第2章 应用层

学习目标

- 网络应用协议的概念和实现原理
  - 传输层(transport-layer)服务模型
  - 客户-服务器(client-server)架构的概念
  - 对等(peer-to-peer)架构的概念
- 通过典型的流行应用层协议来学习协议
- 创建网络应用
  - socket API

HTTP
FTP
SMTP / POP3 / IMAP
DNS

本次课的主要内容

第2章 应用层

2.1 应用层协议原理

2.2 Web和HTTP

2.3 因特网中的电子邮件

- SMTP, POP3, IMAP

2.4 DNS：因特网的目录服务

2.5 P2P文件分发

2.6 视频流和内容分发网(自学)

2.7 套接字编程：生成网络应用

2.8 小结

网络应用

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- Search
- Data mining
- virtual reality

微软的虚拟全息可视头戴设备HoloLens

http://www.iqiyi.com/v_19rrntlh80.html

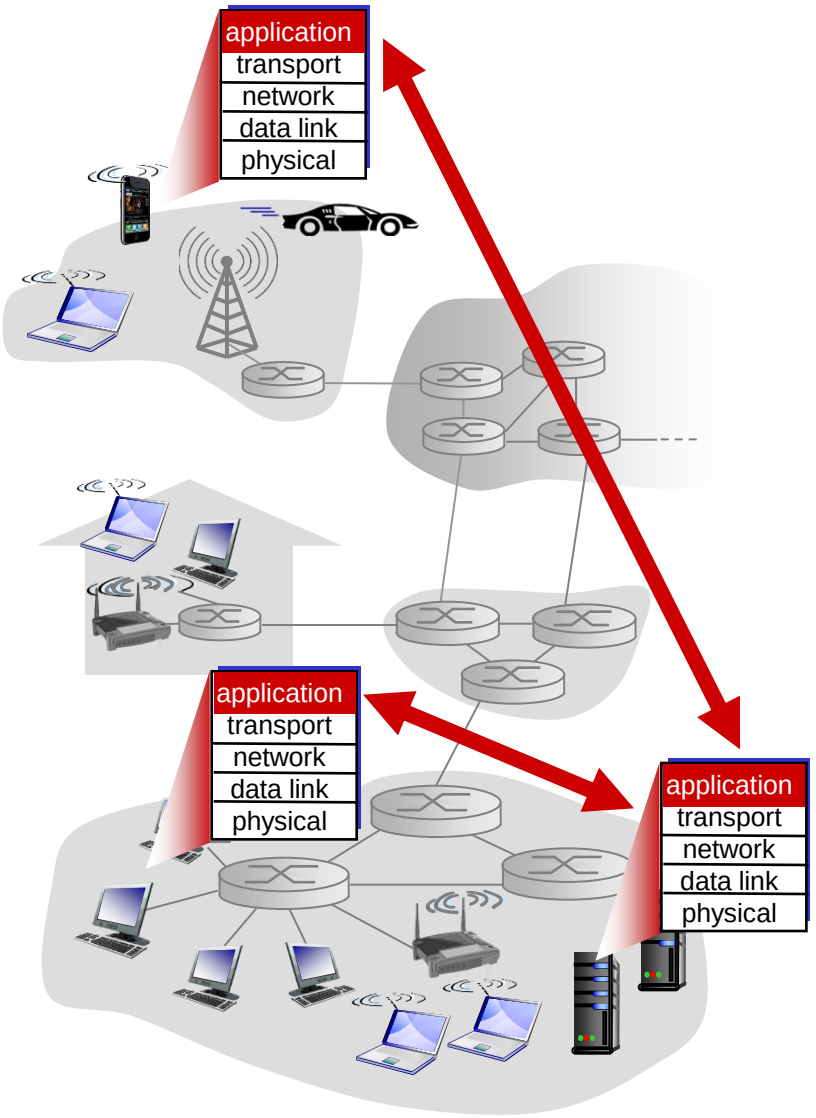
创建一个网络应用

应用程序需要完成

- 运行在(不同的)终端系统上
- 基于网络通信
- e.g., web server software communicates with browser software

网络应用程序不需要为网络核心设备写软件

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

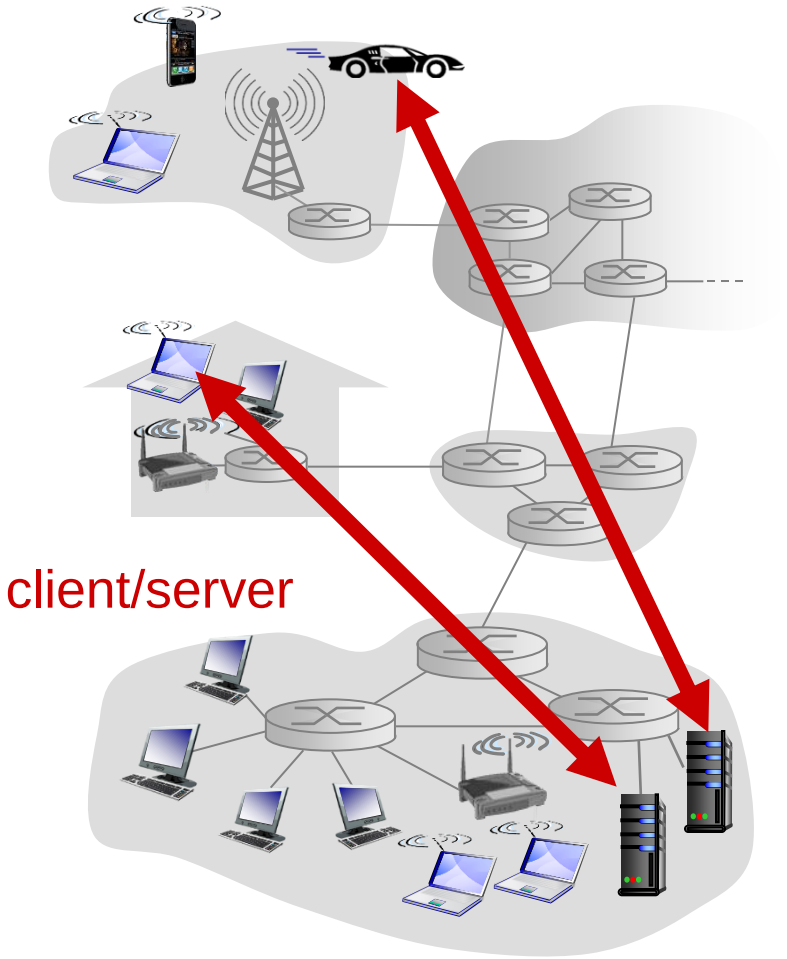


2.1.1 网络应用程序体系结构

- 两种体系结构
  - client-server(C/S, B/S)
  - peer-to-peer (P2P)

Client-Server体系结构

B/S, 浏览器
其他用户代理：
邮件客户端



服务器(server):

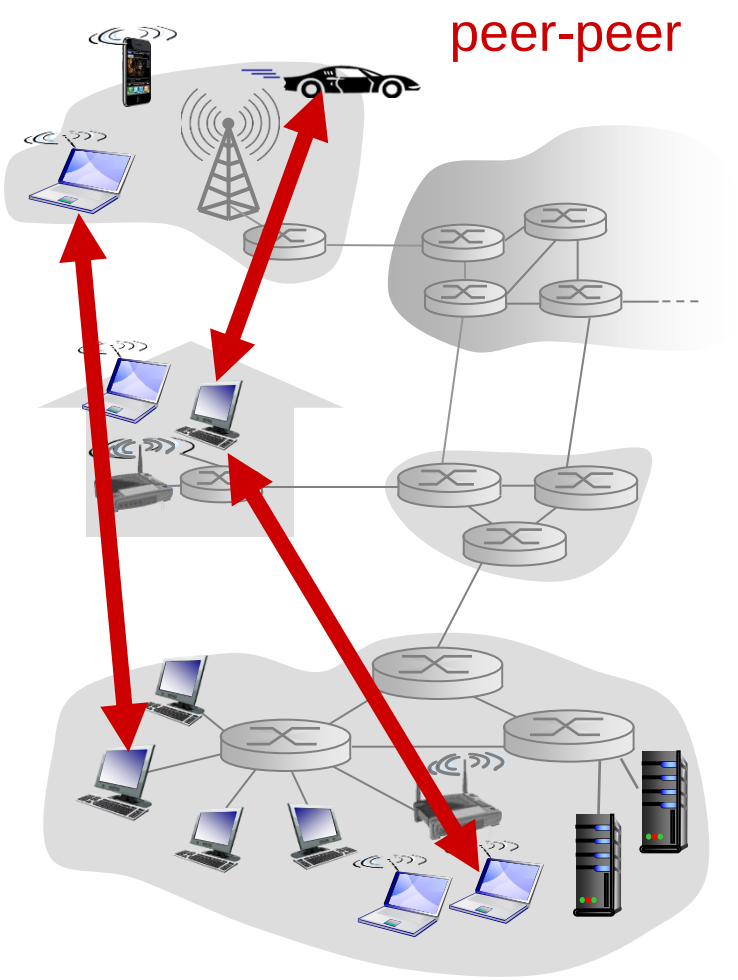
- 保持开机状态
- 固定的IP地址
- 可以采用具有扩展性的数据中心

客户端(client):

- 主动与服务器进行通信
- 可以间断性连接
- 可以采用动态IP地址
- 客户端进程之间相互不同通信

P2P体系结构

- 服务器不需要总是开机状态
- 任意的终端系统之间可以直接通信
- 对等节点从其他对等节点请求服务，同时可以为其他对等节点提供服务
  - 自我可扩展性--新的对等节点带来新的服务能力，同时也增加了新的服务需求
- 对等节点可以间歇性互连和改变IP地址
  - 管理非常复杂



2.1.2 进程通信

进程(process):程序运行在一个主机上

- 在同一个主机上，两个进程通信使用的是进程间通信的方式(由OS完成)
- 不同主机上的进程通信可以采取**消息交换**

clients, servers

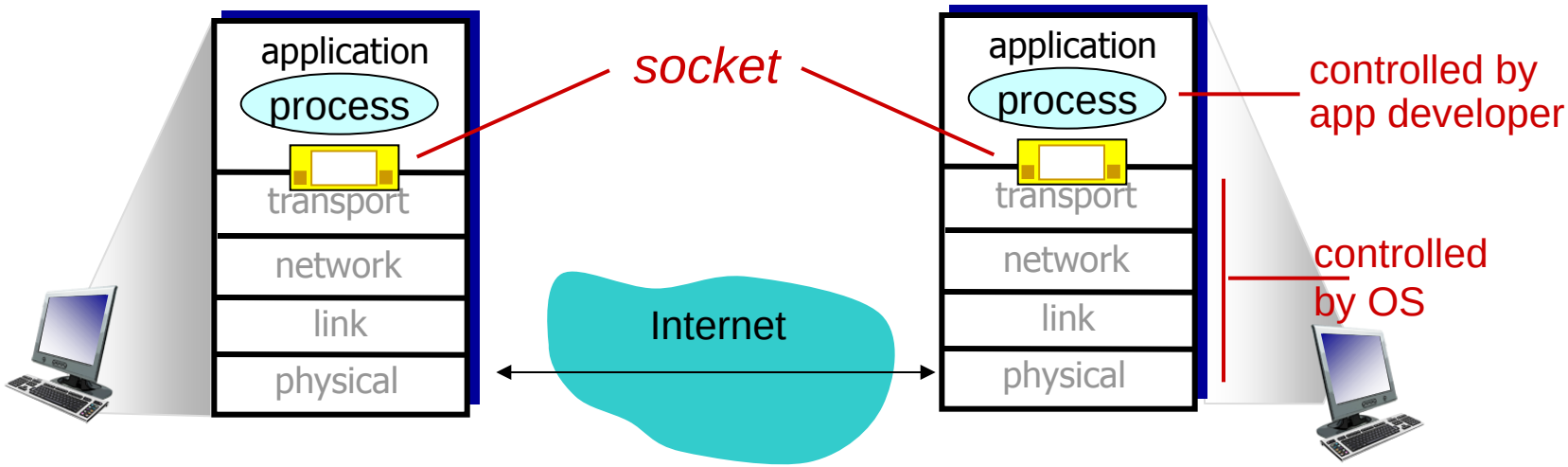
client process: 主动发起通信请求的进程

server process: 被动等待连接请求的进程

- 采用P2P架构的应用程序同时运行客户端进程和服务器端进程

套接字(socket)

- 进程发送/接收消息给/从它的套接字(socket)
- 套接字类比于门
  - 发送进程将消息扔出门
  - 发送进程依赖于门的另一侧的传输基础设施，实现将消息提交给接收进程来提交消息给接收进程的套接字



进程寻址

- 为了接收消息，进程必须有一个标识符(*identifier*)
- 主机设备由一个唯一的32比特的IP地址标识
- 问题：进程运行所在的主机IP地址是否可以用来标识进程?
  - 回答：不行，因为同一台主机上可以运行多个进程

- 标识符(identifier)包括与进行运行主机关联的IP地址和端口号(port numbers).
- 示例 port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80

创建一个新的网络应用程序时，必须分配一个新的端口号。不与公开的端口号或本机已使用的端口号重复。

2.1.3 可供应用程序使用的运输服务

一个应用程序需要运输层提供的传输服务有哪些?

data integrity(数据完整性)

- 某些应用要求100%的数据可靠传输 (e.g., file transfer, web transactions)
- 有的应用能够容忍数据丢失 (e.g., audio)

timing(时间保证)

- 某些应用要求低时延才能有效(e.g., Internet telephony, interactive games)

throughput(吞吐量保证)

- 某系应用要求有最小吞吐量保证才能有效(e.g., multimedia)
- 某些应用无论吞吐量是多少都可以有效(“elastic apps”)

Security(安全保证)

- 加密，数据完整性, …

传输层服务要求: 常见的app

application	data loss	throughput	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	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100’s msec yes and no

2.1.4 因特网提供的运输服务

TCP服务:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP服务:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

问题: 为什么会有UDP协议?

流行的因特网应用及其应用层协议及运输层协议

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	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

2.1.5 应用层协议

- 交换消息的类型
  - e.g., request, response
- 消息语法
  - 消息中字段的划分，以及有哪些字段
- 消息语义
  - 每个字段信息的含义
- 规则
  - 规定进程在什么时候以及如何发送&响应消息

公开协议:

- defined in RFCs
- 允许相互操作
- e.g., HTTP, SMTP

私有协议:

- e.g., Skype

2.2 Web和HTTP

- 带着问题来学习：
  - HTTP使用的传输层协议是TCP/UDP?
  - 简述HTTP协议请求数据的过程，例如访问：
    - www.uestc.edu.cn
  - 怎么判断用户请求使用的是持续连接和非持续连接?
  - 持续连接和非持续连接有什么区别?
  - 怎么理解无状态
  - Web新增的三个功能：cookie、web缓存、条件Get各有什么好处?

基本术语

首先，介绍web应用的基本术语

- web page (网页) 可以包括多个对象(objects)
- 对象可以是 HTML file, JPEG image, Java applet, audio file,…
- 网页包括一个基本的HTML文件，它可以包括多个引用对象
- 每个对象通过一个URL来寻址，e.g.,

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

host name　　path name

http://www.w3school.com.cn/media/media_mimeref.asp

2.2.1 HTTP概况

HTTP: 超文本传输协议

- Web应用的应用层协议
- C/S架构
  - ***client***: 浏览器，用于请求、接收和展示web对象(使用HTTP协议)
  - ***server***: web服务器发送(使用HTTP协议)对象作为对请求的响应

2.2.1 HTTP概况(续)

怎么理解无状态

uses TCP:

- 客户端进程首先向服务器进程发起TCP连接请求(通过创建socket)，port 80
- 服务器进程接受来自客户端进程的TCP连接请求
- HTTP消息(应用层协议消息)在浏览器(HTTP客户端)和web服务器(HTTP服务器)之间交换
- TCP连接关闭

HTTP 是无状态的

- 服务器不维护历史的客户请求信息

aside

维护状态的协议是复杂的!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

2.2.2 HTTP 连接

非持续HTTP连接 (non-persistent HTTP)

- 通过TCP连接最多发送一个对象，然后关闭TCP连接
- 下载多个对象需要建立多个TCP连接

持续HTTP连接 (persistent HTTP)

- 在客户端和服务器之间多个对象可以基于一个TCP连接发送

默认方式下使用持续连接

非持续HTTP

suppose user enters URL:

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

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

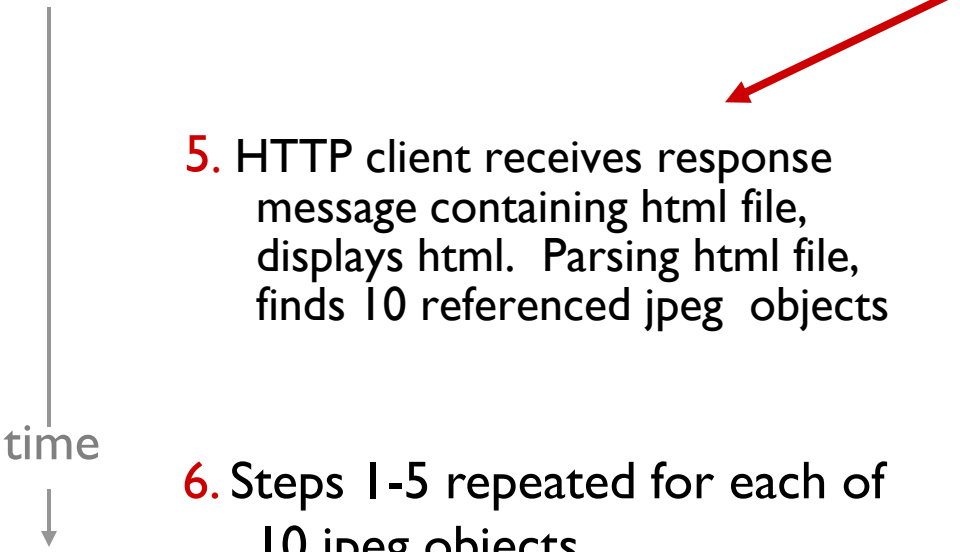
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

非持续HTTP



4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

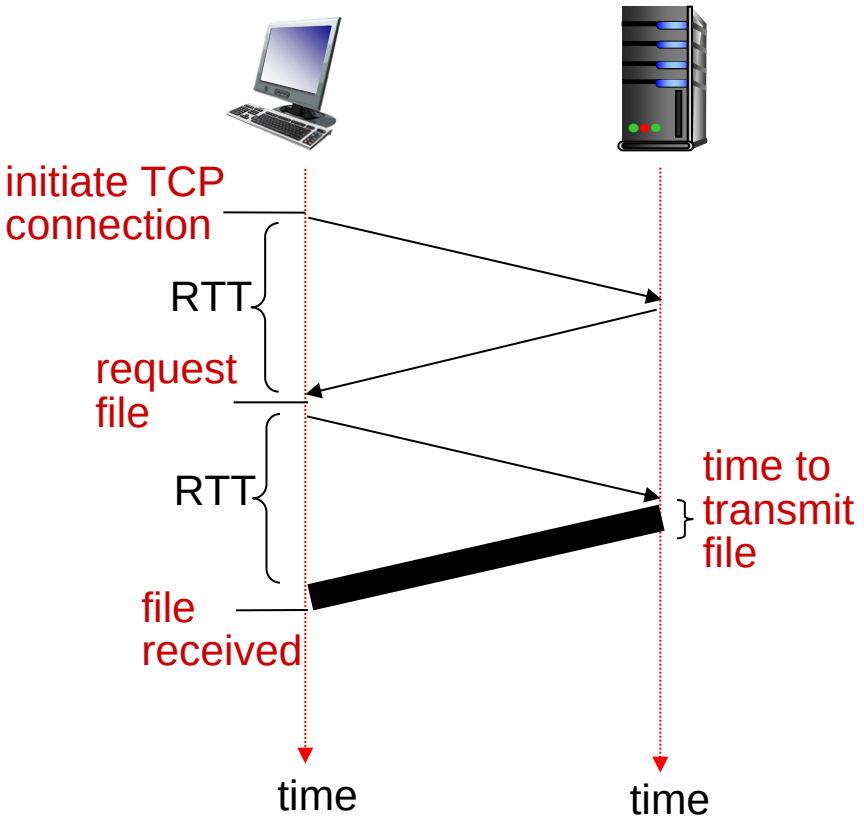
非持续HTTP的响应时间

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- 1个RTT用于建立TCP连接
- 1个RTT用于HTTP请求
- one RTT for HTTP请求和需要HTTP返回响应的前几个字节
- 文件传输时间
- 非持续HTTP

响应时间 = 2RTT+ file transmission time



采用持续连接的HTTP

非持续HTTP的问题:

- 每个对象要求2个RTTs
- 每个TCP连接都会存在OS负载
- 浏览器通常开放并行的TCP连接获取引用对象

持续HTTP:

- 服务器在发送响应后会继续保持TCP连接可用
- 在相同的客户端/服务器之间的后续的HTTP消息可以继续使用这个TCP连接交换
- 客户端在遇到一个对象引用时就发送请求

采用持续连接的HTTP

- 非流水线方式：客户机只能在前一个响应接收到之后才能发出新的请求。
  - 客户机为每一个引用对象的请求和接收都使用一个RTT时延。
  - 会浪费一些服务器资源：服务器在发送完一个对象，等待下一个请求时，会出现空闲状态。
- 流水线方式：客户机可一个接一个连续产生请求（只要有引用就产生），即在前一个请求接收到响应之前可以产生新的请求。服务器一个接一个连续发送响应对象。
  - 节省RTT时延，可能所有引用对象只花费一个。
  - TCP连接空闲时间很短。
- 默认方式：流水线方式的持久连接。

2.2.3 HTTP报文格式

方法（命令）——
- ✓ GET：请求一个对象。
- ✓ POST：提交表单（添加信息）。
- ✓ HEAD：请求返回对象响应报文首部

response

路径名

回车符
换行符

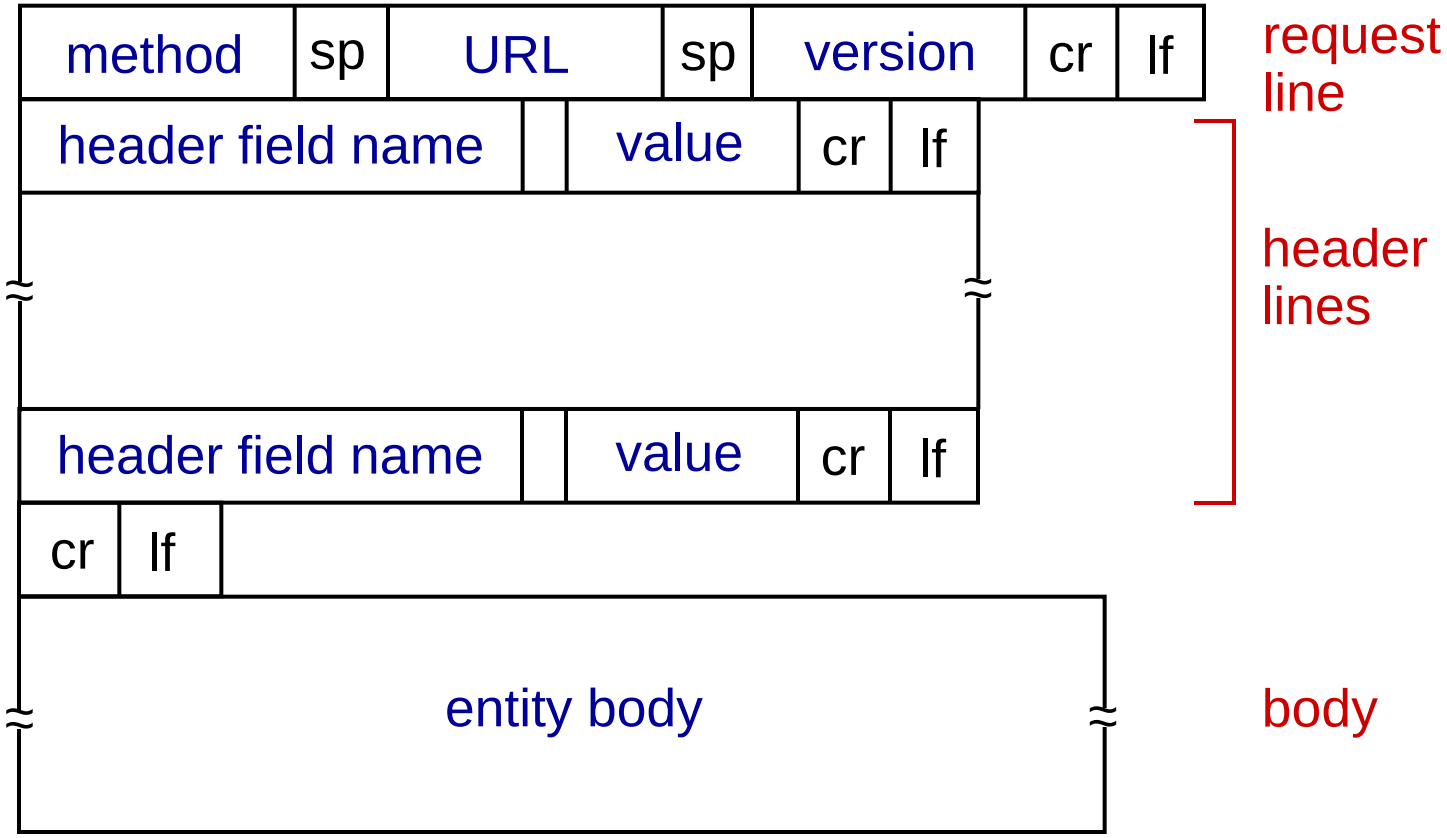
request li
(GET, POST,
HEAD commands)

header
lines

carriage return,
line feed at start
of line indicates
end of header lines

```
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
```

HTTP请求消息的通用格式



上传表单输入

POST 方法:

- Web页面通常包含表单上传的功能
- 输入通过(entity body)上传到服务器

URL 方法:

- 使用GET方法
- 输入通过请求命令行的URL字段上传:

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

方法类型

HTTP/1.0:

- GET
- POST
- HEAD
  - 要求服务器保持请求的对象不响应

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - 将entity body中的文件上传到URL字段指定的位置
- DELETE
  - 删除URL字段中指定的文件

2.2.3 HTTP响应消息

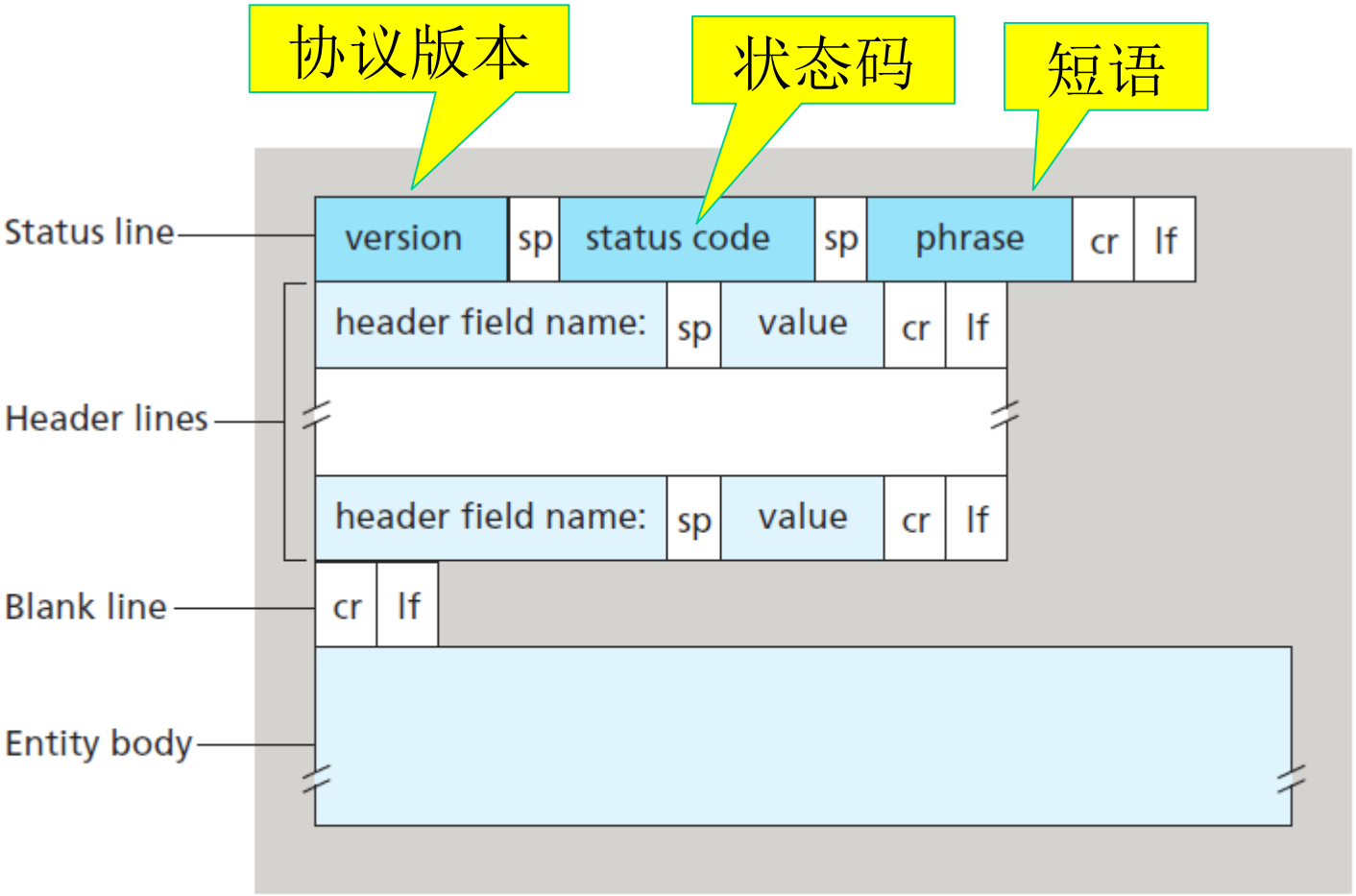
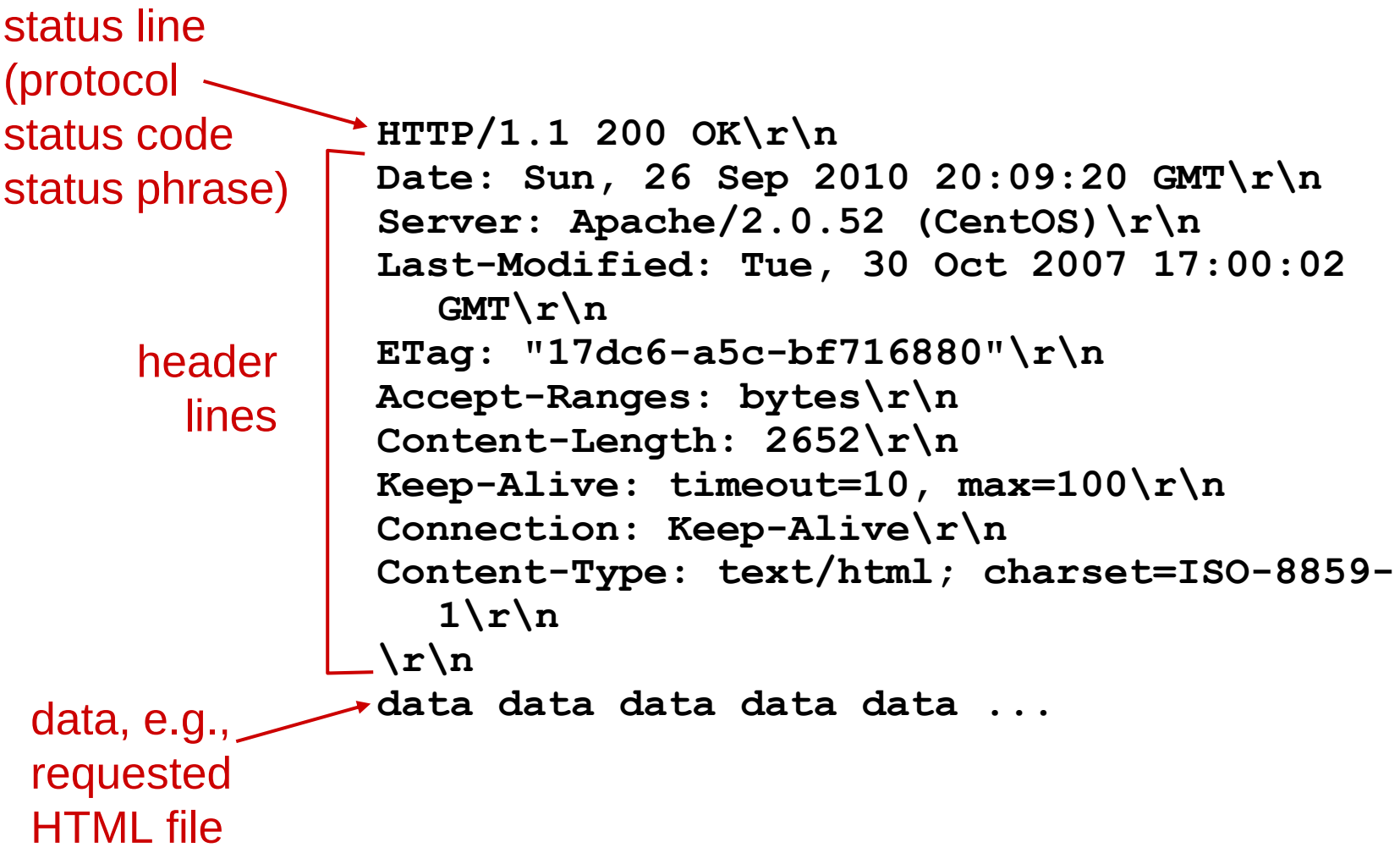


Figure 2.9 ♦ General format of an HTTP response message

2.2.3 HTTP响应消息



2.2.3 HTTP响应消息

状态码出现在服务器到客户端响应消息的第1行中.

典型的状态码如下:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

- requested object moved, new location specified later in this msg (Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

2.2.4 用户与服务器的交互：cookies

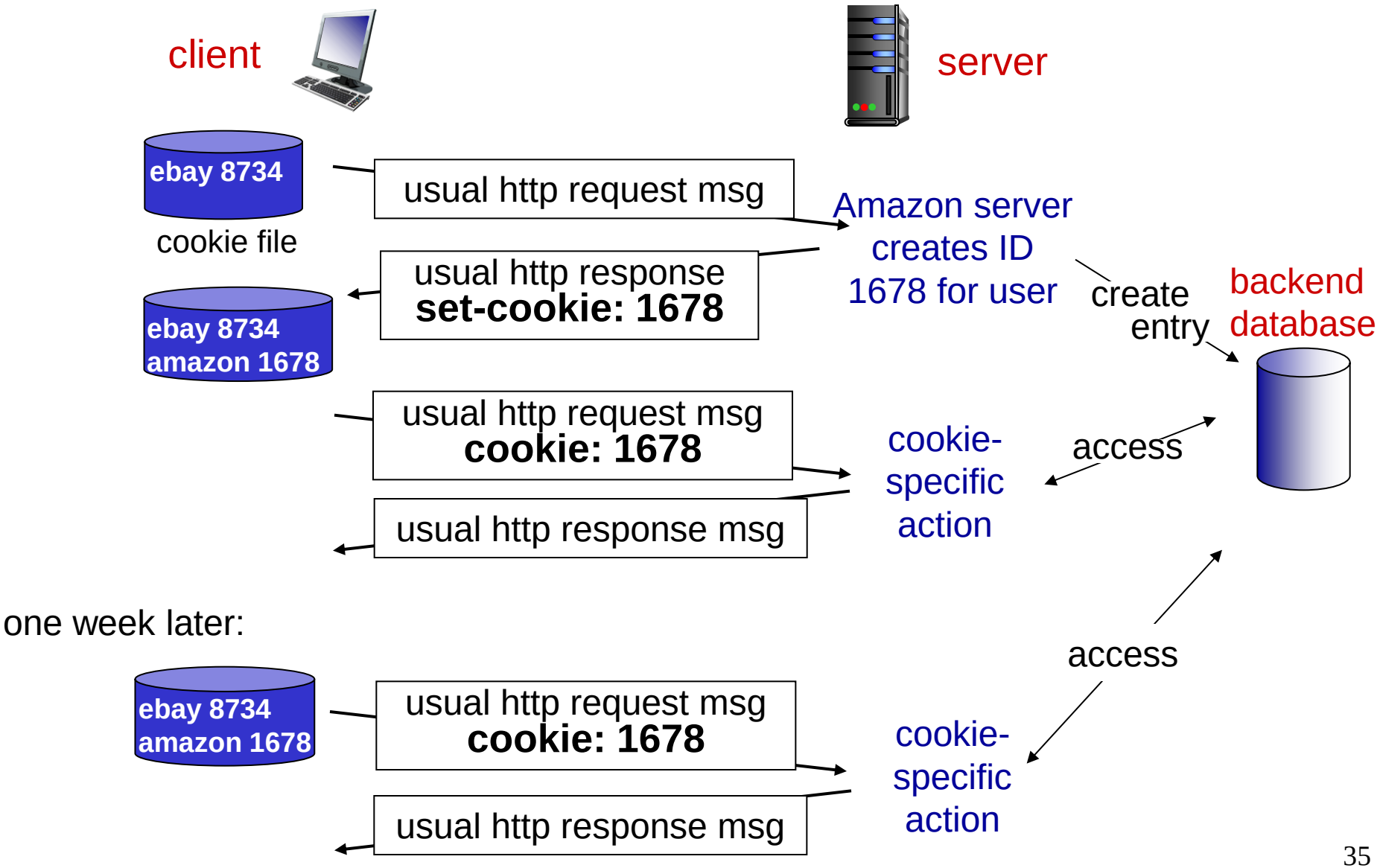
许多网站使用cookies技术四个组件：

1) HTTP响应消息中的一个cookie首部行
2) HTTP请求消息中的一个cookie首部行
3) 保存在用户主机上的cookie文件，由用户的浏览器管理
4) Web站点的后台数据库

例子:

- Susan always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

Cookie：跟踪状态


client
server
ebay 8734
cookie file
usual http request msg
Amazon server creates ID 1678 for user
create entry
backend database
usual http response
set-cookie: 1678
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
access
usual http response msg
one week later:
access
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
usual http response msg

Cookies

cookies 可以用于以下场景:

- authorization(认证)
- shopping carts(购物车)
- recommendations(推荐)
- user session state (Web e-mail)(用户会话状态)

aside

cookies和隐私:

- Cookies允许站点对个人信息进行学习
- 个人通常需要提供性能和email等信息给网站

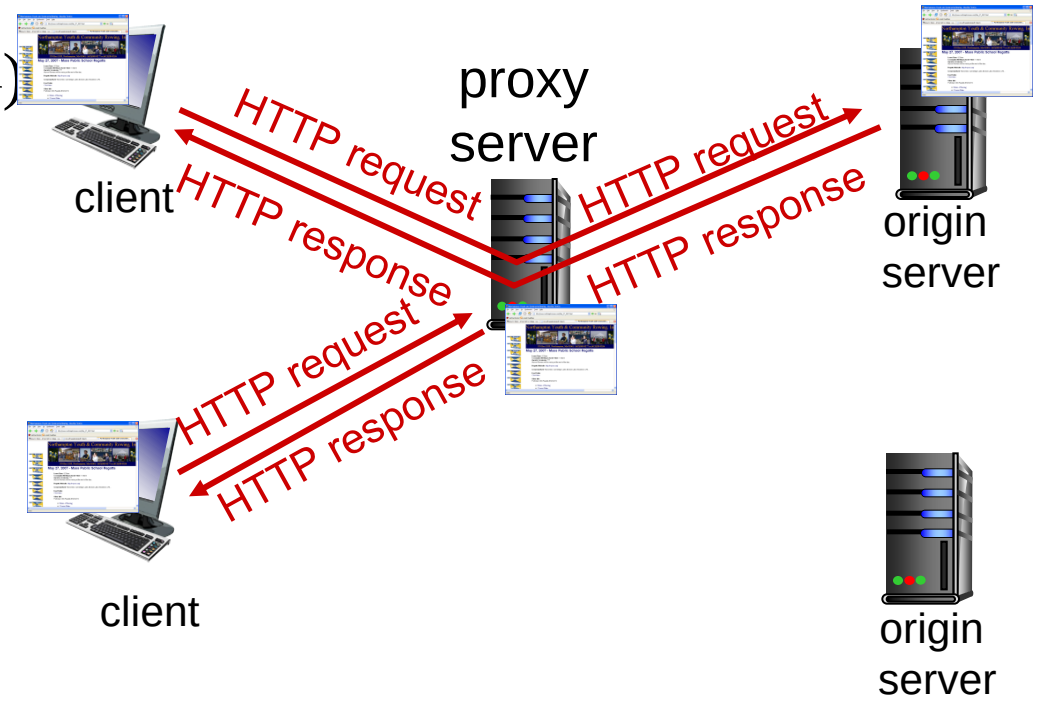
如何保持状态:

- 协议端点：在多个事务上保持发送方/接收方的状态
- cookies: HTTP消息携带状态

2.2.5 web 缓存(代理服务器)

目标: 不需要原始服务器就可以满足用户需求

- 用户设置浏览器：通过cache接入Web
- 浏览器发送所有的HTTP请求给cache(缓存服务器)
  - 如果请求对象存在于缓存服务器：缓存服务器直接返回对象给客户
  - 如果请求对象不在缓存服务器中，则缓存服务器向原始服务器请求，然后返回对象给客户



2.2.5 web 缓存

- 缓存服务器扮演了客户端和服务器
  - 对于客户端的原始请求来说，它是服务器
  - 对于原始的服务器来说，它是客户端
- 典型的cache是由ISP安装(university, company, 住宅的ISP)

可以使得接入带宽不那么富裕的内容提供商能够有效的提供内容给用户

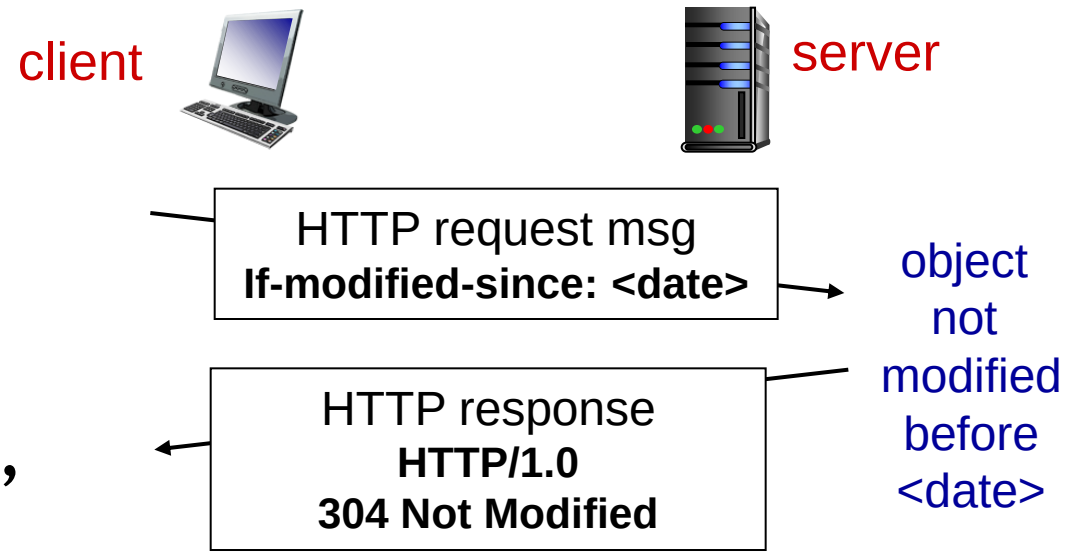
为什么使用cache?

- 减少对于客户端请求的响应时间
- 减少原始服务器接入链路的流量
- Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)

2.2.6 条件GET

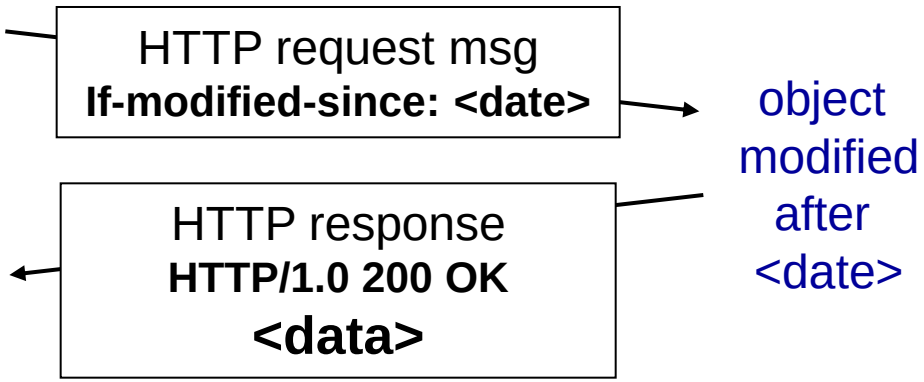
问题：缓存服务器虽然能够减少响应时间，但是怎么确保缓存服务器上的object是最新的?

解决方法：使用条件GET，允许缓存服务器证实它的对象是最新的。

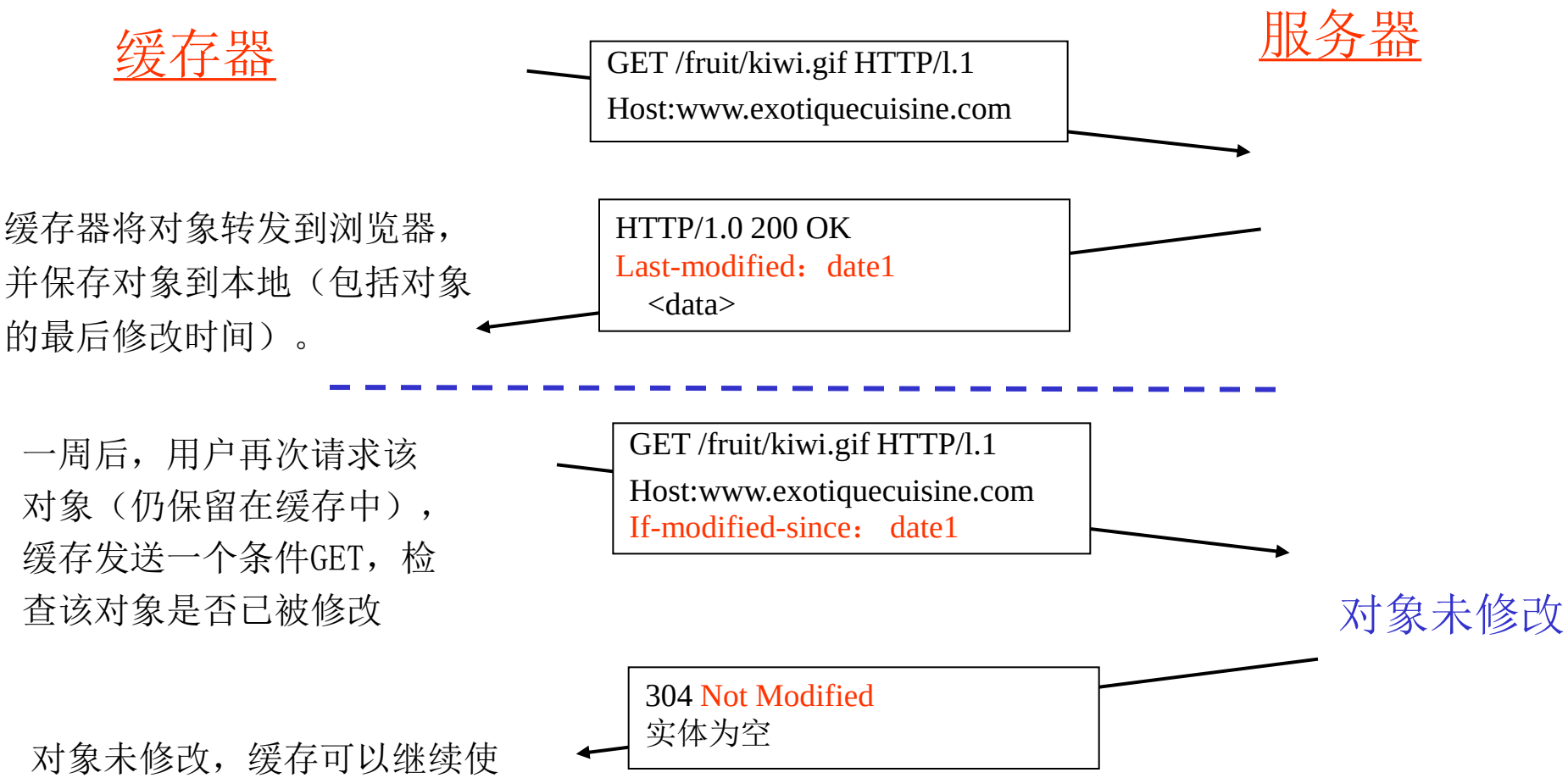


具体方法:

- 发送包含上次更新的时间
- 如果没有发生修改，则返回一个空数据的响应报文
- 如果发生了修改，则返回一个最新的数据



2.2.6 条件GET


缓存器
服务器
GET /fruit/kiwi.gif HTTP/l.1
Host:www.exotiquecuisine.com
缓存器将对象转发到浏览器，并保存对象到本地（包括对象的最后修改时间）。
HTTP/1.0 200 OK
Last-modified： date1
<data>
一周后，用户再次请求该对象（仍保留在缓存中），缓存发送一个条件GET，检查该对象是否已被修改
GET /fruit/kiwi.gif HTTP/l.1
Host:www.exotiquecuisine.com
If-modified-since： date1
对象未修改
304 Not Modified
实体为空
对象未修改，缓存可以继续使用该对象的拷贝，并转发给用户浏览器。

课堂练习

www上每一个网页都有一个独立的地址，这些地址统称为（ ）

A、IP地址　　B、域名地址

C、统一资源定位符　　D、www地址

Cookie主要包括哪几个部分？Cookie的作用是什么？会带来什么问题？（6分）

P114-P1

P114-P4，P5（下次课评讲）

有兴趣的同学可以尝试利用Wireshark抓取包含cookie信息的数据包来进行分析

描述引入Web缓存后浏览器访问网页的过程?

课堂练习

- 现有一个网页由8个对象文件组成，8个对象文件在一个web服务器上，使用带流水线的持久HTTP连接显示这个网页需要等待多少个RTT（ ）

A. 3个RTT B. 4个RTT C. 9个RTT D. 16个RTT

- 下列HTTP报文首部行正确的是（ ）

A. GET HTTP/1.1 /somedir/index.htm B. HTTP/1.1 OK

C. POST /somedir/index.htm HTTP/1.1 D.GET 200 OK

- 两个不同的Web页面（例如，www.uestc.edu.cn/index.html和mail.uestc.edu.cn/index.html）可以使用一个持久连接发送。( )
- HTTP响应报文不会有空的报文体。（ ）