第2章 应用层

本章学习目标

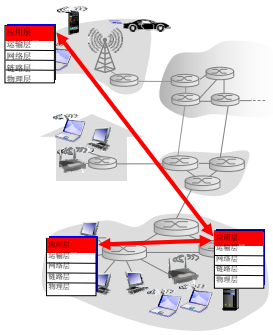
- 网络应用的原理与实现
  - 应用层所需的网络服务
  - 客户和服务器
  - 进程和传输层接口
  - 创建网络应用程序
    - 套接字接口
- 通过观察流行的网络应用，学习相关协议
  - Web：HTTP
  - 电子邮件：SMTP / POP3 / IMAP
  - 域名解析：DNS
  - 对等文件分发:P2P
  - 地址分配：DHCP(网络层)

第二章:应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

2.1.1 网络应用程序体系结构

- 研发网络应用程序的核心目标：能够在不同的端系统上运行，并通过网络相互通信
- 只将应用限制在端系统，用户应用程序代码与网络核心设备无关
- 优点：促进了大量网络应用程序的迅速研发和部署。



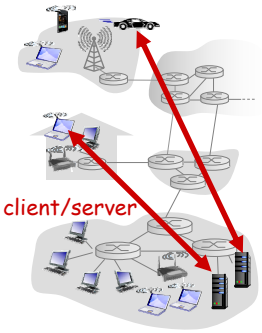
2.1.1 网络应用程序体系结构

- 网络应用程序体系结构：由研发者设计，规定了在各种端系统上组织应用程序的规则
- 两种主流体系结构：
  - 客户—服务器
  - 对等 (P2P)

客户—服务器体系结构

服务器:
- 永远在线
- 具有固定的、众所周知的IP地址
- 使用数据中心提供不间断服务

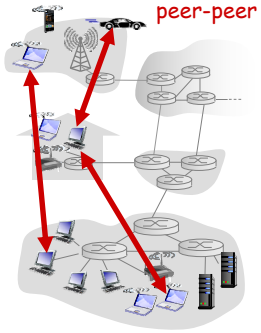
客户机:
- 只与服务器通信
- 可以间歇地连接服务器
- 可以具有动态的IP地址
- 彼此之间并不通信



纯P2P体系结构

- 无需永远在线的服务器
- 任意的端系统直接通信
- 对等方彼此请求服务，并提供服务。
- 对等方间歇地连接，并允许改变IP地址
- 典型应用：BiTorrent、Skype

高度地可扩展、但是难以管理



客户机—服务器与P2P的混合

即时讯息类应用：微信、QQ、Skype等

- 客户端之间的报文，在双方主机之间直接发送。
- 服务器集中式检测/定位用户:
  - 当用户在线时，向中心服务器注册其IP地址
  - 用户联系中心服务器以发现远程伙伴的IP地址

2.1.2 进程通信

进程：运行在端系统上的应用程序。

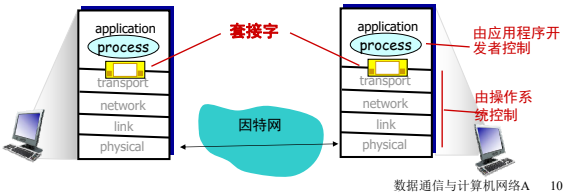
- 同一个端系统中的两个进程使用IPC(进程间通信机制，由操作系统定义)通信.
- 不同端系统中应用程序以进程对的方式通信，进程对通过交换报文而相互通信

客户机进程:
发起通信的进程
服务器进程：
等待连接的进程

- 注意：具有P2P体系结构的应用程序，同时具有客户机进程和服务器进程

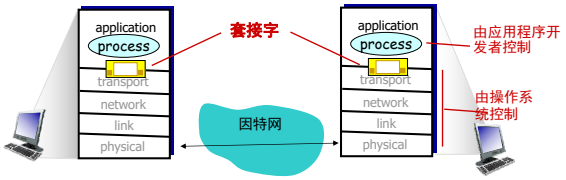
套接字（应用程序编程接口）

- 进程通过套接字在网络上发送/接收报文
- 套接字类似于门
  - 发送进程将报文推出门外
  - 发送进程依赖门的另一侧传输层基础设施，将报文送到接收进程的套接字



套接字（应用程序编程接口）

- 应用程序开发者仅能控制套接字在应用层一侧的动作，对于传输层，仅能：
  - 选择传输层协议;
  - 确定一些传输层参数，如最大缓存，最大报文段等。



进程寻址

- 对于发送方进程，必须标识接收方进程
- 问题：接收方主机的IP地址是否足以唯一标识接收进程？
- 答案: 不行，因为在同一台主机上能够运行许多进程。

- 合理的标识必须包括：
  - 主机IP地址
  - 主机上该进程相关的端口号
- 例如向gaia.cs.umass.edu web服务器发送HTTP消息：
  - IP address: 128.119.245.12
  - Port number: 80

2.1.3 应用程序需要什么样的运输服务？

可靠的数据传输

- 某些应用（如音频）能够容忍一定程度的数据丢失
- 其他应用（如文件传输，Telnet）要求100%可靠数据传输

实时性

- 某些应用（如因特网电话、交互式游戏）对数据交付有严格的时间限制，要求“有效的”低时延

吞吐量

- 对某些带宽敏感的应用（如多媒体），必须提供“有效的”最小量的带宽
- 其他应用（”弹性应用“）能够根据需要，充分利用可供使用的所有带宽

安全性

- 必须能为应用提供安全性服务
- 为发送主机加密发送进程传输的所有数据
- 将数据交付给接收进程之前解密这些数据
- 数据完整性、端点鉴别

2.1.4 因特网提供的运输服务

TCP服务:

- 面向连接: 客户机和服务器之间需要的建立
- 可靠传输：在发送和接收进程之间
- 流控制: 发送方不会淹没接收方
- 拥塞控制: 当网络过载时抑制发送方
- 不提供:实时性、吞吐量保证、安全性

UDP服务:

- 在发送进程及接收进程之间的不可靠数据传输
- 不提供：可靠性，流控，拥塞控制，定时、带宽保证、安全性

安全的TCP

TCP和UDP:

- 无加密:明文口令如果进入套接字，将在网络上以明文传输

SSL:

- 提供加密、数据完整性、端点鉴别
- 位于应用层，应用程序使用SSL库，向SSL套接字传输明文，SSL加密该数据并将其传递给TCP套接字，经因特网传输，由接收方SSL进行解密

普通应用的运输服务要求

应用程序	数据丢失	带宽	时间敏感
文件传输	不能丢失	弹性	不
电子邮件	不能丢失	弹性	不
Web 文档	不能丢失	弹性	不
实时音频/视频	容忍丢失	音频: 5kbps-1Mbps 视频:10kbps-5Mbps	是, 100’ s msec
流式存储音频/视频	容忍丢失	同上	是, 几秒
交互式游戏	容忍丢失	几kbps以上	是, 100 msec
智能讯息	不能丢失	弹性	是和不是

因特网应用：应用协议与运输协议

应用	应用层协议	支撑的传输层协议
电子邮件	SMTP [RFC 2821]	TCP
远程终端访问	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流媒体	HTTP、RTP	TCP或UDP
因特网电话	SIP、RTP	TCP或UDP，通常用UDP

2.1.5 应用协议定义

- 应用层协议定义了：
  - 交换的报文类型，如请求和响应报文
  - 各种类型报文的语法：报文各字段及其详细描述
  - 字段的语义，即字段中信息的含义
  - 进程何时、如何发送和响应报文的规则
- 网络应用和应用层协议的关系：协议只是应用的一部分
  - 如：Web应用和HTTP协议

2.1.6 常见的网络应用

- E-mail
- Web
- 即时通讯
- 远程登陆
- P2P文件共享
- 多用户网络游戏
- 流式存储视频(如YouTube)

- 因特网电话(例如Skype)
- 实时视频会议
- 社交网络
- 信息搜索
- ……

第二章：应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

Web和HTTP

某些专业术语

- Web页由对象组成
- 对象可以是HTML文件，JPEG图片，Java小程序，音频文件，...
- 多数Web页含有一个基本HTML文件以及若干引用对象
- 通过每个对象的URL地址引用对象
- URL的例子:

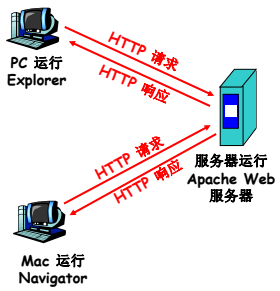
www.someschool.edu/someDept/pic.gif

主机名　　路径名

2.2.1 HTTP概述

HTTP: 超文本传送协议

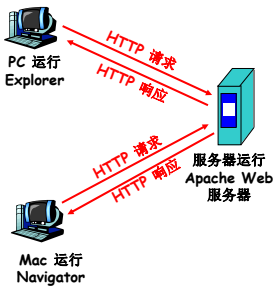
- Web的应用层协议
- 由运行在不同端系统中的客户程序和服务器程序，通过HTTP 报文实现Web应用
- HTTP定义了报文的结构，以及报文交换的格式
  - HTTP 1.0: RFC 1945
  - HTTP 1.1: RFC 2616
  - HTTP 2.0:RFC 7540



2.2.1 HTTP概述

HTTP: 超文本传送协议

- 客户/服务器模式
  - 客户: 请求、接收、显示Web对象，又称Web浏览器
  - 服务器: 存储Web对象，并响应客户请求向其发送对象



2.2.1 HTTP概述 (续)

使用TCP:

- 客户（浏览器）向服务器发起TCP连接(产生套接字)，端口80
- 服务器从客户接受TCP连接
- 浏览器和Web服务器进程，通过套接字访问TCP，交换HTTP报文 (应用层协议报文)
- 关闭TCP 连接

HTTP是”无状态的“

- 服务器不存储客户过去请求的任何信息

维护“状态”的协议是复杂的!

- 过去历史（状态）必须维护
- 如果服务器/客户机崩溃，“状态”的视图可能不一致，必须要重新建立

2.2.2 HTTP连接

非持续连接的 HTTP

- 至多一个对象经过一个TCP连接发送.
- HTTP/1.0使用非持久HTTP

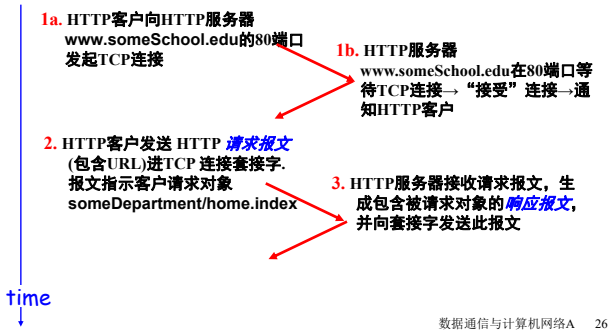
持续连接的HTTP

- 多个对象能够经过客户和服务器之间的单个TCP连接发送.
- HTTP/1.1以默认模式使用持久连接

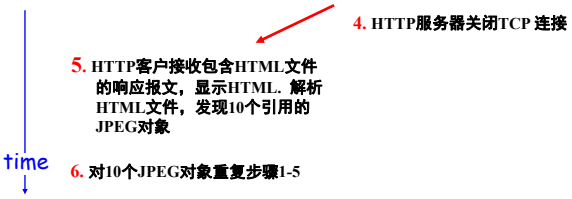
非持续连接的HTTP

假定用户输入URL （包括文本和对10个jpeg图片的引用）

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



非持续连接的HTTP(续)



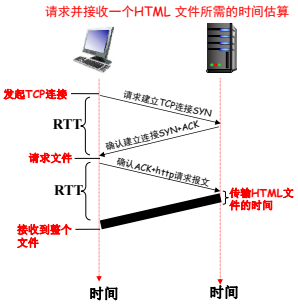
响应时间建模

往返时延RRT的定义:

- 一个短分组从客户到服务器然后再返回客户所花费的时间

响应时间:

- 总计 = 2RTT+文件传输时间



持续连接的HTTP

非持续连接的HTTP问题:

- 每个对象要求 2RTT
- 操作系统必须为每个TCP连接分配缓冲区和保持TCP变量

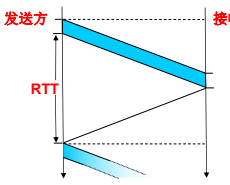
持续连接的HTTP

- 在发送响应后，服务器保持TCP连接打开
- 在相同的客户/服务器之间的后继HTTP报文通过该连接发送

持续连接的HTTP

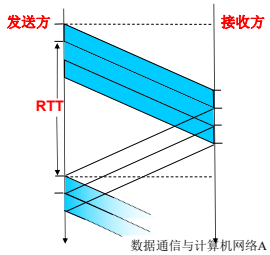
无流水线的持续:

- 仅当前面的响应已经收到，客户才发出新的请求
- 对每个引用对象一个RTT



有流水线的持续:

- 在HTTP/1.1 为默认
- 只要客户遇到一个引用对象，它发送请求
- 对于所有引用的对象花费一个RTT时间



2.2.3 HTTP报文格式

- 两类HTTP报文：请求，响应
- HTTP请求报文：
  - 普通ASCII文本书写

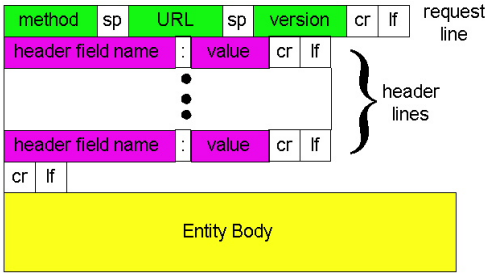
请求行 (GET, POST, HEAD、PUT、DELETE)　方法　URL　版本

```
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
```

首部行

回车，换行指示报文的结束

HTTP 请求报文: 通用格式



上载表单输入

- Web页通常包括表单输入，响应回来的Web页的特定内容依赖于用户在表单字段中输入的值

Post方法:

- Post报文的entity body（实体体）字段的输入被上传到服务器

URL方法:

- 使用 GET方法
- 将表单字段的输入传送到正确URL，然后被上传到服务器

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


方法类型

HTTP/1.0

- GET
- POST
- HEAD
  - 服务器收到HEAD方法的请求时，用一个HTPP报文响应，但并不返回请求对象
  - 通常用于调试跟踪

HTTP/1.1

- GET, POST, HEAD
- PUT
  - 向URL字段中定义的路径，上传在实体主体字段中定义的文件
- DELETE
  - 删除服务器上在URL字段中定义的文件

HTTP 响应报文

状态行 (协议状态码状态短语)

```
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 ...
```

首部行

数据，如请求的HTML文件

HTTP响应状态码

在服务器到客户机响应报文中的首行，一些编码的例子:

200 OK
- 请求成功，请求的对象在返回的响应报文中

301 Moved Permanently
- 请求的对象已被永久转移，新的URL在响应报文的Location:首部行中指定

400 Bad Request
- 请求报文不能被服务器理解

404 Not Found
- 被请求的文档不在该服务器上

505 HTTP Version Not Supported
- 服务器不支持请求报文使用的http协议

2.2.4 用户与服务器的交互：cookies

许多重要的Web站点使用cookies

四个部分:

1)在HTTP响应报文中有cookie首部行

2)在HTTP请求报文中有cookie首部行

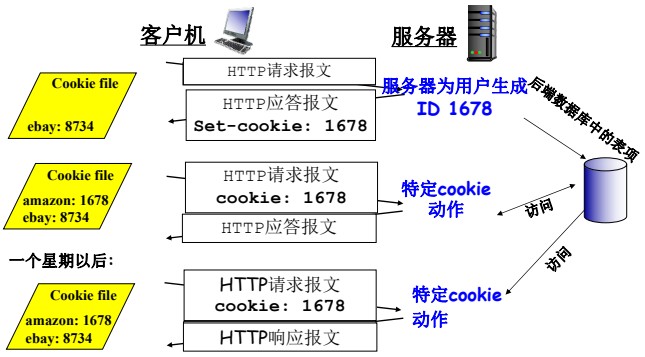
3)保持在用户主机中的 cookie文件，并由用户浏览器管理

4)位于Web站点的后端数据库

例子:

- Susan总是从相同的PC访问因特网
- 她首次访问一个特定的电子商务站点
- 当起始HTTP请求到达站点时，站点产生一个独特的ID，并为ID在后端数据库中生成一个表项

Cookies: 保持“状态” (续)



Cookies (续)

cookies 能够做:

- 鉴别、跟踪用户
- 购物车
- 推荐商品
- 用户会话状态(Web电子邮件)

Cookies和隐私：

- Cookies使得站点了解用户个人隐私信息
- 可能向站点提供名字和电子邮件

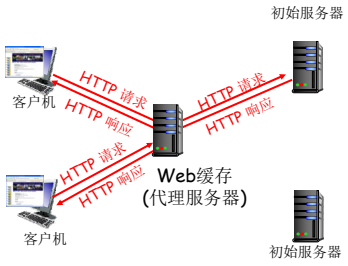
如何保持“状态”:

- cookies:利用http消息携带状态

2.2.5 Web缓存(代理服务器)

目标: 满足客户机请求而无需访问初始服务器

- 用户设置浏览器：允许通过Web缓存访问对象
- 浏览器向Web缓存发送所有HTTP请求
  - 对象在缓存中：Web缓存返回对象
  - 否则Web缓存向初始服务器请求对象，然后向客户机返回对象



2.2.5 Web缓存

- Web缓存即是客户机，同时又是服务器
  - 相对于客户端浏览器是服务器
  - 相对于初始服务器是客户端
- 典型的Web缓存通常由ISP(大学，公司和区域ISP)安装

为什么使用Web缓存?

- 减少客户端请求的响应时间
- 减小机构访问链路的流量
- 因特网密集安装缓存使得内容提供商能有效地交付内容（对P2P文件共享也是这样）

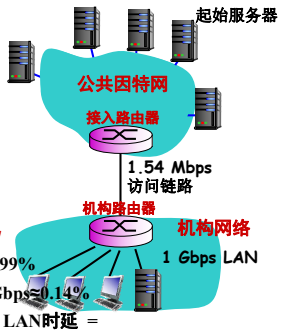
例子

假定

- 平均对象长度 = 100k比特
- 机构网络内浏览器到初始服务器的平均访问速率= 15个/秒，通过浏览器发送的平均数据率=15个/秒*100k比特=1.5Mbps
- 接入路由器到任何初始服务器的平均时延= 2秒，即因特网时延

结果

- 访问链路流量强度=1.5Mbps/1.54Mbps≈99%
- 局域网流量强度 = 15个/秒*100k比特/1Gbps=0.14%
- 总时延 = 因特网时延 + 访问链路时延 + LAN时延 = 2秒 + 分钟级 + 微秒级



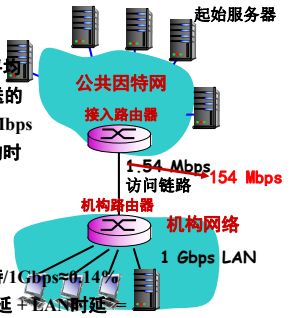
例子： 提高访问链路的带宽

假定

- 平均对象长度 = 100k比特
- 机构网络内浏览器到初始服务器的平均访问速率= 15个/秒，通过浏览器发送的平均数据率=15个/秒*100k比特=1.5Mbps
- 接入路由器到任何初始服务器的平均时延= 2秒，即因特网时延

结果

- 访问链路流量强度=~~99%~~ 9.9%
- 局域网流量强度 = 15个/秒*100k比特/1Gbps=0.14%
- 总时延 = 因特网时延 + 访问链路时延 + LAN时延 = 2秒 + ~~分钟级~~ + 微秒级 → 毫秒级



但升级访问链路带宽通常费用可观

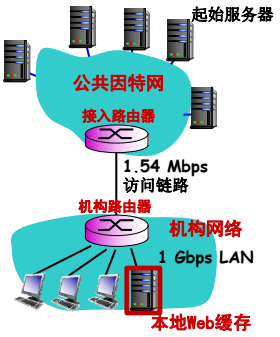
例子： 安装web缓存

假定

- 平均对象长度 = 100k比特
- 机构网络内浏览器到初始服务器的平均访问速率= 15个/秒，通过浏览器发送的平均数据率=15个/秒*100k比特=1.5Mbps
- 接入路由器到任何初始服务器的平均时延= 2秒，即因特网时延

结果

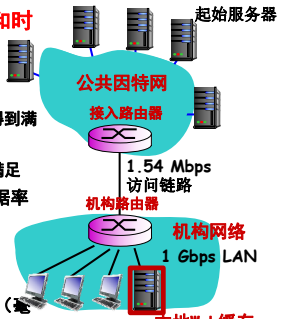
- 局域网流量强度= ？
- 访问链路流量强度= ？
- 如何计算流量强度和时延呢？
- 安装Web缓存性价比高！



例子： 安装web缓存

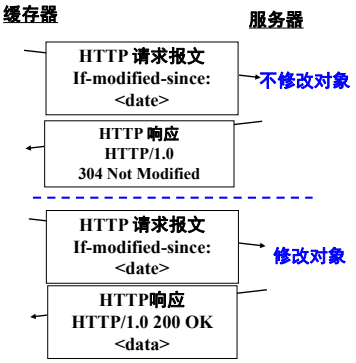
计算安装了Web缓存后访问链路流量和时延：

- 假设Web缓存命中率为40%
  - 40%的客户端请求几乎能在Web缓存立即得到满足
  - 60%的请求经过访问链路，由初始服务器满足
- 通过浏览器发送，经过访问链路的平均数据率 1.5Mbps*0.6=0.9Mbps
  - 访问链路流量强度=0.9/1.54=58%
  - 时延约为几十毫秒，可忽略不计
- 总平均时延= 因特网时延 + 访问链路时延 （毫秒级可忽略不计）+ LAN时延（微秒级，可忽略不计） = 0 .6*(2.01秒) + 0.4*0.010秒



条件GET方法

- 目标:如果缓存中有最新缓存版本，就不发送该对象
- Web缓存: 在HTTP请求`If-modified-since: <date>`中，指定缓存版本的日期
- 服务器：如果缓存的副本是最新的，响应报文中不包含对象: `HTTP/1.0 304 Not Modified`



第二章:应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

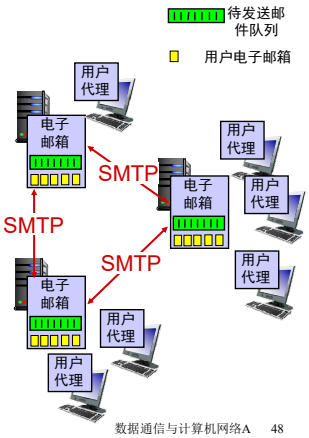
2.3 电子邮件

三个主要部分:

- 用户代理UA
- 邮件服务器Mail Server
- 简单邮件传输协议: SMTP

用户代理

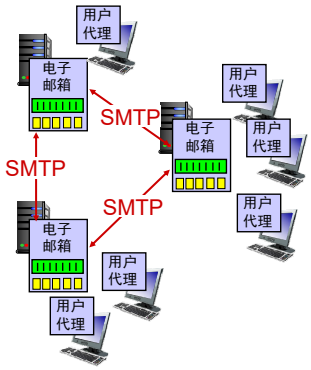
- 亦称为 “邮件阅读器”
- 写作、编辑、阅读邮件报文
- 例如:Outlook, firefox
- 到达和即将发送的邮件均存储于服务器



2.3 电子邮件

邮件服务器

- 维护用户邮箱，用户邮箱存储该邮箱的用户代理发来的报文
- 维护待发送的邮件报文队列
- 邮件服务器之间的使用SMTP协议发送电子邮件

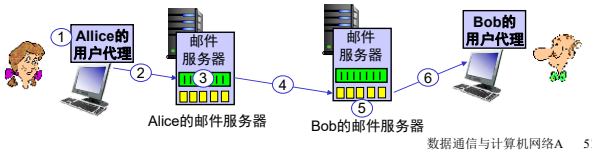


2.3.1 电子邮件: SMTP [RFC 2821]

- 使用TCP可靠地传输电子邮件报文，端口25
- 直接传输：从发送邮件服务器→接收邮件服务器
- 传输的三个阶段
  - 握手 (欢迎)
  - 报文的传输
  - 关闭
- 命令/响应的交互方式
  - 命令: ASCII文本
  - 响应: 状态码和短语
- 报文必须是7比特ASCII格式

场景: Alice 向 Bob发送报文

1) Alice利用用户代理写邮件，邮件接收者是 bob@someschool.edu
2) Alice的用户代理使用SMTP协议向Ailice邮件服务器发送件报文；报文在待发送报文队列中排队
3) Alice的邮件服务器建立与Bob的邮件服务器的TCP连接
4) Alice的邮件服务器使用SMTP协议，利用TCP连接发送Alice的报文
5) Bob的邮件服务器将Alice的报文放入Bob邮箱
6) Bob的用户代理通过POP3协议或IMAP协议，从Bob邮件服务器获取Alice发来的邮件，并阅读。



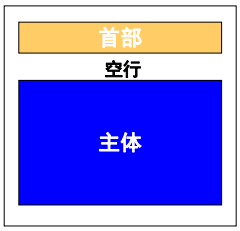
2.3.2 简单的SMTP交互实例（RFC 821)

```
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 <CRLF>.<CRLF>
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```


2.3.3 邮件报文格式

邮件内容的格式在RFC822文档中定义

- 必须包括的首部行：
  - To:用于指定一个或多个收件人的邮件地址，邮件地址格式为：用户名@邮箱服务器域名，如username@example.com。
  - From:邮件系统自动填入，指定发件人的邮件地址。
- 可选的常见首部行：
  - Subject:指定邮件的主题
  - Date：指定邮件的发送时间
  - cc ：指定邮件的抄送地址。
  - bcc ：指定邮件的暗送地址
- 主体为ACSII格式表示的邮件内容，由用户自由撰写。



2.3.3 邮件报文格式

首部行

1．Return-Path: <it315_test@sina.com>
2．Delivered-To: it315_test@mx72.mail.sohu.com
3．Received: from smtp.sina.com.cn (unknown [202.108.3.177]) by sohumx139.sohu.com (Postfix) with SMTP id E4F9802C1249 for <it315_test@sohu.com>; Thu, 10 Nov 2005 16:39:50 +0800 (CST)
4．Received: (qmail 49221 invoked from network); 10 Nov 2005 08:39: 33 -0000
5．Received: from unknown (HELO it315_test) (218.246.5.151) by smtp.sina.com.cn with SMTP; 10 Nov 2005 08:39:33 -0000
6．From: it315_test@sina.com
7．To: it315_test@sohu.com
8．subject:test
9．Message-Id: <20051110083950.E4F9802C1249@sohumx139.sohu.com>
10．Date: Thu, 10 Nov 2005 16:39:50 +0800 (CST)
11．Status: RO
12．X-UIDL: 1131611863.21509_77.mx72
13．

主体 → 14．test!!!

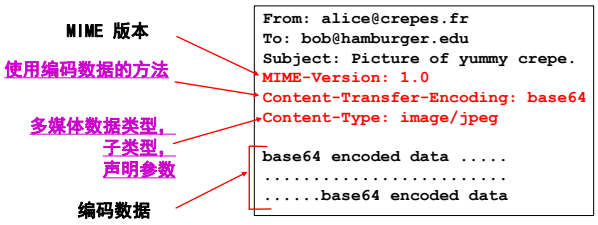
- Received 字段的基本格式为 Received from A by B for C 其中A为发送方，B为接收方，C为收件人的邮箱地址。
- 该字段的内容由接收邮件的SMTP服务器填写，常常被用来追踪邮件传输的路线和分析邮件的来源，
- 这封邮件的传输路径：从IP地址为【218.246.5.151】的机器发出→【smtp.sina.com.cn】→【sohumx139.sohu.com】→【it315_test@sohu.com】。
- 第4行是sina的SMTP服务器内部调用的一个邮件发送模块添加的，它说明sina的SMTP服务器接收到邮件后，再通过这个邮件发送模块将邮件转发出去。

2.3.3 邮件报文格式: MIME

- MIME: 多用途因特网邮件扩展协议（Multipurpose Internet Mail Extensions）, RFC 2045, 2056
- 是一种使电子邮件除了包含一般的纯文本以外，还可携带图片、声音和视频等二进制文件的协议。

2.3.3 邮件报文格式: MIME

- 在报文首部的附加行声明MIME内容类型



2.3.3 邮件报文格式: 多媒体扩展

- Content-Transfer-Encoding字段：
  - 说明传送时是如何对邮件主体进行编码
  - RFC1521定义了五种编码方式，对于RFC821，只有前3种有效：
    - 默认的NVT格式7位ASCII字符
    - 64基本字符编码(Base 64 encoding)
    - 引用的可打印编码(Quoted-Printable encoding)
    - 8位ASCII字符，包括字符行，某些为非ASCII字符且第8bit置1
    - binary编码

2.3.3 邮件报文格式: 多媒体扩展

- Content-Transfer-Encoding字段：
  - 64基本字符编码(Base 64 encoding)

对于任意的二进制文件，可用 base64 编码。这种编码方法是先把二进制代码划分为一个个 24 位长的单元，然后把每一个 24 位单元划分为 4 个 6 位组。每一个 6 位组按以下方法转换成 ASCII 码。6 位的二进制代码共有 64 种不同的值，从 0 到 63。用 A 表示 0，用 B 表示 1，等等。26 个大写字母排列完毕后，接下去再排 26 个小写字母，再后面是 10 个数字，最后用“+”表示 62，而用“/”表示 63。再用两个连在一起的等号“==”和一个等号“=”分别表示最后一组的代码只有 8 位或 16 位。回车和换行都忽略，它们可在任何地方插入。

下面是一个 base64 编码的例子：

24 位二进制代码	01001001 00110001 01111001			
划分为 4 个 6 位组	010010	010011	000101	111001
对应的 base64 编码	S	T	F	5
用 ASCII 编码发送	01010011 01010100 01000110 00110101			

不难看出，24 位的二进制代码采用 base64 编码后变成了 32 位，开销为 25%。

2.3.3 邮件报文格式: 多媒体扩展

- Content-Transfer-Encoding字段：
  - 引用的可打印编码(Quoted-Printable encoding)

另一种编码称为 quoted-printable，这种编码方法适用于所传送的数据中只有少量的非 ASCII 码，例如汉字。这种编码方法的要点就是对于所有可打印的 ASCII 码，除特殊字符等号“=”外，都不改变。等号“=”和不可打印的 ASCII 码以及非 ASCII 码的数据的编码方法是：先将每个字节的二进制代码用两个十六进制数字表示，然后在前面再加上一个等号“=”。例如，汉字的“系统”的二进制编码是：11001111 10110101 11001101 10110011（共有 32 位，但这四个字节都不是 ASCII 码），其十六进制数字表示为：CFB5CDB3。用 quoted-printable 编码表示为：=CF=B5=CD=B3，这 12 个字符都是可打印的 ASCII 字符，它们的二进制编码[0]需要 96 位，和原来的 32 位相比，开销达 200%。而等号“=”的二进制代码为 00111101，即十六进制的 3D，因此等号“=”的 quoted-printable 编码为“=3D”。

2.3.3 邮件报文格式: 多媒体扩展

- Content-Type字段：指明邮件内容类型，默认为无格式ASSCII文本

内容类型	子类型	描述
text	plain richtext enriched	无格式文本 简单格式文本，如粗体、斜体、下划线等 richtext的简化和改进
multipart	mixed parallel digest altenative	多个正文部分，串行处理 多个正文部分，可并行处理 一个电子邮件的摘要 多个正文部分，具有相同的语义内容
message	rfc822 partial external-body	内容是另一个RFC822邮件报文 内容是一个邮件报文的片段 内容是指向实际报文的指针
application	octet-stream postscript	任意二进制文件 一个Postscript程序
image	jpeg gif	ISO 10918格式的图像 CompusServer的图形交换格式的图像
audio	basic	用8bit ISDN baoμ律格式编码的音频
video	mpeg	ISO 11172格式编码的视频

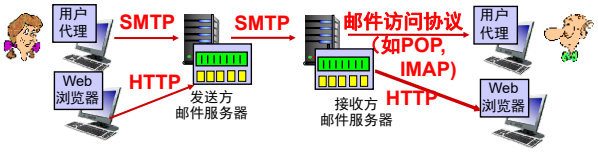
2.3.3 邮件报文格式: 多媒体扩展

- MIME协议实例

```
From: "Tang Li Yong" <tly@infosec.pku.edu.cn>
To: <wangzhao@jbu.com.cn>
Subject: SG  0.2发布
Date: Thu, 21 Nov 2002 20:56:46 +0800
MIME-Version: 1.0
Content-Type: multipart/mixed;
              boundary="----=_NextPart_000_00E6_01C291A0.80B8BA20"

This is a multi-part message in MIME format.

------=_NextPart_000_00E6_01C291A0.80B8BA20
Content-Type: text/plain;
              charset="gb2312"
Content-Transfer-Encoding: 8bit
主要更新：
1. Flash文件系统支持
2. bootloader
With best regards,

--Tang Li Yong
```

2.3.4 邮件访问协议

- HTTP: Hotmail , Yahoo! Mail等



- SMTP: 发送/存储邮件到接收方邮件服务器
- 邮件访问协议：从接收方邮件服务器获取邮件
  - POP: 邮局协议 [RFC 1939]
    - 授权 (代理 <-->服务器) 并下载
  - IMAP: 互联网邮件访问协议 [RFC 1730]
    - 更多特色 (更复杂)
    - 操作存储在服务器上的报文

POP3协议（RFC 1939）

特许阶段

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

事务阶段, 客户:

- list: 列出报文成员
- retr: 由数字获取报文
- dele: 删除
- quit

```
S: +OK POP3 服务器 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 服务器 signing off
```


POP3 和 IMAP

POP3其他情况

- 前面的例子使用了 “下载并删除” 模式
- 如果Bob改变客户机，则不能重读电子邮件
- “下载并保留”：在不同客户机上能重复获取报文

IMAP

- 在一个地方保持所有报文：服务器
- 允许用户在文件夹中组织报文
- IMAP跨越会话保持用户状态：
  - 文件夹名和报文ID和文件夹名之间的映射

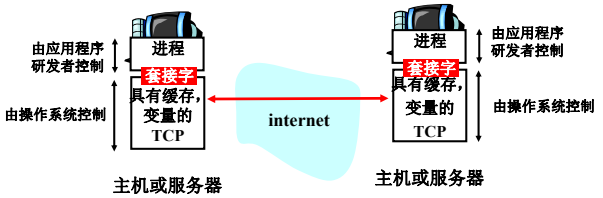
第二章: 应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

使用TCP套接字编程

目标: 学习如何构建使用套接字通信的客户机/服务器应用程序

套接字: 应用程序进程和端到端运输协议(UCP 或TCP)之间的接口



套接字编程

套接字 API

- 在 BSD4.1 UNIX引入, 1981
- 由应用程序显式产生、使用和释放
- 通过套接字 API 提供两类运输服务:
  - UDP：不可靠数据报
  - TCP：可靠, 面向字节流

套接字编程

例子 客户机-服务器 app:

1) 客户机从标准输入 (`inFromUser` stream)读入一行字符，经套接字 (`outToServer` stream) 发送给服务器

2) 服务器从套接字读该行

3) 服务器将字符全部转换成大写, 向客户机发送

4) 客户机从套接字 (`inFromServer` stream)读出并打印已被修改的行

TCP套接字编程

客户机 必须联系服务器

- 服务器进程必须预先运行
- 服务器必须已经生成 套接字(门)，以欢迎客户机的联系

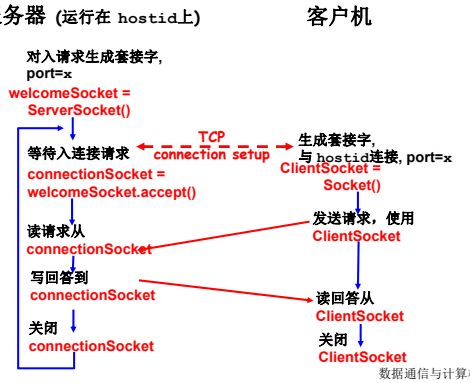
客户机联系服务器，通过:

- 创建客户机本地TCP 套接字
- 定义服务器进程的IP地址，端口号
- 当客户机 产生套接字时：客户机TCP创建到服务器 TCP 的连接
- 当客户机联系时, 服务器 TCP 为服务器进程生成新的套接字，以与 客户机通信
  - 允许服务器与多个 客户端交谈
  - 源端口号用于区分不同客户端

从应用程序观点看

TCP在客户机和服务器之间提供可靠的、按序的字节传输(管道)

客户机/服务器套接字交互: TCP



例子: Python客户机 (TCP)

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

create TCP socket for server, remote port 12000

No need to attach server name, port

例子: Python服务器 (TCP)

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

第二章：应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

*UDP*套接字编程

UDP: 在客户机 and 服务器之间无“连接”

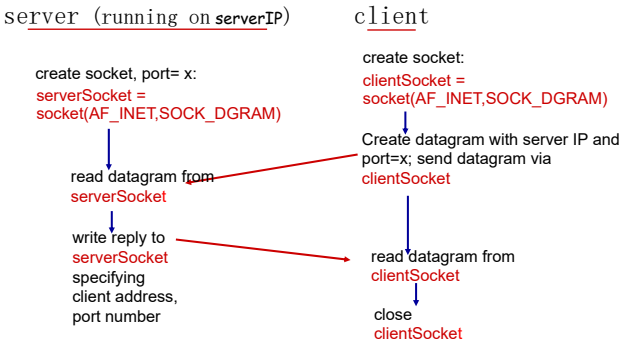
- 没有握手
- 发送方为每个分组附加上目的地的IP地址和端口号
- 服务器必须从接收到的分组提取IP地址，端口号

UDP: 接收到的传输数据可能失序或丢失

应用程序观点

UDP为客户机和服务器间的字节组(数据报)提供不可靠的传输

客户机/服务器 套接字交互: UDP



例子: Python UDP 客户端

Python UDP Client

include Python's socket library
```
from socket import *
serverName = 'hostname'
serverPort = 12000
```
create UDP socket for server
```
clientSocket = socket(socket.AF_INET,
                      socket.SOCK_DGRAM)
```
get user keyboard input
```
message = raw_input('Input lowercase sentence:')
```
Attach server name, port to message; send into socket
```
clientSocket.sendto(message,(serverName, serverPort))
```
read reply characters from socket into string
```
modifiedMessage, serverAddress =
                      clientSocket.recvfrom(2048)
```
print out received string and close socket
```
print modifiedMessage
clientSocket.close()
```


例子: python服务器 (UDP)

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

create UDP socket
bind socket to local port number 12000
loop forever
Read from UDP socket into message, getting client's address (client IP and port)
send upper case string back to this client

第二章：应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

DNS: 域名系统

人: 许多标识符

- 社会保障卡号, 名字, 护照等

因特网主机、路由器:

- IP地址(32 bit) ：用于数据报寻址
- “域名”, 如www.yahoo.com - 由人所使用

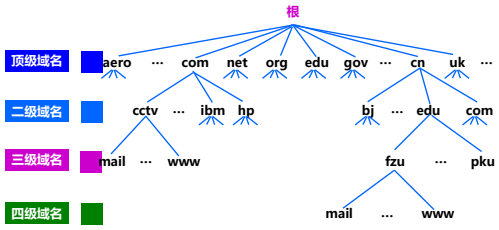
问题: IP地址和域名之间的映射?

域名系统:

- 分布式数据库 由层次化的许多域名服务器实现
- 应用层协议： 主机、路由器与域名服务器通信以解析域名 (即IP地址/域名转换)
  - 注意：因特网核心功能，作为应用层协议实现
- 复杂性位于网络“边缘”

DNS域名结构

- 因特网的域名采用了层次树状结构的命名方法
- 例如域名：www.fzu.edu.cn

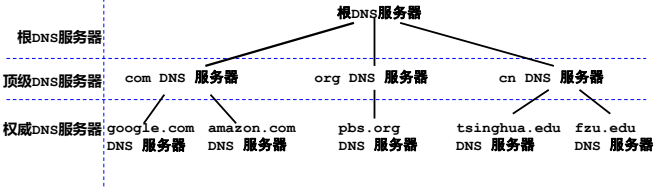


DNS域名结构——常见顶级域名

- 国家顶级域名：.cn 表示中国，.jp表示日本，.uk 表示英国等等。
- .com（公司和企业）
- .net（网络服务机构
- .org（非赢利性组织）
- .edu（美国专用的教育机构）
- .gov（美国专用的政府部门）
- .mil（美国专用的军事部门）
- .int（国际组织）

- .aero （航空运输企业）
- .biz （公司和企业）
- .cat （加泰隆人的语言和文化团体）
- .coop（合作团体）
- .info （各种情况）
- .jobs （人力资源管理者）
- .mobi（移动产品与服务的用户和提供者）
- .museum （博物馆）
- .name（个人）
- .pro （有证书的专业人员）
- .travel（旅游业）

分布式、等级制数据库



客户机要求www.amazon.com 的IP地址:

- 客户机请求根服务器以发现com DNS服务器
- 客户机请求com DNS服务器以得到 amazon.com DNS服务器
- 客户机请求amazon.com DNS服务器以得到www.amazon.com的IP地址

DNS: 根名字服务器

- 当本地名字服务器不能分解名字时联系它
- 根名字服务器：
  - 如果域名-IP地址映射未知，联系权威域名服务器
  - 获得映射
  - 将结果返回本地域名服务器

世界范围的13个根名字服务器

顶级域和权威服务器

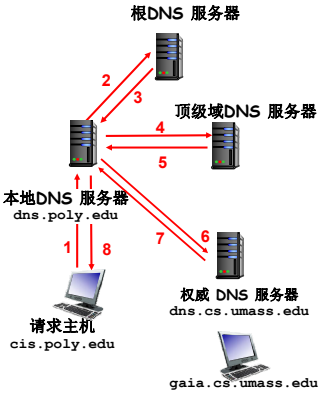
- 顶级域(TLD)服务器: 负责com, org, net, edu等，以及所有顶级国家域 uk, fr, ca, jp.例如：
  - Network Solutions维护com顶级域服务器
  - Educause维护 edu顶级域服务器
- 权威DNS服务器: 机构内部的DNS 服务器为机构内的服务器(如Web和电子邮件)提供权威的IP地址-域名映射关系
  - 由机构或服务提供商维护

本地域名服务器

- 并不严格属于DNS等级结构
- 每个ISP(住宅ISP、公司、大学)有一个
  - 也称为“默认域名服务器”
- 当主机发出DNS查询请求时，请求先被发送到其本地域名服务器
  - 本地域名服务器缓存最近解析过的域名-IP地址对(但可能会失效)
  - 作为代理，将查询请求提交到DNS等级结构中的域名解析服务器

DNS解析实例

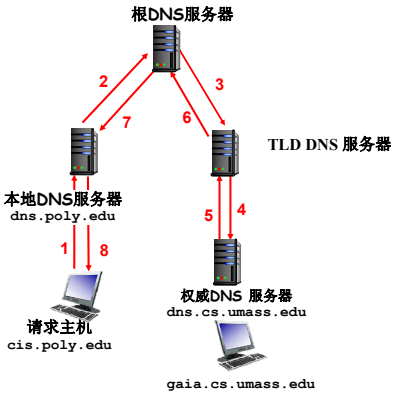
- 位于cis.poly.edu的主机请求解析gaia.cs.umass.edu的 IP地址
- 迭代查询:
  - 向域名服务器发起查询请求，查询消息包含gaia.cs.umass.edu
  - 被联系的域名服务器返回其他域名的服务器IP地址
    - “我不知道该域名，但你可以询问这个服务器.....”



递归请求

递归查询:

- 将域名解析的负担放在被请求的域名服务器
- 对于DNS体系结构中高层的域名服务器是沉重的负担?



DNS: 缓存和更新记录

- (任何)域名服务器接收到一个IP地址-域名映射关系的应答报文，都会将该映射信息缓存在本机
  - 在一定时间后，缓存项超时(失效)
  - 顶级域服务器地址通常缓存在本地域名服务器
    - 因此无需经常访问根服务器
- 更新/通知机制正由IETF设计
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS记录

DNS: 分布式数据库存储资源记录 (RR)

RR 格式: (name,type, value ,ttl)

- Type=A
  - name 是主机名
  - Value是IP地址
- Type=NS
  - name 是域 (如 foo.com)
  - Value是该域的权威域名服务器的主机名
- Type=CNAME
  - name 是别名
  - value 是规范的主机名
  - www.ibm.com type CNAME east.backup2.ibm.com
    - 表示www.ibm.com的规范主机名是east.backup2.ibm.com，该规范名指向实际的服务器
- Type=MX
  - Value是别名为name的邮件服务器的规范主机名

DNS RR资源记录示例

CNAMERR 最普遍的用法是向多台计算机或 Web 服务器使用的一个 IP 地址提供永久的 DNS 域别名，用于基于服务的名称（如 www.example.microsoft.com）的通用名称解析。下例显示了如何使用 CNAMERR 的基本语法。

alias_name IN CNAME primary_canonical_name

下面再举一例进行说明。在如下示例中，需要使用名为 host-a.example.microsoft.com 的计算机同时充当名为“www.example.microsoft.com”的 Web 服务器和名为“ftp.example.microsoft.com”的 FTP 服务器。要实现命名该计算机的预期目的，可在 example.microsoft.com 区域中添加和使用下列 CNAME 项。

```
host-a      IN    A       10.0.0.20
ftp         IN    CNAME   host-a
www         IN    CNAME   host-a
```

DNS RR资源记录示例

如果您后来决定将 FTP 服务器移至独立于 host-a 上的 Web 服务器的另一台计算机，只要为 ftp.example.microsoft.com 改变区域中的 CNAMERR 并向主持 FTP 服务器的新计算机的区域添加其他的 ARR 即可。在以上示例的基础上，如果新计算机被命名为 host-b.example.microsoft.com，则新的和修改的 A 和 CNAMERR 记录如下。

host-a	IN	A	10.0.0.20
host-b	IN	A	10.0.0.21
ftp	IN	CNAME	host-b
www	IN	CNAME	host-a

- 课外阅读：用关键字"主要资源记录类型及应用示例"在百度文库中查询

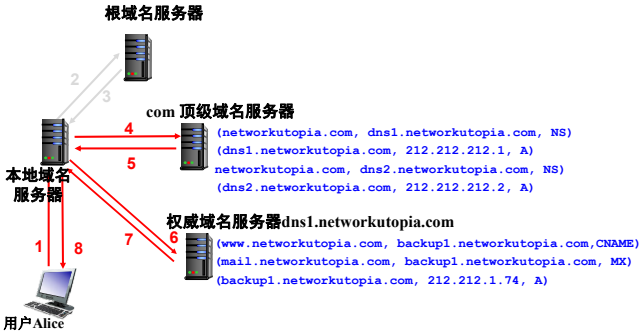
将记录插入DNS

- 例子:创建新公司Network Utopia
- 向注册登记机构 (如 ： Network Solutions)注册域名networkuptopia.com，需要提供权威域名服务器(基本和辅助的) 的名字和IP地址
  - 注册登记机构将权威域名服务器对应的RR记录插入com 顶级域名服务器:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
    (networkutopia.com, dns2.networkutopia.com, NS)
    (dns2.networkutopia.com, 212.212.212.2, A)
- 公司需要为用户提供Web和E-mail服务
  - 因此需要在权威域名服务器中插入www.networkuptopia.com的 A记录；插入mail.networkutopia.com的A记录，或者插入mail.networkutopia.com的 Type MX 记录（假设电子邮件和Web服务在同一个物理服务器上实现）。
    (www.networkutopia.com, 212.212.1.74, A)
    (mail.networkutopia.com, 212.212.1.74, A)

或

(www.networkutopia.com, backup1.networkutopia.com,CNAME)
(mail.networkutopia.com, backup1.networkutopia.com, MX)
(backup1.networkutopia.com, 212.212.1.74, A)

DNS记录举例(忽略ttl字段)



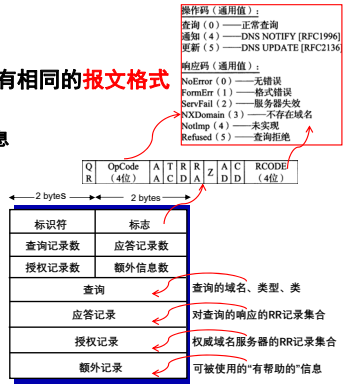
- Alice要观看www.networkutopia.com的Web页面
- 本地域名服务器如果缓存了com顶级域名服务器的IP地址，则无需请求根域名服务器

DNS 协议、报文

DNS协议：查询和应答, 都具有相同的报文格式

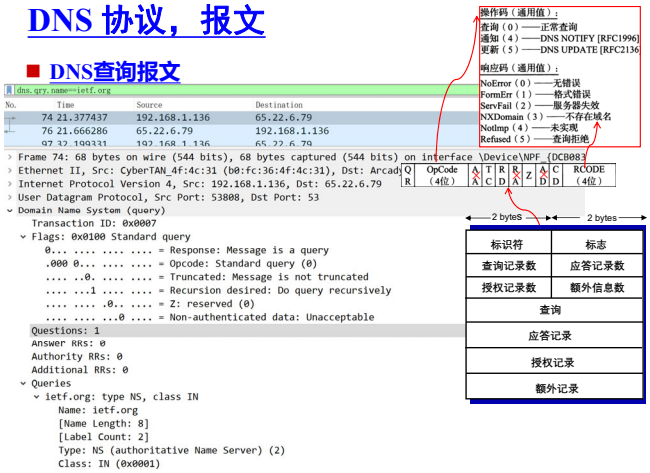
报文首部

- 标识符: 16 bit，应答与查询消息使用相同的标识符
- 标志:16 bit
  - QR:查询0或响应1
  - AA:是否授权应答
  - TC:可截断的（应答总长度超过512字节，仅返回512字节）
  - RD:要求服务器执行递归查询
  - RA:是否递归可用
  - Z:0
  - AD:真实数据（如果是授权应答，AD 置1）
  - CD:如果禁止校验，CD置1



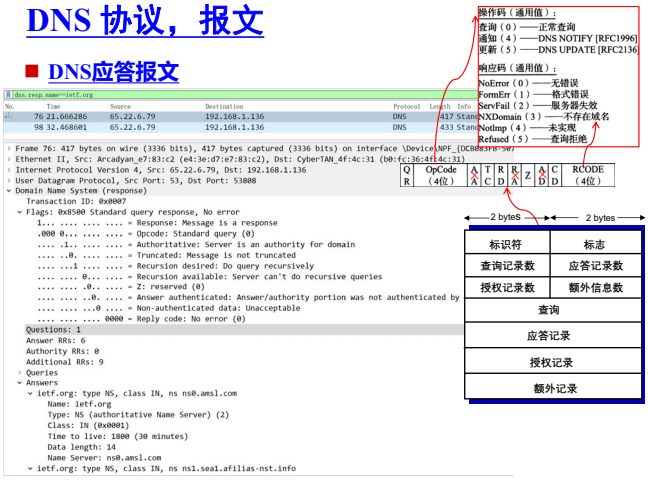
DNS 协议，报文

- DNS查询报文



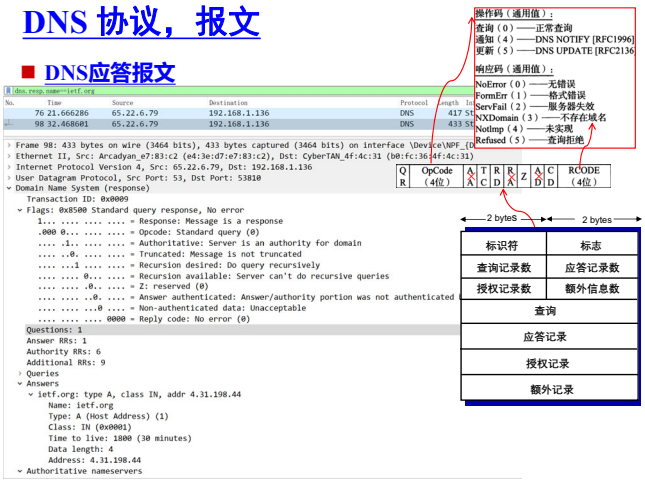
DNS 协议，报文

- DNS应答报文

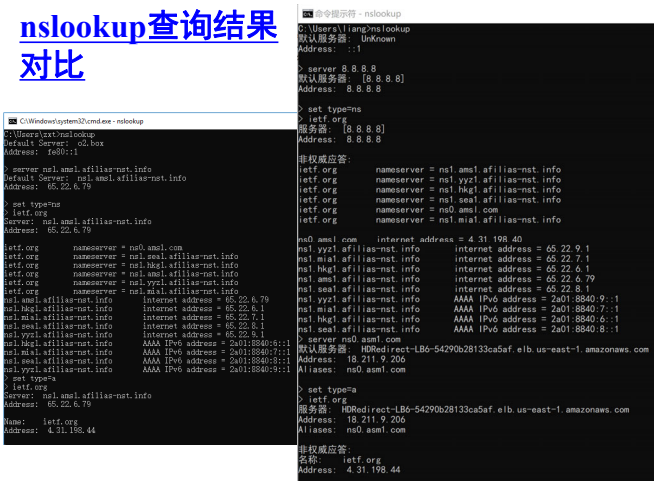


DNS 协议，报文

- DNS应答报文



nslookup查询结果对比



第二章：应用层

- 2.1 应用层协议原理
- 2.2 Web和HTTP
- 2.3 电子邮件
  - SMTP, POP3, IMAP
- 2.4 TCP套接字编程
- 2.5 UDP套接字编程
- 2.6 DNS
- 2.7 P2P文件分发

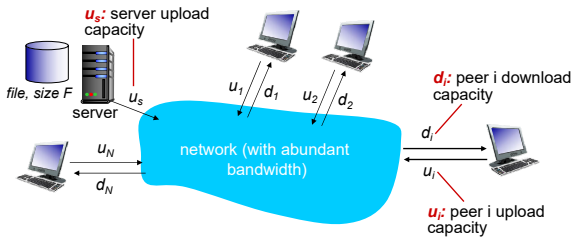
P2P 文件分发

例子

- Alice在她自己的笔记本机上运行P2P客户应用程序
- 间歇地与因特网相连；得到对每个连接的新IP地址
- 寻找歌曲 “Hey Jude”
- 应用程序显示具有Hey Jude拷贝的其他对等方
- Alice选择其中一个对等方 Bob
- 文件从Bob的PC拷贝到Alice笔记本：HTTP
- 在Alice 下载时， Alice也向其他用户上载
- Alice所有对等方既是瞬时服务器，同时也可能是客户机 = 高度可扩展

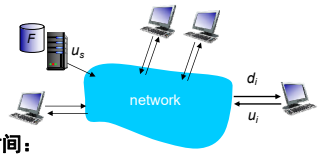
文件分发: 客户/服务器 vs P2P

思考：从一个服务器发送一个大小为 F的文件到N个客户，需要花多长时间？



文件分发: 客户/服务器

- 服务器必须向N个客户的每一个，传输该文件的一个副本：
  - 发送一个副本的时间: F/u_s
  - 发送N个副本的时间: NF/u_s
- 每一个客户端下载该文件的时间：
  - d_{min} = 具有最小下载速率的客户的下载速率
  - N个客户的分发时间至少是: F/d_{min}



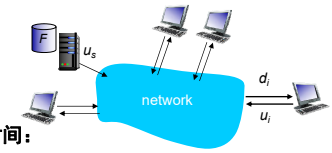
$$D_{c-s} > \max\{NF/u_s, F/d_{min}\}$$

随着客户N的数量线性增长

文件分发: P2P

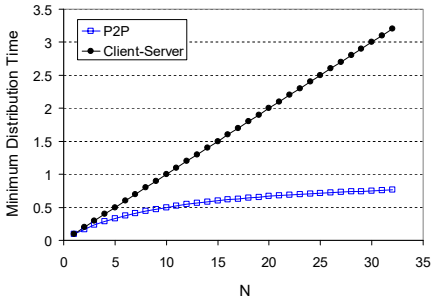
- 服务器必须经过其接入链路至少发送该文件一次
  - 发送时间: F/u_s
- 每一个对等方下载该文件的时间：
  - d_{min} = 具有最小下载速率的客户的下载速率
  - N个客户的分发时间至少是: F/d_{min}
- 系统必须向N个对等方总共交付NF比特：
  - 系统总体上载速率是 $u_s + \Sigma u_i$

$$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + Su_i)\}$$



文件分发: 客户/服务器 vs P2P

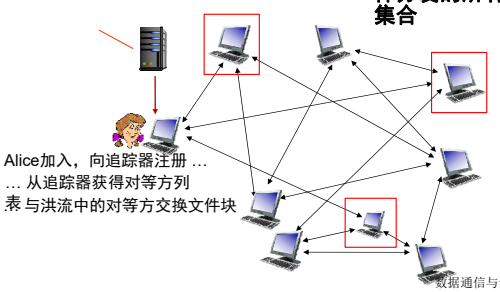
客户上载速率 = u，F/u = 1 小时，$u_s = 10u$，$d_{min} \geq u_s$



文件分发: BitTorrent

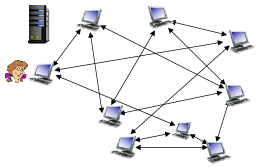
- 文件被分为 256Kb块
- 在一个洪流中，对等方彼此下载文件块

追踪器：每个洪流的基础设施节点　洪流：参与一个特定文件分发的所有对等方的集合



文件分发: BitTorrent

- 对等方首次加入洪流:
  - 没有文件块，但随着时间流逝，会从其他对等方下载越来越多的块
  - 向追踪器注册以获得对等方列表，与这个列表子集中的对等方建立连接（“邻居”）



- 当它下载时，同时也将块上载到其他对等方
- 可能会改变与之交换数据块的对等方
- 可能仅下载部分块就离开洪流，并在以后重新加入洪流
- 一旦某个对等方拥有了整个文件，它可能（自私地）离开或（利他地）留在洪流中

BitTorrent：请求和发送文件块

请求块:最优先稀缺

- 在任何给定的时间，不同的对等点都有不同的文件块子集
- Alice定期向每个邻居询问他们拥有的块列表
- Alice请求哪些在她邻居中副本最少的块

发送块：tit-for-tat

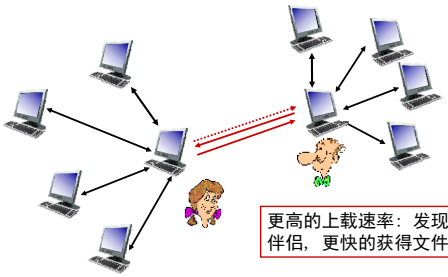
- Alice确定当前以最高速率向她提供数据的排名前4的邻居，发送数据块
  - 每10秒Alice重新计算该速率，并可能修改这4个邻居的集合
- 每30秒：随机选择另一个对等方，开始发送数据块

BitTorrent的激励机制： tit-for-tat

(1) Alice “乐观选择了” Bob

(2) Alice 成为Bob的前4位上载者之一; Bob回报Alice

(3) Bob成为Alice 的前4位上载者之一



更高的上载速率：发现更好的对换伴侣，更快的获得文件！

第2章: 小结

- 应用程序体系结构
  - 客户机-服务器
  - P2P
  - 混合
- 应用程序服务要求:
  - 可靠，带宽，时延
- 因特网传输服务模型
  - 面向连接，可靠: TCP
  - 不可靠，数据报: UDP
- 特定协议:
  - HTTP
  - SMTP, POP, IMAP
  - DNS
- 套接字编程

第2章: 小结

更为重要的是: 学习协议

- 典型的请求/回答报文交换:
  - 客户机请求信息或服务
  - 服务器用数据、状态码响应
- 报文格式:
  - 首部: 定义有关数据的信息
  - 数据: 将要通信的信息
- 控制 vs. 数据报文
  - 带内，带外
- 集中式 vs. 分散式
- 无状态 vs. 有状态
- 可靠 vs. 不可靠报文传输
- 复杂性放在网络边缘