计算机网络

第2章 应用层

目　　录

2.1 应用层协议原理

- **常见的网络应用**
  - 上网浏览新闻——IE、Maxthon、FireFox……
  - 处理电子邮件——Outlook Express、FoxMail、Outlook……
  - 和熟悉的或者陌生的朋友聊天——ICQ、QQ、MSN Messenger、UC……
  - 网络电话——SkyPe、QQ、Net2Phone……
  - 网络游戏对战——CS、魔兽世界、联众……
  - 资源共享——FTP、BT、电骡……
  - 在线视频——VOD、ppLive……
  - 搜索引擎——Google、百度、MSN Search……

2.1 应用层协议原理

看了这么多成功的应用，可能你跃跃欲试，很想编写一种类似于Google这么的超级网络应用,期待自己有一天也能一步登天，迈入世界级的IT风云人物之列，甚至试图问鼎一下世界首富……

那么目前的你应该做些什么呢？

2.1 应用层协议原理

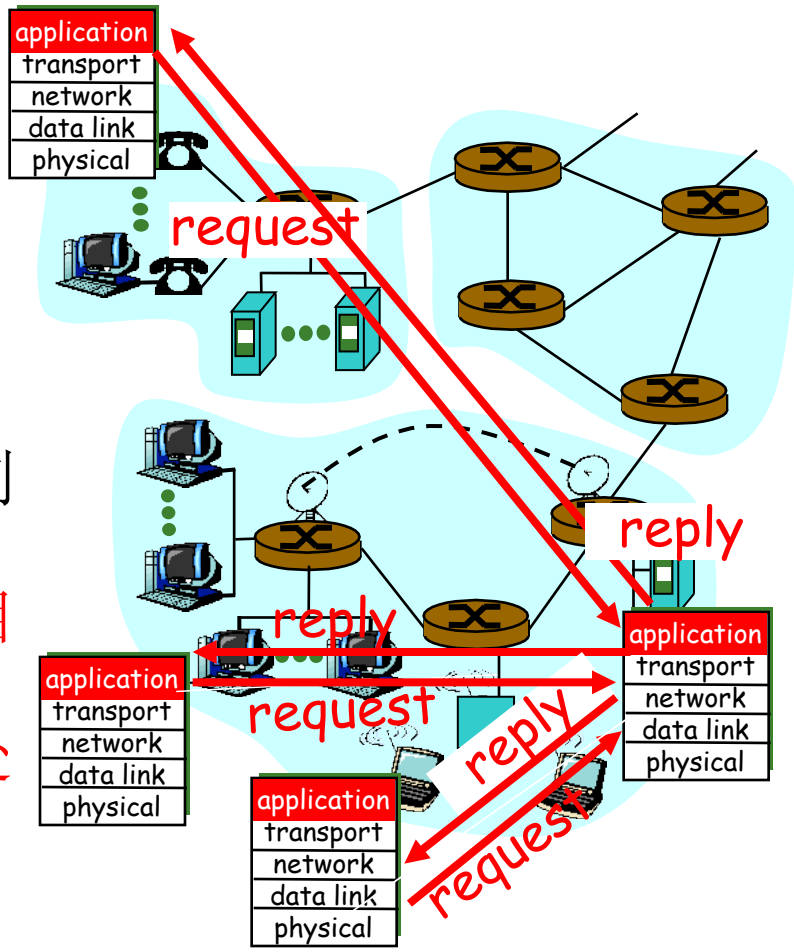
- 知道什么是网络应用程序
- 可以向网络发送数据
- 可以从网络接受数据
- 可以对数据进行处理
- 可能还能够
- 将数据呈现在界面上，以非常友好的方式让你知道它在做什么，省得你说它怠工
- 时不时的弹出一个小窗口，提示你不要太辛勤工作了，以表示对你无微不至的关怀
- ……

2.1 应用层协议原理

- **决定你的杀手级网络应用所采用的体系构造**
  - 客户机/服务器体系构造（C/S）
  - P2P体系构造
  - 混合体系构造

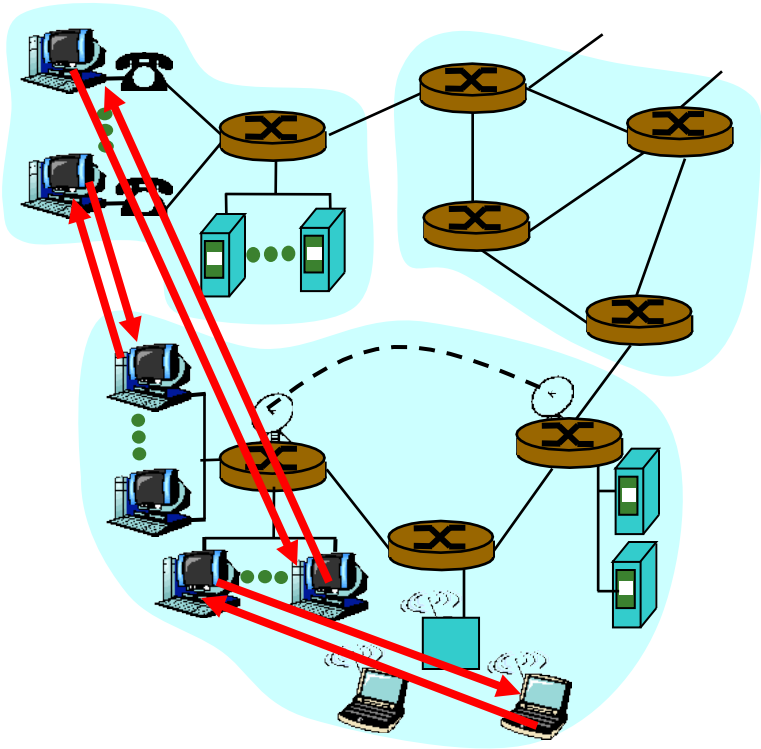
2.1 应用层协议原理

- **客户机/服务器体系构造的特征是怎样的?**
  - 存在一种能够向客户机提供服务的服务器，e.g.，WEB服务器
  - 存在一种或者多种主动连接服务器，试图从服务器那里获取所需服务的客户机，e.g.，IE浏览器
  - 尤其注意1：客户机之间不能相互通信
  - 尤其注意2：为提升服务器的处理能力，一般采用服务器群集（Server Farm）



2.1 应用层协议原理

- **那P2P体系构造又以什么方式来体现呢？**
  - 任何一方既提供服务又享有服务
  - 结点之间能够直接通信
  - 结点的地址以及他们之间的连接可能随时发生变化
  - 例如: Gnutella
  - 尤其注意：P2P体系构造非常轻易扩展，但也尤其难以管理



2.1 应用层协议原理

- 混合体系构造
  - 那混合体系构造自然而然就是C/S体系构造和P2P体系构造的混合体喽！
  - 请大家回忆一下第一种P2P应用Napster和及时通信（IM），一切就都明白了！

2.1 应用层协议原理

- **网络应用涉及各个构成部分的交互**
  - 同一台主机上的进程之间通信的规则，由操作系统制定，和计算机网络无关，本课程就不讨论了。需要了解的，请回头看看《操作系统原理》及有关书籍
  - 不同主机上的进程之间通信的规则，当然就和网络有关了，这套规则在计算机网络中，称之为“**应用层协议**”，也是本章要点讨论的内容

2.1 应用层协议原理

- **当你的网络应用程序Run起来后，就变成了网络应用进程。产生了如下问题：**
  - 当你的网络应用和其他人开发的网络应用共同运营在一台主机上时，怎样把你的网络应用区别开来？
  - 通信子网只负责把数据交付到主机，并不负责把数据交付到应用，主机怎样懂得数据该交付到哪个网络应用？

2.1 应用层协议原理

- **一种例子**
  - 你有一种信箱（非电子的阿），而且每天都会查看一次信箱，取走新的信件和报纸，当你有信件需要寄送时，直接投递到邮局的邮筒里
  - 目前对你的信箱进行一种小小的技术改造
    - 信箱提成两部分：发送部分和收取部分，发送部分存储需要寄送的信件，邮局送来的邮件存储在收取部分
    - 信箱添加电子感应和告知装置
      - 一旦发送部分有了新的信件，则告知邮局
      - 一旦收取部分有了新的信件，则告知你

下续

2.1 应用层协议原理

- 一种例子
  - 调整一下规则
    - 当你有外发信件时，放置到信箱的发送部分，信箱会自动告知邮局，然后邮局工作人员上门取走信件
    - 当你有该收取的信件时，邮局工作人员放置在你的信箱的收取部分，信箱也会自动告知你，有新的信件
  - 邮局工作人员怎样才干懂得哪个信箱是你的呢？
    - 小区名称

      这么足够了吗？

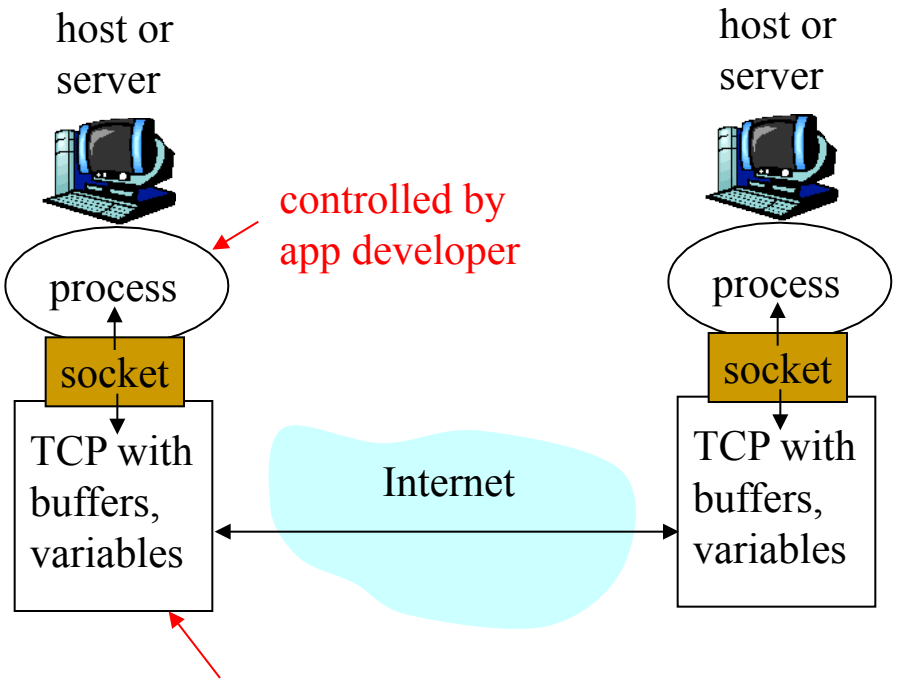
    - 门牌号码

2.1 应用层协议原理

- **类比到因特网，提供了类似的处理措施，那就是“套接字（Socket）”**
  - 每个网络应用进程都有一种属于自己的套接字，该套接字在整个因特网上独一无二
    - 主机地址：标识该网络应用进程运营在因特网上哪一台主机上，一般使用32位的IP地址进行标识
    - 端口地址：在该主机上标示该网络应用进程，一般使用16位的端标语进行标识
      - e.g.，WEB Server：80；Mail Server：25；
    - 所以套接字的长度为48位

2.1 应用层协议原理

- **进程经过套接字来接受和发送报文**
- **套接字相当于一种通道**
  - 发送进程将报文交给套接字
  - 套接字将这些报文传播到接受进程的套接字



2.1 应用层协议原理

- 因特网会给网络应用提供诸多不同类型的服务，你的网络应用需要哪些服务呢？
  - 数据的可靠传播：你的网络应用是否需要？
  - 带宽的自动控制：你的网络应用是否带宽敏感？
  - 传播和反馈的实时性

2.1 应用层协议原理

- 常见应用程序对传播服务的要求

应用程序	数据丢失	带宽	实时性
文件传播	不丢失	弹性	无
e-mail	不丢失	弹性	无
Web 网页	不丢失	弹性	无
实时音频/视频	允许丢失 允许丢失	音频: 5Kb-1Mb 视频:10Kb-5Mb	100’s msec
存储音频/视频	允许丢失	同上	few secs
交互式游戏	允许丢失	几 Kb/s 以上	100’s msec
金融应用	不丢失	弹性	yes and no

2.1 应用层协议原理

- **因特网运送层将所提供的服务整合成两类传播服务，你的网络应用使用哪一类传播服务，你该做出决定了！**
  - TCP
    - *面对连接:* 在客户端和服务器进程之间需要建立连接
    - *可靠传播：*在发送和接受进程之间
    - *流量控制:* 发送数据的速度决不超出接受的速度
    - *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

2.1 应用层协议原理

- UDP
  - 在客户端和服务器进程之间实现"不可靠的"数据传播
  - *不提供*:连接建立, 可靠性确保,流量控制,拥塞控制,实时性, 最小带宽承诺

2.1 应用层协议原理

- **因特网常见应用采用的传播协议**

应用	应用协议	所依赖的传播协议
e-mail	smtp [RFC 821]	TCP
远程终端访问	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
文件传播	ftp [RFC 959]	TCP
流媒体	专有协议 (e.g. RealNetworks)	TCP or UDP
远程文件服务器	NFS	TCP or UDP
IP电话	专有协议 (e.g., Vocaltec)	typically UDP

2.1 应用层协议原理

- 祝贺你！至此你已经取得了构造属于你自己的杀手级网络应用所需要的最基本最基本的知识！
- 但是这还远远不够，你还需要继续学习
  - 协议究竟怎样工作
  - 套接字怎样工作
  - 传播层的服务是怎样提供的
  - IP地址是怎么回事
  - 网卡和网线起了什么样的作用
  - 怎样确保网络应用的安全性和性能

  ……

2.2 WEB应用和HTTP协议

- 历史的回忆
- 19世纪70年代，电话的发明，扩展了人类通信的范围，增强了人类通信的实效性
- 20世纪23年代，广播收音机和电视的发明，极大的丰富了人类可获取信息
- 20世纪90年代，WEB的发明，极大的提升了人类主动获取信息的能力
- 广播收音机/电视和WEB的异同点
- 都是广播和按需操作
- 你不能公布电视节目，但能够公布WEB内

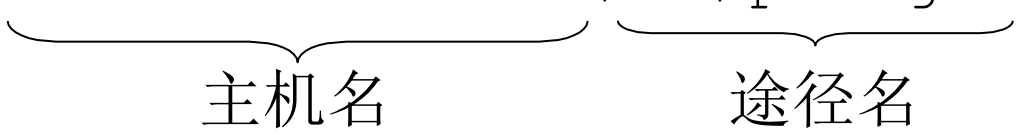
2.2 WEB应用和HTTP协议

- **WEB的构成**
  - WEB服务器：IIS、Apache、TomCat……
  - 浏览器：IE、Maxthon、Firefox
  - 协议
    - 信息体现的协议——HTML
    - 信息传播的协议——HTTP

尤其阐明：WEB属于C/S模式

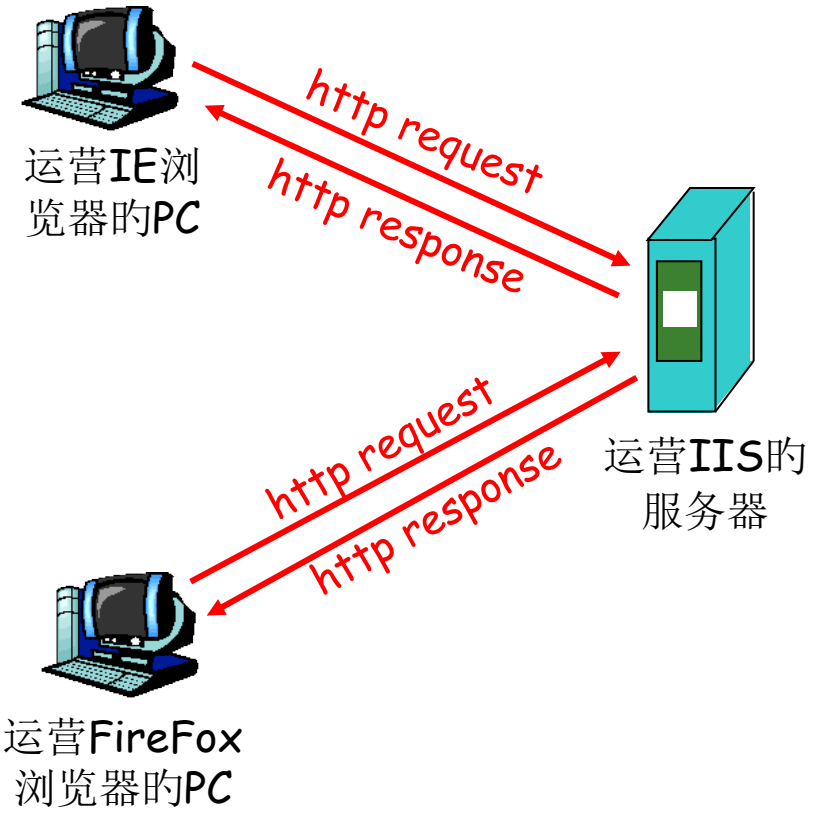
2.2 WEB应用和HTTP协议

- **WEB内容的体现**
  - Web 页面由某些对象构成。
  - 对象能够是HTML文件，JPEG图片，音频文件，Java Applet……
  - HTML文件是Web页面的基础，它能够涉及多种各样的对象，是一种容器对象
  - 任何一种对象都能够用 URL来定位
  - URL的例子：



2.2 WEB应用和HTTP协议

- **WEB内容的传播——HTTP协议**
  - 客户端/服务器模式
    - *客户端:* 浏览器祈求、接受、展示 Web对象（objects）
    - *服务器:* Web 服务器发送对象对祈求进行响应
  - http1.0: RFC 1945
  - http1.1: RFC 2068



2.2 WEB应用和HTTP协议

http: TCP 传播服务:

- 客户端开启TCP连接(创建套接字) 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)之间进行互换
- 关闭TCP 连接

http 是 "无状态（stateless）"的

- 服务器不保存任何访问过的祈求信息

小评论

保存状态的协议很复杂哟!

- 过去的历史 (状态) 需要保存
- 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，还需重新核对

2.2 WEB应用和HTTP协议

HTTP1.0的传播模式——非持久性连接

假设顾客键入了一种
(该网页包括文本并引用了10 jpeg 图片)

1a. http 客户端开启 TCP 连接到上的http 服务器 (进程). Port 80 是 http 服务器的默认端口.

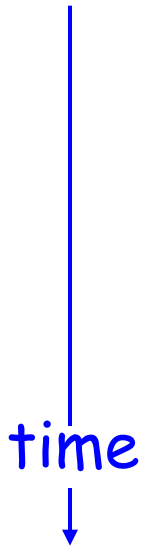
1b. 在www.hust.edu.cn 上的 http 服务器在 port 80 等待 TCP 的连接祈求. "接受" 连接并告知客户端

2. http客户端发送 http *祈求报文* (涉及URL) 进入 TCP 连接插口（socket）

3. http 服务器接受到祈求报文, 形成 *响应报文* (包括了所祈求的对象， cs/home.index), 将报文送入插口（ socket）

time

2.2 WEB应用和HTTP协议



4. http 服务器关闭 TCP 连接.

5. http 客户端接受到了包括html文件的响应报文。分析 html 文件, 发觉 10 个引用的 jpeg 对象

6. 对10 jpeg objects 逐一反复1-5 步

2.2 WEB应用和HTTP协议

- **非持久性连接工作机制分析**
  - 取对象需要2 RTTs
    - TCP 连接
    - 对象祈求/传送
  - 许多浏览器同步打开多种并行的连接来改善性能

请考虑：假如有1万台客户机访问WEB服务器的某个页面，该页面有100个对象，那就意味着需要100万个连接，1个服务器能够扛得住么？

2.2 WEB应用和HTTP协议

- **HTTP1.1引入的新传播模式——持久连接**
  - 服务器在发送响应后，不再断开TCP连接，而是保持该连接，用于后续对象的传送，直至该连接“休息”了一种较长的时间后，方断开该连接
  - 降低了对服务器端连接数的需要，从而降低了对服务器端套接字资源的占用，提升了服务器的负载能力
  - 持久连接又能够分为
    - 非流水线方式：一种对象传播完毕方能传播下一种
    - 流水线方式：能够一次性发送全部祈求，慢慢接受

2.2 WEB应用和HTTP协议

- **HTTP报文类型**
  - HTTP祈求报文
  - HTTP响应报文

2.2 WEB应用和HTTP协议

- **HTTP祈求报文**
  - 一段经典的HTTP祈求报文

祈求行
(GET, POST,
HEAD 命令)

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

首部
诸行

(额外的 carriage return, line feed)

回车、换行表达
报文结束

2.2 WEB应用和HTTP协议

- HTTP祈求报文的一般格式

2.2 WEB应用和HTTP协议

- **祈求行支持的措施**
  - HTTP1.0 定义的措施
    - GET
      - 向服务器祈求指定URL的对象
    - POST
      - 用于向服务器提交表单数据
      - 也能够同步祈求一种WEB页面
      - 尤其注意：能够不使用POST措施，而使用GET措施发送表单数据以获取新的WEB页面。e.g. 搜索引擎
    - HEAD
      - 祈求服务器返回一种响应报文，但是该报文中并不包括祈求的对象。该措施经常用来进行故障跟踪。

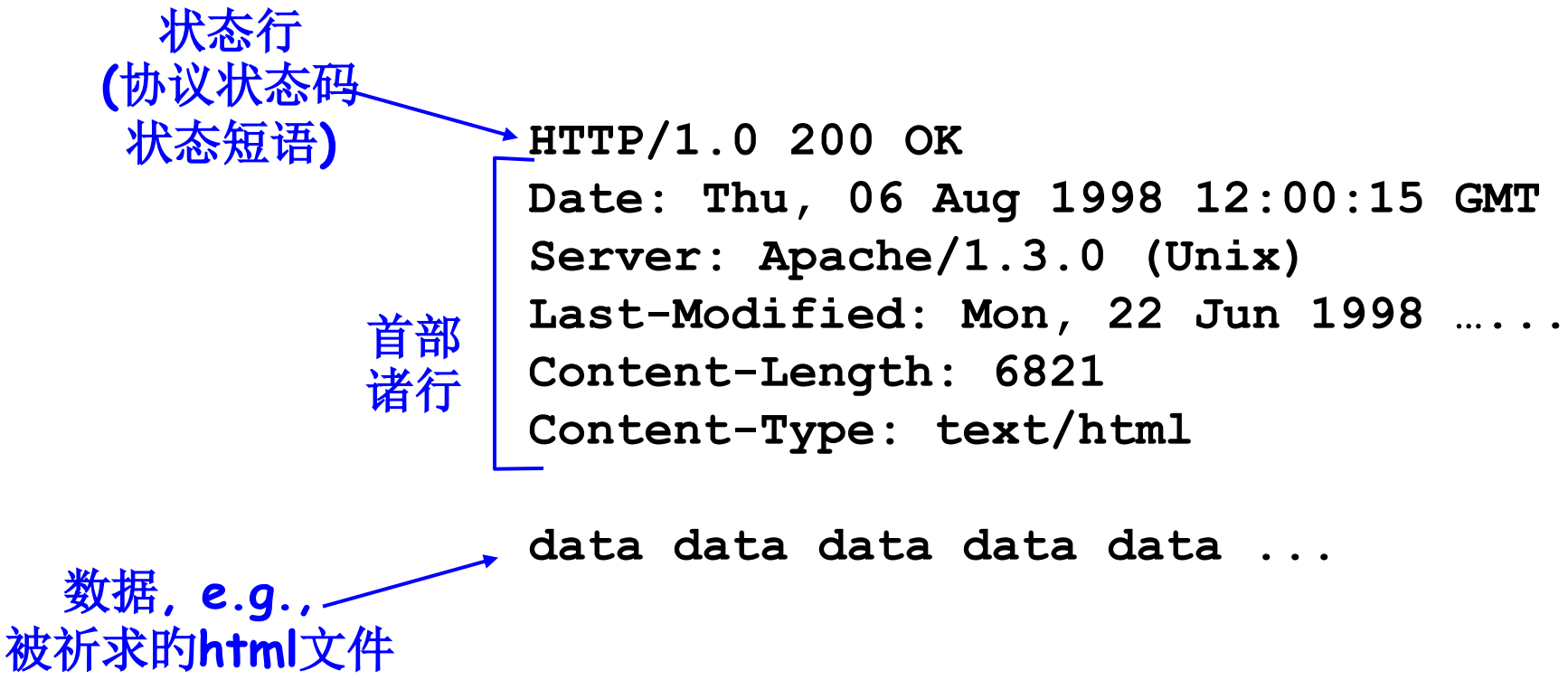
下续

2.2 WEB应用和HTTP协议

- HTTP1.1新定义的措施
  - PUT
    - 上传的文件放在实体主体字段中，目的途径由URL字段标明
  - DELETE
    - 删除URL字段中指定的文件

2.2 WEB应用和HTTP协议

- **HTTP响应报文**
  - 一段经典的HTTP响应报文



2.2 WEB应用和HTTP协议

- HTTP响应报文的一般格式

2.2 WEB应用和HTTP协议

- **常见的HTTP响应状态码和短语**

200 OK

- 祈求成功, 被祈求的对象在报文中

301 Moved Permanently

- 被祈求的对象被移动过, 新的位置在报文中有阐明(Location:)

400 Bad Request

- 服务器不懂祈求报文

404 Not Found

- 服务器上找不到祈求的对象

505 HTTP Version Not Supported

- 服务器不支持祈求报文使用的HTTP协议版本

2.2 WEB应用和HTTP协议

- **了解HTTP报文格式的最佳措施就是自行测试**

1. 用Telnet 连接测试用的服务器:

```
$telnet www.hust.edu.cn 80
```

打开 TCP 连接到 port 80 (默认的http 服务器端口) 位于后续键入的内容将发送到 的 80 号端口

2. 键入一条 http祈求报文:

```
GET /content/content_11786.html
```

将该指令键入后 (按两次回车键), 就将此最短之 (但是完整的) GET 祈求发到了 http 服务器

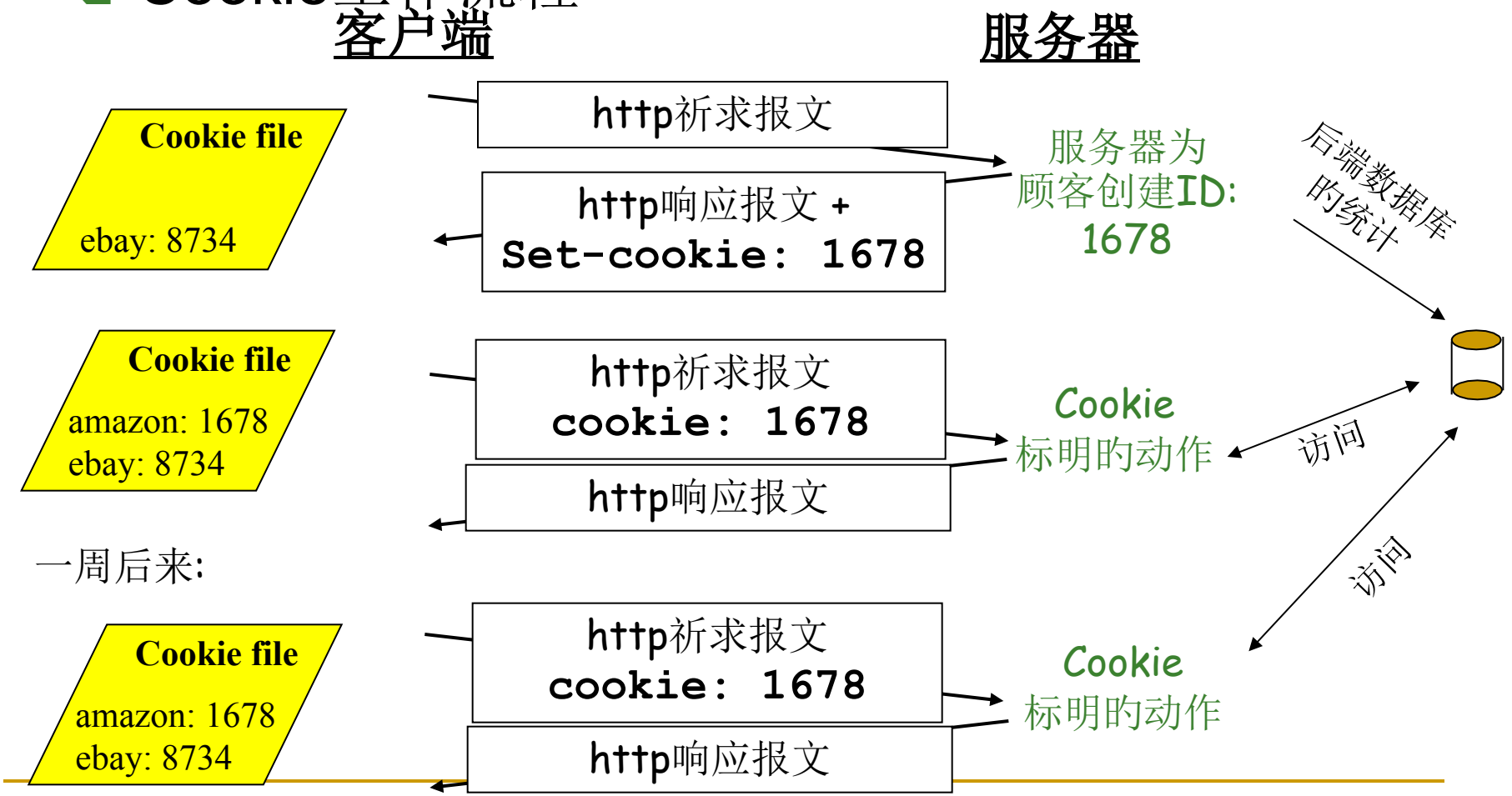
3. 请注意观察http服务器发回的响应报文!

2.2 WEB应用和HTTP协议

- **顾客-服务器交互：Cookie**
  - WEB站点使用Cookie的目的
    - 限制顾客的访问
    - 把内容和顾客身份关联起来
  - Cookie技术的构成部分:
    - 在HTTP响应报文中有一种Cookie首部行
    - 在HTTP祈求报文中也有一种Cookie首部行
    - 在顾客的端系统中保存了一种Cookie文件，由顾客浏览器负责管理
    - 在Web站点有一种后端数据库

2.2 WEB应用和HTTP协议

- Cookie工作流程



2.2 WEB应用和HTTP协议

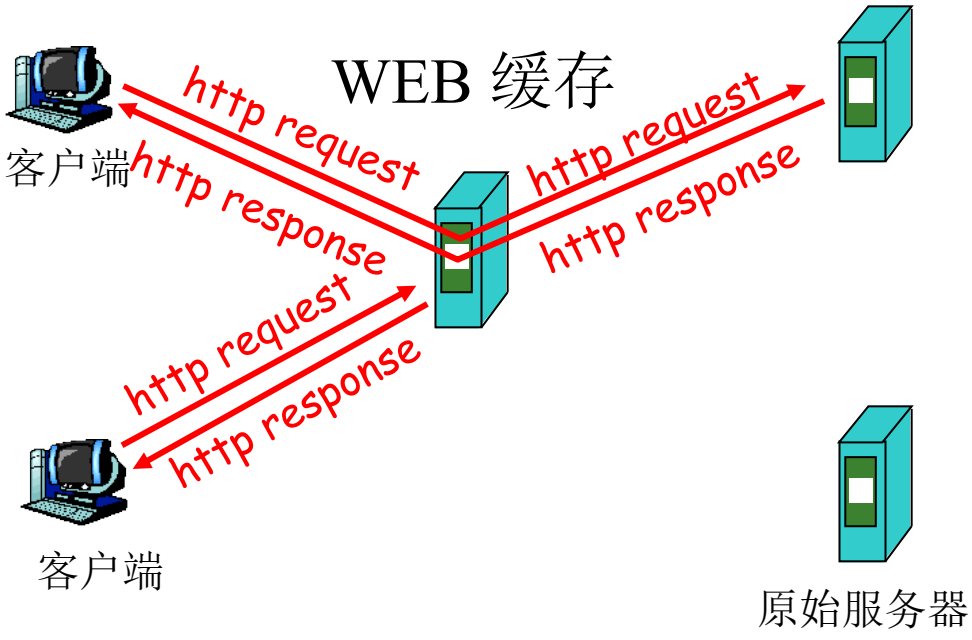
- Cookies能为我们带来什么好处呢？
  - 认证
  - “购物车”
  - “推荐”
  - 顾客会话状态 (Web e-mail)

Cookies和私密性:

- Cookies允许网站取得相当多的顾客的信息
- 你可能会向网站提供你的姓名和E_Mail地址
- 搜索引擎也能够使用cookie和重定向技术取得诸多的信息
- 广告企业也能够经过顾客访问过的网站来取得顾客的有关信息

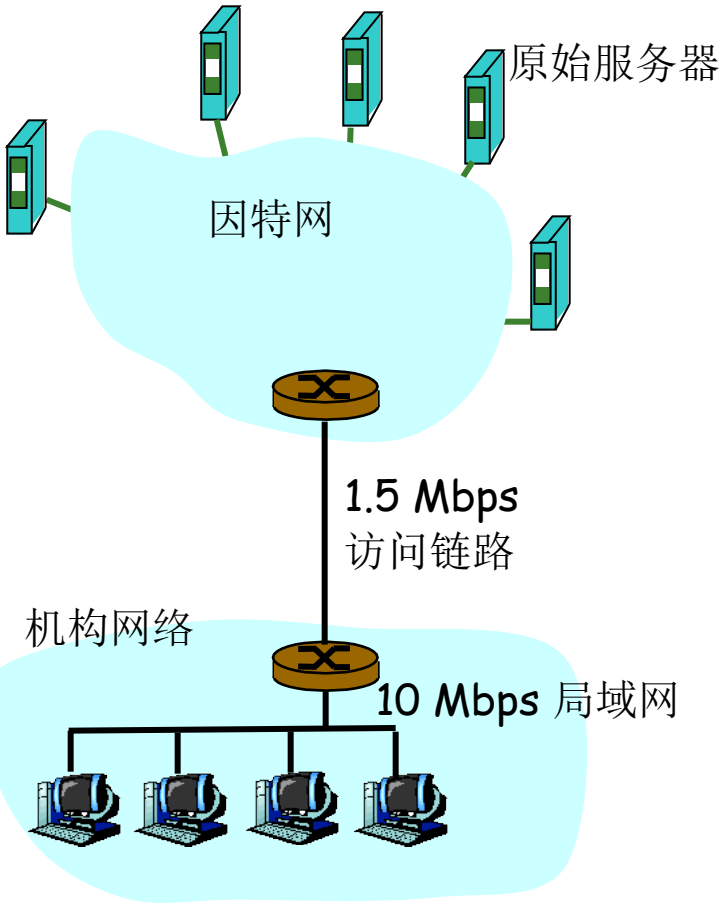
2.2 WEB应用和HTTP协议

- **WEB缓存**
  - 目的
    - 加速客户端访问WEB页面的速度，降低时延
    - 降低局域网与外部因特网互换的数据量，从而在到达同等服务质量的同步，能够使用较小的网络带宽，节省费用



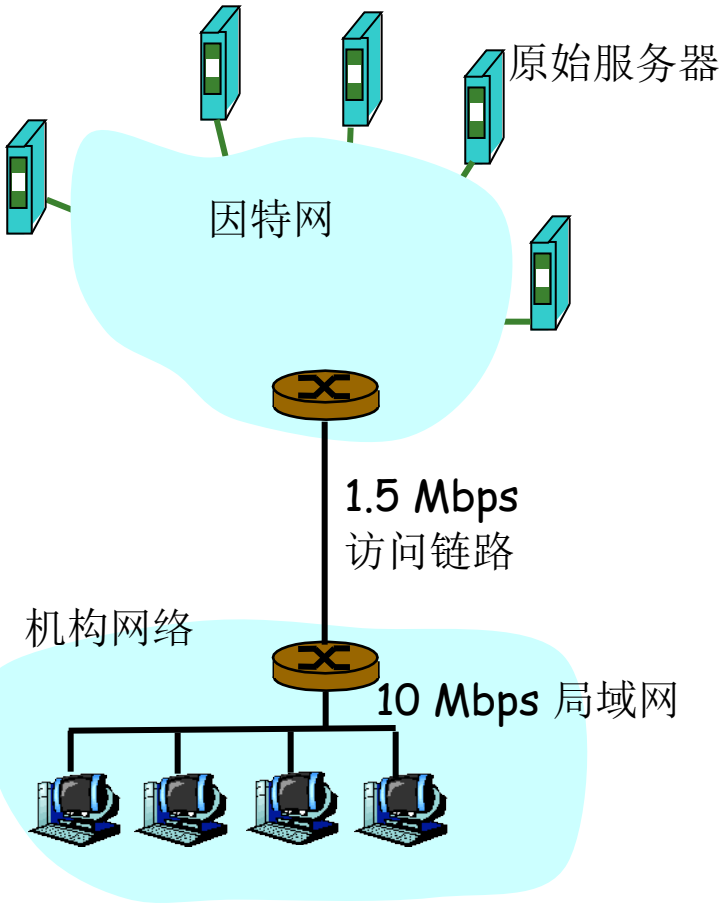
2.2 WEB应用和HTTP协议

- **缓存举例**
  - 假设
    - 平均对象的大小 = 100,000 bits
    - 浏览器对这些对象的平均访问速率为15个/秒
    - 从因特网一侧的路由器转发HTTP祈求到它收到响应报文的平均时间为 2秒



2.2 WEB应用和HTTP协议

- 成果
  - 总延迟 = Internet延迟 + 访问延迟 + 局域网延迟
  - 局域网的流量强度 =0.15
  - 接入链路的流量强度 =1
  - 当流量强度为1时，时延可能非常大，从而造成总时延可能得以分钟计了



流量强度=La/R=15个祈求/秒 x 100kb/祈求/R

2.2 WEB应用和HTTP协议

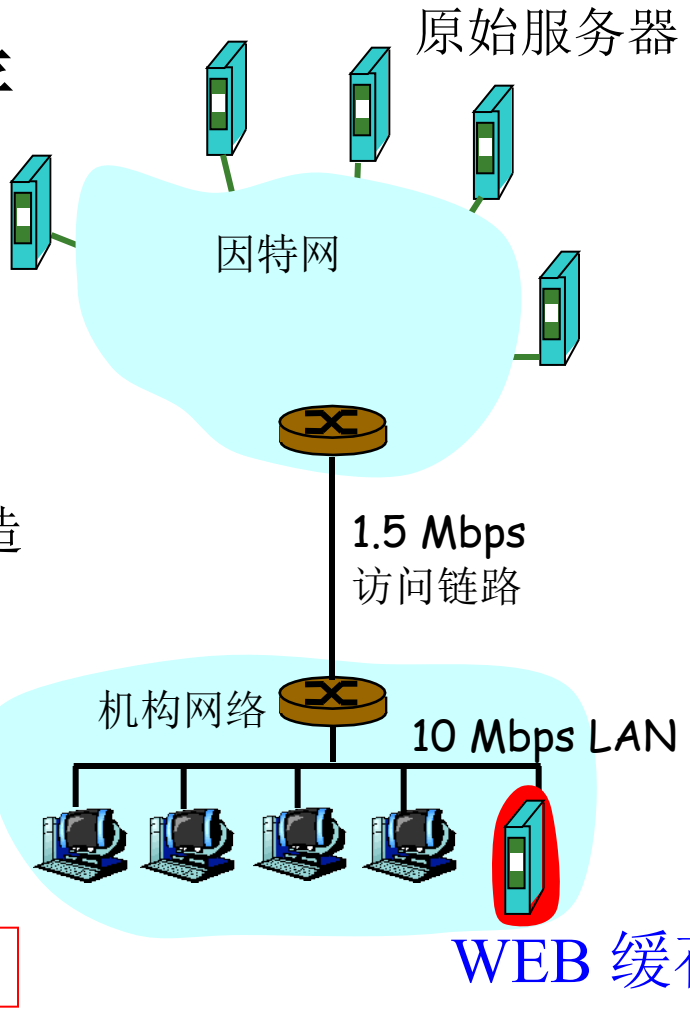
- **改善方案1——增长出口带宽**
  - 假设
    - 增长到10Mbps
  - 成果
    - 局域网的流量强度 =0. 15
    - 接入链路的流量强度 =0.15
    - 总延迟 = Internet延迟 + 访问延迟 + 局域网延迟
      ≈ 2 sec

请注意：增长出口带宽的费用是非常昂贵的
在武汉，1.5Mbps的ADSL，费用大约为130元/月左右
2Mbps的DDN数字链路，费用大约为6000-12023元/月左右

2.2 WEB应用和HTTP协议

- **改善方案2 ——架设WEB缓存**
  - 假设命中率为0.4
  - 成果
    - 40% 的祈求几乎能够立即得到响应
    - 60% 的祈求必须从服务器上取得响应
    - 接入链路的流量强度降低到0.6, 其造成的延迟能够忽视（例如10msec）。
    - 总的平均延迟 = Internet 延迟 + 访问延迟 + 局域网延迟
      ≈ 0.6 x (2.01) 秒 + 0.4 x (0.01s)
      略不小于1.2 secs，好于方案1

代价：一台一般PC+一套免费的WEB缓存软件



2.2 WEB应用和HTTP协议

- **条件GET措施的使用**
  - 目的：更新WEB缓存中的WEB对象副本
  - 举例
    - WEB缓存向WEB服务器发送祈求报文

```
GET /fruit/kiwi.gif HTTP/1.1
```