

# 第八章 应用层

崔 勇

清华大学



计算机网络  
教案社区

## 致谢社区成员

广州大学 姜誉	华北理工大学 郭海龙
西安理工大学 王志晓	西南大学 于显平
清华大学 崔勇	河北大学 蔡红云
河北大学 梁晓艳	河北大学 杨晓晖



# 回顾与展望

## ➤ 经典传输协议

- 端到端传输数据并分发给具体应用
- UDP：无连接、不可靠
- TCP：有连接、可靠

## ➤ 应用如何百花齐放

- baidu.com还是182.61.200.6 or IPv6?
- 文字/图片/视频还是控制？语义是核心
- 邮件、web还是流媒体？即时通信？

可靠性、时延、带宽，应用关心什么？





# 本章目标

1. 掌握应用**进程通信**方式以及**服务进程**工作模式
2. 掌握**域名系统DNS**基本原理和工作机制
3. 掌握**电子邮件系统**体系结构及基本工作原理
4. 了解**WWW系统**结构及其应用技术，掌握**HTTP**及其工作原理
5. 了解**流媒体**基本概念和流媒体动态自适应传输



# 本节内容

- 8.1 应用层概述
  - 8.2 域名系统
  - 8.3 电子邮件
  - 8.4 Web和HTTP
  - 8.5 流式音频和视频
1. 应用进程通信方式
  2. 服务器进程工作方式
  3. 选取合适的传输服务



# 应用进程通信方式

- 不同主机上应用进程间的通信和协同
  - **应用进程**: 为解决具体应用问题而彼此通信的进程
  - **每个应用层协议是为了解决某一类应用问题**
  - 两台主机通信实际是其上对应的两个应用进程(process)在通信
- **应用层**: 规定应用进程之间在通信时所遵循的协议

客户/服务器 (**C/S**, Client/Server) 方式

浏览器/服务器 (**B/S**, Browser/Server) 方式

对等 (**P2P**, Peer to Peer) 方式



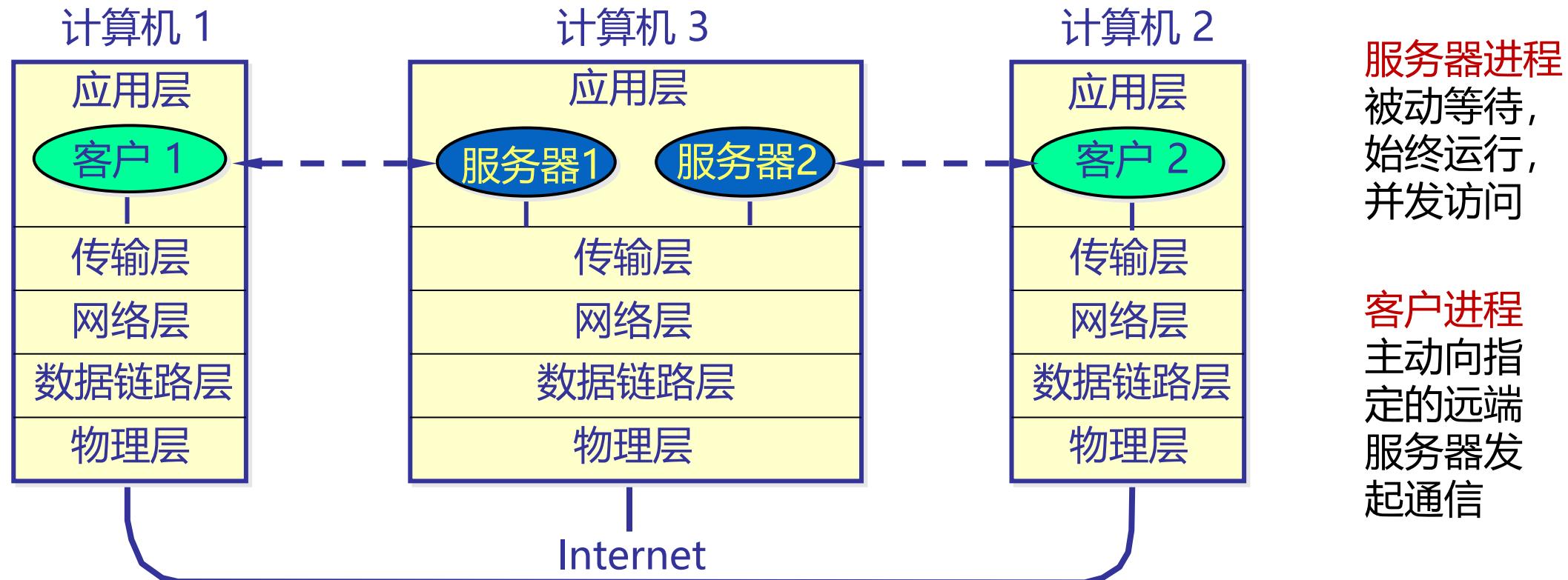
## (1) 客户/服务器 (C/S, Client/Server) 方式

- 应用层的许多协议是基于C/S方式
  - 在移动互联网环境下，每个应用APP都是一个客户端
  - 客户(client)和服务器(server)是指通信中所涉及的2个**应用进程**
  - **客户/服务器**方式描述的是**应用进程之间服务和被服务**的关系
  - **客户**是服务请求方 (**主动**请求服务，被服务)
  - **服务器**是服务提供方 (**被动**接受服务请求，提供服务)
- C/S方式可以是面向连接的，也可以是无连接的
  - 面向连接时，**C/S通信关系一旦建立，通信就是双向的，双方地位平等，都可发送和接收数据**



# 应用进程通信方式 C/S

- 功能较强的计算机可同时运行多个服务器进程





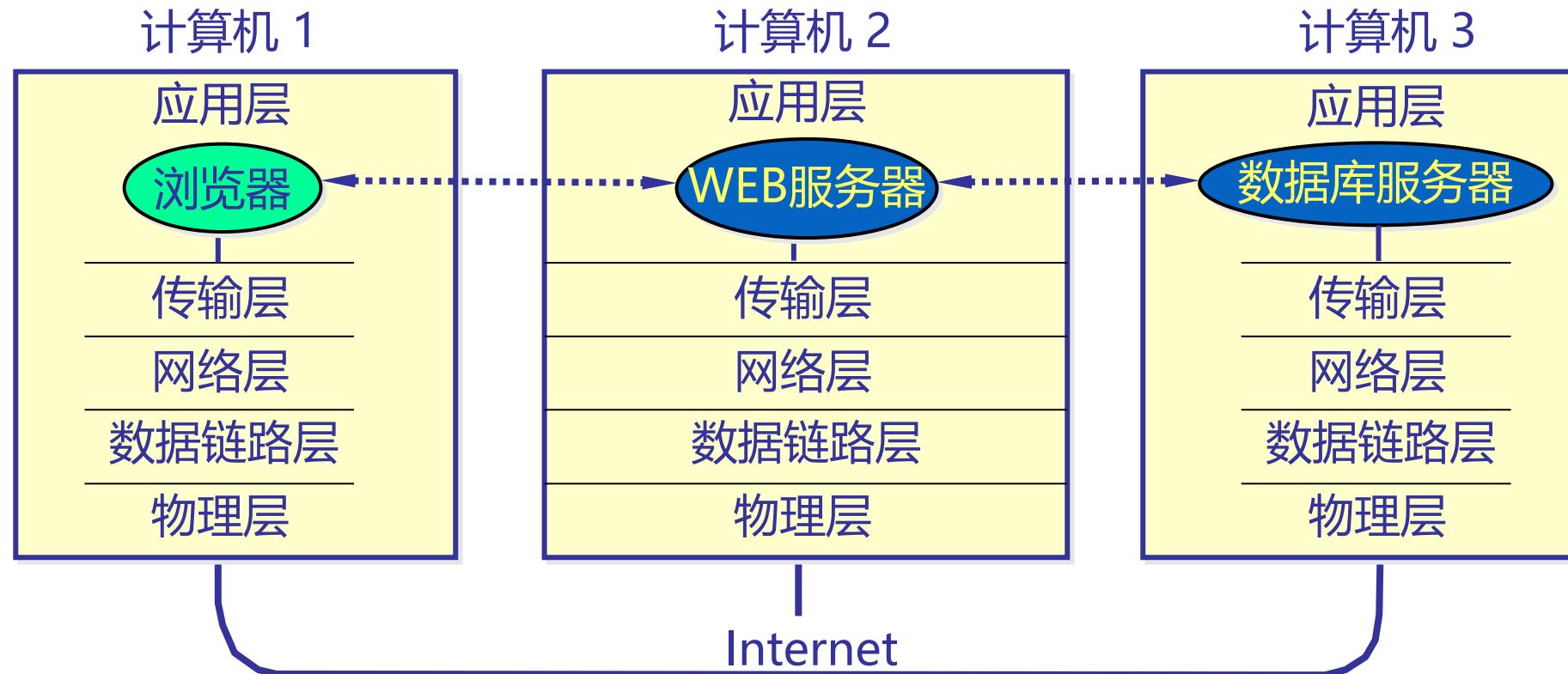
## (2) 浏览器/服务器(B/S, Browser/Server) 方式

- B/S方式可以看做**C/S方式**的特例，即客户软件改为浏览器了
- B/S方式采取**浏览器请求、服务器响应**的工作模式
- 在B/S方式下，**用户界面完全通过Web浏览器实现**，一部分事务逻辑在前端实现，但主要的事务逻辑在**服务器端**实现





# 应用进程通信方式 B/S



B/S方式的3层架构示意



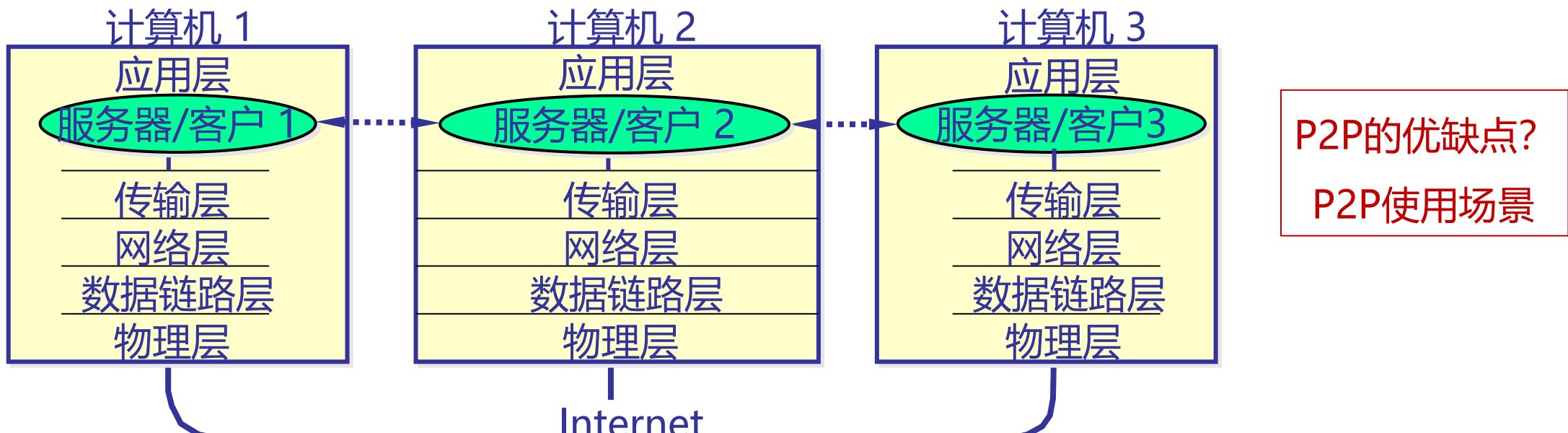
## ➤ B/S方式的特点

- 界面统一，使用简单。客户端只需要安装**浏览器**软件
- 易于维护。对应用系统升级时，只需更新服务器端的软件，减轻了系统维护和升级的成本
- 可扩展性好。采用标准的**TCP/IP**和**HTTP协议**，具有良好的扩展性
- 信息共享度高。HTML是数据格式的一个**开放标准**，目前大多数流行的软件均支持**HTML**
- 需要注意的是，在一种浏览器环境下开发的界面在另一种浏览器环境下可能有不完全适配的情况，这时需要安装对应的浏览器



## (3) 对等 (P2P, Peer to Peer) 方式

- 对等方式是指两个进程在通信时并不区分服务的请求方和服务的提供方
- 只要两个主机都运行P2P软件，它们就可以进行平等、对等的通信
- P2P方式从本质上看仍然是使用了C/S方式，但强调的是通信过程中的对等，这时每一个P2P进程既是客户同时也是服务器





# 服务器进程工作方式

## ➤ 循环方式(iterative mode)

- 一次只运行一个服务进程
- 当有多个客户进程请求服务时，服务进程就按请求的先后顺序依次做出响应 (阻塞方式)

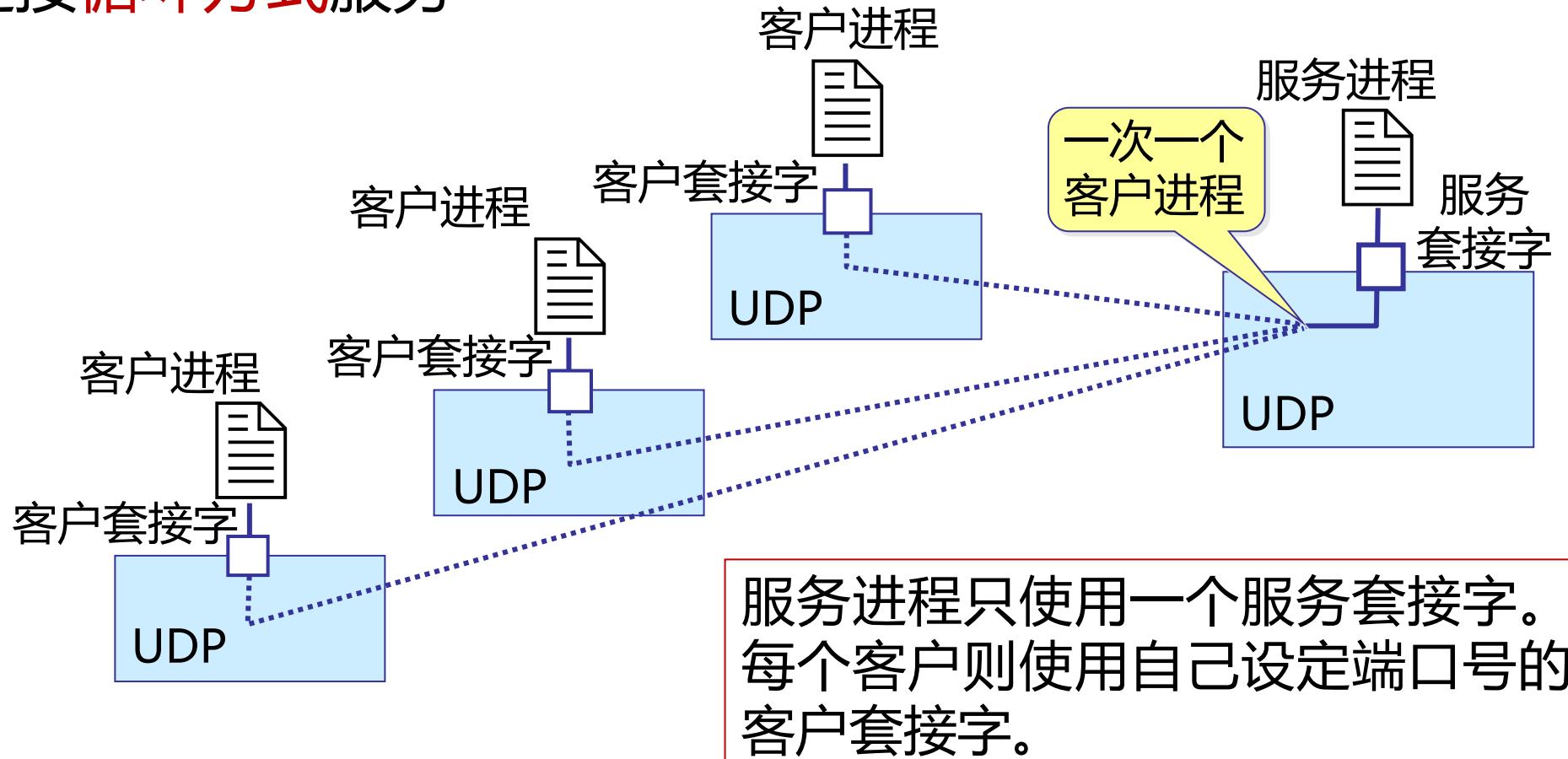
## ➤ 并发方式(concurrent mode)

- 可以同时运行多个服务进程
- 每一个服务进程都对某个特定的客户进程做出响应 (非阻塞方式)



# 服务器进程工作方式

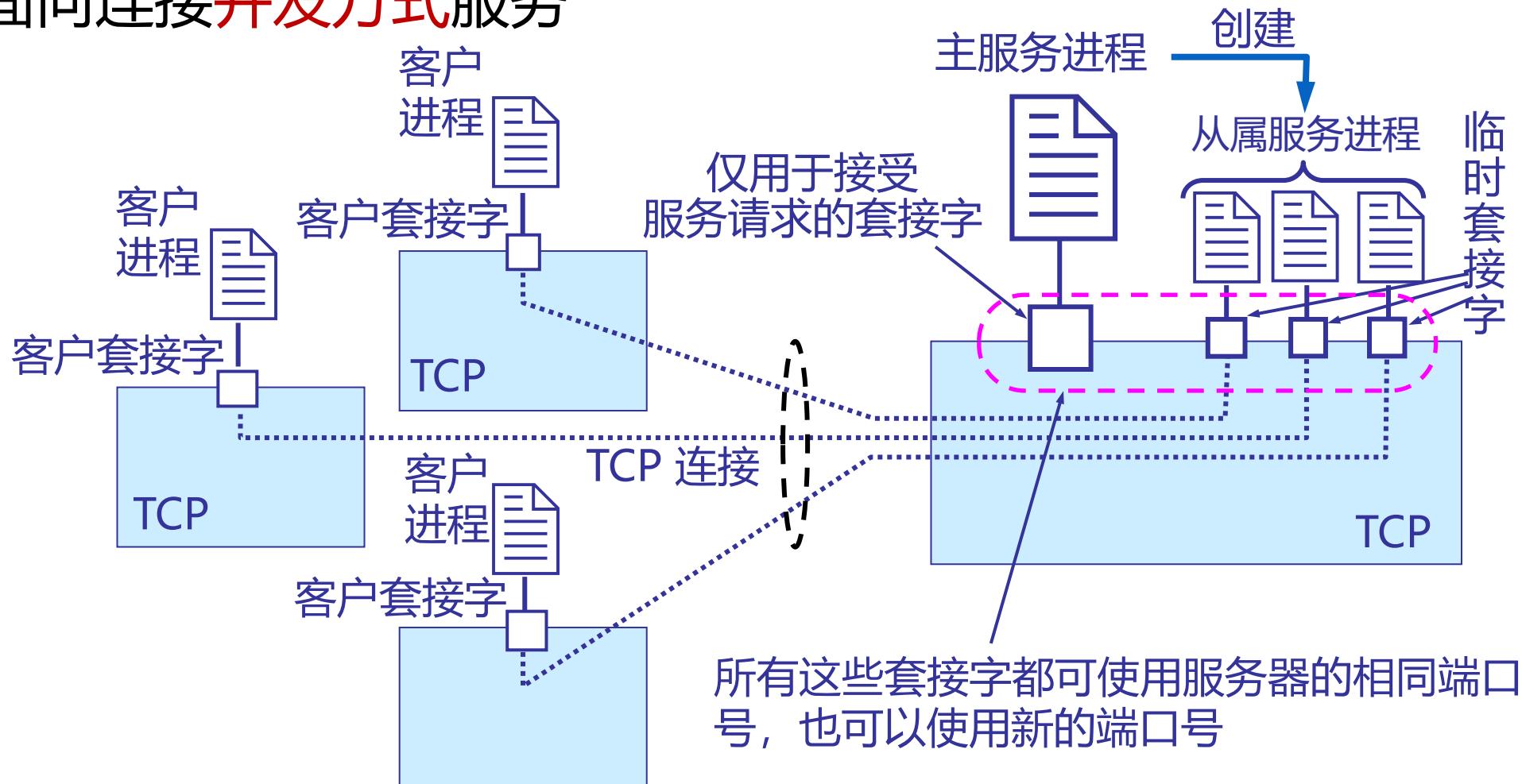
## 无连接循环方式服务





# 服务器进程工作方式

## 面向连接并发方式服务





# 选取合适的传输服务

- 应用程序需要什么样的传输服务?
- 数据可靠
  - 有的应用程序(如文件传输)要求100%的可靠传输
  - 有的应用程序(比如音频)可以容忍一定程度的数据丢失
- 带宽
  - 有的程序(如多媒体)需要一定的带宽才能工作
  - 有的程序则使用它所能得到的全部带宽
- 延迟
  - 有的程序要求低延迟, 比如IP电话和交互游戏
  - 超低时延: VR、云游戏和工业控制



# 选取合适的传输服务

应用	数据丢失	带宽	时间敏感
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: 5Kb-1Mb video:10Kb-5Mb	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
financial apps	no loss	elastic	yes and no



# 选取合适的传输服务

## TCP服务：

- 面向连接：用户端和服务器需要建立连接
- 接收和发送进程间的可靠传输
- 流量控制：发送方不会淹没接收方
- 拥塞控制：网络负载过高时限制发送方发送
- 不提供：延迟保证，最小带宽保证

## UDP服务：

- 接收和发送进程间的不可靠传输
- 不提供：连接建立，可靠性、流量控制、拥塞控制和带宽保证

需要新的传输服务?  
怎么办?

协议标准 v.s. 私有协议



# 本节内容

8.1 应用层概述

8.2 域名系统

8.3 电子邮件

8.4 Web和HTTP

8.5 流式音频和视频

1. 历史和概述
2. 域名系统名字空间和层次结构
3. 域名服务器
4. 域名解析过程
5. 域名系统查询和响应(选讲)
6. 域名系统高速缓存
7. 域名系统隐私(选讲)

我记不住 IP 怎么办?



清华大学  
Tsinghua University



计算机网络教案社区



## ➤ ARPANET

- Hosts.txt: 列出所有计算机名字与它们IP地址
- 网络规模不大时工作得很好

## ➤ 域名系统DNS (Domain Name System)

- 互联网重要的基础设施之一，向所有需要域名解析的应用提供服务，主要负责将可读性好的**域名映射成IP地址**
- Internet采用**层次结构的命名树**作为主机的名字，并使用**分布式的**域名系统DNS，Internet的DNS是一个**联机分布式数据库系统**
- 名字到域名的解析是由若干个**名字服务器**(Name Server)或**域名服务器**(Domain Name Server)完成的





## Internet的域名结构

- Internet的域名结构采用了**层次树状结构的命名方法**
- 域名的结构由若干个分量组成，各分量之间用**小数点(.)隔开**，总长不超过255个字符
- 合法域名中，点“.”的个数至少为一个
- 通常，点“.”对应的英文单词为dot，也可以读为point

... .**三级域名.二级域名.顶级域名**

www.tsinghua.edu.cn

mails.tsinghua.edu.cn



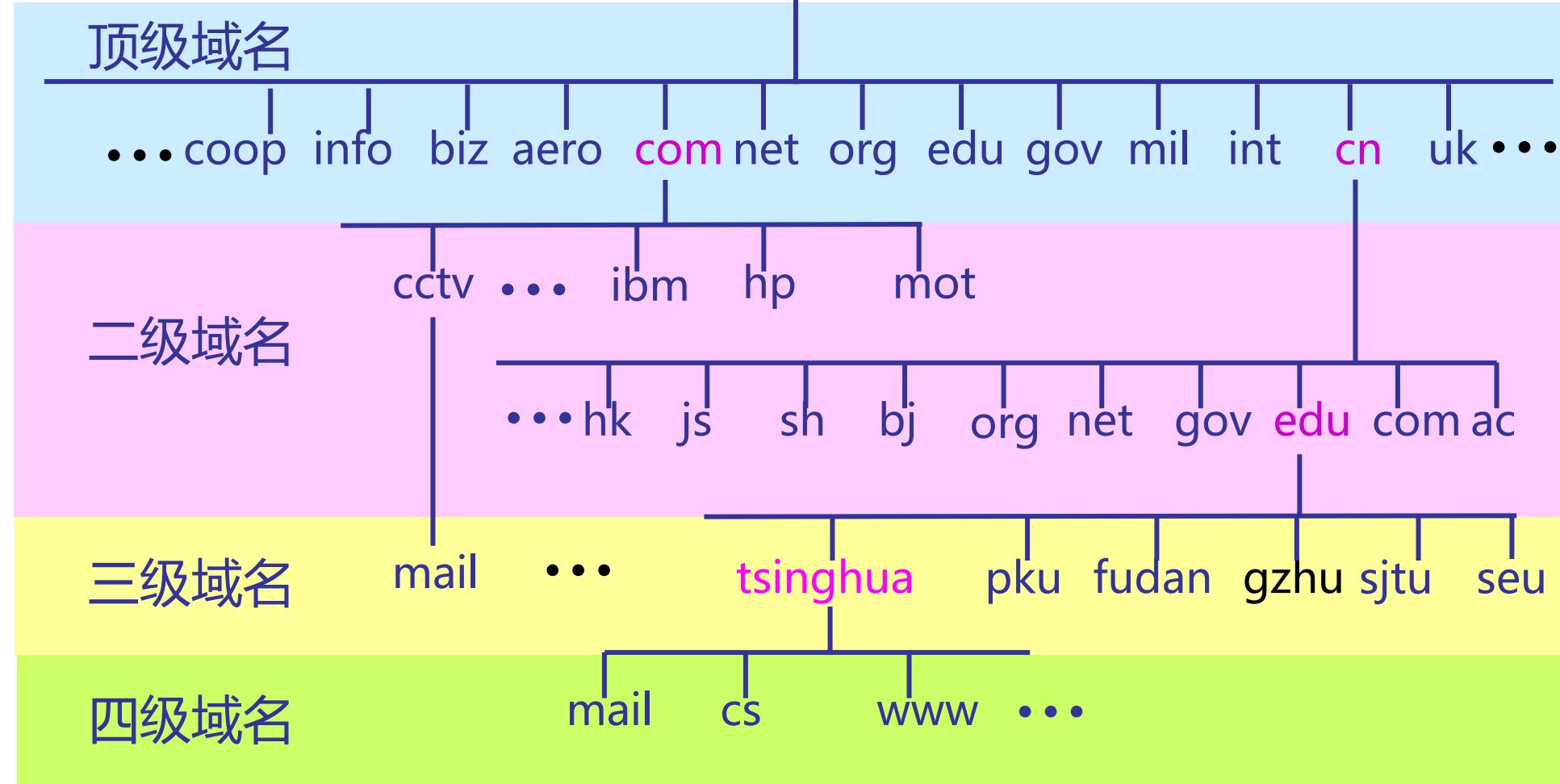
# 域名系统名字空间和层次结构

按级划分

树根

层次树状结构

域树





## 早期的通用顶级域名

- .com 表示公司企业 (commerce)
- .net 表示网络服务机构，例如NIC和NOC (network)
- .org 表示非赢利性组织 (organization)
- .edu 表示教育机构(美国专用) (education)
- .gov 表示政府部门(美国专用) (government)
- .mil 表示军事部门(美国专用) (military)
- .int表示政府间国际合约建立的国际性组织 (international)
- .mobi 用于提供移动产品和服务的用户和供应商 (mobile)



## 早期的通用顶级域名

- .museum 用于博物馆
- .name 用于个人
- .pro 用于有资质的专业人员及其实体，例如会计、律师和医师等自由职业者
- .tel 用于商业和个人公布其联系方式
- .travel 用于旅游业实体
- 2012年开始，公司名可以作为新的顶级域名(18万美元)
- 也可以注册 **.中国、.公司、.网络** 等顶级域名(多语种域名国际标准RFC3454、RFC3490、RFC3491、RFC3492)



国家顶级域名 .cn 下的二级域名分为三类

- **类别域名** 7个
  - .edu.cn 教育
  - .gov.cn 政府
  - .org.cn 非营利组织
  - .net.cn 网络服务
  - .com.cn 工商金融等企业
  - .ac.cn 科研
  - .mil.cn 国防机构
- **行政区域名** 34个：省、直辖市、自治区、特区等行政区域名，  
每个行政区域名为两个字母，例如北京bj、河北he等
- **无类别域名**：例如，cuiyong.net 等



# 域名系统名字空间和层次结构

## ➤ 域名管理机构分级负责域名注册

- Internet域名管理机构:  
ICANN (Internet Corporation for Assigned Names  
and Numbers)  
[www.icann.org](http://www.icann.org)
- 二级域名该国自行确定
- 三级域名注册由其所属二级域名机构负责, 以此类推

## ➤ 我国的域名注册机构

- 二级域名注册由中国互联网络信息中心(CNNIC)负责
- .edu.cn下三级域名注册由CERNET负责

## ➤ 历史上的开心网

- kaixin001.com? kaixin.com

开心吗?

申请注册域名



注册服务商



注册管理机构



ICANN

注册管理机构与注册服务商之间的关系

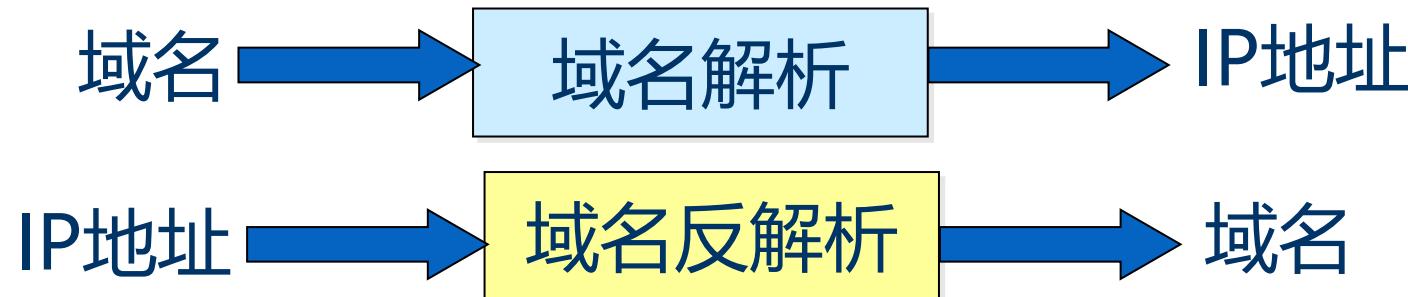


## ➤ 域名服务器

- 保存关于**域树**(domain tree)的结构和设置信息的服务器程序称为**名字服务器**(name server)或**域名服务器**, 负责域名解析工作

## ➤ 域名解析的目标

- 域名与IP地址可以是**一对一、一对多或者多对一**的关系
- 域名解析过程对用户透明**



域名数量多, 动态性强, 服务器信息不全怎么办?  
(1) 各司其职, (2) 互相帮助!



## 名字服务器类别

- 域名系统的名字服务器分为两大类
  - **权威名字服务器**(authoritative name server)
    - 一种根据**本地**知识知道一个DNS区(zone)的内容的服务器，它可以回答有关该DNS区的查询而无需查询其他服务器
    - 每个DNS区至少应有一个IPv4可访问的权威名字服务器提供服务
    - 如：dns.tsinghua.edu.cn
  - **递归解析器**(recursive resolver)/递归服务器
    - 以递归方式运行的、使用户程序联系域(domain)名字服务器的程序



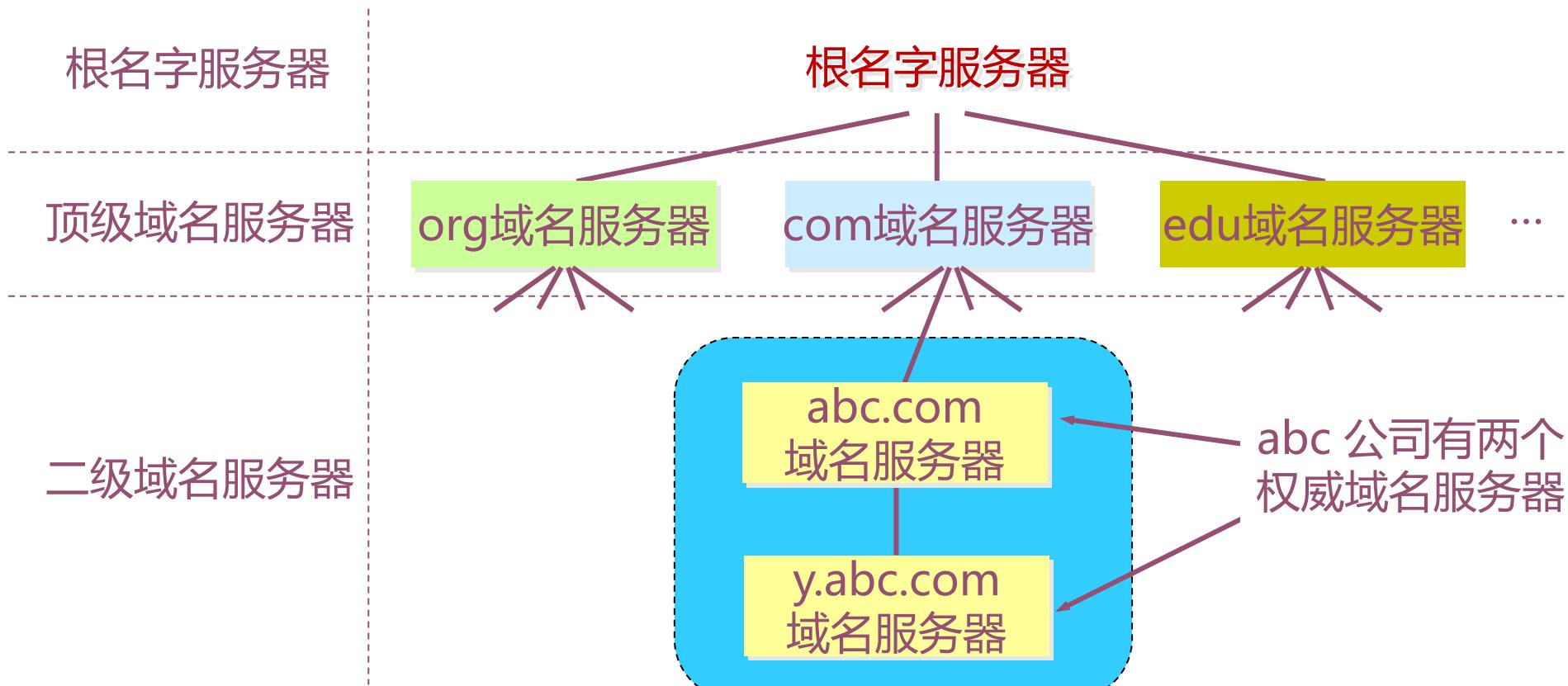
## 权威名字服务器类别

- 根据对应域的层次，权威名字服务器又进一步分为以下类别
  - 根名字服务器(root name server) /**根服务器**(root server)
  - 顶级域名字服务器(TLD name server)
  - 二级域名字服务器(second level domain name server)
  - 三级域名字服务器(third level domain name server)
  - .....
- **本地域名服务器**
  - 三级域及以下的名字服务器(例如gzhu.edu.cn)通常在用户本地区域
  - 因此三级域及以下的名字服务器也统称为**本地域名服务器**



# 域名服务器

## 层次树状结构的权威名字服务器





## ➤ 域名解析过程

- 该应用进程将域名放在DNS请求报文（**UDP数据报, 端口号为53**）发给递归服务器（使用UDP是为了减少开销）
- 递归服务器得到查询结果后, 将对应IP地址放在应答报文中返回给应用进程

## ➤ 域名查询

- 有**递归查询**(recursive query)和**迭代查询**(或循环查询, iterative query)两种方式
  - 主机向递归解析器/本地域名字服务器的查询一般采用递归查询
  - 递归解析器/本地域名字服务器向根服务器可以采用递归查询, 但一般**优先采用迭代查询**

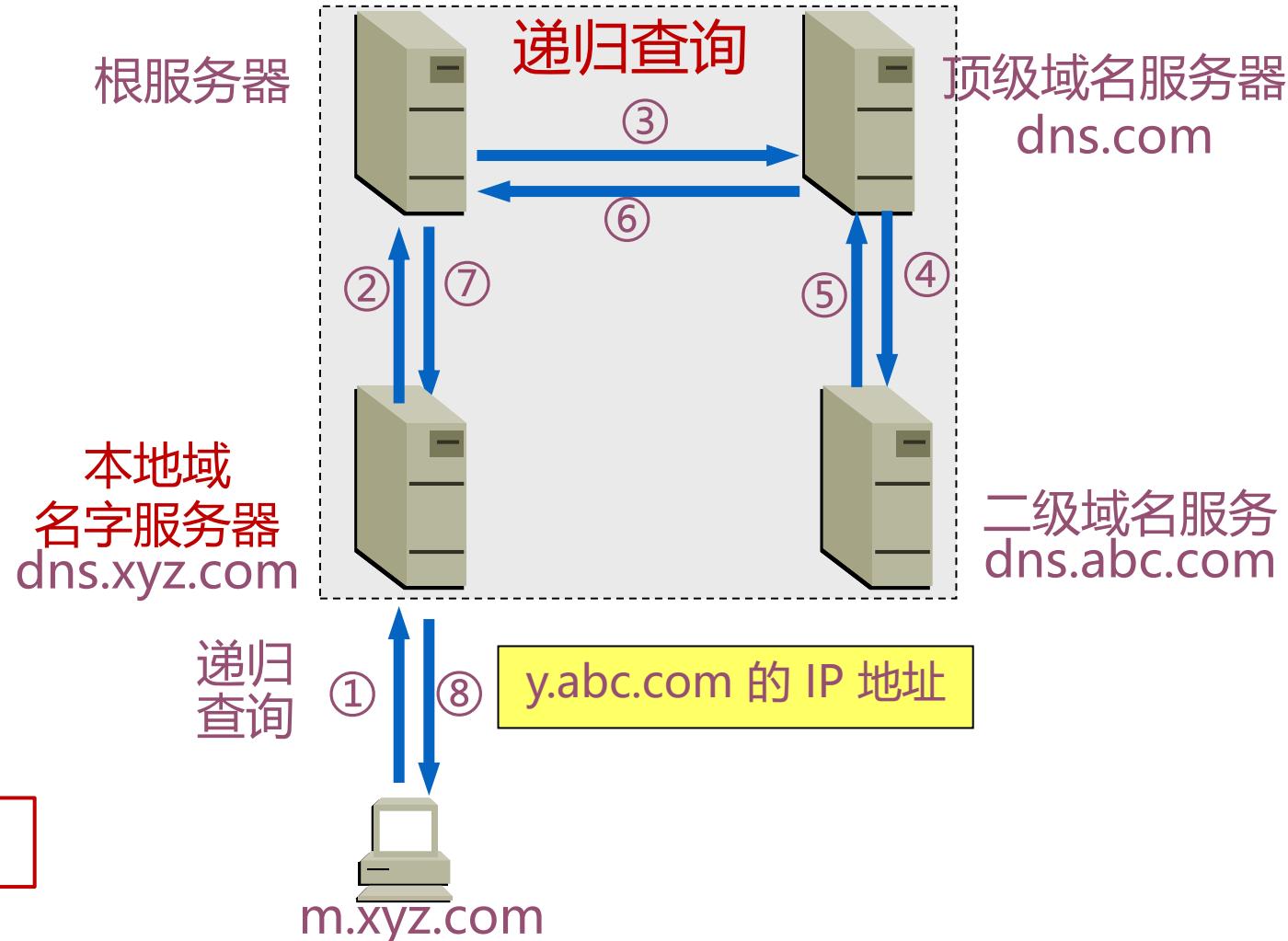


# 域名解析过程

## ➤ 递归查询

- 域名服务器收到查询请求，但不知道被查询域名的IP地址
- 该域名服务器以DNS客户的身份，向下一步域名服务器发出查询请求
- 即替递归服务器继续查询
- 较少使用

为什么呢？





# 域名解析过程

## ➤ 迭代查询

- 域名服务器把下一步应查询的域名服务器IP地址告诉**本地域名字服务器**
- 由本地域名字服务器继续向该域名服务器查询
- 直到得到所要解析的域名的IP地址，或者查询不到所要解析的域名的IP地址
- 通常使用该方式

根服务器

本地域  
名字服务器  
dns.xyz.com

递归  
查询



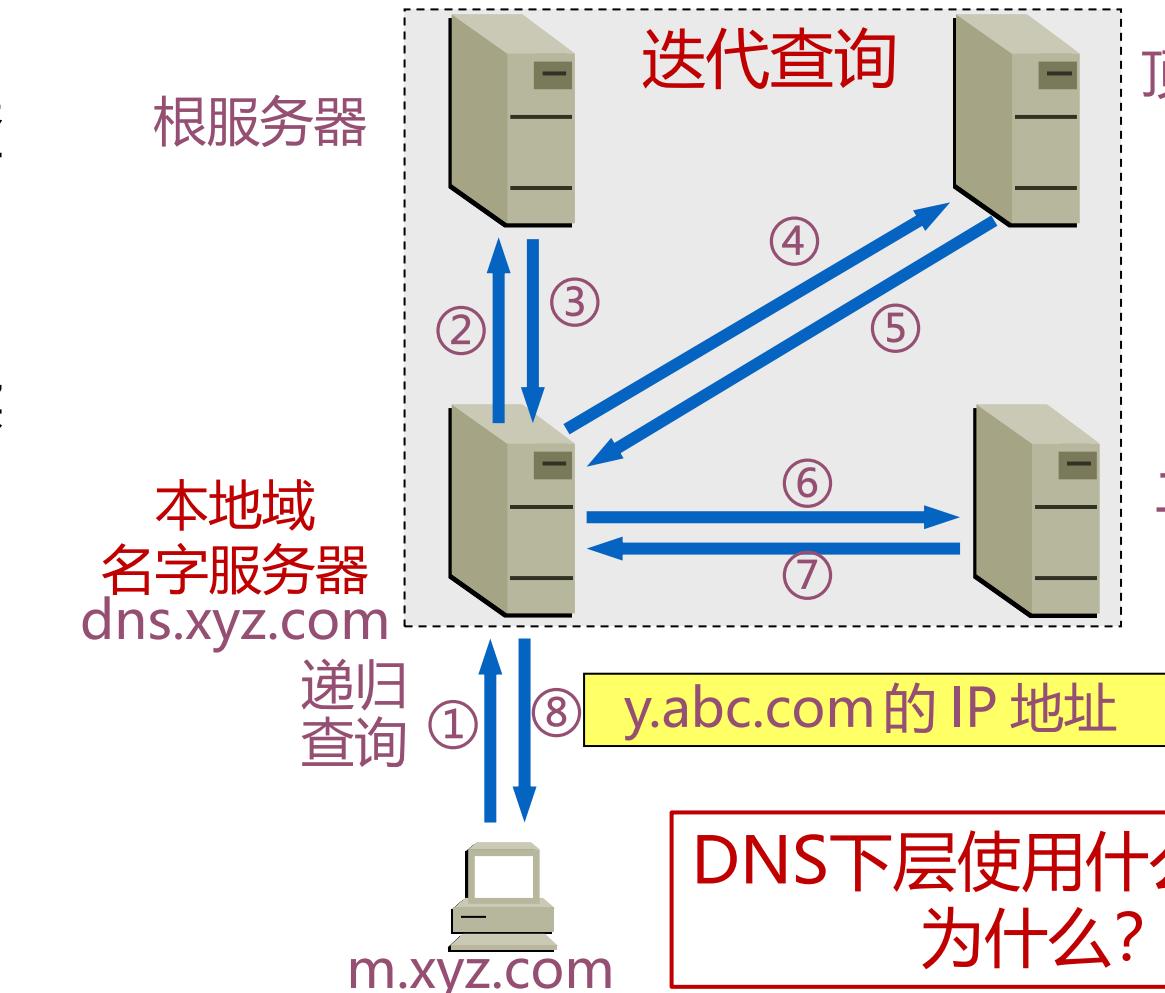
m.xyz.com

迭代查询

顶级域名服务器  
dns.com

二级域名服务器  
dns.abc.com

DNS下层使用什么协议?  
为什么?





## 根服务器

➤ 根服务器共有**13套**(不是13台机器)

- 这些根服务器相应的域名分别是：  
`a.rootservers.net - m.rootservers.net`
- 更改根服务器数据只在`a.rootservers.net`上进行，然后同步到另外12套中，这样既能保证数据一致性，也提高了域名服务可靠性
- 每套都可以有多个镜像(mirrored)根服务器，其内容定期与上述对应的根服务器同步 (**同步需要一定的时间才能完成**)
- 根服务器之间采用任意播(anycast)技术互联
- 目前全球已设置了**1000多台镜像根服务器**，用户域名解析请求时延进一步降低，使世界上大部分DNS域名服务器都能**就近**找到一个根服务器



## DNS报文格式的资源记录RR格式

- **域名**: DNS请求的域名
- **类型**: 资源记录的类型
- **类**: 地址类型
- **生存时间**: 以秒为单位, 表示资源记录的生命周期, 一般用于缓存数据的时间, 也表明该资源记录的稳定程度
- **资源数据长度**: 资源数据的长度
- **资源数据**: 表示按查询段要求返回的相关资源记录的数据

## DNS资源记录格式

域名 (NAME)
类型 (TYPE)
类 (class)
生存时间 (TTL)
资源数据长度 (RDLENGTH)
资源数据 (RDATA)

### Answers

```
dashboard.snapcraft.io: type A,
Name: dashboard.snapcraft.io
Type: A (Host Address) (1)
Class: IN (0x0001)
Time to live: 5
Data length: 4
Address: 91.189.92.18
```

## DNS资源记录抓包示例



DNS资源记录的类型

RR格式: (name, ttl, class, type, value)

➤ **type=A**

- name是主机名
- value是主机IPv4地址

➤ **type=AAAA**

- name是主机名
- value是主机IPv6地址

➤ **type=CNAME**

- name是某些“规范”名称的别名，例如 www.cs.vu.nl.是主机名papac022.vu.nl.的别名
- value是IP地址

➤ **type=MX**

- name是请求域名
- value是与请求域名关联的SMTP邮件服务器的名称

• google.com.	299	IN	<b>AAAA</b>	2a00:1450:4017:804::200e
• dip0.connect.de.	86400	IN	<b>A</b>	46.82.174.69
• www.cs.vu.nl.	60	IN	<b>CNAME</b>	papac022.vu.nl.



# 域名系统高速缓存

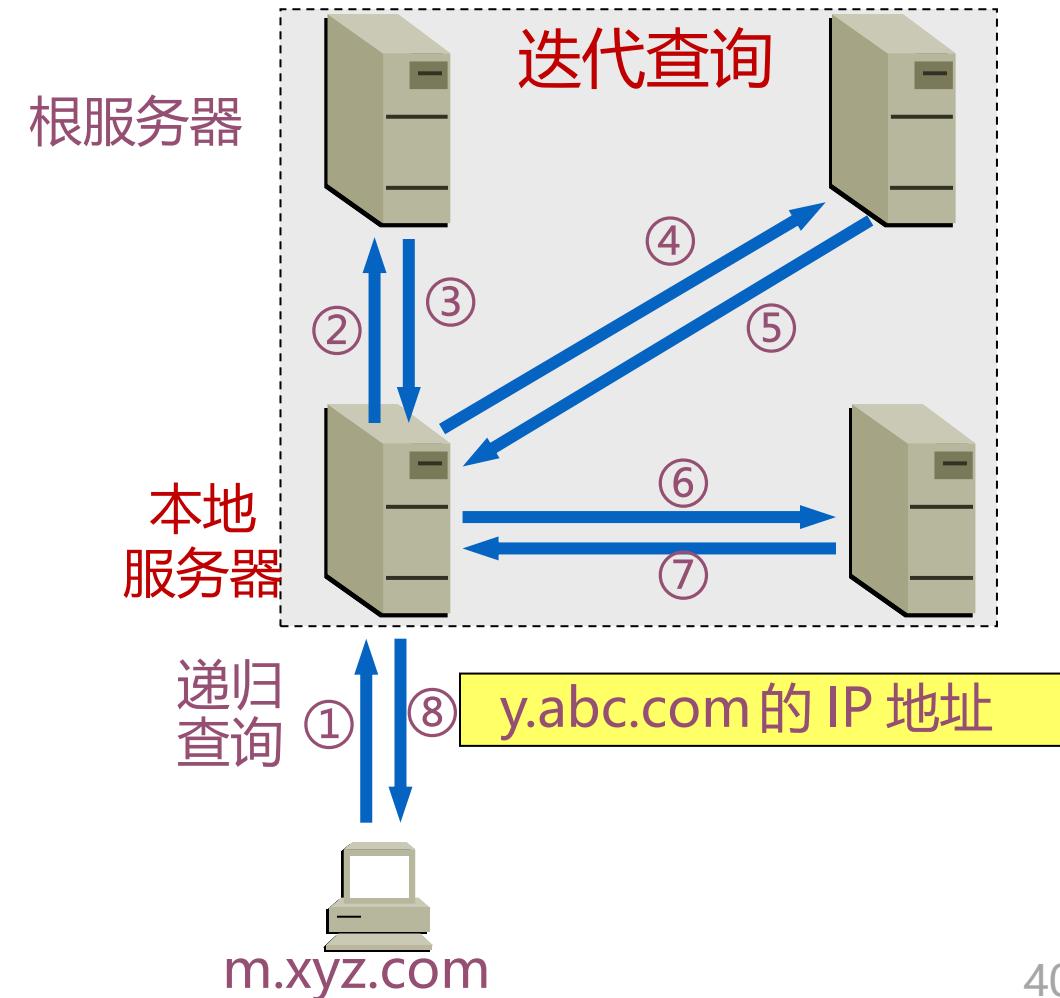
## 如何提升DNS查询效率?

### ➤ 域名系统高速缓存

- 域名服务器广泛使用**高速缓存**
- 存放最近查询过的域名以及从何处获得域名映射信息的记录
- 减轻根域名服务器的负荷
- 使Internet上的DNS查询请求和回答报文的数量大为减少

### ➤ 缓存多长时间呢?

- 缓存中保存数据有一定的时限
- **生存时间TTL**





## ➤ DNS面临安全问题

- 设计之初没有过多考虑**安全问题**
- 几乎所有DNS流量都是基于**UDP明文传输**的
- DNS的资源记录未加上任何的**认证**和**加密措施**

根服务器会被切断吗?  
断了根怎么办?

## ➤ DNSSEC

- 依靠**数字签名**保证DNS应答报文的**真实性和完整性**，包括通过身份验证拒绝DNS数据存在的机制
- 域名服务器用自己的**私有密钥**对**资源记录**(Resource Record, RR)进行**签名**，解析服务器用域名服务器的**公开密钥**对收到的**应答信息**进行**验证**



# 本节内容

- 8.1 应用层概述
- 8.2 域名系统
- 8.3 电子邮件
- 8.4 Web和HTTP
- 8.5 流式音频和视频

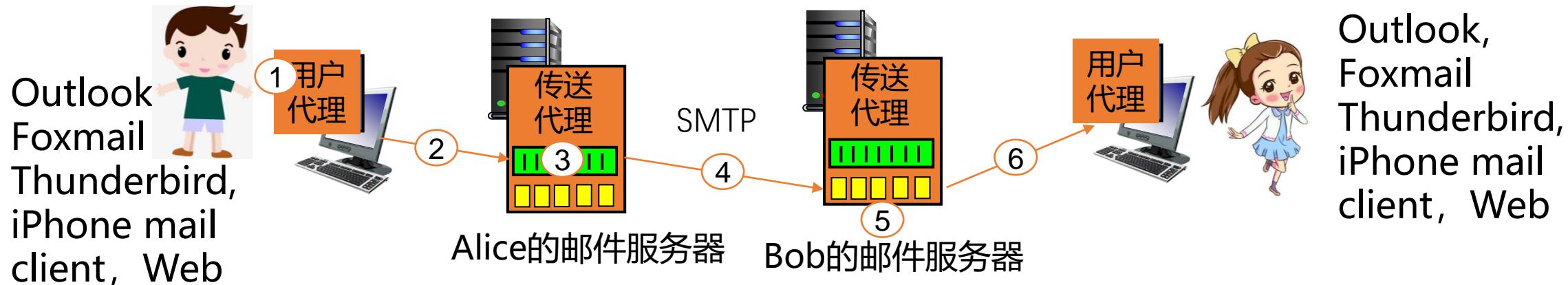
1. 体系结构和服务
2. 用户代理
3. 邮件格式
4. 邮件传送
5. 最后传递

写封信能有多难?  
都是数据，怎么区分纸、字和信封?  
邮箱究竟在哪里？用户的电脑里？



# 电子邮件系统体系结构

- 用户代理 (user agent) —— 邮件客户端
- 传输代理 (message transfer agent) —— 邮件服务器
- 简单邮件传输协议SMTP (Simple Mail Transfer Protocol)



电子邮件系统常采用C/S工作模式



- 邮件传输代理将邮件从发件人中继给收件人
- **SMTP**: 简单邮件传输协议
  - SMTP利用**TCP**可靠地从客户向服务器传递邮件，使用**端口25**
  - SMTP的3个阶段：连接建立、邮件传送、连接关闭
  - **命令/响应** (HTTP为请求/响应)
    - 命令: ASCII字符串
    - 响应: 状态码+短语
  - SMTP是一个简单的ASCII协议



## ➤ 不断重复：命令-响应

- 客户端发送的行用“C:”标识
- 服务发送的行用“S:”标识

## ➤ 来自客户的第一条命令HELO

## ➤ 收件人列表

- RCPT命令可以将一个邮件发送给多个收件人
- 此处只有一个收件人，只使用一个RCPT命令

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@xyz.com>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@someschool.edu>
S: 250 bob@someschool.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
```

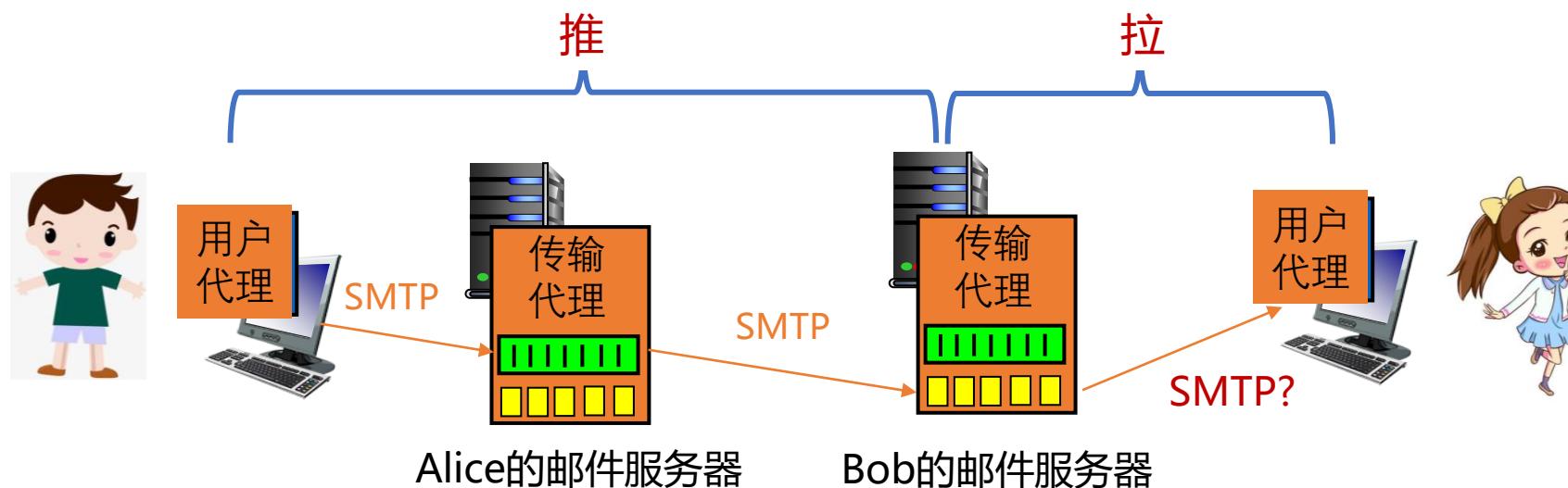
alice@xyz.com给bob@someschool.edu发送一个邮件



# 最后传递

## ➤ 接收方能使用 SMTP 吗？

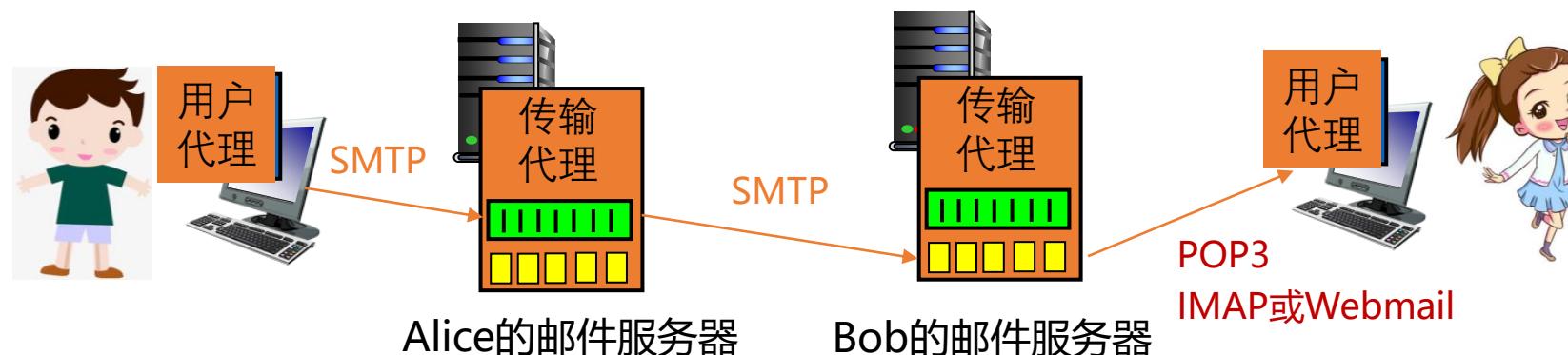
- 邮件服务器总是在线，能随时被访问；但用户代理并非总在线
- 发送邮件的SMTP是一个**推协议**，而取邮件则是一个**拉操作**
- 需要设计新的**拉操作**协议来解决最后传递的问题





## ➤ 最终交付（邮件访问）协议

- 从邮件服务器的邮箱中获取邮件
- 邮件已经到达Bob的邮箱，需要传送到Bob的用户代理以便显示
- **POP3**: Post Office Protocol-Version 3, 第三版邮局协议
- **IMAP**: Internet Message Access Protocol, Internet邮件访问协议
- **Webmail (HTTP)** : 基于Web的电子邮件

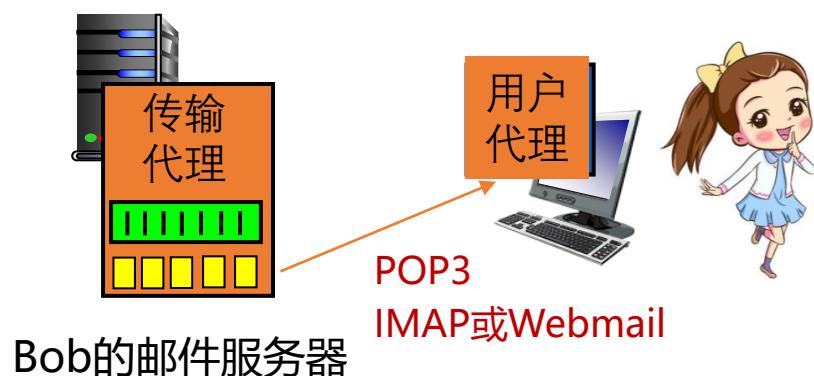




# POP3协议

- POP3由RFC1939定义，是一个非常简单的最终交付协议
  - 当用户代理打开一个到端口110上的TCP连接后，客户/服务器开始工作
- POP3的三个阶段
  - 认证(Authorization): 处理用户登录的过程
  - 事务处理(Transactions): 用户收取电子邮件，并将邮件标记为删除
  - 更新(Update): 将标为删除的电子邮件删除

POP3使用**客户/服务器**工作方式  
在用户PC机中运行POP客户程序，而在邮件  
服务器中则运行POP服务器程序





# POP3的三个阶段

## ➤ 认证阶段

- 客户命令:
  - user: 用户名
  - pass: 密码
- 服务器响应
  - +OK
  - -ERR

S: +OK POP3 server ready

C: user bob

S: +OK

C: pass hungry

S: +OK user successfully logged on



# POP3的三个阶段

## ➤ 事务处理阶段

- list: 邮件列表
- retr: 通过邮件号获取邮件
- dele: 删除

## ➤ 更新阶段

- quit

好简单啊!  
优点还是缺点?

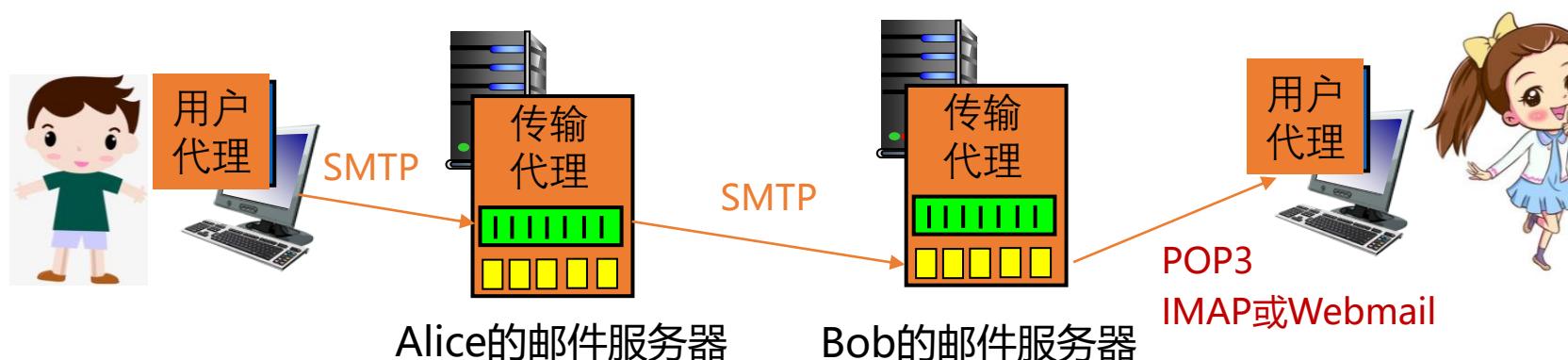
```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```



# IMAP

## ➤ IMAP—Internet邮件访问协议[RFC 2060]

- 用于最终交付的主要协议
- IMAP是POP3的改进版
- 邮件服务器运行侦听**端口143**的IMAP服务器
- 用户代理运行一个IMAP客户端
- 客户端连接到服务器并开始发出命令

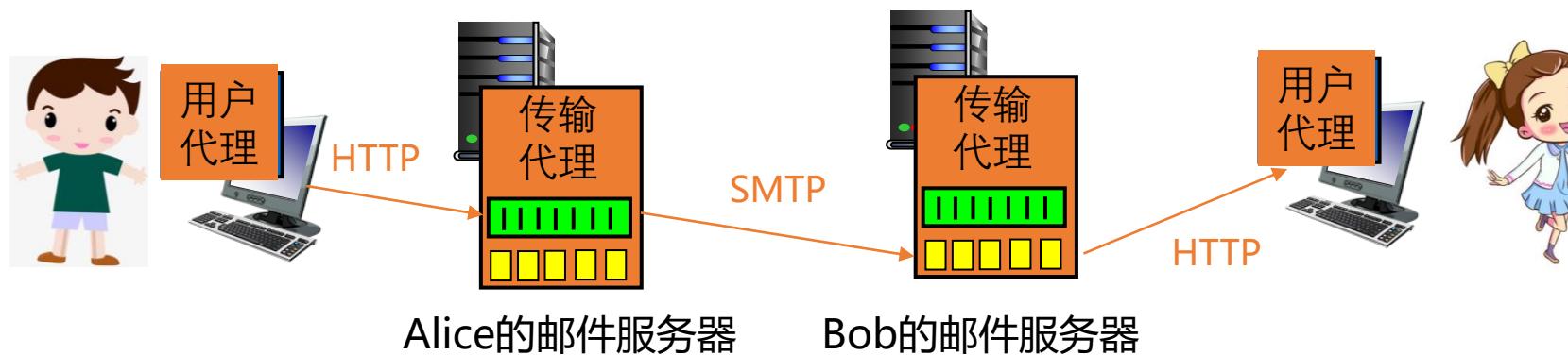




- IMAP允许用户使用不同计算机**同步和处理邮件**
  - IMAP服务器把**每个邮件与一个文件夹联系起来**，当邮件到达服务器时，它与收件人的INBOX文件夹相关联
  - 收件人能够把邮件移到一个新的、用户创建的**文件夹中**，阅读或删除邮件等
  - IMAP为用户提供了在远程文件夹中**查询邮件**的命令，按指定条件去查询**匹配的邮件**
  - 与POP3不同，IMAP服务器维护了IMAP会话的**用户状态信息**，例如，文件夹的名字以及哪些邮件与哪些文件夹相关联
- IMAP具有**允许用户代理获取邮件某些部分**的命令
  - 用户可以只读取一个邮件的首部，或只是一个多部分MIME邮件的一部分
  - 用户代理和邮件服务器之间使用低带宽连接时，用户可能不想取回邮箱中的所有邮件，可以**避免**可能包含如音频或视频片断的**大邮件**



- Webmail——基于Web的电子邮件
  - 提供电子邮件服务的IMAP和SMTP替代方案
  - 使用Web作为界面，用户代理就是普通的浏览器
  - 用户及其远程邮箱之间的通信通过HTTP进行





# 目 录

- 8.1 应用层概述
  - 8.2 域名系统
  - 8.3 电子邮件
  - 8.4 Web和HTTP
  - 8.5 流式音频和视频
- 1. WWW体系结构概述
  - 2. 静态Web（对象）
  - 3. 动态Web和Web应用
  - 4. HTTP协议
  - 5. Web缓存技术与Web代理
  - 6. Web安全与隐私

2000年互联网泡沫：  
比特流如何引爆全球



清华大学  
Tsinghua University



计算机网络教案社区



# WWW体系结构与协议

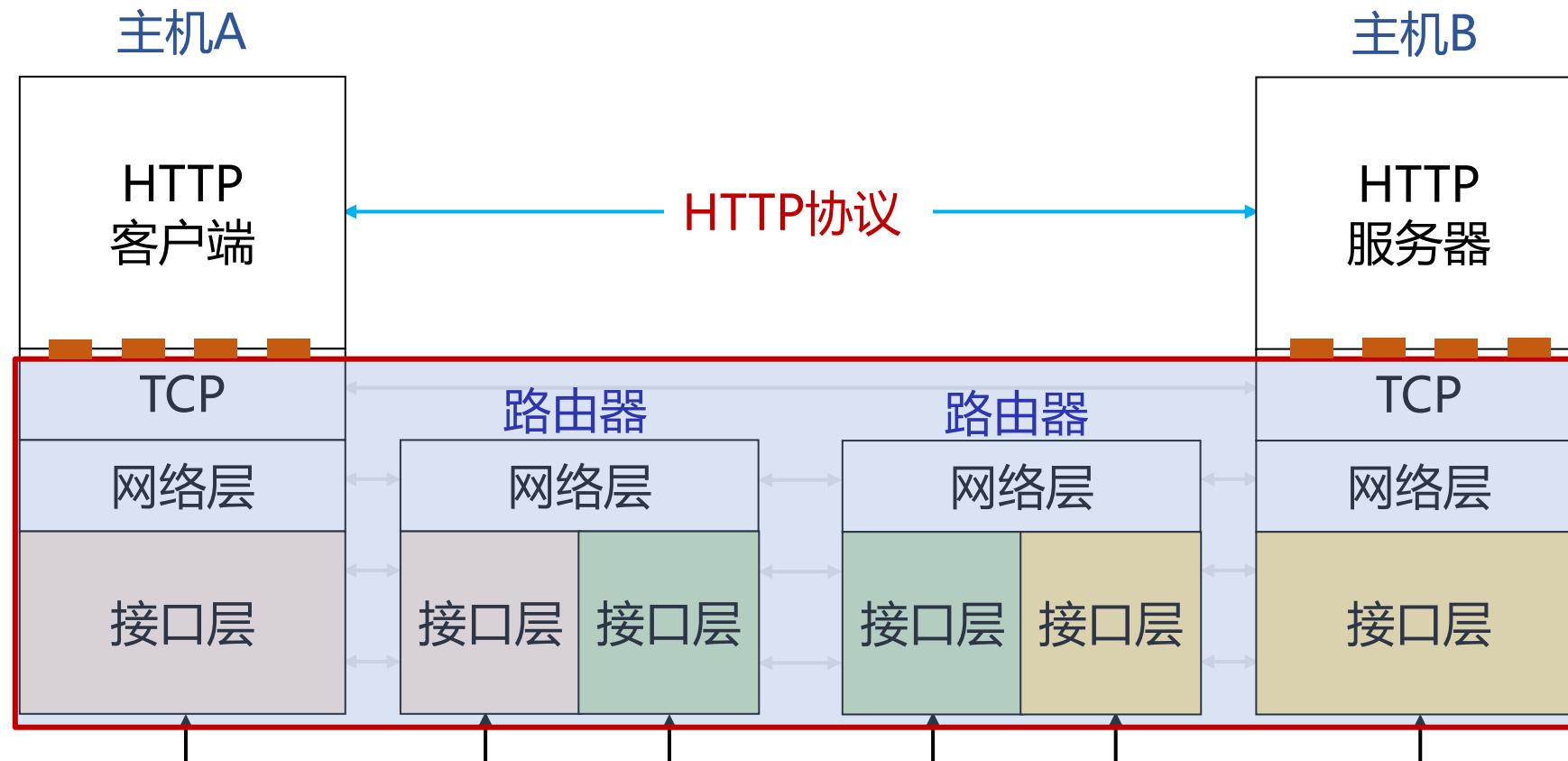


清华大学  
Tsinghua University



计算机网络教案社区

- WWW=World Wide Web=万维网
- HTTP服务器和客户端，以及它们之间执行的HTTP协议





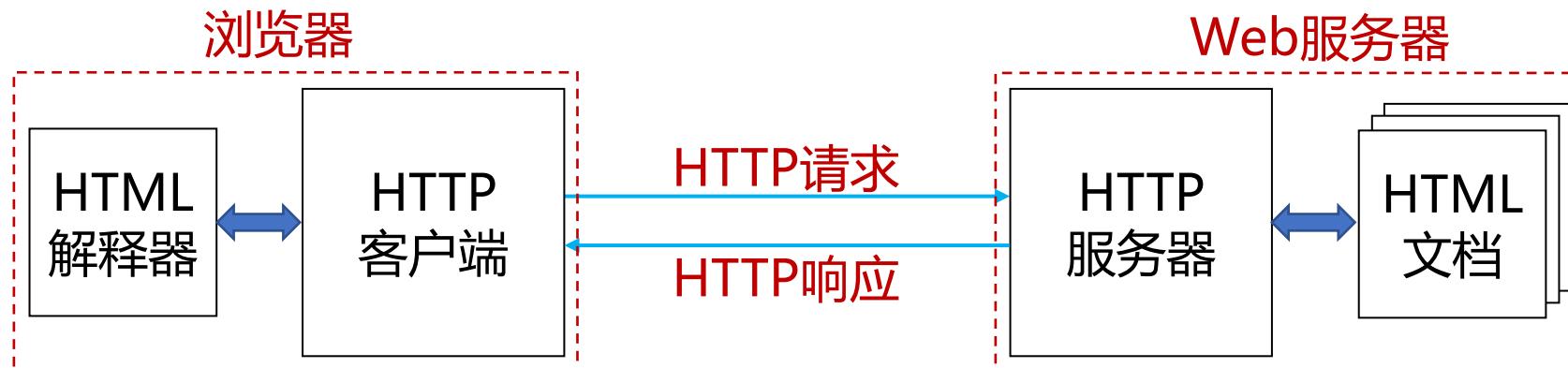
# WWW体系结构与协议

## ➤ 服务器

- Web页面 (HTML文档) : 包含到多种对象或链接
- Web对象 (包括: 静态对象和动态对象) : 文档、图像、视频、声音、脚本、文件等
- 对象用URL (统一资源定位符) 编址:  
**协议类型:// 主机名:端口 / 路径和文件名**

## ➤ 客户端

- 发出请求、接收响应、解释HTML文档并显示





# 统一资源定位器URLs

## 统一资源定位器URLs (Uniform Resource Locators)

- 例如：<http://www.phdcomics.com:8000/comics.php>

协议类型      主机名即服务器      端口      路径和文件名

名字	用途	实例
http	超文本HTML	<a href="http://www.xaut.edu.cn/xxgk/xxjj.htm">http://www.xaut.edu.cn/xxgk/xxjj.htm</a>
https	安全超文本	<a href="https://www.overleaf.com/">https://www.overleaf.com/</a>
ftp	FTP	<a href="ftp://ftp.xaut.edu.cn">ftp://ftp.xaut.edu.cn</a>
file	本地文件	<a href="file:///usr/xaut/prog.c">file:///usr/xaut/prog.c</a>
mailto	发送邮件	<a href="mailto:xautmail@xaut.edu.cn">mailto:xautmail@xaut.edu.cn</a>
rtsp	流媒体	<a href="rtsp://youtube.com/montypython.mpg">rtsp://youtube.com/montypython.mpg</a>
sip	多媒体呼叫	<a href="sip:eve@advesary.com">sip:eve@advesary.com</a>



## ➤ WEB网页对象

- 静态对象与静态网页
  - 文本，表格，图片，图像和视频等多媒体类型的信息（实现语言：标记语言，如：**超文本标记语言HTML**, XML, PHP等）
  - 字体、颜色和布局等风格类型的信息（实现语言：层叠样式表CSS）
- 动态对象与动态网页
  - 交互信息，比如，用户注册信息、登录信息等（实现：PHP/JSP等语言+MySQL等数据库）
- 链接
  - 超链接（HyperLinks）



# 静态Web与HTML

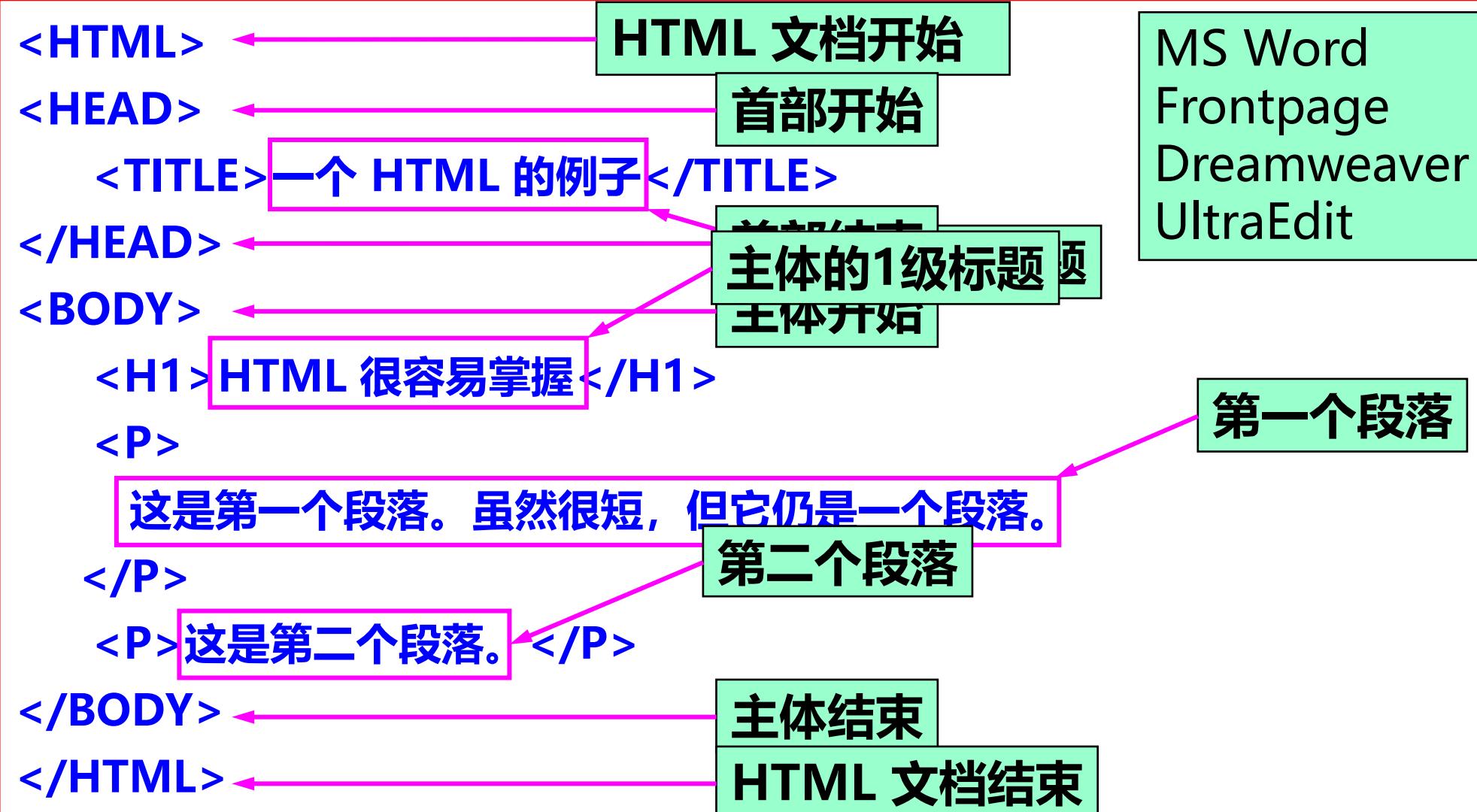
特点	HTML1.0	HTML2.0	HTML3.0	HTML4.0	HTML5.0
超链接、图像、列表	√	√	√	√	√
活动地图和图像		√	√	√	√
表单		√	√	√	√
方程式			√	√	√
工具条			√	√	√
表格			√	√	√
访问功能				√	√
对象嵌入				√	√
风格				√	√
脚本				√	√
视频和音频				√	√
内联矢量图					√
XML表示					√
后台线程					√
浏览器存储					√
画曲线					√

```
<html>
  <head>
    <title>计算机网络 </title>
  </head>
  <body>
    .....
  </body>
</html>
```



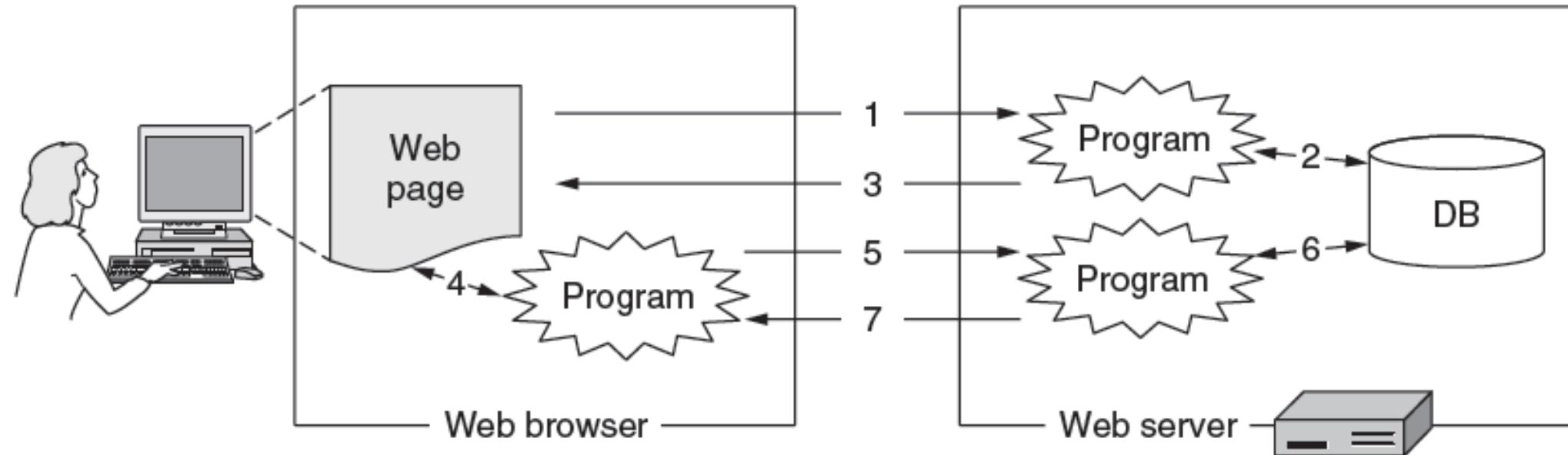


# HTML文档与静态网页





- 为什么需要动态Web?
- 动态Web页面
  - 通用网关接口CGI (\*.cgi)
  - 脚本语言+数据库技术 (\*.php, \*.asp, \*.aspx)

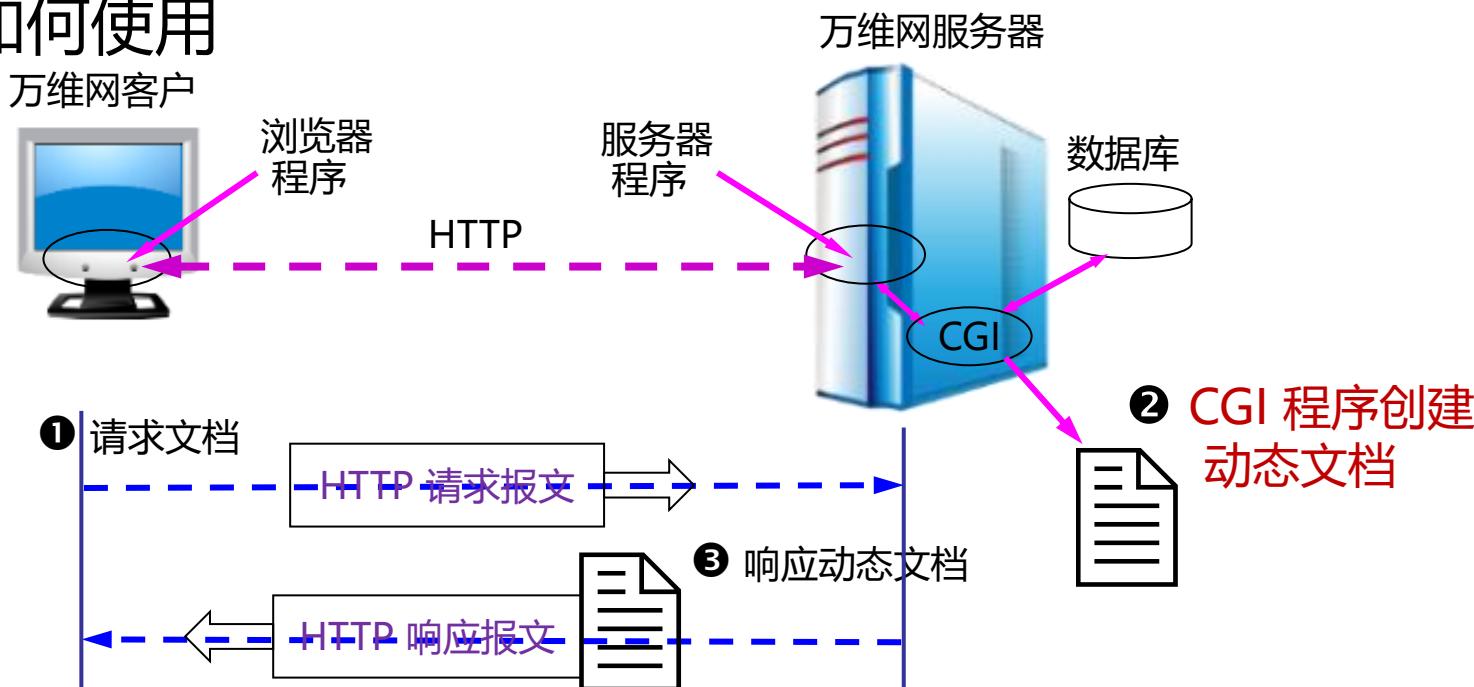




# 通用网关接口CGI

➤ CGI (Common Gateway Interface) 是一种标准

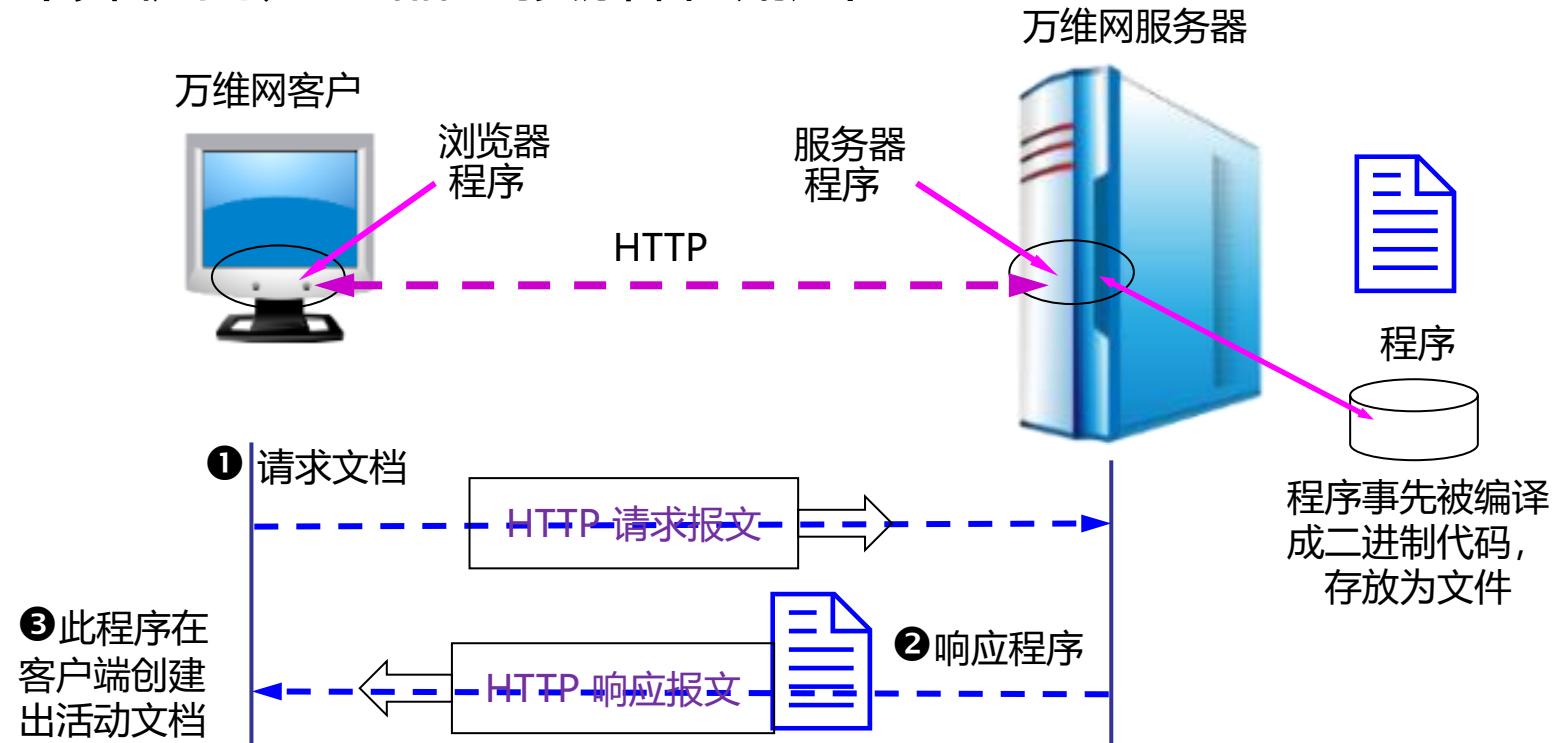
- 定义了动态文档应如何创建
- 输入数据应如何提供给应用程序
- 输出结果应如何使用





## “脚本” 程序

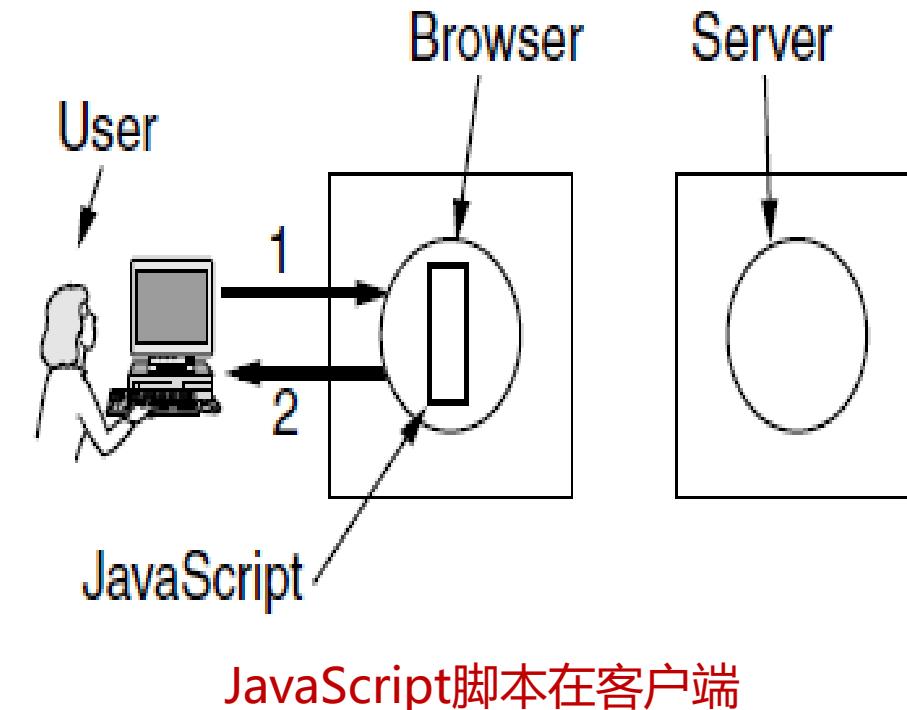
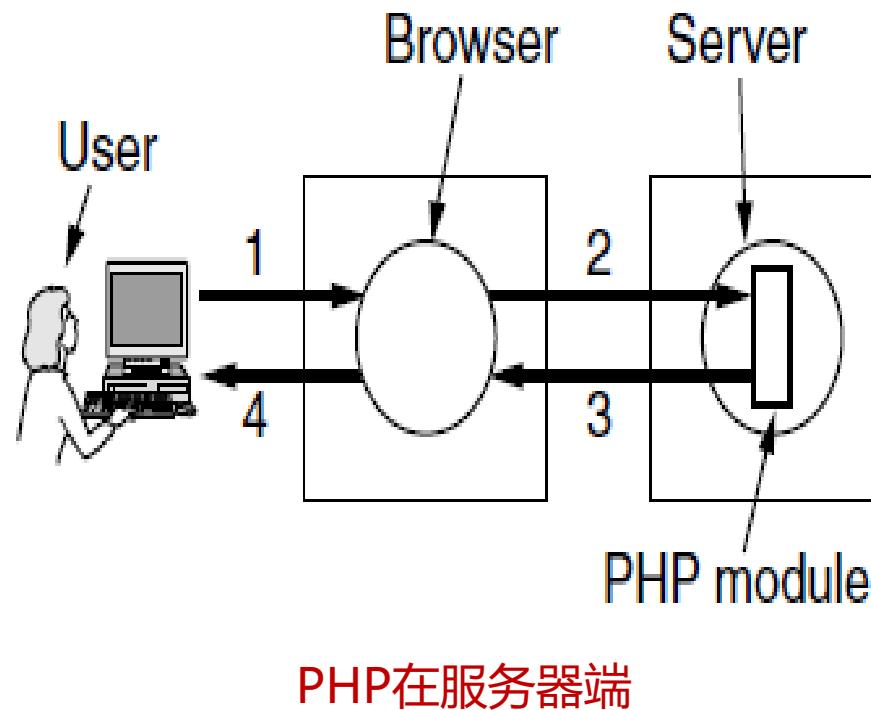
- 它被另一个程序（即解释程序如Web浏览器）解释执行
- 不是由计算机的处理器直接解释或执行





# 动态Web的执行

- 脚本程序的执行差异（在服务器端和客户端执行脚本的不同之处）
  - 最好的动态Web开发形式：显示与脚本程序相分离（显示+脚本+CSS3）





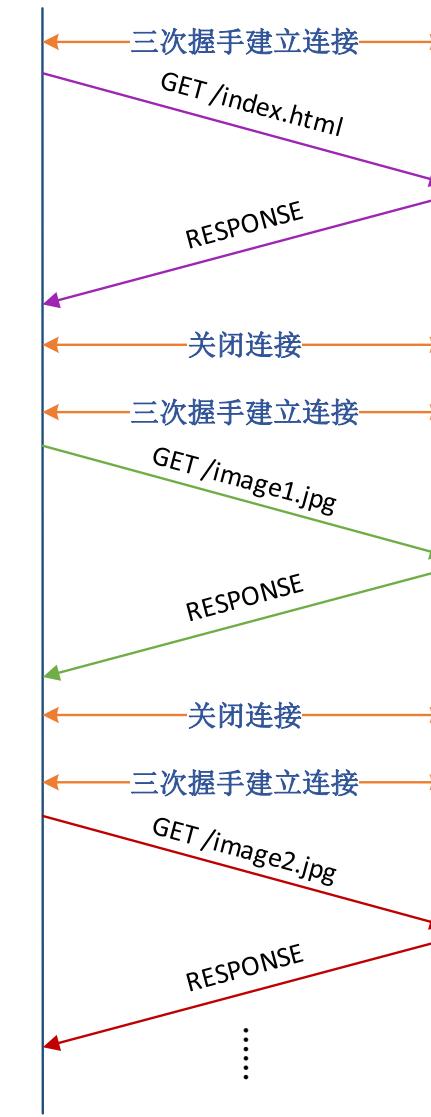
- 超文本传输协议HTTP ( HyperText Transfer Protocol)
  - 在传输层通常使用TCP协议，缺省使用TCP的80端口
- 如何简化HTTP服务器的处理
  - 无状态协议：效率低、但简单
  - 有状态协议：维护状态相对复杂，需要维护历史信息，在客户端或服务器出现故障时，需要保持状态的一致性等
  - HTTP为无状态协议，服务器端不保留之前请求的状态信息
- HTTP标准
  - HTTP/1.0: RFC 1945 (1996年)
  - HTTP/2: RFC 7540 (2015年) 、 RFC 8740 (2020年)



# HTTP1.0执行过程

## ➤ 假设用户输入URL

- <http://cuiyong.net/test.html>
- 如该页面包含2幅jpg图像
  - 需要**执行三次完整的连接过程**（即：连接的建立，数据传输，连接终止），**包含三次TCP过程（含三次HTTP过程）**
    - 三次握手连接建立
    - Request (Get /index.html)
    - Response
    - 关闭连接
    - .....
  - 如若含100副图像呢？





## ➤ 非持久连接

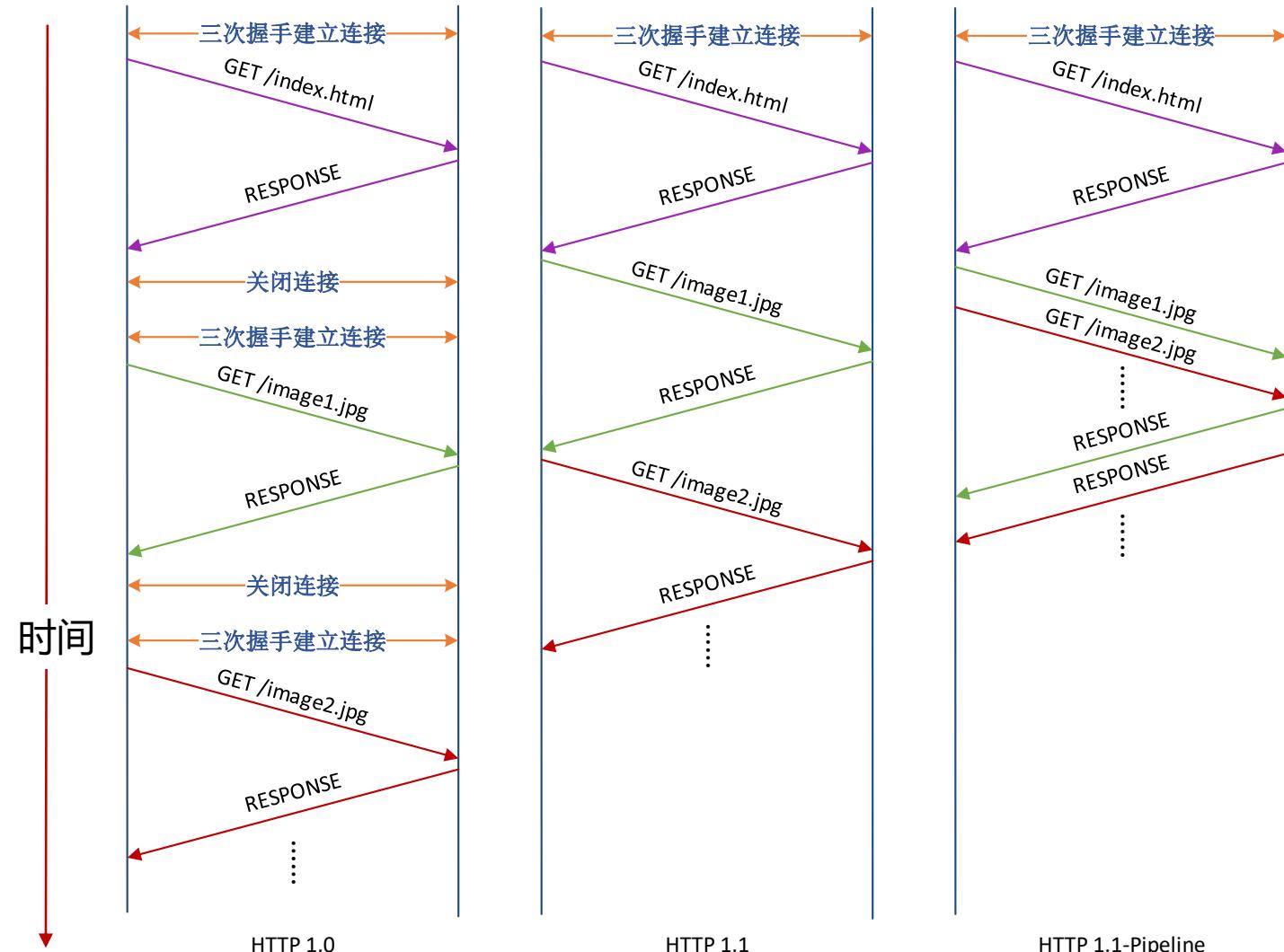
- HTTP/1.0缺省为非持久连接
  - 服务器接收请求、响应、关闭TCP连接
- 获取每个对象需要两阶段
  - 建立TCP连接
  - 对象请求和传输
- 每次连接需要经历TCP慢启动阶段

## ➤ 持久连接

- HTTP/1.1缺省为持久连接
  - 在相同的TCP连接上，服务器接收请求、响应；再接收请求、响应；响应后保持连接
- HTTP/1.1支持流水线机制
  - 需要按序响应
- 经历较少的慢启动过程，减少往返时间
- 降低响应时间



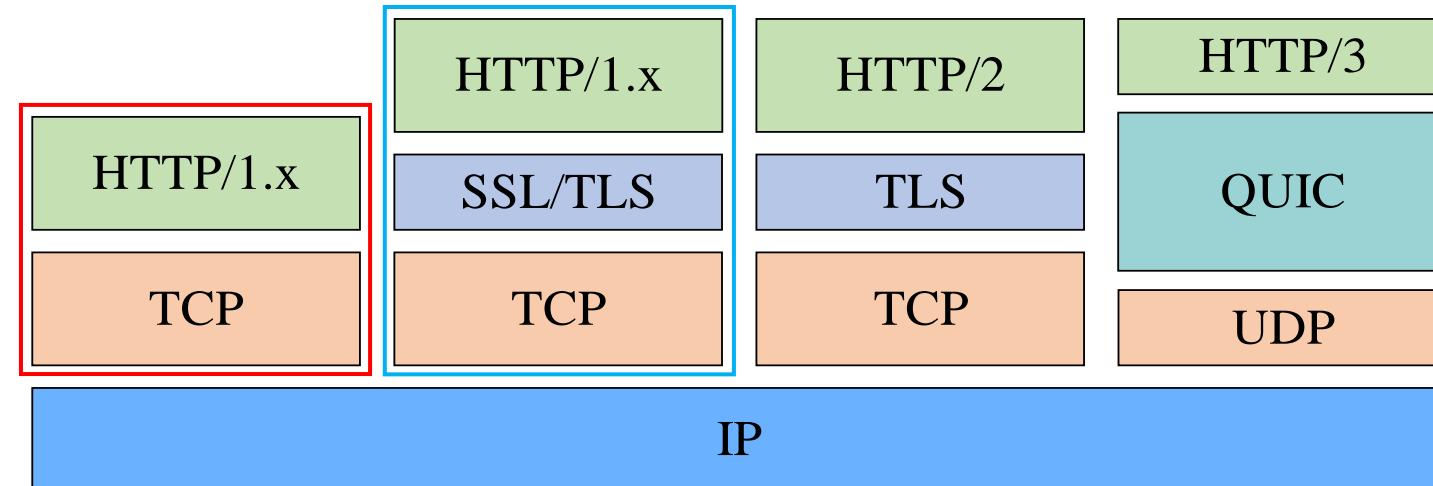
# HTTP 1.x比较





# HTTP发展现状

- HTTP/1.0 (1996)
  - 无状态, 非持久连接
- 与HTTP/1.1 (1999)
  - 支持长连接和流水线机制
  - 缓存策略优化、部分资源请求及断点续传
- HTTPS: HTTP+TLS (2008)
  - 增加SSL/TLS (TLS 1.2) 层, 在TCP之上提供安全机制
- HTTP/2.0 (2015、2020)
  - 目标: 提高带宽利用率、降低延迟
  - 增加二进制格式、TCP多路复用、头压缩、服务端推送等功能





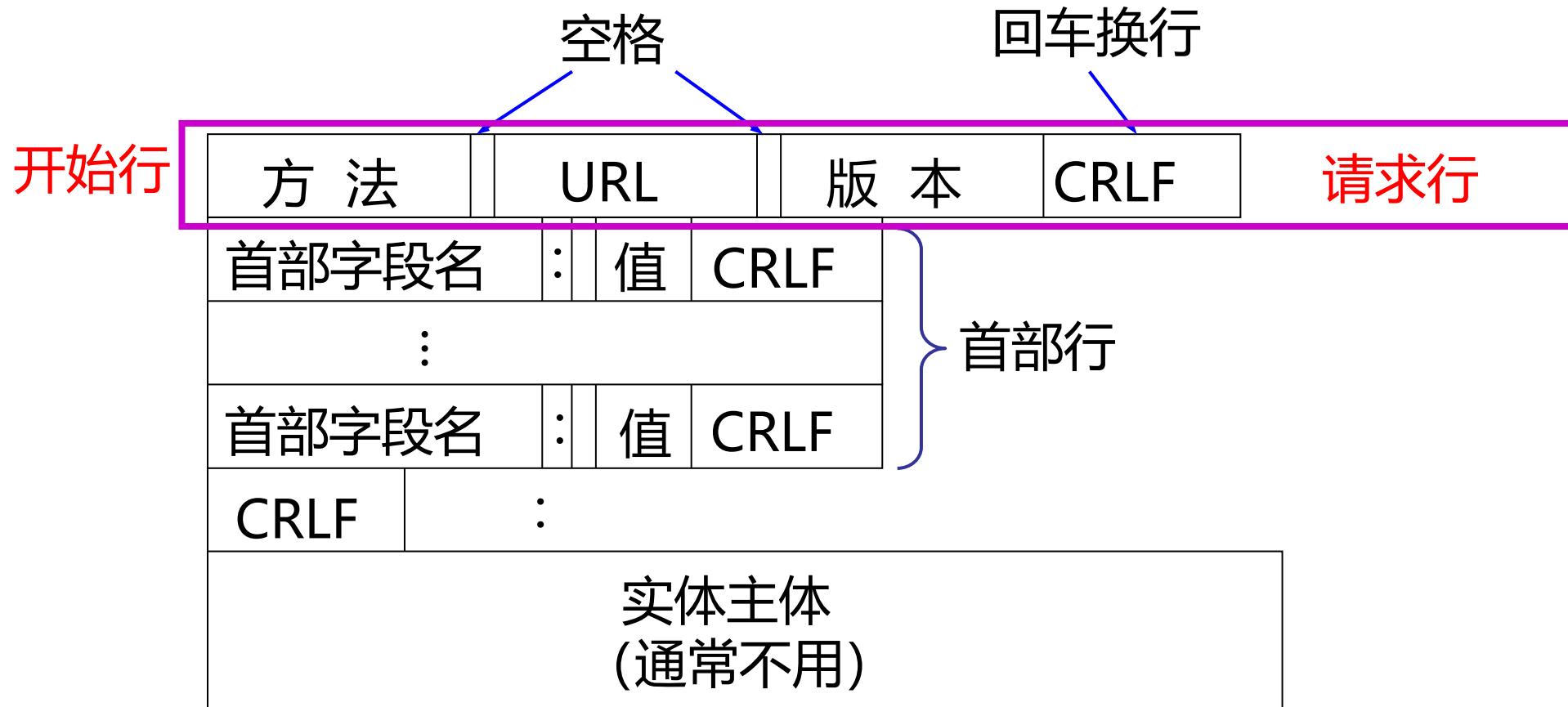
# HTTP报文结构：请求报文



➤ 报文由三个部分组成，即开始行、首部行和实体主体

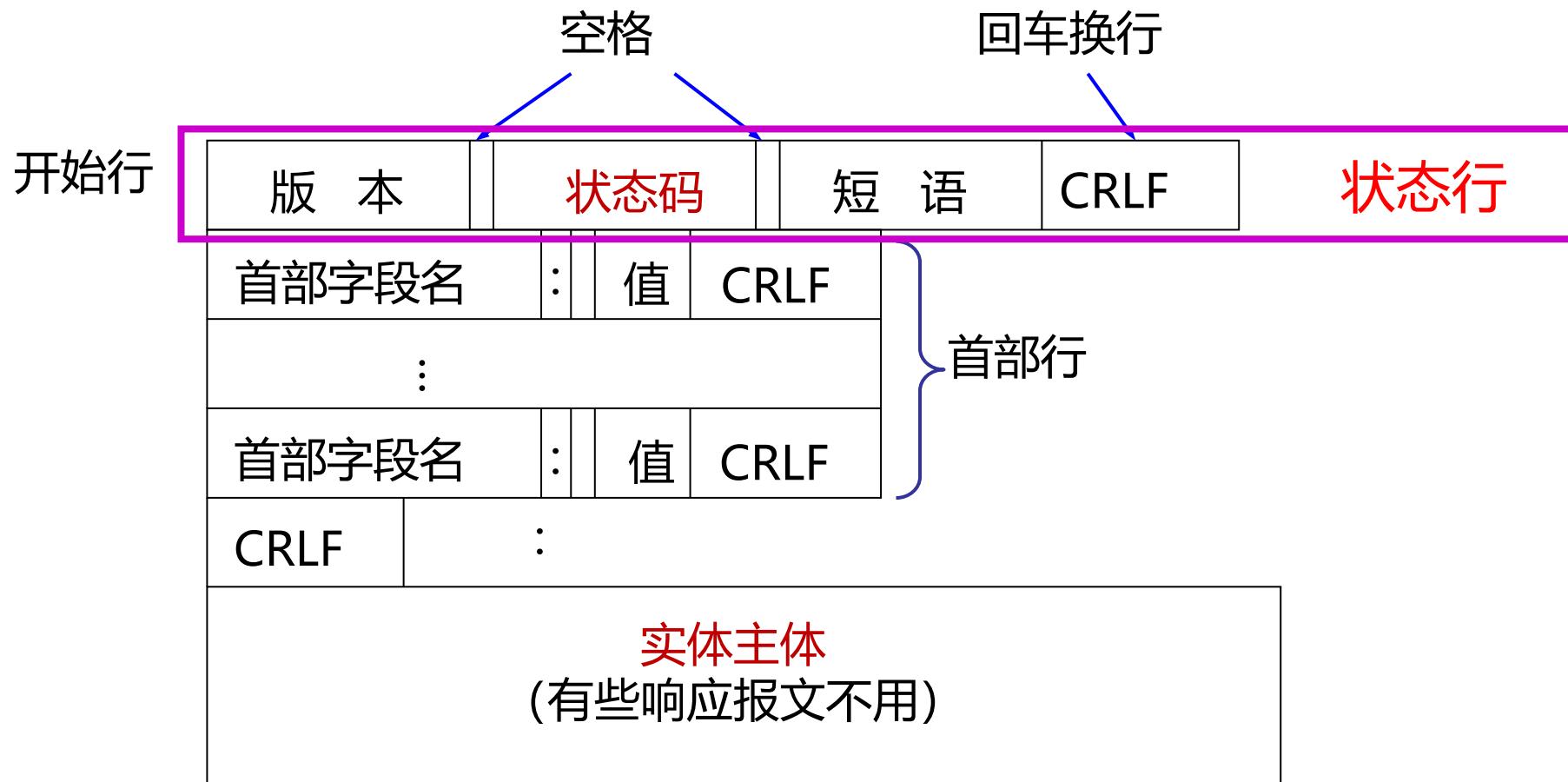
➤ 在请求报文中，开始行就是请求行

```
GET /~ross/index.html HTTP/1.0
```





# HTTP报文结构：响应报文



示例: HTTP/1.1 200 OK <data>



# HTTP响应报文：状态码

## ➤ 状态码都是三位数字

- **1xx** 表示通知信息的，如请求收到了或正在进行处理。
- **2xx** 表示成功，如接受或知道了。
- **3xx** 表示重定向，表示要完成请求还必须采取进一步的行动。
- **4xx** 表示客户的差错，如请求中有错误的语法或不能完成。
- **5xx** 表示服务器的差错，如服务器失效无法完成请求。

## ➤ 典型的状态码

**200 OK**

请求成功，被请求的对象包含在该响应的数据部分

**301 Moved Permanently**

请求的对象被移走，新的位置在响应中通过Location: 给出

**400 Bad Request**

服务器不能解释请求报文

**404 Not Found**

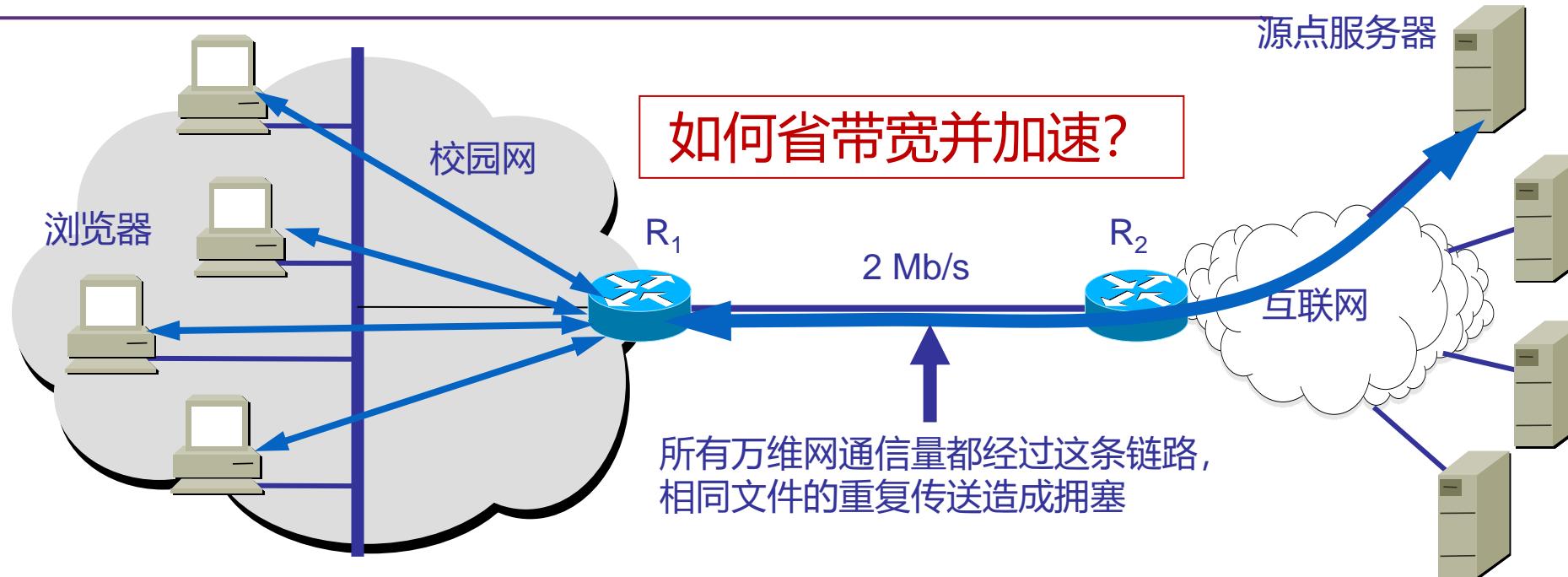
服务器中找不到请求的文档

**505 HTTP Version Not Supported**

服务器不支持相应的HTTP版本



# 共享HTTP的网络瓶颈



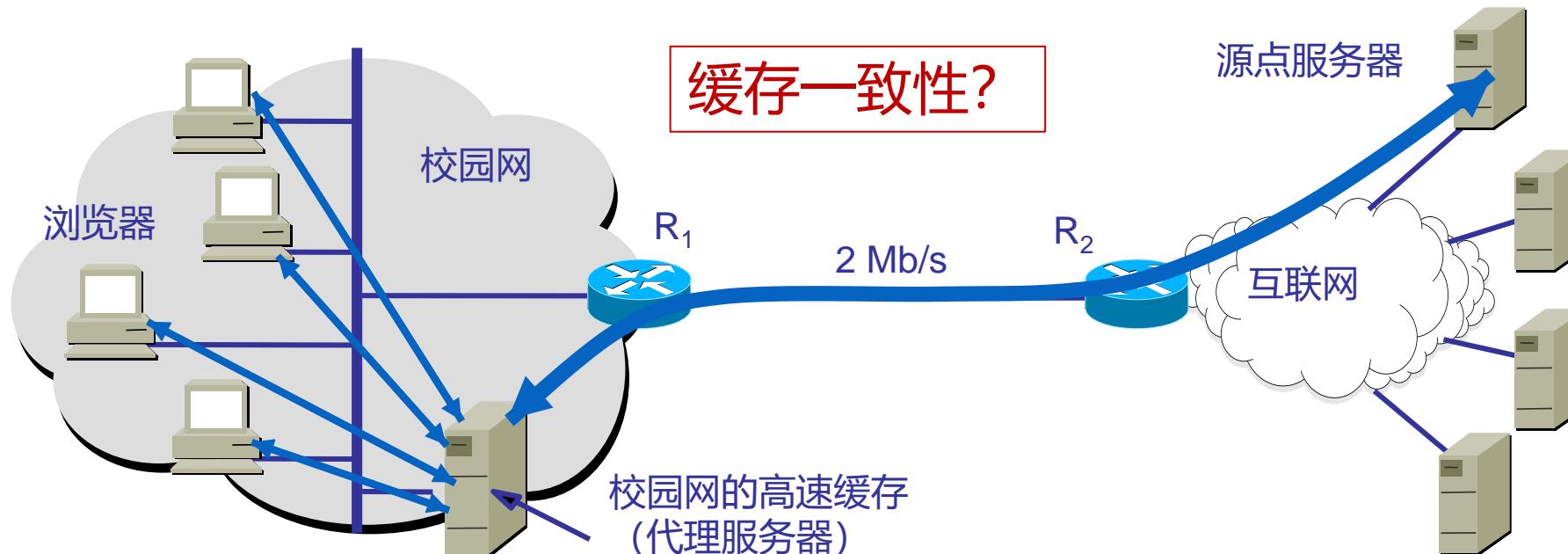
## ➤ 代理服务器(proxy server)

- 又称为万维网高速缓存(Web cache), 它代表浏览器发出 HTTP 请求
- 万维网高速缓存把最近的一些请求和响应暂存在本地磁盘中
- 当与暂时存放的新请求到达时, 万维网高速缓存就把暂存的响应发送出去, 而不需要按 URL 的地址再去互联网访问该资源



# 使用高速缓存的情况

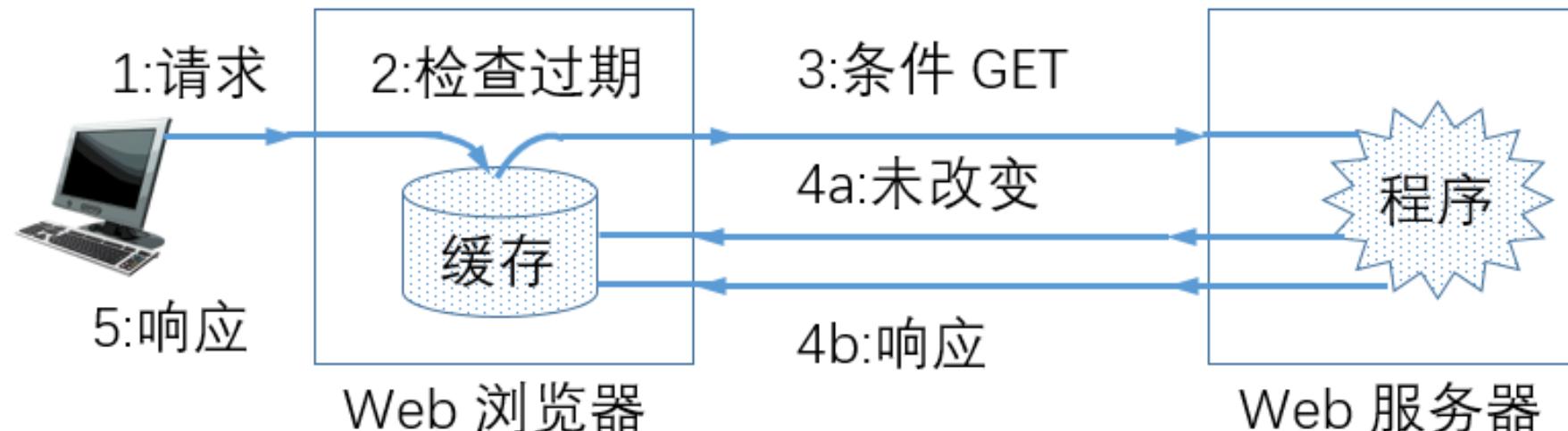
- (1) 浏览器访问服务器时，先与Proxy/高速缓存建立连接，并发出 HTTP 请求报文
- (2) 若高速缓存已经存放了所请求的对象，则将此对象放入 HTTP 响应报文中返回给浏览器
- (3) 否则，高速缓存就代表用户浏览器，与源点服务器建立 TCP 连接，并发送 HTTP 请求报文
- (4) 源点服务器将所请求的对象放在 HTTP 响应报文中返回给校园网的高速缓存
- (5) 高速缓存收到此对象后，先复制在其本地存储器中，再放在 HTTP 响应报文中返回给用户





## ➤ 浏览器缓存

- 目标：再次访问缓存在浏览器主机中的Web页副本，不必从原始服务器读取
- **缓存一致性**：怎么保证Web页副本与原始服务器是一致的？





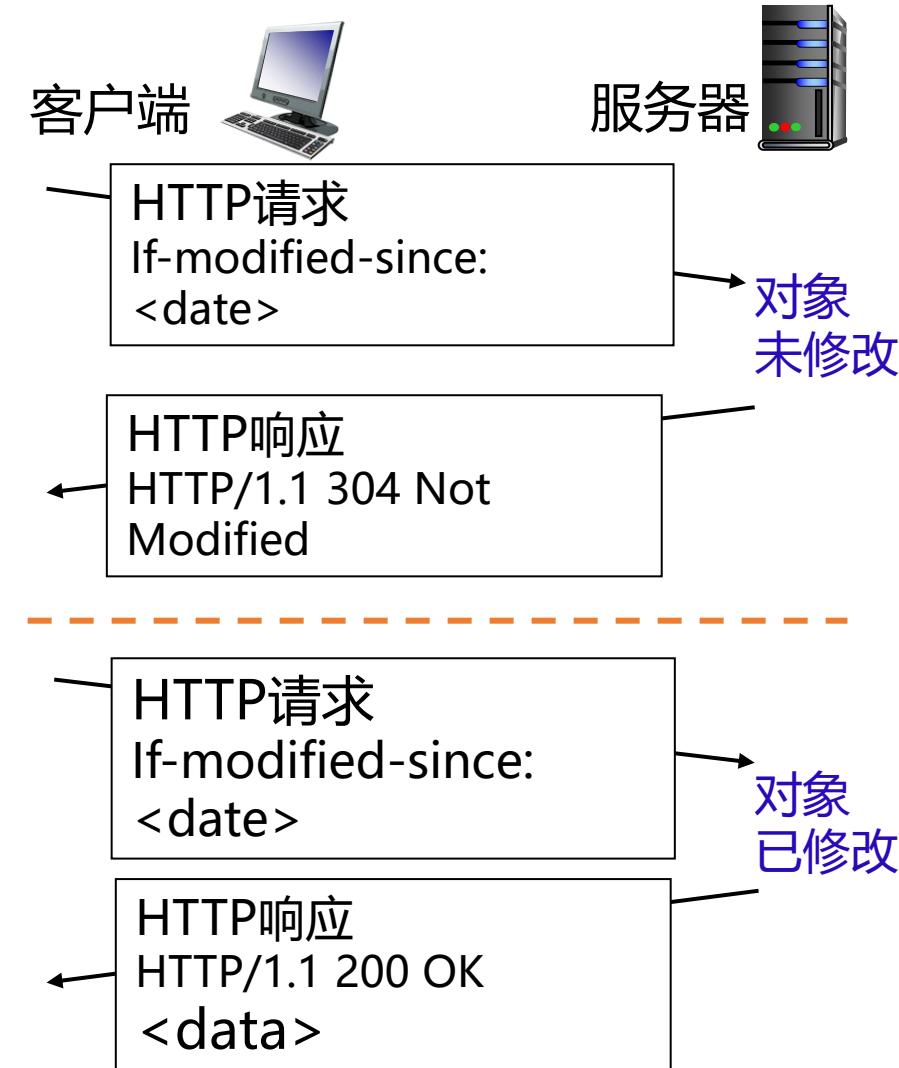
## ➤ 询问式策略

- 通过特殊的关键字头询问原始服务器，Web副本对应的原始Web页是否已更新
- 客户端**：在发送的HTTP请求中指定缓存的时间，请求头包含**If-modified-since: <date>**
- 服务器**：如果缓存的对象是最新的，在响应时无需包含该对象，响应头包含**HTTP/1.1 304 Not Modified**

否则服务器响应**HTTP/1.1 200 OK <data>**

## ➤ 原始服务器明确指令限制缓存某些Web页

- 服务器返回Web页时，带一个no-cache禁止缓存
- 需要授权访问的Web页也限制缓存





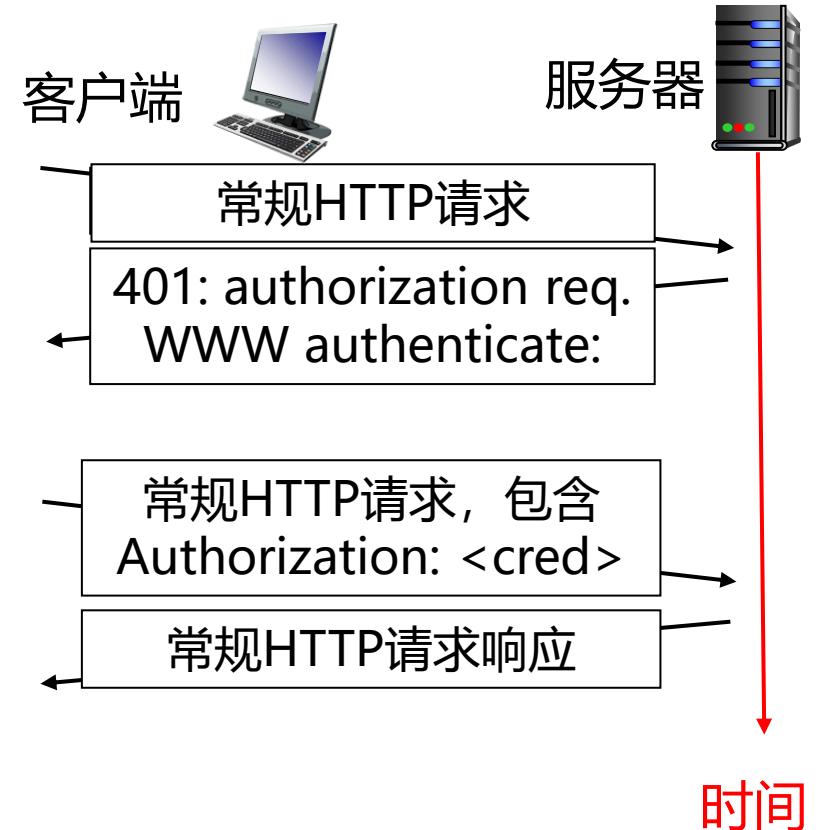
## ➤ Web访问限制与安全

- 并非所有Web页都会向公众开放
- Web服务器可以限定客户端访问的**IP地址空间**，比如限制只向公司内部员工开放
- Apache服务器将设置限制访问规则的文件.htaccess放置在被限制访问的页面所在的目录，客户端访问时进行规则匹配
- **认证方法**：在浏览器客户端的HTTP请求中给出“**用户名-密码**”，服务器进行HTTP验证



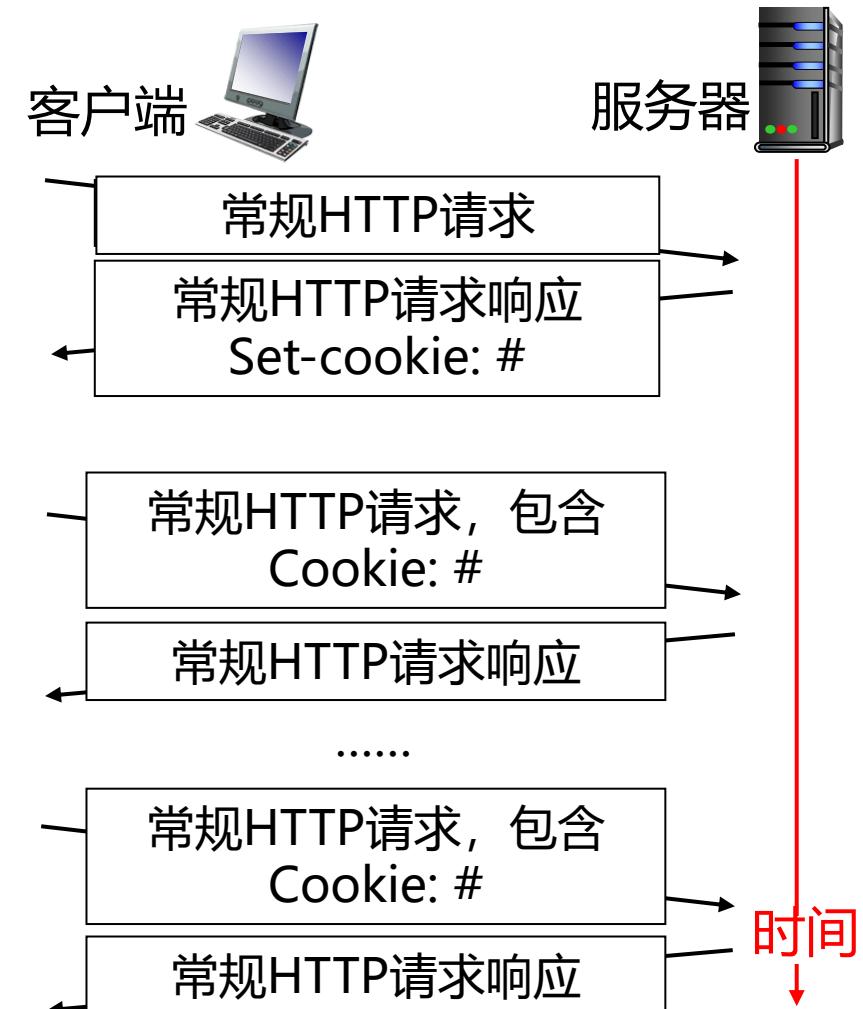
## ➤ Web访问安全

- 无状态：客户端需要在每个请求中携带认证信息
- 认证方法：通常在HTTP请求中使用“用户名-密码”
- 每个请求头中包含关键字**authorization**：
- 如果请求头中无**authorization:**，则服务器拒绝访问，并在响应头中包含**WWW authenticate:**





- 无状态的HTTP如何跟踪用户状态?
  - 如何控制注册或付费用户? 如何跟踪购物车里的内容? 如何提供个性化的服务?
  - 服务器用**cookies**保持用户状态
  - HTTP在响应的首部行里使用一个关键字头**set-cookie**, 选择的**cookie号具有唯一性**
  - 后继的HTTP请求中使用服务器响应分配的**cookie**,
  - **Cookie文件**保存在用户主机中, 内容是服务器返回的一些附加信息, 由用户主机的浏览器管理
  - Web服务器建立后端数据库, 记录用户信息, **cookie作为关键字**





# Web安全与隐私：Cookie



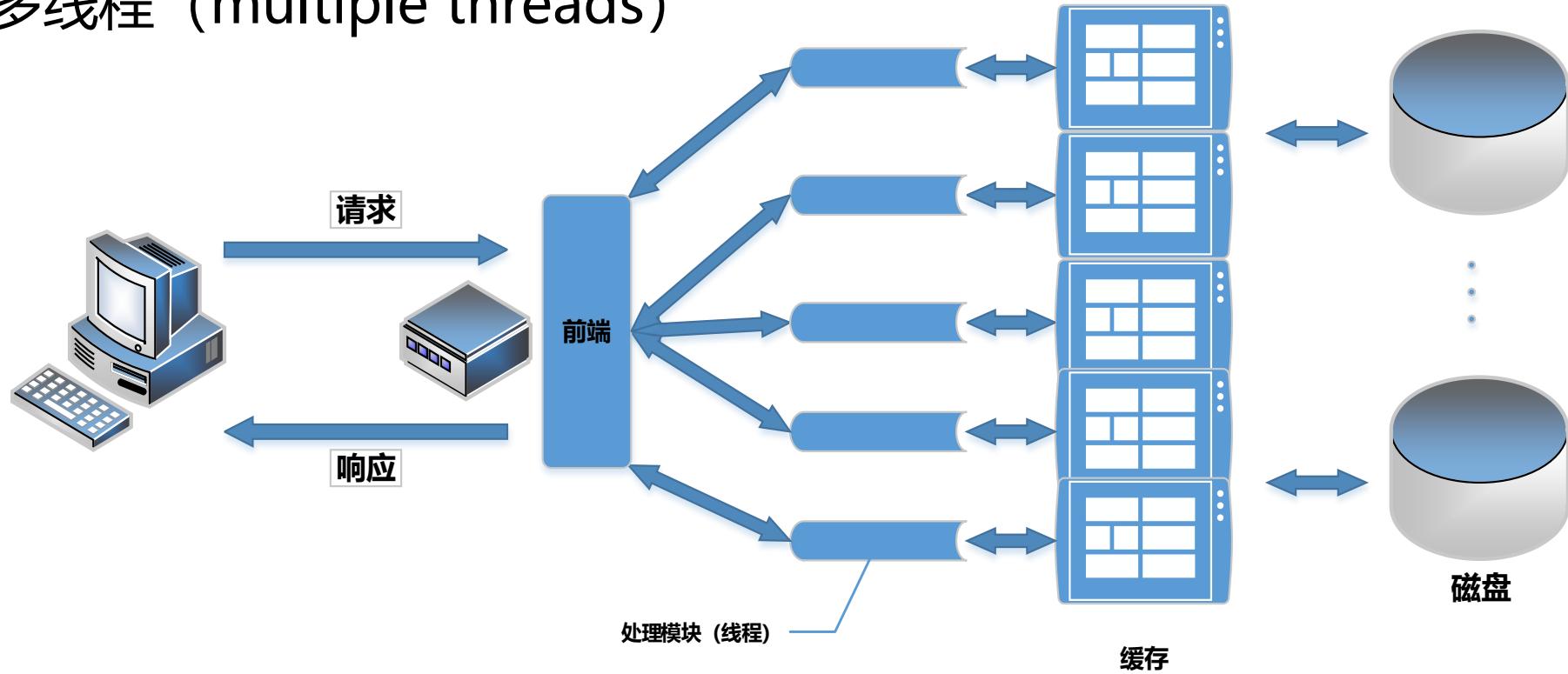


- Cookie技术是把**双刃剑**：能分析用户喜好，向用户进行个性化推荐
  - 用Cookie在某网站标识用户信息，查找用户以前浏览网站记录
  - 用Cookie记录用户购物清单
  - 用Cookie可以保存4K内容，跟踪用户浏览网站的喜好
  - 用Cookie跨站点跟踪用户点击广告
- Cookie技术是把**双刃剑**：能跟踪用户网络浏览痕迹，泄露用户隐私
  - Cookie跟踪用户以前浏览过哪些网站，跟踪用户频繁浏览哪类网站
  - Cookie收集用户信息，用户网络交互时关注的关键词
- Cookie嵌入间谍程序？
  - 这是误区，Cookie只保存文本串，没有可执行程序
  - 用户可以设置浏览器限制使用Cookie



# WWW性能提升方法

- 前端 (front end) : BFE, GFE
- 缓存 (caching) , 如代理技术
- 多线程 (multiple threads)





# 本节内容

8.1 应用层概述

8.2 域名系统

8.3 电子邮件

8.4 Web和HTTP

8.5 流式音频和视频

1. 流媒体概述
2. 数字音视频与编码
3. 流式存储媒体
4. 直播与实时音视频
5. 流媒体动态自适应传输

学习资料.avi 该怎么传输播放  
短视频和直播、会议什么区别?  
视频为什么会卡住



清华大学  
Tsinghua University



计算机网络教案社区



# 流媒体概述

## 常见的流媒体服务

- **点播**: 提前录制好, 边下载边播放 (起始时延 $<10\text{s}$ ; 类VCR操作 (例如拖动进度条)  $<1\sim2\text{s}$ )
- **直播**: 边录制边上传, 边下载边播放 (大规模直播往往有数秒的时延)
- **实时交互** : 双方或多方实时交互式通信 (时延 $<400\text{ms}$ 可接受, VR则需要 $<25\text{ms}$ )



视频点播



视频直播



腾讯会议  
Tencent Meeting



实时交互



## ➤ 流媒体概念

- 连续媒体（音视频）经压缩编码、数据打包后，经过网络发送给接收方
- 接收方对数据进行重组、解码和播放

## ➤ 流媒体的特性

- 端到端时延约束
- **时序性约束**：流媒体数据必须按照一定的顺序连续播放
- 具有一定程度的**容错性**：丢失部分数据包也可完成基本功能

## ➤ 流媒体面临的挑战

- 约束条件：网络特性（带宽有限、动态变化、延迟与抖动、丢失、异构性）
- 目标：流媒体服务质量要素（画质、启动延迟、平滑、交互性）
- 如何在“尽力服务”的网络传输条件下获得良好的视频质量？



# 流媒体概述



期望：高清、低延迟、不卡顿？可是网络不稳定……  
场景：点播、直播、会议、连麦、云游戏、VR、……

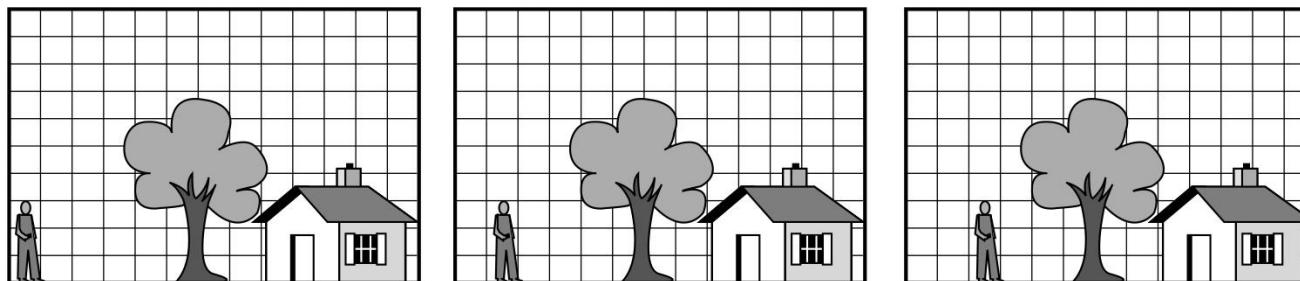


## ➤ MPEG视频压缩

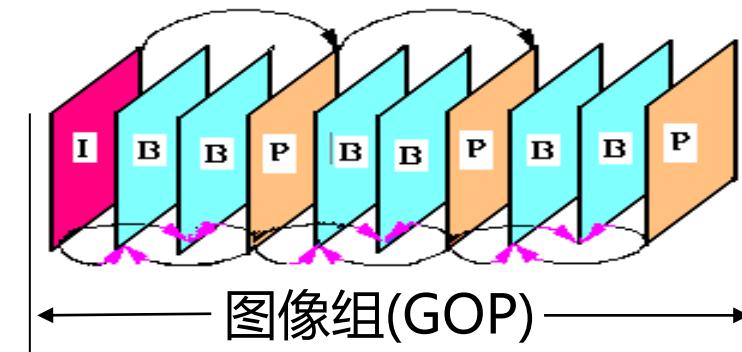
- 摄像机固定不变，演员慢慢走来走去：从前一帧中减去当前帧，对两帧之差运行JPEG算法
- 摄像机在推拉移动：需要某种方法来补偿这种运动，这正是MPEG擅长的

## ➤ MPEG的输出包括3类帧

- **帧内编码帧 (I帧)**：包含了压缩的静止图片（帧内编码，用JPEG来压缩静止图像）
- **预测帧 (P帧)**：是与前一帧的逐块差值（帧间编码，消除跨帧的冗余度）
- **双向帧 (B帧)**：是与前一帧和后一帧的逐块差值（帧间编码）



视频连续帧之间的差异往往较小，可将差值进行编码



一组连续的IPB画面  
没有I帧，P帧和B帧就无法解码



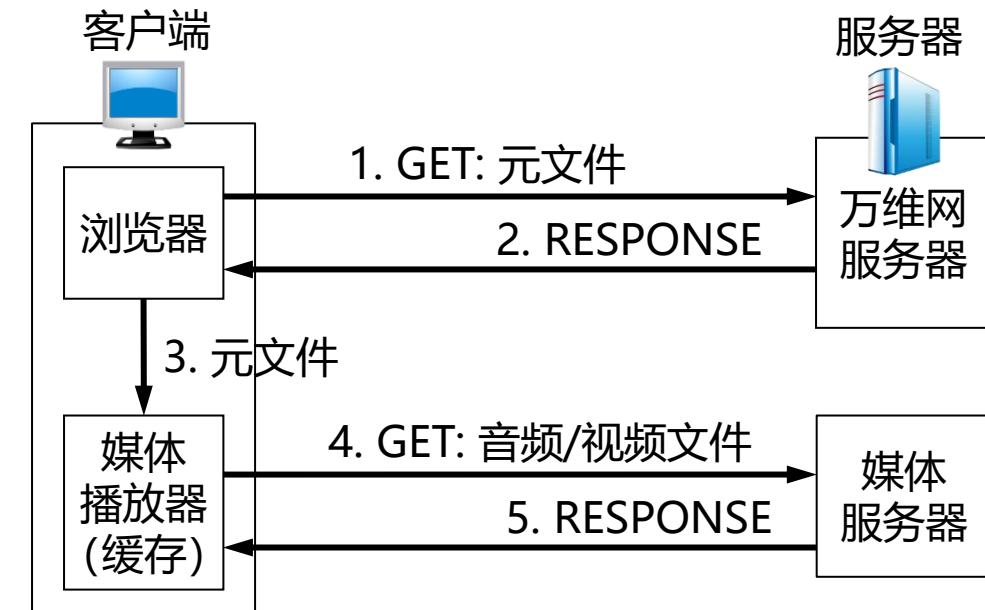
- **I帧**必须周期性地出现在媒体流中
  - 流媒体直播中，后加入的观众需要收到**I帧**才能成功解码
  - 如果任意一帧发生了接收错误，则后续**非I帧**无法解码（由于B/P帧依赖损坏的帧）
  - 快进或者回退到某位置时，解码器需要从该位置前面的**I帧**开始计算
- 常见帧速率：24帧/秒、30帧/秒、60帧/秒（VR视频）
- 常见分辨率：标清SD 480p/576p、高清HD 720p、全高清FHD 1080p、4K超高清 2160p、8K超高清 4320p





## ➤ 浏览器从服务器下载流媒体文件

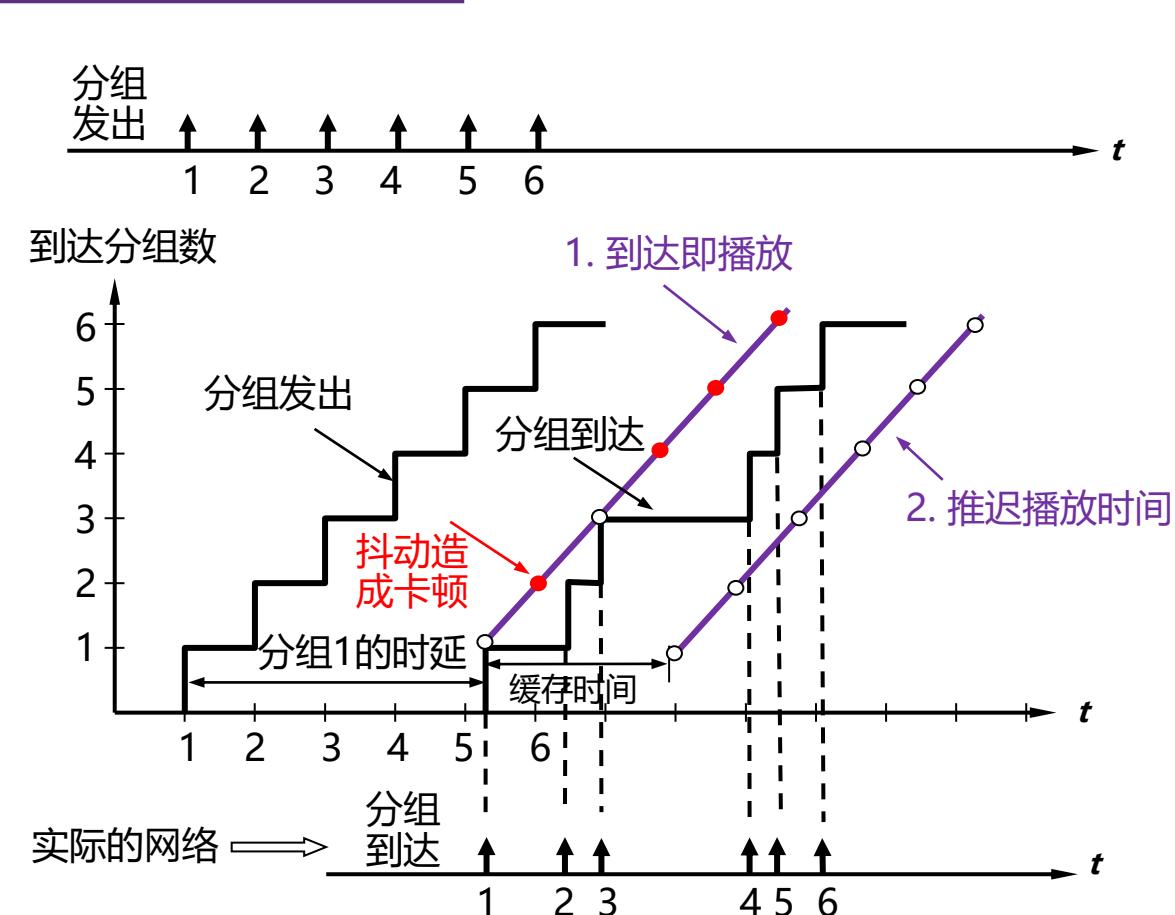
- 浏览器用户使用 HTTP 的 GET 报文接入到万维网服务器；这个超链指向一个元文件（有音/视频文件的统一资源定位符 URL）
- 万维网服务器把该元文件装入 HTTP 响应报文的主体，发回给浏览器
- 浏览器调用媒体播放器，把提取出的元文件传输给媒体播放器
- 媒体播放器使用元文件中的 URL，向媒体服务器发送 HTTP 请求报文，要求下载音/视频文件（如对应的某个GOP）
- 媒体服务器发送 HTTP 响应报文，把音/视频文件发送给媒体播放器；媒体播放器边下载边解压缩边播放（通过时间戳同步音频流和视频流）



流式存储媒体的典型下载过程



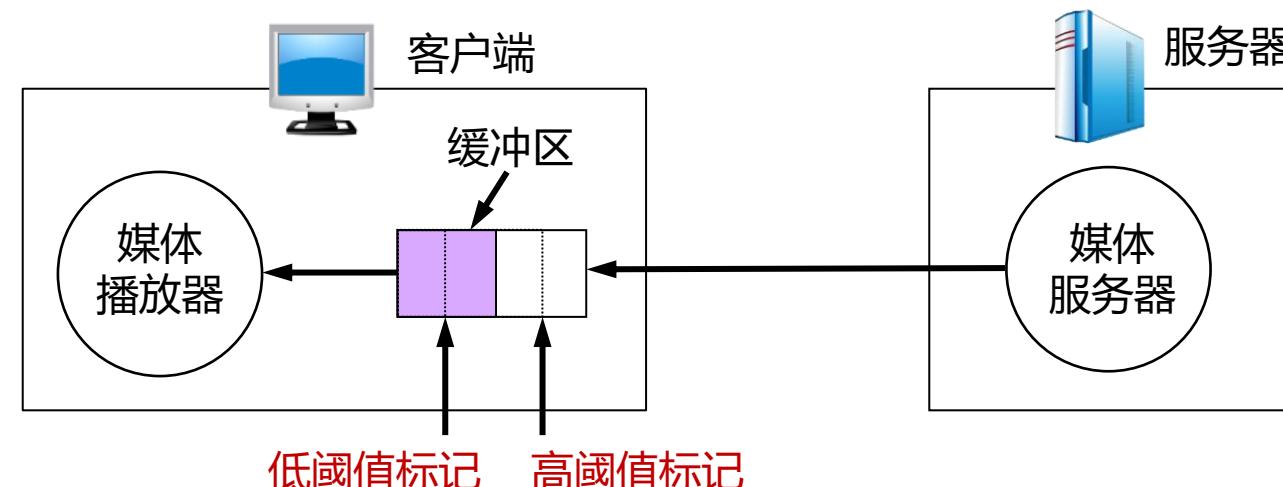
- 发送端以恒定速率产生数据分组
- 网络传输后的结果
  - 由于**网络传输的抖动特性**，分组到达接收端时变成了非恒定速率
  - 此时如果到达时就随即播放，则会出现卡顿
  - (分组1、3到达接收端可播放，分组2、4、5、6未到达，出现卡顿)
- 如何应对网络传输的抖动特性
  - 在接收端经过缓存后，再以恒定速率播放(推迟播放时间)
  - 能够在一定程度上消除了时延的抖动
  - 但付出的代价是增加了时延





## ➤ 客户端缓冲区

- 客户端播放的是本地缓冲区的内容，而不是立即播放来自网络的实时内容
- 缓冲区内容小于低阈值标记：数据即将播完，容易出现卡顿；需要加速传输
- 缓冲区内容大于高阈值标记：增大播放时延，占用存储空间；可以减慢传输
- 需要的决策：需要多大缓存，服务器以多快速率发送，才能在不稳定的网络中，尽量满足用户期望：高清、低延迟、不卡顿
- 上述决策需要特定网络协议支持



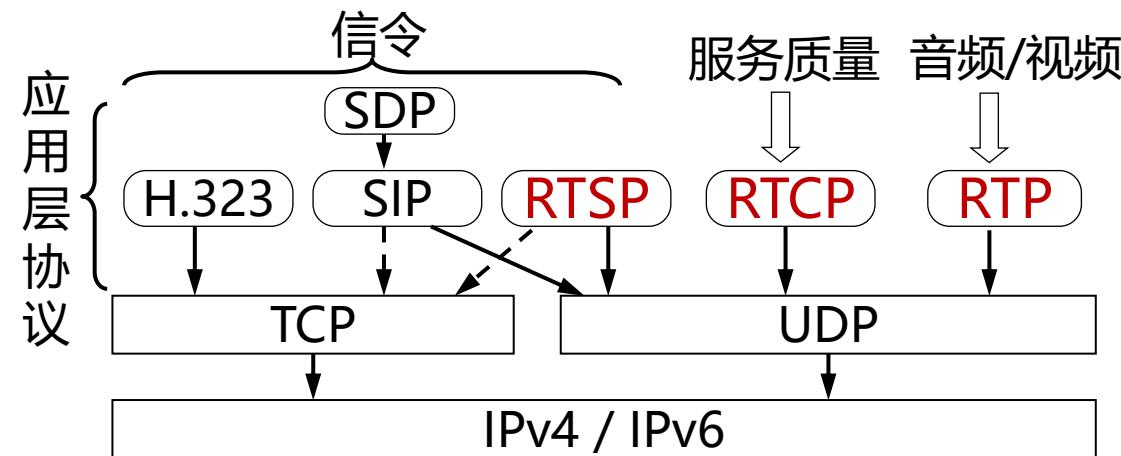


- 实时音频/视频所需要的几种应用协议
  - 一种是信令协议，对建立的连接起控制作用，如**RTSP**
  - 一种是数据分组传送协议，使音/视频能够以时延敏感属性传送，如**RTP/RTCP**

## ➤ 使用TCP，还是UDP？

- UDP不可靠但效率高，更适合实时类应用
- UDP需要自行实现流控算法，增加了成本和复杂性
- UDP传输音视频可能会被路由器丢弃或防火墙阻拦，而TCP可以畅通无阻

实际流媒体系统往往先尝试  
UDP，如果失败则转为TCP





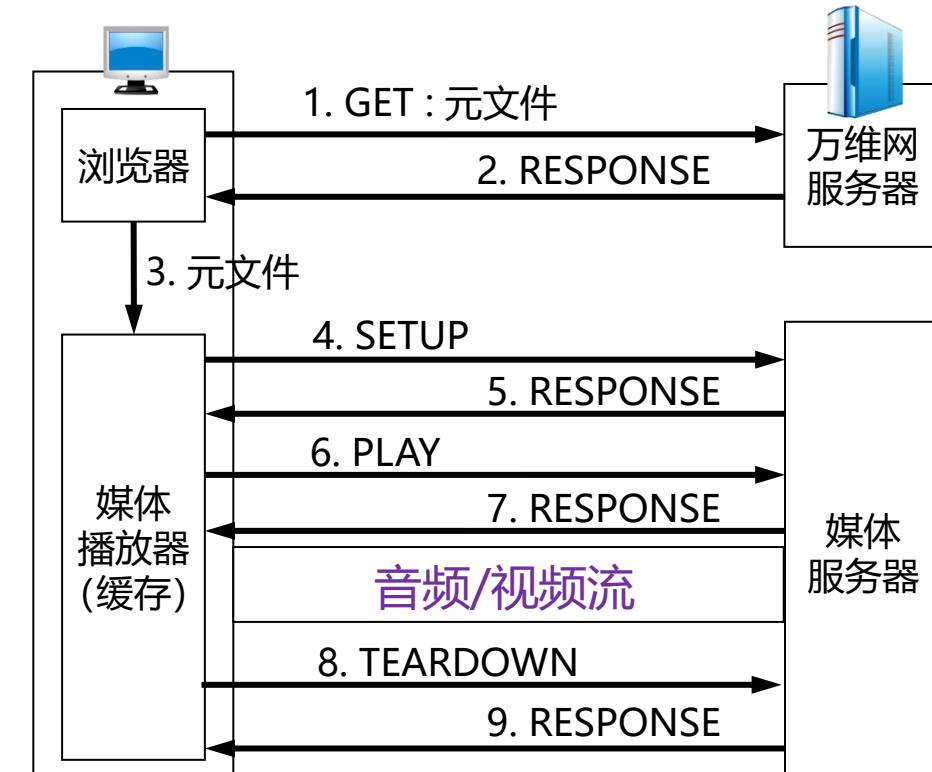
## ➤ 实时流式协议RTSP (Real-Time Streaming Protocol)

- RTSP本身并不传送数据，是一个多媒体播放控制协议
- RTSP对用户下载的实时数据的播放情况进行控制，如：暂停/继续、后退、前进等，又称为“互联网录像机遥控协议”
- RTSP是有状态的协议，它记录用户所处于的状态（初始化状态、播放状态或暂停状态）
- RTSP控制分组既可在TCP上传送，也可在UDP上传送
- RTSP没有定义音频/视频的压缩方案，也没有规定音频/视频在网络中传送时应如何封装在分组中
- RTSP协议本身没有规定音频/视频流在媒体播放器中应如何缓存，由协议的具体实现（和算法）负责



## ➤ 使用 RTSP 的媒体服务器的工作过程

- 浏览器向万维网服务器请求音/视频文件
- 万维网服务器从浏览器发送携带有元文件的响应
- 浏览器把收到的元文件传输给媒体播放器
- RTSP 客户与媒体服务器的 RTSP 服务器建立连接
- RTSP 服务器发送响应 RESPONSE 报文
- RTSP 客户发送 PLAY 报文，开始下载音/视频文件的**特定位置**
- RTSP 服务器发送响应 RESPONSE 报文
- 开始传输**音视频数据**（使用**RTP协议**）
- RTSP 客户发送 TEARDOWN 报文断开连接
- RTSP 服务器发送响应 RESPONSE 报文



RTSP 协议的工作过程



## ➤ 实时传输协议 RTP (Real-time Transport Protocol)

- RTP 为实时应用提供端到端的**数据传输**, 但不提供任何服务质量的保证
- RTP 是一个协议框架, 只包含了实时应用的一些共同的功能
- RTP 不对多媒体数据块做任何处理, 而只是向应用层提供一些附加的信息, 让应用层知道应当如何进行处理

## ➤ 实时传输控制协议 RTCP (RTP Control Protocol)

- **RTCP 是与 RTP 配合使用的控制协议**
- RTCP 的主要功能: 服务质量的监视与反馈、媒体间的同步、播组中成员的标识
- RTCP 分组也使用 UDP 传送, 但 RTCP 并不对声音或视像分组进行封装
- 可将多个 RTCP 分组封装在一个 UDP 用户数据报中
- RTCP 分组周期性地在网上传送, 它带有发送端和接收端对服务质量的统计信息报告



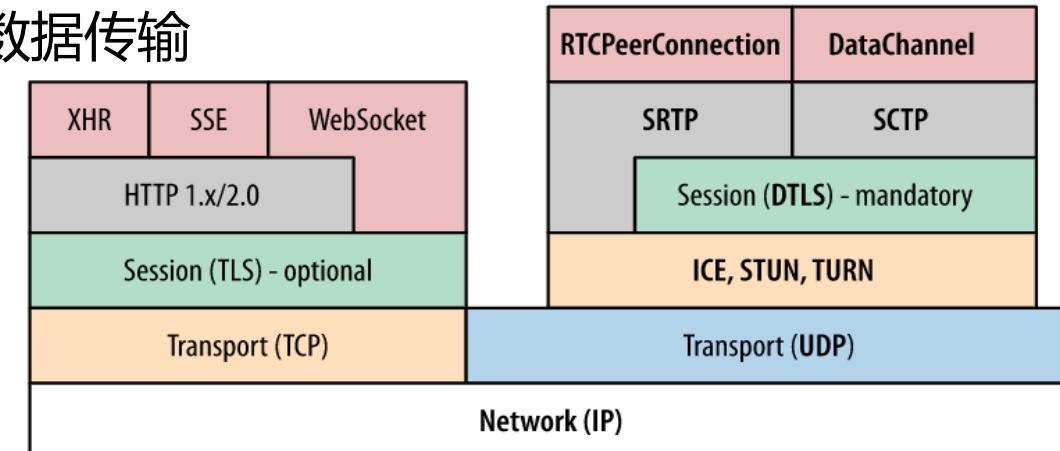
## ➤ 网页实时通信 WebRTC (Web Real-Time Communication)

- 由Google发起的实时音视频通信[开源项目](#)
- 建立浏览器之间点对点的连接，实现音/视频流的传输

开源项目与IETF国际标准  
同步推进

## ➤ WebRTC协议栈

- 为了满足实时性需求，其核心协议是在右侧基于 **UDP** 基础上搭建起来的
- **Secure RTP (SRTP)** 与 **Secure RTCP (SRTCP)** 是对媒体数据的封装与传输控制协议
- **RTCPeerConnection** 用来建立和维护端到端连接，提供高效的音视频流传输
- **RTCDataChannel** 用来支持端到端的任意二进制数据传输
- 流控制传输协议**SCTP**，提供类似 TCP 的特性
- DTLS 对传输内容进行加密，是 UDP 版 TLS
- ICE、STUN、TURN 用于内网穿透，应对NAT等私有地址转换的问题





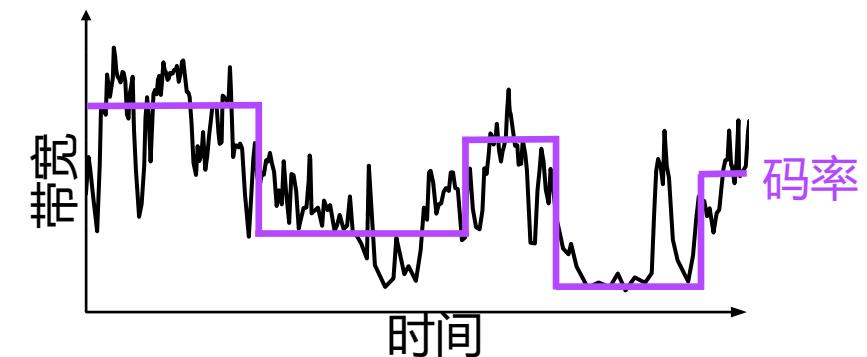
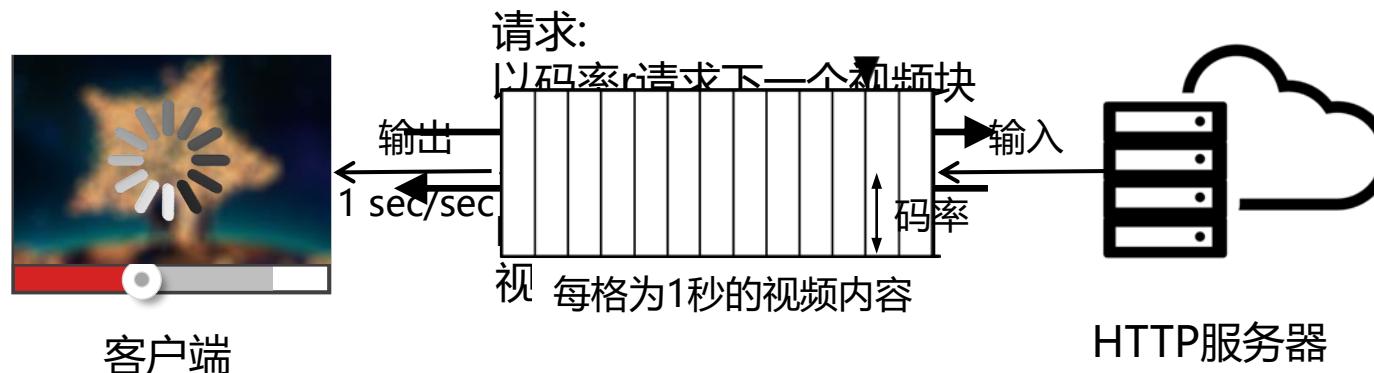
# 流媒体动态自适应传输

## ➤ 视频传输优化的挑战

- 客户端基于当前网络状况，向服务器请求视频块
- 若视频块的码率  $>$  可用带宽：视频块难以及时抵达客户端，出现卡顿
- 若视频块的码率  $<$  可用带宽：视频质量较低，没有充分利用带宽资源
- 如何为视频块选择贴近当前可用带宽的码率？

视频块的码率  
VS  
网络可用带宽

## ➤ 进一步挑战：网络带宽随时间不断变化



网络可用带宽不断变化时，  
如何及时选取合适的码率？



# 流媒体动态自适应传输

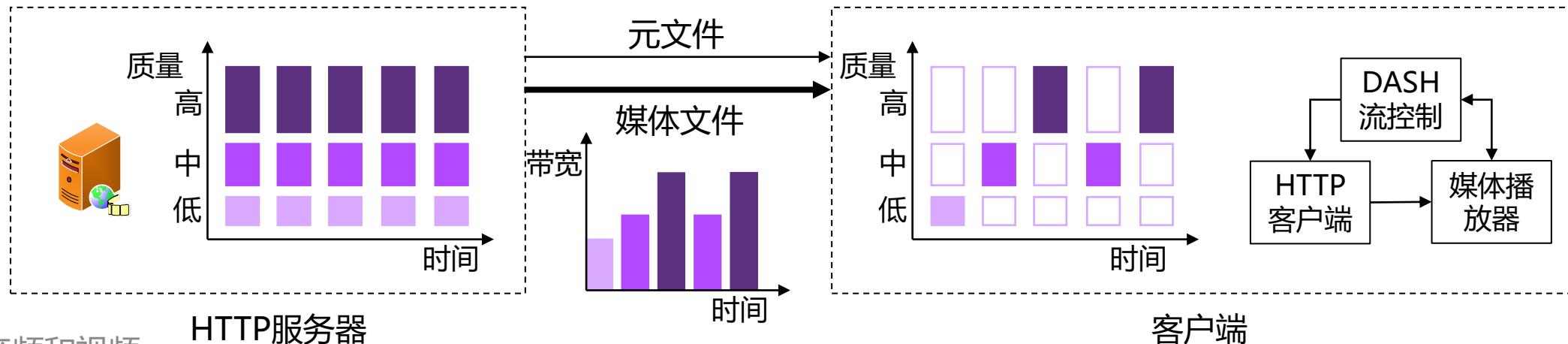
## ➤ DASH (Dynamic Adaptive Streaming over HTTP)

- 动态自适应流媒体传输协议DASH，由MPEG组织制定的标准
- 类似协议：苹果HTTP Live Streaming (HLS)；Adobe的HTTP Dynamic Streaming (HDS)；微软的Microsoft Smooth Streaming

DASH  
选取合适码率的  
协议

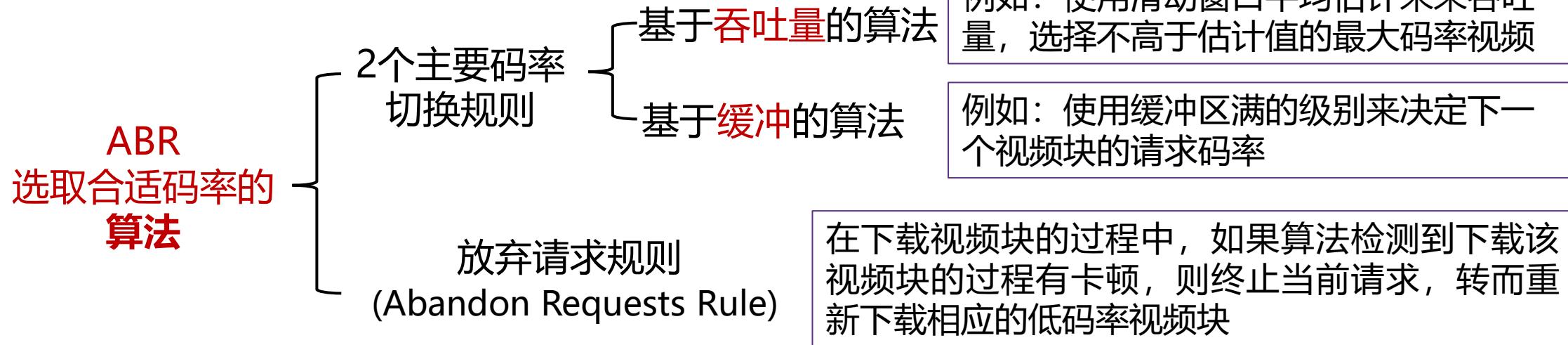
## ➤ DASH 基本思想

- 完整视频被拆分为固定时长 (2s-10s) 的视频片段(segment)，每段提供不同码率
- 视频片段与其对应的元文件 (URL) 一同存放于DASH服务器
- 客户端基于网络条件、缓冲大小等，对每个视频片段，**自适应选择合适的视频码率**来下载





## ➤ DASH中普遍使用的自适应码率ABR (Adaptive bitrate)

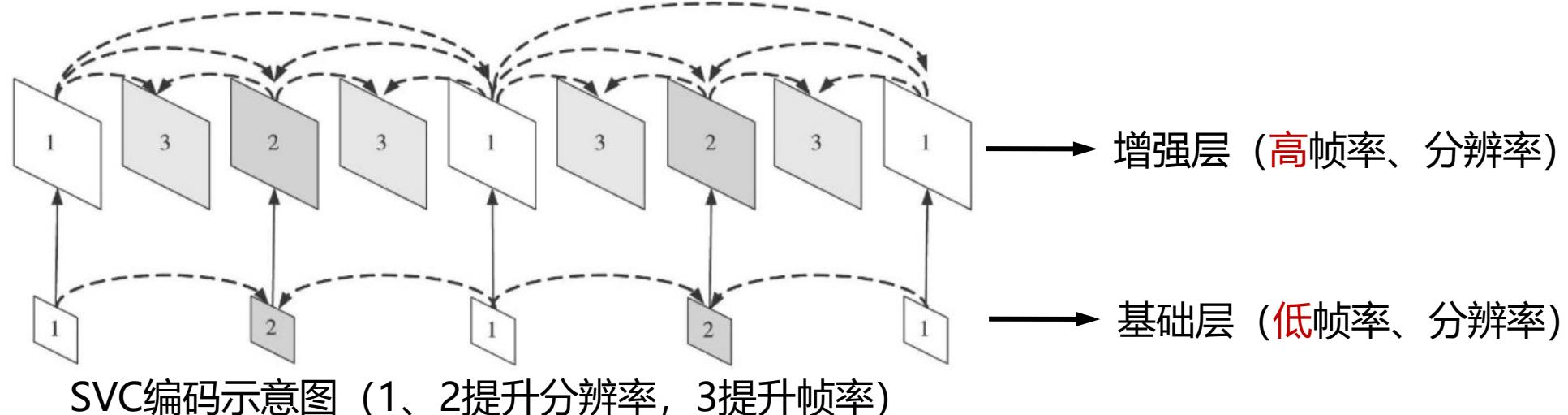


- 自适应码率ABR广泛应用于DASH和直播、实时通信等各类流媒体传输
- 由于IBP帧互相不独立，ABR调节往往以GOP为单位，难以实时调节
- 由于网络可用带宽难于预测，因此寻找与可用带宽匹配的最佳码率，很有挑战
- 选高清还是选低清帧的难题：(1) 选高清可能卡顿，(2) 选低清可能浪费带宽？  
(3) 低清、高清一起传，则收到高清时需要丢弃低清帧，也浪费带宽，如何解决？



## ➤ 可扩展视频编码 SVC (Scalable Video Coding)

- SVC 是以H.264为基础，支持多层分级特性：立足基础层，采用锦上添花的增强层
- 编码器产生的码流包含多个可以单独解码的子码流：不同的码率，帧率和空间分辨率
- 当带宽不足时，只对基础层的码流进行传输和解码，这时解码的视频质量不高
- 当带宽充足时，可以传输和解码增强层的码流来提高视频的解码质量
- 优点：确保传输基础层来避免卡顿，使用富裕的带宽传输增强层，充分利用带宽
- 缺点：编解码复杂度增加，有多个增强层时开销过大





## ➤ 应用层概述

- 进程通讯方式: C/S, B/S, P2P
- 服务器工作方法: 循环、并行
- TCP v.s. UDP, 还是?

## ➤ DNS与域名系统

- 扩展性: 域名空间与层次结构
- 递归/迭代解析: UDP
- 缓存提升效率/TTL, 安全与根

## ➤ 电子邮件 (文字)

- 基于ASCII的协议: 命令-响应
- 推: SMTP (TCP:25)
- 拉: POP3 (TCP:110), IMAP (TCP:143)

## ➤ 万维网Web与HTTP

- 统一资源定位符URL
- 静态HTML与两类动态web
- 无状态HTTP演进: 非持久到持久连接
- 缓存/代理, 安全隐私与cookie

## ➤ 流式音视频

- MPEG编码: IPB帧与GOP
- 流式存储媒体: HTTP的下载与播放
- 网络抖动的问题与应对
- 流媒体直播技术: RTSP/RTP
- 动态自适应DASH (ABR/SVC)



# 作业

- 完成课后习题，详见 PPT 后附录
  - 1 应用层模型
  - 2 DNS
  - 3 EMAIL
  - 4 web and HTTP
  - 5 流媒体
  - 注：附加题不需要提交
- 截止时间：下下周三晚11:59，提交网络学堂



# 致谢社区本章贡献者



计算机网络教案社区



姜誉

广州大学

1、应用层概述  
2、DNS



郝海龙

华北理工大学

3、电子邮件



王志晓

西安理工大学

4、WWW



于显平

西南大学



崔勇

清华大学

5、流式音频和  
视频



蔡红云

河北大学

6、内容分发



梁晓艳

河北大学



杨晓晖

河北大学

7、其它应用层  
协议

《计算机网络：自顶向下方法(原书第7版)》，库罗斯、罗斯，陈鸣译，机械工业出版社，2018年6月

《计算机网络（第5版）》，Tanenbaum & Wetherall，清华大学出版社，2012年3月

《计算机网络（第7版）》，谢希仁，电子工业出版社，2017年1月

《计算机网络教程（第6版）》，吴功宜，电子工业出版社，2018年3月

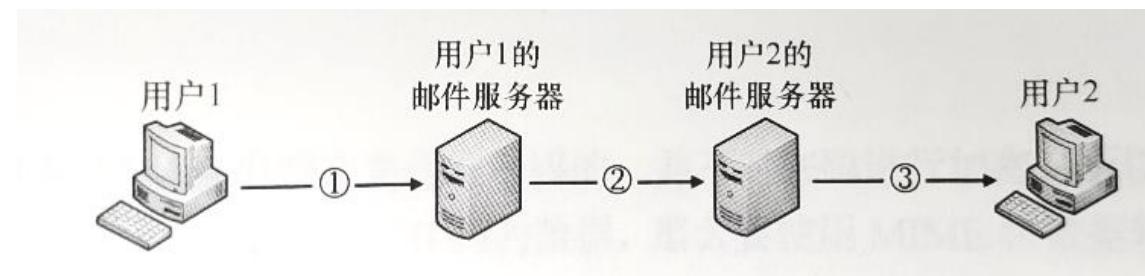
《计算机网络（第3版）》，徐敬东、张建忠，清华大学出版社，2013年6月

特别致谢：  
部分内容取材于此



# 作业

- 1. 下列关于网络应用模型的叙述中，错误的是（）
  - A. 在 P2P 模型中，结点之间具有对等关系
  - B. 在客户/服务器（C/S）模型中，客户与客户之间可以直接通信
  - C. 在 C/S 模式中，主动发起通信的是客户，被动通信的是服务器
  - D. 在向多个用户发送同一个文件时，P2P 模型通常比 C/S 模型需要的时间短
- 2. DNS 使用 UDP 而非 TCP，如果一个 DNS 分组丢失，没有自动恢复，那么这会引起什么问题吗？如果会，应该如何解决？
- 3. 若用户1与用户2之间发送和接收电子邮件的过程如下图所示，则图中1、2、3阶段分别使用的应用层协议可以是哪些？请分别列出尽可能多的结果。选取协议范围在课上所讲述的内容中选取即可。





# 作业

- 4. 12. 【2015 统考真题】某浏览器发出的 HTTP 请求报文如下：

```
GET /index.html HTTP/1.1
Host: www.test.edu.cn
Connection: Close
Cookie: 123456
```

下列叙述中，错误的是（ ）。

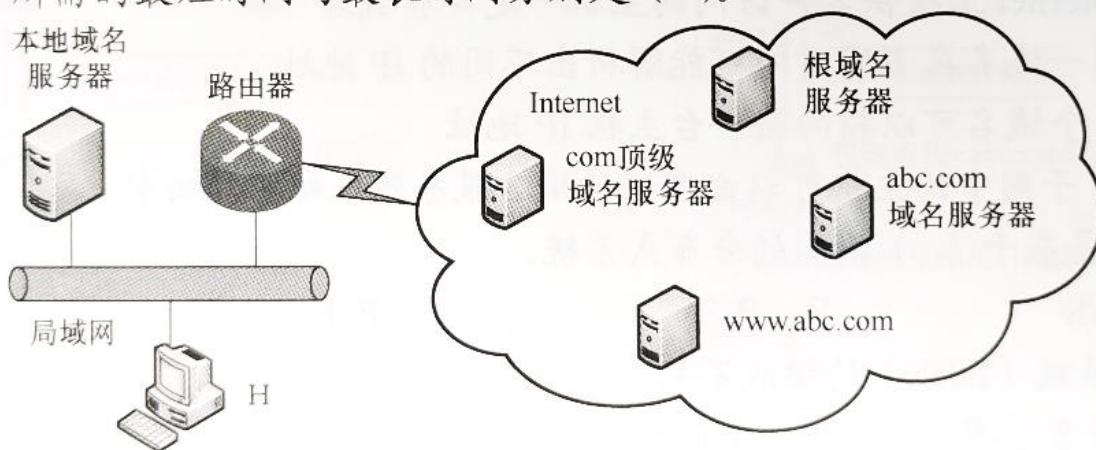
- A. 该浏览器请求浏览 index.html
- B. index.html 存放在 www.test.edu.cn 上
- C. 该浏览器请求使用持续连接
- D. 该浏览器曾经浏览过 www.test.edu.cn

- 5. 一个音频流服务器到一个媒体播放器的单向“距离”为 100ms，它以 1Mbps 的速率输出。如果媒体播放器有一个 2MB (MegaBytes) 的缓冲区，我们认为播放器只有播放和暂停两个操作且每一条指令在服务器接收后会被立刻执行。请问为了保证每一条指令都可以被正确执行且音频在播放过程中不会卡顿，那么应该在缓冲区有多少数据的时候开始播放？为了不浪费缓冲资源，又应该在缓冲区有多少数据的时候停止下载数据？



# 思考题

12. 【2020 统考真题】假设下图所示网络中的本地域名服务器只提供递归查询服务，其他域名服务器均只提供迭代查询服务；局域网内主机访问 Internet 上各服务器的往返时间 (RTT) 均为 10ms，忽略其他各种时延。若主机 H 通过超链接 <http://www.abc.com/index.html> 请求浏览纯文本 Web 页 index.html，则从单击超链接开始到浏览器接收到 index.html 页面为止，所需的最短时间与最长时间分别是（ ）。



- A. 10ms, 40ms    B. 10ms, 50ms    C. 20ms, 40ms    D. 20ms, 50ms

12. D 题中 RTT 均为局域网内主机（主机 H、本地域名服务器）访问 Internet 上各服务器的往返时间，且忽略其他时延，因此主机 H 向本地域名服务器的查询时延忽略不计。最短时间：本地主机建立中有该域名到 IP 地址对应的记录，因此不需要 DNS 查询时延，直接和 www.abc.com 服务器建立 TCP 连接再进行资源访问，TCP 连接建立需要 1 个 RTT，接着发送访问请求并收到服务器资源响应需要 1 个 RTT，共计 2 个 RTT，即 20ms；最长时间：本地主机递归查询本地域名服务器（延时忽略），本地服务器依次迭代查询根域名服务器、com 顶级域名服务器、abc.com 域名服务器，共 3 个 RTT，查询到 IP 地址后，将该映射返回给主机 H，主机 H 和 www.abc.com 服务器建立 TCP 连接再进行资源访问，共 2 个 RTT，因此最长时间需要  $3 + 2 = 5$  个 RTT，即 50ms。