



## 第二章 应用层

## 我们的目标:

- ❖ 网络应用协议的概念、实现方面
  - 传输层服务模型
  - 客户机-服务器模式
  - 点到点（对等）模式
- ❖ 通过研究流行的应用层协议来了解协议
  - 超文本传送协议 HTTP
  - 文件传送协议FTP
  - SMTP / POP3 / IMAP
  - 域名服务器(Domain Name Server)DNS
- ❖ 创建网络应用程序
  - 套接字API

1 2.1 应用层协议原理

2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

5 2.5 DNS

6 2.6 P2P 应用

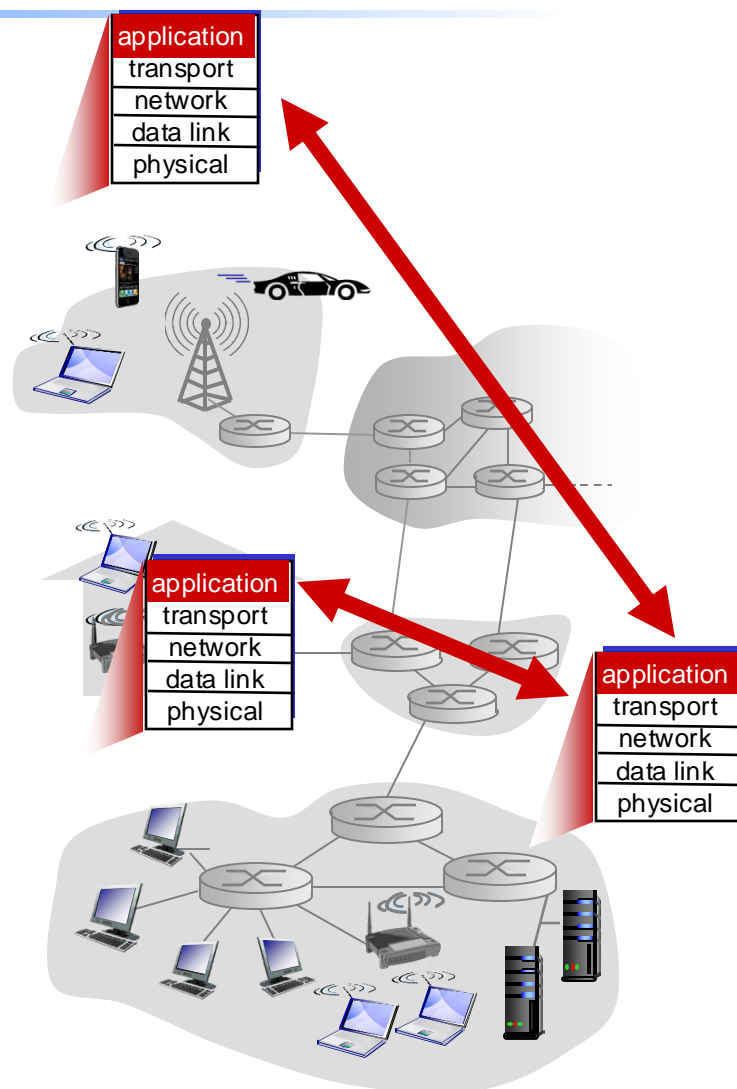
- ❖ 电子邮件
- ❖ Web
- ❖ 文本消息传送
- ❖ 远程登录
- ❖ P2P文件共享
- ❖ 多用户网络游戏
- ❖ 流媒体存储视频(YouTube、Hulu、Netflix)
- ❖ 网络电话(如Skype)
- ❖ 实时视频会议
- ❖ 网络社交
- ❖ 搜索
- ❖ ...
- ❖ ...

编写程序，能够：

- ❖ 运行于(不同的)终端系统
- ❖ 实现基于网络的通信
- ❖ 例如，网络服务器软件与浏览器软件通信

无需为网络核心设备（路由器、交换机）  
编写软件（这里指应用层软件）

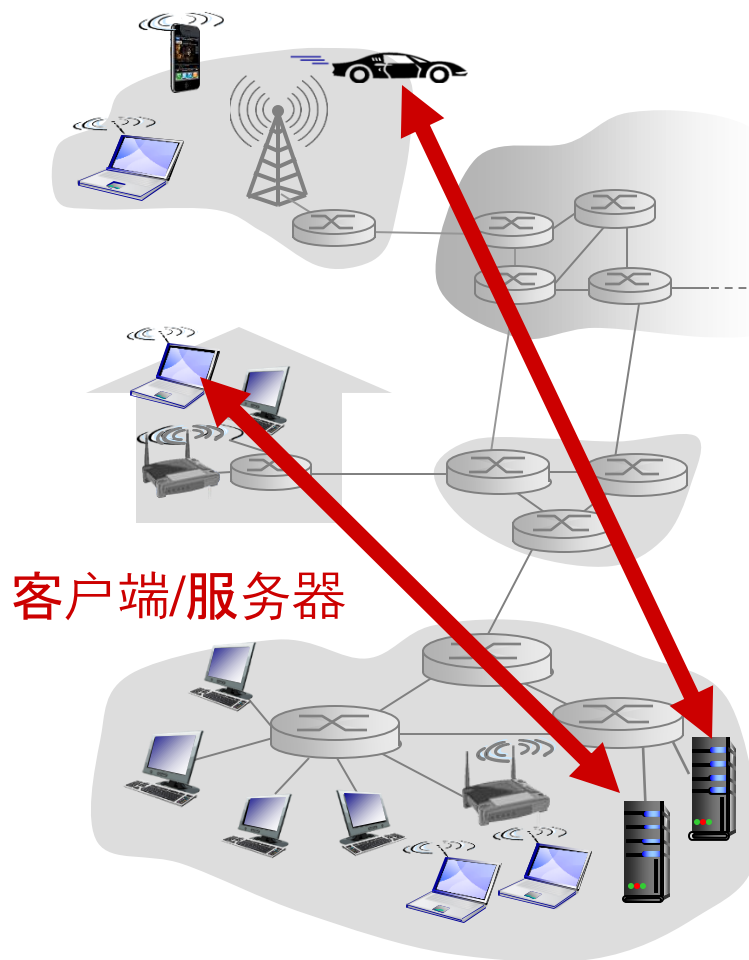
- ❖ 网络核心设备不运行用户应用程序
- ❖ 终端系统上的应用允许快速应用开发、部署



## 应用程序的可能结构:

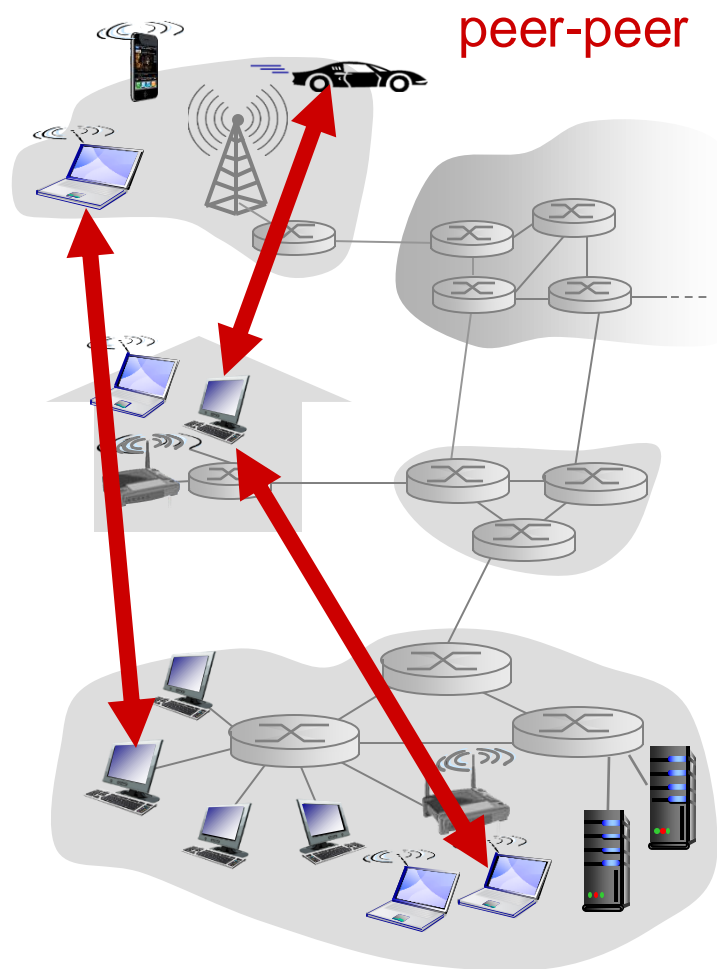
- ❖ 客户-服务器体系结构
- ❖ 对等模式(P2P)

```
10.184.2 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
10.184.2: Enter host <Alt+R>
[gealarm@websmap2 ~]$
[gealarm@websmap2 ~]$ uptime
18:31:30 up 5256 days, 4:40, 2 users, load average: 0.10, 0.15, 0.10
[gealarm@websmap2 ~]$ rpm -qi tar | grep Install
Install Date: 2009/27 Build Host: hs20-bc1-6.build.redhat.com
[gealarm@websmap2 ~]$ cat /proc/cpuinfo | grep "model name"
model name      : Intel(R) Xeon(R) CPU           E5504  @ 2.00GHz
model name      : Intel(R) Xeon(R) CPU           E5504  @ 2.00GHz
model name      : Intel(R) Xeon(R) CPU           E5504  @ 2.00GHz
model name      : Intel(R) Xeon(R) CPU           E5504  @ 2.00GHz
[gealarm@websmap2 ~]$ uname -a
Linux websmap2 2.6.18-164.el5 #1 SMP Tue Aug 18 15:51:48 EDT 2009 x86_64
x86_64 x86_64 GNU/Linux
[gealarm@websmap2 ~]$ free -g
              total        used        free      shared    buffers     cached
Mem:           3          3          0          0          0          2
-/+ buffers/cache: 0          3          0
Swap:          1          0          1
[gealarm@websmap2 ~]$ cat /etc/*rele*
EBUPT STANDARD LINUX RELEASE 5.4.1
cat: /etc/lsb-release.d: %
Red Hat Enterprise Linux Server release 5.4 (Tikanga)
[gealarm@websmap2 ~]$
[gealarm@websmap2 ~]$ date
202432:33:14 CST
[gealarm@websmap2 ~]$
```



## ❖ Peers

- 没有始终开启的服务器
- ❖ 应用程序在间断连接的端系统之间直接通信，这些端系统称为“对等方”
- ❖ 对等方可以访问其他对等方提供的服务，反过来也为其他对等方提供服务
  - 对专用服务器的依赖最小(或没有)
  - 具备自扩展性——新的对等体带来了新的服务能力以及新的服务需求
- ❖ 分散结构
  - 管理、安全性和可靠性



**进程:**在主机中运行的程序

- ❖ 在同一主机中，两个进程使用**进程间通信**(由操作系统OS定义)进行通信
- ❖ 运行在不同主机上的进程则使用**消息**交换机制（应用层协议）进行通信

客户端，服务器

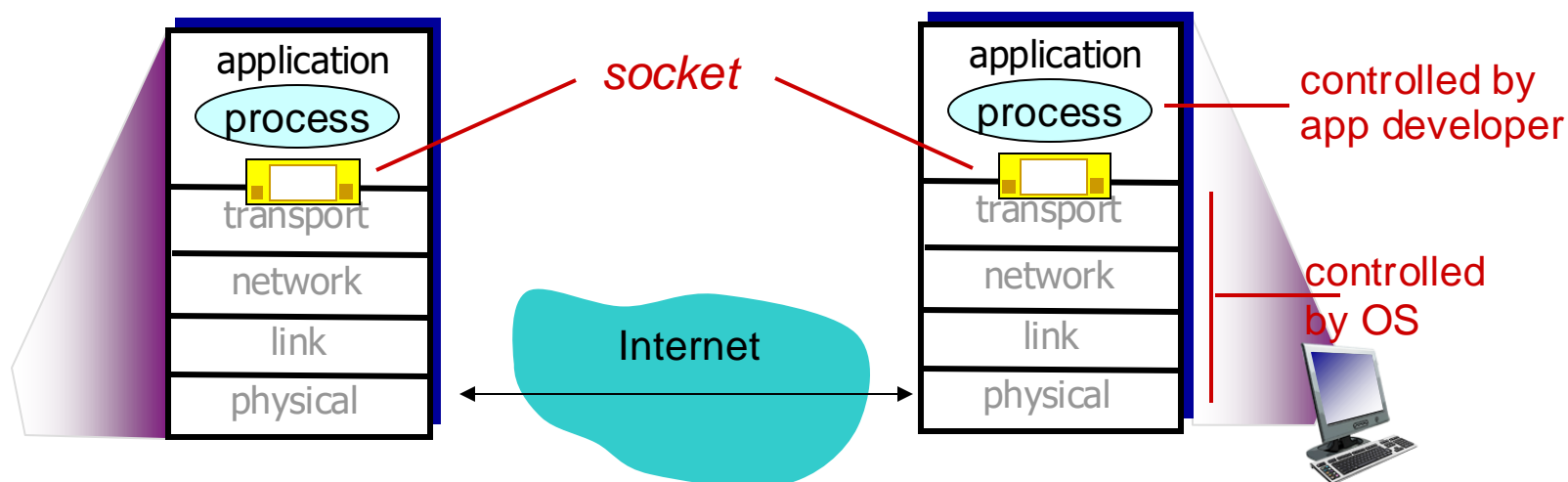
客户端进程：发起通信（如HTTP请求服务器资源）

服务器进程：等待客户端进程连接（监听）

- 此外:P2P体系架构的应用程序同时拥有客户端进程和服务端进程



- ❖ 进程通过套接字发送或接收报文
- ❖ 将套接字比拟为门（任意门）
  - 发送进程将消息推出门外（socket）
  - 发送进程依赖于与另一端的门之间的网络传输基础设施将消息传递给接收方进程的socket



- ❖ 要接收消息，进程必须有一个唯一的标识符（ID）
- ❖ 主机设备具有唯一的32位IP地址
- ❖ 问:运行进程的主机的IP地址能否满足唯一标识一个进程
  - 答:不行, 在同一主机上, 可同时运行多个进程
- ❖ 标识符包括与主机上的进程相关的IP地址和端口号。
- ❖ 示例端口号:
  - HTTP服务器:80
  - 邮件服务器:25
- ❖ 向job.nwpu.edu.cn web服务器发送HTTP消息:
  - IP地址:61.150.43.9
  - 端口号:80

- ❖ 报文交换的类型,
  - 例如请求、响应
- ❖ 报文语法:
  - 消息中的哪些字段以及字段是如何描述的
- ❖ 报文语义
  - 字段中信息的含义
- ❖ 规则: 进程发送以及接收报文的时机和方式

## 开放协议:

- ❖ 在RFC中定义
- ❖ 允许互操作性
- ❖ 例如HTTP、SMTP

## 专有协议:

- ❖ 例如Skype、QQ

## 可靠的数据传输

- ❖ 一些应用程序(例如文件传输、web 事务)要求100%可靠的数据传输
- ❖ 其他应用程序(如语音)可以承受一些损失

## 实时性

- ❖ 一些应用程序(如IP电话、交互式游戏)需要尽可能低的延时

## 吞吐量

- ❖ 某些应用（例如多媒体）有吞吐量的最低限制来确保其效果
- ❖ 其他一些应用（弹性应用）能够根据情况或多或少的利用可供使用的吞吐量

## 安全

- ❖ 加密、数据完整性.....

应用	数据丢失	吞吐量	时间敏感的
文件传输	没有损失	弹性的	不
电子邮件	没有损失	弹性的	不
Web文档	没有损失	弹性的	不
实时音频/视频	容许损失的	音频:5kbps-1Mbps 视频:10kbps-5Mbps	是的, 100毫秒
存储的音频/视频	容许损失的	同上	是的, 几秒钟
交互式游戏	容许损失的	几kbps以上	是的, 100毫秒
文本消息传送	没有损失	弹性的	既肯定又否定

## TCP服务:

- ❖ **可靠的数据传输**
- ❖ **流量控制:** 发送数据的速度决不超过接收的速度
- ❖ **拥塞控制:** 当网络超负荷时, 束紧发送端口, 减缓发送速度
- ❖ **不提供:** 实时性, 最小带宽承诺, 安全
- ❖ **面向连接:** 在客户端和服务端进程之间需要建立连接

## UDP服务:

- ❖ **无连接的**
- ❖ 在客户端和服务端进程之间实现 **“不可靠的” 数据传输**
- ❖ **不提供:** 可靠性保证, 流量控制, 拥塞控制, 实时性, 最小带宽承诺, 安全性, 建立连接
- ❖ **Q:** 为何会二者均有, 为何有UDP

应用	应用 层协议	潜在的 传输协议
电子邮件	SMTP [RFC 2821]	TCP
远程终端访问	Telnet[RFC 854]	TCP或UDP
网	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流媒体	HTTP(例如YouTube), RTP [RFC 1889]	TCP或UDP
互联网电话	SIP, RTP, proprietary (例如Skype)	TCP或UDP

## TCP和UDP

- ❖ 没有加密
- ❖ 以明文形式发送到套接字  
遍历Internet的明文密码

## SSL (Secure Sockets Layer 安全套接层)

- ❖ 提供加密的TCP连接
- ❖ 数据完整性
- ❖ 端点认证

## SSL位于应用层

- ❖ 应用程序通过SSL类库调用  
TCP服务

## SSL Socket API

- ❖ 密码在传输过程中被加密
- ❖ 参见第7章



1 2.1 应用层协议原理

2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

5 2.5 DNS

6 2.6 P2P 应用

首先, 回顾一下...

- ❖ 网页由对象组成
- ❖ web页面包含那些对象 (元素)
- ❖ HTML文件, JPEG图片, Java applet, 音频文件...
- ❖ HTML文件是web页面的基本组成, HTML中包换若干其他对象的引用
- ❖ 每个对象由URL寻址

`www.someschool.edu:80/someDept/pic.gif`

host name

path name

## HTTP:超文本传输协议

- ❖ Web的应用层协议
- ❖ 客户机/服务器模型
  - **客户端**: 浏览器, 请求、接收, 并展示web元素
  - **服务器**: web服务器响应客户端请求并发送web元素



## 使用TCP:

- ❖ 客户端启动TCP连接(创建 Socket) 到服务器, 端口 80
- ❖ 服务器接受来自客户端的 TCP 连接
- ❖ HTTP报文(应用层协议报文) 在浏览器 (http client) 和Web服务器(http server)之间进行交换
- ❖ 关闭TCP 连接

## HTTP是“无状态”的

- ❖ 服务器不保留任何访问过的请求信息

### 保留状态的协议很复杂!

- ❖ 过去的历史 (状态) 需要保留
- ❖ 一旦浏览器/服务器崩溃, 它们各自的状态视图就会发生分歧, 还需要重新核对

## 非持续HTTP连接

- ❖ 通过TCP连接最多发送一个对象
  - 然后连接关闭
- ❖ 下载多个对象需要多个连接

## 持续HTTP连接

- ❖ 在客户端和服务端之间建立一个连接就可以发送多个对象

# 非持续HTTP连接

假设用户输入URL:

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

(指向文字、10个  
jpeg图像等内容)

1a. HTTP客户端在端口80上发起  
到www.someSchool.edu的  
HTTP服务器(进程)的TCP连接

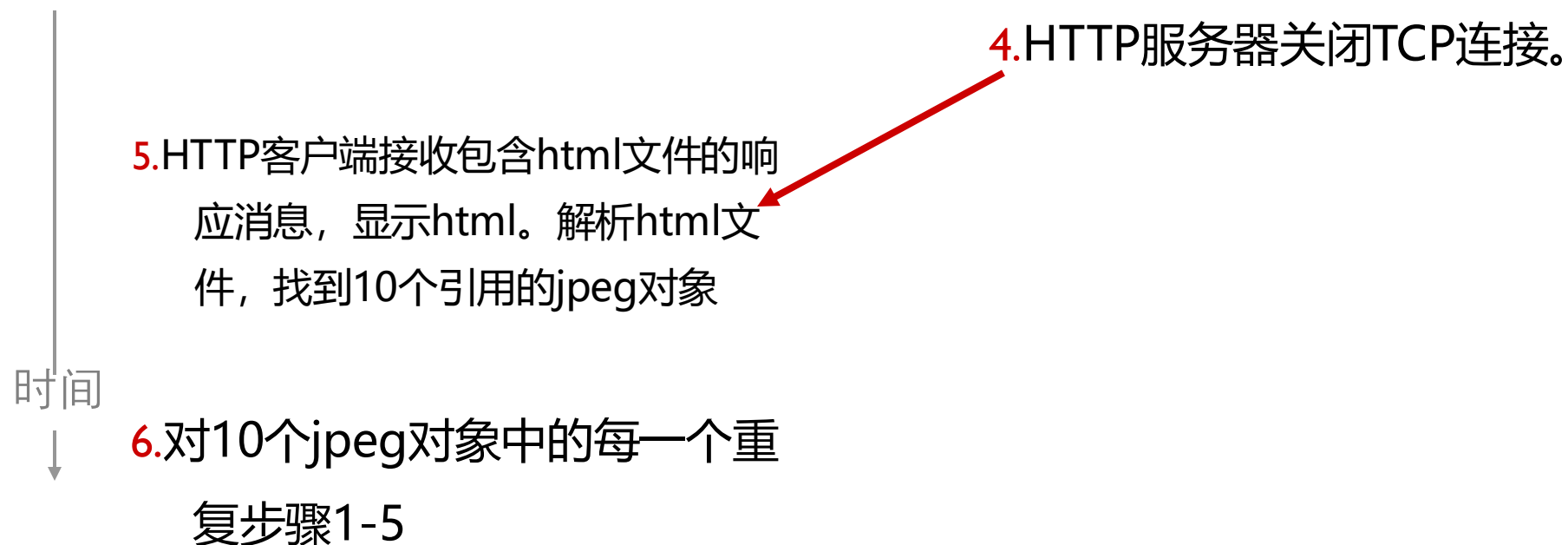
1b. 主机www.someSchool.edu上  
的HTTP服务器在端口80等待  
TCP连接。 “接受” 连接, 通知  
客户端

2. HTTP客户端将HTTP请求消息(  
包含URL)发送到TCP连接套  
接字。消息表明客户端需要  
某个部门/家庭对象. index

3. HTTP服务器接收请求消息,  
形成包含所请求对象的响应  
消息, 并将消息发送到其套  
接字中

时间

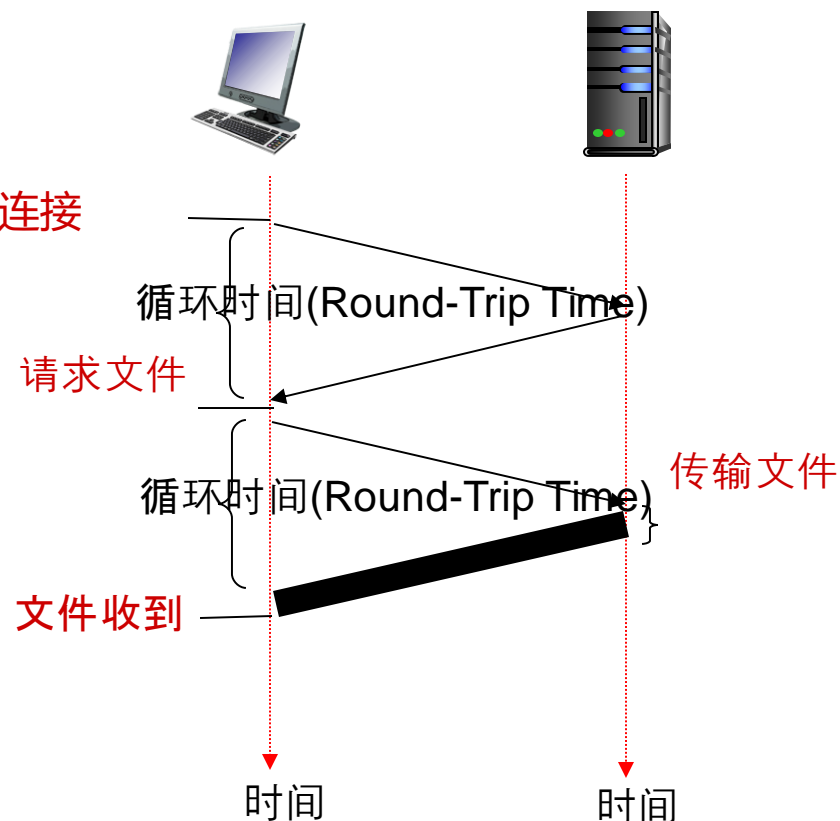




**RTT(往返时间):**一个小数据包从客户端传输到服务器并返回所需的时间

**HTTP响应时间:**

- ❖ 一个RTT用于初始化TCP连接
- ❖ 另一个RTT用于HTTP请求和相应
- ❖ 文件传输时间
- ❖ 非持续HTTP相应时间 =  $2RTT +$  文件传输的时间





## 非持续HTTP问题:

- ❖ 每个对象需要2个RTT
- ❖ 每个对象请求均需重新建立TCP连接, 增加了操作系统的负担
- ❖ 浏览器获取引用对象时通常并行的开启多个TCP连接 (增加了服务器的负担)

## 持续HTTP:

- ❖ 服务器发送完响应后保存TCP连接
- ❖ 在同一客户端/服务器上之后的HTTP报文, 可直接通过之前开启的TCP连接发送
- ❖ 客户端在遇到应用对象时会立即向服务器端发送请求
- ❖ 所有的引用对象请求只需一个RTT

❖ 两种类型的HTTP消息: 请求、响应

❖ HTTP请求消息:

▪ ASCII(人类可读格式)

请求行  
(GET, POST,  
HEAD commands)

首部行

回车,  
开始时换行  
线的表示  
标题行结束

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

回车字符  
换行字符

## 1. Mosaic/0.9 ;

The first web browser, Mosaic, was released in 1993 by the National Center for Supercomputing Applications (NCSA).

## 2. Mozilla/Version [Language] (Platform; Encryption) ;

When Netscape Communications began developing their web browser, its codename was Mozilla (short for “Mosaic Killer”).

## 3. Mozilla/Version (Platform; Encryption [; OS-or-CPU description])

Shortly after the release of Netscape Navigator 3, Microsoft released their first publicly available web browser, Internet Explorer 3. Since Netscape was the dominant browser at the time, many servers specifically checked for it before serving up pages.

## 4. Firefox: Gecko, APPLE: Webkit, Google: Chrome -----

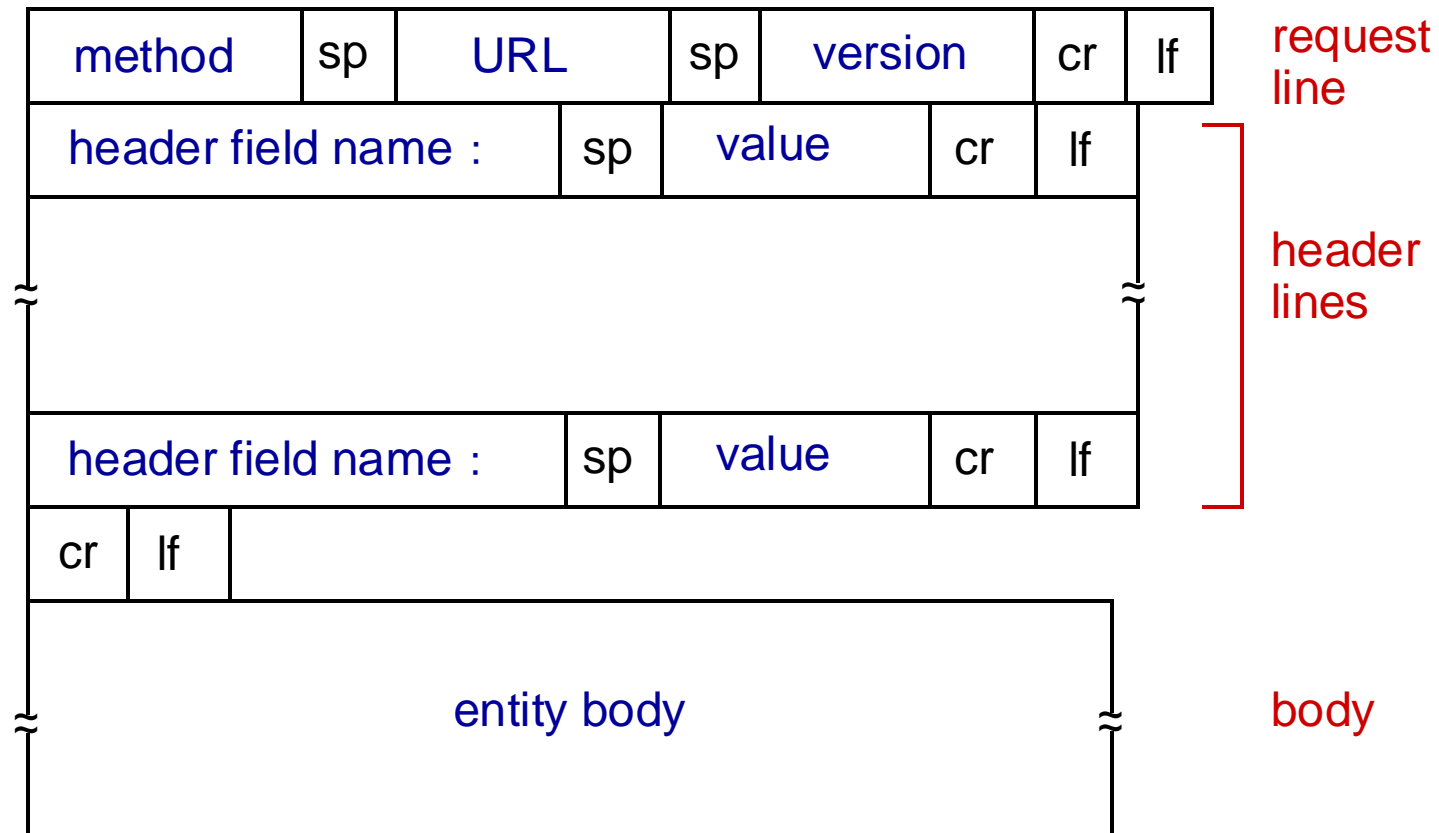
Google’s Chrome web browser uses WebKit as its rendering engine but uses a different JavaScript engine. For Chrome’s initial beta release, version 0.2, the user-agent string carries along all of the information from WebKit as well as an extra section for the Chrome version.

**Mozilla/5.0 (Platform; Encryption; OS-or-CPU; Language)**

**AppleWebKit/AppleWebKitVersion (KHTML, like Gecko)**

**Chrome/ChromeVersion Safari/SafariVersion**

# HTTP请求消息:一般格式



## HTTP/1.0 (RFC 1945) :

- ❖ GET
- ❖ POST
- ❖ HEAD
  - 服务器接受请求但并不返回对象，通常用于开发者调试跟踪

## HTTP/1.1 (RFC 2068) :

- ❖ GET, POST, HEAD
- ❖ PUT
  - 上传文件（对象）到服务器上由URL指定的路径
- ❖ DELETE
  - 删除服务器上的文件

## POST方法:

- ❖ web页面包含表单的input元素
- ❖ 输入内容封装在entity body中上传至服务器

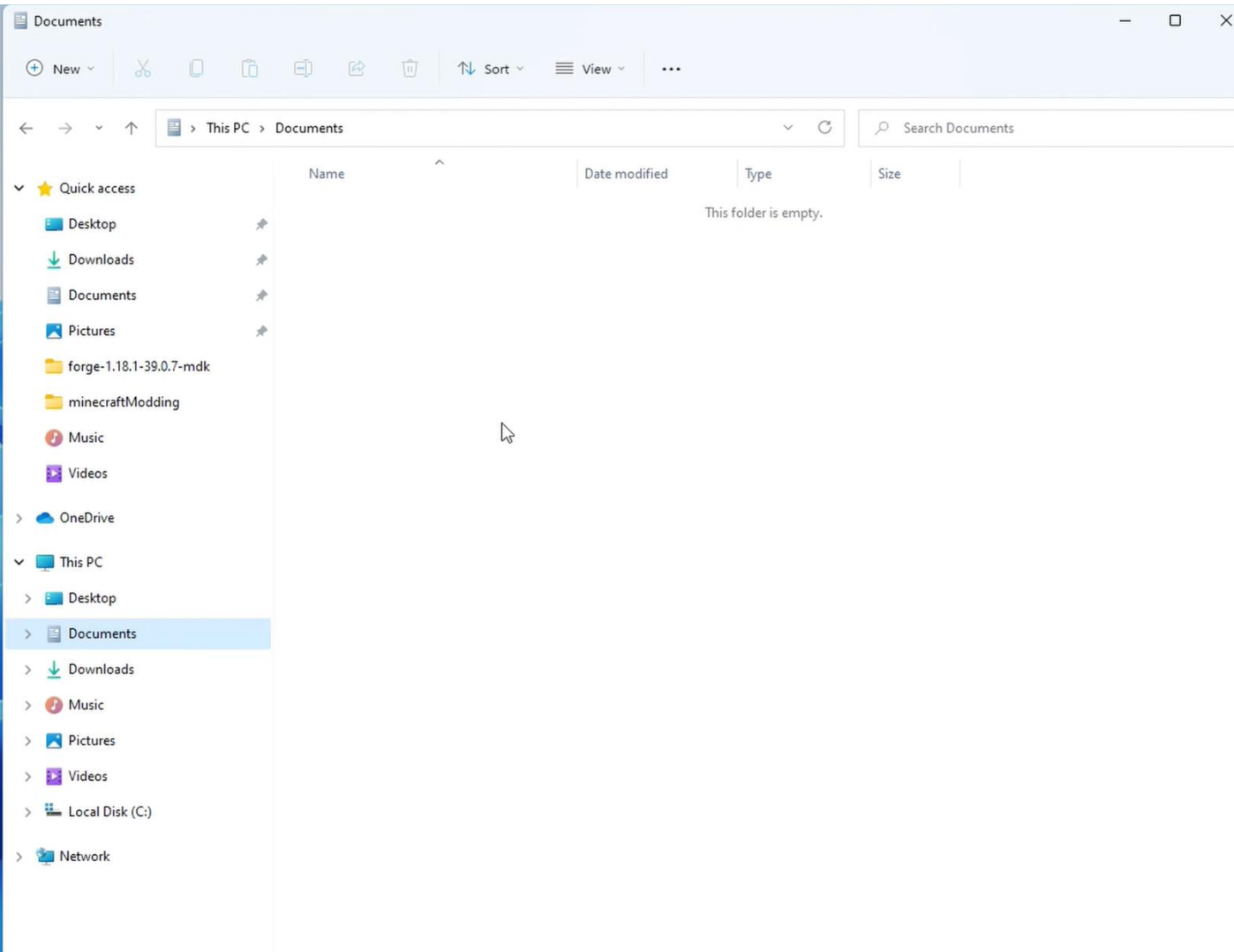
## URL方法:

- ❖ 使用GET方法
- ❖ 上传内容包含在请求行的URL连接中:

<https://cn.bing.com/search?q=chagpt4+价格&form=QBLH&sp=-1&lq=0&pq=chagpt4+%E4%BB%B7%E6%A0%BC&sc=2-10&qsn=&sk=&cvid=7957507DDA5A4C3DA0E439AFD257A56F&ghsh=0&ghacc=0&ghpl=>

# HTTP请求消息:一般格式

软件学院 · 计算机网络



# HTTP响应消息 (重点)

状态行  
(协议  
状态代码  
状态短语)

首部行

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

数据, 例如,  
请求  
HTML文件



- ❖ 状态代码出现在服务器到客户端响应消息的第一行。
- ❖ 一些示例代码:

## 200 OK

- 请求成功，请求的对象在该报文后续内容中

## 301永久移动

- 请求对象已被永久转移，新的地址在该报文后续的Header line 的Location字段中

## 400错误请求

- 一个通用差错代码，请求不能被服务器理解

## 404未找到

- 服务器上找不到请求的文档（对象）

## 505：服务器不支持请求报文使用的HTTP协议版本

## 1.远程登录到您最喜欢的Web服务器:

```
telnet cis.poly.edu 80
```

打开到端口80的TCP连接  
(默认HTTP服务器端口)在cis.poly.edu。  
发送任何输入的内容  
到cis.poly.edu的80号端口

## 2.键入GET HTTP请求:

```
GET /~ross/ HTTP/1.1  
Host:cis.poly.edu
```

通过键入以下内容(按回车  
返回两次), 您发送  
这是最小的(但是完整的)  
获取对HTTP服务器的请求

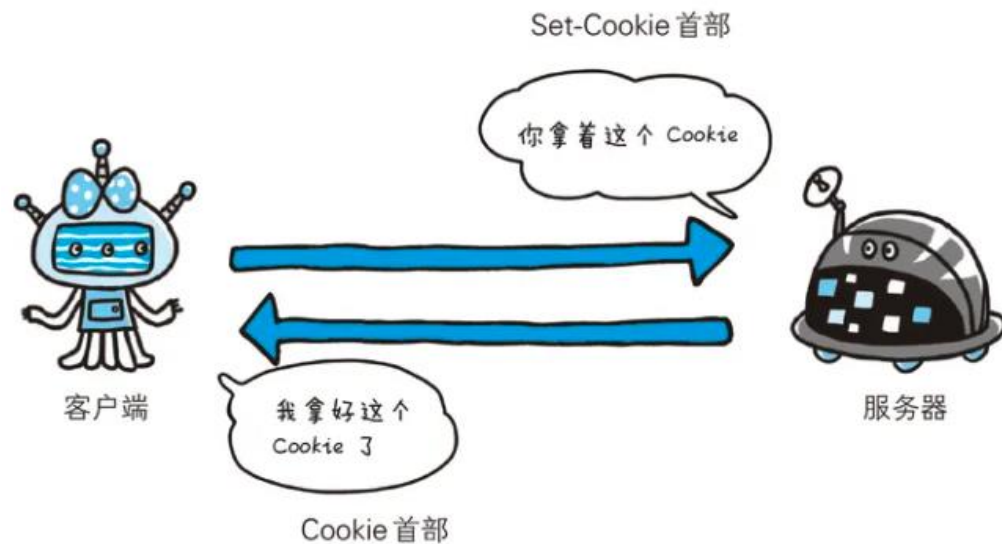
## 3.看HTTP服务器发送的响应消息!

(或者使用Wireshark查看捕获的HTTP请求/响应)

许多网站使用cookies

**四个组件:**

- 1) HTTP响应报文的cookie首部行
- 2) 下一次HTTP请求报文的cookie首部行
- 3) 由浏览器维护的用户主机上



首部字段名	说明	首部类型
Set-Cookie	开始状态管理所使用的Cookie信息	响应首部字段
Cookie	服务器接收到的Cookie信息	请求首部字段

许多网站使用cookies

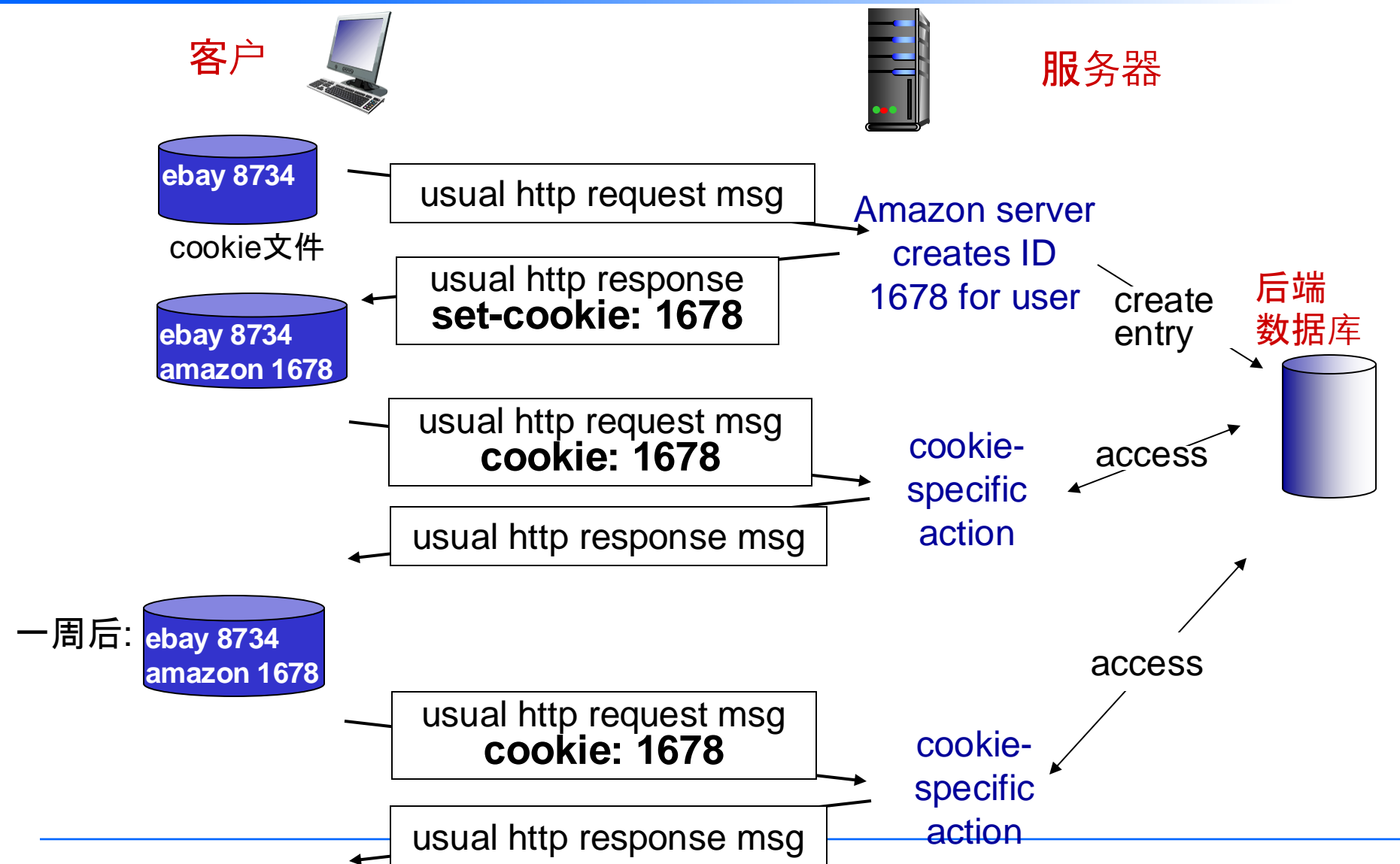
## 四个组件:

- 1) HTTP响应报文的cookie首部行
- 2) 下一次HTTP请求报文的cookie首部行
- 3) 由浏览器维护的用户主机上的cookie文件
- 4) 位于web站点的后端数据库

## 示例:

- ❖ 苏珊总是通过电脑上网
- ❖ 首次访问特定的电子商务网站
- ❖ 当初始HTTP请求到达站点时, 站点创建:
  - 唯一ID
  - ID在后端数据库中的条目

# Cookies:保持“状态” (续)



## 哪些cookies可以用于:

- ❖ 授权、认证
- ❖ 购物车
- ❖ 用户推荐
- ❖ 用户session状态

## cookies和隐私:

- ❖ cookies允许网站了解更多关于你的信息
- ❖ 您可以向站点提供名称和电子邮件

## 如何保持“状态”:

- ❖ 基于协议: 在多个事物的发送/接收报文时维护状态信息
- ❖ cookie: 通过http报文携带状态信息

- P4. 考虑当浏览器发送一个 HTTP GET 报文时，通过 Wireshark 俘获到下列 ASCII 字符串（即这是一个 HTTP GET 报文的实际内容）。字符 `<cr>` `<lf>` 是回车和换行符（即下面文本中的斜体字符串 `<cr>` 表示了单个回车符，该回车符包含在 HTTP 首部中的相应位置）。回答下列问题，指出你在下面 HTTP GET 报文中找到答案的地方。

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

- 由浏览器请求的文档的 URL 是什么？
- 该浏览器运行的是 HTTP 的何种版本？
- 该浏览器请求的是一条非持续连接还是一条持续连接？
- 该浏览器所运行的主机的 IP 地址是什么？
- 发起该报文的浏览器的类型是什么？在一个 HTTP 请求报文中，为什么需要浏览器类型？

- a) The document request was `http://gaia.cs.umass.edu/cs453/index.html`. The Host : field indicates the server's name and `/cs453/index.html` indicates the file name.
- b) The browser is running HTTP version 1.1, as indicated just before the first `<cr><lf>` pair.
- c) The browser is requesting a persistent connection, as indicated by the Connection: keep-alive.
- d) This is a trick question. This information is not contained in an HTTP message anywhere. So there is no way to tell this from looking at the exchange of HTTP messages alone. One would need information from the IP datagrams (that carried the TCP segment that carried the HTTP GET request) to answer this question.
- e) Mozilla/5.0. The browser type information is needed by the server to send different versions of the same object to different types of browsers.



P5. 下面文本中显示的是来自服务器的回答，以响应上述问题中 HTTP GET 报文。回答下列问题，指出你在下面报文中找到答案的地方。

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>
Keep-Alive: timeout=max=100<cr><lf>Connection:
Keep-Alive<cr><lf>Content-Type: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-
//w3c//dtd html 4.0transitional//en"><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1"><lf> <meta
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT
5.0; U) Netscape]"><lf> <title>CMPSCI 453 / 591 /
NTU-ST550ASpring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>
```

- 服务器能否成功地找到那个文档？该文档提供回答是什么时间？
- 该文档最后修改是什么时间？
- 文档中被返回的字节有多少？
- 文档被返回的前 5 个字节是什么？该服务器同意一条持续连接吗？

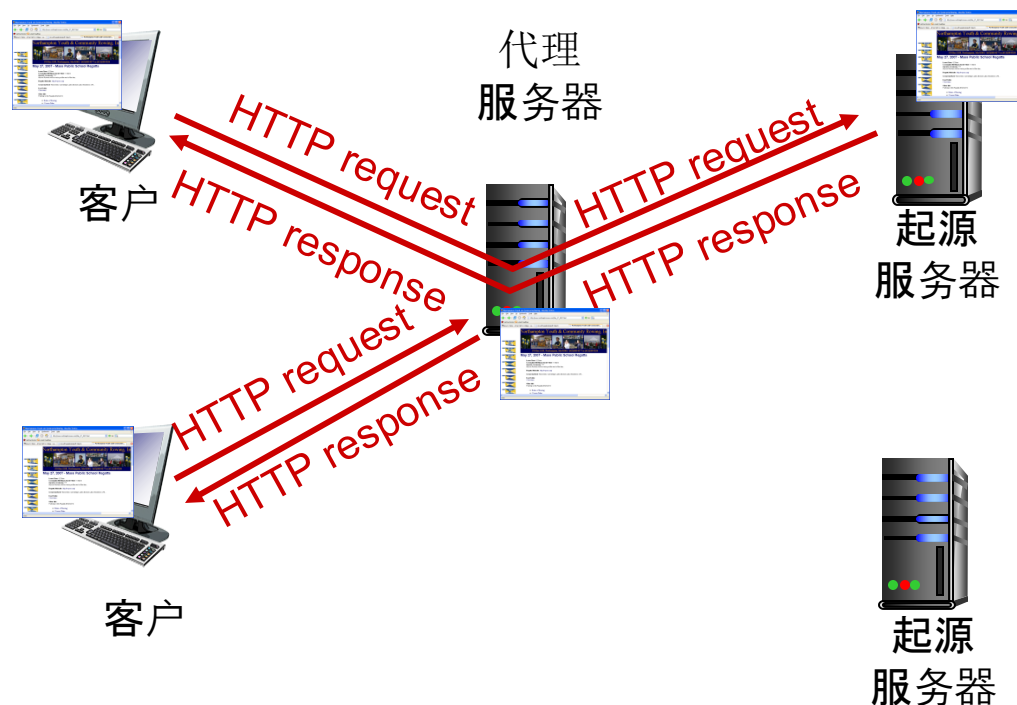
P5. 下面文本中显示的是来自服务器的回答，以响应上述问题中 HTTP GET 报文。回答下列问题，指出你在下面报文中找到答案的地方。

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>
Keep-Alive: timeout=max=100<cr><lf>Connection:
Keep-Alive<cr><lf>Content-Type: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-
//w3c//dtd html 4.0transitional//en"><lf><html><lf>
<head><lf> <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1"><lf> <meta
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT
5.0; U) Netscape]"><lf> <title>CMPSCI 453 / 591 /
NTU-ST550ASpring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>
```

- 服务器能否成功地找到那个文档？该文档提供回答是什么时间？
- 该文档最后修改是什么时间？
- 文档中被返回的字节有多少？
- 文档被返回的前 5 个字节是什么？该服务器同意一条持续连接吗？

*目标:在不涉及源服务器的情况下满足客户端请求*

- ❖ 用户设置浏览器: Web 访问经由代理服务器
- ❖ 客户端发送所有的 http 请求到代理服务器
  - 代理服务器保存了请求的对象: 代理服务器返回请求的对象
  - 否则代理服务器从原始服务器请求对象,再将其返回给客户端



## 为什么需要Web缓存?

- ❖ 代理服务器同时具备了客户端和服务器的角色
  - 对于原始的客户端请求而言，代理服务器扮演服务器的角色
  - 而对于原始服务器而言，代理服务器则是客户端
- ❖ 通常缓存由ISP(大学、公司、住宅ISP)安装
- ❖ 响应时间较短:缓存与客户端比较“接近”
- ❖ 减少了往来与远程服务器间的数据流量，因为从学校或本地ISP通往外部的链路往往是网络瓶颈
- ❖ 使得服务较匮乏的提供商可以更高效的发布信息（P2P文件共享与此类似）

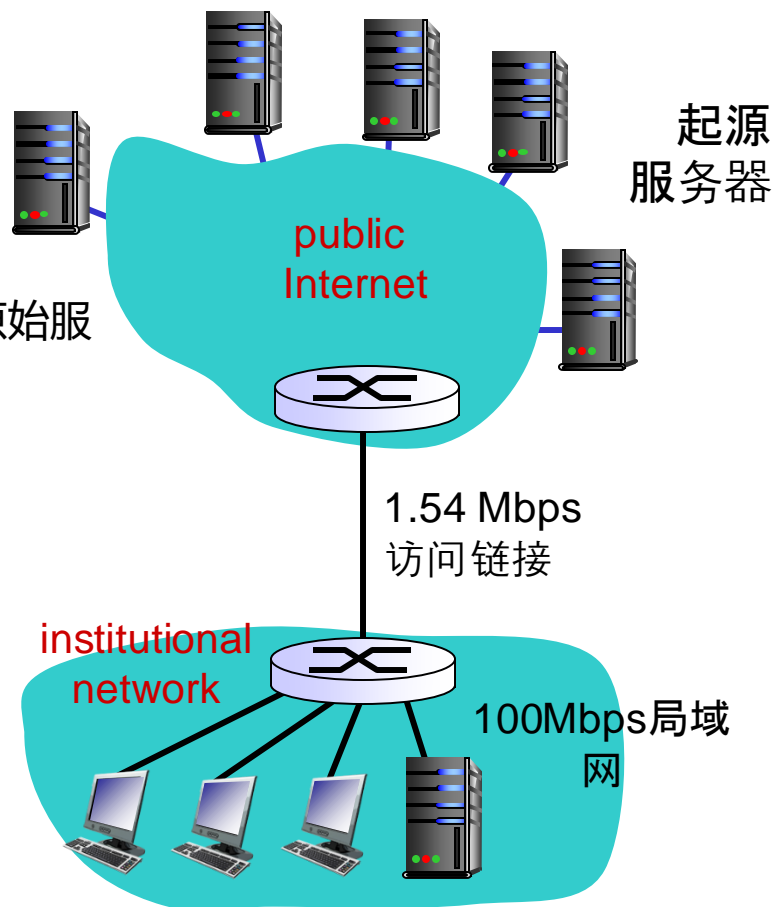
# 缓存示例:

## 假设:

- ❖ 每个对象的平均大小为: 100Kb
- ❖ 从浏览器到原始服务器的请求量为: 15/sec
- ❖ 到浏览器的平均速率为: 1.50Mbps
- ❖ 从因特网接入链路一侧的路由器转发报文到任意原始服务器的RTT为: 2sec
- ❖ 接入链路的速率: 1.54Mbps

## 结论:

- ❖ 局域网的使用率 (流量强度) =  $(15 \text{ 个请求/s}) \times (100 \text{ Kb/请求}) / (100 \text{ Mbps}) = 1.5\%$
- ❖ 接入链路使用率 (流量强度) =  $(15 \text{ 个请求/s}) \times (100 \text{ Kb/请求}) / (1.54 \text{ Mbps}) = 99\%$
- ❖ 总时延 = 因特网时延 + 接入时延 + LAN时延 = 2sec + 分钟级 + 微秒级



# 缓存示例:胖访问链接

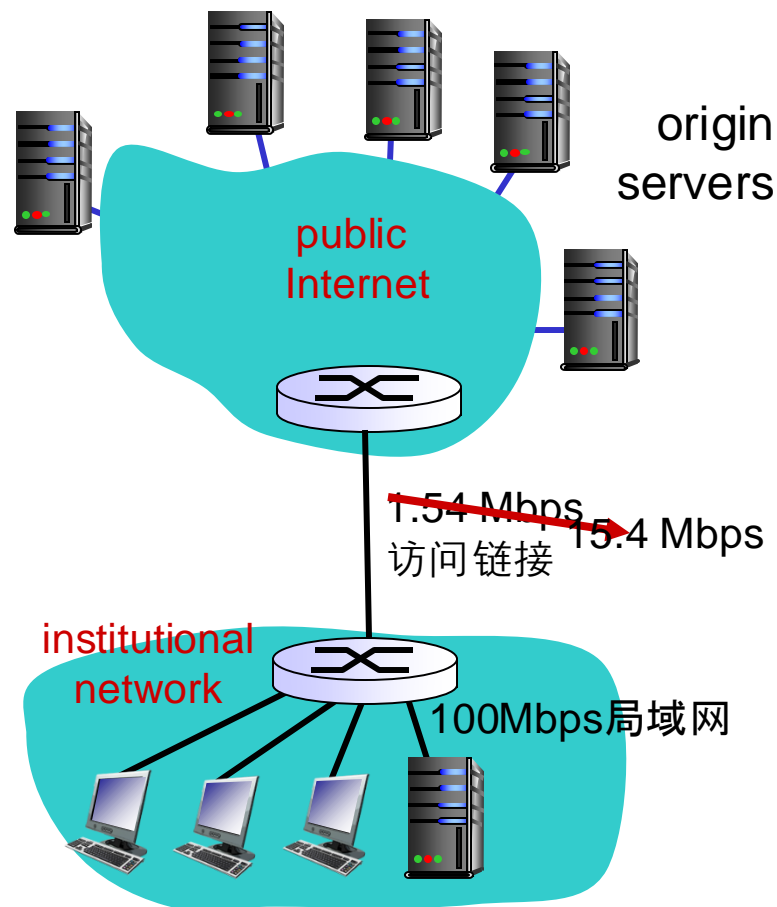
## 假设:

- ❖ 平均对象大小:每个100K
- ❖ 从浏览器到源服务器的平均请求速率:15/秒
- ❖ 从机构路由器到任何源服务器的RTT:2秒
- ❖ 接入链路速率:~~1.54 Mbps~~

## 结论:

- ❖ 局域网流量强度:0.015
  - ❖ 入口强度 ~~$\approx 1$~~   $\rightarrow$  0.099
  - ❖ 总延迟 = 互联网延迟 + 访问延迟 + 局域网延迟
- ~~= 2秒 + 分钟 + 秒~~
- 毫秒

15.4 Mbps



成本:接入链路速度提高(不便宜!)

# 缓存示例:安装本地缓存

## 假设:

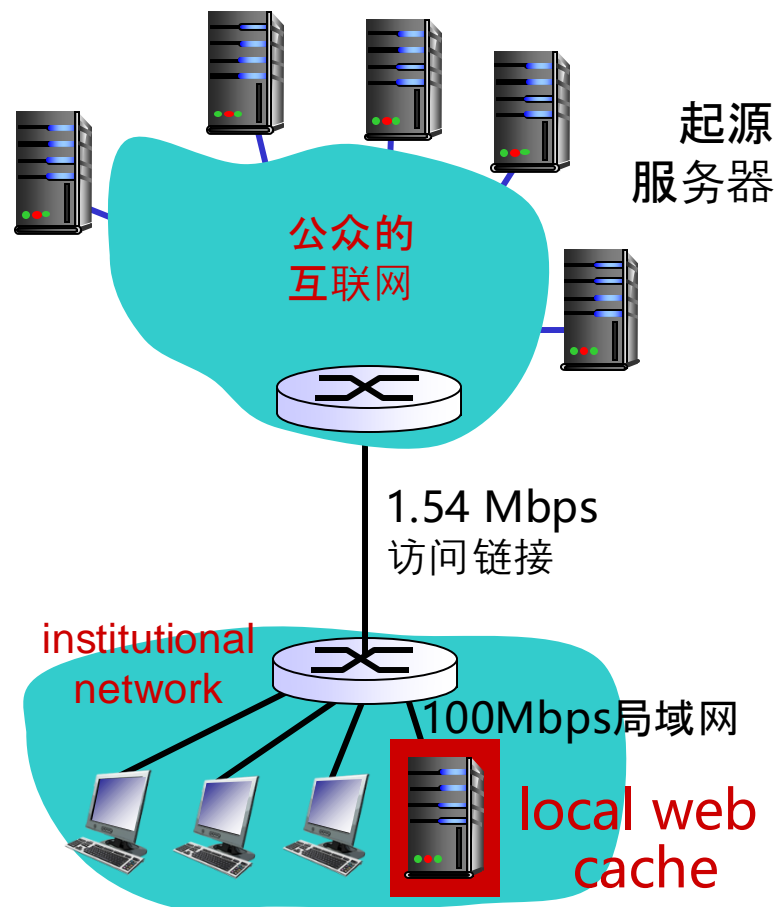
- ❖ 平均对象大小:100K
- ❖ 从浏览器到源服务器的平均请求速率:15req/秒
- ❖ 从机构路由器到任何源服务器的RTT:2秒
- ❖ 接入链路速率:1.54 Mbps

## 后果:

- ❖ 局域网流量强度:0.015
- ❖ 访问流量强度 = **100%**
- ❖ 总延迟 = 互联网延迟 + 访问延迟 + 局域网延迟

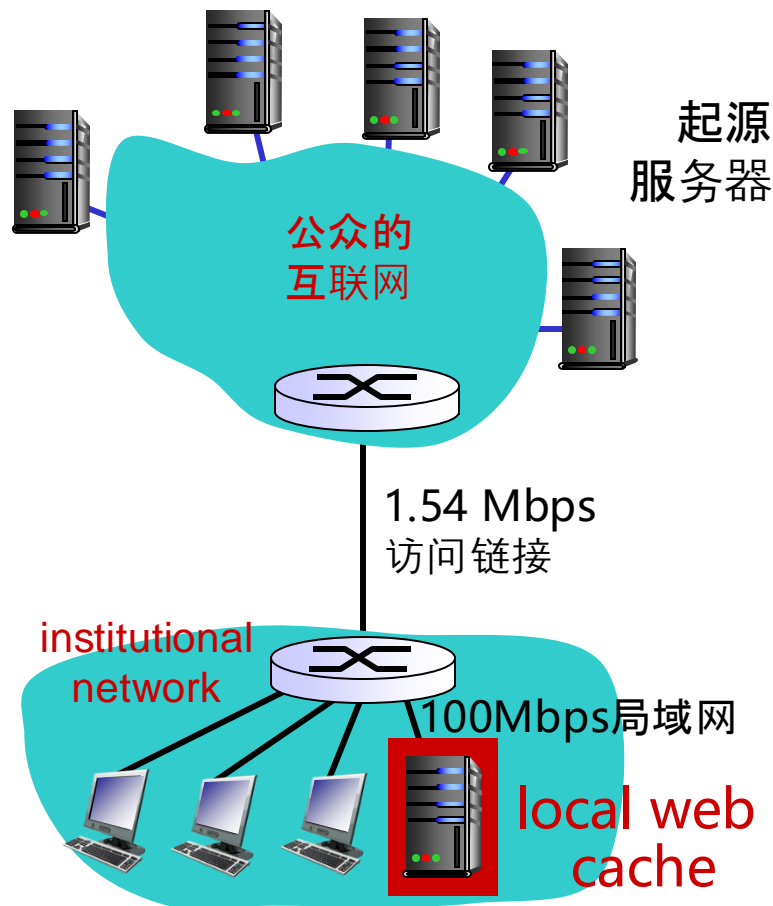
如何计算链接  
利用率, 延迟?

成本:web cache(便宜!)



## 计算访问链路利用率、缓存延迟:

- ❖ 假设缓存命中率为0.4
  - 在缓存中满足40%的请求, 在源中满足60%的请求
- ❖ 接入链路利用率:
  - 60%的请求使用访问链接
- ❖ 通过接入链路到浏览器的数据速率 =  $0.6 * 100K * 15/\text{秒} = 0.9 \text{ Mbps}$ 
  - 流量强度 =  $0.9 / 1.54 = 0.58$
- ❖ 总延迟
  - =  $0.6 * (\text{来自原始服务器的延迟}) + 0.4 * (\text{缓存满足时的延迟})$
  - =  $0.6(2.01) + 0.4(\sim \text{微秒})$
  - =  $\sim 1.2 \text{ 秒}$
  - 低于15.4 Mbps的链接(也更便宜! )





缓存/客户端



服务器

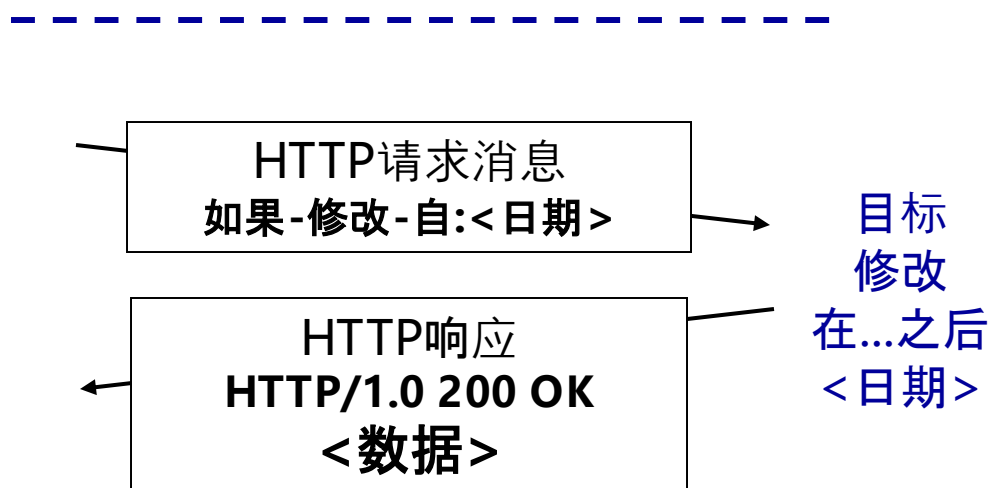
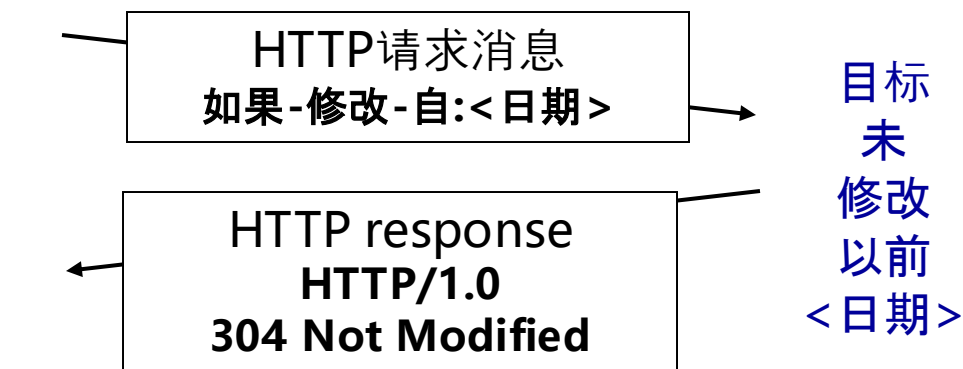
- **目标:** 如果客户端缓存了最新的请求对象, 则服务器不必重复发送

- 减去了对象传输的时延
- 降低了链路的流量强度

• 代理服务器 (客户端): 在http请求报文中声明所缓存拷贝的生成日期

If-modified-since: <date>

- 服务器: 如果客户端缓存的拷贝是最新的, 则在响应报文中不发请求的对象:
- HTTP/1.0 304 Not Modified



# HTTP vs. HTTPS

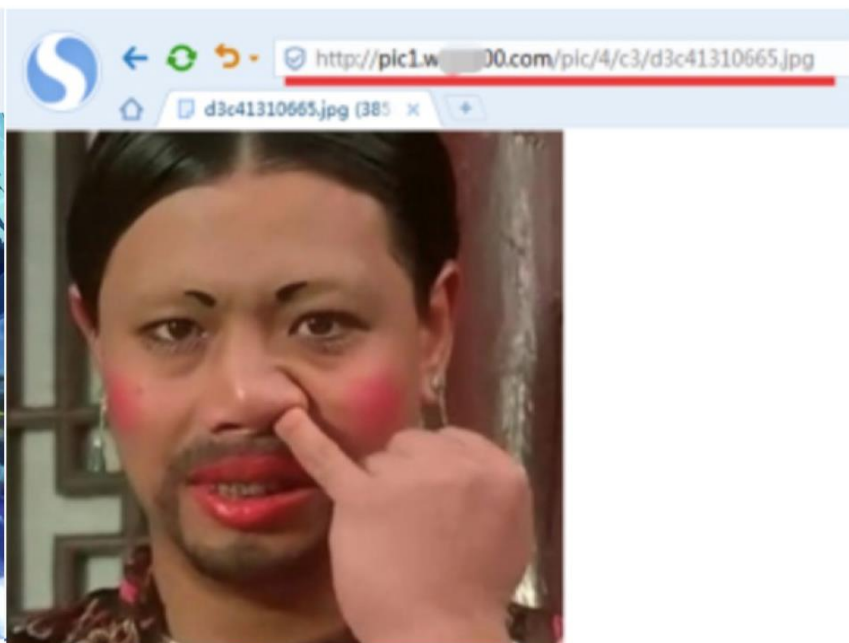
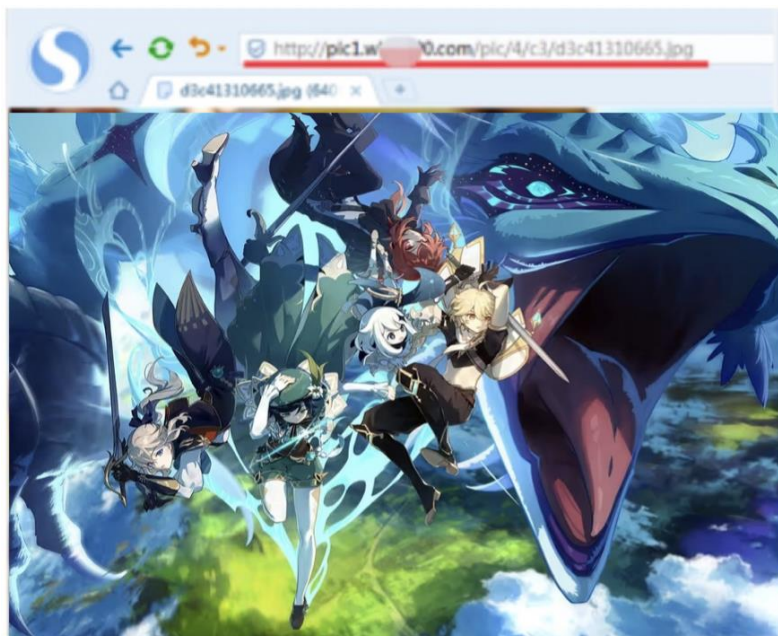
正在捕获 本地连接

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

ip.addr==183.158.100.100

TCP三次握手

No.	Time	Source	Destination	Protocol	Length	Info
273	3.035865	10.0.2.137	183.158.100.100	TCP	74	53119→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
275	3.038219	183.158.100.100	10.0.2.137	TCP	74	80→53119 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1360 SA
276	3.038261	10.0.2.137	183.158.100.100	TCP	66	53119→80 [ACK] Seq=1 Ack=1 Win=66052 Len=0 TSval=2878487 TS
277	3.039414	10.0.2.137	183.158.100.100	HTTP	493	GET /pic/4/c3/d3c41310665.jpg HTTP/1.1 客户端发送请求内容
278	3.041236	183.158.100.100	10.0.2.137	TCP	66	80→53119 [ACK] Seq=1 Ack=428 Win=30720 Len=0 TSval=17407877



1 2.1 应用层协议原理

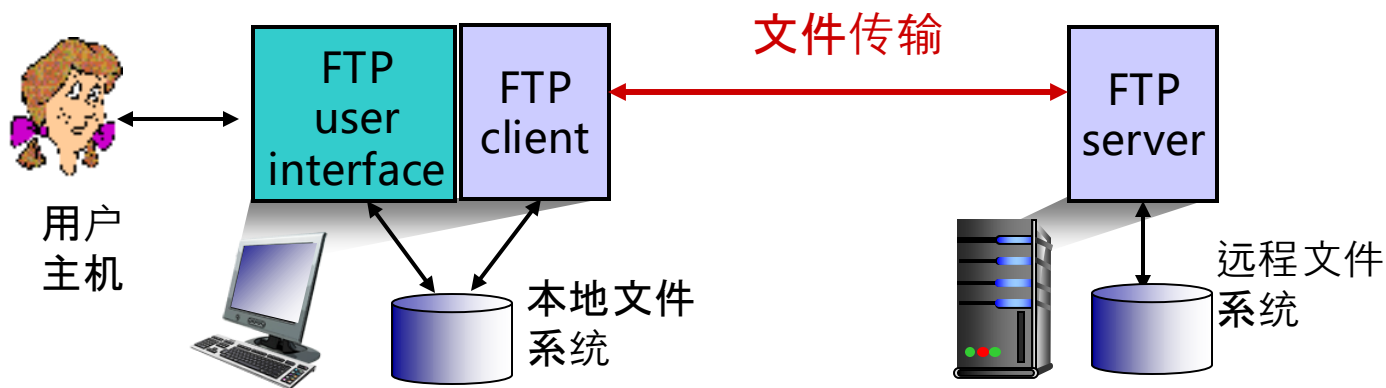
2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

5 2.5 DNS

6 2.6 P2P 应用



- ❖ 实现文件到远程服务器的上传、下载
- ❖ 客户机/服务器模式
  - **客户端**:启动传输(无论与往来远程主机)
  - **服务器**:远程主机
- ❖ ftp: RFC 959

- ❖ FTP客户端通过TCP协议的连接FTP服务器的21端口
- ❖ 客户端认证基于控制连接
- ❖ 客户端浏览远程服务器的目录，发送控制命令等也是基于控制连接
- ❖ 当服务器收到文件传输命令后，**服务器**开发另一个到客户端的TCP数据连接
- ❖ 文件传输结束后，服务器关闭该连接



- ❖ 当有其他文件传输时，服务器均会开启一个TCP数据连接
- ❖ 可以说，FTP的控制信息是“**带外**”传送的
- ❖ FTP 维持状态（state）：当前目录、先前的认证信息等

## 命令示例:

- ❖ 通过控制信道以ASCII文本形式发送
- ❖ 用户用户名
- ❖ 传递密码
- ❖ LIST返回当前目录中的文件列表
- ❖ RETR文件名检索(获取)文件
- ❖ STOR文件名将文件存储(上传)到远程主机上

## 样本返回代码

- ❖ 状态代码和短语(与HTTP类似)
- ❖ 331用户名正常，需要密码
- ❖ 125数据连接已经打开；传输开始
- ❖ 425无法打开数据连接
- ❖ 452写入文件时出错

1 2.1 应用层协议原理

2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

5 2.5 DNS

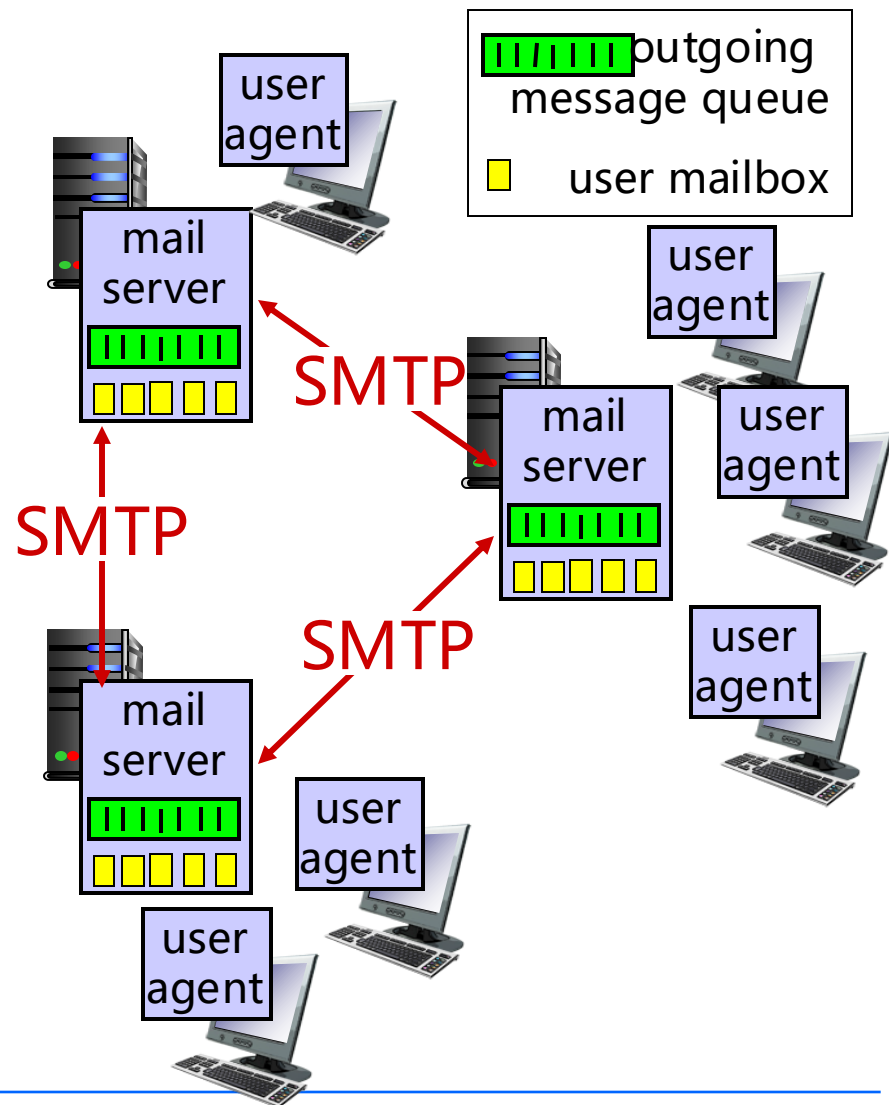
6 2.6 P2P 应用

## 三个主要组件:

- ❖ 用户代理
- ❖ 邮件服务器
- ❖ 简单邮件传输协议:SMTP

## 用户代理人 (UA)

- ❖ 亦称之为“邮件阅读器”
- ❖ 写作、编辑、阅读邮件报文
- ❖ 例如: Outlook、Thunderbird、iPhone mail client
- ❖ 发送、接收的报文存储在邮件服务器上





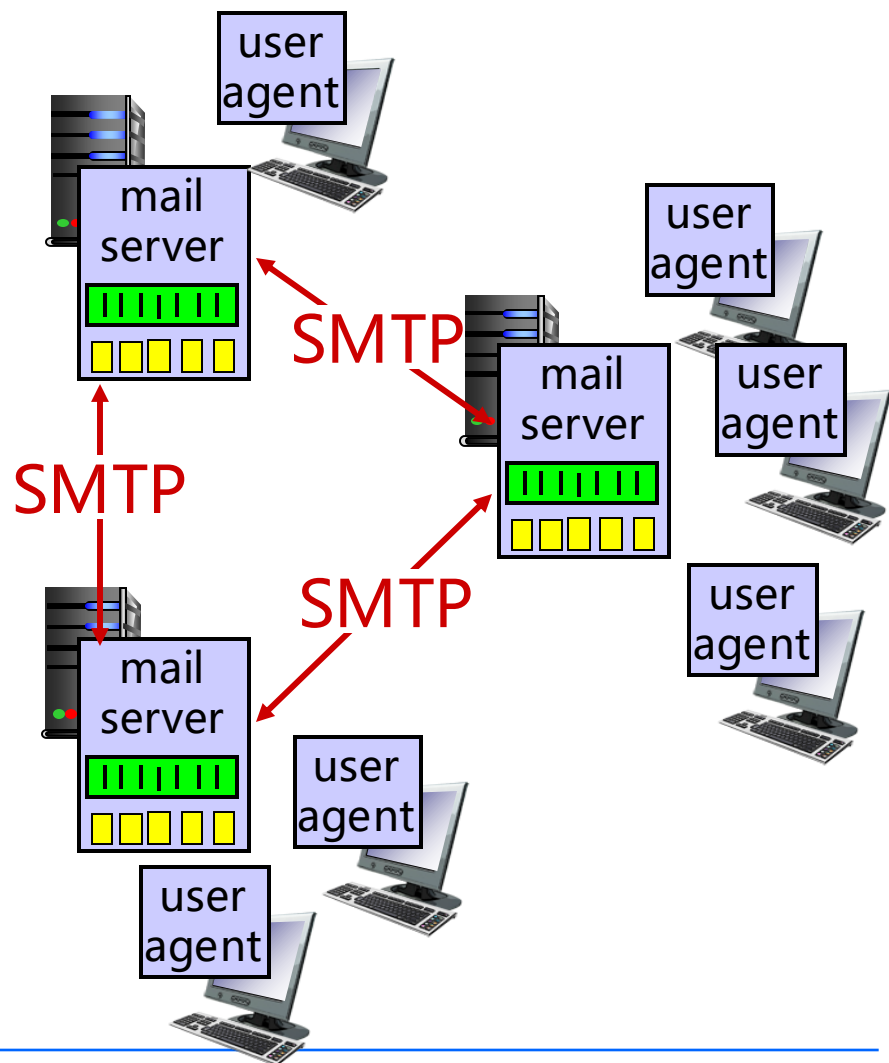
邮件服务器:

邮箱: 包含了收到的用户邮件

报文队列: 包含了要发送的邮件  
报文

SMTP协议: 用于邮件服务器之间  
发送邮件

- 客户端: 将邮件发送到邮件服务器
- 服务器: 接收和转发邮件

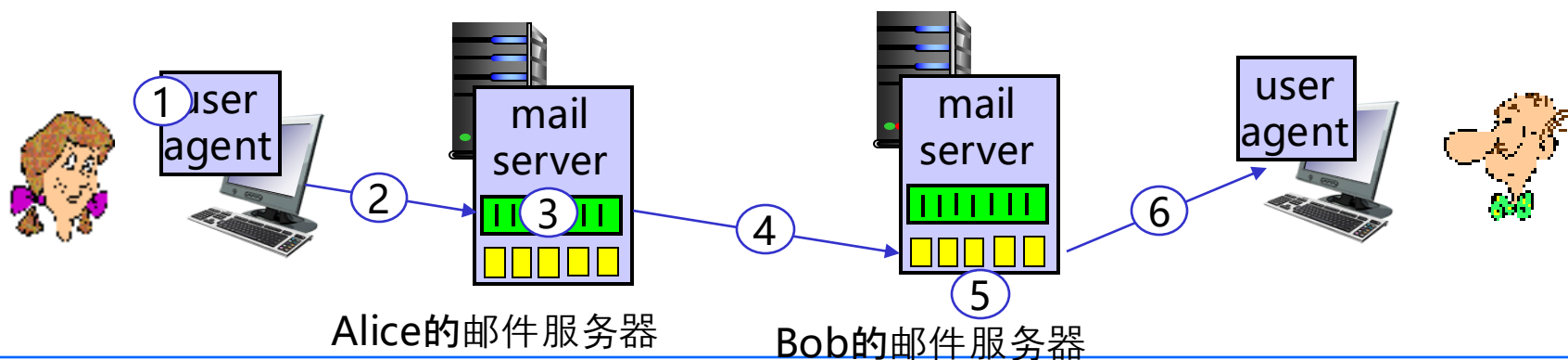


- ❖ 使用 **TCP** 可靠的传送邮件报文, 端口25
- ❖ 直接传输: 发送服务器到接收服务器
- ❖ 传输的三个阶段
  - 握手(打招呼)
  - 报文传输
  - 结束
- ❖ 命令/响应交互
  - **命令**: ASCII文本
  - **响应**: 状态码和短语
- ❖ 邮件报文必须使用7-bit ASCII表示

# 场景:Alice给Bob发邮件

- 1) Alice使用UA编写” 给Bob  
“bob@someschool.edu的消息
- 2) Alice的UA向她的邮件服务器发送  
消息；放置在消息队列中的消息
- 3) SMTP的客户端打开与Bob的邮件  
服务器的TCP连接

- 4) SMTP客户端通过TCP连接发  
送Alice的消息
- 5) Bob的邮件服务器将消息放入  
Bob的邮箱中
- 6) Bob调用他的用户代理来读取  
消息



```
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
```

如果发送邮件服务器有几个报文发往同一个接收邮件服务器，**它可以通过同一TCP连接发送这些所有的报文**。对每个报文，该客户用一个新的MAIL FROM: crpes.fr开始，用一个独立的句点行指示该邮件的结束，并且仅当所有邮件发送完后才发送QUIT。

## 与HTTP相似性

- ❖ SMTP使用持续连接
- ❖ SMTP请求报文 (首部 & 信体) 全部使用 7-bit ASCII码
- ❖ SMTP服务器用 CRLF.CRLF 表示邮件报文的结束
- 某些代码组合不允许出现在报文中 (e.g., CRLF.CRLF). 此类数据必须进行编码 (通常使用 **base-64** 或 quoted printable)

**base64**是为了将任意字符串编码转换成只由26个大小写英文字母、0到9、+和/组成的字符串的一种 编码方式， $26*2+10+2$  等于64，所以叫base64

## 与 HTTP的对比:

- ❖ HTTP: pull (拉)
- ❖ SMTP: push (推)
- ❖ 都使用 ASCII 命令/响应交互, 状态码
- ❖ HTTP: 每个对象分装在各自己的响应报文中
- ❖ SMTP: 多个对象在一个多分部的报文中传送

- ❖ **telnet 202.117.80.4 25 测试 (西工大邮件服务器mail.nwpu.edu.cn)**
- ❖ 见220服务器回复
- ❖ 输入HELO, MAIL FROM, RCPT TO, DATA, QUIT 命令

上面的操作可以让你不使用电子邮件客户端(阅读器)发送电子邮件

SMTP: 交换邮件报文的协议

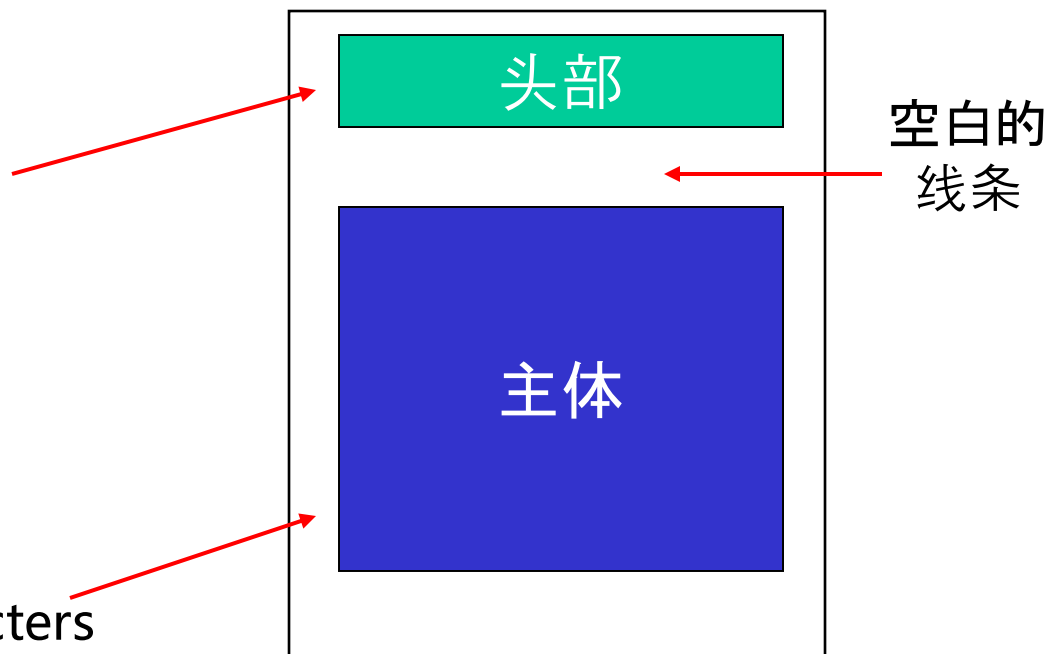
RFC 822-5322: 文本报文格式标准

❖ 首部行, e.g.,

- To:
- From:
- Subject:
- 不同 于 smtp 命令

❖ 信体

- 即 “报文”, ASCII characters only



MIME:多用途互联网邮件扩展, RFC 2045, 2046, 2049

- ❖ MIME版本
- ❖ 内容传输编码
- ❖ 内容类型:类型/子类型 ; 因素
  - 文本
  - 图像
  - 声音的
  - 录像
  - 应用

来自:alice@crepes.fr  
发往:bob@hamburger.edu  
主题:图片。

**MIME**版本:1.0

内容传输编码:base64

内容类型:图像/jpeg

base64编码数据.....

.....

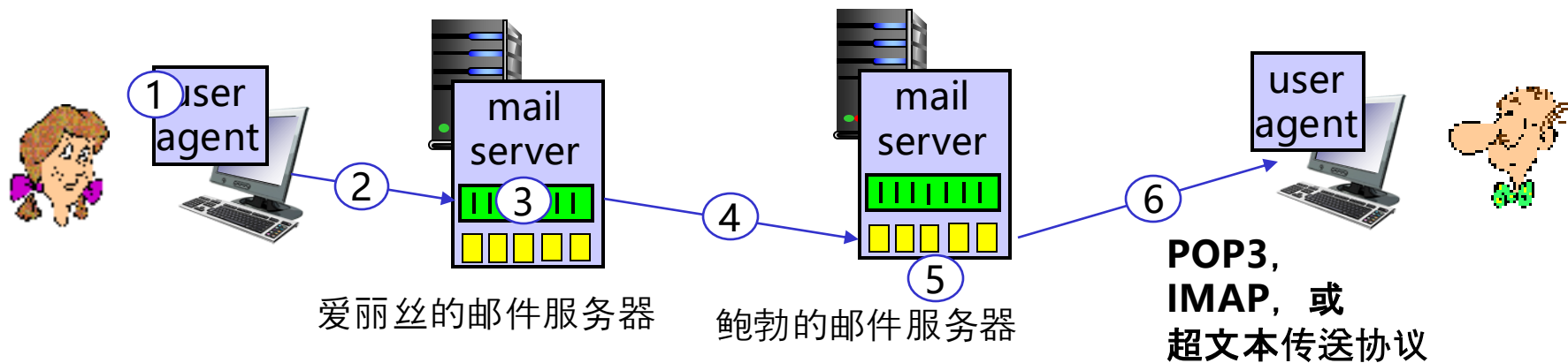
.....base64编码数据



POP3:邮局协议—第3版

IMAP:互联网邮件访问协议

HTTP:基于网络的电子邮件



## 授权阶段

- ❖ 客户端命令:
  - 用户:声明用户名
  - 通过:密码
- ❖ 服务器响应
  - +OK

## 传输阶段, 客户:

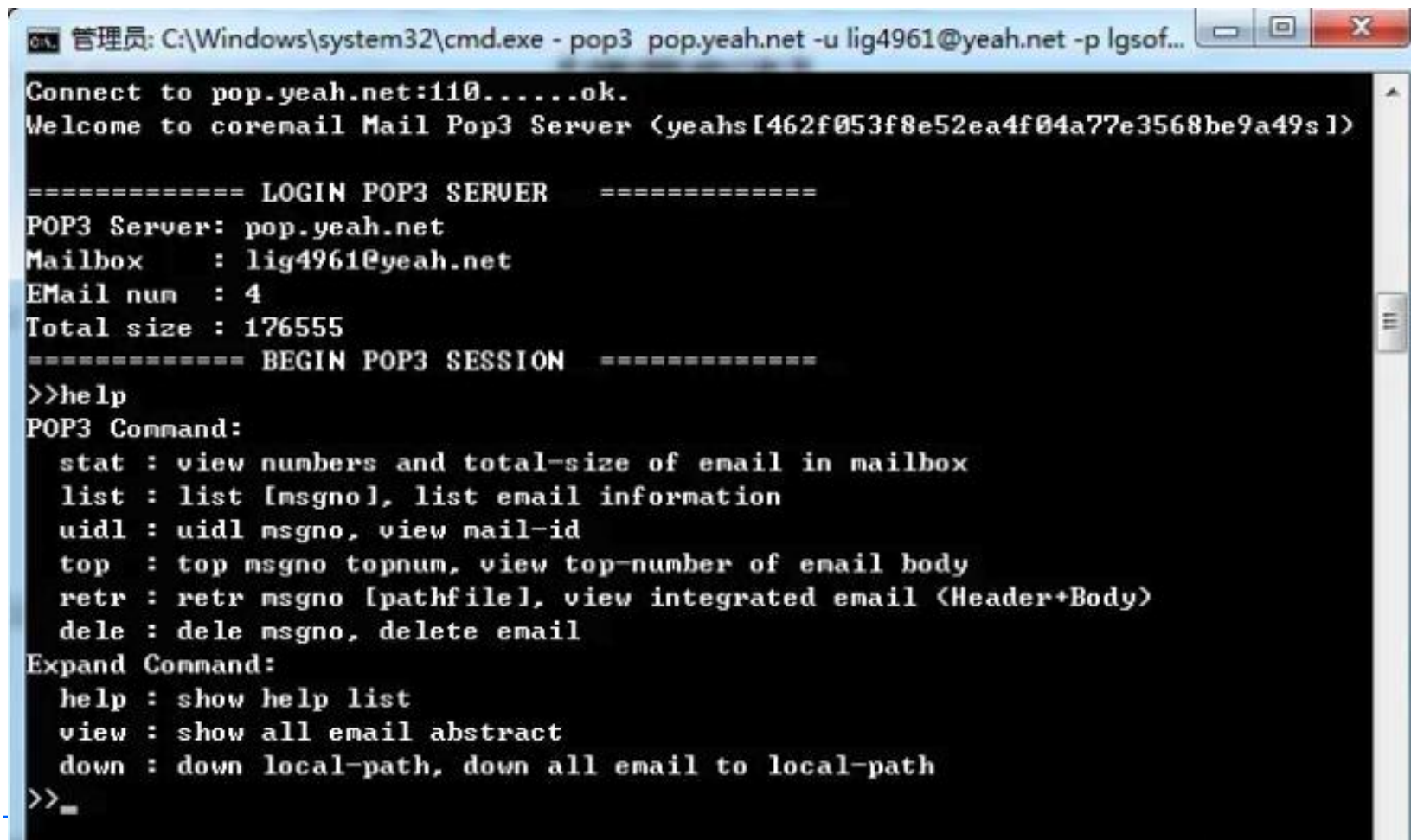
- ❖ 列表:列出消息编号
- ❖ retr:读取相应消息
- ❖ dele:删除
- ❖ quit:退出

## 更新阶段

```
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
```

telnet pop.yeah.net 110



```
管理员: C:\Windows\system32\cmd.exe - pop3 pop.yeah.net -u lig4961@yeah.net -p lgsof...
Connect to pop.yeah.net:110.....ok.
Welcome to coremail Mail Pop3 Server <yeahs[462f053f8e52ea4f04a77e3568be9a49s1]>

===== LOGIN POP3 SERVER =====
POP3 Server: pop.yeah.net
Mailbox      : lig4961@yeah.net
EMail num    : 4
Total size   : 176555
===== BEGIN POP3 SESSION =====
>>help
POP3 Command:
  stat : view numbers and total-size of email in mailbox
  list : list [msgno], list email information
  uidl : uidl msgno, view mail-id
  top   : top msgno topnum, view top-number of email body
  retr  : retr msgno [pathfile], view integrated email <Header+Body>
  dele  : dele msgno, delete email
Expand Command:
  help : show help list
  view  : show all email abstract
  down  : down local-path, down all email to local-path
>>_
```

## 关于POP3的更多信息

- ❖ 上述示例采用POP3 “下载并删除” 模式（在quit时，将retr的邮件删除）
  - Bob在其他客户端无法重读邮件
- ❖ POP3 “下载并保留” 模式：可在不同的客户端获得报文的拷贝
- ❖ POP3服务器并不在POP3会话过程中携带状态信息（但在服务器端维护少量状态，如哪些邮件以retr，在quit时将其删除）

## IMAP: Internet Mail Access Protocol, Internet邮件访问协议

- ❖ 将所有的报文保留在邮件服务器上
- ❖ 允许用户以目录形式组织邮件
- ❖ 在会话过程中保持用户状态
  - 例如：文件夹的名字，哪些报文与哪些文件夹相关联

1 2.1 应用层协议原理

2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

5 2.5 DNS

6 2.6 P2P 应用

自然人: 诸多标识:

- 身份证, 姓名, 护照

因特网主机, 路由器:

- IP 地址 (32 bit) – 用于数据报寻址
- “域名”, e.g.,  
www.nwpu.edu.cn – 帮助记忆

Q: IP 地址和域名之间如何映射(转换)?

Domain Name System:

- ❖ 分布式数据库: 由许多域名服务器按层次构成
- ❖ 应用层协议: 主机、路由器、域名服务器互相通信进行域名解析 (地址/域名翻译)
  - 注意: 因特网之核心功能, 应用层之协议
  - 网络 “边缘” 上之复杂实体

## DNS服务

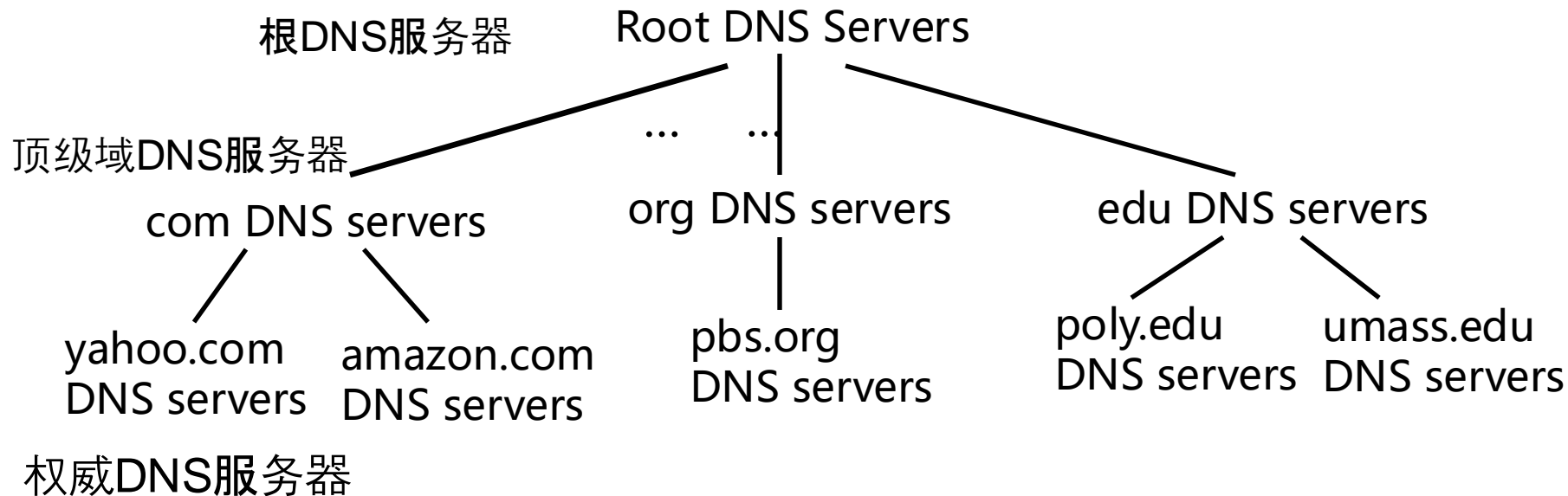
- ❖ 主机名与IP地址之间的映射
- ❖ 主机别名
  - 规范的别名
- ❖ 邮件服务器别名
- ❖ 负载均衡
  - 冗余Web服务器:  
多个IP地址同时对应一个域名

## 集中式DNS存在的问题

- ❖ 单点故障
- ❖ 通信容量
- ❖ 远距离集中式数据库
- ❖ 维护

*A: 没有可扩展能力*

# DNS：分布式、层次结构的数据库



*客户要取得www.amazon.com的IP地址:*

- ❖ 先请求根域名服务器，得到com域的DNS服务器地址;
- ❖ 再请求.com域的DNS服务器，得到amazon.com DNS服务器
- ❖ 最后在请求amazon.com DNS服务器，得到www.amazon.com的IP地址



## 顶级域(TLD)服务器:

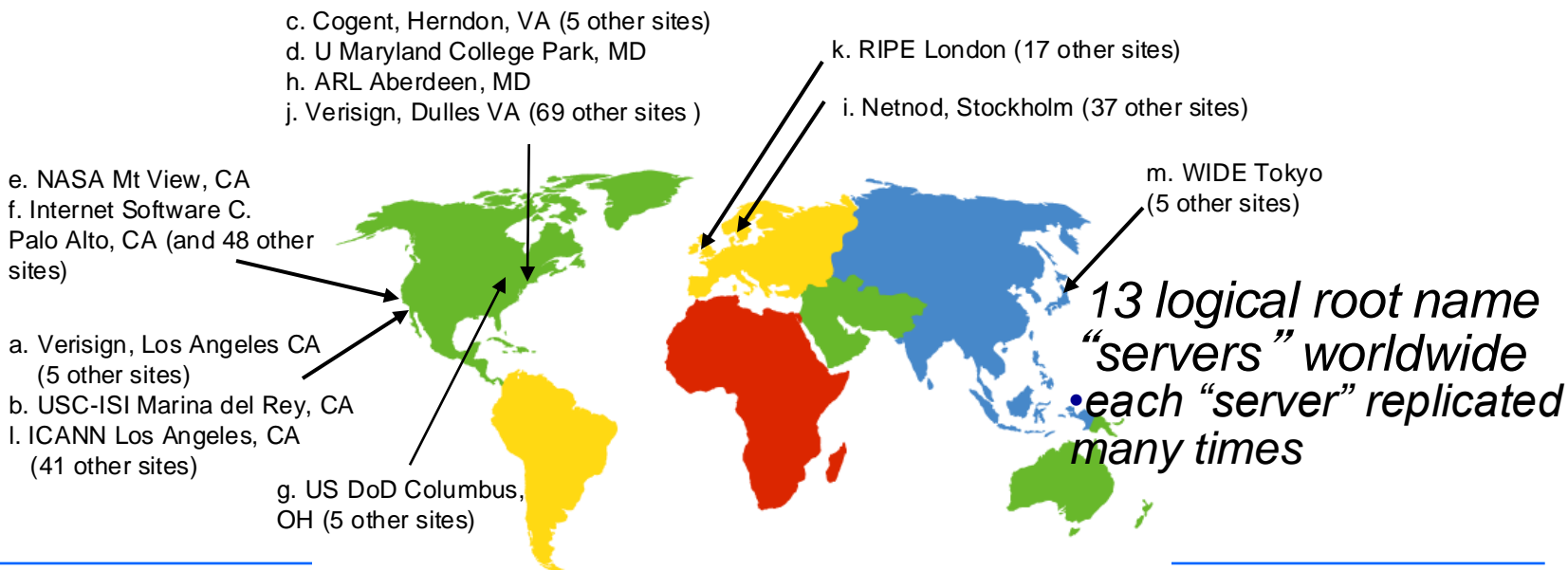
- 负责顶级域名以及所有国家顶级域名的解析: 例如 .com、.net 或 url 中最后一个点之后的任何内容
- Verisign Global Registry Services公司负责维护.com顶级域服务器
- Educause 公司负责维护.edu顶级域服务器

## 权威DNS服务器:

- 因特网上具有公共可访问主机的每个组织机构必须提供可访问的DNS服务器, 实现自己的主机与IP地址之间的映射
- 亦可付费维护在其他组织或服务提供商的权威DNS服务器上

- ❖ 严格来讲，本地DNS服务器并不属于DNS服务器层级结构中
- ❖ 每个ISP都提供一台本地DNS服务器
  - 亦称之为“默认域名服务器”
- ❖ 当查询DNS时，请求首先被发送到本地域名服务器
  - 若本地缓存中有域名-地址的解析，则直接返回结果
  - 否则，请求域名服务器层级结构，如前所述

- ❖ 当本地域名服务器不能解析时，就向根域名服务器查询
- ❖ 根域名服务器：
  - 如果域名映射未知，则向授权域名服务器查询
  - 取得查询结果，将查询结果返回本地域名服务器

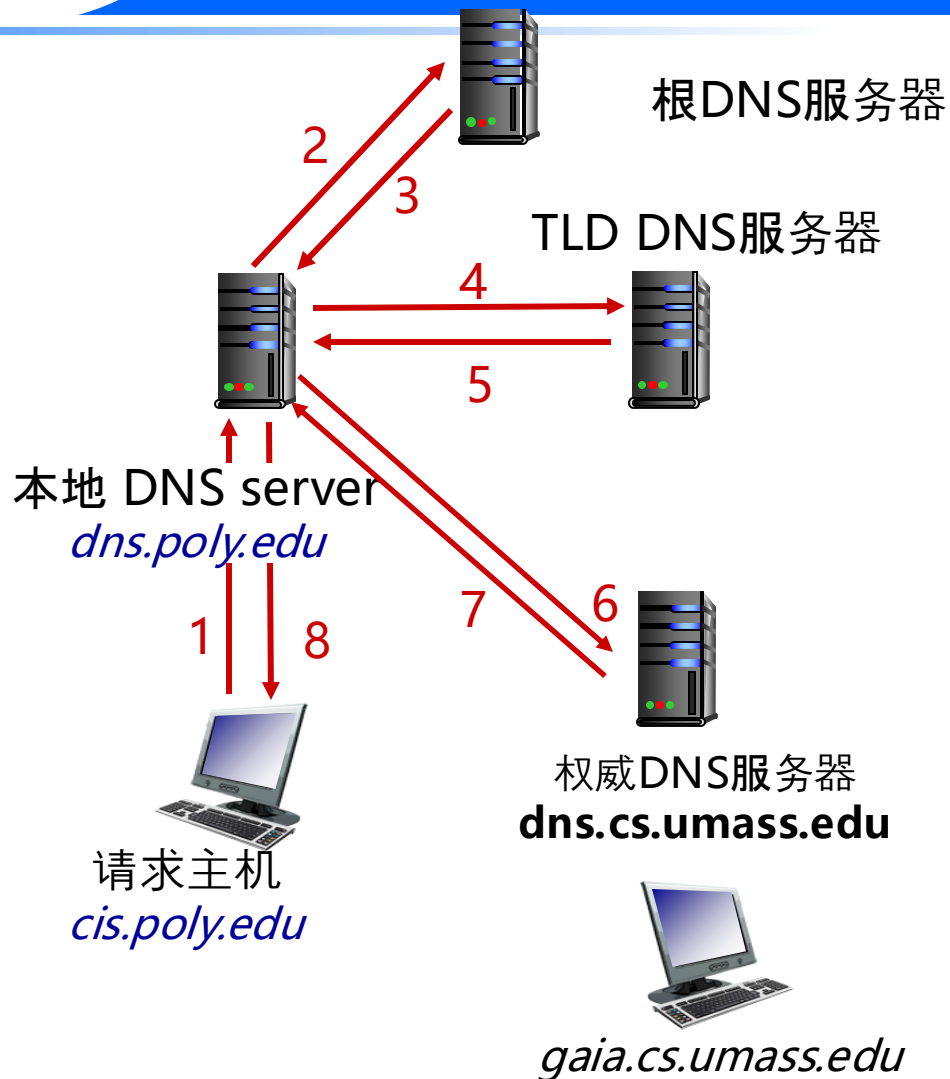


# DNS名称解析案例

- ❖ cis.poly.edu的主机需要gaia.cs.umass.edu的IP地址

## 迭代查询:

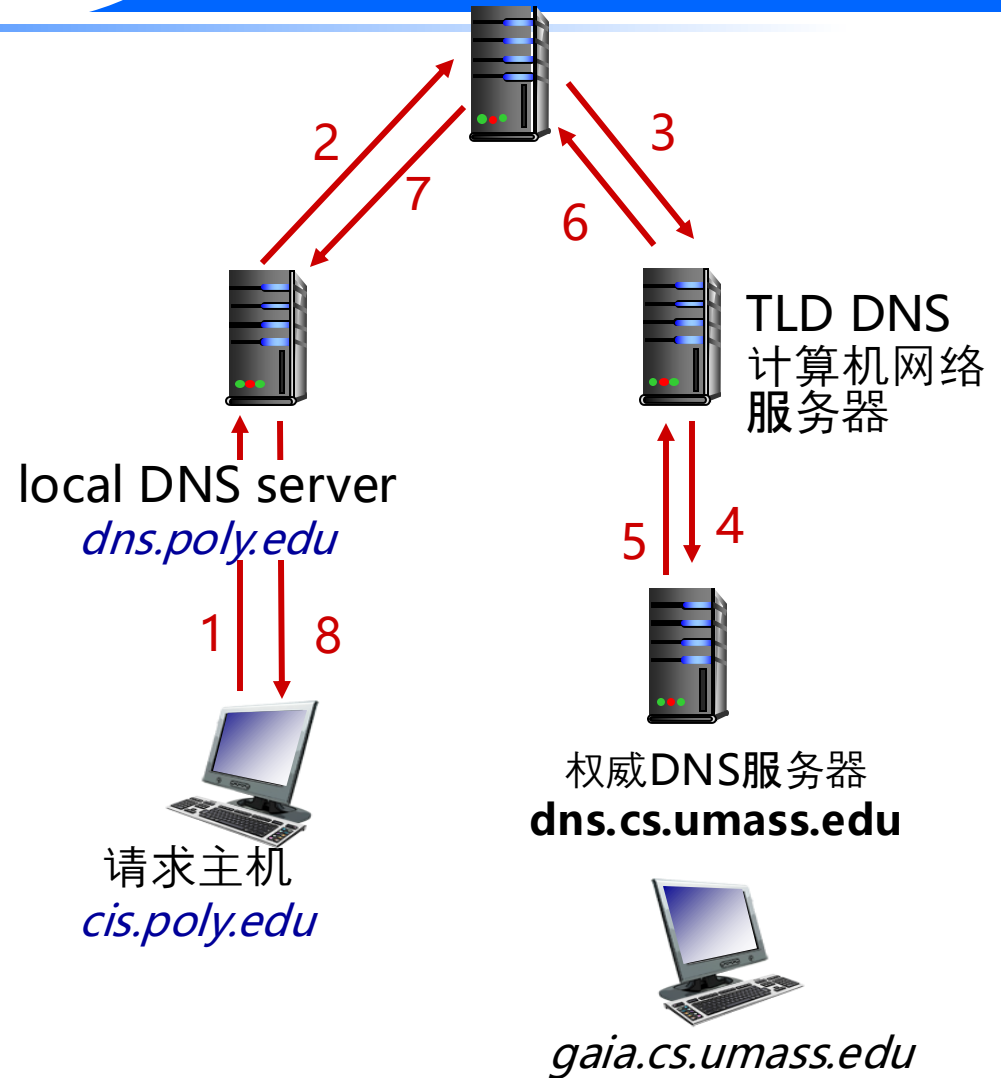
- ❖ 被查询的服务器直接把可查询的服务器地址报回
- ❖ “不知道这个域名, 但可以从这个服务器查到”



# DNS名称解析示例

## 递归查询:

- ❖ 将域名解析的负担放在联系的域名服务器上
- ❖ 对根域名服务器造成工作负担



- ❖ 一旦 (任何) 域名服务器得知了某个映射, 就将其缓存
  - 在一定的时间间隔后缓存的条目将会过期(自动消除)
  - 本地域名服务器通常会缓存顶级域名服务器地址, 因此, 根DNS服务器其实并不被经常访问
- ❖ 缓存的记录可能会~~过期~~ (失效)
  - 如果某域名变更了IP地址, 则直至TTL失效, 整个因特网才能知道
- ❖ 更新/通知 机制由 IETF负责设计
  - RFC 2136

*DNS: 本质上以一个存储资源纪录 (RR)的分布式数据库*

RR格式:(名称、值、类型、ttl)

## 类型=A

- 名称是主机名
- 值是IP地址

## 类型=NS

- 名称是域名(例如, foo.com)
- 值是该域的权威名称服务器的主机名

## 类型=CNAME

- 名称是别名
- 值是规范名称

- 例如www.ibm.com真的是  
类型=MX  
servereast.backup2.ibm.com
- 与上面类似, 但用于邮件服务器

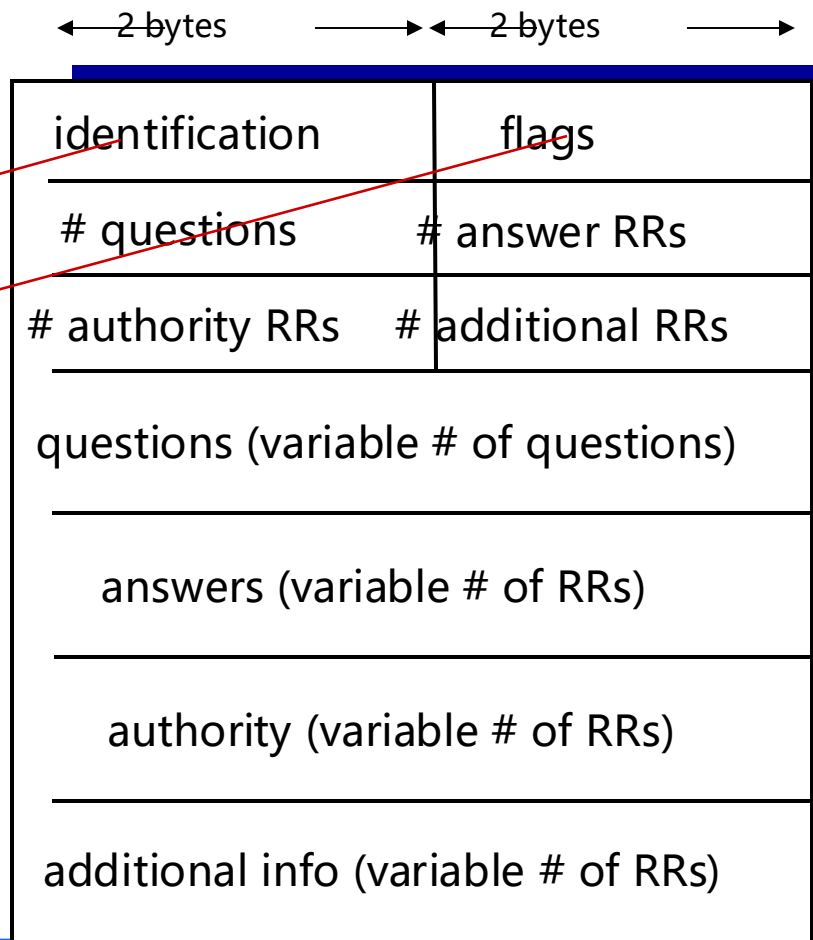
❖ 查询和应答报文, 格式相同

## 报文首部

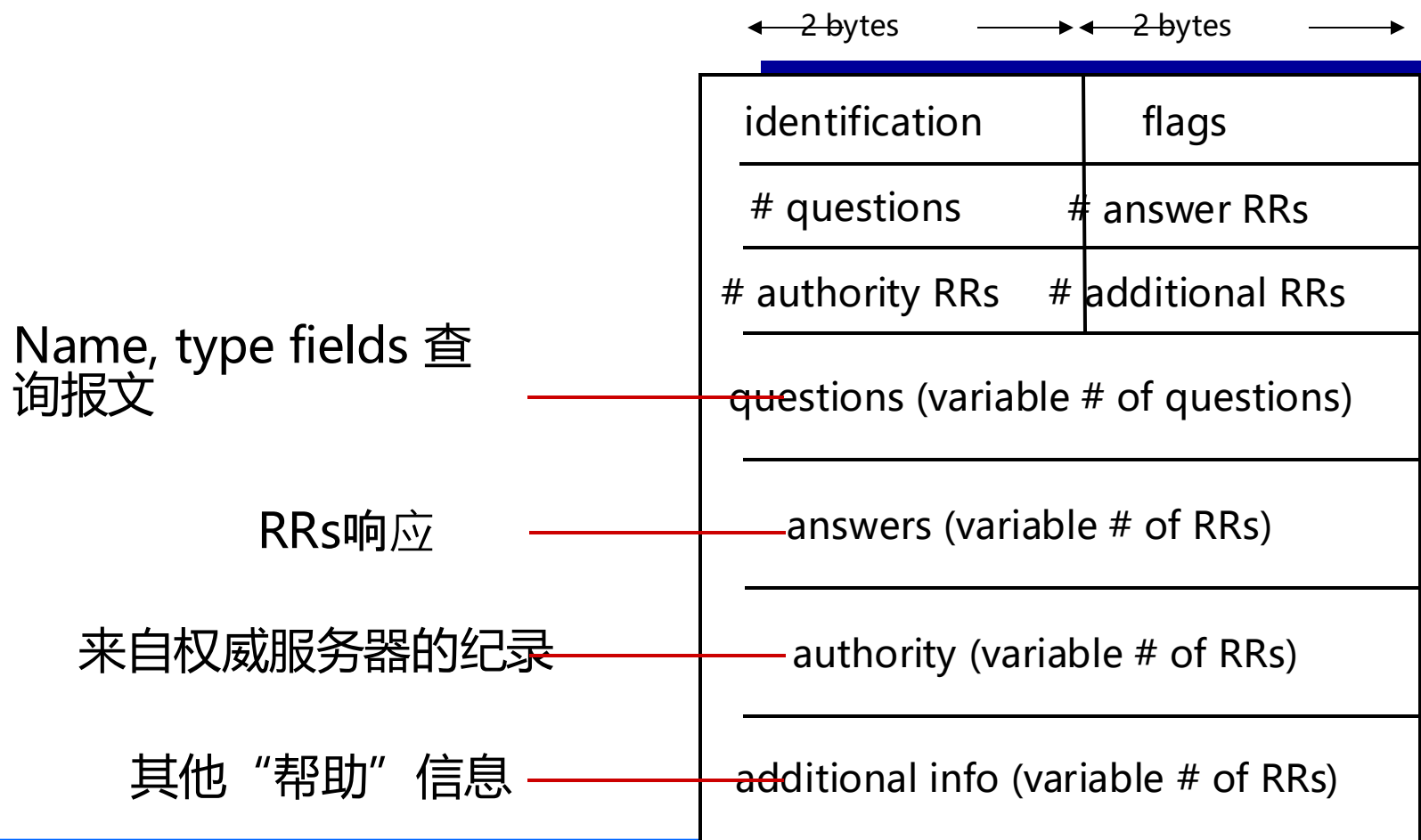
❖ identification: 16 bit # 用于查询, 应答报文使用同样的 #

❖ flags:

- 查询 或 应答
- 希望递归
- 可以递归
- 授权应答







- ❖ 例如，新成立了Network Utopia公司
- ❖ 需要注册networkuptopia.com域名
  - 提供权威域名服务器的域名和IP地址
  - 在.com域服务器上新增两个RRs
- ❖ 创建权威DNS服务器，增加type A记录和type MX 记录

1 2.1 应用层协议原理

2 2.2 Web 和 HTTP

3 2.3 FTP

4 2.4 电子邮件（SMTP, POP3, IMAP）

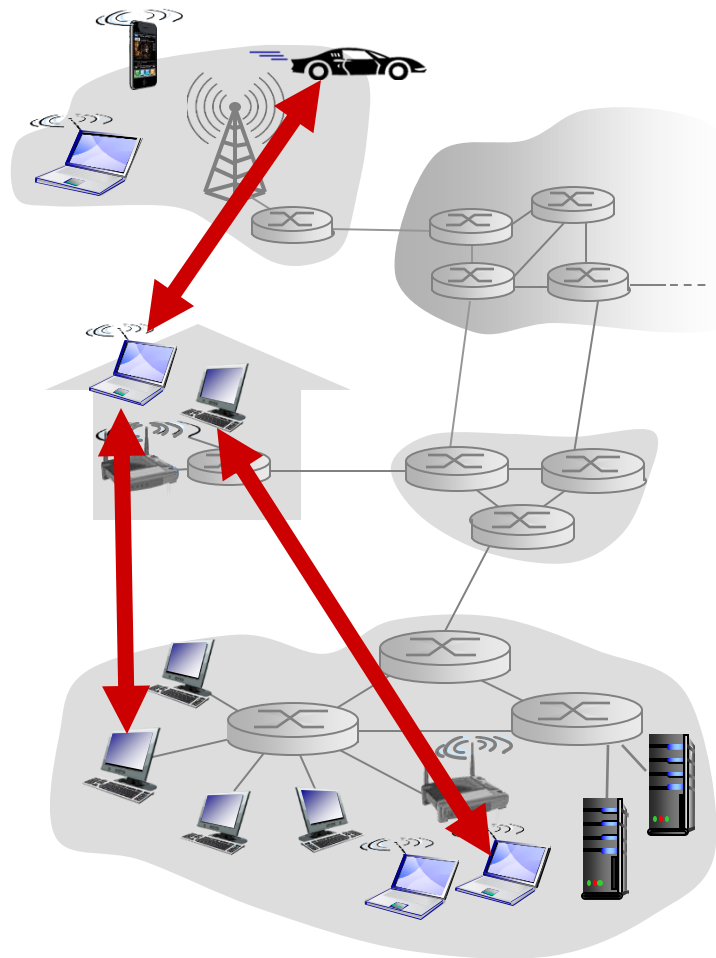
5 2.5 DNS

6 2.6 P2P 应用

- ❖ 没有始终运行着的服务器
- ❖ 任意的端系统（称之为对等方）之间直接通信
- ❖ 对等方之间间歇性的互联，其IP地址也可能改变

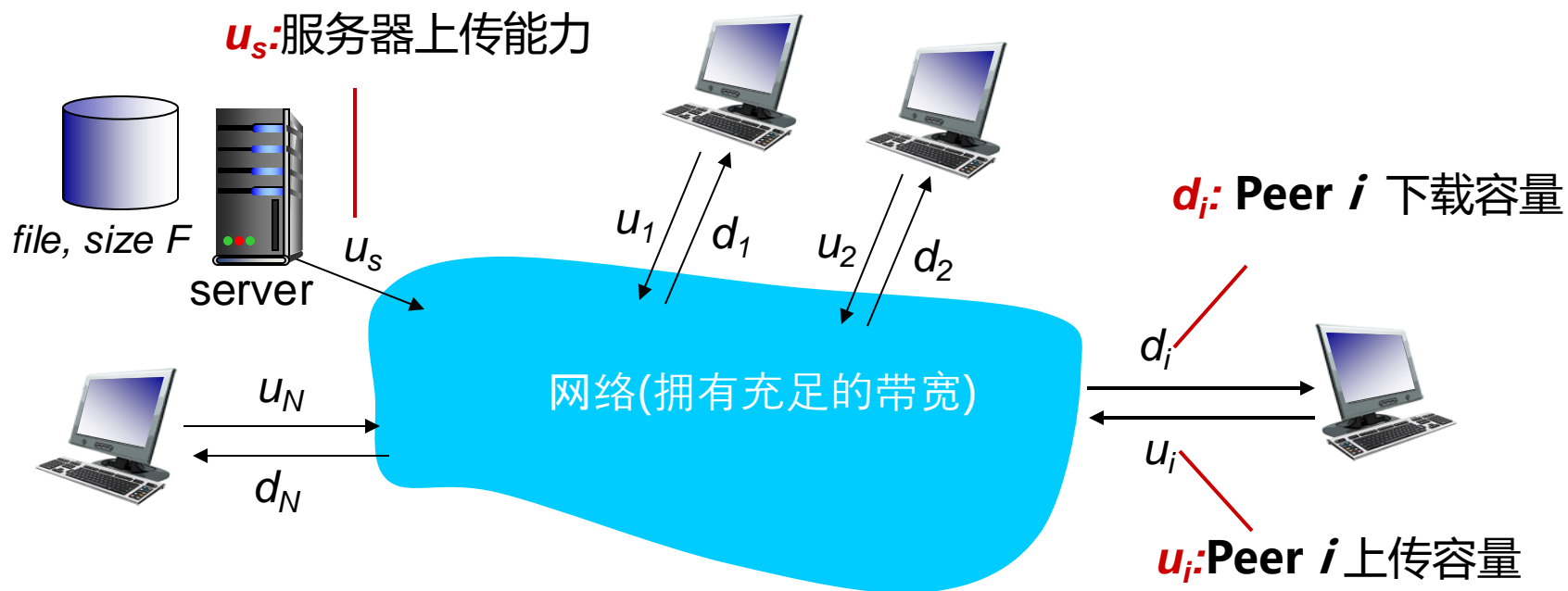
## 示例:

- 文件分发
- 流媒体
- VoIP



Q: 将文件 (大小为 $F$ ) 从服务器分发到 $N$ 个节点, 需要多少时间

- 每个节点上传/下载的数据容量是有限的



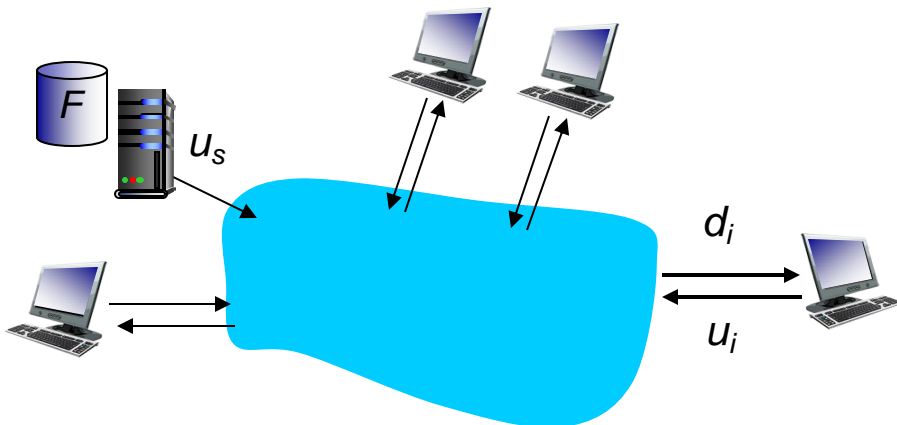
# 文件分发时间: 客户-服务器模式

❖ **服务器处理:** 必须顺序的发送N个拷贝

- 发送一份拷贝的时间: $F/u_s$
- 发送N份拷贝的时间: $NF/u_s$

❖ **客户端:** 每个客户端下载文件拷贝

- $d_{\min}$  = 最小客户端下载速率
- 最长客户端下载时间: $F/d_{\min}$

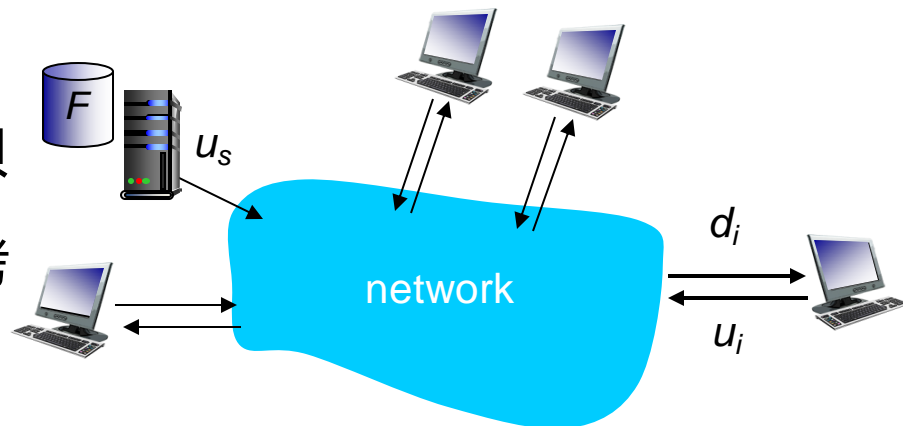


分配F的时间到N个客户端, 使用  
客户端-服务器方法

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

随着N线性增加

- **服务器处理**: 至少上传一个文件拷贝
- **客户端**: 每个客户端下载一个文件拷贝
- 全部**客户端**总下载量为  $NF$  bits



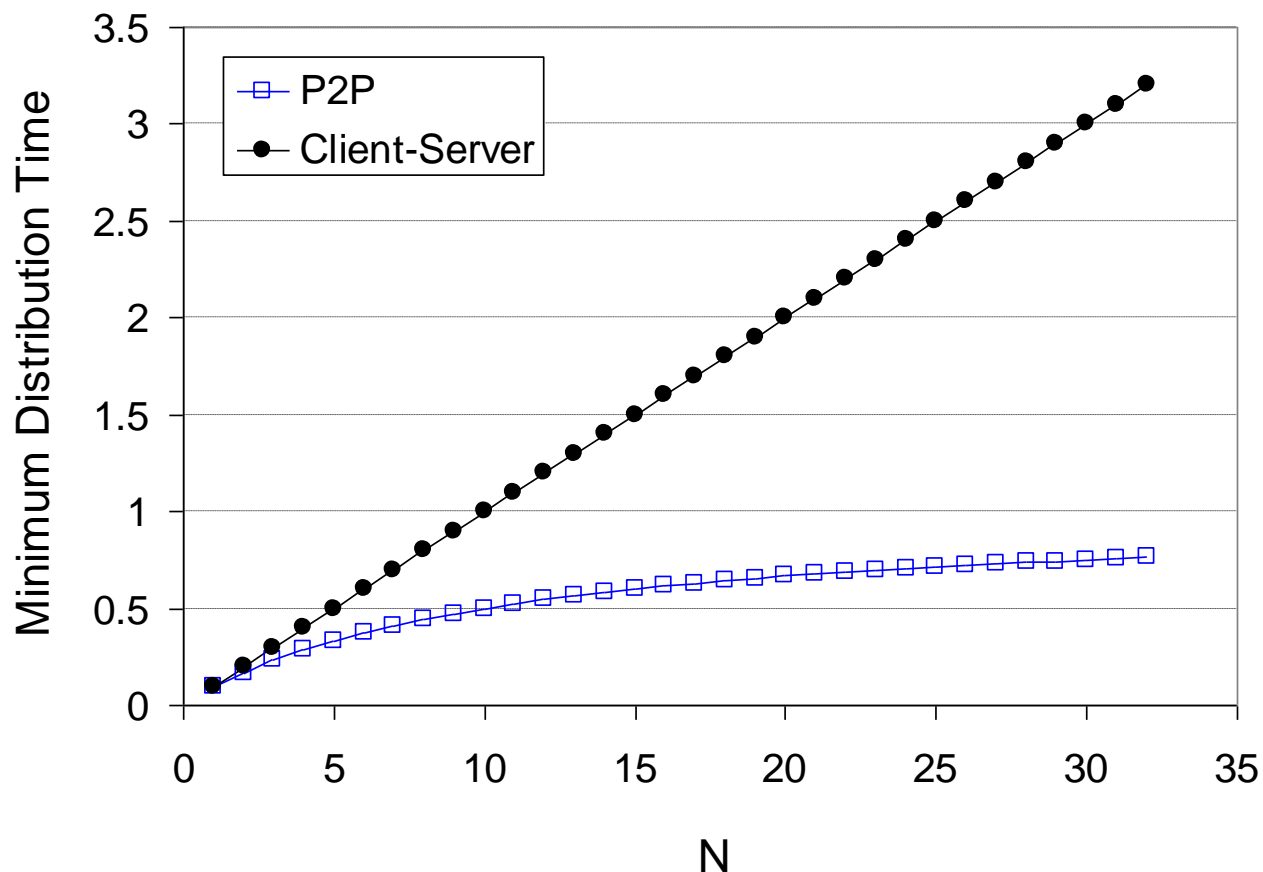
分配  $F$  的时间

到  $N$  个客户端, 使用  $D_{P2P} \geq \max\{F/u_s, F/d_{min}\}$  ?  
P2P 方法

随  $N$  线性增加...

...但这也是如此, 因为每个对等体都会带来服务容量

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

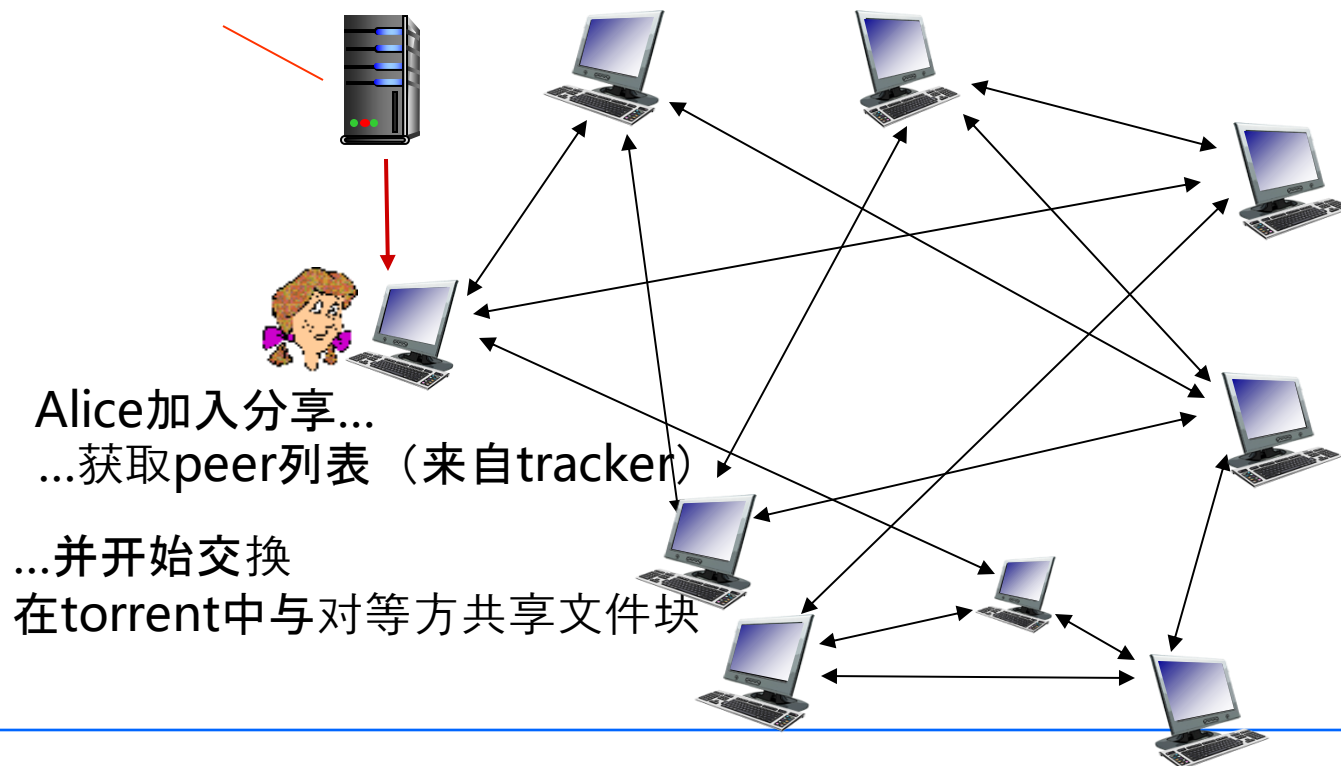




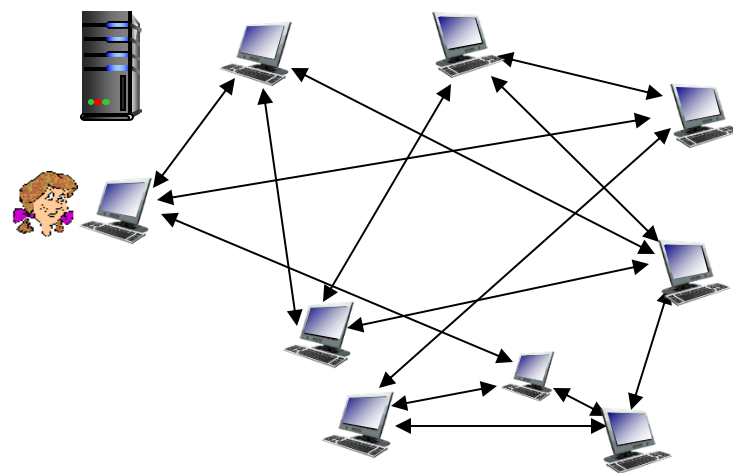
- ❖ 文件被分割为一个个大小为256Kb的文件块
- ❖ 对等方之间彼此发送/接收这些文件块

*tracker* : 追踪器, 记录所有参与的peer

*torrent* : 参与某一个文件分发的所有peer的集合



- ❖ 对等方加入torrent
  - 一开始没有文件块，但很快会从其他peers得到
  - 注册tracker，并得到peers的列表，然后与临近的peers连接
- ❖ 在下载的同时，也在向其它peer发送文件块
- ❖ 与其交换文件的peers可能会不断改变
- ❖ 注意：对等方可能加入或退出
- ❖ 当一个对等方获得全部文件后，可以离开或者继续为其它peers提供服务



## 请求文件块:

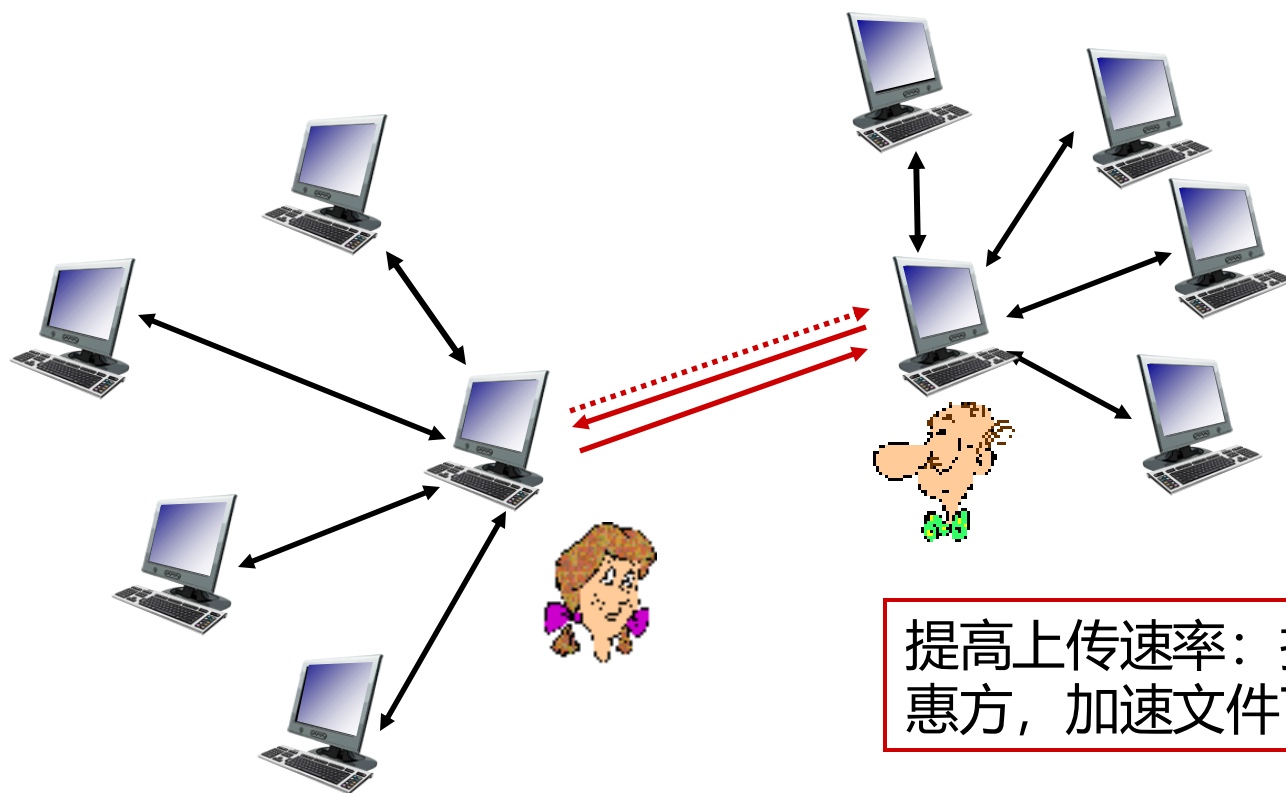
- ❖ 在任何时间，不同的peer拥有整个文件块的不同子集
- ❖ Alice会周期性的询问每个对等方其所拥有的文件块列表
- ❖ Alice向其他对等方请求自己没有的文件块，稀缺资源优先（所谓稀缺块就是邻居中拷贝数量最少的块）

## 发送文件块

- ❖ Alice将自己的文件块发送给此时给Alice发送文件块速率最高的前4个对等方
  - Alice不会发送文件块给其他对等方
  - 每10秒重新评估一次
- ❖ 每30秒：随机的选择其他对等方，并开始对其发送文件块
  - 发送文件块给该对等方，则Alice很可能入选对方的top4
  - 新选的对等方亦可能入选top4

tit-for-tat：一报还一报，以牙还牙，博弈论中一种激励策略

- (1) Alice乐观的选择Bob并给其发送文件块
- (2) Alice成为Bob的top4之一， Bob回报Alice
- (3) Bob成为Alice的top4之一



*我们对网络应用的学习现已完成！*

❖ 应用架构

- 主从式网络
- 对等网络

❖ 应用服务要求:

- 可靠性、带宽、延迟

❖ 互联网传输服务模式

- 面向连接, 可靠: TCP
- 不可靠, 数据报: UDP

❖ 具体协议:

- 超文本传送协议 HTTP
- 文件传送协议 FTP
- SMTP、POP、IMAP
- 域名服务器(Domain Name Server) DNS
- P2P: BitTorrent

*最重要的是:学会了协议!*

❖ 典型的请求/回复消息交换:

- 客户请求信息或服务
- 服务器以数据、状态代码  
响应

❖ 消息格式:

- 标题:给出数据信息的字段
- 数据:正在交流的信息

*重要主题:*

- ❖ 控制与数据消息
  - 带内、带外
- ❖ 集中式与分散式
- ❖ 无状态与有状态
- ❖ 可靠与不可靠的消息传输
- ❖ “网络边缘的复杂性”

### 作业

- ❖ R5、R12、R15、R19、R21
- ❖ P4, P5, P22