应用层

网络应用的体系架构

- 客户-服务器模式（CS）
- 对等模式（P2P）
- 混合体

CS架构

- server:
  - always-on 主机
  - 永久IP address
  - 可扩展：服务器集群
- clients:
  - 与服务器通信
  - 可能是间歇性的连
  - 可能是动态的IP地址
  - 通常相互之间不直接通信

对等体系架构

- 每个节点既是服务机也是客户机
- 参与的主机简写性连接且可以改变IP地址

混合体

文件搜索：集中

文件传输：P2P

在线检测：集中

两个用户之间通讯：P2P

进程通信

进程：在主机上运行的应用程序

同一个主机内使用进程间通信机制通信（操作系统定义）

不同主机之间通过交换报文来通信

- 客户端进程：发起通信的进程

禁止商用

- 服务端进程：等待连接的进程

分布式进程通信

对进程进行编址

进程为了接收报文，必须有一个标识，即IP地址和端口号

本质上，一对主机进程之间的通信由2个端节点构成

层间信息传输，以传输层为例子

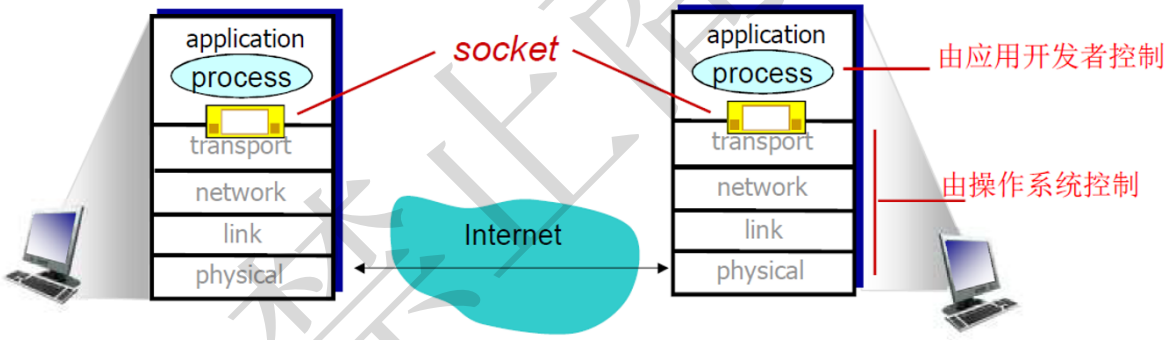
应用层将必要的信息传给传输层，进行TCP报文段（UDP数据报）的封装

但是如果每次都要传输报文，携带的信息太多，容易出错，使用一个代号来标识通信的双方或者单方。

目的：使得穿过层间的信息量最少

套接字

可以将套接字看作应用程序对下层的门户



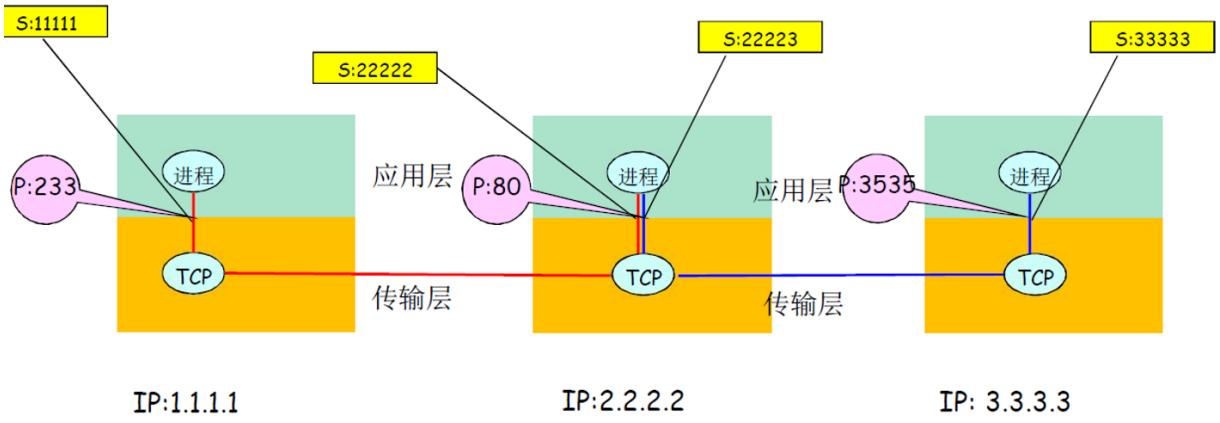
禁止商用

TCP socket

TCP socket：源IP，源端口，目的IP，目的端口

TCP服务：两个进程之间通信之间要建立连接，两个进程之间的通信会**持续一段时间**，通信关系稳定

可以用一个整数表示两个应用实体之间的通信关系，**本地**标识。套接字是四元组（源IP，源端口，目的IP，目的端口）的一个**具有本地意义的标识**



UDP socket

UDP socket：本地IP，本地端口

UDP服务：两个进程之间的通信之前无需建立连接。

可以用一个整数表示**本应用实体的标识。**

但是传输报文时必须提供对方的IP和端口

应用层协议

定义运行在不同端系统上的应用进程如何交换报文

web与HTTP

web页：由一些对象组成

HTTP

超文本传输协议

使用TCP：客户机发起与服务器的TCP连接，端口号为80

HTTP是**无状态**的，服务器不维护关于客户的任何信息

非持久HTTP

最多只有一个对象在TCP连接上发送

下载多个对象需要多个TCP连接

HTTP/1.0采用非持久连接

禁止商用

特点

- 每个对象需要两个RTT
- 操作系统必须为每个TCP连接分配资源
- 浏览器通常打开并行TCP连接，来获取对象

持久HTTP

多个对象在TCP连接上发送

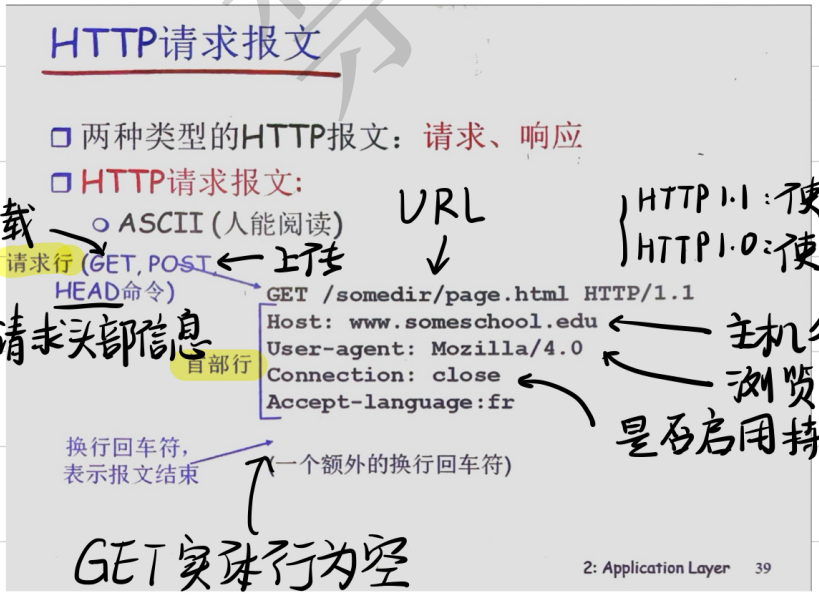
HTTP/1.1默认使用持久连接（也可使用非持久的）

特点

- 服务器在发送响应后，仍然保持TCP连接
  - 非流水方式：
    - 客户机只有收到前一个响应后才能发出新的请求
    - 每个引用对象花费1个RTT
  - 流水方式：
    - HTTP/1.1的默认模式
    - 客户端遇到一个引用对象就立刻产生一个请求
    - 所有对象使用1个RTT是可能的
- 客户端遇到一个引用对象的时候，就尽快发送该对象的请求

HTTP报文

请求报文



禁止商用

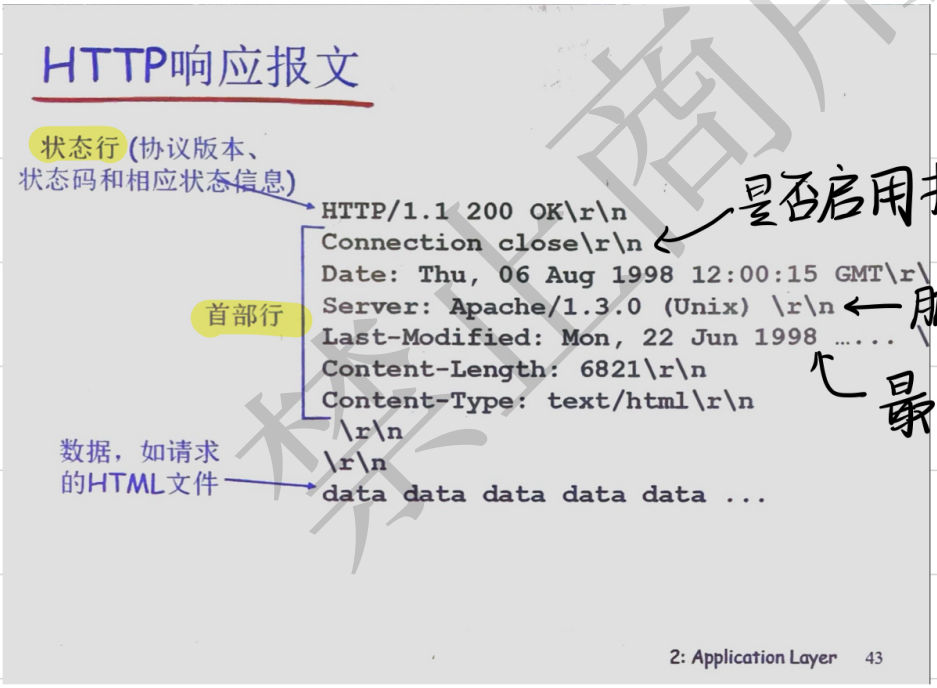
递交表单

- POST方式：包含在实体主体中的输入被递交给服务器
- URL方式：输入内容通过请求行的URL字段上传

方法类型

- HTTP/1.1
  - GET,POST,HEAD
  - PUT：将实体主体中的文件上传到URL字段规定的路径
  - DELETE：删除URL字段规定的文件
- HTTP/1.0
  - GET
  - POST
  - HEAD：只请求头部信息

响应报文



状态码

位于服务器响应报文的首行

200 OK

- 请求成功，请求对象包含在响应报文的后续部分

301 Moved Permanently

- 请求的对象已经被永久转移了；新的URL在响应报文的Location:首部行中指定
- 客户端软件自动用新的URL去获取对象

400 Bad Request

- 一个通用的差错代码，表示该请求不能被服务器解读

404 Not Found

- 请求的文档在该服务上没有找到

505 HTTP Version Not Supported

响应时间模型

往返时间RTT

一个RTT用来发起TCP连接

一个RTT用来HTTP请求并等待HTTP连接时间

*响应时间 = 2*RTT + 传输时间*

用户端状态Cookies

组成部分

- HTTP响应消息头部
- HTTP请求消息头部
- 保留在客户端的Cookies
- Web站点后台数据库

实现

在多个事务上，发送端和接收端维持状态

cookies：http报文携带状态信息

作用

Cookies允许站点知道许多关于用户的信息，便于内容提供商更好提供服务

Web Cache（代理服务器）

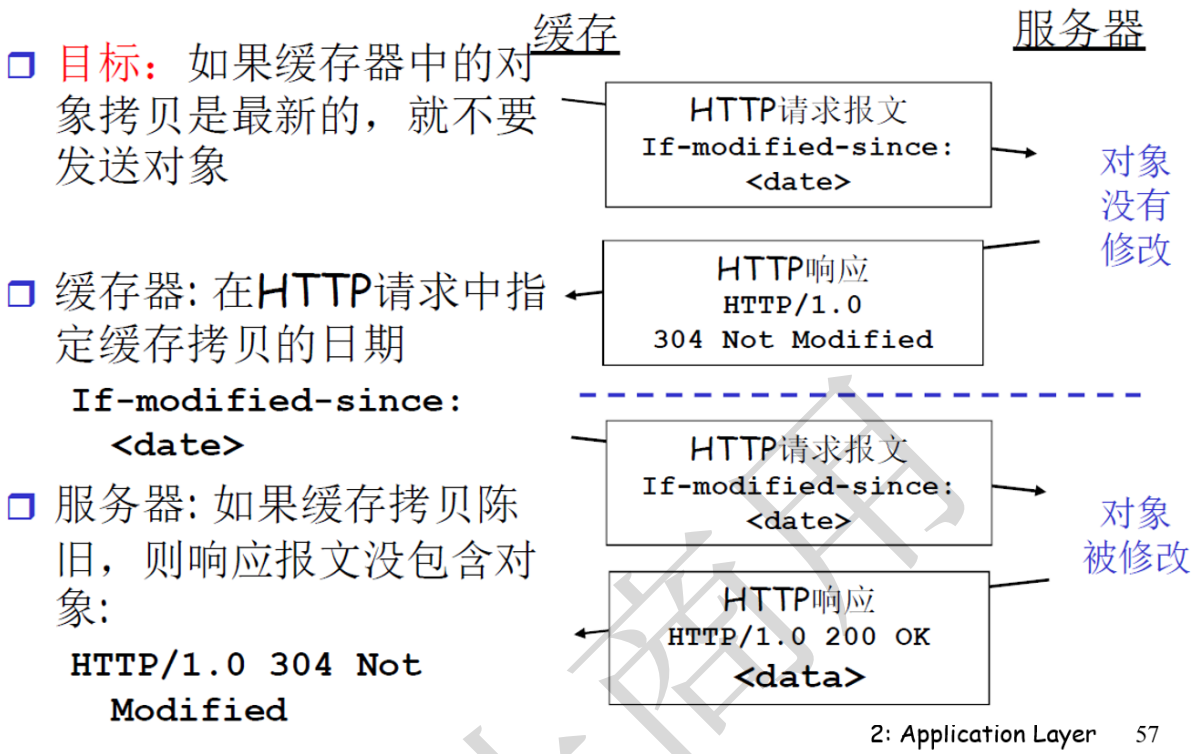
与计算机中的Cache作用和实现相似

禁止商用

作用

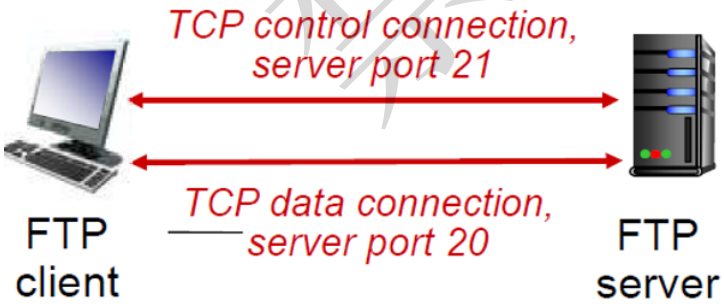
- 减少客户端请求响应时间
- 减少瓶颈路线流量

条件GET方法



FTP文件传输协议

控制连接和数据连接分开



使用分离的**控制连接**，称为**带外传送**

FTP服务器维护用户的状态信息（有状态）

禁止商用

FTP命令/响应

命令样例：

- 在控制连接上以ASCII文本方式传送
- USER *username*
- PASS *password*
- LIST：请服务器返回远程主机当前目录的文件列表
- RETR filename：从远程主机的当前目录检索文件(gets)
- STOR filename：向远程主机的当前目录存放文件(puts)

返回码样例：

- 状态码和状态信息 (同HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can’ t open data connection
- 452 Error writing file

Email

组成

- 用户代理
- 邮件服务器
- 简单邮件传输协议（SMTP）

SMTP

使用TCP从客户端可靠的传输邮件信息到服务器

使用端口25

直接传送：发送服务器直接发送邮件至接收服务器

- 握手
- 传输
- 关闭

只支持传输7bit ASCII码内容，若要传送其他数据要使用MIME协议将数据转化为ASCII码

SMTP和HTTP比较

HTTP: 拉 pull

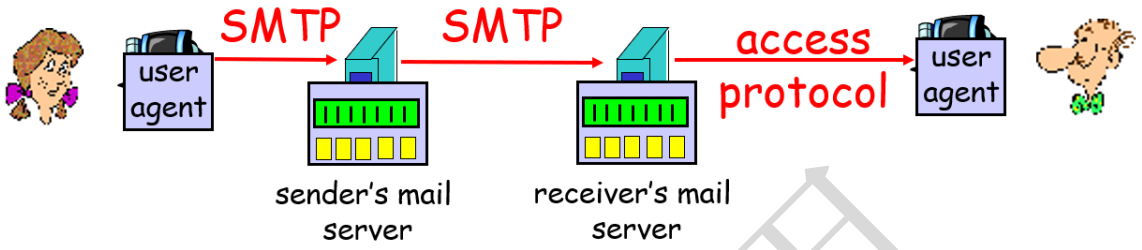
禁止商用

SMTP: 推 push

都使用ASCII command/response 进行交互, status codes

- HTTP:
  - 每个 object 都有响应消息
- SMTP:
  - 多个objects 在同一个消息中发送

邮件访问协议



- SMTP: 发送邮件到接收者服务器
- Mail access protocol: 从邮件服务器获取邮件
  - POP: Post Office Protocol [RFC 1939]
    - 认证 (agent <-->server) 和下载
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - 更多的特征
    - 能够在服务器上对邮件做某些操作
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3

- 认证阶段
- 事务阶段

POP3是**无状态**的协议

在传输层使用明文传输密码

IMAP

所有信息保留在一个地方

所有用户通过文件夹组织消息

IMAP中sessions中保存用户的状态

- 文件夹的名字

禁止商用

- 文件夹和消息的ID对应关系

DNS

域名系统：完成域名对于IP地址的映射

- 分布式数据库
  - 具有层次的多个域名服务器
- 应用层协议
  - 主机，路由，域名服务器
  - 相互通信完成名字解析

注意:DNS是一个应用层部署的核心协议，协议的复杂性位于网络的边缘

DNS服务器

- 主机名到IP地址的翻译
- 主机别名
- 邮件服务器别名
- 负载均衡

分布式的DNS有利于扩展

查询过程

先在本地服务器中查询，再到根服务器中查询，最后到根服务器中保存的权威服务器中查询

根域名服务器

与权威服务器联系完成域名映射

返回给本地域名服务器结果

TDL和权威服务器

TDL：顶级域名服务器，负责com, org, net, edu, etc,以及所有区域顶级域名 uk, fr, ca, jp

（授权）权威域名服务器：一个单位的域名服务器，本单位负责服务器与IP地址的映射

本地域名服务器

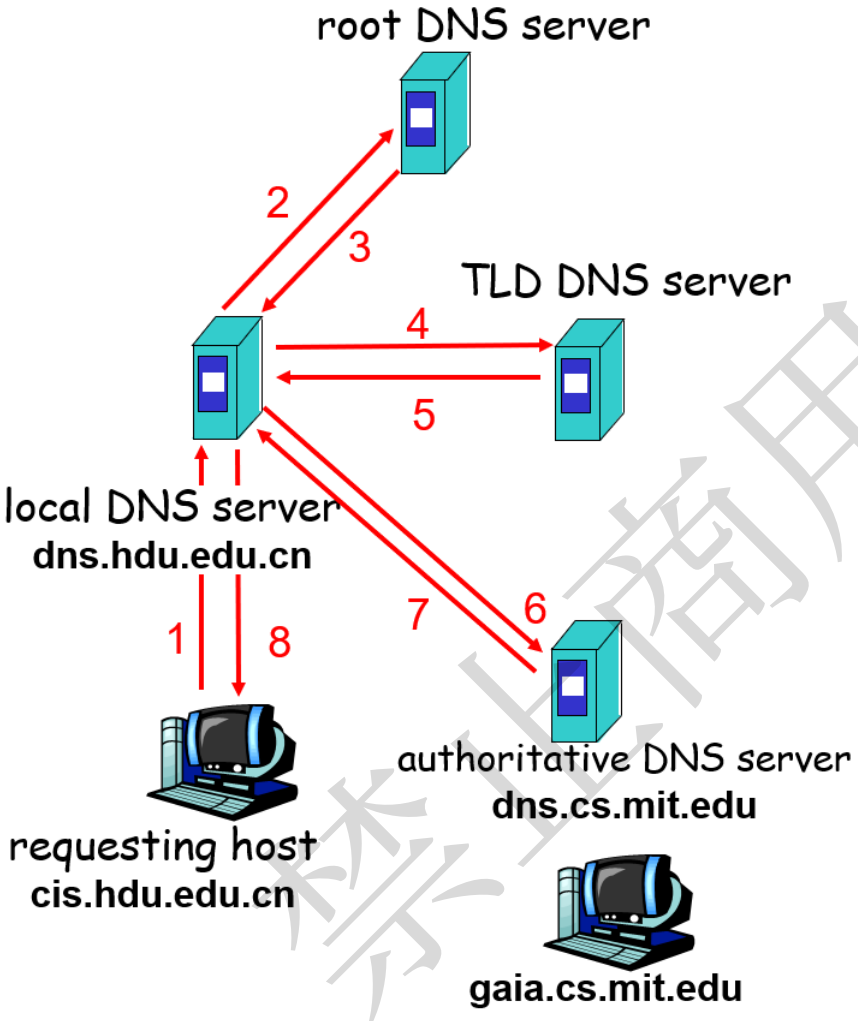
本地域名服务器也称“默认域名服务器”，不是严格的属于域名服务器的层次结构

禁止商用

- 每个 ISP (区域ISP, 公司, 大学) 都有自己的LNS.
- 当主机进行DNS 查询, 查询转发给LNS
    - LNS作为代理, 向域名服务器系统转发查询

迭代查询

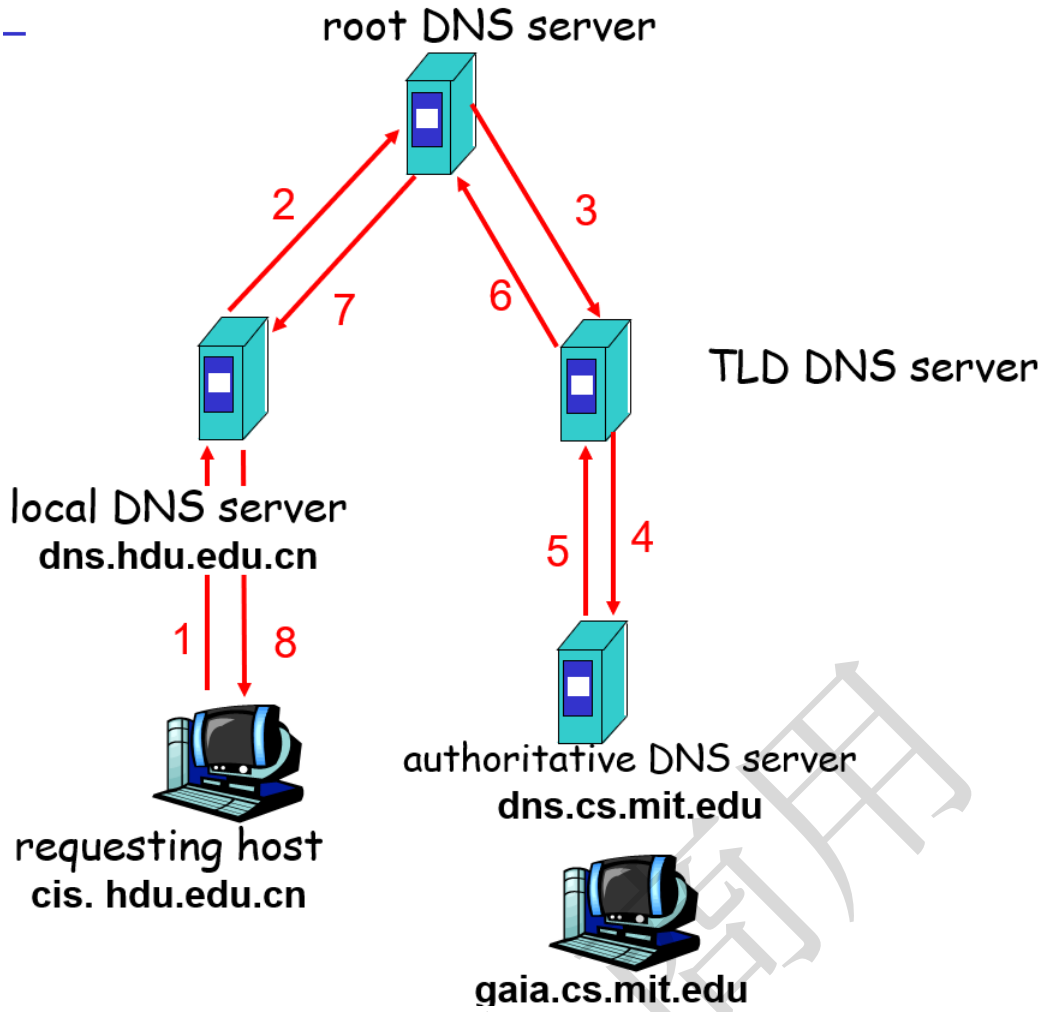
LNS依次查询域名服务器层次结构中的各个服务器，最终获得该主机地址



禁止商用

递归查询

将查询的负担完全交给了查询对象



DNS的Cache和记录更新

- 一个 DNS server 了解到域名映射后, 它会caches 映射
  - cache 有一定的 timeout 时间
  - TLD servers 通常会在LNS cache保存

DNS记录

DNS: 分布式存储数据库记录(resource records **RR**)

- RR format: (name, value, type, ttl)
  - Type=A
    - name 是主机名
    - value 是地址
  - Type=NS
    - name 域名(e.g. foo.com)
    - value 是本域的权威域名服务器
  - Type=CNAME
    - name 是某台主机的别名
    - value是权威的名字

禁止商用

    - Type=MX
        - value 是邮件服务器的名字

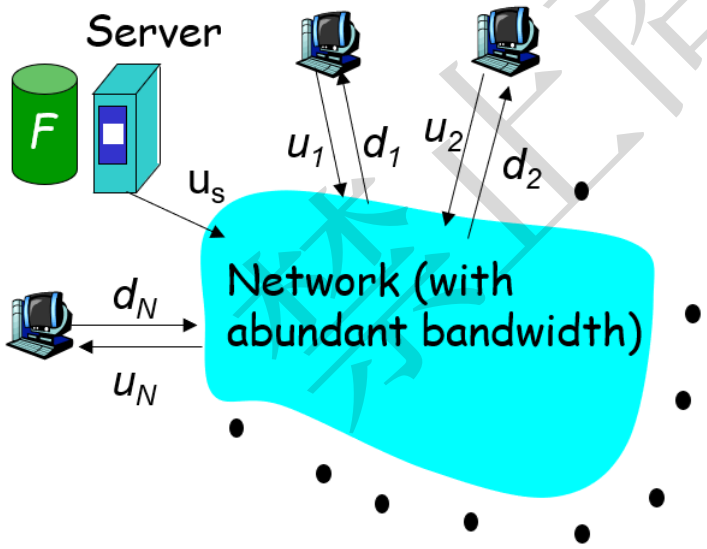
DNS安全性

- DDOS攻击
    - 流量攻击根服务器
    - 流量攻击TLD服务器
- 重定向攻击
    - 查询劫持
    - DNS污染：发送虚假的DNS响应

P2P应用

- 没有根服务器
- 端系统之间直接通讯
- 端系统：经常改变IP

文件分发



$$P2P = \max\{F/u_s, F/\min(d_i), NF/(u_s + \sum u_i)\}$$

BitTorrent

Peer 加入 torrent:

- 开始没有 chunks, 但是慢慢从其它地方下载积累
- 在 tracker 注册获得 peers列表,与相邻的 peers（“neighbors”)取得联系
- Peer downloading, 同时 uploading chunks 给其他 peers.
- Peer 加入离开是动态

学生商用

- Peer 下载完成, 可能离开 (selfishly) 或继续 (altruistically)

Pulling Chunks

- 不同 peers 具有不同的文件 chunks
- 周期性的peer (Alice) 问相邻peer获得 chunks 列表.

发送chunks 给四个速度最快的邻居，随机选择一个节点加入传输

最稀缺优先：针对没有的块在邻居中决定最稀缺的块，并首先请求最稀缺的块

使用对换算法：根据当前以最高速率向自己提供数据的邻居，给出优先权

DHT

在数百万的peer间分配 (key, value) 对

- 键值对在peers间均匀分布
- 所有的peer 能够query 在数据库中查询关键字
- 数据库返回 key的value
- 查询过程中, 相关的peers节点间交换少量的消息
- 每个 peer 仅知道几个其它的peer节点
- Peer 能够加入和离开(churn)

环状 DHT

带近路的环状DHT

在环状DHT的基础上添加近路

视频流化服务和CSN：上下文

提高视频播放性能

通过全网部署节点，存储服务内容

码率：每秒的数据量

了解

禁止商用