哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场

《计算机网络》

第2章 应用层

主讲人：李全龙

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

你使用过哪些网络应用？


Baidu 百度
百度一下，你就知道！


微信
Weixin


網易 NETEASE
www·163·com
网聚人的力量


百度云


支付宝
Alipay.com


淘宝网
Taobao.com

网络应用有哪些特点？

与单机应用有哪些本质性的不同？

网络应用应采取什么样的体系结构？

网络应用的体系结构

- 客户机/服务器结构(Client-Server, C/S)
- 点对点结构(Peer-to-peer, P2P)
- 混合结构(Hybrid)

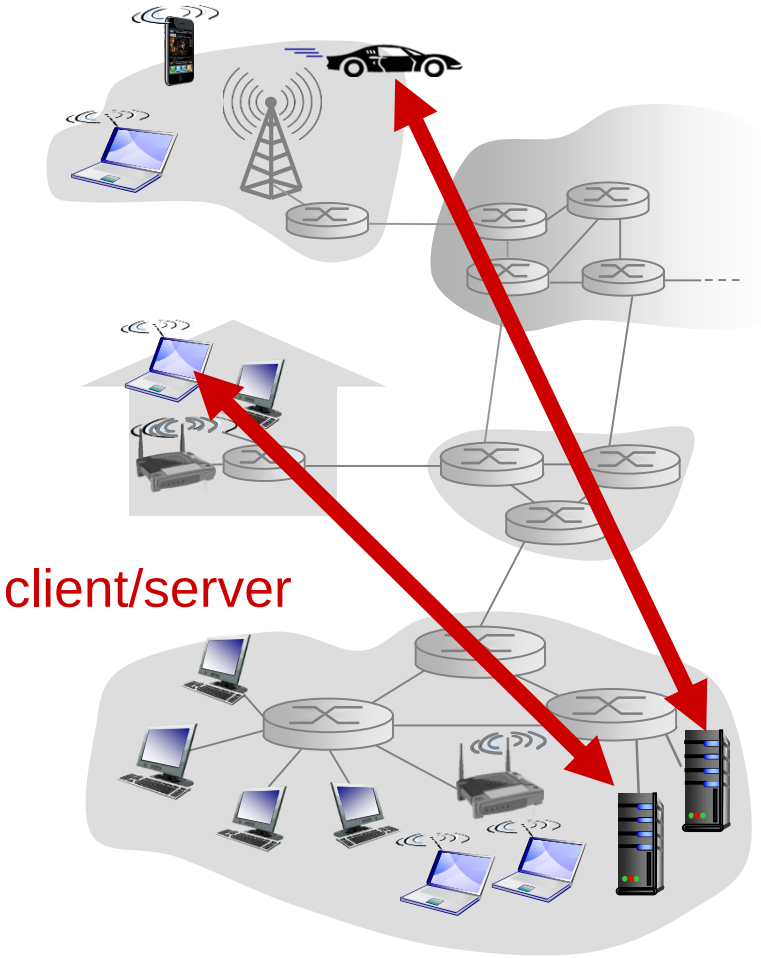
客户机/服务器结构

服务器（server）

- 7*24小时提供服务
- 永久性可访问地址/域名
- 利用大量服务器实现可扩展性

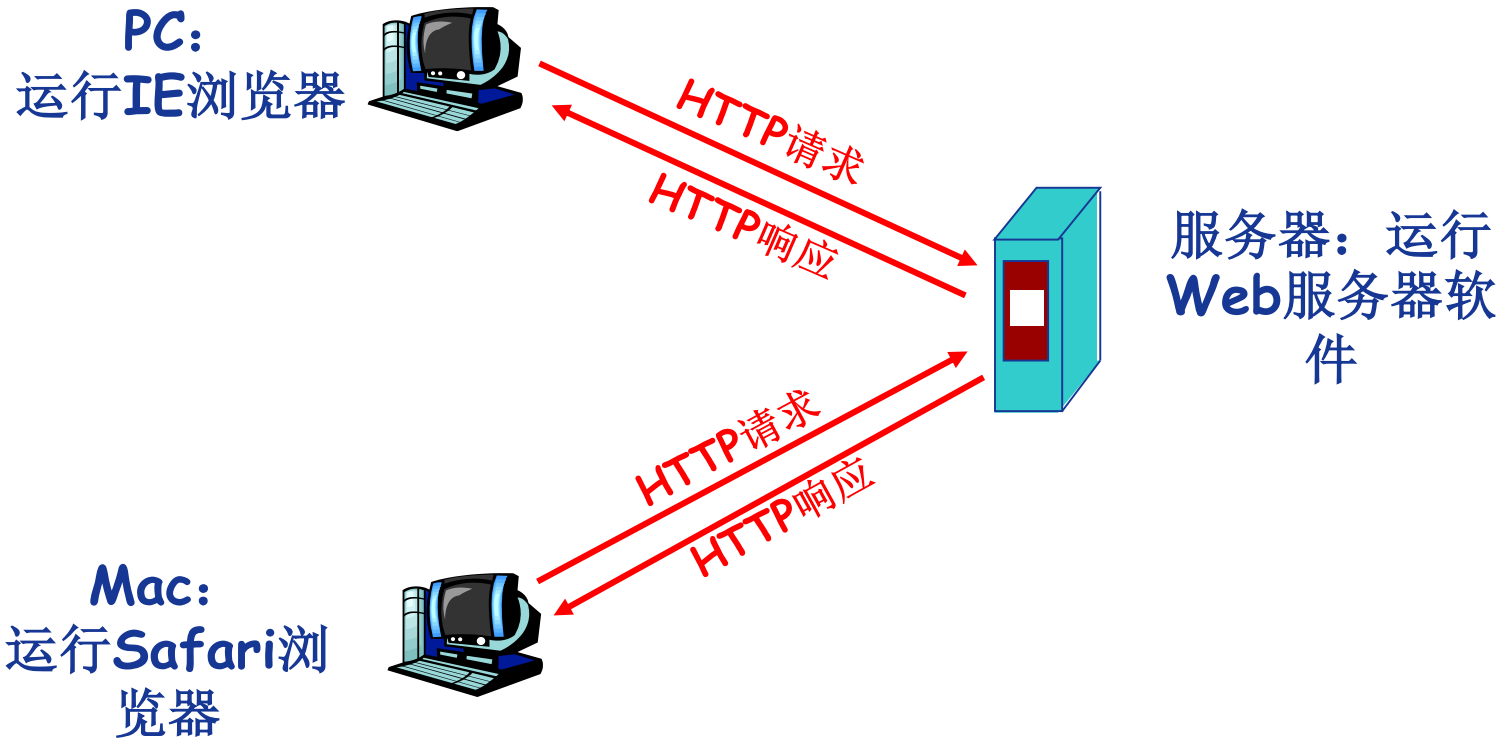
客户机（client）

- 与服务器通信，使用服务器提供的服务
- 间歇性接入网络
- 可能使用动态IP地址
- 不会与其他客户机直接通信



客户机/服务器结构

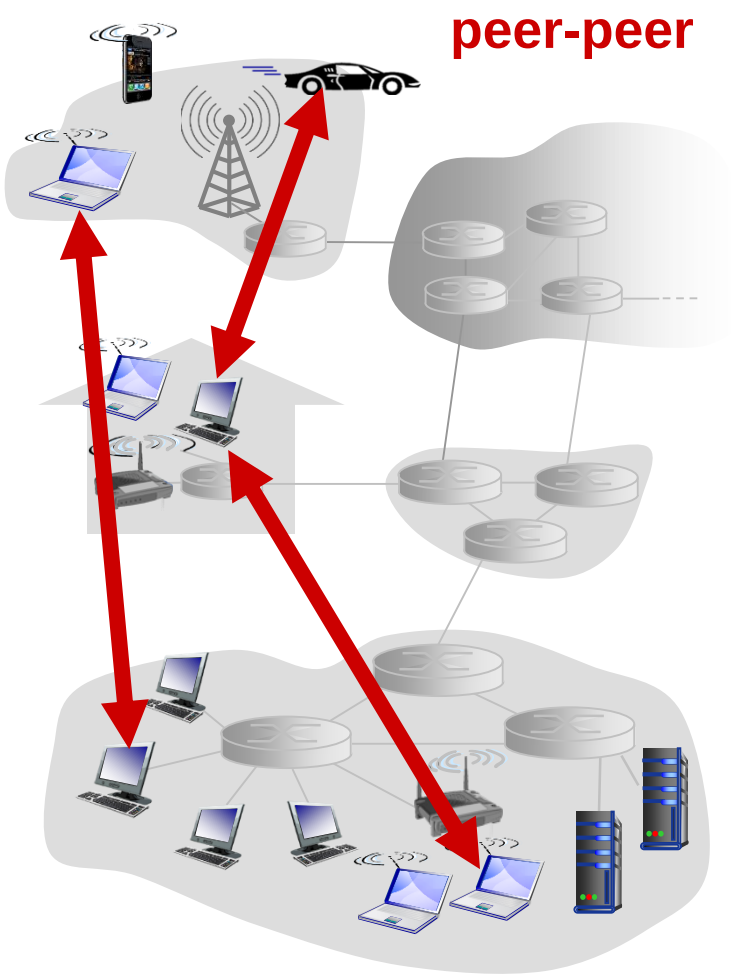
- 例子：Web



纯P2P结构

- 没有永远在线的服务器
- 任意端系统/节点之间可以直接通讯
- 节点间歇性接入网络
- 节点可能改变IP地址

- 优点：高度可伸缩
- 缺点：难于管理



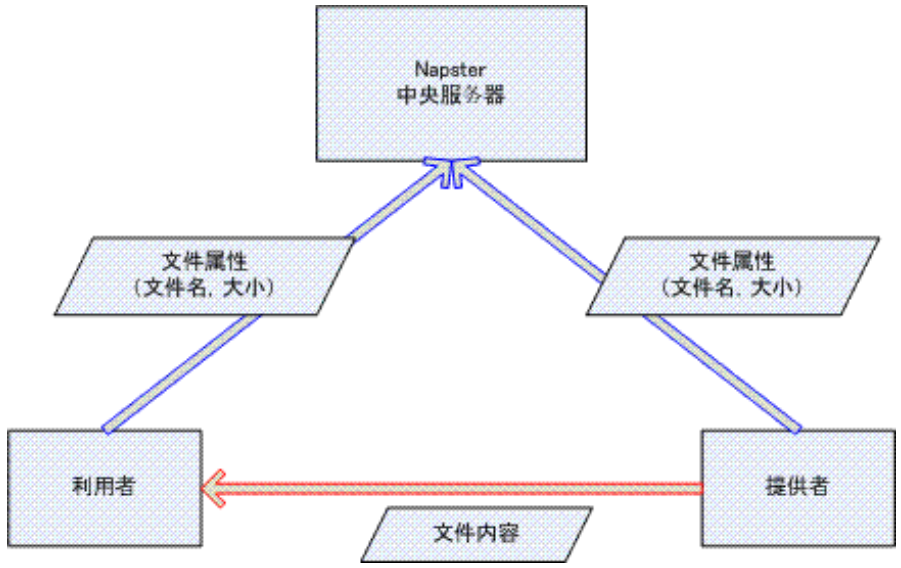
混合结构

能否将两种结构混合在一起使用？

混合能够利用两者的优点同时规避两者的缺点吗？

Napster

- 文件传输使用P2P结构
- 文件的搜索采用C/S结构——集中式
  - 每个节点向中央服务器登记自己的内容
  - 每个节点向中央服务器提交查询请求，查找感兴趣的内容



主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

网络应用的基础：进程间通信

- 进程：
  - 主机上运行的程序
- 同一主机上运行的进程之间如何通信？
  - 进程间通信机制
  - 操作系统提供
- 不同主机上运行的进程间如何通信？
  - 消息交换

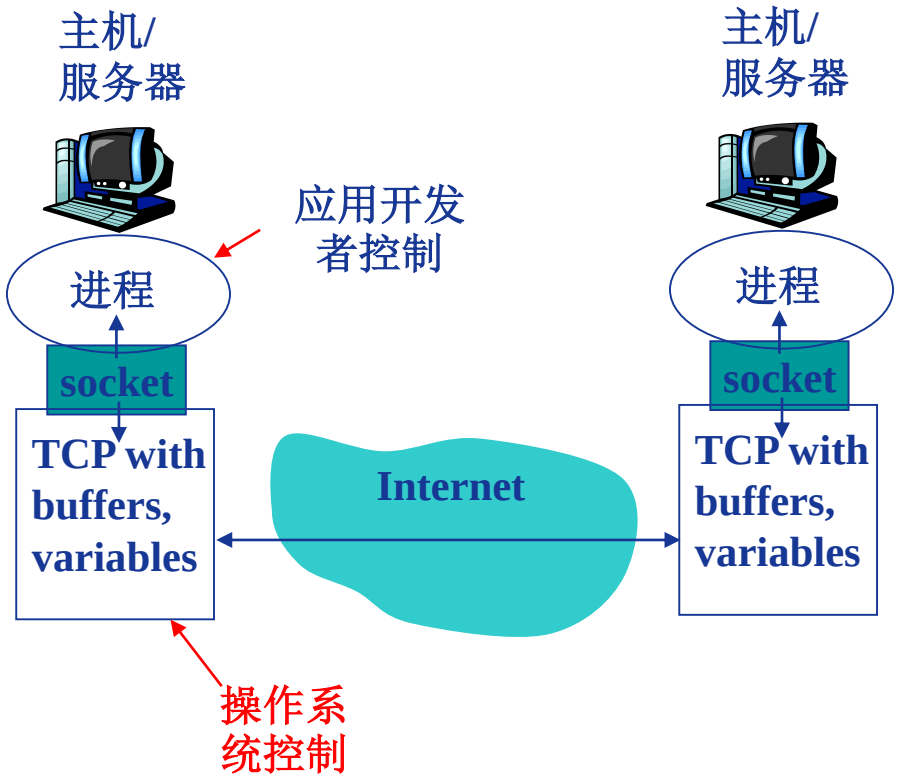
客户机进程：发起通信的进程

服务器进程：等待通信请求的进程

采用P2P架构的应用是否存在客户机进程/服务器进程之分？

套接字: Socket

- 进程间通信利用socket发送/接收消息实现
- 类似于寄信
  - 发送方将消息送到门外邮箱
  - 发送方依赖（门外的）传输基础设施将消息传到接收方所在主机，并送到接收方的门外
  - 接收方从门外获取消息
- 传输基础设施向进程提供API
  - 传输协议的选择
  - 参数的设置



如何寻址进程？

- 不同主机上的进程间通信，那么每个进程必须拥有标识符
- 如何寻址主机？——IP地址
  - Q：主机有了IP地址后，是否足以定位进程？
  - A：否。同一主机上可能同时有多个进程需要通信。
- 端口号/Port number
  - 为主机上每个需要通信的进程分配一个端口号
  - HTTP Server: 80
  - Mail Server：25
- 进程的标识符
  - IP地址+端口号

协议	本机IP地址: 端口号	外部IP地址: 端口号	状态
TCP	192.168.0.100:49225	202.108.23.105:5287	ESTABLISHED
TCP	192.168.0.100:49241	sinwns1011813:https	ESTABLISHED

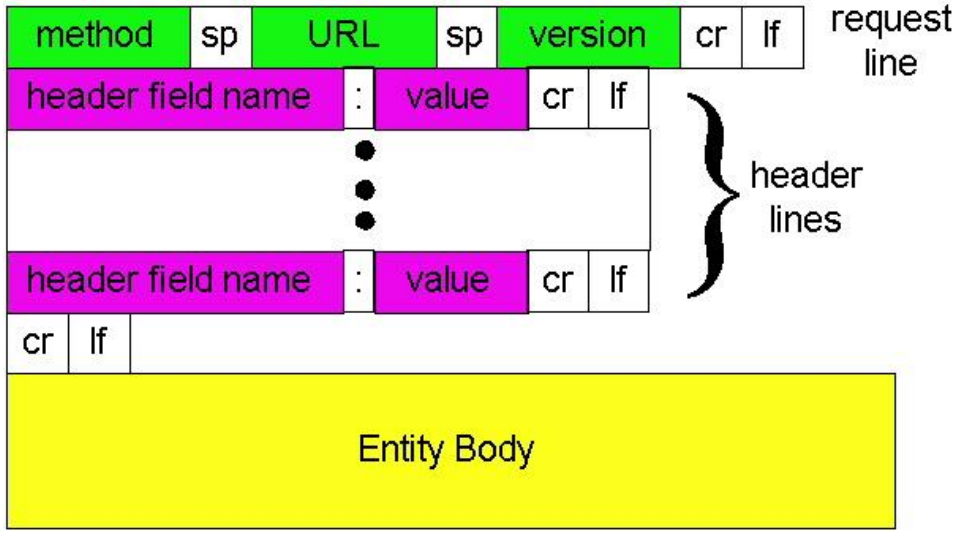
应用层协议

- 网络应用需遵循应用层协议
- 公开协议
  - 由RFC(Request For Comments)定义
  - 允许互操作
  - HTTP，SMTP，……
- 私有协议
  - 多数P2P文件共享应用

应用层协议的内容

- 消息的类型(type)
  - 请求消息
  - 响应消息
- 消息的语法(syntax)/格式
  - 消息中有哪些字段(field)？
  - 每个字段如何描述
- 字段的语义(semantics)
  - 字段中信息的含义
- 规则(rules)
  - 进程何时发送/响应消息
  - 进程如何发送/响应消息

HTTP请求消息的格式



主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

网络应用对传输服务的需求

- 数据丢失(data loss)/可靠性(reliability)
  - 某些网络应用能够容忍一定的数据丢失：网络电话
  - 某些网络应用要求100%可靠的数据传输：文件传输，telnet
- 时间(timing)/延迟(delay)
  - 有些应用只有在延迟足够低时才“有效”
  - 网络电话/网络游戏
- 带宽(bandwidth)
  - 某些应用只有在带宽达到最低要求时才“有效”：网络视频
  - 某些应用能够适应任何带宽——弹性应用：email

典型网络应用对传输服务的需求

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet提供的传输服务

TCP服务

- **面向连接**: 客户机/服务器进程间需要建立连接
- **可靠的传输**
- **流量控制**: 发送方不会发送速度过快，超过接收方的处理能力
- **拥塞控制**: 当网络负载过重时能够限制发送方的发送速度
- **不提供时间/延迟保障**
- **不提供最小带宽保障**

UDP服务

- **无连接**
- **不可靠的数据传输**
- 不提供：
  - 可靠性保障
  - 流量控制
  - 拥塞控制
  - 延迟保障
  - 带宽保障

典型网络应用所使用的传输层服务

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Vonage,Dialpad)	typically UDP

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- **HTTP协议**
- **HTTP报文**
- **Cookie技术**
- **Web缓存/代理**

Email应用

- **SMTP协议**
- **Email消息格式/RFC 822**
- **邮件接收协议**

FTP

DNS

- **域名结构**
- **域名服务器**
- **域名解析过程**
- **资源记录RR**

P2P应用

- **P2P文件分发**
- **P2P索引技术**

网络应用开发

- **Socket API**
- **Socket编程**

Web与HTTP

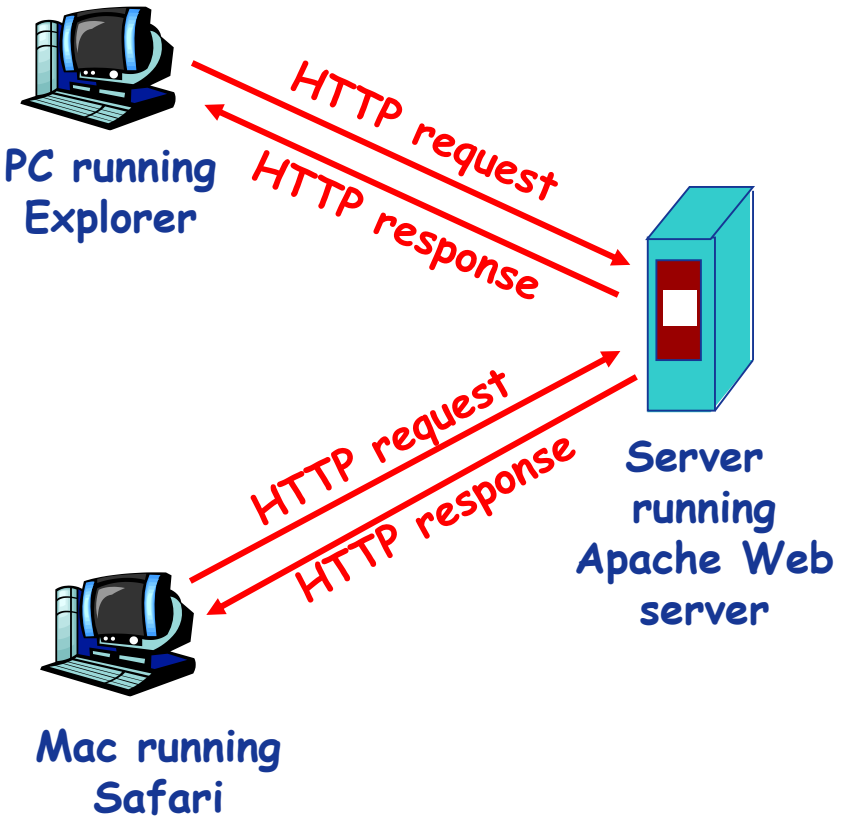
- World Wide Web: Tim Berners-Lee
  - 网页
  - 网页互相链接
- 网页(Web Page)包含多个对象(objects)
  - 对象：HTML文件、JPEG图片、视频文件、动态脚本等
  - 基本HTML文件：包含对其他对象引用的链接
- 对象的寻址(addressing)
  - URL(Uniform Resoure Locator)：统一资源定位器　RFC1738
  - Scheme://host:port/path

www.someschool.edu/someDept/pic.gif

host name　　path name

HTTP协议概述(1)

- 万维网应用遵循什么协议？
- 超文本传输协议
  - HyperText Transfer Protocol
- C/S结构
  - 客户—Browser：请求、接收、展示Web对象
  - 服务器—Web Server：响应客户的请求，发送对象
- HTTP版本：
  - 1.0： RFC 1945
  - 1.1： RFC 2068



HTTP概述(2)

- 使用TCP传输服务
  - 服务器在80端口等待客户的请求
  - 浏览器发起到服务器的TCP连接(创建套接字Socket)
  - 服务器接受来自浏览器的TCP连接
  - 浏览器(HTTP客户端)与Web服务器(HTTP服务器)交换HTTP消息
  - 关闭TCP连接

有状态的协议更复杂:

- 需维护状态(历史信息)
- 如果客户或服务器失效，会产生状态的不一致，解决这种不一致代价高

- 无状态(stateless)
  - 服务器不维护任何有关客户端过去所发请求的信息

HTTP连接的两种类型

- 非持久性连接(Nonpersistent HTTP)
  - 每个TCP连接最多允许传输一个对象
  - HTTP 1.0版本使用非持久性连接
- 持久性连接(Persistent HTTP)
  - 每个TCP连接允许传输多个对象
  - HTTP 1.1版本默认使用持久性连接

非持久性连接(1)

假定用户在浏览器中输入URL

www.someSchool.edu/someDepartment/home.index

包含文本和指向10个jpeg图片的链接

1a. HTTP客户端向地址为www.someSchool.edu的服务器上的HTTP服务器进程(端口80) 发起TCP连接请求。

1b. HTTP服务器在端口80等待TCP连接请求，接受连接并通知客户端。

2. HTTP客户端将HTTP请求消息(包含URL地址)通过TCP连接的套接字发出，消息中所含的URL表明客户端需要对象someDepartment/home.index

3. HTTP服务器收到请求消息，解析，产生包含所需要对象的响应消息，并通过套接字发给客户端。

时间

非持久性连接(2)

4. HTTP服务器关闭TCP连接。

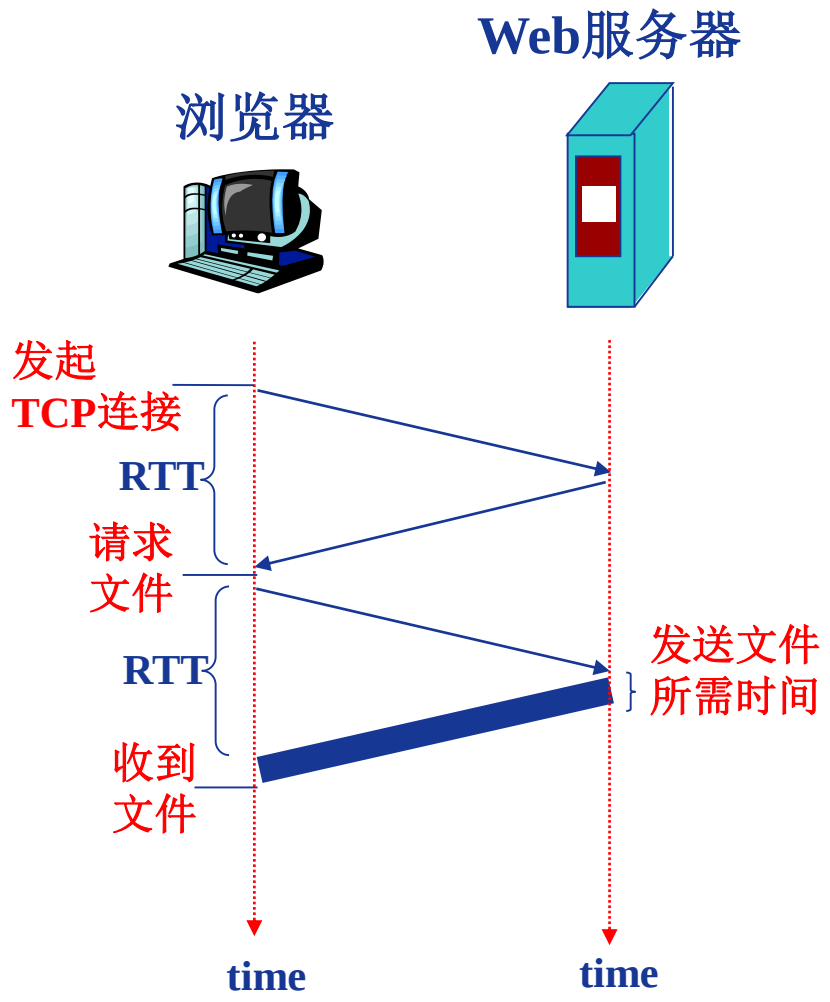
5. HTTP客户端收到响应消息，解析html文件，显示html文件，发现有10个指向jpeg对象的超连接。

时间

6. 为每个jpeg对象重复步骤1-5。

响应时间分析与建模

- RTT(Round Trip Time)
  - 从客户端发送一个很小的数据包到服务器并返回所经历的时间
- 响应时间(Response time)
  - 发起、建立TCP连接：1个RTT
  - 发送HTTP请求消息到HTTP响应消息的前几个字节到达：1个RTT
  - 响应消息中所含的文件/对象传输时间
  - **Total=2RTT +文件发送时间**



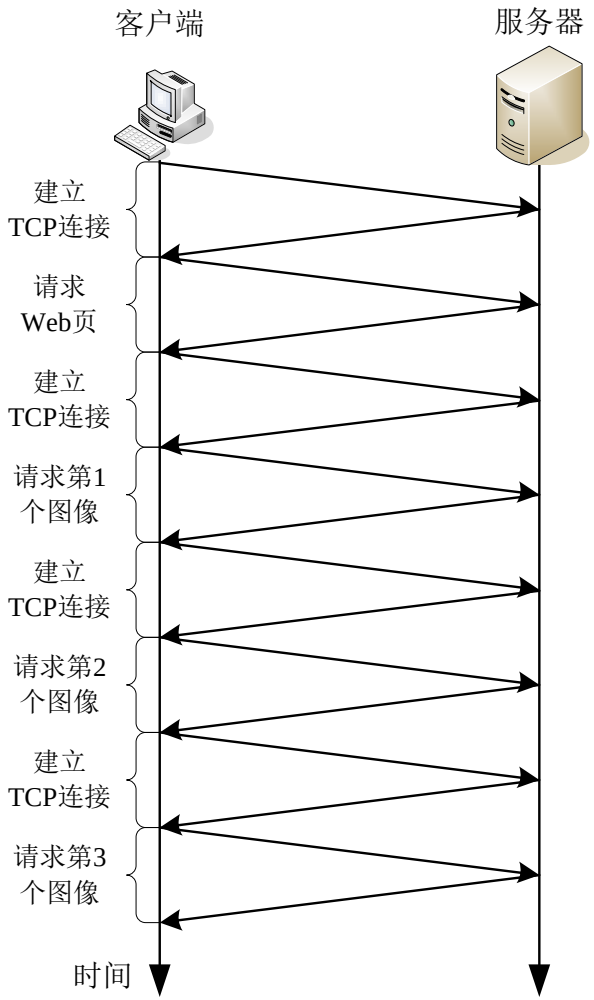
持久性HTTP

- 非持久性连接的问题
  - 每个对象需要2个RTT
  - 操作系统需要为每个TCP连接开销资源(overhead)
  - 浏览器会怎么做？
    - 打开多个并行的TCP连接以获取网页所需对象
    - 给服务器端造成什么影响？
- 持久性连接
  - 发送响应后，服务器保持TCP连接的打开
  - 后续的HTTP消息可以通过这个连接发送
- 无流水(pipelining)的持久性连接
  - 客户端只有收到前一个响应后才发送新的请求
  - 每个被引用的对象耗时1个RTT
- 带有流水机制的持久性连接
  - HTTP 1.1的默认选项
  - 客户端只要遇到一个引用对象就尽快发出请求
  - 理想情况下，收到所有的引用对象只需耗时约1个RTT

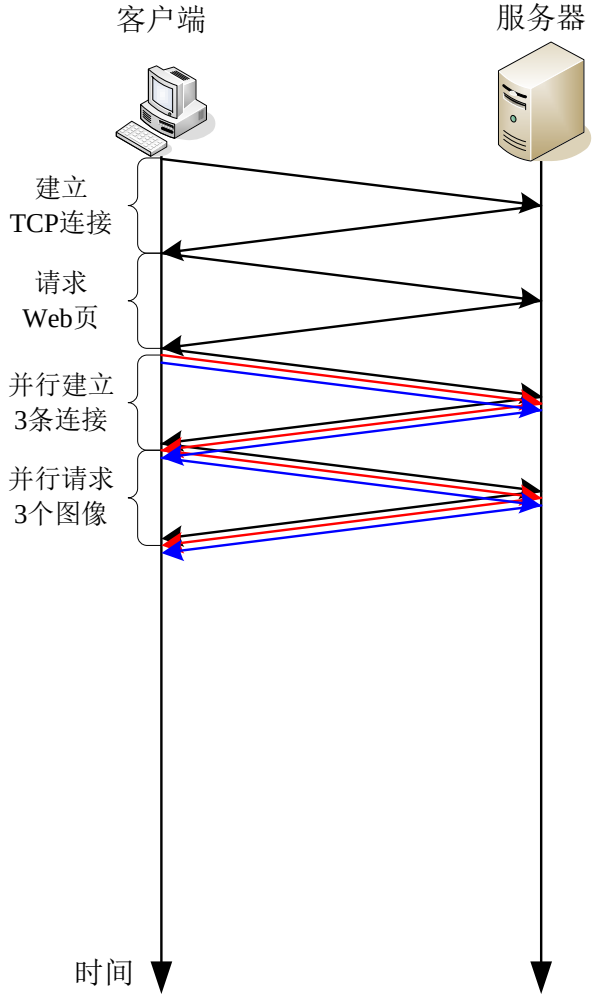
例题1

如果请求浏览一个引用3个JPEG小图像的Web页，则非持久连接（默认）的HTTP/1.0、使用并行连接的HTTP/1.0、非管道化持久连接的HTTP/1.1和管道化持久连接的HTTP/1.1分别需要多长时间？

HTTP/1.0

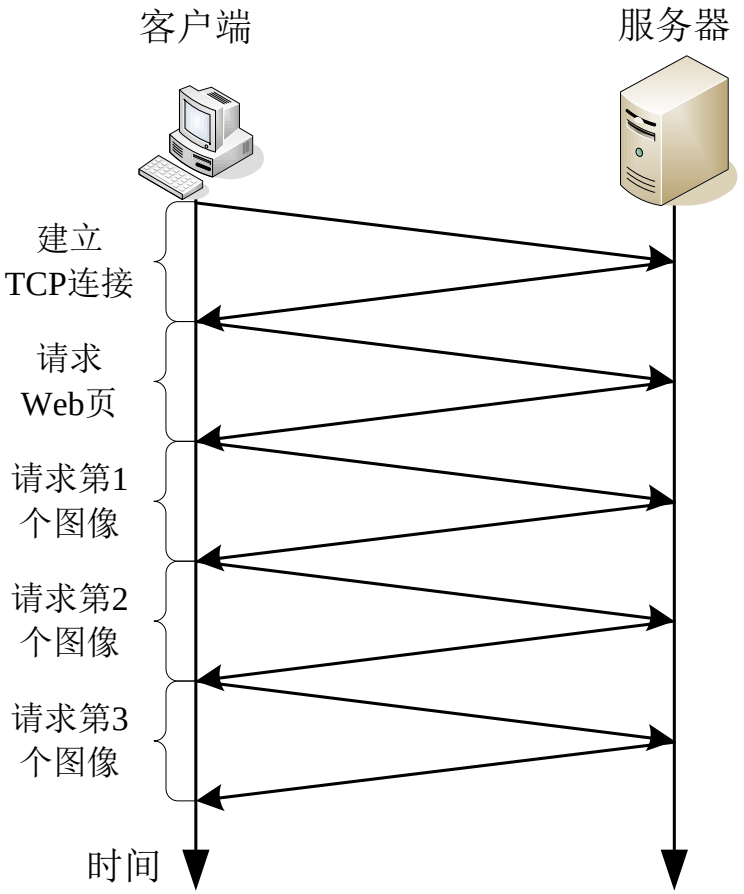


非持久连接（默认）的HTTP/1.0

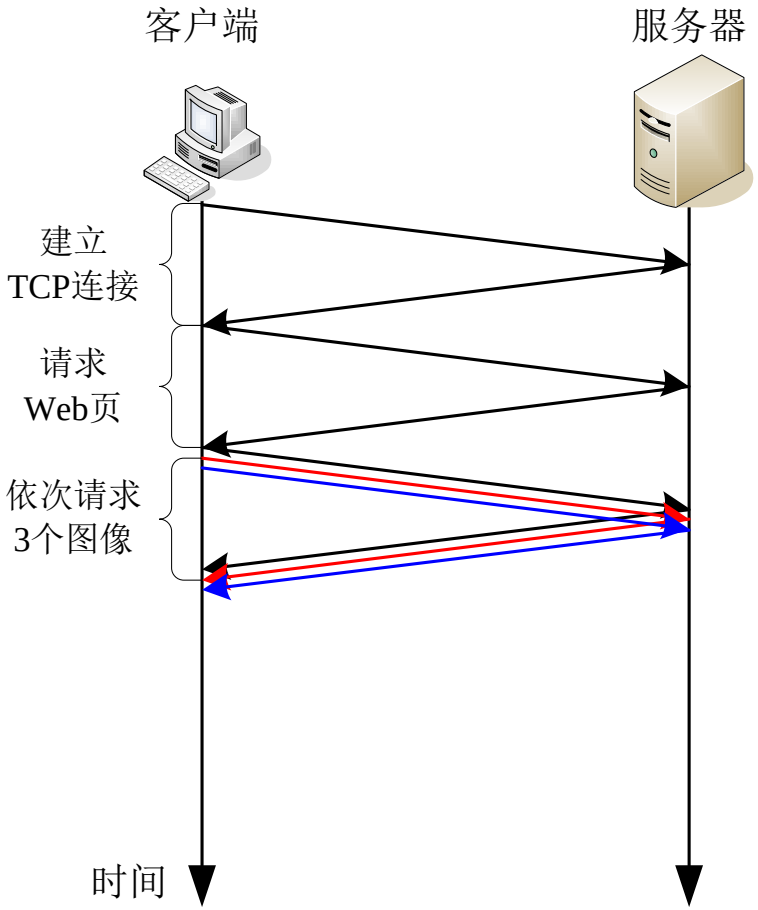


使用并行连接的HTTP/1.0

HTTP/1.1



非管道化持久连接的HTTP/1.1



管道化持久连接的HTTP/1.1

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

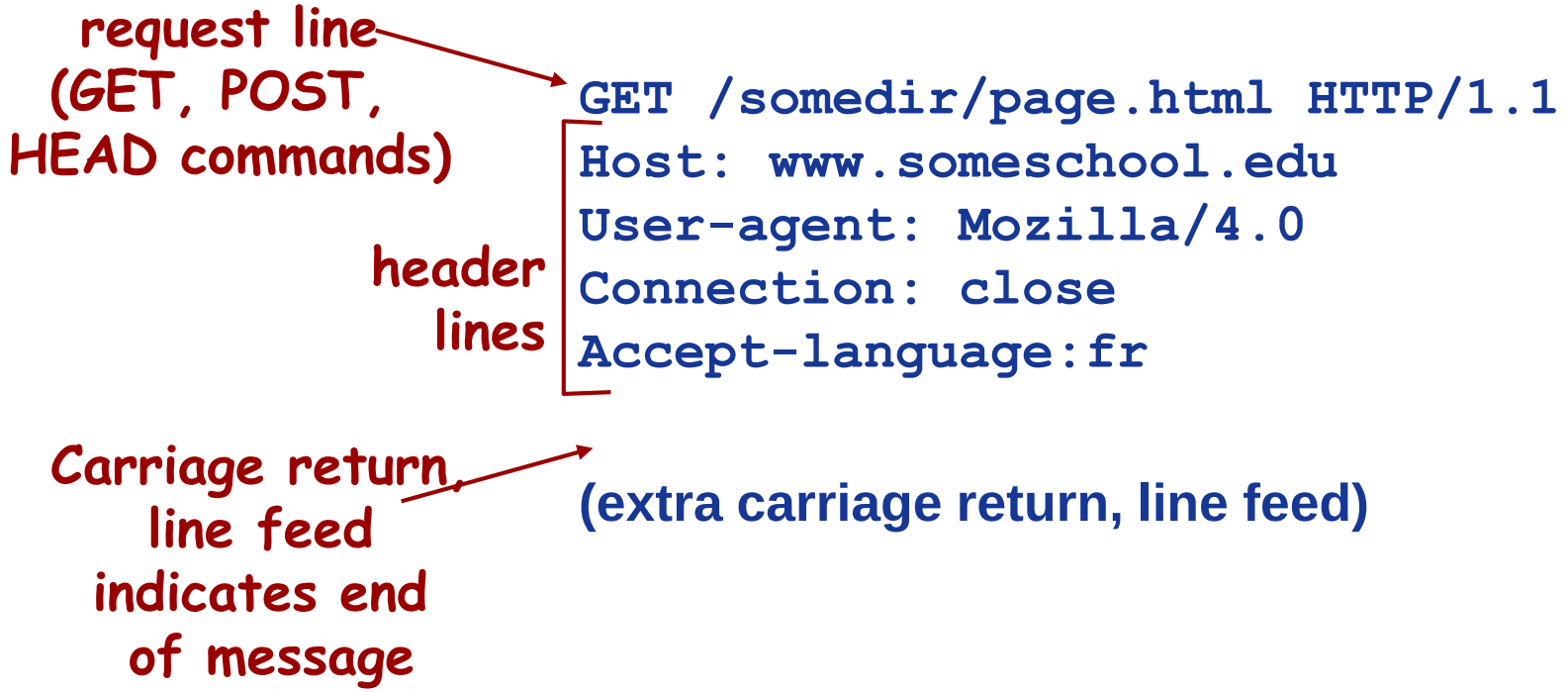
- P2P文件分发
- P2P索引技术

网络应用开发

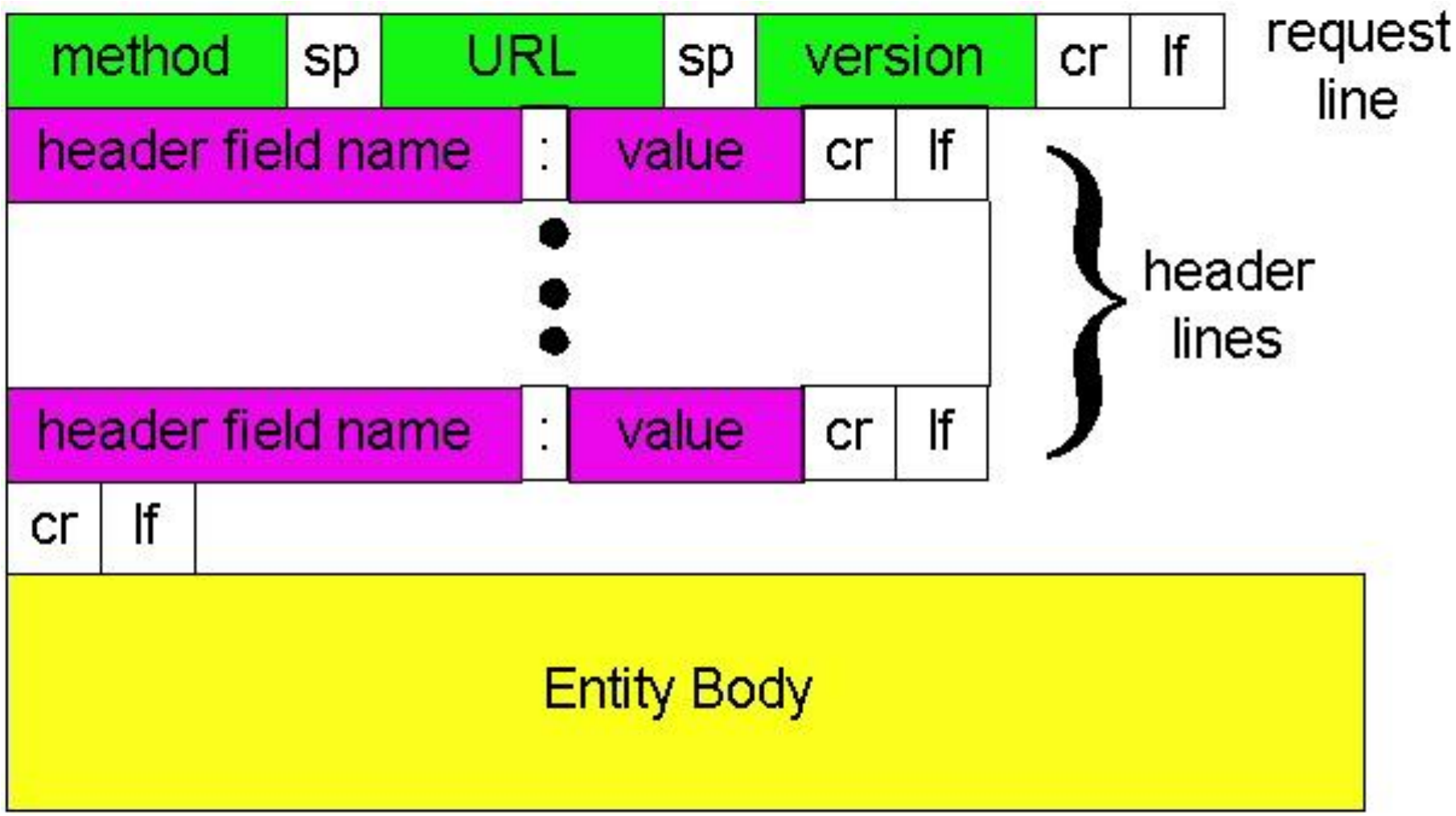
- Socket API
- Socket编程

HTTP请求消息

- HTTP协议有两类消息
  - 请求消息(request)
  - 响应消息(response)
- 请求消息
  - ASCII：人直接可读



HTTP请求消息的通用格式



上传输入的方法

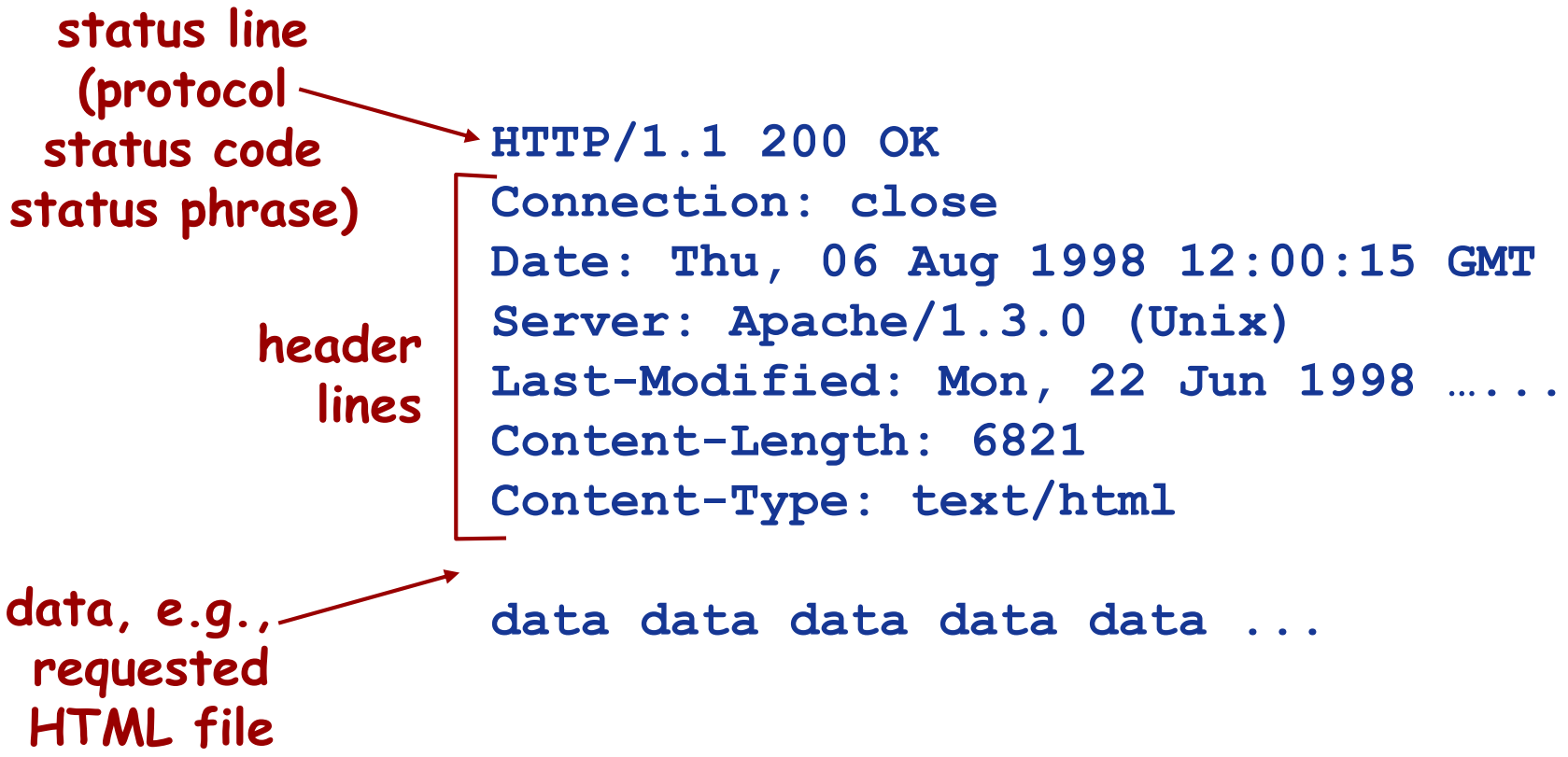
- POST方法
  - 网页经常需要填写表格(form)
  - 在请求消息的消息体(entity body)中上传客户端的输入
- URL方法
  - 使用GET方法
  - 输入信息通过request行的URL字段上传

```
www.somesite.com/animalsearch?monkeys&banana
```

方法的类型

- HTTP/1.0
  - GET
  - POST
  - HEAD
    - 请Server不要将所请求的对象放入响应消息中
- HTTP/1.1
  - GET, POST, HEAD
  - PUT
    - 将消息体中的文件上传到URL字段所指定的路径
  - DELETE
    - 删除URL字段所指定的文件

HTTP响应消息


status line
(protocol
status code
status phrase)
header
lines
data, e.g.,
requested
HTML file
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
data data data data data ...

HTTP响应状态代码

- 响应消息的第一行

- 示例
  - 200 OK
  - 301 Moved Permanently
  - 400 Bad Request
  - 404 Not Found
  - 505 HTTP Version Not Supported

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

为什么需要Cookie？

HTTP协议无状态

很多应用需要服务器掌握客户端的状态，如网上购物，如何实现？

Cookie技术

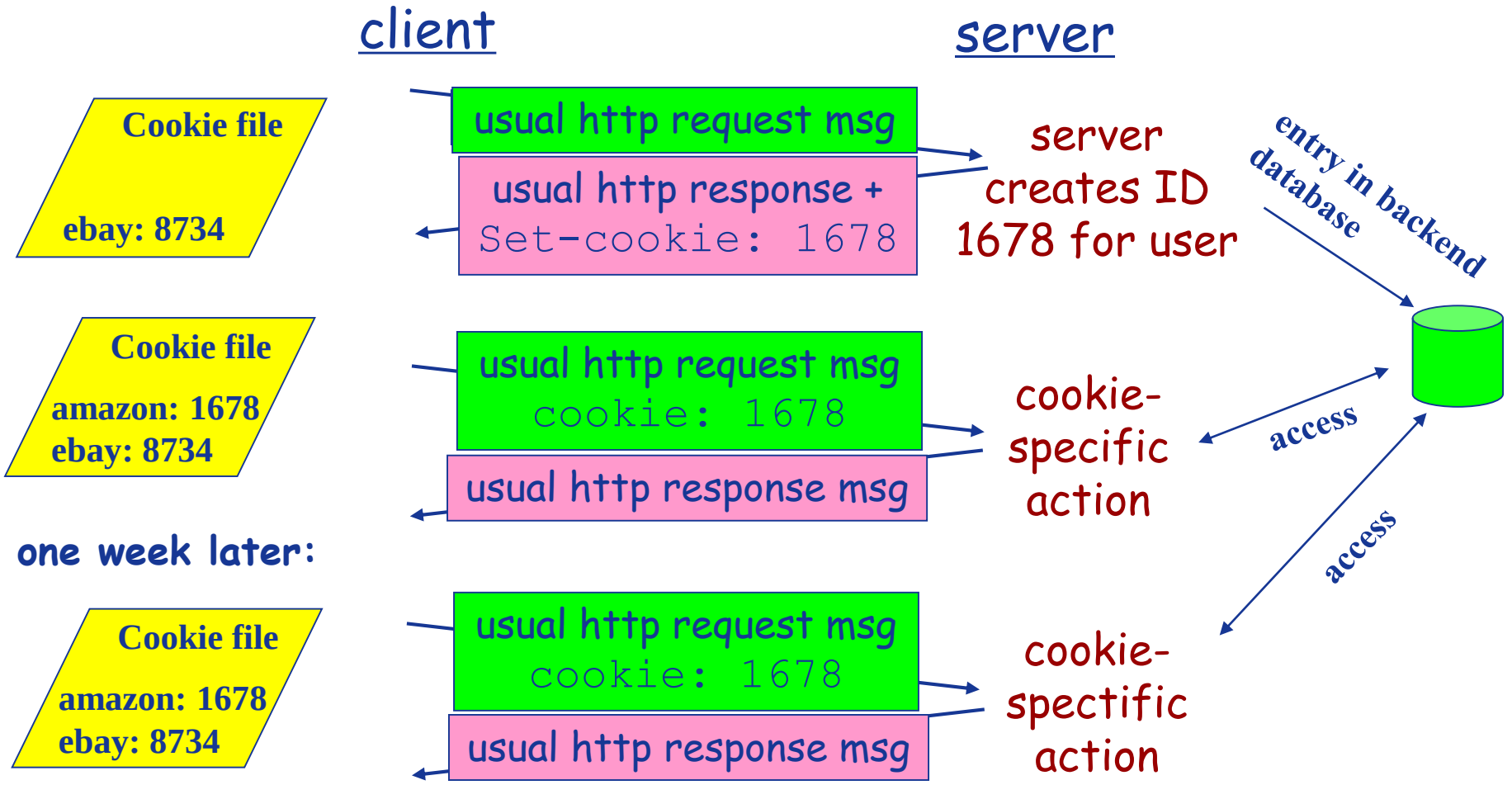
Cookie技术

- 某些网站为了辨别用户身份、进行session跟踪而储存在用户本地终端上的数据（通常经过加密）。
- RFC6265

Cookie的组件

- HTTP响应消息的cookie头部行
- HTTP请求消息的cookie头部行
- 保存在客户端主机上的cookie文件，由浏览器管理
- Web服务器端的后台数据库

Cookie的原理


client
server
Cookie file
ebay: 8734
usual http request msg
usual http response +
Set-cookie: 1678
server
creates ID
1678 for user
entry in backend
database
Cookie file
amazon: 1678
ebay: 8734
usual http request msg
cookie: 1678
usual http response msg
cookie-
specific
action
access
one week later:
Cookie file
amazon: 1678
ebay: 8734
usual http request msg
cookie: 1678
usual http response msg
cookie-
spectific
action
access

Cookie的作用

- Cookie能够用于：
  - 身份认证
  - 购物车
  - 推荐
  - Web e-mail
  - ……
- 隐私问题

cookie时代终结 苹果谷歌微软抢占新标准滩头

腾讯科技 [微博] 晨曦 2013年11月05日16:25 分享

[导读]科技巨头均已经开始研发或者采用替代cookie的技术，并把重点放在了移动端。

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

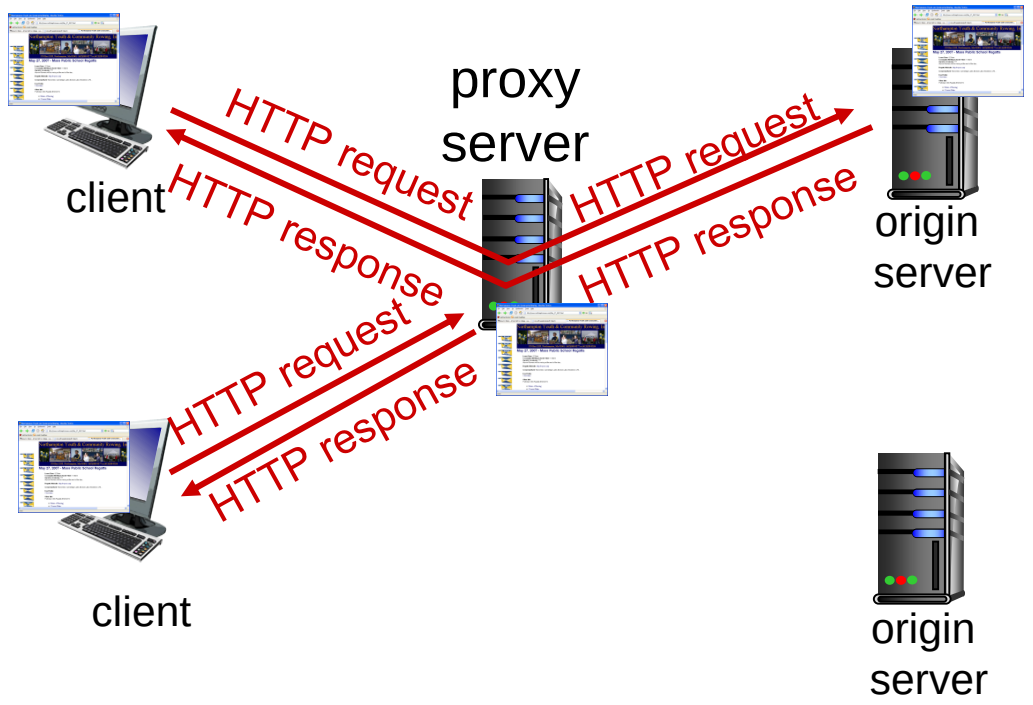
Web缓存/代理服务器技术

- 功能
  - 在不访问服务器的前提下满足客户端的HTTP请求。

- 为什么要发明这种技术？
  - 缩短客户请求的响应时间
  - 减少机构/组织的流量
  - 在大范围内(Internet)实现有效的内容分发

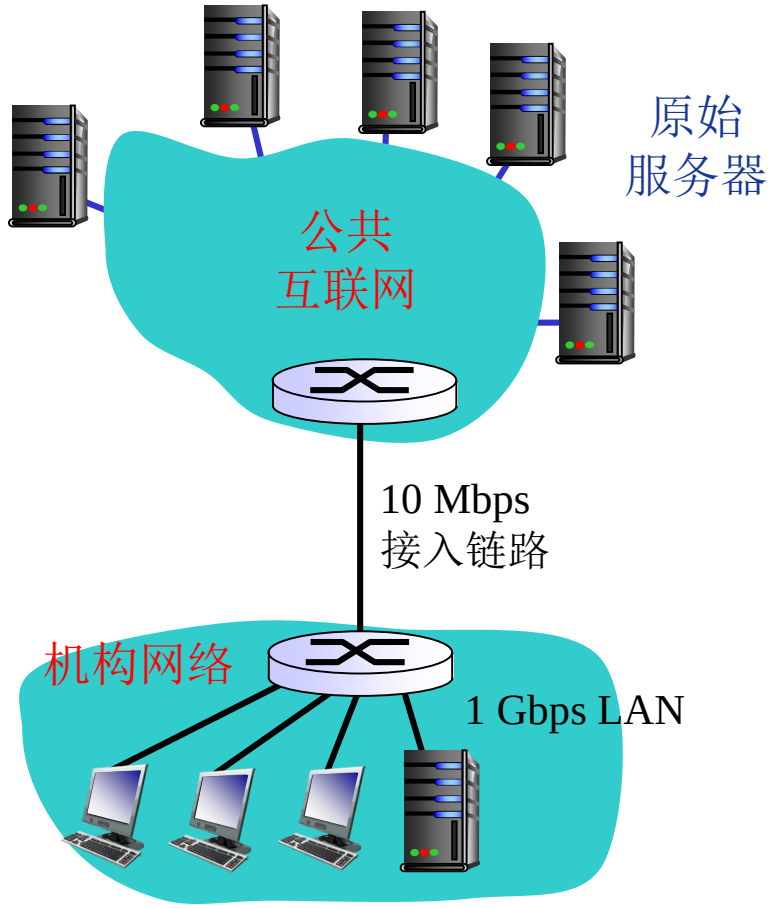
Web缓存/代理服务器技术

- Web缓存/代理服务器
  - 用户设定浏览器通过缓存进行Web访问
  - 浏览器向缓存/代理服务器发送所有的HTTP请求
    - 如果所请求对象在缓存中，缓存返回对象
    - 否则，缓存服务器向原始服务器发送HTTP请求，获取对象，然后返回给客户端并保存该对象
- 缓存既充当客户端，也充当服务器
- 一般由ISP(Internet服务提供商)架设



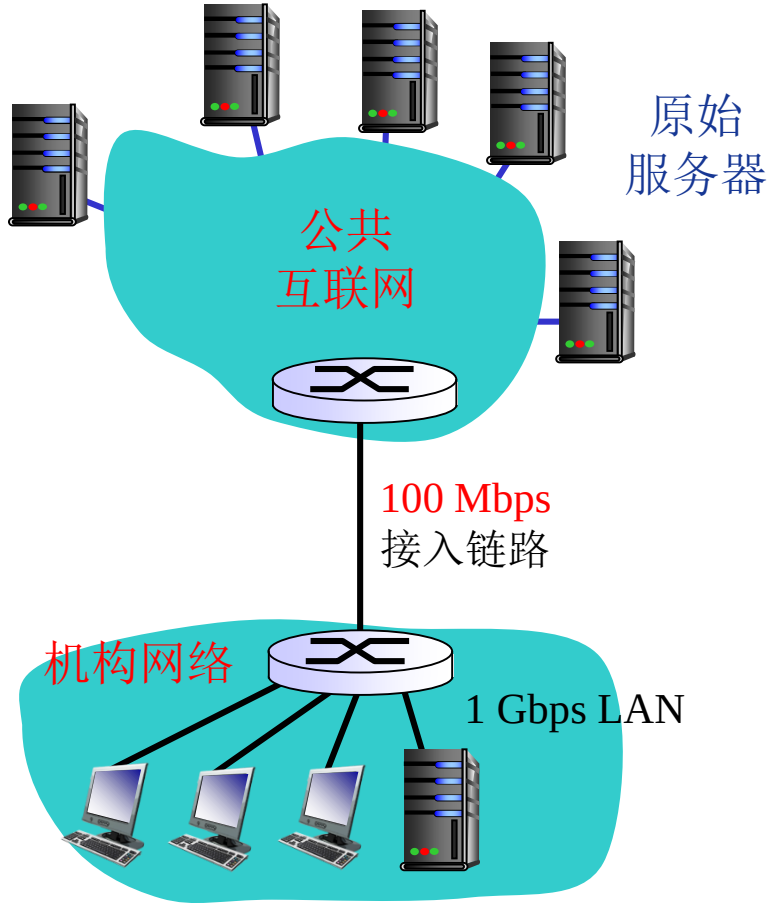
Web缓存示例(1)

- 假定：
  - 平均对象大小=1 Mbits
  - 机构网络中浏览器平均每秒有10个/秒到原始服务器的请求
  - 从机构路由器到原始服务器的往返延迟RTT=2秒
  - 接入链路带宽：10 Mbps
- 网络性能分析：
  - 局域网(LAN)的利用率=1%
  - 接入互联网的链路利用率=100%
  - 总的延迟=互联网上的延迟+访问延迟+局域网延迟=2秒+几分钟+几微秒



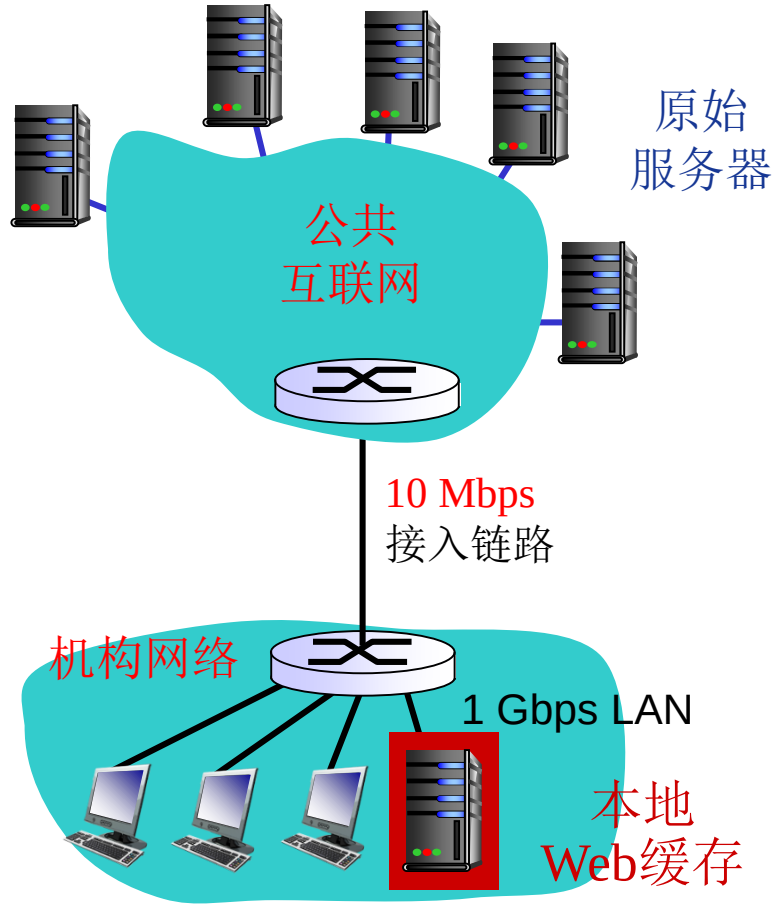
Web缓存示例(2)

- 解决方案1：
  - 提升互联网接入带宽=100Mbps
- 网络性能分析：
  - 局域网(LAN)的利用率=1%
  - 接入互联网的链路的利用率=10%
  - 总的延迟=互联网上的延迟+访问延迟+局域网延迟=2秒+几微秒+几微秒
- 问题：
  - 成本太高



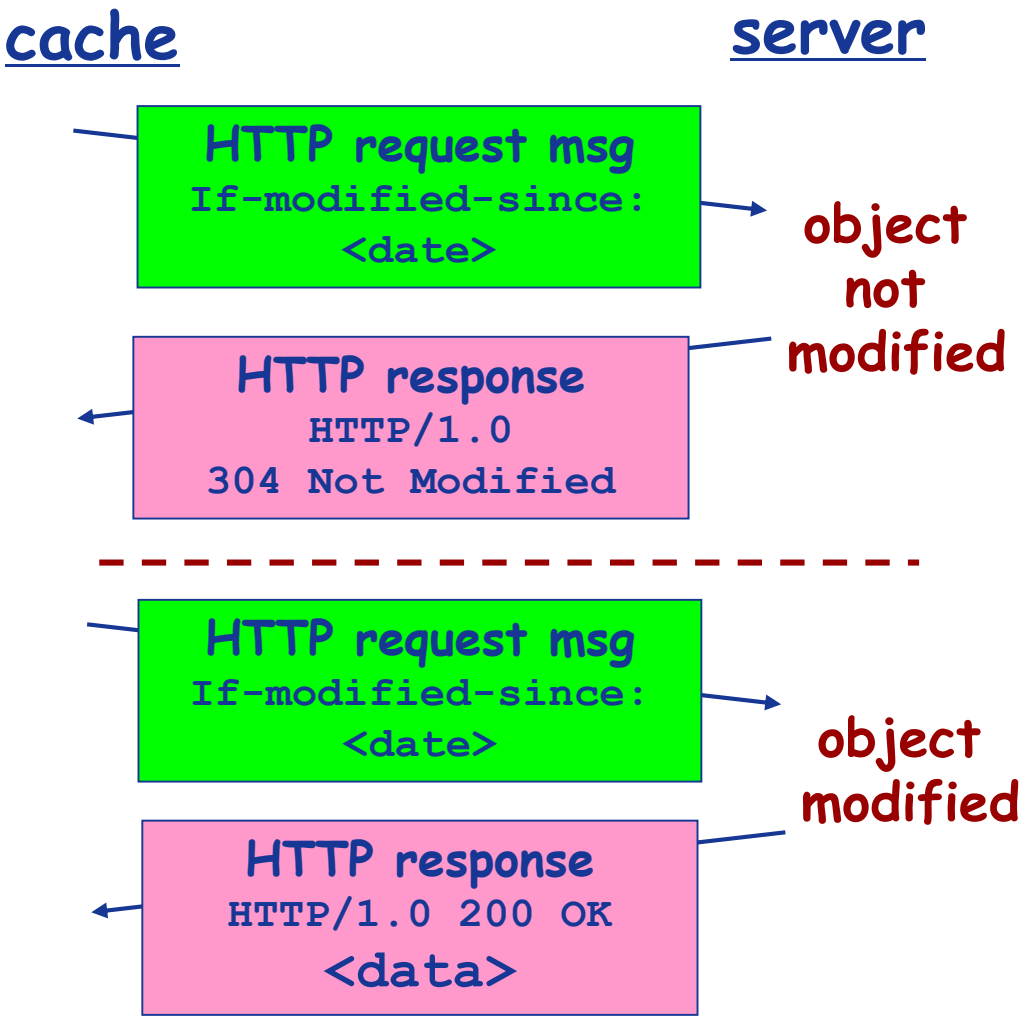
Web缓存示例(3)

- 解决方案2：
  - 部署本地Web缓存
  - 假定缓存命中率=0.4
- 网络性能分析：
  - 40%的请求立刻得到满足
  - 60%的请求通过原始服务器满足
  - 接入链路的利用率下降到60%，从而其延迟可以忽略不计，例如10微秒
  - 总的平均延迟=互联网上的延迟+访问延迟+局域网延迟=0.6×2.0秒+0.4×n微秒≈1.2秒



条件性GET方法

- 目标：
  - 如果缓存有最新的版本，则不需要发送请求对象
- 缓存：
  - 在HTTP请求消息中声明所持有版本的日期
  - If-modified-since: <date>
- 服务器：
  - 如果缓存的版本是最新的，则响应消息中不包含对象
  - HTTP/1.0 304 Not Modified



主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

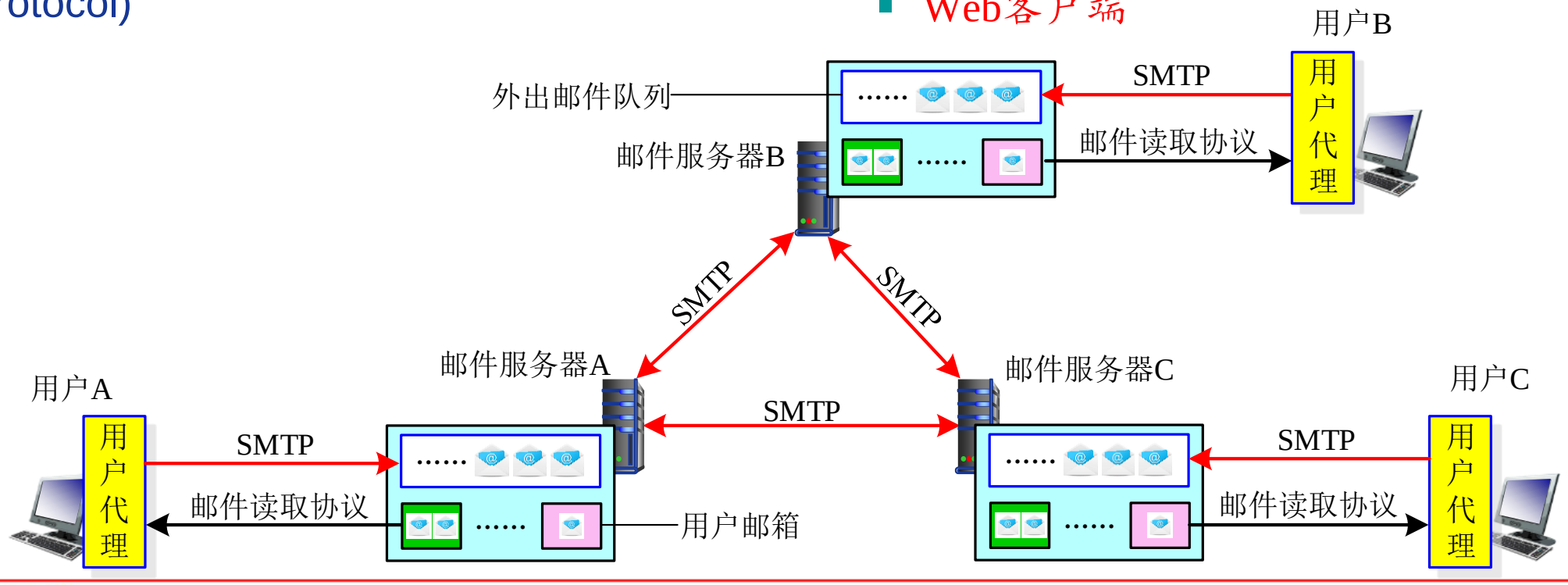
- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

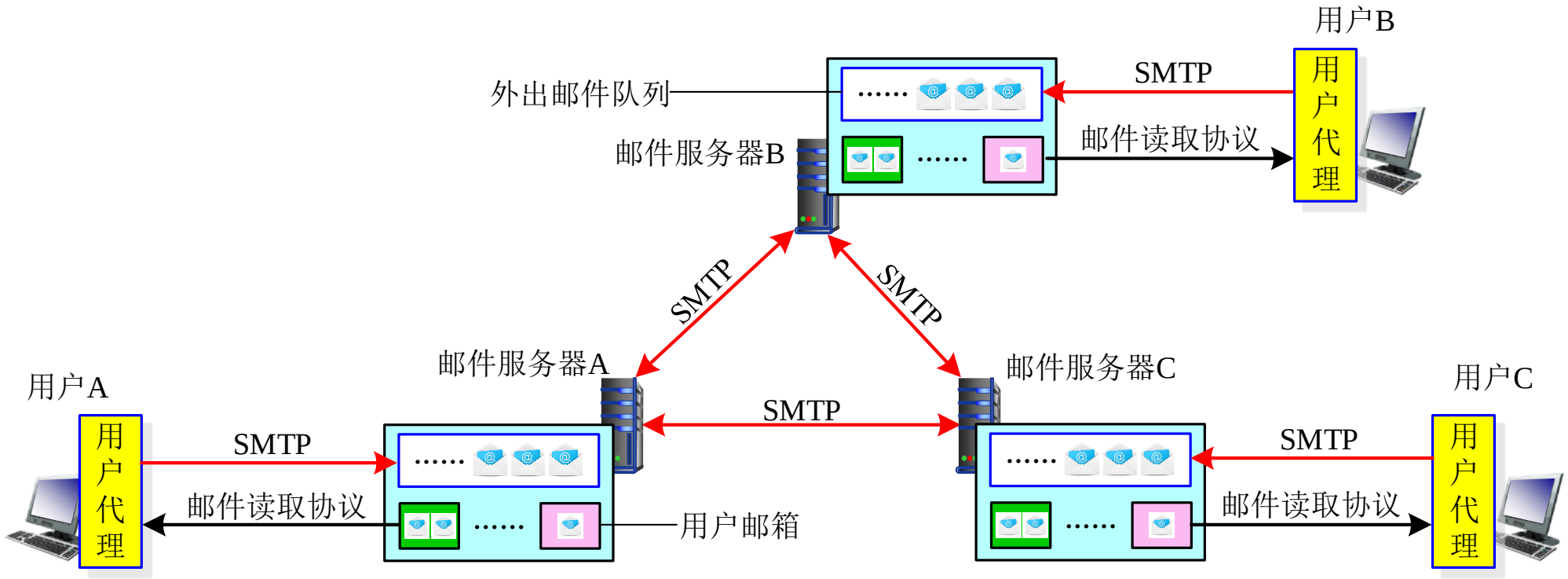
Email应用的构成(1)

- Email应用的构成组件
  - 邮件客户端(user agent)
  - 邮件服务器
  - SMTP协议(Simple Mail Transfer Protocol)
- 用户代理（邮件客户端）
  - 读、写Email消息
  - 与服务器交互，收、发Email消息
  - Outlook, Foxmail, Thunderbird
  - Web客户端



Email应用的构成(2)

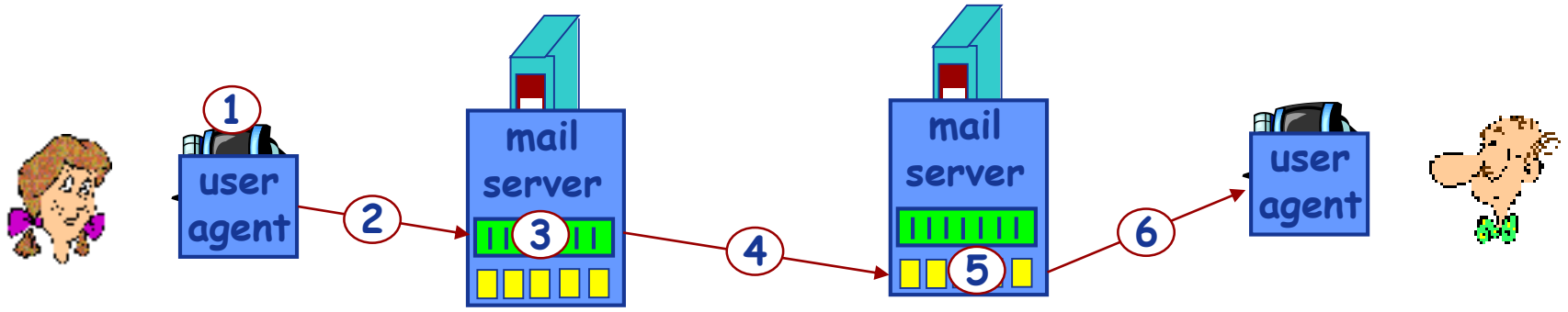
- 邮件服务器(Mail Server)
  - 邮箱：存储发给该用户的Email
  - 消息队列(message queue)：存储等待发送的Email
- SMTP协议
  - 邮件服务器之间传递消息所使用的协议
  - 客户端：发送消息的服务器
  - 服务器：接收消息的服务器



SMTP协议: RFC 2821

- 使用TCP进行email消息的可靠传输
- 端口25
- 传输过程的三个阶段
  - 握手
  - 消息的传输
  - 关闭
- 命令/响应交互模式
  - 命令(command): ASCII文本
  - 响应(response): 状态代码和语句
- Email消息只能包含7位ASCII码

Email应用示例



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
```

SMTP协议

- 使用持久性连接
- 要求消息必须由7位ASCII码构成
- SMTP服务器利用CRLF.CRLF确定消息的结束。

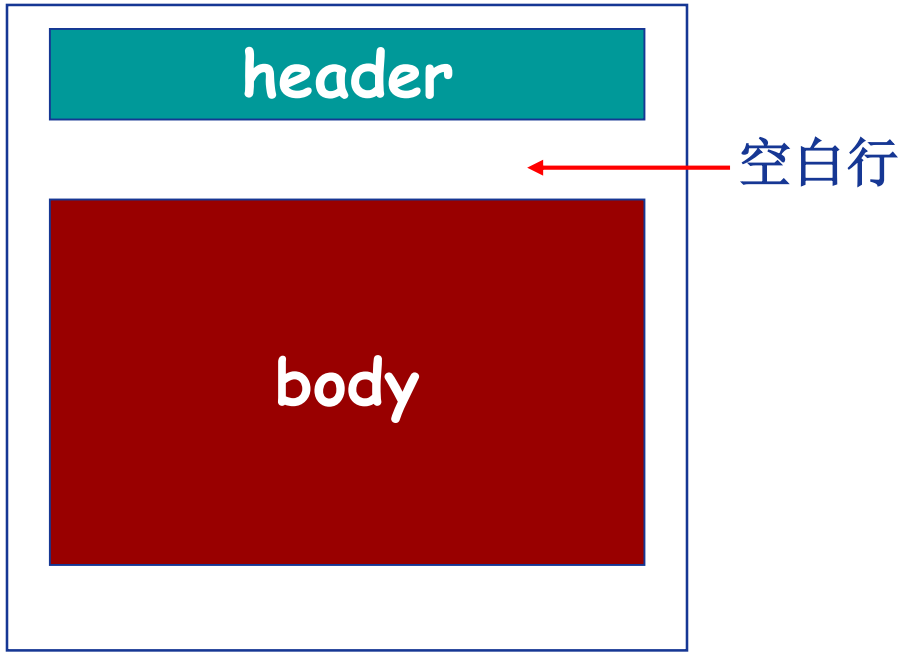
与HTTP对比:

- HTTP: 拉式(pull)
- SMTP: 退式(push)
- 都使用命令/响应交互模式
- 命令和状态代码都是ASCII码
- HTTP: 每个对象封装在独立的响应消息中
- SMTP: 多个对象在由多个部分构成的消息中发送

Email消息格式

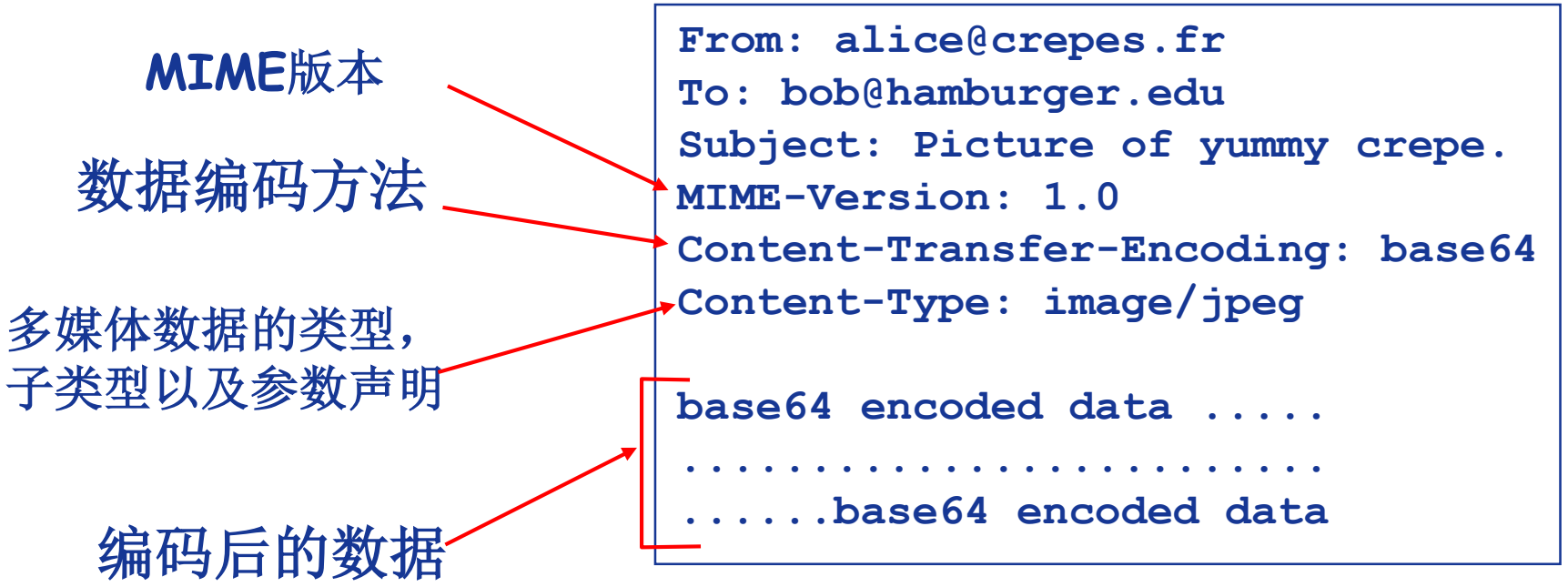
- SMTP：email消息的传输/交换协议
- RFC 822：文本消息格式标准
  - 头部行(header)
    - To
    - From
    - Subject

    与SMTP命令不同
  - 消息体(body)
    - 消息本身
    - 只能是ASCII字符

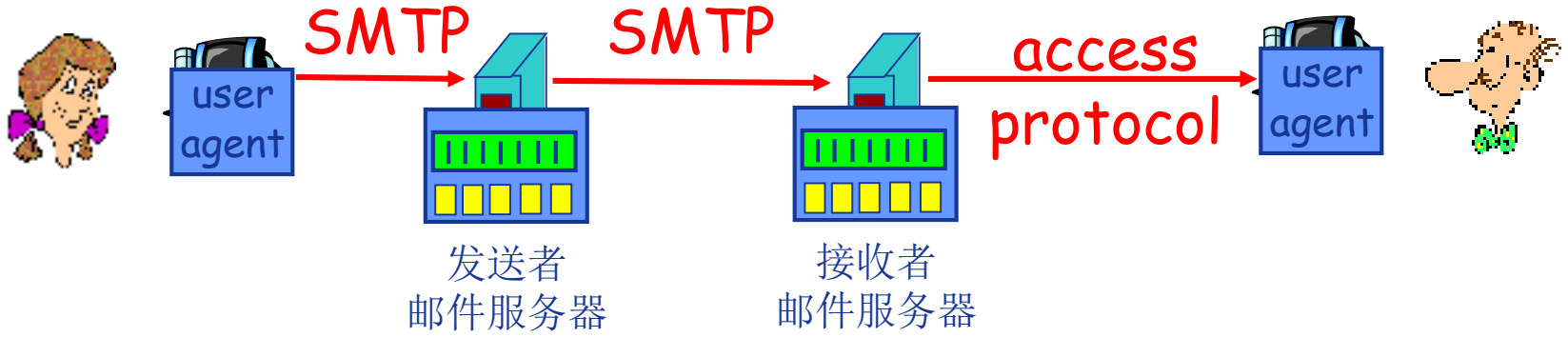


Email消息格式：多媒体扩展

- MIME：多媒体邮件扩展 RFC 2045, 2056
  - 通过在邮件头部增加额外的行以声明MIME的内容类型



邮件访问协议



- 邮件访问协议：从服务器获取邮件
  - POP: Post Office Protocol [RFC 1939]
    - 认证/授权(客户端←→服务器)和下载
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - 更多功能
    - 更加复杂
    - 能够操纵服务器上存储的消息
  - HTTP：163, QQ Mail等。

POP协议

- 认证过程
  - 客户端命令
    - User：声明用户名
    - Pass: 声明密码
  - 服务器响应
    - +OK
    - –ERR
- 事务阶段
  - List：列出消息数量
  - Retr：用编号获取消息
  - Dele: 删除消息
  - Quit

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

```
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
```

POP协议

- “下载并删除”模式
  - 用户如果换了客户端软件，无法重读该邮件
- “下载并保持”模式：不同客户端都可以保留消息的拷贝
- POP3是无状态的

IMAP协议

- 所有消息统一保存在一个地方：服务器
- 允许用户利用文件夹组织消息
- IMAP支持跨会话(Session)的用户状态:
  - 文件夹的名字
  - 文件夹与消息ID之间的映射等

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

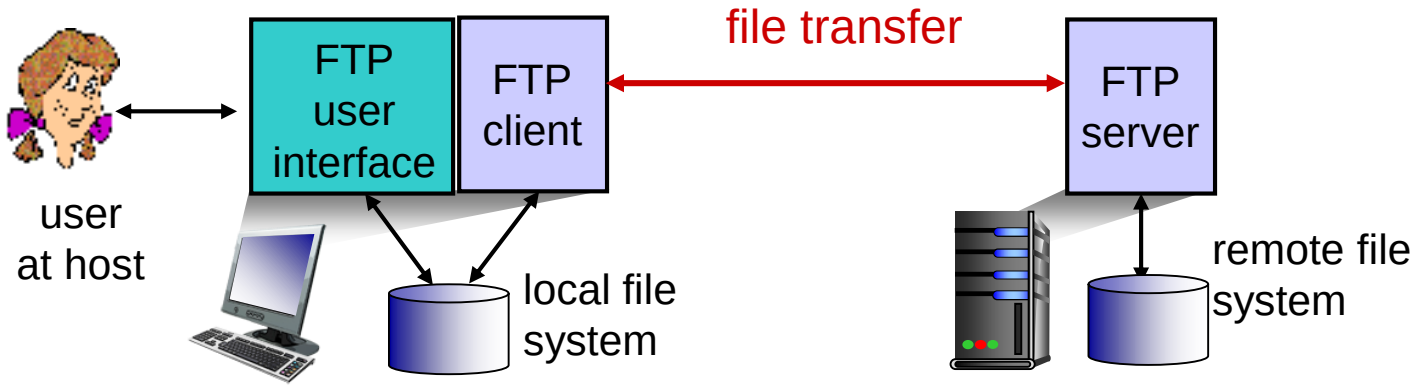
P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

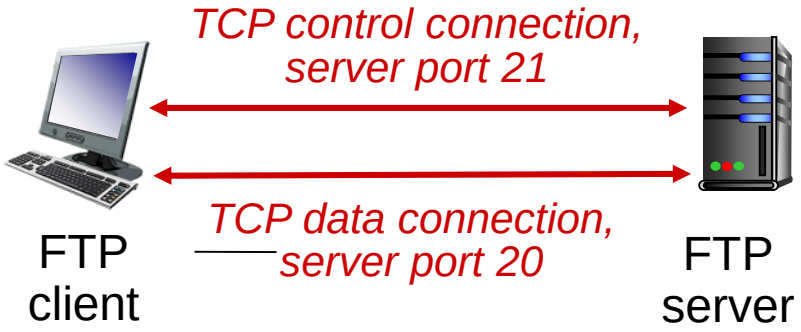
FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: separate control, data connections

- FTP client contacts FTP server at port 21, using TCP
- client authorized over control connection
- client browses remote directory, sends commands over control connection
- when server receives file transfer command, *server* opens *2nd* TCP data connection (for file) *to* client
- after transferring one file, server closes data connection



- server opens another TCP data connection to transfer another file
- control connection: *"out of band"*
- FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

sample commands:

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

- Socket API
- Socket编程

DNS：Domain Name System

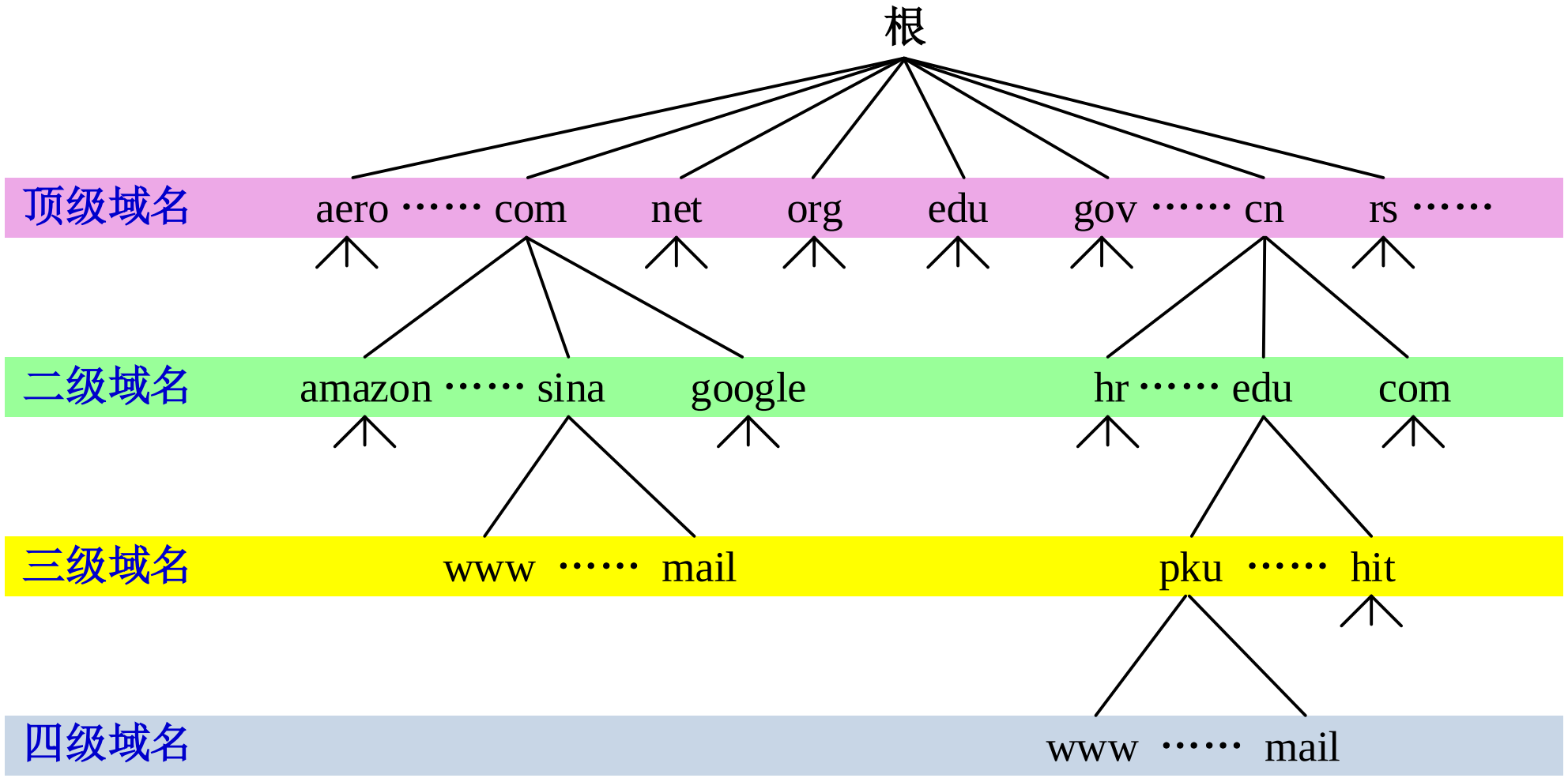
- Internet上主机/路由器的识别问题
  - IP地址
  - 域名：www.hit.edu.cn
- 问题：域名和IP地址之间如何映射？

- 域名解析系统DNS
  - 多层命名服务器构成的分布式数据库
  - 应用层协议：完成名字的解析
    - Internet核心功能，用应用层协议实现
    - 网络边界复杂

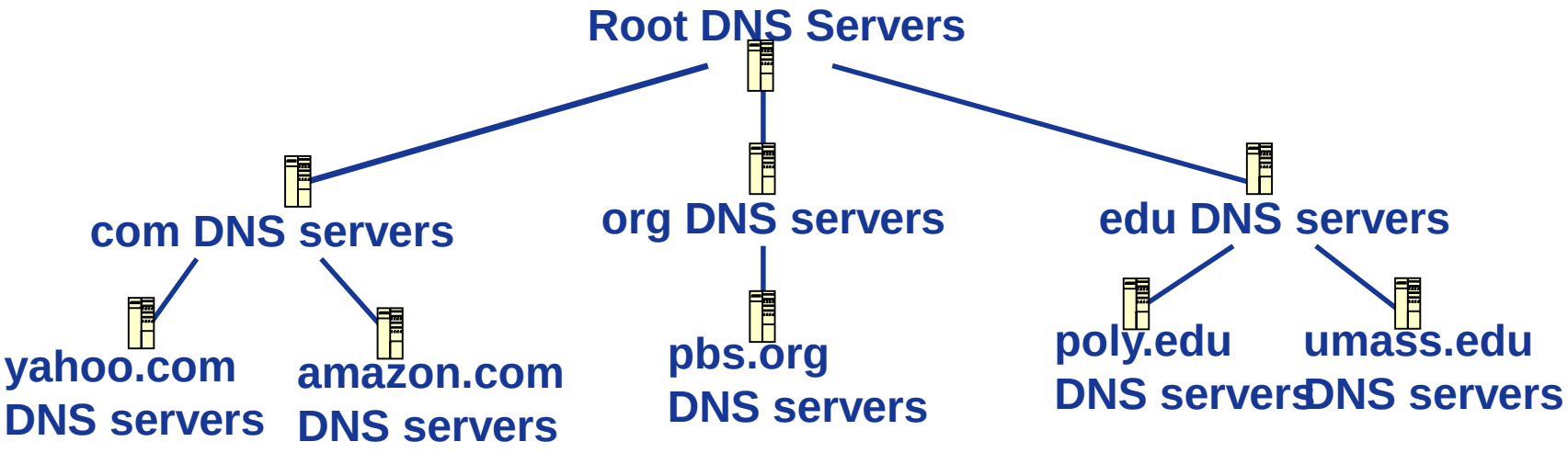
DNS

- DNS服务
  - 域名向IP地址的翻译
  - 主机别名
  - 邮件服务器别名
  - 负载均衡：Web服务器
- 问题：为什么不使用集中式的DNS？
  - 单点失败问题
  - 流量问题
  - 距离问题
  - 维护性问题

DNS—域名层次结构


根
顶级域名
aero …… com
net
org
edu
gov …… cn
rs ……
二级域名
amazon …… sina
google
hr …… edu
com
三级域名
www …… mail
pku …… hit
四级域名
www …… mail

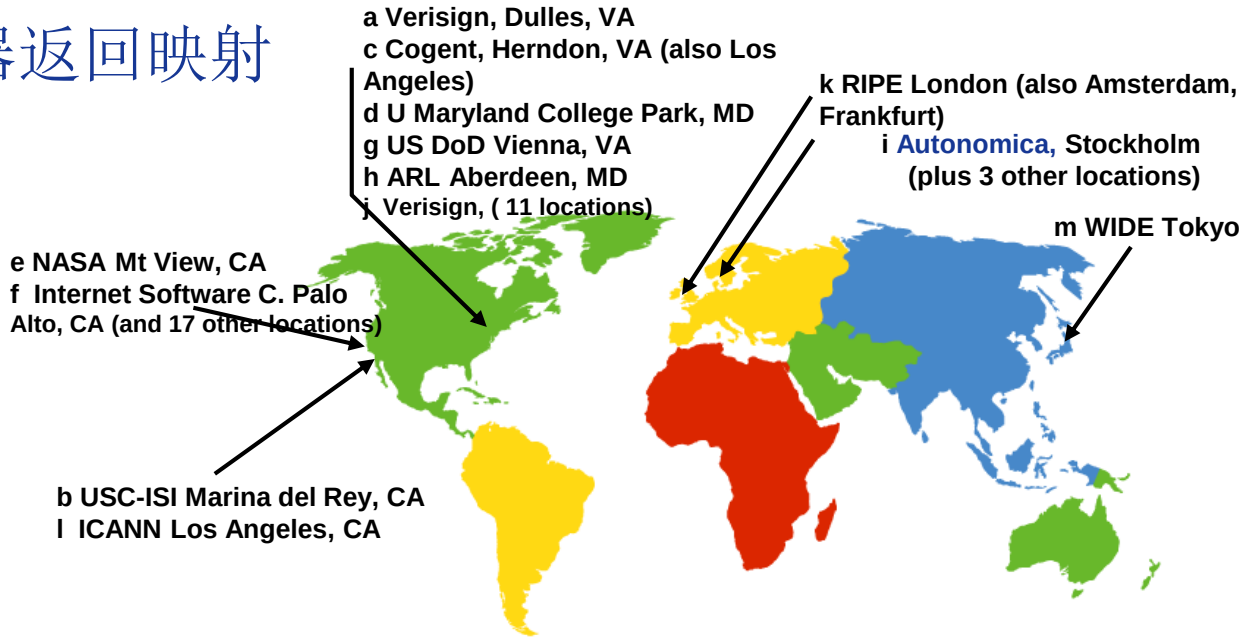
分布式层次式数据库



- 客户端想要查询www.amazon.com的IP
  - 客户端查询根服务器，找到com域名解析服务器
  - 客户端查询com域名解析服务器，找到amazon.com域名解析服务器
  - 客户端查询amazon.com域名解析服务器，获得www.amazon.com的IP地址

DNS根域名服务器

- 本地域名解析服务器无法解析域名时，访问根域名服务器
- 根域名服务器
  - 如果不知道映射，访问权威域名服务器
  - 获得映射
  - 向本地域名服务器返回映射



全球有13个根域名服务器

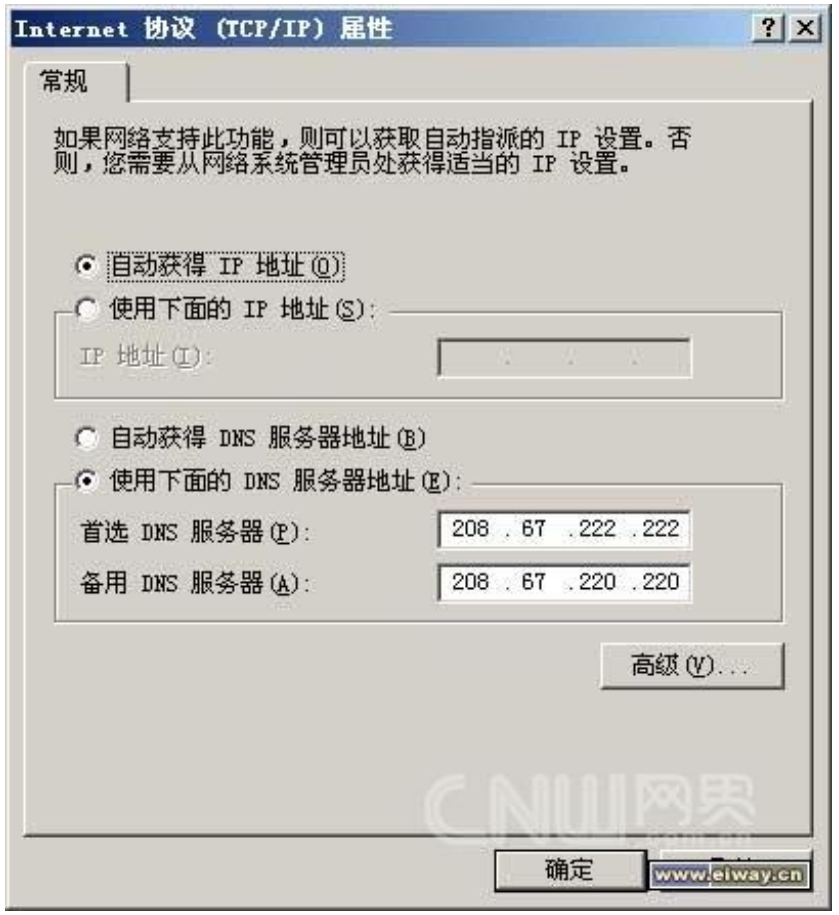
TLD和权威域名解析服务器

- 顶级域名服务器(TLD, top-level domain): 负责com, org, net,edu等顶级域名和国家顶级域名，例如cn, uk, fr等
  - Network Solutions维护com顶级域名服务器
  - Educause维护edu顶级域名服务器

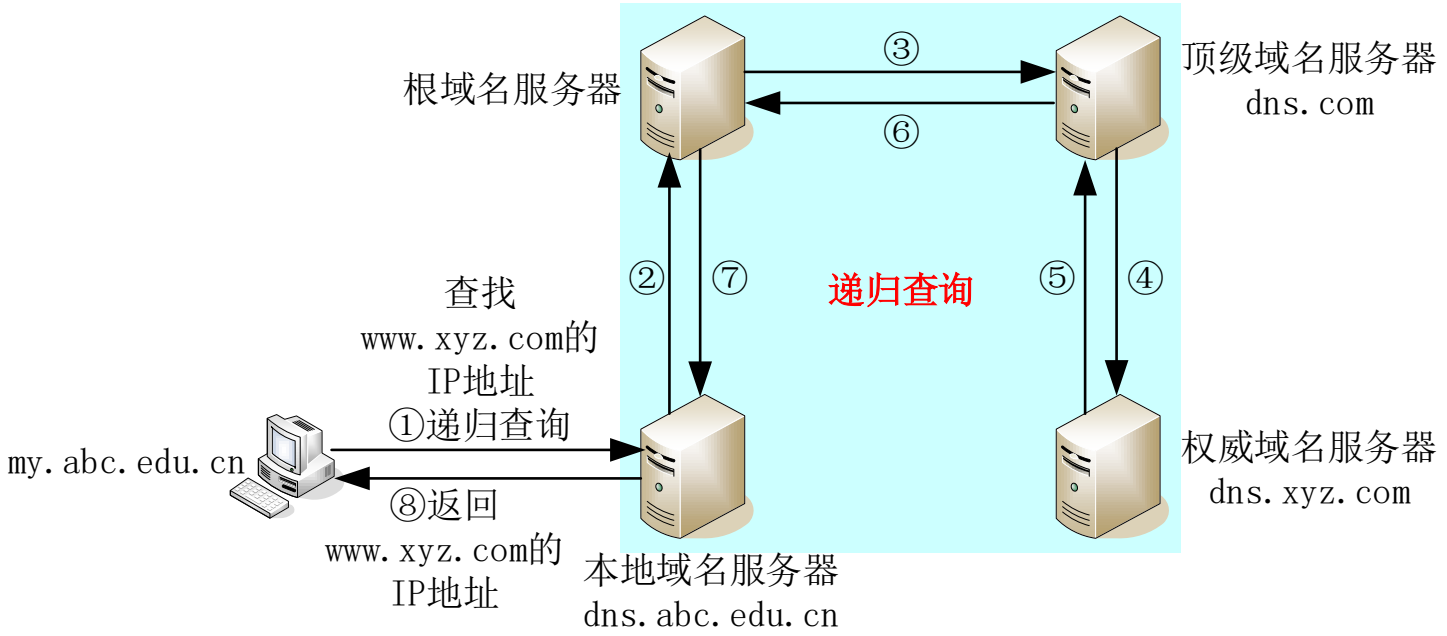
- 权威(Authoritative)域名服务器：组织的域名解析服务器，提供组织内部服务器的解析服务
  - 组织负责维护
  - 服务提供商负责维护

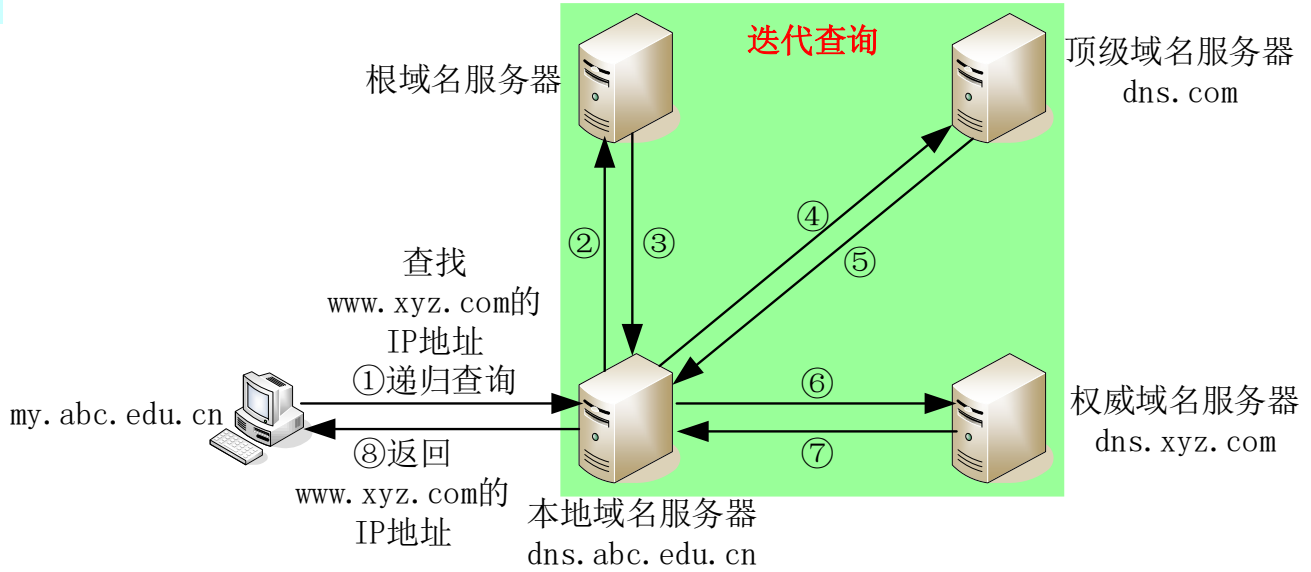
本地域名解析服务器

- 不严格属于层级体系
- 每个ISP有一个本地域名服务器
  - 默认域名解析服务器
- 当主机进行DNS查询时，查询被发送到本地域名服务器
  - 作为代理(proxy)，将查询转发给（层级式）域名解析服务器系统



DNS查询过程


根域名服务器
③
⑥
顶级域名服务器
dns. com
②
⑦
递归查询
⑤
④
查找
www. xyz. com的
IP地址
①递归查询
my. abc. edu. cn
⑧返回
www. xyz. com的
IP地址
本地域名服务器
dns. abc. edu. cn
权威域名服务器
dns. xyz. com


迭代查询
根域名服务器
顶级域名服务器
dns. com
④
②
③
⑤
查找
www. xyz. com的
IP地址
①递归查询
⑥
my. abc. edu. cn
⑧返回
⑦
www. xyz. com的
IP地址
本地域名服务器
dns. abc. edu. cn
权威域名服务器
dns. xyz. com

例题2

- 如果本地域名服务器无缓存，当采用递归方法解析另一网络某主机域名时，用户主机、本地域名服务器发送的域名请求消息数分别为
  - A. 一条、一条
  - B. 一条、多条
  - C. 多条、一条
  - D. 多条、多条

【解析】域名递归解析过程中，主机向本地域名服务器发送DNS查询，被查询的域名服务器代理后续的查询，然后返回结果。所以，递归查询时，如果本地域名服务器无缓存，则主机和本地域名服务器都仅需要发送一次查询，故正确答案为**A**。

DNS记录缓存和更新

- 只要域名解析服务器获得域名—IP映射，即缓存这一映射
  - 一段时间过后，缓存条目失效（删除）
  - 本地域名服务器一般会缓存顶级域名服务器的映射
    - 因此根域名服务器不经常被访问

- 记录的更新/通知机制
  - RFC 2136
  - Dynamic Updates in the Domain Name System (DNS UPDATE)

DNS记录

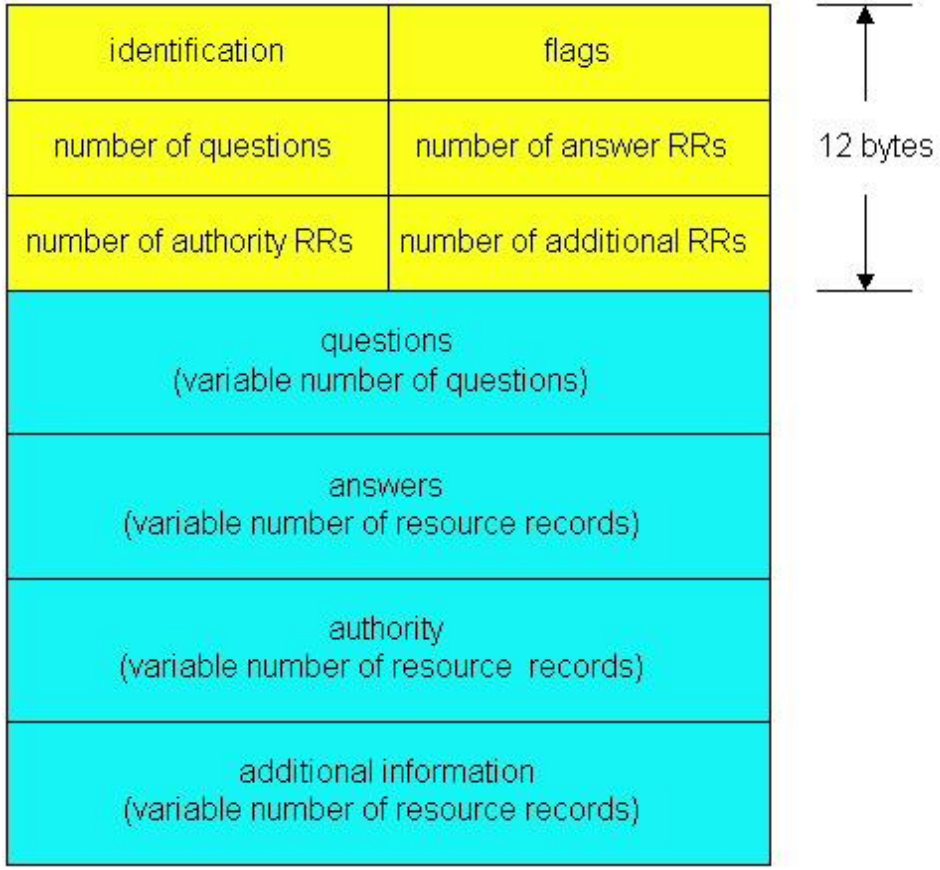
- 资源记录(RR, resource records)

RR format: `(name, value, type, ttl)`

- Type=A
  - Name: 主机域名
  - Value: IP地址
- Type=NS
  - Name: 域(edu.cn)
  - Value: 该域权威域名解析服务器的主机域名
- Type=CNAME
  - Name: 某一真实域名的别名
    - www.ibm.com – servereast.backup2.ibm.com
  - Value: 真实域名
- Type=MX
  - Value是与name（别名）相对应的邮件服务器

DNS协议与消息

- DNS协议：
  - 查询(query)和回复(reply消息)
  - 消息格式相同
- 消息头部
  - Identification: 16位查询编号，回复使用相同的编号
  - flags
    - 查询或回复
    - 期望递归
    - 递归可用
    - 权威回答



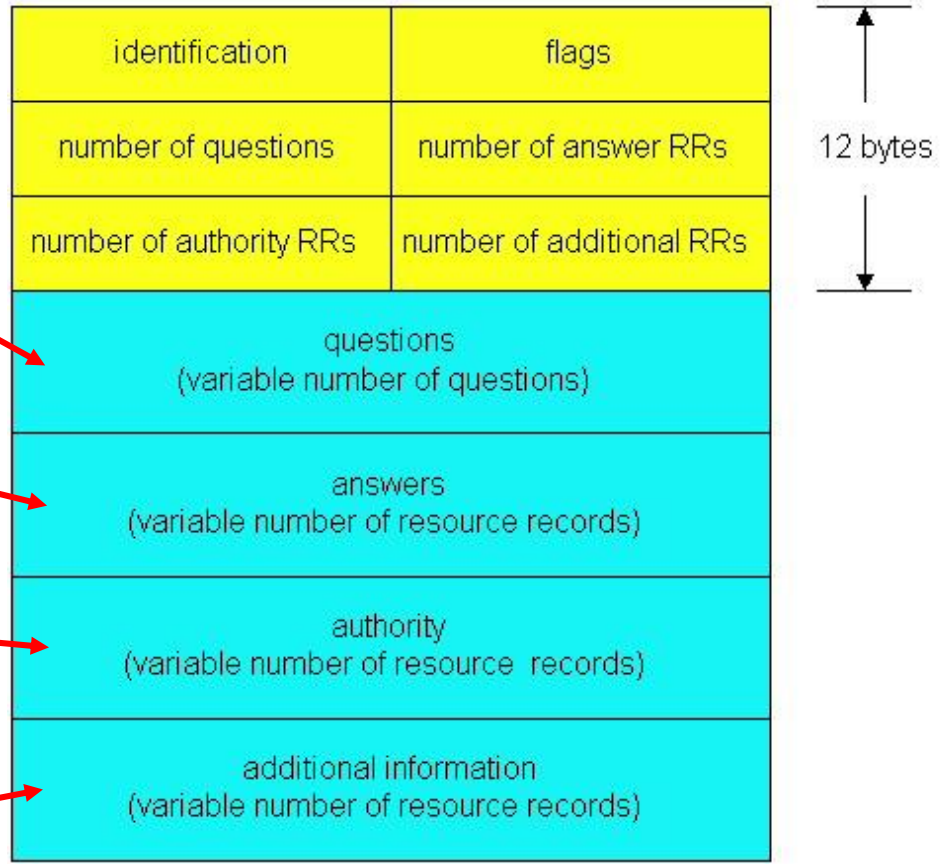
DNS协议与消息

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



如何注册域名？

- 例子：你刚刚创建了一个公司 “Network Utopia”
- 在域名管理机构(如Network Solutions)注册域名networkutopia.com
  - 向域名管理机构提供你的权威域名解析服务器的名字和IP地址
  - 域名管理机构向com顶级域名解析服务器中插入两条记录

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- 在权威域名解析服务器中：
  - 为www.networkuptopia.com加入Type A记录
  - 为xxx@networkutopia.com加入Type MX记录、 Type A记录
  - ……

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发

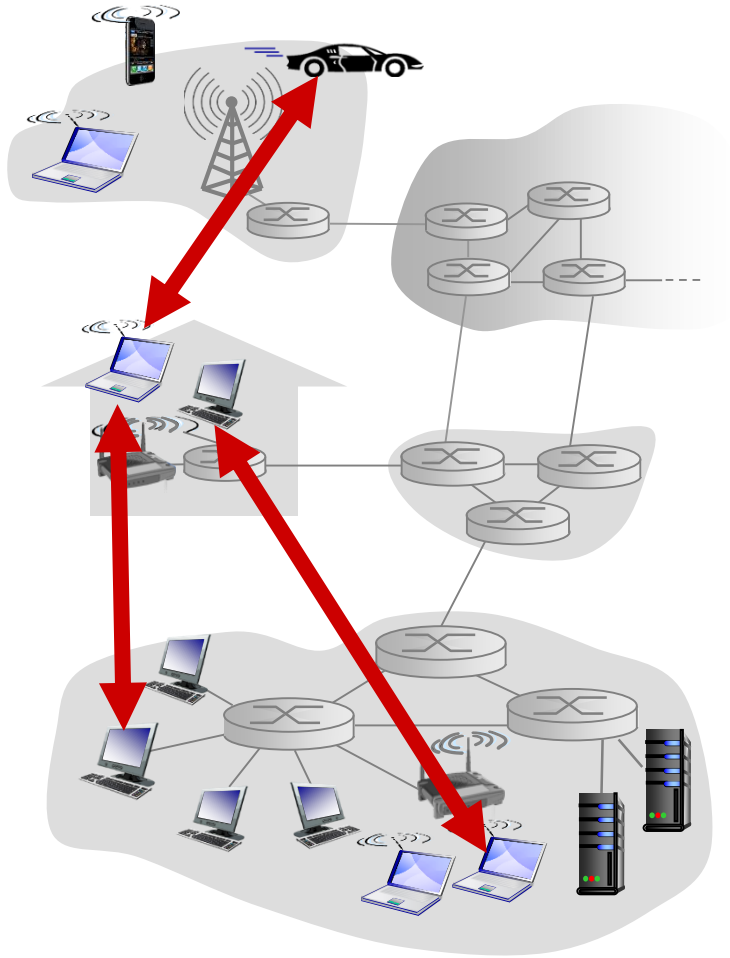
- Socket API
- Socket编程

纯P2P架构

- Peer-to-peer

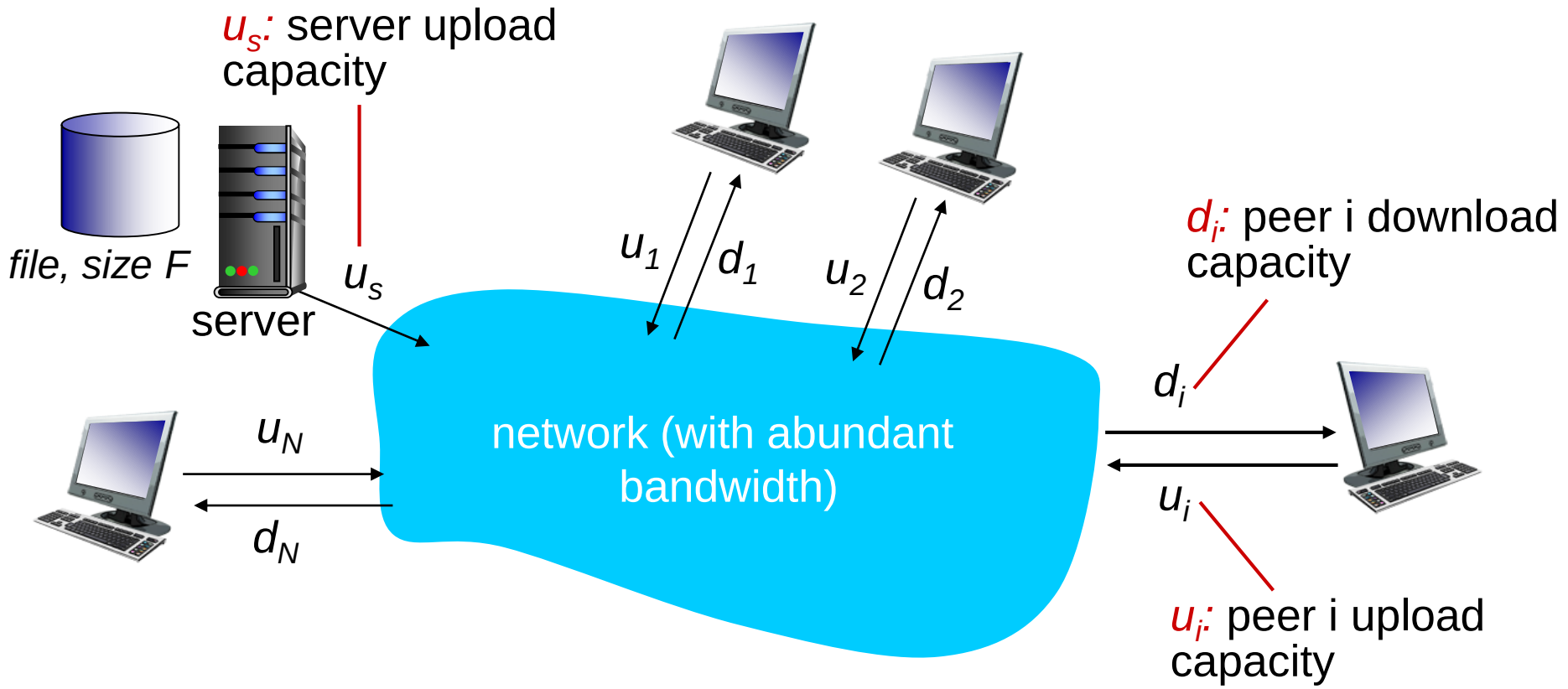
- 没有服务器
- 任意端系统之间直接通信
- 节点阶段性接入Internet
- 节点可能更换IP地址

- 以具体应用为例讲解



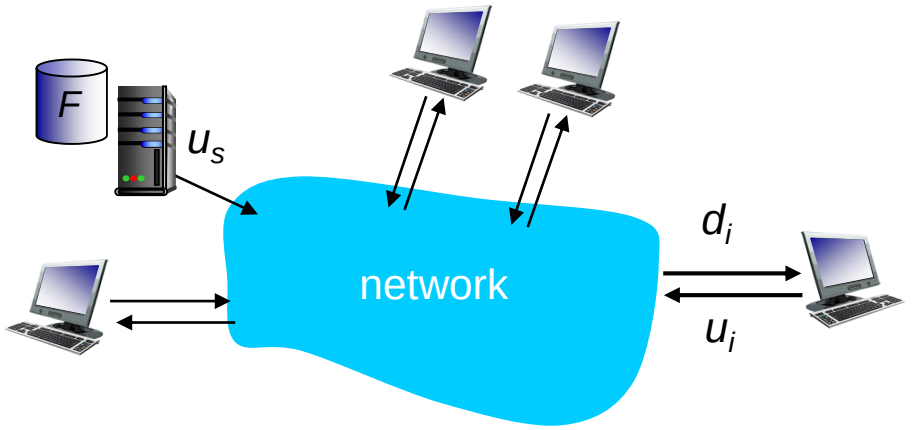
文件分发：客户机/服务器 vs. P2P

问题：从一个服务器向N个节点分发一个文件需要多长时间?



文件分发：客户机/服务器

- 服务器串行地发送N个副本
  - 时间： NF/u_s
- 每个客户机必须下载1个文件副本
  - 客户i下载1个副本时间：F/d_i
  - 下行带宽最小的客户下载1个副本时间：F/d_{min}



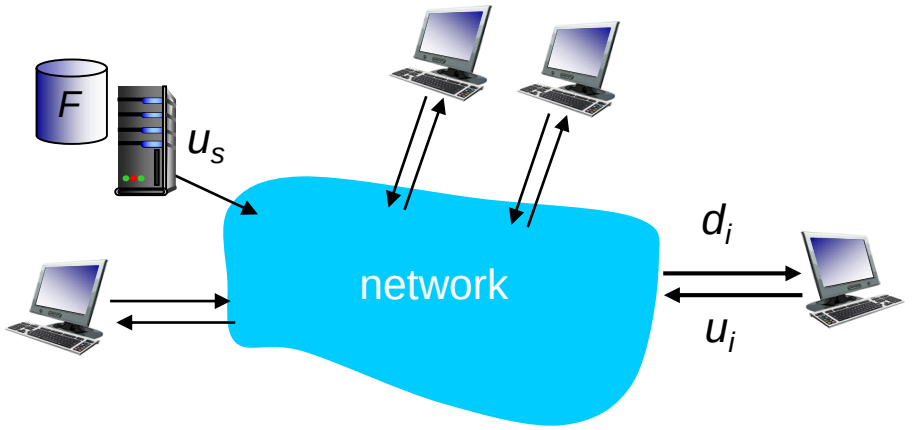
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

文件分发：P2P

- 服务器必须至少上传1个副本
  - 时间： F/u_s
- 每个客户机必须下载1个文件副本
  - 客户i下载1个副本时间：F/d_i
  - 下行带宽最小的客户下载1个副本时间：F/d_{min}
- 服务器与客户共同上传NF比特数据时间：
  $NF/(u_s + \Sigma u_i)$



time to distribute F to N clients using P2P approach

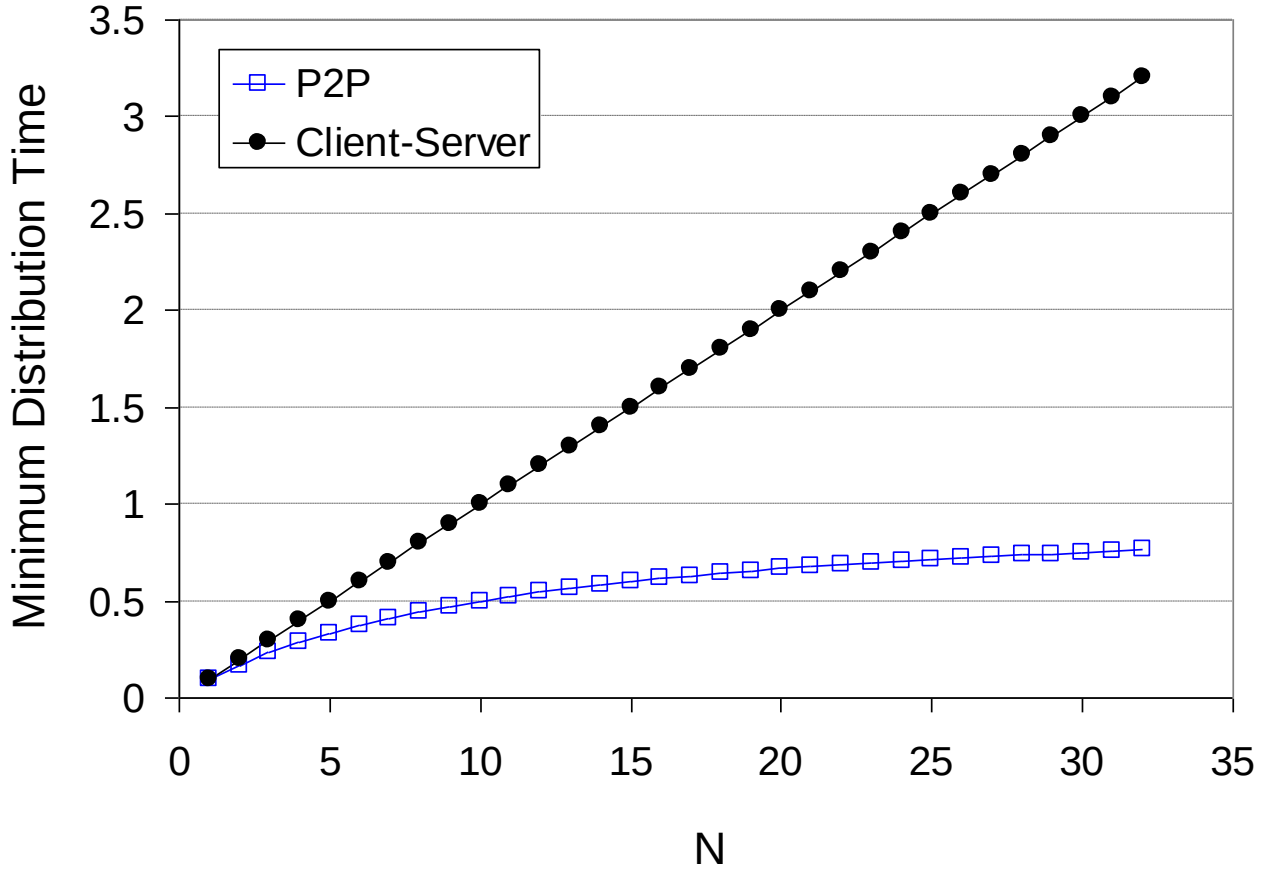
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

客户机/服务器 vs. P2P: 例子

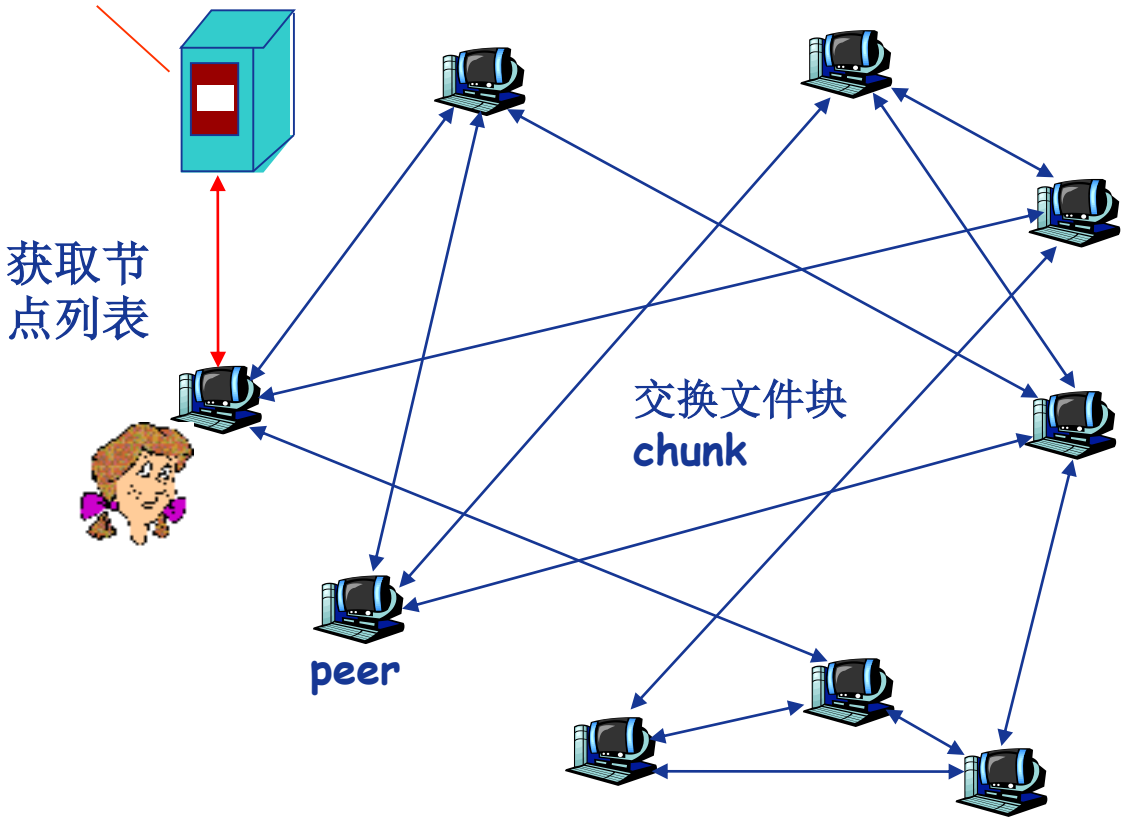
客户端上传速率= u， F/u = 1小时， $u_s = 10u$， $d_{min} \geq u_s$



文件分发：BitTorrent

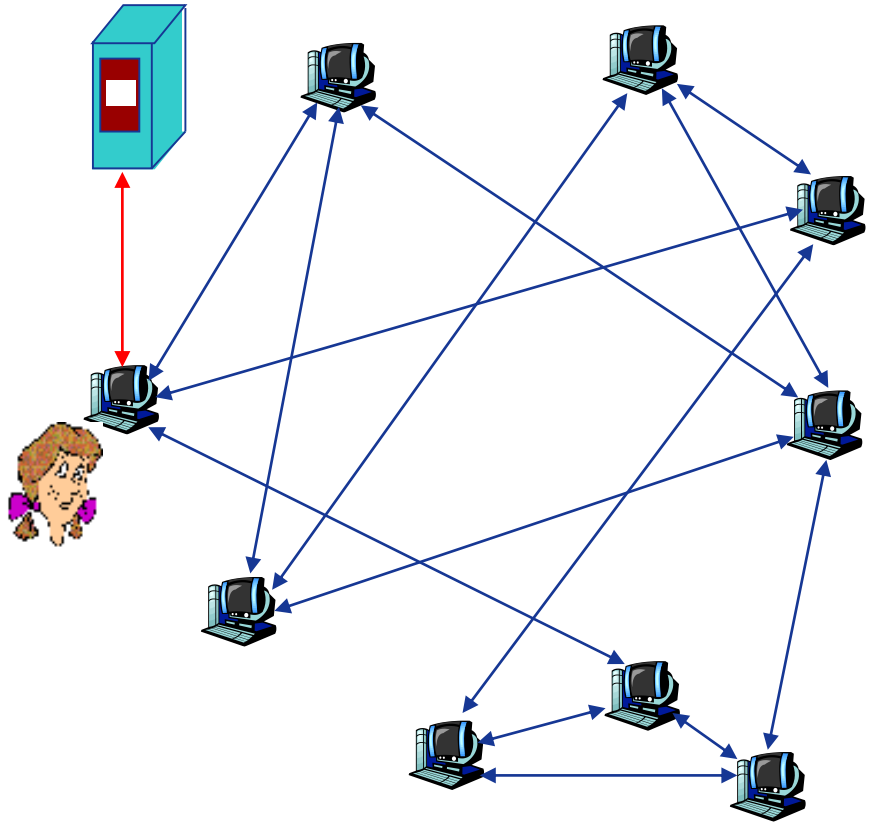
tracker: 跟踪参与torrent的节点

torrent: 交换同一个文件的文件块的节点组



BitTorrent (1)

- 文件划分为256KB的chunk
- 节点加入torrent
  - 没有chunk，但是会逐渐积累
  - 向tracker注册以获得节点清单，与某些节点（“邻居”）建立连接
- 下载的同时，节点需要向其他节点上传chunk
- 节点可能加入或离开
- 一旦节点获得完整的文件，它可能（自私地）离开或（无私地）留下

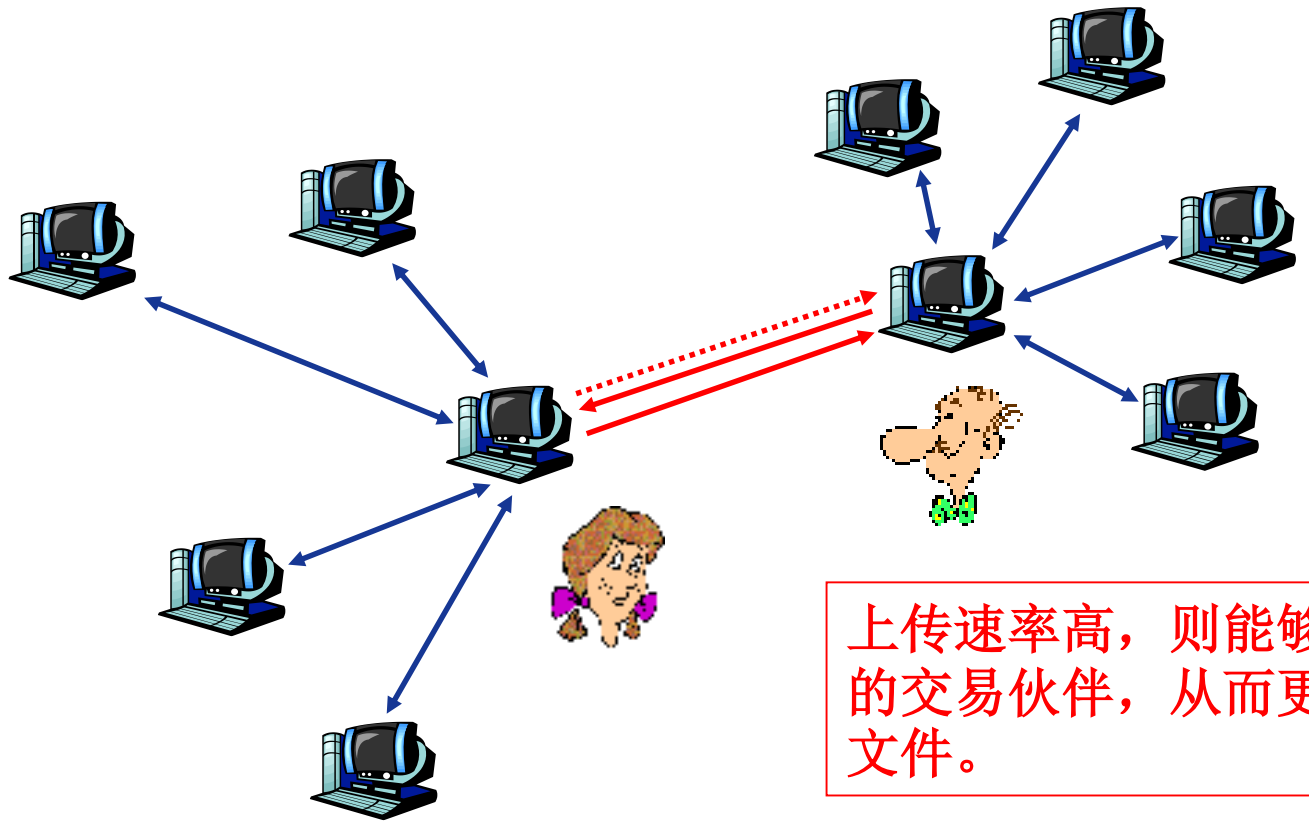


BitTorrent (2)

- 获取chunk
  - 给定任一时刻，不同的节点持有文件的不同chunk集合
  - 节点(Alice)定期查询每个邻居所持有的chunk列表
  - 节点发送请求，请求获取缺失的chunk
    - 稀缺优先
- 发送chunk: tit-for-tat
  - Alice向4个邻居发送chunk：正在向其发送Chunk，速率最快的4个
    - 每10秒重新评估top 4
  - 每30秒随机选择一个其他节点，向其发送chunk
    - 新选择节点可能加入top 4
    - “optimistically unchoke”

BitTorrent: Tit-for-tat

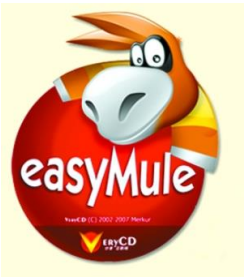
(1) Alice "optimistically unchokes" Bob
(2) Alice becomes one of Bob's top-four providers; Bob reciprocates
(3) Bob becomes one of Alice's top-four providers



上传速率高，则能够找到更好的交易伙伴，从而更快地获取文件。

P2P: 搜索信息

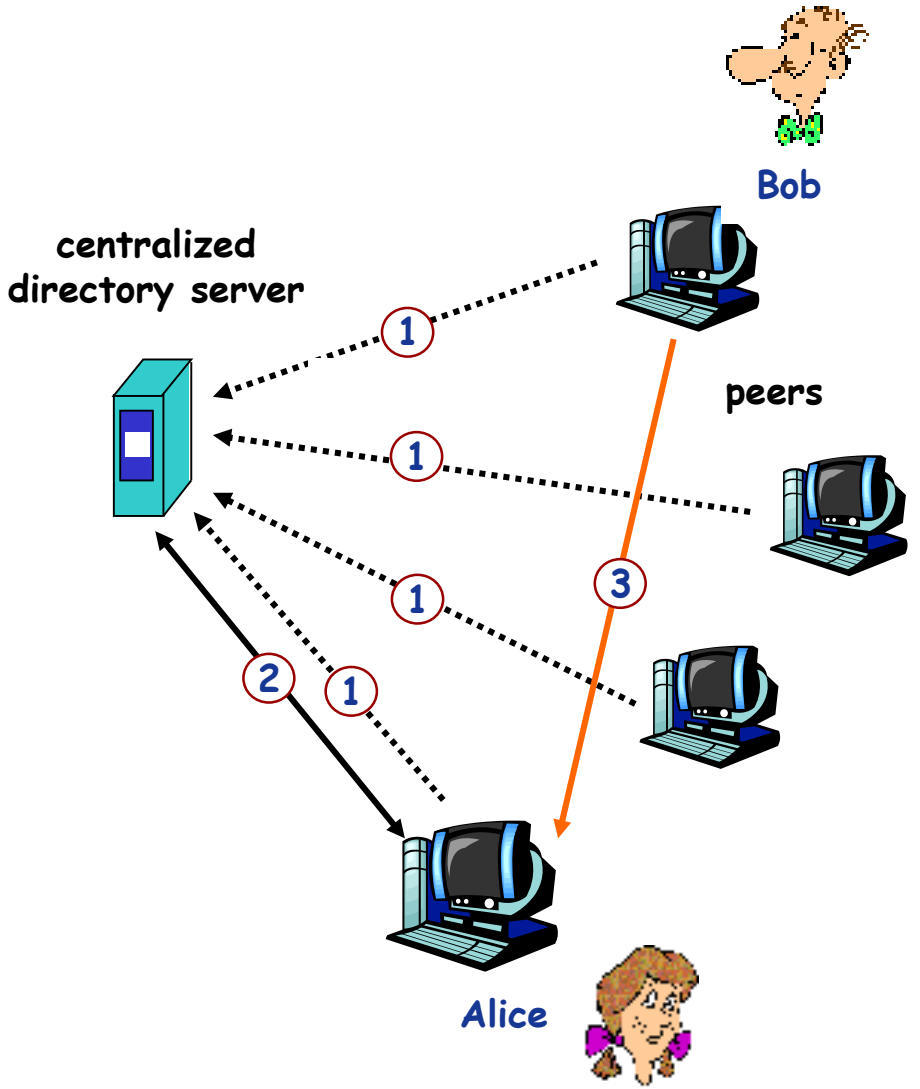
- P2P系统的索引：信息到节点位置(IP地址+端口号)的映射
- 文件共享(电驴)
  - 利用索引动态跟踪节点所共享的文件的位置
  - 节点需要告诉索引它拥有哪些文件
  - 节点搜索索引，从而获知能够得到哪些文件



- 即时消息(QQ)
  - 索引负责将用户名映射到位置
  - 当用户开启IM应用时，需要通知索引它的位置
  - 节点检索索引，确定用户的IP地址

集中式索引

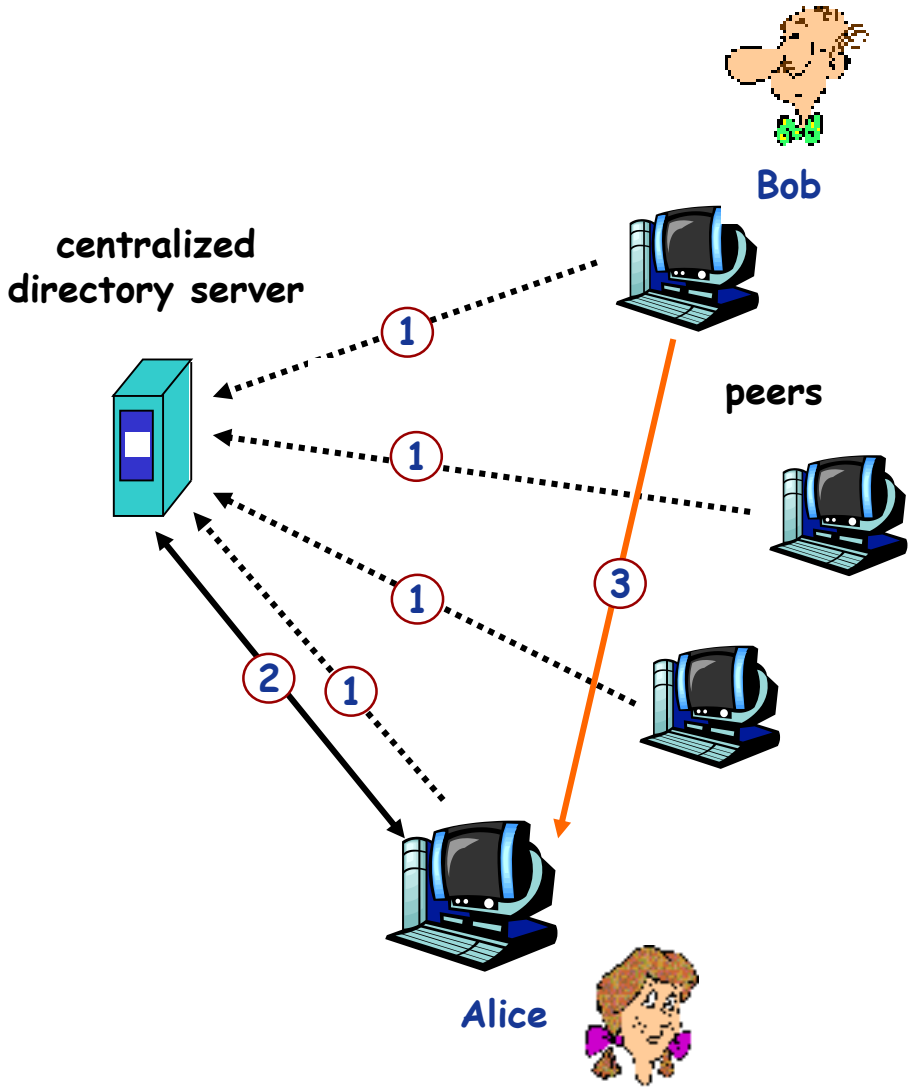
- Napster最早采用这种设计
  - 1) 节点加入时，通知中央服务器：
    - IP地址
    - 内容
  - 2) Alice查找“Hey Jude”
  - 3) Alice从Bob处请求文件



集中式索引的问题

内容和文件传输是分布式的，
但是内容定位是高度集中式的

- 单点失效问题
- 性能瓶颈
- 版权问题



洪泛式查询: Query flooding

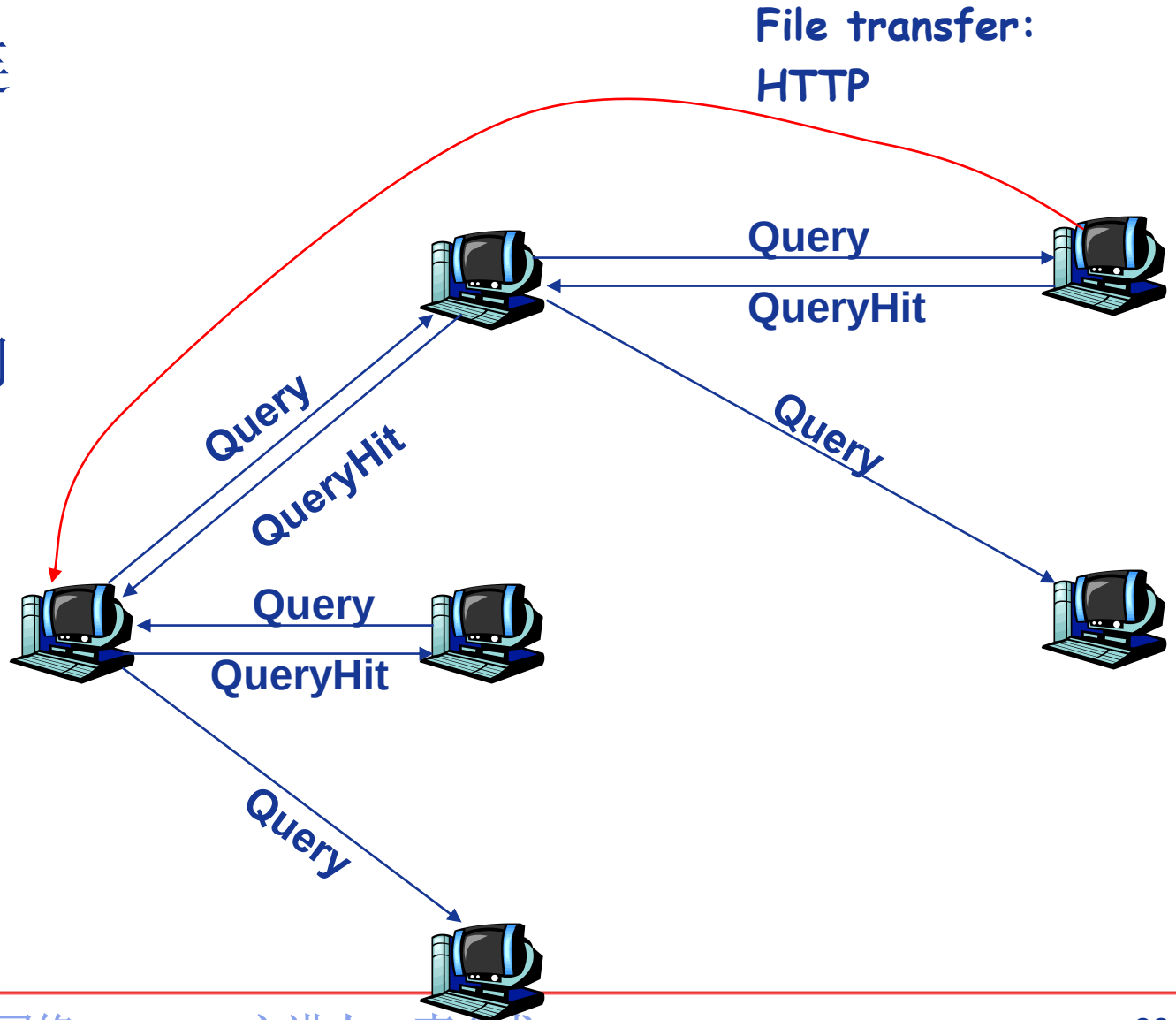
- 完全分布式架构
- Gnutella采用这种架构
- 每个节点对它共享的文件进行索引，且只对它共享的文件进行索引

覆盖网络(overlay network): Graph

- 节点X与Y之间如果有TCP连接，那么构成一个边
- 所有的活动节点和边构成覆盖网络
- 边：虚拟链路
- 节点一般邻居数少于10个

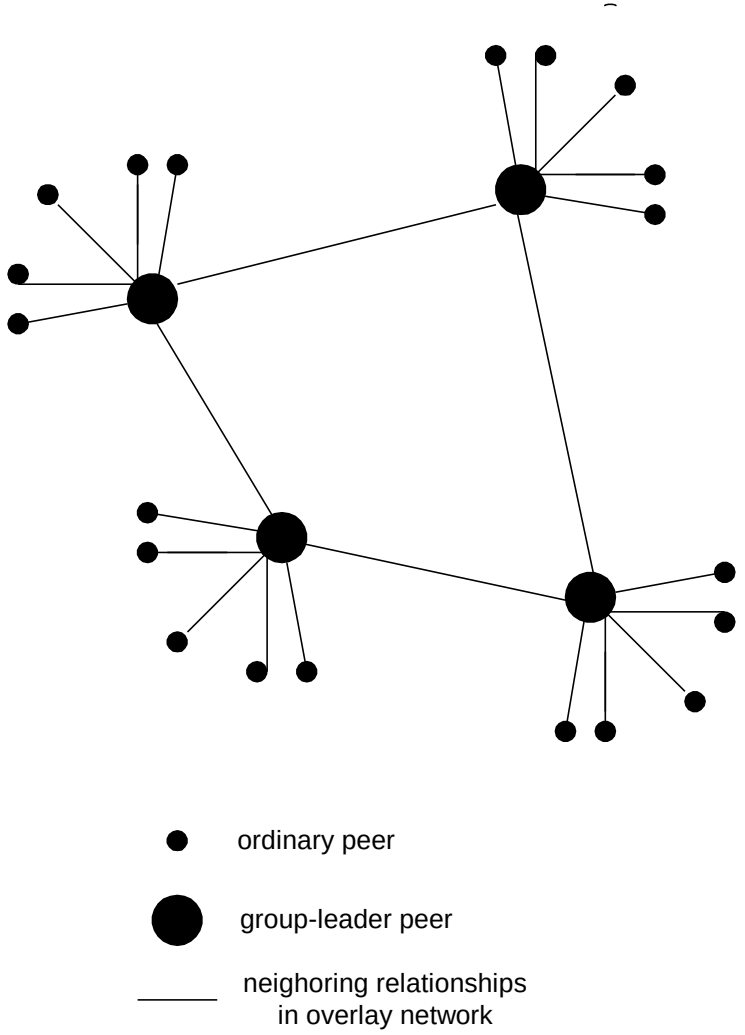
洪泛式查询: Query flooding

- 查询消息通过已有的TCP连接发送
- 节点转发查询消息
- 如果查询命中，则利用反向路径发回查询节点



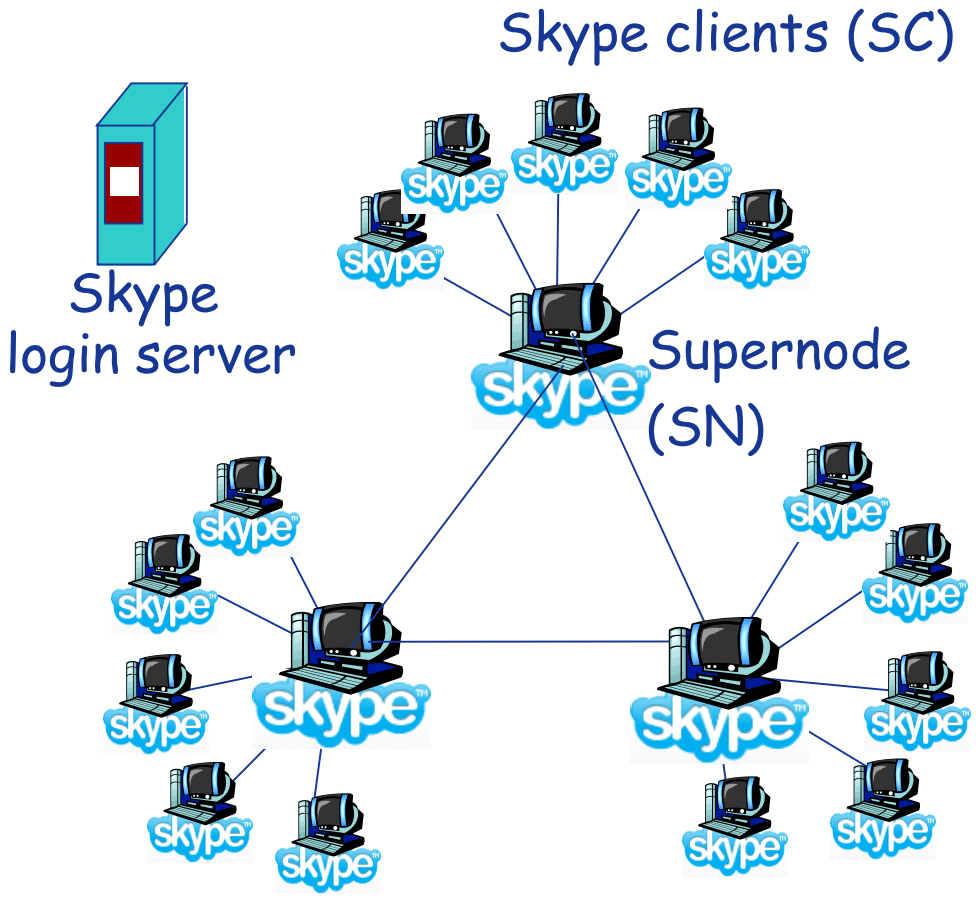
层次式覆盖网络

- 介于集中式索引和洪泛查询之间的方法
- 每个节点或者是一个超级节点，或者被分配一个超级节点
  - 节点和超级节点间维持TCP连接
  - 某些超级节点对之间维持TCP连接
- 超级节点负责跟踪子节点的内容



P2P案例应用：Skype

- 本质上是P2P的：用户/节点对之间直接通信
- 私有应用层协议
- 采用层次式覆盖网络架构
- 索引负责维护用户名与IP地址间的映射
- 索引分布在超级节点上



Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*
- database has (key, value) pairs; examples:
  - key: ss number; value: human name
  - key: movie title; value: IP addresses
- a peer queries DHT with key
  - DHT returns values that match the key
- peers can also insert (key, value) pairs
- Distribute the (key, value) pairs over the (millions of peers)

Q: how to assign keys to peers?

- central issue:
  - assigning (key, value) pairs to peers.
- basic idea:
  - convert each key to an integer
  - Assign integer to each peer
  - put (key,value) pair in the peer that is closest to the key

DHT identifiers

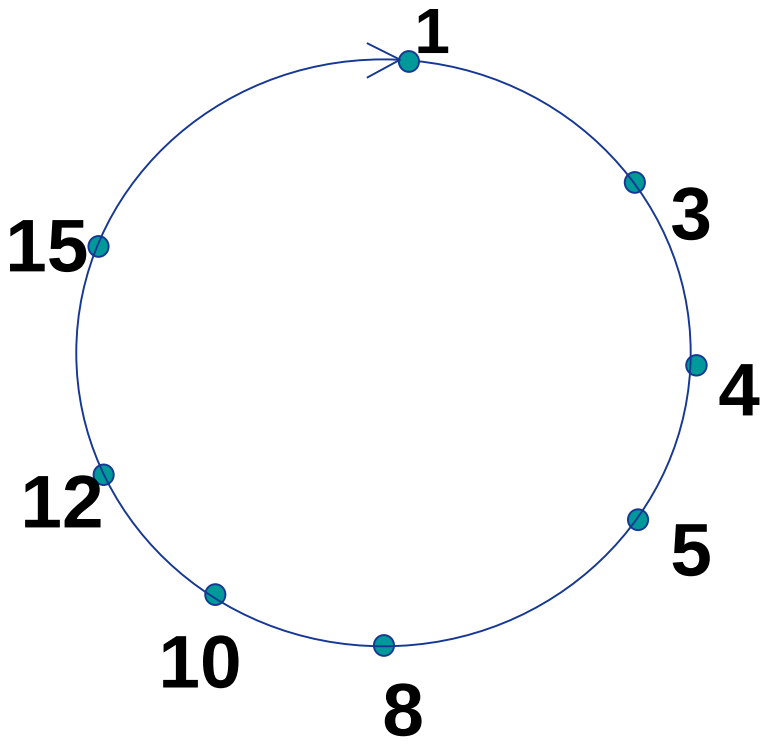
- assign integer identifier to each peer in range [0,2^n-1] for some *n*.
  - each identifier represented by *n* bits.

- require each key to be an integer in same range
- to get integer key, hash original key
  - *e.g.*, key = hash("Led Zeppelin IV")
  - this is why its is referred to as a *distributed "hash" table*

Assign keys to peers

- rule: assign key to the peer that has the *closest* ID.
- convention in lecture: closest is the *immediate successor* of the key.
- e.g., *n*=4; peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

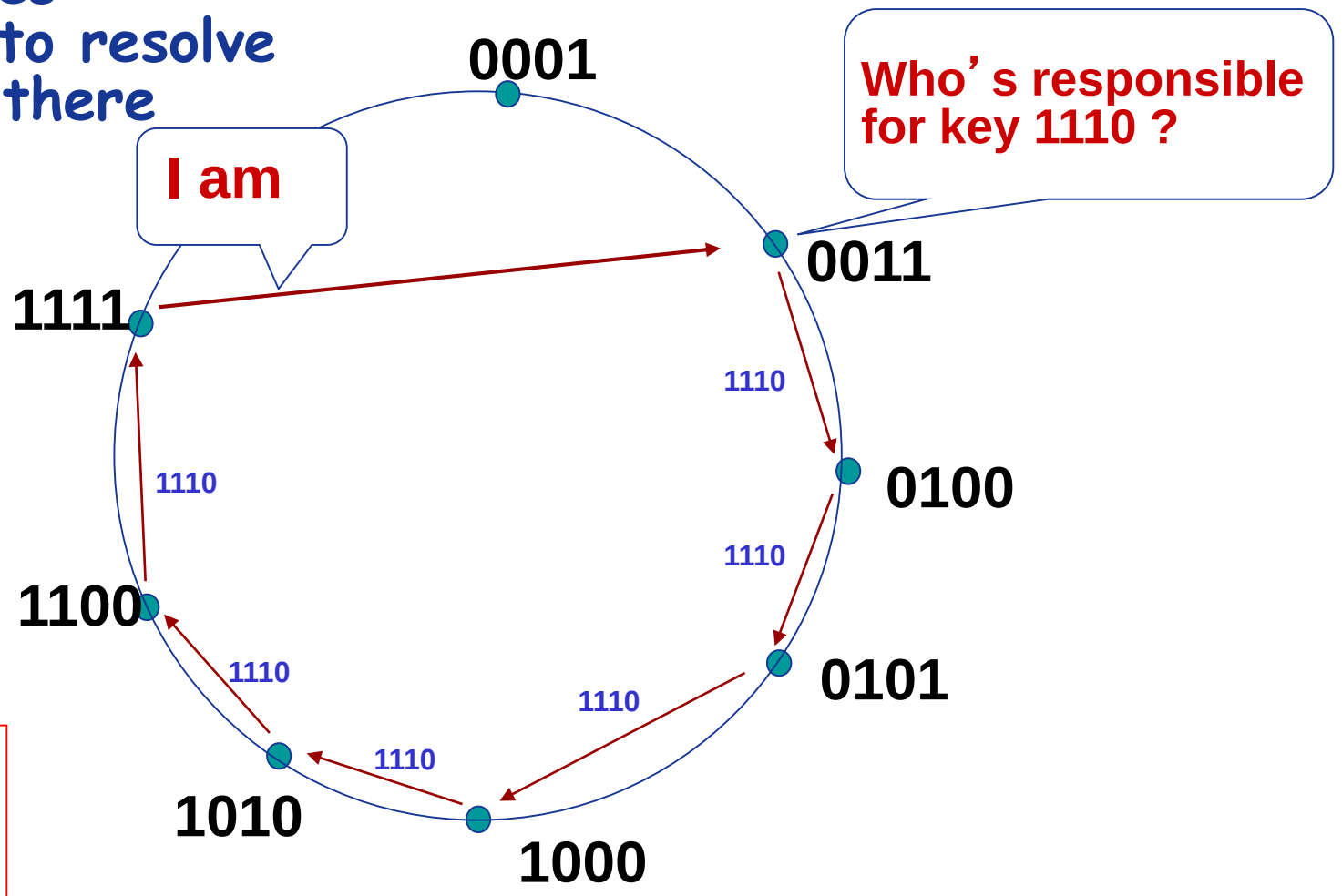
Circular DHT (1)



- each peer *only* aware of immediate successor and predecessor.
- “overlay network”

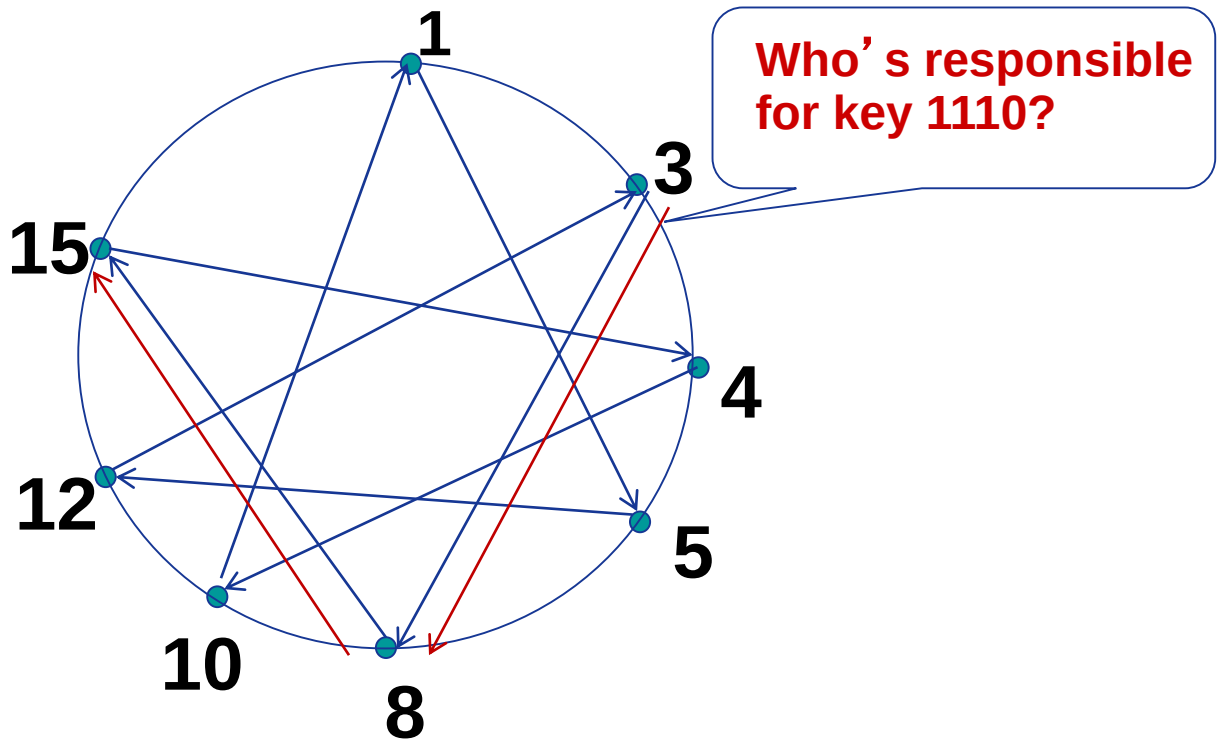
Circular DHT (1)

O(N) messages on avgerage to resolve query, when there are N peers



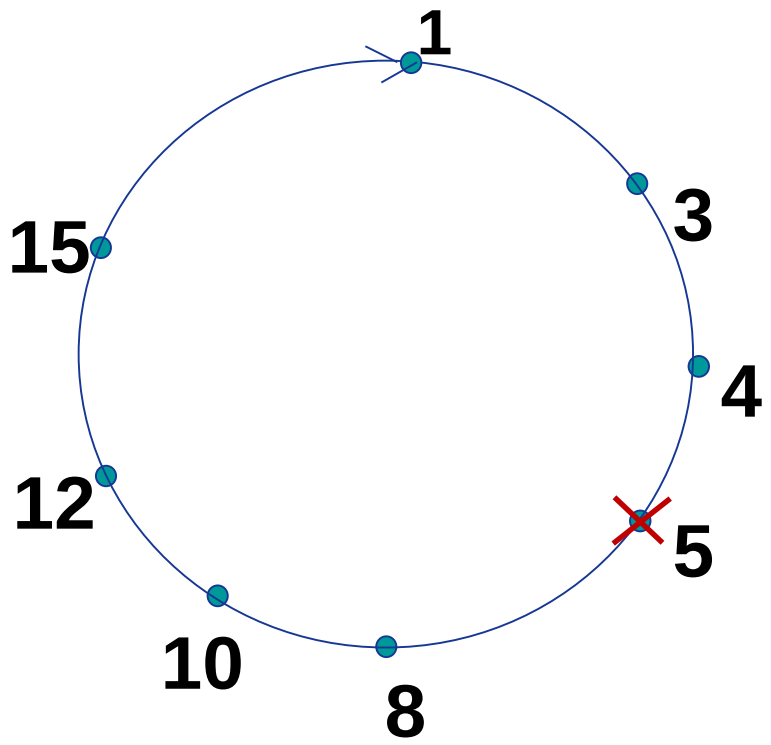
Define closest as closest successor

Circular DHT with shortcuts



- each peer keeps track of IP addresses of predecessor, successor, short cuts.
- reduced from 6 to 2 messages.
- possible to design shortcuts so *O(log N)* neighbors, *O(log N)* messages in query

Peer churn



handling peer churn:

- peers may come and go (churn)
- each peer knows address of its two successors
- each peer periodically pings its two successors to check aliveness
- if immediate successor leaves, choose next successor as new immediate successor

example: peer 5 abruptly leaves

- peer 4 detects peer 5 departure; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor.
- what if peer 13 wants to join?

主要内容

网络应用体系结构?

网络应用进程通信?

网络应用需求与传输层服务

Web应用

- HTTP协议
- HTTP报文
- Cookie技术
- Web缓存/代理

Email应用

- SMTP协议
- Email消息格式/RFC 822
- 邮件接收协议

FTP

DNS

- 域名结构
- 域名服务器
- 域名解析过程
- 资源记录RR

P2P应用

- P2P文件分发
- P2P索引技术

网络应用开发（略）

- Socket API
- Socket编程

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

立足航天，服务国防，面向国民经济主战场

谢 谢！