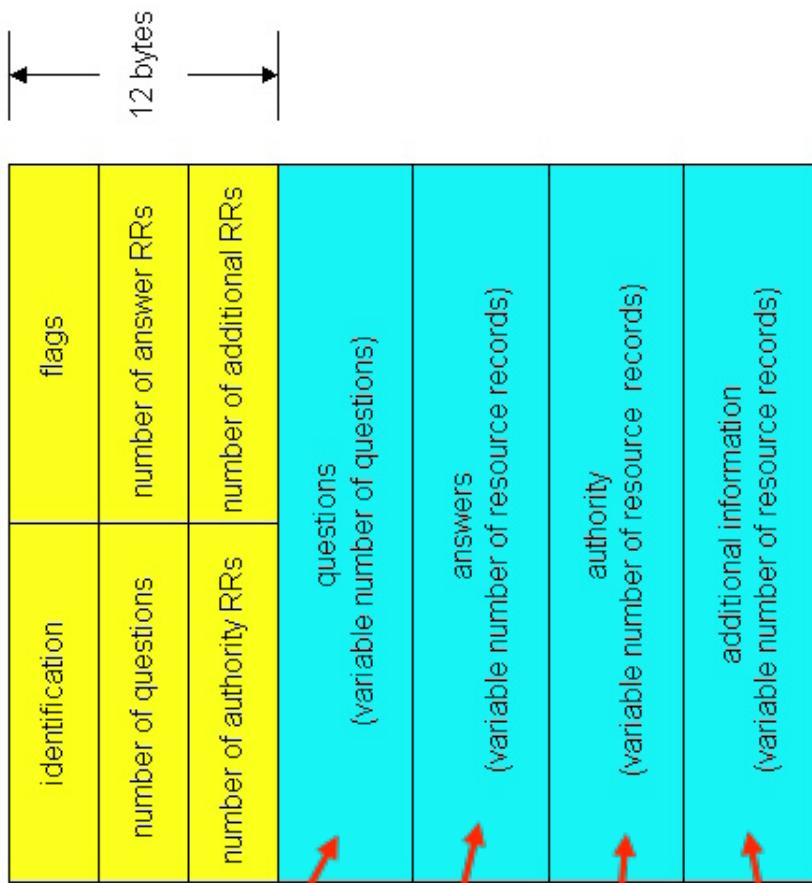
识别位：用于查询的16位编号，回复该查询的消息使用相同的编号

标志位：

- 查询或者回递归
- 是否要递归
- 是否权威
- 回复是否权威



DNS 协议，消息



域名,为查询类型的字段

回复查询消息的
资源记录

权威服务器的记录

可能会用到的
额外的“有用的消息”

在DNS中插入记录



- 示例：添加“Network Utopia”的DNS信息
- 在DNS注册公司(例如，Network Solutions) 注册域名 networkutopia.com
 - 提供（主要和次要的）权威域名服务器的域名，IP地址
 - 注册公司在com顶级域名服务器中插入两条资源记录：

(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
- 为www.networkutopia.com创建权威域名服务器的A类型记录；为networkutopia.com 创建MX类型记录
- 人们如何得到你的网站的IP地址？

提纲

- 引言
- 核心问题：应用需要自己的协议
- 传统应用
- 多媒体应用
- 基础设施服务
 - 域名解析(DNS)
 - 网络管理(SNMP)
- 覆盖网络
- 总结



什么是网络管理



- **自治系统 (aka “网络”):** 数以千计及以上的硬件/软件组件
- 其他复杂系统需要监测和控制:
 - 喷气式飞机
 - 核电站
 - 其他?

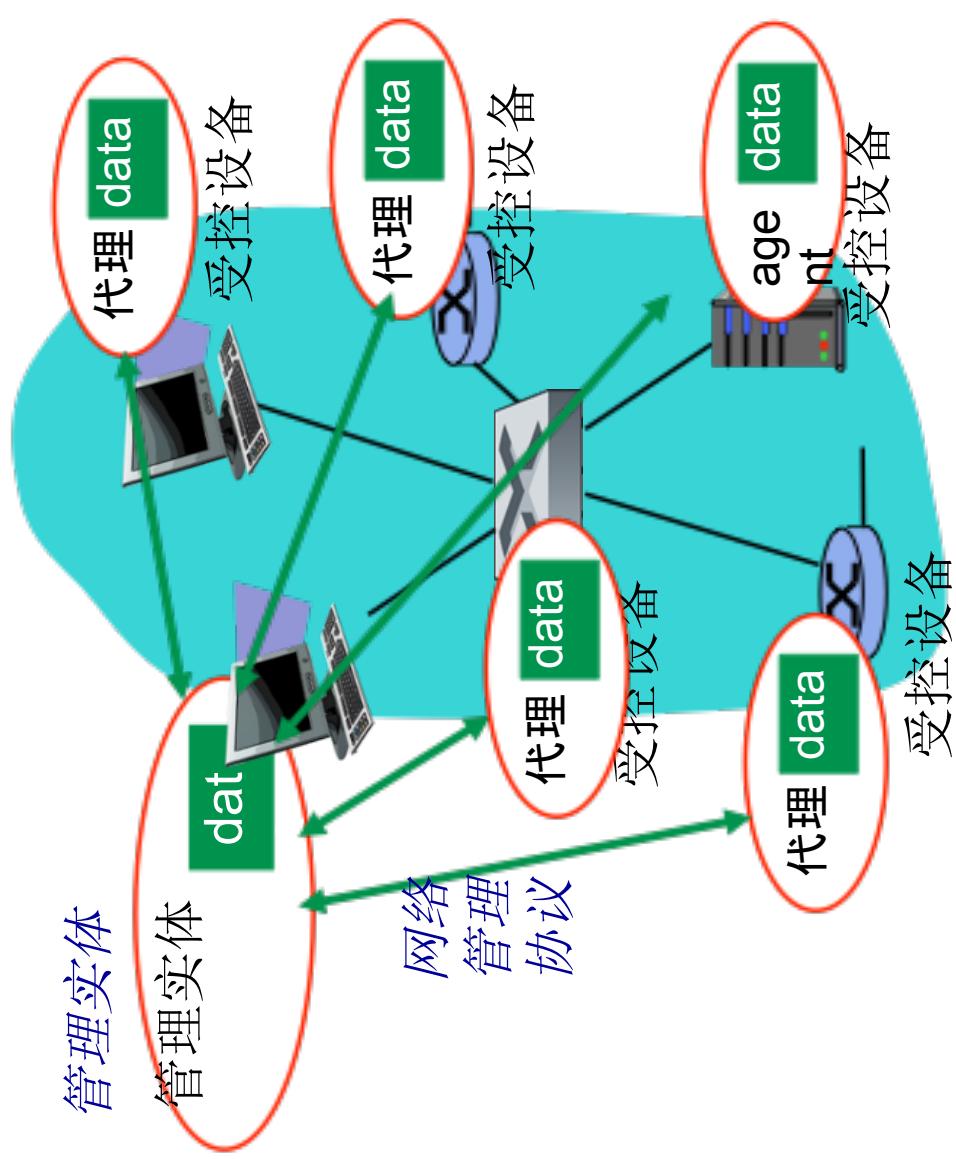
“**网络管理**要求以合理的成本部署、集成、协调硬件、软件、人力进行监测、调试、配置、分析、评估和控制网络和元件资源，以满足实时性、运行性能和服务质量。”



网络管理架构

定義：

受控设备managed 包含受控对象，其数据存储在管理信息库(MIB)



网络管理标准

OSI CMIP

- 通用管理信息协议
Common Management Information Protocol
- 设计于 1980's: 试图统一网络管理标准
● 标准化进程太慢
- 面向互联网 (SGMP)
● 从简单开始
● 部署后迅速采用
● 演化: 更大, 更复杂
- 现在: SNMP V3
- 事实的网络管理标准



SNMP 概要: 4 个关键部分



管理信息库 (MIB):

- 分布式存储网络管理数据

管理信息的结构 (SMI):

- MIB对象的数据描述语言

SNMP协议

- 在管理实体与受控对象之间交换信息和命令
- 安全和管理能力
- SNMPv3的主要增加特性

SMI: 数据定义语言



目的: 合理定义且无异议的

管理数据语法和语义

基本数据结构:

参考说明文档

OBJECT-TYPE

数据结构、状态、受控对象的语义

MODULE-IDENTITY

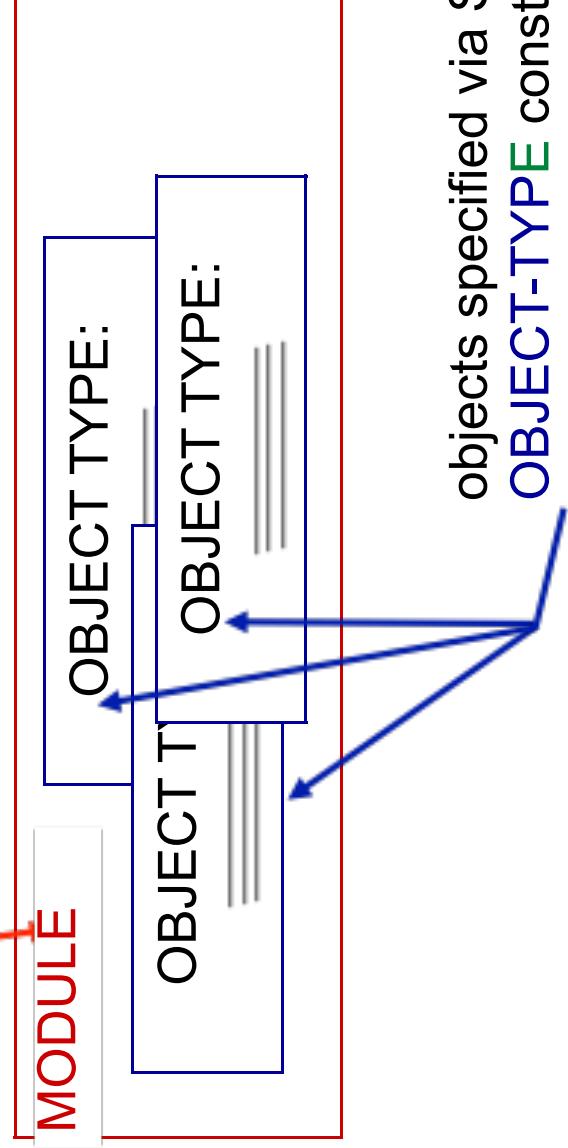
将相关对象编组存入MIB
模块

Basic Data Types

INTEGER
Integer32
Unsigned32
OCTET STRING
OBJECT IDENTIFIED
IPAddress
Counter32
Counter64
Gauge32
TimeTicks
Opaque

SNMP MIB

通过SMI定义MIB模块
MODULE-IDENTITY
(100 标准化MIBs, 更多是由供应商自己定义)



SMI: 对象、模块与例程



OBJECT-TYPE: ipInDelivers

MODULE-IDENTITY: ipMIB

iplnDelivers OBJECT TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
“The total number of input datagrams successfully delivered to IP user-protocols (including ICMP)”
::= { ip 9}

MIB 样例: UDP 模块



Object ID Name Type Comments

1.3.6.1.2.1.7.1	UDPIInDatagrams	Counter32	total # datagrams delivered at this node
1.3.6.1.2.1.7.2	UDPNoPorts	Counter32	# undeliverable datagrams: no application at port
1.3.6.1.2.1.7.3	UDInErrors	Counter32	# undeliverable datagrams: all other reasons
1.3.6.1.2.1.7.4	UDPOutDatagrams	Counter32	# datagrams sent
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	one entry for each port in use by app, gives port # and IP address

SNMP命名

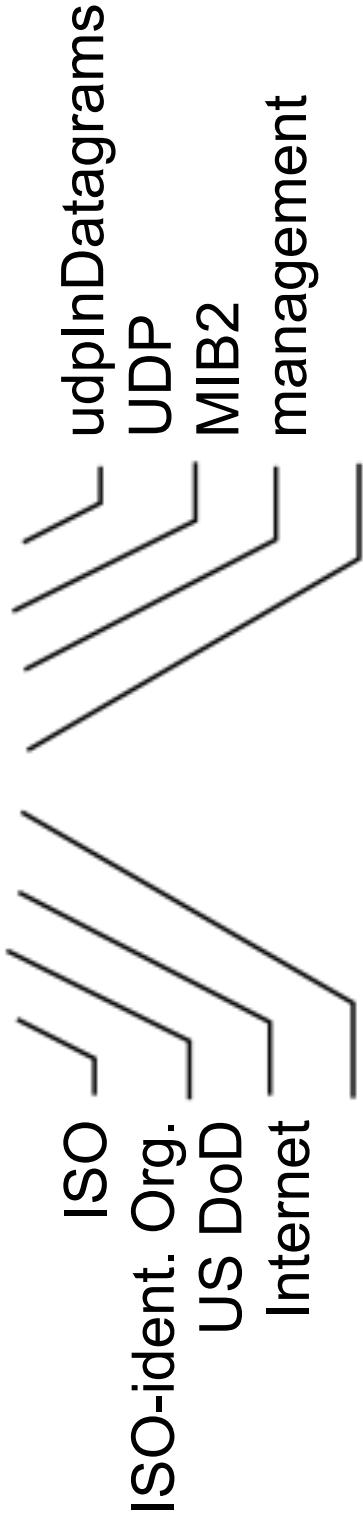


问题: 如何为所有可能的网络标准的每一个可能的标准对象命名 (协议、数据等)??

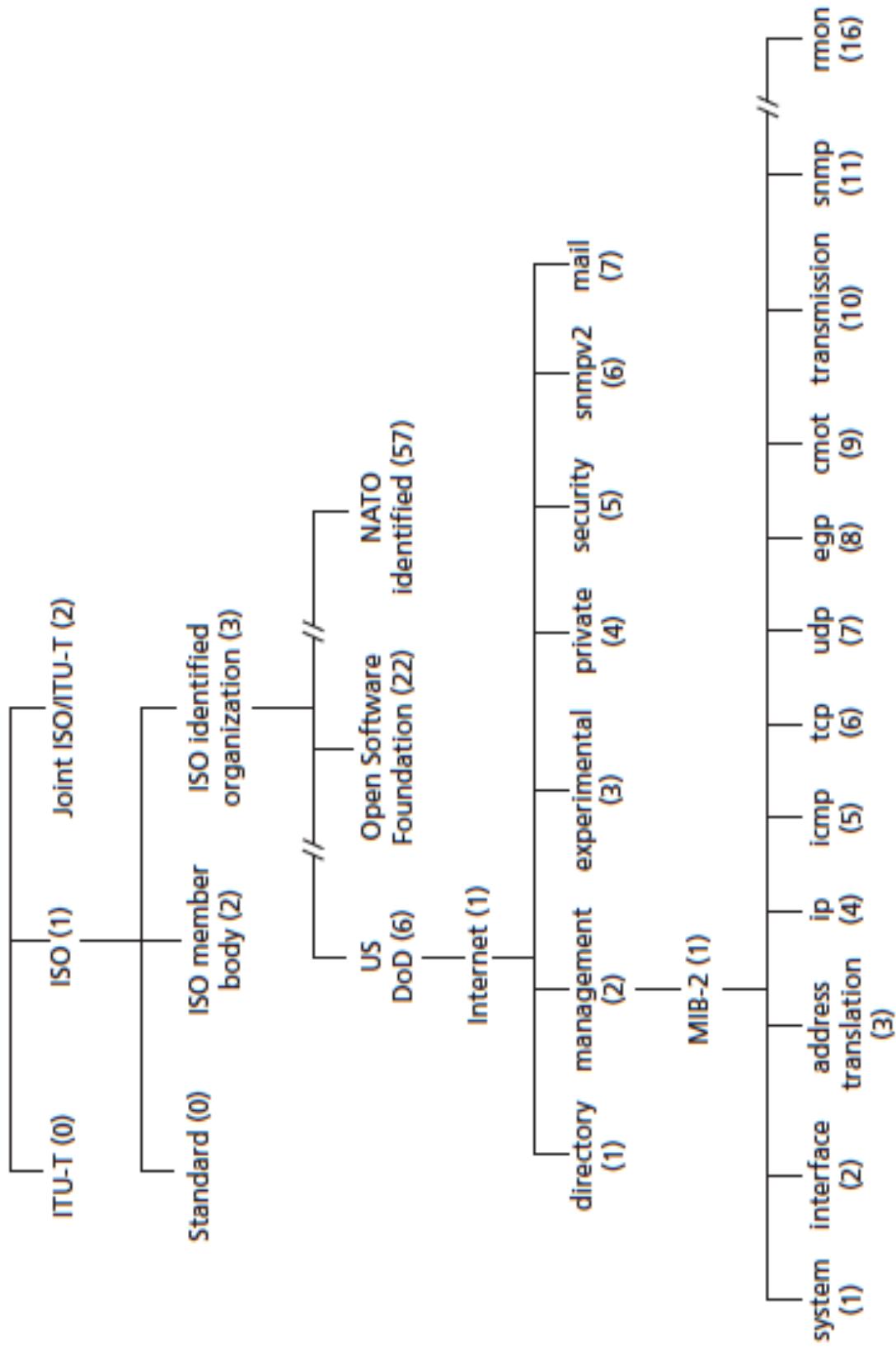
回答: ISO对象标志符树:

- 所有对象采用分级别命名
- 每个分支点有名字和数字

1.3.6.1.2.1.7.1

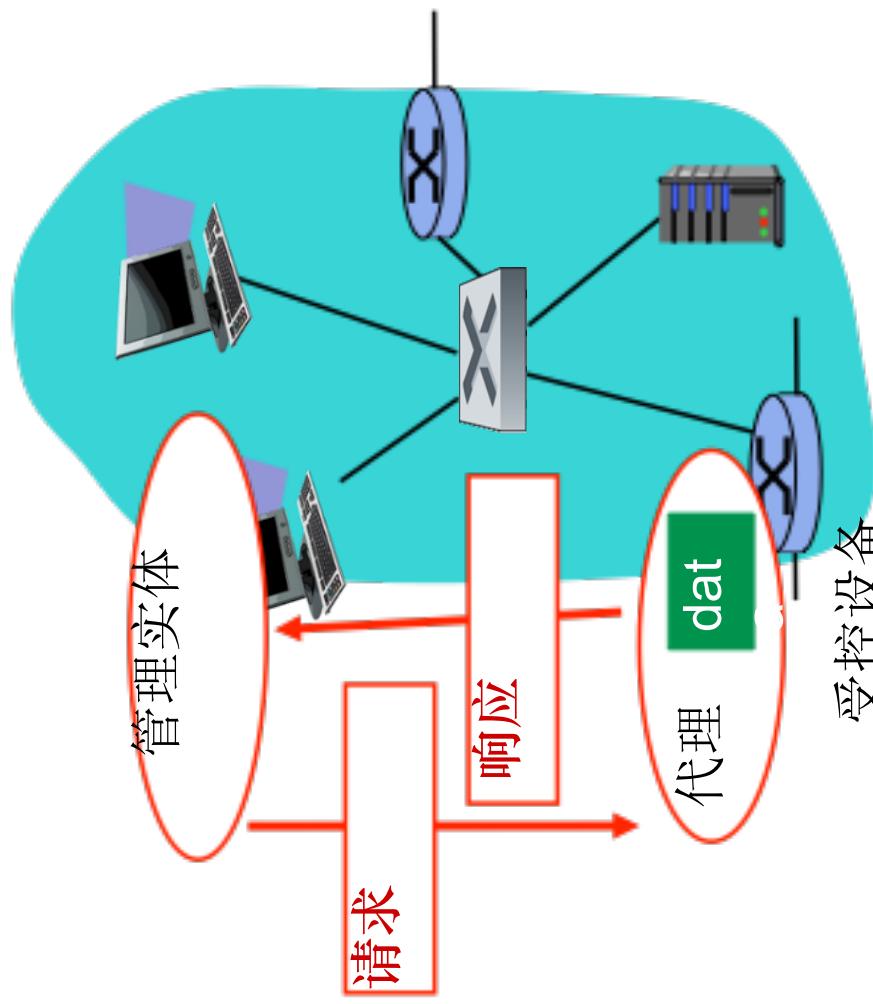


OSI 对象标志符符符

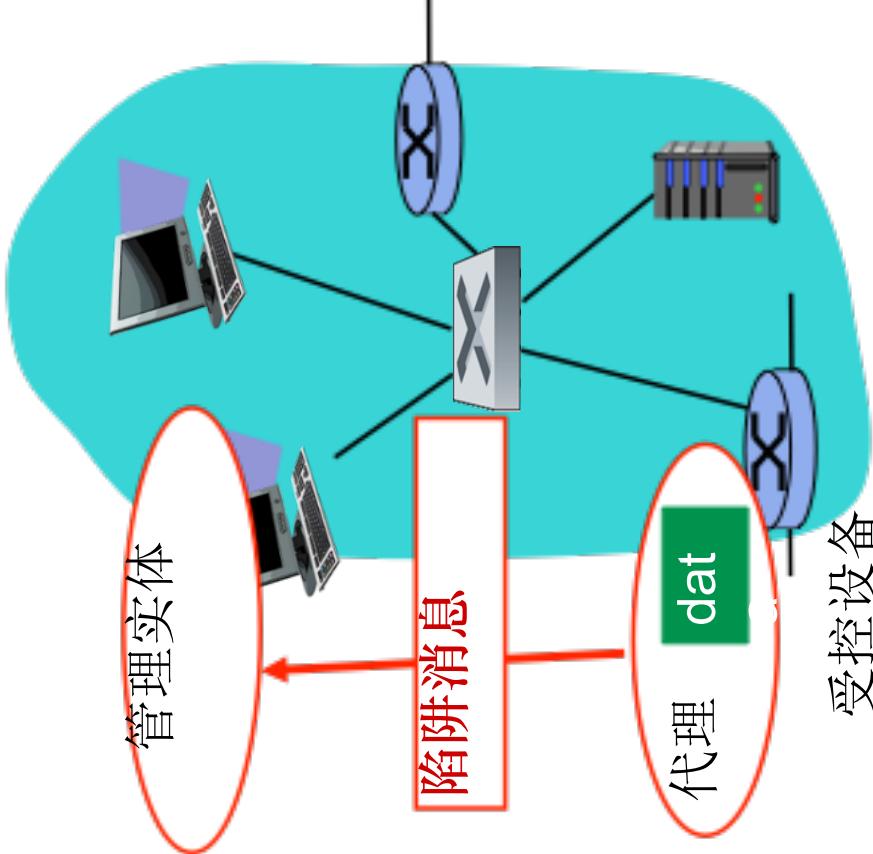


SNMP 协议

两种方法传递MIB信息和命令：



请求 / 响应模式



陷阱模式



SNMP 协议：消息类型

Message type

GetRequest
GetNextRequest
GetBulkRequest

Function

Mgr-to-agent: “get me data”
(instance,next in list, block)

InformRequest

Mgr-to-Mgr: here's MIB value

SetRequest

Mgr-to-agent: set MIB value

Response

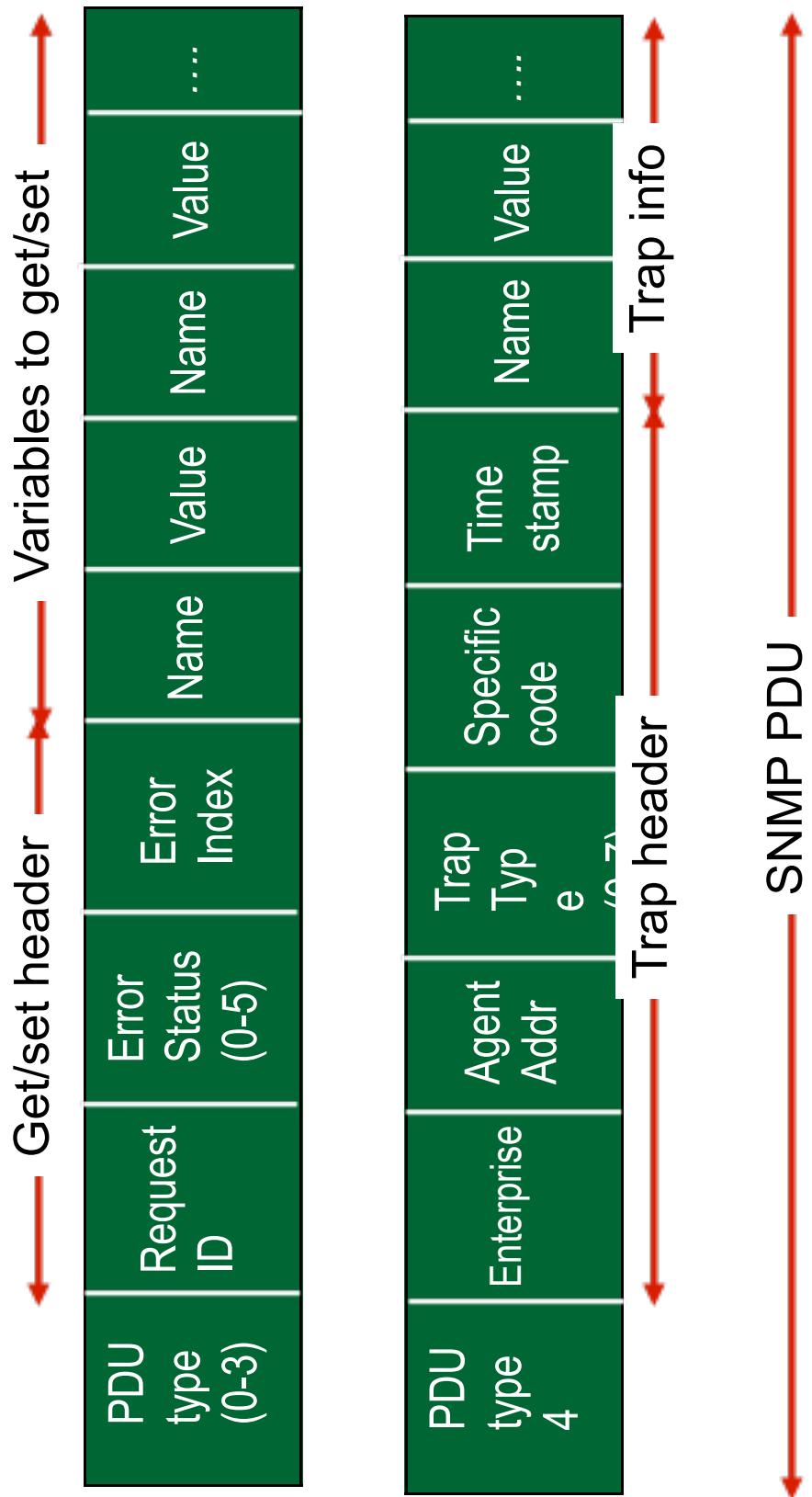
Agent-to-mgr: value, response to
Request

Trap

Agent-to-mgr: inform manager
of exceptional event



SNMP 协议：消息格式



SNMP 安全与管理

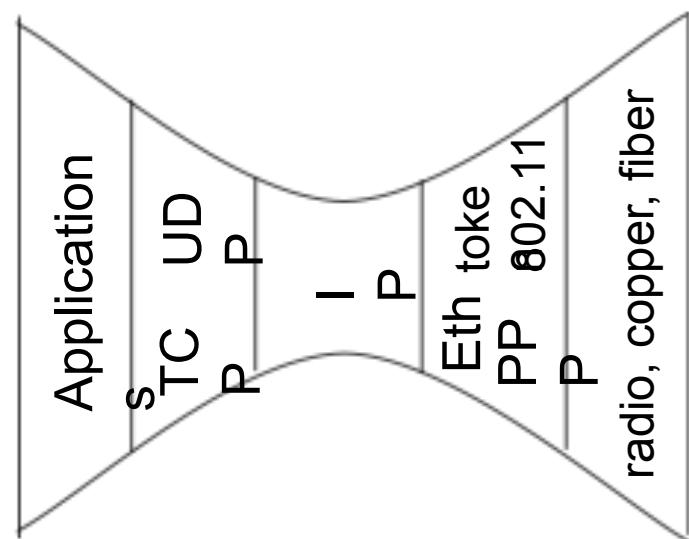


- 加密: DES加密SNMP消息
- 认证: 计算、发送MIC(m,k): 通过消息(m)计算MIC哈希值, 共享密钥 (k)
- 保护回放攻击: 使用nonce
- 可视化准入控制:
- SNMP主体为不同用户维护准入权限和策略数据库
- 数据块本身可作为受控对象进行访问!

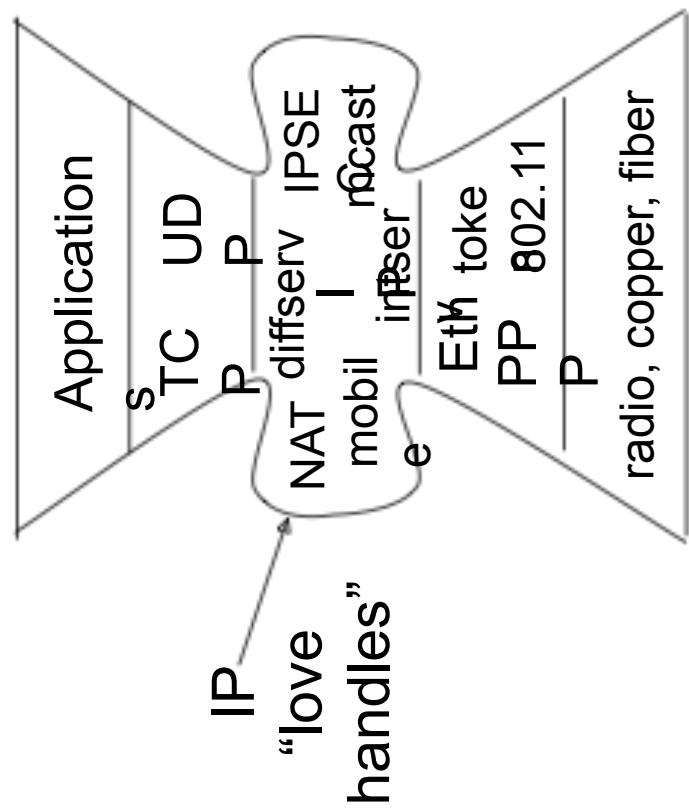


- 引言
- 核心问题: 应用需要自己的协议
- 传统应用
- 多媒体应用
- 基础设施服务
 - 覆盖网络
 - 路由覆盖
 - 对等网
 - 内容分发网络
- 总结

扩展视野：如何支持新的应用？

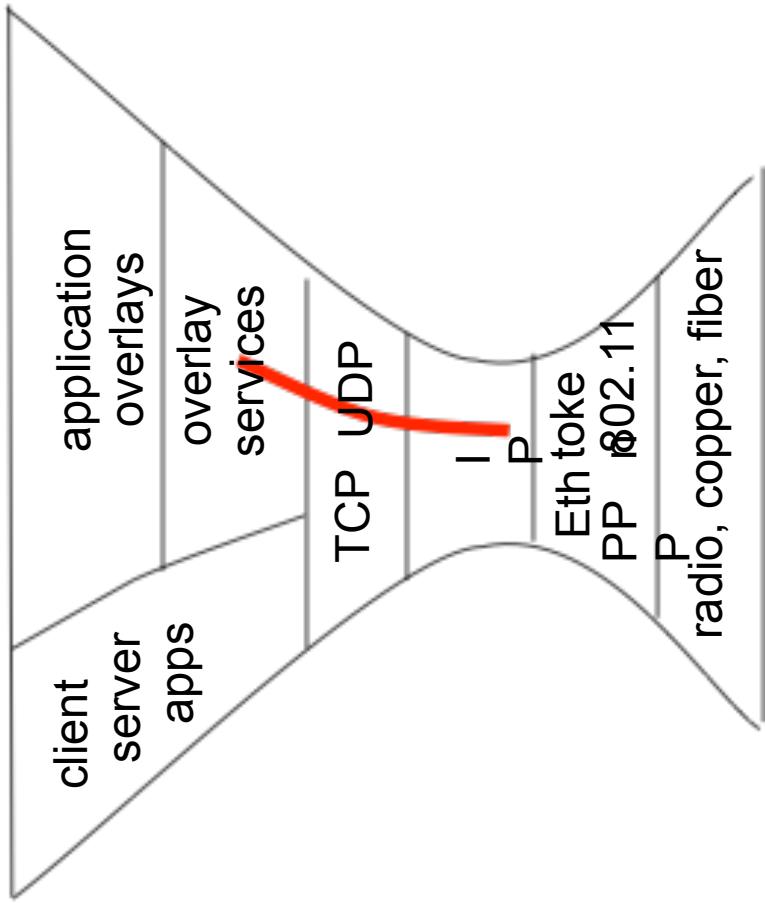
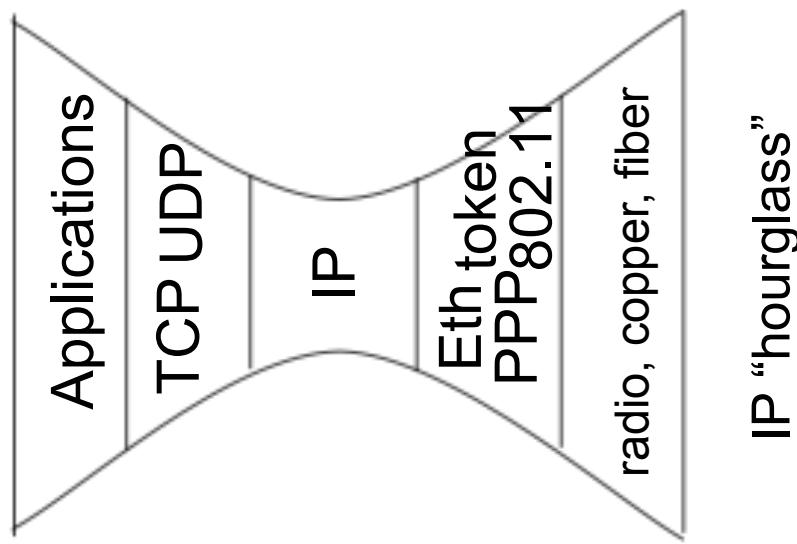


IP “hourglass”

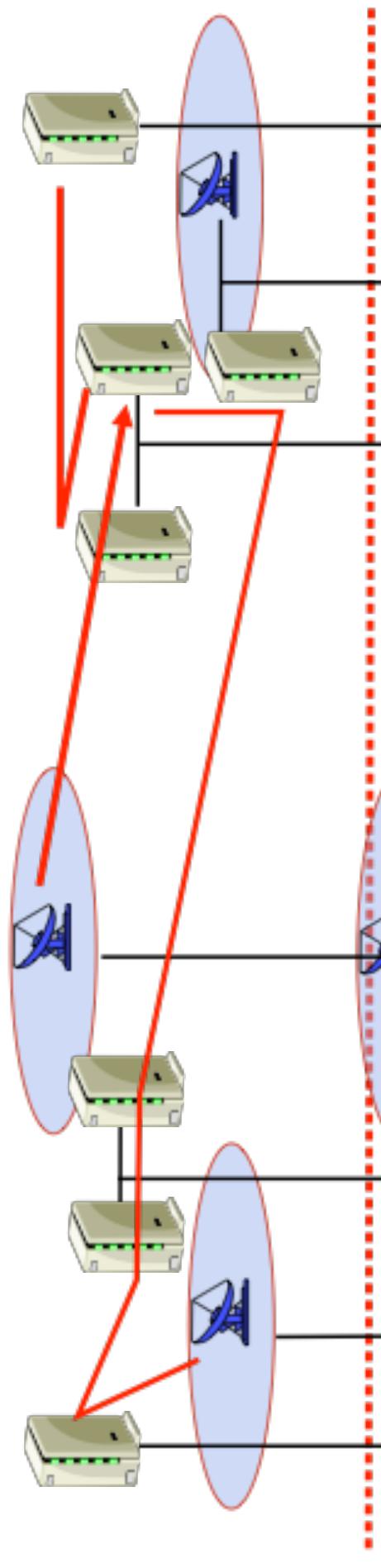


Middle-age IP “hourglass” ?

扩展视野：如何支持新的应用？

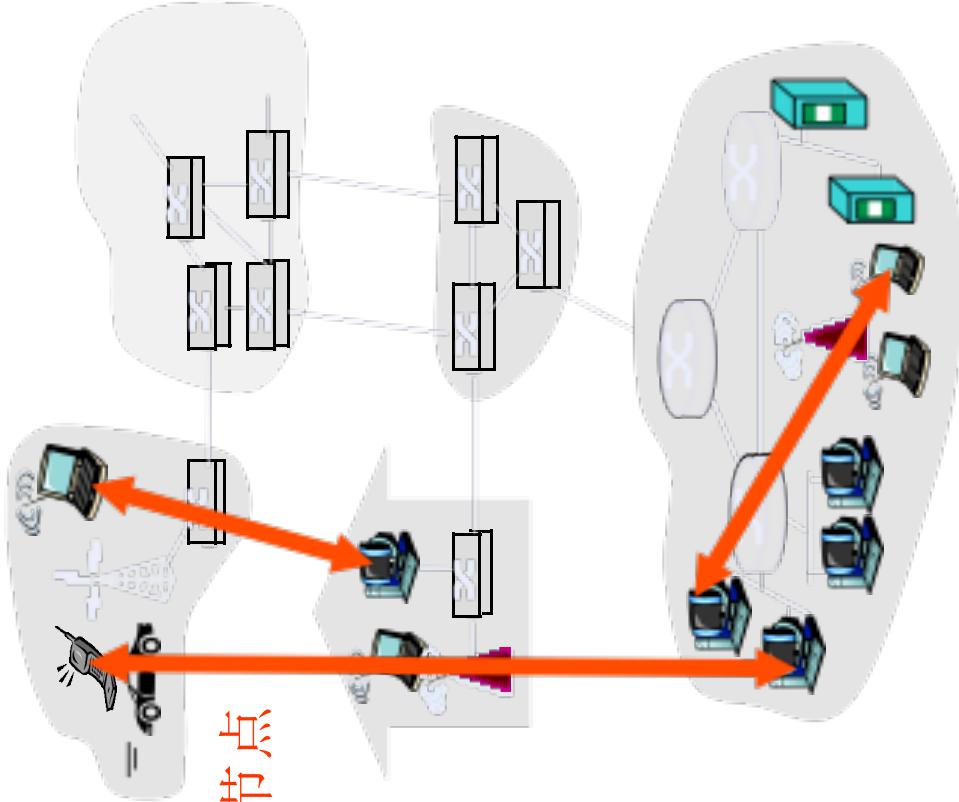


覆盖网络：在应用层完成网络互联



纯P2P结构

- 没有总是在线的服务器
 - 任意的端结点直接通信
 - 节点之间间歇的连接和互节点-节点换邻居的IP地址
- 三个主题:
- 文件分发
 - 搜索信息
 - 案例研究: Skype



P2P 网络



网络架构

非结构式
结构式
案例分析

Napster

Gnutella

Kazza

BitTorrent

.....

P2P 系统发展



- 1999: Napster, End System Multicast (ESM)
 - 2000: Gnutella, eDonkey
 - 2001: KaZaA
 - 2002: eMule, BitTorrent
 - 2003: Skype
 - 2004: CoolStreaming, PPLive
 - 2005: PPStream, SopCast, TVKoo, TVAnts, ...
 - ...

P2P 技术

文件共享

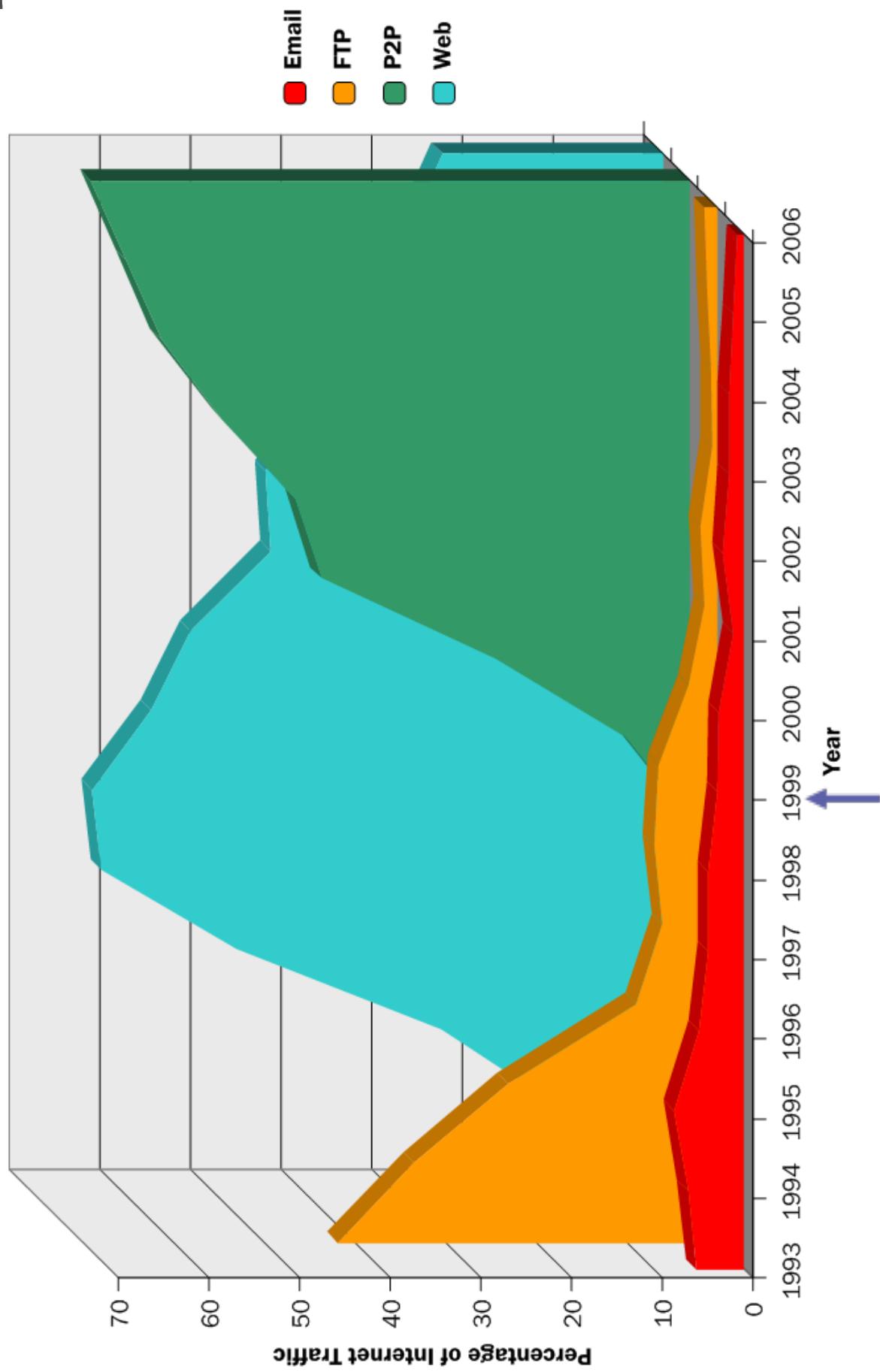
- Napster, Gnutella, KaZaA, BitTorrent
- 直播流媒体
- CoolStreaming, PPLive, PPStream, UUSee, ...
- 点播流媒体
- P2VoD, P2Cast, ...
- IP电话
- Skype, ...
- ...





互联网流量

CacheLogic Research | Internet Protocol Trends 1993 to 2006



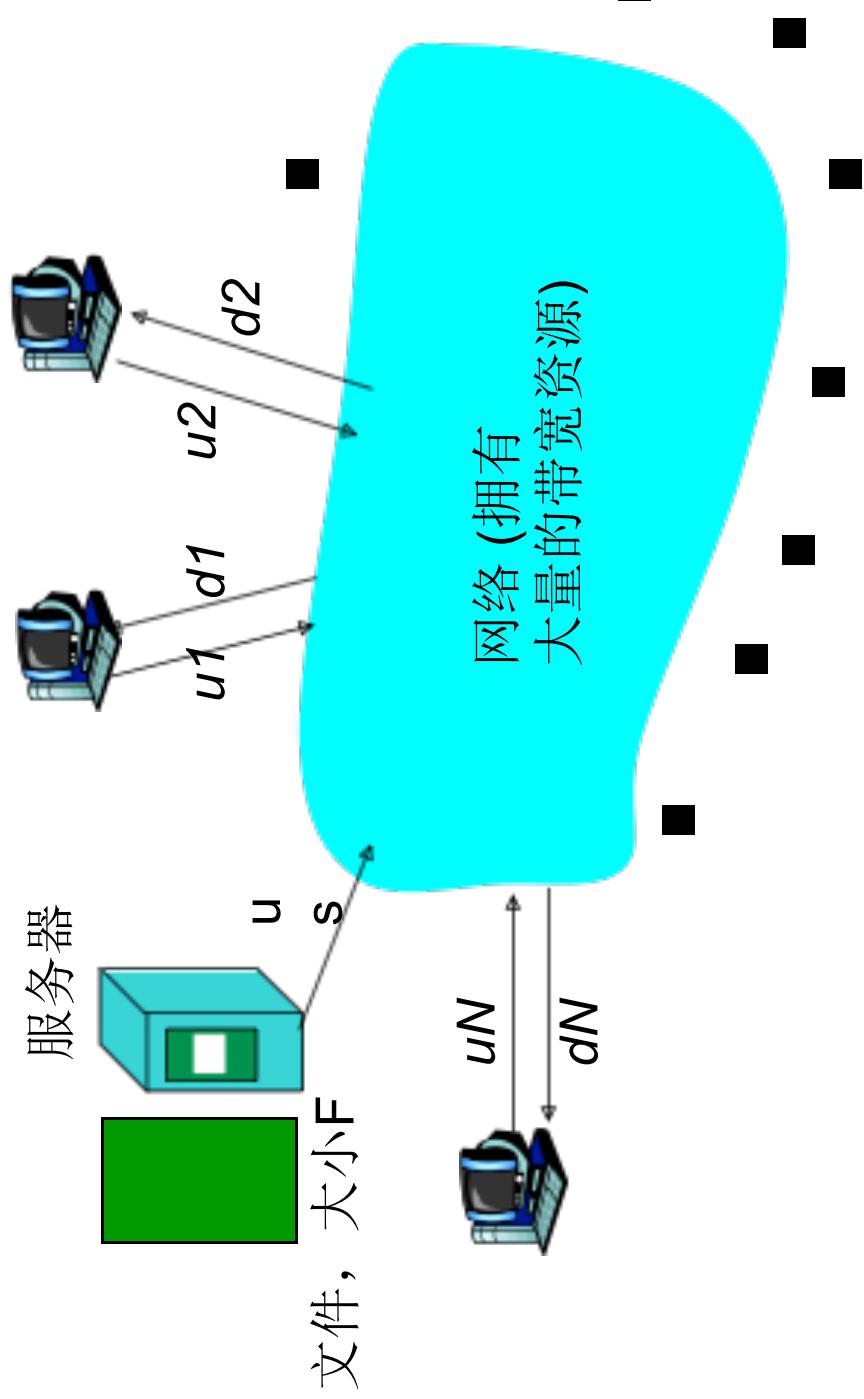
Napster



文件分发：服务器-客户端 vs P2P

问题：从一个服务器将文件分发给N个结点，需要多长时间？

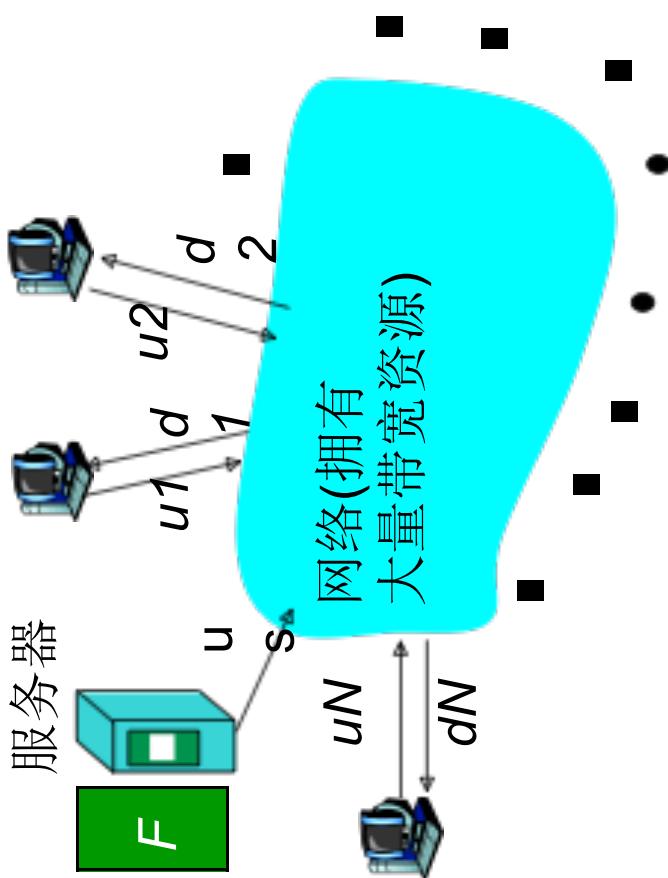
us: 服务器上传带宽



文件分发时间：服务器-客户端



- 服务器相继发送 N 个文件副本：
- 需要时间： NF/us
- 客户端需要 F/di 的时间来下载文件



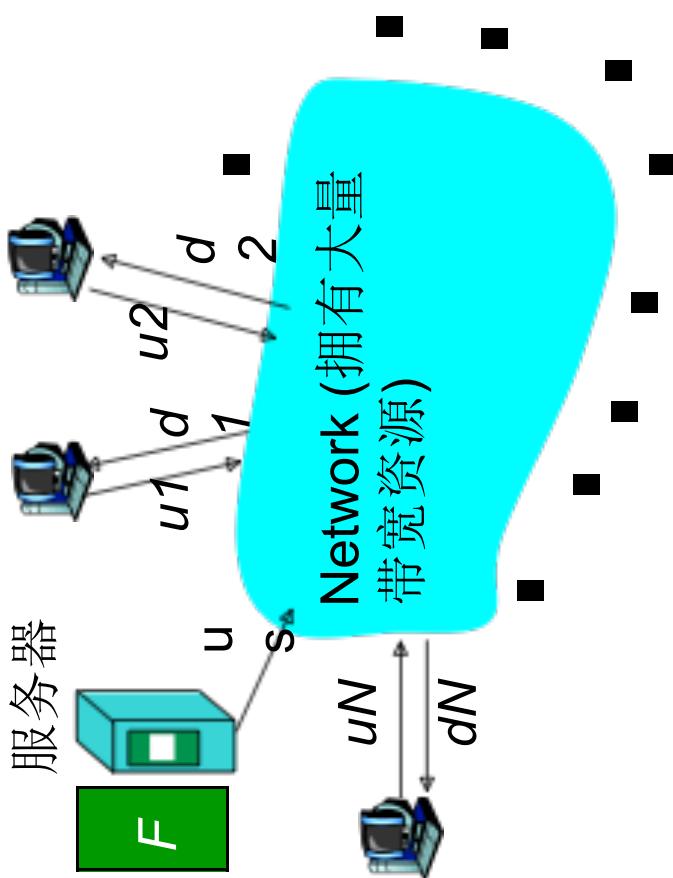
使用客户端/服务器方法，将文件发送给 N 个客户端所需要的时间

$$= dcs = \max_i \{ NF/us, F/min(di) \}$$

随着客户端数量增加
(当 N 很大时)

文件分发时间: P2P

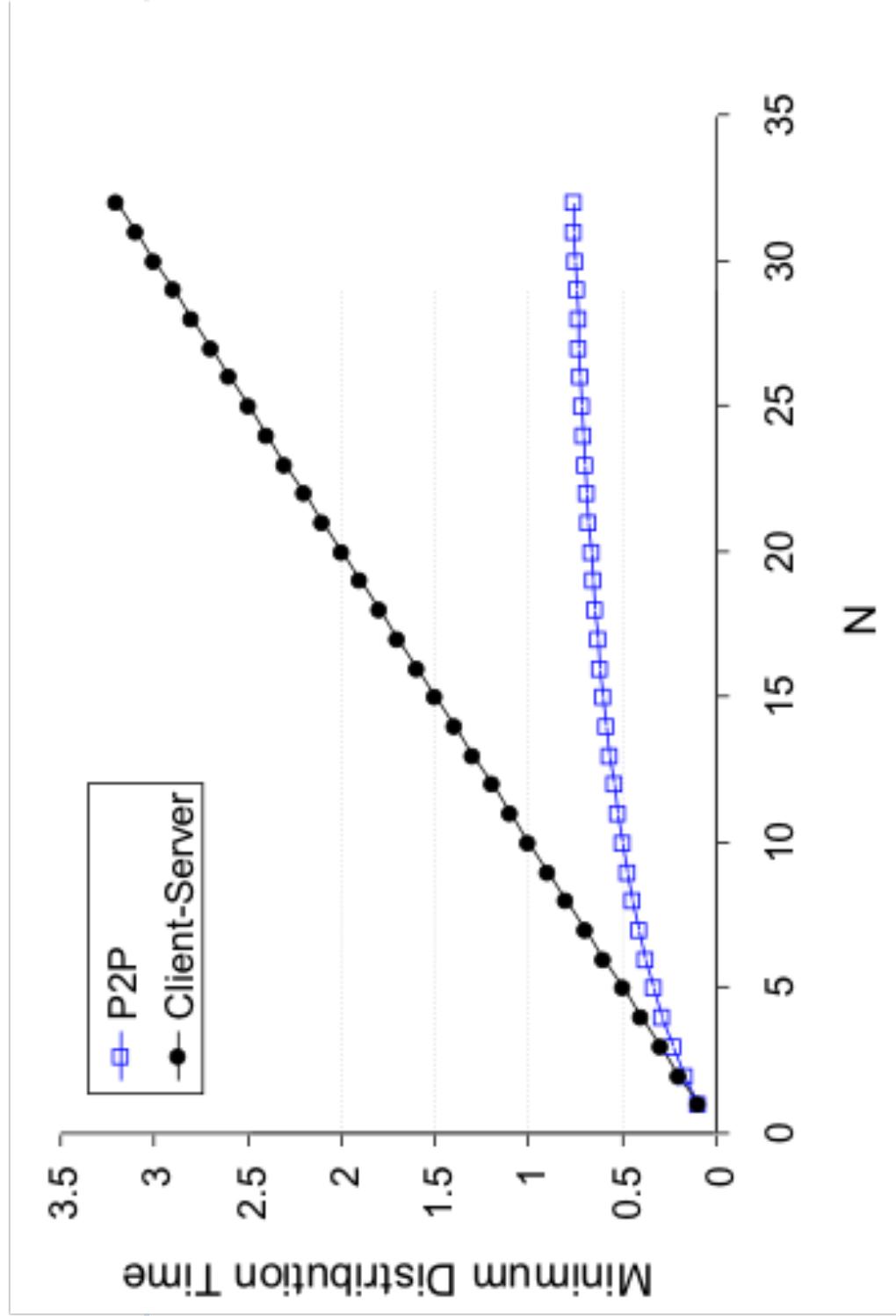
- 服务器起码需要传送一份文件副本: 所需时间 F/us
- 客户端i下载文件需要时间 F/di
- 总传输量:N个文件, 总计大小为NF, 必须被下载到N个客户端
- 可能的最高上传速率: $us + \sum_{ui} S_{ui}$



$$d_{P2P} = \max \{ F/us, F/min(di), NF/(us + \sum_{ui} S_{ui}) \}$$

服务器-客户端端点 VS. P2P: 示例

客户端上传速率 = U , $F/U = 1$ 小时, $US = 10U$, $d_{min} \geq US$

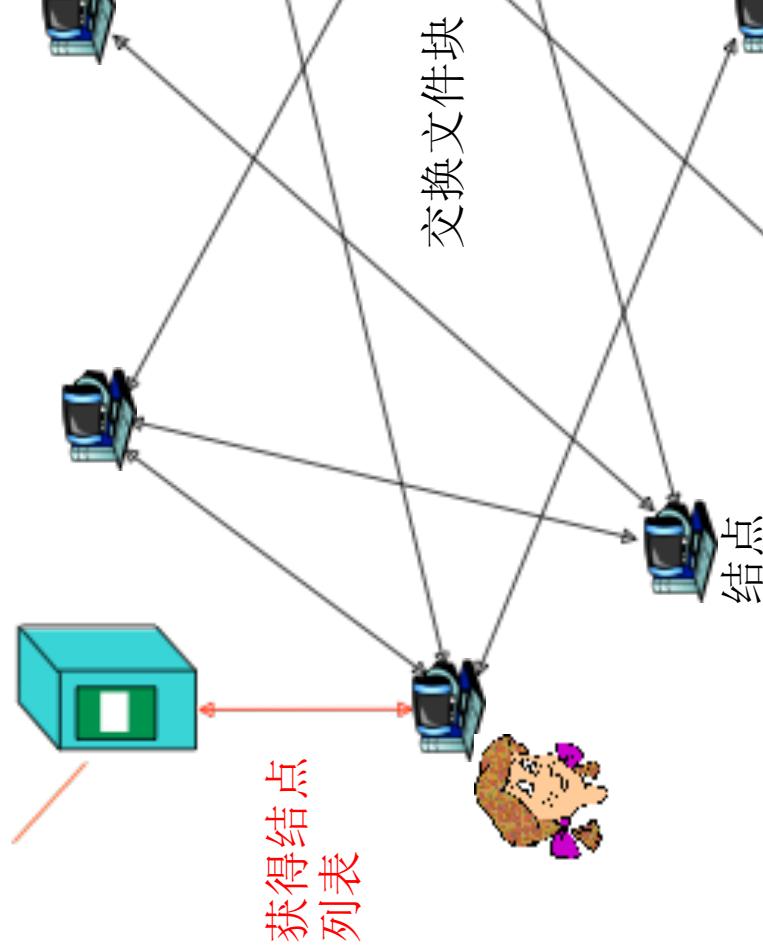


文件分发: BitTorrent

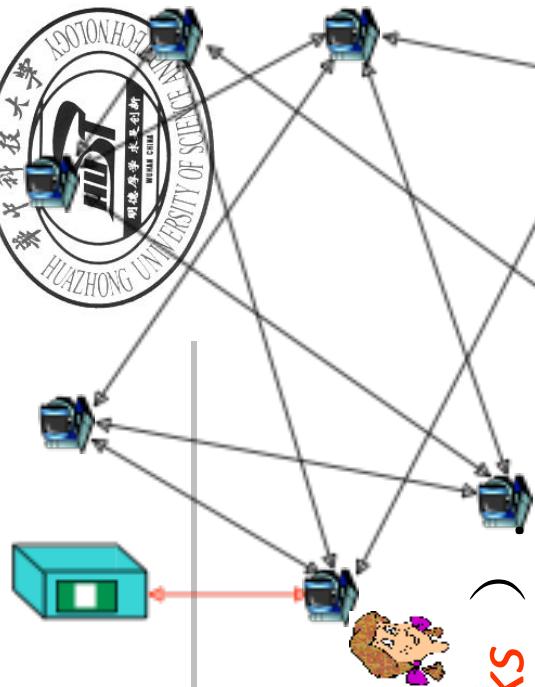
r P2P文件分发

tracker: 追踪参与这个
下载种子的结点

torrent: 互相交换文件
某个文件分块的一组结点



BitTorrent (1)



- 文件被分割成每块256KB的文件块 (**chunks**)
- 当结点加入这个torrent时:
 - 没有文件块，但是随着时间的推移，结点会积累文件块
- 在tracker服务器中注册，从而得到一个结点列表，然后与列表中的某些结点连接(“邻居”)
- 在下载文件块的同时，向其他结点上传已有的文件块。
- 结点随时加入，随时地离开
- 当结点下载完整个文件后，它可能（自私地）离开，也有可能（无私地）留下

BitTorrent (2)

拉式下载数据块

- 在任何时候，不同的结点拥有不同的文件块
- 某个节点（Alice）周期性的向它的邻居节点询问它们拥有的文件块列表。
- Alice发送请求消息，请求下载她没有的文件块
- 最少优先算法

发送文件块:针锋相对 (tit-for-tat)

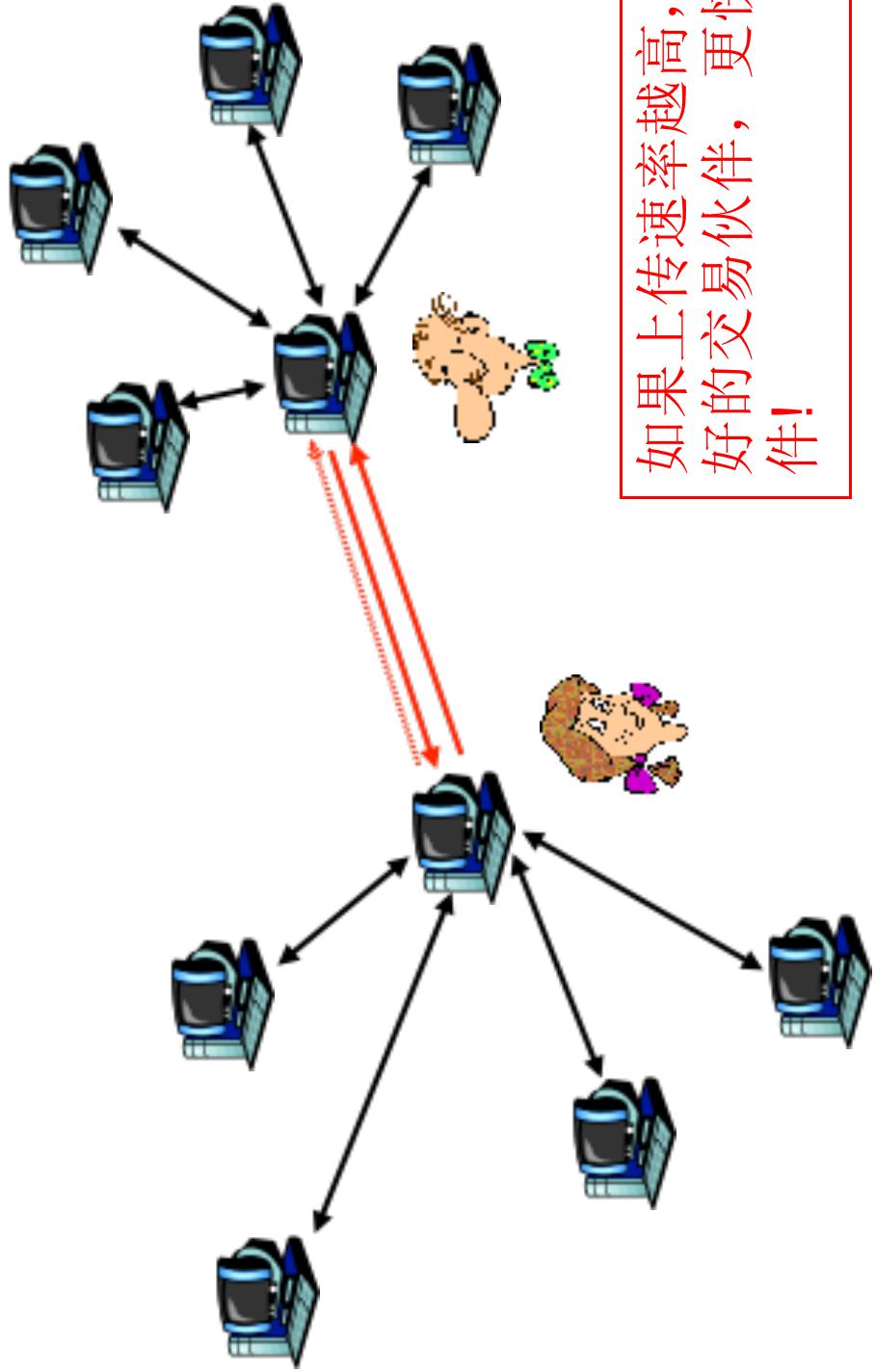
- Alice以最快的速度率给4个邻居节点发送文件块
- 每隔10秒，重新评估处于前4的节点
 - 每隔30秒：随机的选择另一个节点，开始发送文件块
 - 新选择的节点可能处于排名的前4位
 - “乐观的 unchoke策略”



BitTorrent: Tit-for-Tat 针锋相对



- (1) Alice “乐观的 unchoke” Bob
- (2) Alice成为了Bob排名前四位的文件提供者之一; Bob将对此做出报答
- (3) Bob成为了Alice前四位文件提供者之一



分布式哈希表(DHT)



- DHT = 分布式P2P数据库
- 数据库有(**键, 值**)数据对;
 - 键: 身份证号码; 值: 人名
 - 键: 内容类型; 值: IP地址
- 结点使用键值(key)向数据库提出询问
- 数据库将符合键值的数据(value)返回
- 结点也可以插入(键, 值)数据对

DHT 识别符



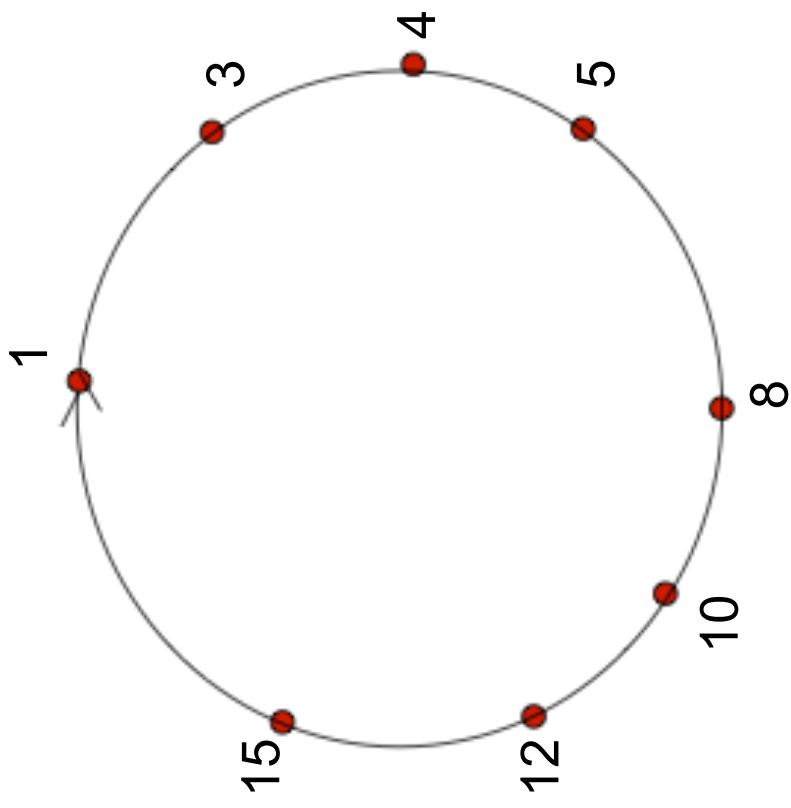
- 为每个节点设置一个整数的识别符，范围在 $[0, 2^n - 1]$ 。
 - 每个识别符可以用 n 个比特位表示。
 - 要求每个键值都是在**相同范围内**的整数。
 - 为了得到整数的键值，hash 原始的键值。
 - 例如， $\text{key} = h(\text{"Led Zeppelin IV"})$
 - 这就是人们为什么称之为分布式“哈希”表。

如何给节点设定键值？



- 中心问题：
- 为节点设定(键, 值)数据对。
- 规则：将键值分配给具有**最近ID**的结点。
- 约定：最近的ID是键值的直接后继。
- 例如： $n=4$; 节点： $1, 3, 4, 5, 8, 10, 12, 14$;
 $\text{key} = 13$, 那么他的后继节点 = 14
 $\text{key} = 15$, 他的后继节点 = 1

环形 DHT (1)

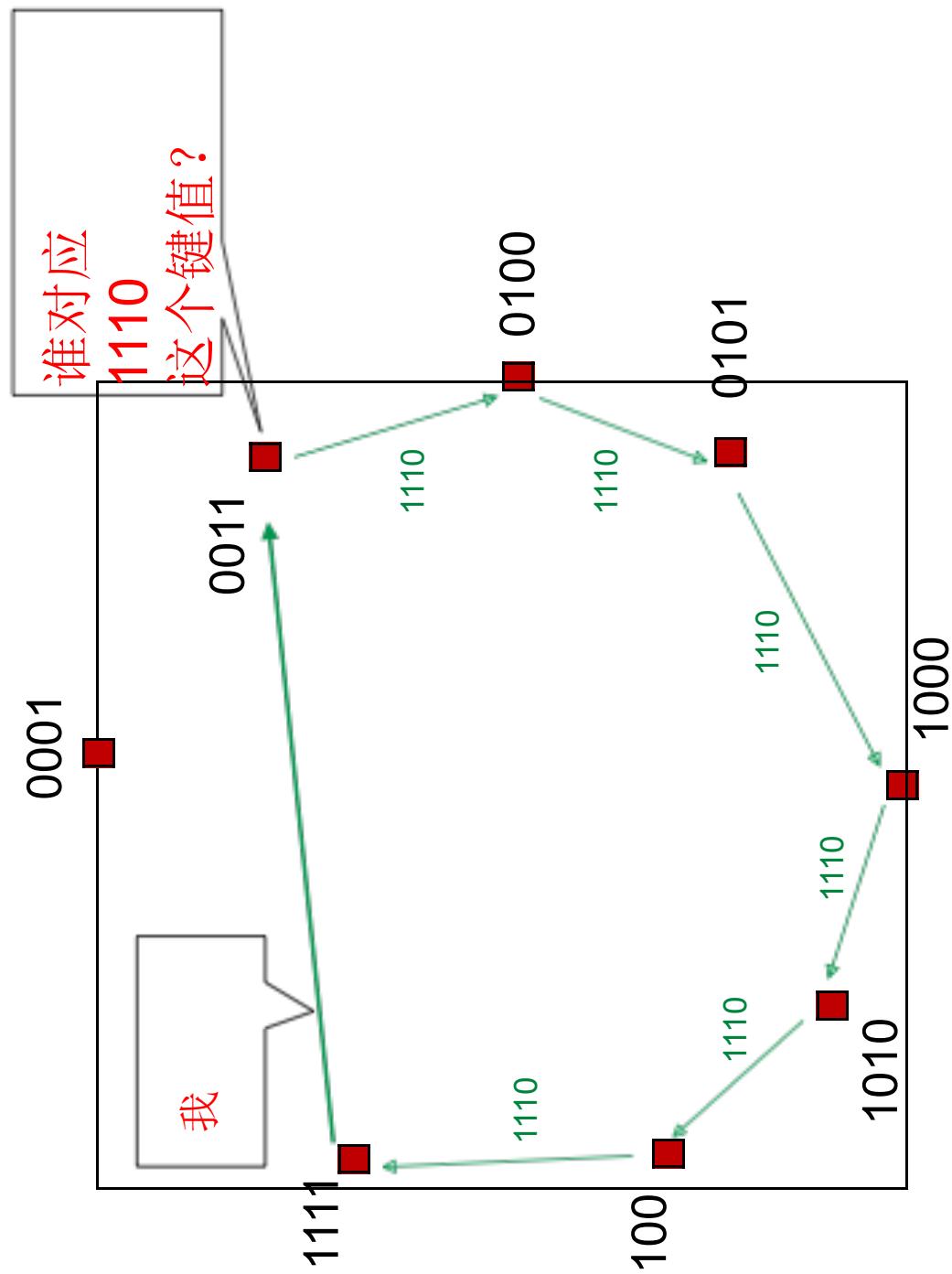


- 每个节点只知道它的直接后继和直接前驱。
- “覆盖网络”



环形 DHT (2)

当存在N个节点时，
平均O(N)条消息用
来解析询问消息



谁对应
1110
这个键值？

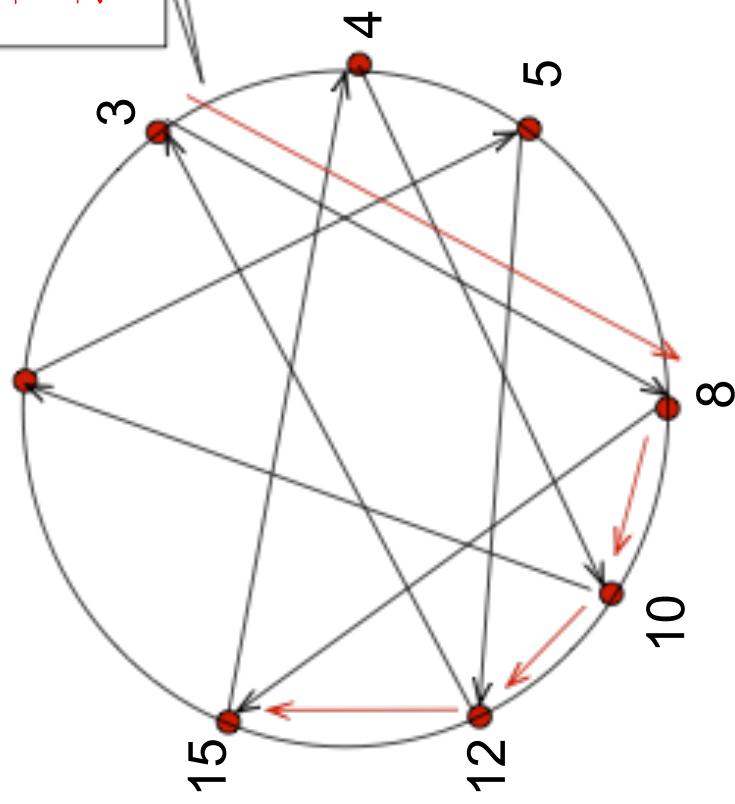
最近的
是最近点
定义是
意是后继
的



带捷径的环形DHT



谁对应
这个键值？

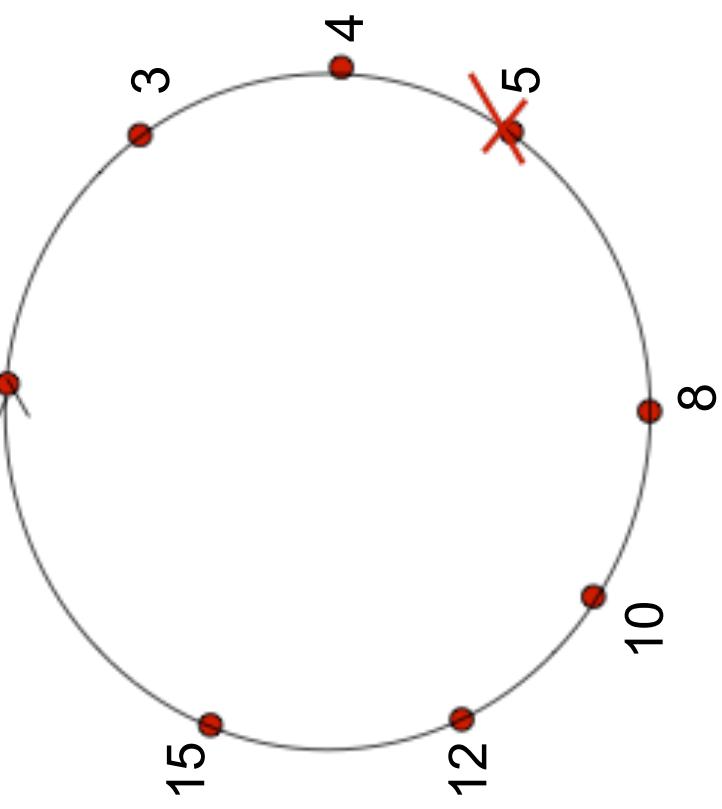


- 每个节点记录前驱，后继，捷径的IP地址。
- 消息数量从6条减少到4条。
- 设计捷径后，每个节点有 $O(\log N)$ 个邻居，在解析询问时产生 $O(\log N)$ 条消息。

节点的动态特性



- 为了解决节点的动态特性，要求每个节点记录它两个后继节点的IP地址。
- 每个节点周期性的ping它的两个后继节点，来检查他们是否仍在。
- 节点5突然的离开节点4察觉到了；将节点8设置成它的直接后继；将节点8的直接后继节点设置成它的第二后继。
- 如果节点13想要加入，将如何操作？



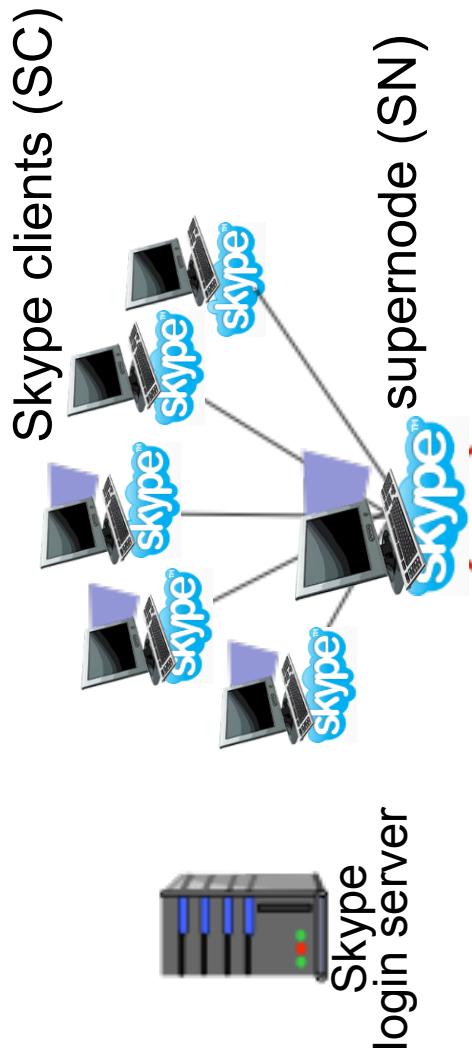
Voice-over-IP: Skype



- 私有应用层协议(可通过反向工程解构)

消息加密

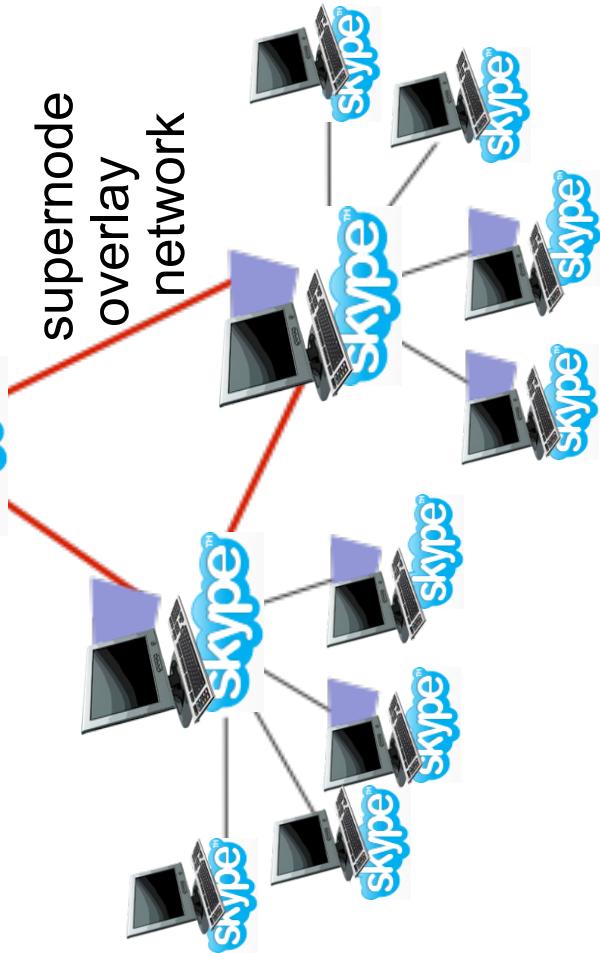
- P2P组件:



客户端: skype节点之直接互连，完成话间直通语音通话

超级节点(SN): 赋予特殊功能的skype节点

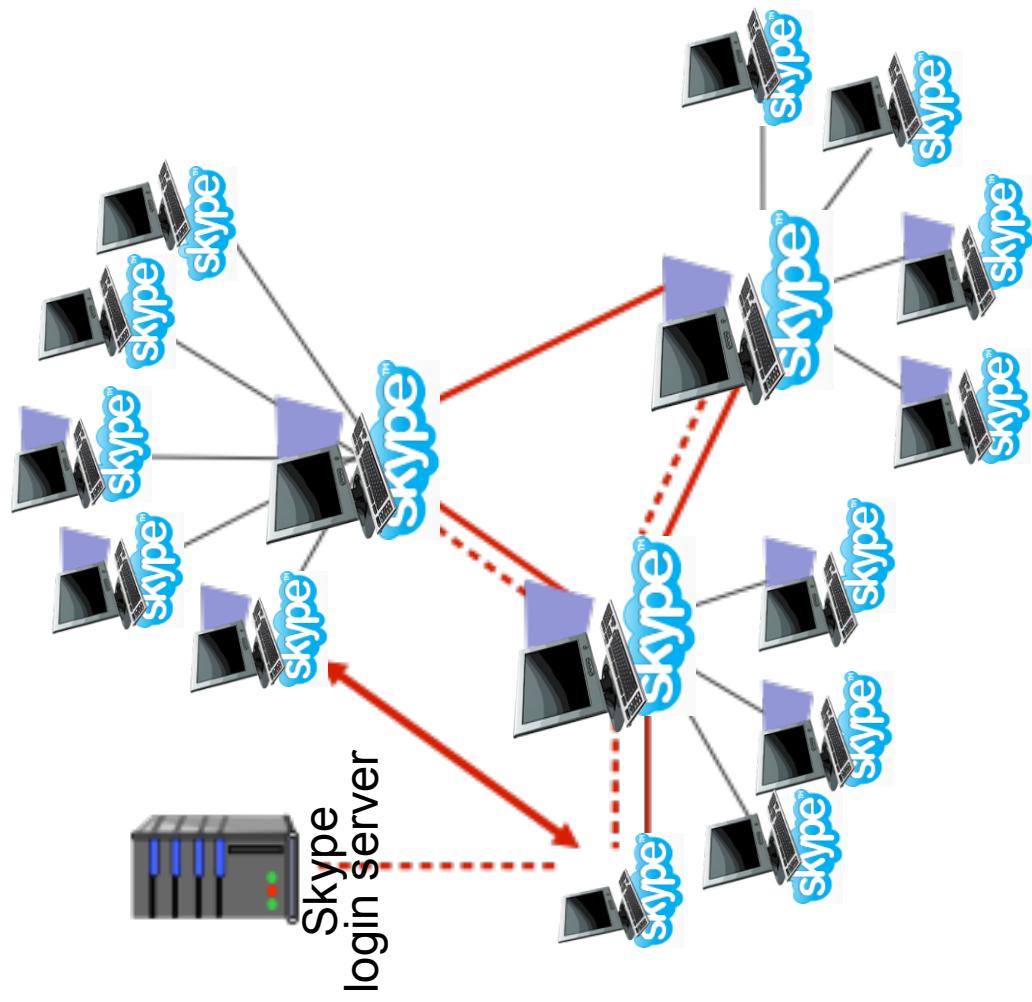
覆盖网络: 通过超级节点位置的配合确定普通节点位置
登录服务器





P2P voice-over-IP: skype

Skype客户端操作流程：



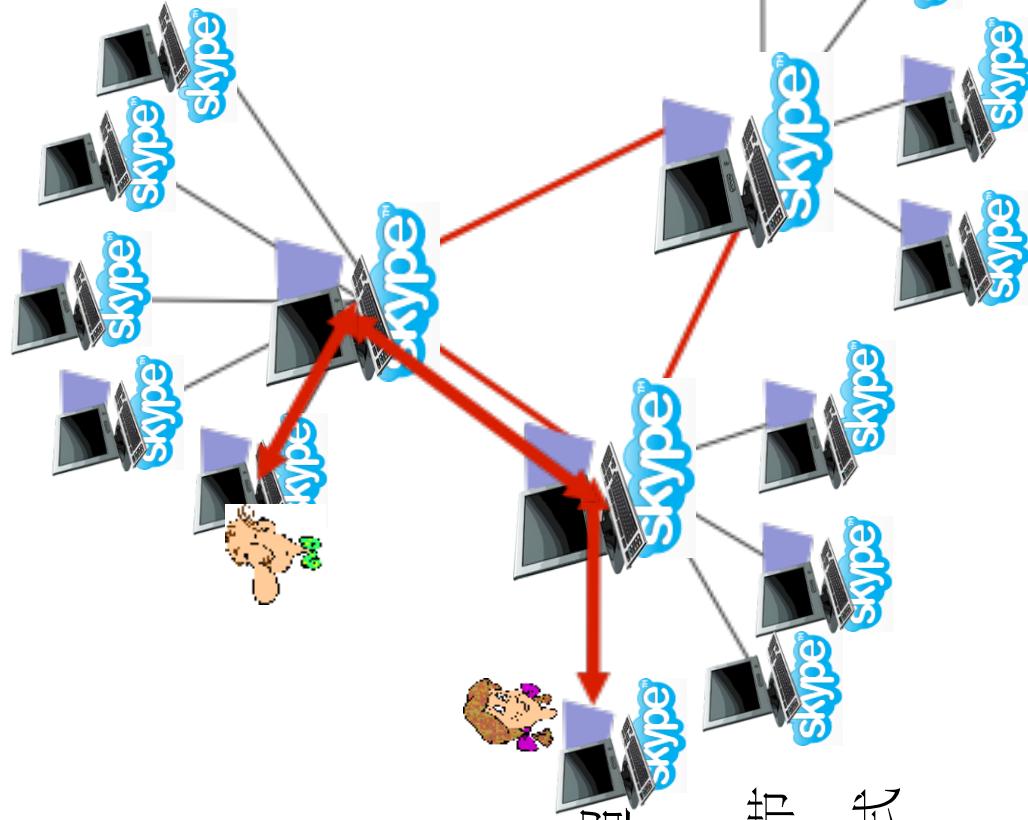
1. 通过TCP联系登录服务
器加入Skype网络
2. 提供用户名和密码登录集中
式skype用户管理服务器
3. 超级节点构成覆盖网络，普通
超级节点从超級节点的IP地址
或者客户端的朋友列表
• 向被叫节点直接发起呼叫
4. 向被叫节点直接发起呼叫

Skype: 节点作为中继



- **问题:**当Alice和Bob都处于“地址转换”设备后面

- NAT阻止外部的结点向内部节点发起呼叫
内部节点可以向外网发起
链接请求



- **中继方案:** Alice和Bob分别同其超
级节点维护开放式链路
Alice指示其超级节点连接Bob
Alice的超级节点连接Bob的超级节
点
Bob的超级节点通过对应的开放式
链接连接到Bob

内容分发网络(CDN)



- **挑战:** 如何将选自百万级视频内容以流媒体的形式同时传输给成千上万用户?

- **选项1:** 单个大型超级服务器

- 单点故障
- 网络拥塞瓶颈
- 远离客户
- 出口链路传输多个相同视频内容

....简而言之:这种方法无法扩展

内容分发网络(CDN)



- **挑战:** 如何将选自百万级视频内容以流媒体的形式同时传输给成千上万用户?

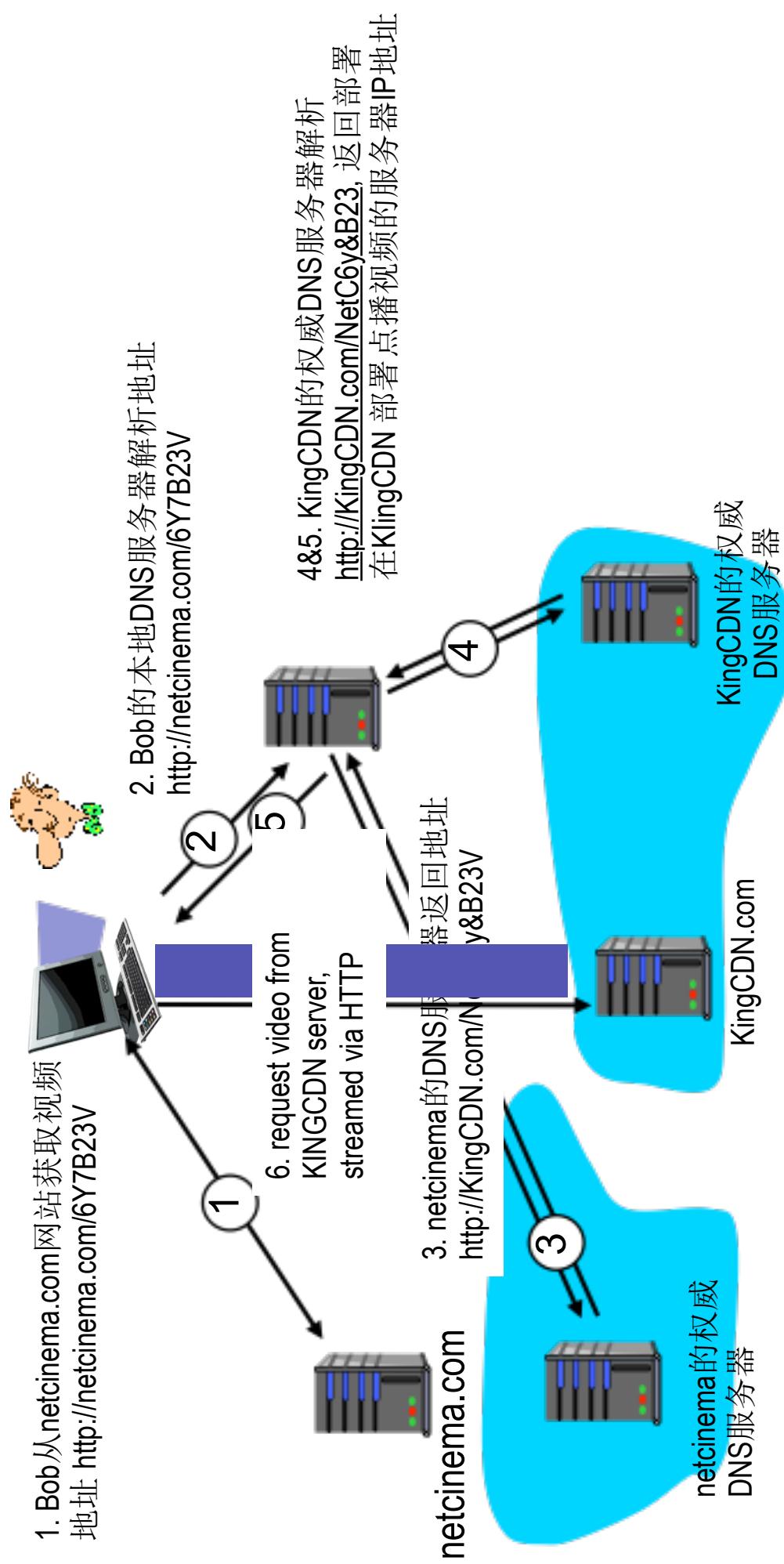
- **选项2:** 通过多个分布在不同地理位置的站点(CDN)存储和传输视频内容的多个备份
- **深度部署:** 将CDN服务器深度部署到很多接入网中接近用户
 - Akamai采用, 服务器遍布1700地点
 - 集中部署: smaller 在接近接入网络的网络汇聚点不是少量(10's)大型服务器群
 - Limelight采用



CDN: “简单”内容访问场景

Bob (用户) 点播视频 <http://netcinema.com/6Y7B23V>

• 视频存储在CDN的地址 <http://KingCDN.com/NetC6y&B23V>



CDN集群选择策略



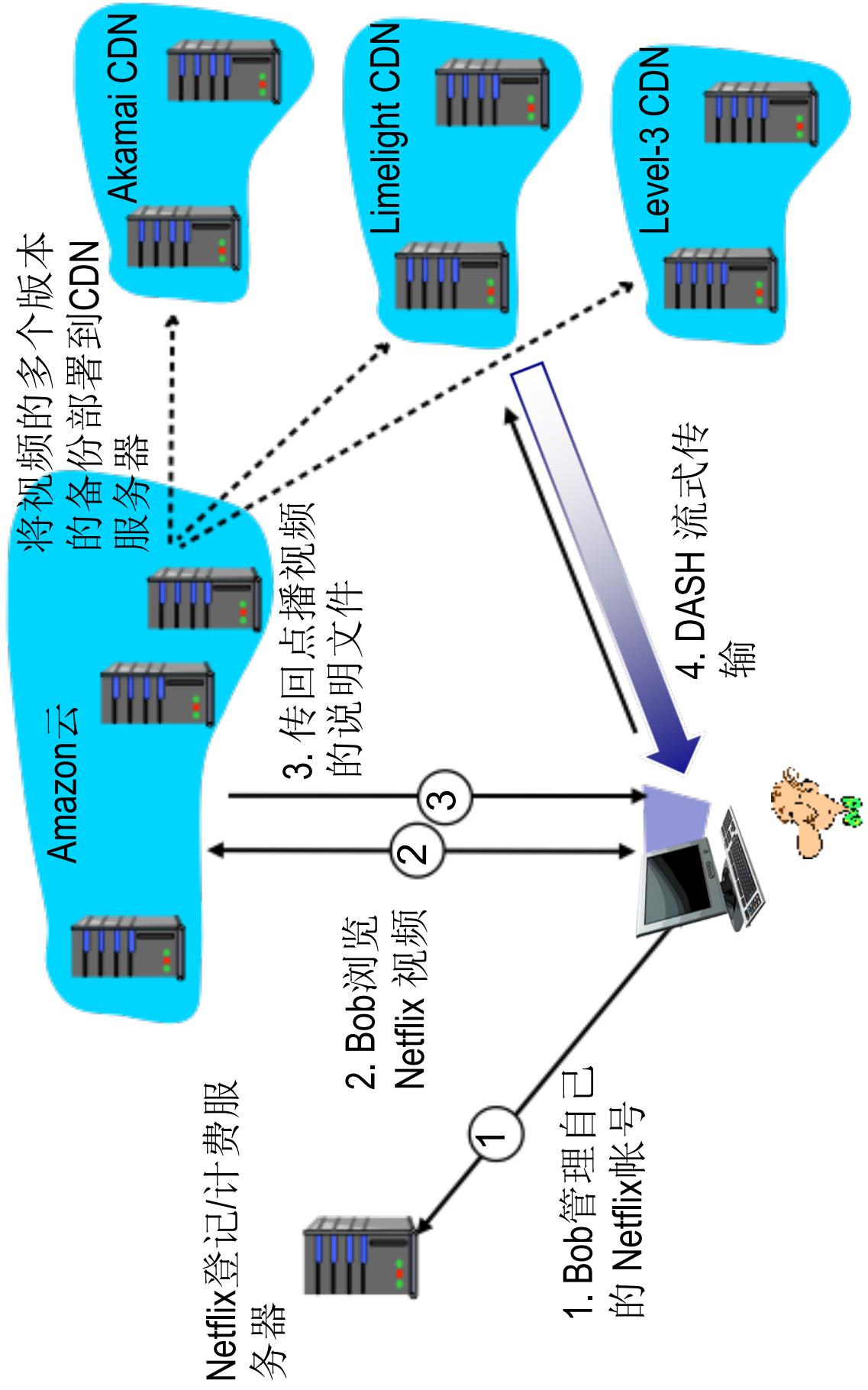
- **挑战:** CDN的DNS如何选择“好”的CDN节点向客户端传输视频?
 - 选择地理位置靠近客户的CDN节点
 - 选择距离客户时延最短(或者最小跳数)的CDN节点, 可周期性ping接入ISP, 向CDN DNS报告测量结果
 - IP anycast
- **可选:** 让客户选择一提供CDN服务器器列表
 - 客户ping服务器, 选择“最佳”
 - Netflix方式

案例: Netflix

- 2011年占据美国30%下载流量
- 自身拥有很少网络架构，主要是使用第三方服务：
 - 自己的登记、付款服务器
 - 第三方Amazon云服务：
- Netflix将媒体控制中心部署在Amazon云端
- 使用不同编码，在云端生产不同版本的视频
- 将不同版本视频从云端部署到CDN服务器
- 用户浏览器部署在云端的Netflix网站页面
- 使用三家第三方CDN网络服务存储和传输
- Netflix内容：Akamai, Limelight, Level-3



案例: Netflix





- 引言
- 核心问题：应用需要自己的协议
- 传统应用
- 多媒体应用
- 基础设施服务
- 覆盖网络
- 总结

总结



- 网络架构
 - 客户端-服务器
 - 对等(P2P)
 - 应用服务需求:
 - 可靠性、带宽、时延
 - 基础设施服务
 - 域名解析(DNS)
 - 网络管理(SNMP)
- 非常重要: 80%网络开销
 - 传递信息的协议
 - 仍缺少科学系统的方法

应用协议:

- SMTP, POP, IMAP
- HTTP
- DNS
- P2P: BitTorrent, DHT
 - 典型请求/回答消息交换
 - 客户端请求信息或服务
 - 服务器返回数据或状态码
- 消息格式
 - 报头: 字段描述数据信息
 - 数据: 通信的信息



- Chapter 9 in L. L. Peterson and B. S. Davie, *Computer Networking: A System Approach (5th edition)*, Morgan Kaufmann, 2012
- Chapter 2/8 in James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach (6th edition)*, Pearson Education Inc., 2012
- 吴功宜, 计算机网络 (第3版), 清华大学出版社社, 2011

谢谢！

华中科技大学
电子信息与通信学院
Email: itec@hust.edu.cn
网址: <http://itec.hust.edu.cn>



附录