

# 应用层

笔记源文件: [Markdown](#), [长图](#), [PDF](#), [HTML](#)

1 协议模式: 传输层服务模式, C-S模式, P2P模式

2 协议: HTTP, FTP, SMTP/pop3/imap, DNS

3 网络编程: 套接字socketAPI

## 1. 应用层协议原理

将app限制在端系统, 研发网络应用的核心在于能在不同端系统运行+通信

### 1.1. 网络APP体系结构

1 C-S结构: 比如Web/FTP/电子邮件

1. 服务器: 不间断的主机, 有永久IP
2. 客户端: 与服务器(间歇)通信, 有动态IP

2 P2P结构: 适用于流量密集的APP, 高可扩展但难管理

1. 服务器: 没有
2. 端系统: 实质上又当C又当S, 互相直连通信, 间歇式连接, IP变化

3 混合体系结构

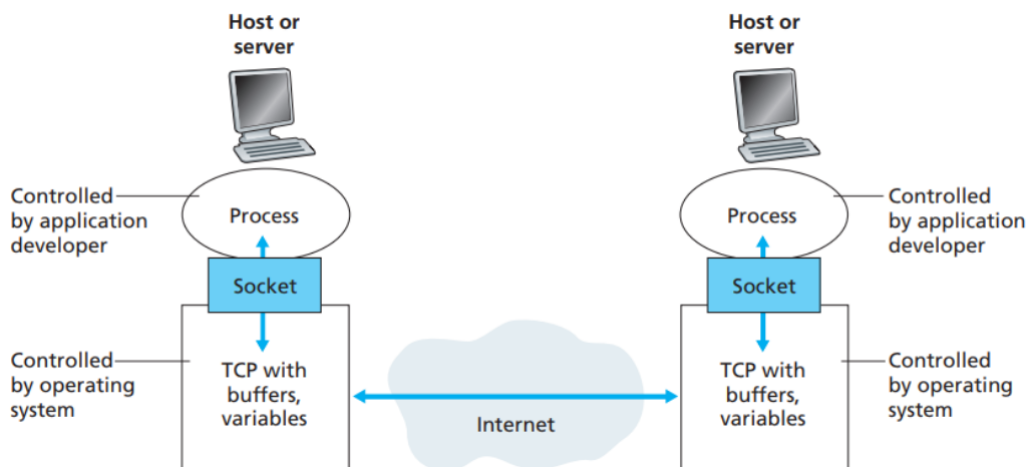
1. Skype: 一方面是基于IP的P2P应用, 但是查找远程方地址时又用C-S
2. Telegram: 两用户聊天是P2P, 但集中式服务基于C-S(用户上线请求注册IP, 查找好友IP)

### 1.2. 进程通信

1 两种进程通信: 同一主机中的两进程(通过OS内核完成), 不同主机间进程(通过消息交换完成)

2 两种进程: 客户端进程(发起通信), 服务器进程(等待被联系)

3 socket套接字: aka网络和APP间的API, 一种软件接口, 连接进程和网络(即应用层与传输层)



4 进程寻址:

1. 即进程的标识, 格式为IP/标识主机+端口号/标识进程

- 1. 邻近性的标识，格式为IP(标识主机)端口号(标识进程)
- 2. 端口：例如HTTP服务端口为80，邮件服务端口为25

### 1.3. 应用层协议概述

**1** 应用层协议定义了什么

- 1. 交换报文的类型：例如请求/响应
- 2. 消息语法：消息中有什么字段&字段如何描述
- 3. 字段语法：字段中消息的含义
- 4. 规则：进程何时/如何响应报文

**2** 种类：

- 1. 公共协议：在RFC(请求注解)中定义，例如HTTP/SMTP/BitTorrent
- 2. 专用协议：例如Skype

### 1.4. Internet的QoS(服务质量)要求

丢包率，实时性(低时延)，吞吐量，安全性

应用程序	允许丢包?	吞吐量	具有时间敏感性?
文件传输	0	弹性的	0
e-mail	0	弹性的	0
Web文档	0	弹性的	0
实时音频/视频	1	音频: 5kbps-1Mbps 视频:10kbps-5Mbps	1(100ms)
存储式音频/视频	1	音频: 5kbps-1Mbps 视频:10kbps-5Mbps	1(几秒)
互动游戏	1	超过几kbs	1(100ms)
实时发信息	0	弹性的	是或不是

### 1.5. Internet提供的传输服务

**1** TCP服务

- 1. 提供：连接管理，可靠性控制，流量控制，拥塞控制
- 2. 不提供：实时性，最小吞吐保障，安全性
- 3. TCP pro：安全套接字层(SSL)，是TCP在应用层上的强化

**2** UDP服务：只提供不可靠数据传输

**3** 流行Internet APP所采用的应用层/传输层协议

应用	应用层协议	支撑的传输层协议
email	SMTP [RFC 2821]	TCP

应用	应用层协议	支撑的传输层协议
远程终端访问	Telnet [RFC 854]	TCP
网页	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流式多媒体	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP(局域网内)
网络电话	SIP, RTP, proprietary	通常是UDP

## 2. Web和HTTP

### 2.1. Web&HTTP概述

#### 2.1.1. Web

##### 1 Web页面组成

1. 核心：基本HTML文件，包含主要内容结构+其他对象的引用
2. 其他对象：包括图像/java小程序/音频等

##### 2 每个对象都可以URL寻址：以someschool.edu域名为例

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

主机名

路径名

#### 2.1.2. HTTP(超文本传输协议)

##### 1 地位：是web在应用层的协议，用于通信

##### 2 C-S模型：客户端浏览器(请求/接收/显示Web对象)，Web服务器(响应请求/发送对象)

##### 3 HTTP的基础：TCP，过程如下

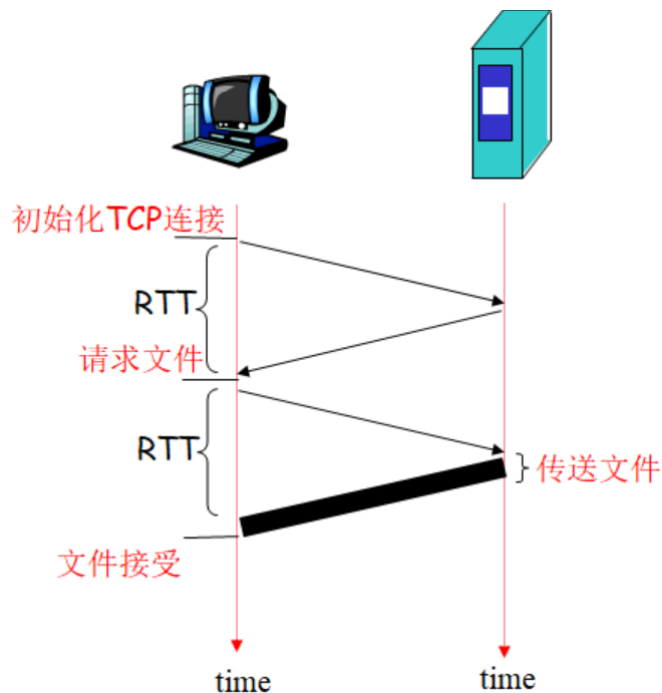
1. HTTP客户端：发起TCP连接→创建套字(端口80)→连接服务器
2. Web服务器：接收TCP连接请求
3. 交换数据，然后关闭TCP

##### 4 性质：是一个无状态的协议，服务器不保存客户端以前的请求

### 2.2. 非连续/连续HTTP

#### 2.2.1. 非连续HTTP

##### 1 含义：建立一次TCP只传送一个文件，传一个文件耗时2RTT+文件发送时间



**2** 示例：假设 `www.xjtu.edu/dxxb/home.index` 包含10张JPG+HTML文件，11个对象同在一服务器

过程为：

1. HTTP客户端：在端口80发起连接 `www.xjtu.edu` 的TCP连接
2. HTTP客户端：向服务器发出HTTP请求报文(包含路径名 `/dxxb/home.index`)
3. HTTP服务器：收到请求，从磁盘找出对象 `home.index` 并封装为响应报文，通过套接字发出
4. HTTP服务器：其服务进程通知TCP断开TCP连接
5. HTTP客户端：完整收到报文后，断开TCP连接

`home.index` 是HTML文件(包含十张图片的索引)，根据索引再重复1-5来请求图片，共11次TCP连接

**3** 弊端：传输每个对象需要2RTT，浏览器总要打开并行TCP连接来获取对象

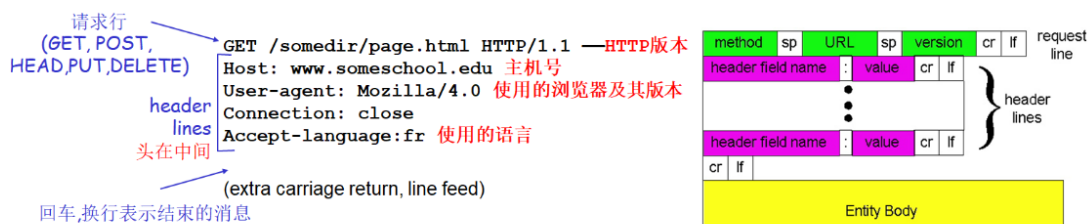
## 2.2.2. 持续的HTTP

**1** 含义：建立一次TCP可以发送多个对象，初始对象需要2RTT/后续就只要1RTT了

**2** 实现方式

1. 服务器发送响应后保持连接打开，通过打开的连接传输后续HTTP消息
2. 客户端一遇引用的对象就发送请求

## 2.3. HTTP报文格式：开始行+首部行+实体主体



分为HTTP请求/响应报文，区别在于开始行为请求行/响应状态行

## 2.4. 用户和服务器的交互：Cookie

1 cookie是什么：存储在客户端本地的小文件，用于保存身份验证/用户偏好

2 cookie的四个组件：

1. 首部行cookie：在HTTP请求/响应报文的首部行中，都会有一行cookie
2. 用户端的本地cookie：由浏览器管理，必要时发给服务器
3. Web站点后端数据库的cookie：保存用户的身份信息/偏好/使用历史

3 示例：以使用Chrome访问<https://pornhub.com/>为例

首次访问<https://pornhub.com/>网站

1. <https://pornhub.com/>的Web站点：生成唯一识别码 <xxx>，以此为索引项在后端数据库建立表项
2. <https://pornhub.com/>服务器：给Chrome响应一个HTTP报文(包含 Set-Cookie:<xxx>首部)
3. Chrome：看到返回的 Set-Cookie:<xxx>后，在特定cookie文件加一行

```
<xvideo主机名> <识别码aaa>
<missav主机名> <识别码bbb>
....新增以下....
<pornhub主机名> <识别码xxx>
```

继续访问<https://pornhub.com/>网站

1. 每请求一个Web页面，浏览器就查询Cookie文件
2. 抽取pornhub的识别码→将pornhub项加到HTTP请求报文的Cookie首部

再次访问<https://pornhub.com/>网站

1. Chrome：在请求报文中放入cookie首部行
2. <https://pornhub.com/>服务器：会记住你喜欢看的猫娘

## 2.5. Web缓存&代理服务器

### 2.5.1. Web缓存概述

1 目的：减少客户端全球时间，节省流量

2 示例：假设Chrome请求xjtu.edu/temp.gif

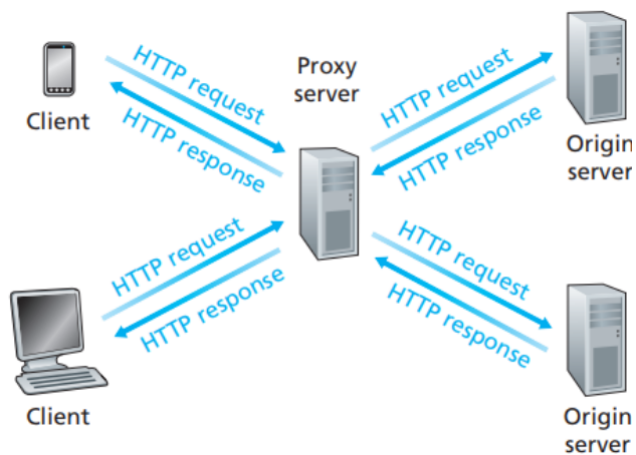
1. 浏览器：建立xjtu.edu $\xleftrightarrow{TCP}$ Web缓存器

2. Web缓存器：检查内部是否有temp.gif副本

◦ 有：将副本 $\xrightarrow[\text{返回给}]{\text{HTTP响应}}$ Chrome

◦ 没有：缓存器建立一个新的TCP连接，通过新连接获取该对象副本，再返回给Chrome

3 Web缓存类型：代理Cache，客户端/服务端Cache，分布式Cache



### 2.5.2. 代理Cache

1 目的：不涉及源服务器条件下，满足客户需求

2 工作机理

1. 用户设置浏览器：通过缓存访问Web
2. 浏览器：将所有HTTP请求发往代理缓存，缓存中有所需对象则直接访问缓存，没有的话再访问源服务器

⚠ 访问代理获得服务的速度，远快于访问源服务器

### 2.5.3. 客户端Cache

1 客户端Cache的形成：代理服务器将对象发浏览器时，浏览器会在本地缓存该对象+最后修改日期

2 何时使用本地Cache：再代理服务器中的对象被修改前，都是用本地缓存的对象

3 如何知道代理服务器的对象是否被改：条件GET方法，条件GET报文满足

1. 请求报文使用GET请求报文
2. 请求报文包含If-Modified-Since首部

### 2.5.4. 分布式Cache

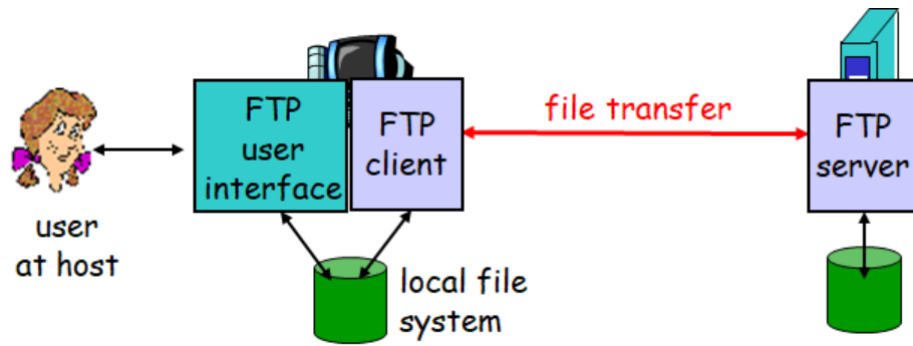
缓存间互相合作，改缓存中无访问的对象→访问其邻居，邻居都没有→才访问源服务器

### 2.5.5. 服务器Cache

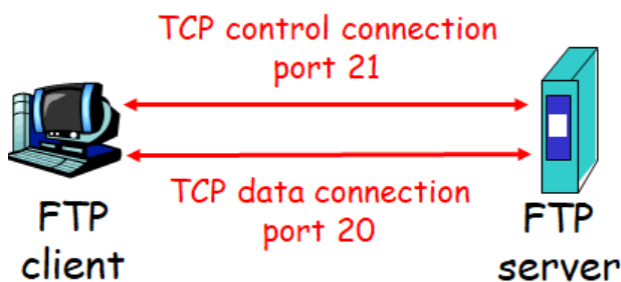
多服务器集群，内容可相同/不同，通过其中的轻载服务器(缓存)响应HTTP请求，广泛采用

## 3. Email

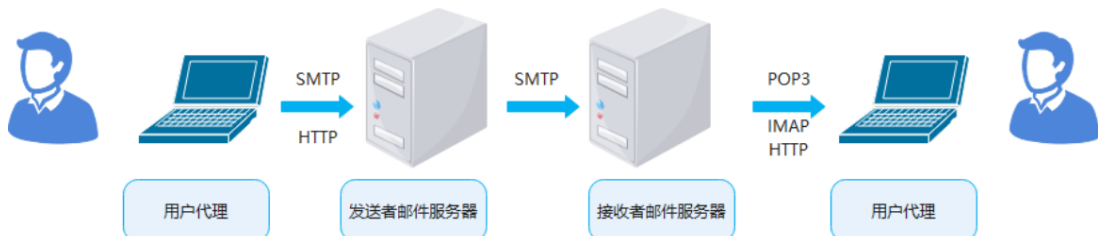
### 3.1. FTP(File Transfer Protocol)



- 1 概述：作用是向主机传文件，采用C-S模式，FTP的端口号是21(用于连接控制)
- 2 两种连接：控制连接(21, 用于传输控制命令)，数据连接(20, 传输实际的文件内容)



### 3.2. Email组成



#### 3.2.1. 用户代理

- 1 比如Outlook/Gmail/Apple-mail
- 2 供用户读/写/发邮件，将邮件发往服务器的外出报文队列/从服务器中取邮件

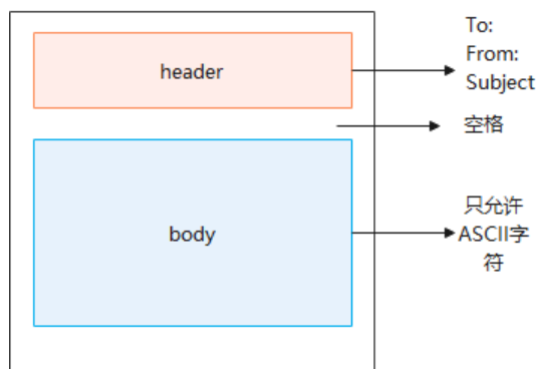
#### 3.2.2. 邮件服务器(核心)

- 1 Email一般过程：发送方代理→发送方邮件服务器→接收方邮件服务器→分发给接收方代理
- 2 Email不成功时：报文还在邮件服务器的报文队列中，多次尝试，再不济就删除并通知发送方

#### 3.2.3. SMTP(简单邮件传输协议，端口号25)

- 1 功能：客户端 $\xrightarrow[\text{TCP}]{\text{输邮件消息}}$ 服务器，发送服务器 $\xrightarrow{\text{直连}}$ 接收服务器
- 2 特点：运行在邮件服务器上，邮件内容只能是ASCII字符

### 3 SMTP报文格式



### 4 SMTP vs HTTP

1. 二者都是主机-主机传文件，都是用持续的连接
2. HTTP是拉协议(TCP连接由接收端发起)，SMTP是推协议(发送端发起)
3. HTTP一个对象就一个报文，SMTP所有对象塞一个报文里

### 5 非ASCII邮件的扩展：MIME(多用途互联网邮件扩展)

## 3.3. 邮件访问协议：服务器→代理

- 1 POP3(端口110): 第三方邮局协议，内容为授权+下载，是无状态的
- 2 IMAP(端口143): 邮件访问协议，保持用户跨会话的状态(有状态的)
- 3 HTTP: 如Gmail

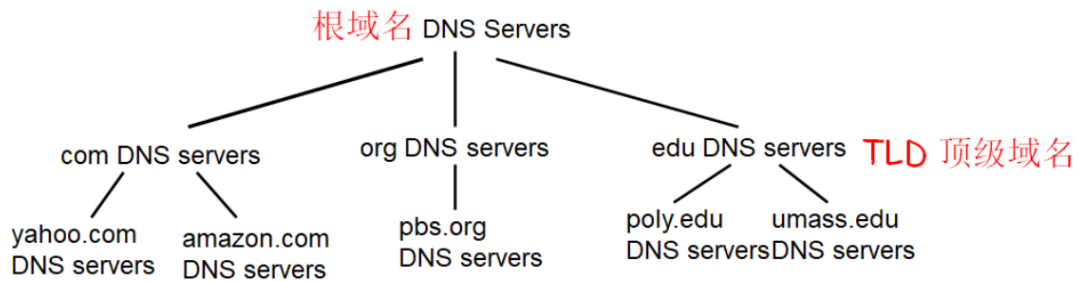
## 4. DNS

### 4.1. 概述

- 1 背景：主机除可用IP&域名标识，如我的域名xjtu.co.uk也是76.223.105.230，主机<sup>多对多</sup>←→IP
- 2 DNS什么是：由分层DNS服务器实现的分布式数据库，也是供主机查询分布式数据库的应用层协议
- 3 DNS运行在UDP上，端口53
- 4 DNS提供什么服务？
  1. 主机(别)名—IP的转换
  2. 邮件服务器的别名，如DNS会将发给danni.hiroaki@ieee.org的邮件指向IEEE的邮件服务器
  3. 负载分配：
    - 例如ieee.org运行在多个服务器上，对应多个IP，则DNS数据库会存储ieee.org的所有IP
    - 客户端请求ieee.org时，DNS按一定顺序返回其所有IP，客户端选择最前排的IP响应



## 4.2. 分布式分层数据库



1 根域名服务器：管理所有顶级域名，不直接将域名转为IP，而是告诉本地域名服务器该去找哪个顶级域名服务器

2 顶级域名服务器

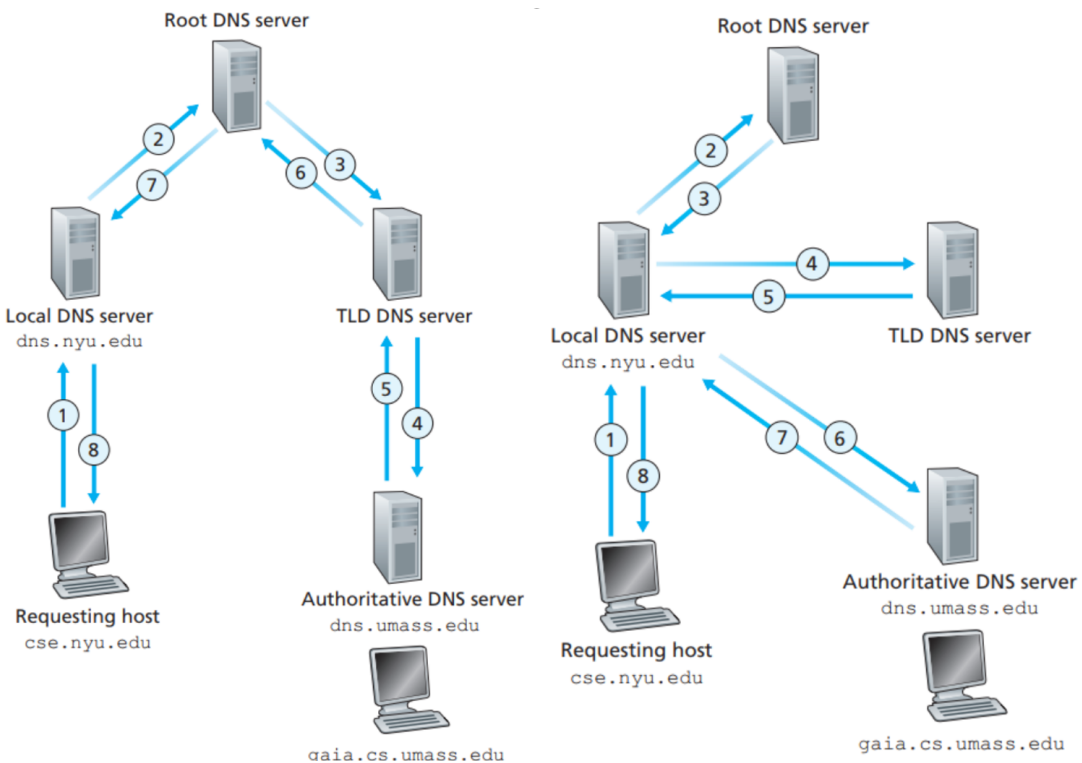
3 注册DNS服务器：不欠费就一直会记住你的域名，负责一个区

4 本地域名服务器：不在图中，一般和主机在同一个局域网

## 4.3. DNS域名解析

1 查找过程：主机先递归查询本地域名服务器，没查到则再以DNS客户身份迭代查询其他域名服务器

2 两种查询方式：以在纽约大学查询Massachusetts大学为例



1. 递归式(深度优先)：左边，本地域名服务器只向根域名服务器查询一次

2. 迭代式(广度优先)：右边，根域名服务器收到本地域名服务器请求，返回IP/下一步要查哪个服务器

3 补充：

1. 由于底层及DNS服务器缓存的存在，跟服务器基本被绕开了

2. 实际上经常两种查询方式一起使用

## 4.4. DNS记录和报文

❶ RR：即资源记录，存储在DNS服务器，格式为 `<Name, Value, Type, TTL>`，具体含义见下

1. TTL：记录存在的生命周期
2. 其他：见下表，注意ieee.org的邮件其实使用的是Gmail，这点在MX里体现

Type	Name(实例)	Value(实例)	说明
A	主机名 (xjtu.co.uk)	IP(76.223.105.230)	主机名到IP的映射
NS	域名	可获得该域IP的DNS主机	用于DNS查询
CNAME	主机名	规范化的主机名	查找规范主机名
MX	邮送域 (ieee.org)	邮件服务器的规范主机名 (gmail.com)	

❷ DNS协议报文：注意前12字节是报头

标识符	标志	} 12个字节
问题数	回答RR数	
authority RR数	附加RR数	
问题 (问题的变量数)		} 查询的名字和类型字段
回答 (资源记录的变量数)		} 响应查询的RR
authority (资源记录的变量数)		} 注册服务器的记录
附加信息 (资源记录的变量数)		} 可被使用的附加有用信息

## 5. 搜索引擎

❶ www可以视为图：页面是结点/URL是边，URL可以让一个页面跳转到另一个页面，好比两点用边相连

❷ www中页面的索引

1. 关键字：URL
2. 数据结构：线性表(存储URL指针+页面指针)，堆(存储可变长的标题+URL)，HASH表(避免重复访问)

❸ 索引创建过程

1. 广度优先搜索：输入/散列/尝试放入URL，若URL已在表中则处理下个URL，不在的话就访问URL并加入表中
2. 对表中每个URL，提取页面/标题的关键词

## 6. Socket编程

---

- 1 服务器：要有一个Socket，通过它来收发段
- 2 客户端：要有一套接字(在本地由端口号识别)，客户端需要知道服务器的IP+Socket端口号