**自顶向下的计算机网络：**

# 应用层

## 应用层协议原理

### 网络应用体系结构

网络的体系结构是固定的，但是网络应用的体系结构可以由程序研发者设计，有一些常用的主流架构：C/S架构，B/S架构和P2P架构。

**C/S架构（client service architecture）**：有一个总是打开的主机作为服务器，服务来自其他主机作为客户的请求。

当然一个主机可能跟不上所有客户的请求，为此需要配备大量的**数据中心（data center）**来创建虚拟服务器。也就是分布式服务

图示

描述已自动生成

**B/S架构（browser service architecture）**: 因为使用浏览器作为服务器，业务逻辑和数据操作都集中在服务器端。

图示

中度可信度描述已自动生成

**P2P体系结构（P2P architecture）**并不依赖于数据中心，而是在主机之间使用直接通信，这些主机称之为**对等方**。P2P具有**自扩展性**（self-scalability），每个对等方都因请求产生工作负载，但每个对等方向其他对等方发送文件也为系统增加了服务能力。未来P2P应用于高度非集中式结构，面临着安全性、性能和可靠性的挑战。

### 进程通信

进行通信的不是**程序program**，而是**进程process**。在两个不同的端系统上的进程，跨越计算机网络交换报文message而相互通信。发送进程向网络发送报文，接受进程接受这些报文并且回送报文进行响应。

1. 客户服务器进程

假设一个进程为客户，一个进程为服务器。在一个Web中，浏览器是一个客户进程，二web服务器是服务进程；对于一个P2P传输文件，请求文件的对等方是客户进程，发送文件的对等方为服务进程。

所以对客户服务器进程**定义**如下：

在一对进程之间的通信会话场景中，发起通信的进程为客户，在会话开始前等待的是服务器。

1. 进程和计算机网络之间的接口

两个进程通信时必须通过下面的网络，进程通过**套接字（socket）**这个软件接口向网络发送报文和接受报文。**套接字也称之为API**(Application programming Interface), 即引用程序和网络之间的应用程序编程接口。因为套接字存在于应用层和运输层之间，开发者可以控制套接字在应用层端的一切，但是对其运输层部分几乎无法控制，能够控制的权限仅仅有：1. 选择运输层协议（如TCP或者UDP）。2. 设定部分运输层的参数，比如最大报文段长度和缓存大小等。

1. 进程寻址

主机进程需要向另一台主机的进程发送分组时，接受的主机需要有一个地址。为了标识接受进程，有两个信息需要定义：主机地址；目的主机中指定接收进程的标识符。

主机由IP地址(IP address)来标识；而为了标识接收进程，需要知道目的地**端口号（Port Number）**

### 可供应用程序选择的运输服务

有四个方向对应用服务进行分类：可靠数据传输，吞吐量，定时和安全性。

1. 可靠数据传输（reliable data transfer）

运输层协议能够潜在的向应用层程序提供一个进程到进程的可靠服务。发送进程将数据传入套接字就能够相信数据能够无差错的到达接收进程。

1. 吞吐量

吞吐量在这里指的是发送进程向接受进程交付比特的速率。由于其他的session会话要共享本信道带宽，可用吞吐量可能会随着时间波动。具有吞吐量要求的应用成为**带宽敏感的应用**（bandwidth- sensitive application）。而**弹性应用**（elastic application）能够根据当前信道可用的带宽来调整，或多或少的使用吞吐量。

1. 定时

运输层协议需要保证定时。有些服务为了有效性需要而对数据交付有着严格的时间限制；对于非实时应用，较低的时延总比高时延好，但对端到端的时延没有严格的限制。

1. 安全性

运输协议能够为应用程序提供一种或者多种的安全服务。

如数据在发送进程加密，在接受进程解密。除了加密还包括了数据完整性和端点鉴别等。

### 因特网提供的运输服务（TCP、UDP）

TCP：

是一种**面向连接**的服务：在应用层数据流动开始前，TCP需要让服务器和客户主机进行相互交换运输层信息，也就是通过握手提醒客户和服务器大量即将到来的分组。完成三次握手建立后**TCP连接建立在了两个进程（发送和接收）的套接字上**，TCP是**全双工连接**，双方进程都可同时进行分组的收发。报文发送结束时，**TCP连接必须被释放**。

同时TCP是可靠的数据服务：当应用程序把字节流传入套接字，它能够依靠TCP将相同的字节流交付给对方的套接字，没有字节的丢失和冗余。

TCP还有拥塞控制的特性，各个TCP公平的分享网络带宽。

**SSL的引入**：

但是在安全性上需要注意的是：无论是TCP还是UDP都没有加密机制，如果明文通过套接字传入，在网络中传输，明文可能会被嗅探或发现。所以因特网界研制出来**加强版TCP，称之为安全套接字层（Secure sockets layer，SSL）**。 除了完成TCP的功能，SSL提供加密，数据完整性和端点鉴别。SSL不是TCP或者UDP之外的第三种传输协议，**而是通过应用层端对TCP进行的强化**。在客户端和服务端引入SSL（SSL有自己的API，）然后发送进程会向SSL传递明文，发送主机中的SSL把明文加密并且把加密数据传输给网络，接收端从网络得到数据后，SSL对数据进行解密，然后SSL通过它的套接字把明文传输给接收端应用层。

UDP：

UDP是一种不提供不必要服务的轻量级运输协议，只提供最小服务。

UDP无连接并且提供的是不可靠服务。进程把一个报文发送给UDP套接字时，UDP无法保证到达，也不能保证数据到达时不是乱序的。

## 应用层协议

应用层协议定义了以下几个内容：

交换的报文类型；各报文类型的语法；字段语义；一个进程何时发送报文以及如何对报文响应。

### Web和HTTP

#### HTTP概述：

Web网页的应用层协议为**HTTP(HyperText Transfer Protocol)超文本传输协议**.该协议定义了在客户程序和服务器程序交换的报文结构以及交换的方式。

接着对Web介绍。**Web page页面**是由对象组成，一个**对象(object)**指的是一个文件，比如一个HTML文件，JPEG图片，java小程序或者视频片段。这些文件可以通过URL寻址。多数Web包含一个**HTML基本文件（base HTML file）**和几个引用对象。因为**WEB浏览器（Web browser）**实现了HTTP客户端，Web服务器（Web server）实现了服务端，其用于存储Web对象。流行的服务器有Apache和Microsoft Internet Information server微软互联网信息服务器。

HTTP定义了浏览器向Web服务器请求页面的方式，以及服务器向浏览器（客户）传送web page。当用户请求一个web页面时，浏览器向服务器发出该页面中包含对象的HTTP请求报文，服务器接收到请求，并且用这些报文进行响应。

**HTTP使用TCP作为运输协议：实现可靠数据传输，不关注下层运输状态，只管通过套接字收取和发送报文。**

**HTTP是一个无状态协议（stateless protocol）：**服务器并不会储存任何客户状态信息。

**HTTP默认端口**：80

**HTTP传送前不需要将多媒体数据编码为ASCII码。**

**顺带一提，HTTPS就是身披SSL协议的HTTP。**当HTTP使用socket把明文报文下方刀运输层，运输层之间通过TCP连接并且传输报文。而HTTPS还是使用HTTP协议进行通信，但是使用SSL/TLS建立全信道，加密数据包。它的主要目的是提供对网站服务器的身份认证，保护交换数据的隐私和完整性。

#### 持续连接和非持续连接

当客户在一个长时间范围内通信时，客户可以发出一系列的请求并且服务器需要对每个请求进行响应。这一系列的请求可以是周期性的或者不间断的，那么此时有所有的求和响应是通过一个相同的TCP连接（**持续连接persistent connection**），还是其实每一个请求响应是由不同的独立的一个TCP连接完成(**非持续连接non-persistent connection**)？

**HTTP默认情况下是持续连接**的：所有请求响应由同一个TCP连接完成。

但是HTTP连接情况是可以被配置为非持续连接的。

1. **采用非持续连接的HTTP：**

在非持续连接的HTTP上请求一个Web页面的步骤：

1. HTTP客户进程向端口号80发起一个服务器（如[www.shareBike.com](http://www.shareBike.com)）的 TCP连接请求
2. HTTP客户经过套接字向服务器发送一个请求报文，这个HTTP请求报文里包含了路径名如/customer/home.index
3. HTTP服务器进程通过套接字接收报文，从其存储器中检索出对象[www.shareBike.com/customer/home.index](http://www.shareBike.com/customer/home.index)，在一个响应报文中封装对象，通过套接字发送这个报文。
4. 当TCP确认客户完整收到报文后，HTTP服务器进程通知TCP断开连接
5. 客户此时已经收到了响应报文，从响应报文提取HTML和这个网页上对象的引用，并且此时已经断开TCP连接。
6. 对每一个对象应用都会重复前几个步骤

当浏览器收到响应报文后会向客户显示该页面，不同的浏览器可能解释该页面的方式不同（比如使用IE和谷歌），而HTTP只定义了HTTP的客户和服务器之间的通信协议。

**所以对于采用非持续连接，每一个对象的传输都有一次tcp连接的建立和释放，当一个页面有一个html基本文件和10个jpeg图片时，这个页面需要11次tcp的建立来传输。**

默认情况下，大部分浏览器会同时进行5-10个并行的TCP连接，每一条连接处理一个事务。如果此时把最大连接数改为1，则形成了串形连接。但是并行连接可以缩短响应时间。

请求并且接收到服务端文件的时间消耗为2个RTT（往返时延）+HTML文件传输时间

因为三次握手的前两次占据了一个RTT，第三次握手时客户端向服务端发送确认报文，第三次握手时可以传输数据，所以会携带请求报文内容；服务端一旦收到请求报文，在TCP连接上发送HTML文件。所以第二次的HTTP请求和响应需要一个RTT，加上HTML的传输时间。

1. **采用持续连接的HTTP**

采用非持续连接的HTTP有两个缺点：每一个请求对象都需要建立和维护一个全新的TCP连接，客户和服务器都需要分配TCP缓冲区和保持TCP变量，对于Web服务器来说这是很大的开销；同时交付一个对象需要2RTT+HTML对象传输时间

采用持续连接的HTTP：当一条连接在经过一个配置的超时间隔内仍未被使用，此时HTTP服务器断开该连接。当连接没有断开时，在相同的客户和服务器之间可以传输多个对象，是的一个完整的页面能够通过一个TCP连接完成传输。甚至，多个服务器对同一个客户的传输时，可以采用单个持续的TCP连接，对象的请求可以一个个的发出而不必等待未决定请求（流水线）的回答

**在HTTP首部中添加Keep-Alive字段，保持持续连接。**

在java开发过程中，客户端内部类的HttpURLConnection自动实现了Keep-Alive

在服务端，Java依然是将Keep-Alive抽象出来，HttpServlet、HttpServletRequest、和HttpServletResponse类自动实现了Keep-Alive。内容长度如果是响应中的一部分，则Keep-Alive会被启用。

#### HTTP报文格式

1. HTTP请求报文

文本

描述已自动生成

如上图，该HTTP报文由5行组成。HTTP请求报文的**第一行为请求行（request line）**，后继的行都叫做**首部行（header line）**

**请求行组成**由三个字段：**方法字段**（Get, Post, Pull, Delete）；**URL字段**（带有请求对象的标识，如上图的page.html）；**HTTP版本字段**

首部行：

1. Host表示了主机，这是Web高速代理缓存所需要的。
2. Connection: close， 表明了不要麻烦的使用持续连接，服务器发送请求对象后就关闭TCP
3. User-agent:在首部行中表明了用户代理，即向服务器发送请求的浏览器类型。Mozilla/5.0就是firefox浏览器。
4. Accpet-language:fr表示用户想要得到的是法语版本，否则发送默认版本。
5. HTTP响应报文：

文本, 信件

描述已自动生成

响应报文有三个部分：一个**初始状态行（status line）**，6个**首部行（header line）**，**实体体（entity-body）**。实体体是主要部分，包含了请求的对象本身。

**状态行有三个字段：协议版本，状态码，状态信息**

200 OK：请求成功

301 Moved Permanently: 重定向，请求的对象已经被转移了，将自动获取新的URL。

400 Bad request：通用差错代码，请求无法被服务器理解。

404 Not found：请求对象不在服务器上。

500 一般是服务器报错，如

505 HTTP version not supported：HTTP版本协议不被服务器支持

首部行：

Connection: close，告诉客户发送完报文后即将关闭TCP连接

Date：服务器响应时的报文日期和时间

Server：类似于请求报文中的user-agent，这里表明服务是由一台Apache Web服务器产生

Last-modified: 该对象上一次创建或者修改的时间，它对本地客户在网络缓存起（代理服务器）上的对象缓存非常重要。

Content-length: 被发送对象的字节数

Content-type: 表明被发送对象类型。

1. 一个通用的HTTP报文：

图示

低可信度描述已自动生成

Sp为space，crlf 回车换行符

当然这里除了请求行和首部行，还携带了一个实体体(entity body)

当使用Get方法时，实体体为空

用户提交表单时使用Post方法，实体体中包含的就是用户在表单字段中输入的值。

Head方法类似于Get方法，服务器收到Head时返回响应报文，但是不返回请求对象，所以Head常用于开发者的调试和跟踪。

Put允许用户上传特定的对象到Web服务器的指定路径。

Delete允许用户删除Web服务器上的对象。

#### Cookie和Session

由于HTTP是无状态协议，但是有时一个Web站点需要识别用户，可能是因为限制访问或者联系内容和用户身份（如用户偏好），为此需要使用到cookie。

Cookie有四个组件：

1. 在HTTP的响应报文中有cookie首部行
2. 在HTTP的请求报文中有cookie首部行
3. 在用户端系统中保留了一个cookie文件，并且由浏览器进行管理
4. 位于Web站点的后端数据库

使用cookie的过程：

图示

描述已自动生成

如图左边**用户端系统保留了一个cookie文件，用于存储用户访问服务器留下的cookie**，可以看到一开始用户访问过eBay这个网站，并且已经设置了**cookie识别码**为8734

此时，用户访问亚马逊Amazon网站，当用户的请求报文到达了amazon服务器，该web服务器此时产生一个cookie的唯一识别码，并且以此为索引在其后端数据库产生一个表项。接下来，amazon服务器在给客户的响应报文中，需要在首部行加上Set-cookie，例子中的识别码为1678. 用户的浏览器接收到了来自于服务器的响应报文，并且在首部行中读取到**Set-cookie**。浏览器在它管理的cookie文件中加上这一行。

在这种情况下，用户接着浏览amazon的网站，每请求一个页面，用户浏览器查询cookie文件并且抽取cookie识别码，放到HTTP请求报文中的首部行。所以Amazon服务器可以跟踪用户站点的活动，比如Amazon使用cookie来支撑购物车服务：维护用户加入购物车的列表，在最后一起结算。

当然如果一开始在表单用户填写了自己的个人信息，这些信息也会随着cookie被传输。但是大多数时候cookie都是用作**用户标识**：浏览器向服务器传输cookie首部，服务器所以认识了用户。**因此，cookie可以在无状态的HTTP协议上添加一层会话层**。

**Session会话：通过服务器来保持状态。它用于将信息临时存储在服务器上，以便跨网站的多个页面使用。**

当客户端向服务器发送请求报文，在服务器回复响应报文时，session就已经产生了。SessionId在服务器上创建，并且保存在数据库中，返回带有cookie的sessionId作为对客户端的响应。

* Session用于在服务器上更安全地存储诸如 UserID 之类的信息，在这种情况下它不能被调整。
* 它还可以将价值形式的信息从一个网页传输到另一个网页。
* 对于不支持 cookie 的浏览器，它可以用作 cookie 的替代方案，以更安全的方式存储变量。

Session和Cookie：

Session是储存信息的服务器端文件，cookie是包含用户信息的客户端文件。

Session 依赖于 Cookie， Cookie不依赖于Session。

Session在浏览器关闭、应用程序注销时结束，Cookie有过期时间设置。

Session根据需求储存信息，Cookie最多存4kb

Session以加密形式储存信息，Cookie存数据在文本文件中。

#### Web缓存（代理服务器）

**Web cache缓存==proxy server代理服务器**，初始服务器来满足HTTP请求的网络实体。

Web缓存由自己的磁盘储存空间，并且在储存空间中保留最近请求过对象的副本。

图示, 示意图

描述已自动生成

Web缓存的工作过程：

假设浏览器正在请求<http://www.someschool.edu/campus.gif>对象

1. 浏览器首先创建一个Web缓存器的TCP连接，并且向Web缓存器的对象发送一个HTTP请求
2. Web缓存器检查本地是否储存了该副本，如果有则向客户浏览器发送HTTP响应报文，返回该对象
3. 如果没有副本，则打开一个该对象的初始服务器[www.somechool.edu的TCP](http://www.somechool.edu的TCP)连接，Web缓存器发出HTTP请求该对象，服务器接收到请求发回响应报文。Web缓存器收到报文后，在本地储存副本，并且向客户浏览器发送HTTP响应报文来发送副本。

所以代理服务器（Web缓存）即使客户也是服务器，它不仅大大减少了客户请求的响应时间，并且大大降低了因特网上的Web流量，改善所有应用的性能。

代理服务器通过使用**CDN（Content Distribution Network）内容分发网络**，使得大量的流量本地化了。（这将会在2.6中更详细的讨论）

#### 条件Get方法

Web缓存减少了用户响应时间，减少了因特网上的Web流量，但是引入了一个新的问题： 代理服务器上存放的对象可能是陈旧的，而对象本身在服务器上可能已经被修改了。所以需要HTTP上的**条件Get方法（conditional Get）**来证实对象是最新的。

如之前HTTP的响应报文格式中，首部行last-modified非常重要，在这里得到了体现。

当间隔了一段时间后，用户请求之前访问过的对象，此时缓存上已经有该对象的副本。但是为了确认该对象在服务器上是否改动，代理服务器需要向目标服务器发送条件Get执行检查：它会在首部行加上**if- modified-since**，而这个字段的值是最近一次收到该对象时，响应报文中last-modified的值。如：

文本

中度可信度描述已自动生成

如果服务器上该对象没有被修改，服务器向代理服务器（缓存）发送响应报文：

文本

描述已自动生成

状态304，Not modified.

如果修改了，则发送内容中实体体需要包含更新后的对象。

### 2.2因特网中的电子邮件

因特网的电子邮件是一种异步通信媒介。它的组成成分主要有三个：**用户代理（user agent），邮件服务器（mail server）和简单邮件传输协议（Simple Mail Transfer Protocol，SMTP）**。

图示

描述已自动生成

每一个邮件的接收方在其中的某个邮件服务器上都有他的**邮箱Mailbox**。邮件发送和接收的过程是：发送方写好邮件后通过用户代理，传输到发送发的邮件服务器上，位于外出报文队列中；发送方邮件服务器通过SMTP协议传输给了接收方的邮件服务器，然后被分发到接收方的邮箱。当接收者在他的邮箱中想要读区该报文时，邮件服务器通过用户名和口令来鉴别接收方。

如果发送方的邮件服务器不能够将邮件交给接收方的服务器，该发送服务器会在一个**报文队列（message queue）**中保持报文，并且每隔30分钟再进行一次尝试；如果几天后仍然不能发送成功，则删除报文并且发送邮件通知发送方。

#### SMTP及其报文格式

**SMTP也是用运输层中的TCP协议**作为可靠数据传输服务，从发送方的邮件服务器发送给接收方的邮件服务器。

和其他应用层一样，SMTP也有客户端（发送方的邮件服务器）和服务端（接收方的邮件服务器），每一台邮件服务器实际上即是客户端也会是服务端，向别人发送邮件是客户，接收邮件是服务。

**SMTP需要把报文以7位ASCII表示**。在发送邮件前，需要将二进制多媒体数据编码为ASCII，在SMTP传输后再进行响应的解码。

**SMTP一般不使用中间的邮件服务器进行传输**。建立TCP时是两个服务器直接连接，如果一方没有开机，报文会停留在报文队列，发送端的邮件服务器会一直尝试给接收方，而不会通过中间的其他邮件服务器。

SMTP通信过程示例：

文本, 信件

描述已自动生成

**SMTP用的是持续连接**:如果发送邮件服务器有几个报文发往同一个接收邮件服务器，它可以通过同一个TCP连接发送这些所有的报文。对每个报文，该客户用一个新的MAIL FROM: crepes.fr开始,用一个独立的句点指示该邮件的结束，并且仅当所有邮件发送完后才发送QUIT。

SMTP报文的格式：

**SMTP报文是由首部和报文体组成。**

文本

中度可信度描述已自动生成

每个首部必须包含一个From：行和一个To：行，可能包含Subject行和其他可选首部行。

在报文首部后，紧接着一个空白行，然后是以7位ASCII码表示的报文。

#### SMTP和HTTP的对比：

**相同点** SMTP和HTTP协议都是用于一台主机向另一台主机传输文件：

HTTP是从Web服务器向客户浏览器传送对象；

SMTP是从发送发的邮件服务器传输报文到接收方的邮件服务器。

**不同点：**

1. HTTP是一个**拉协议(pull protocol)**,人们在Web服务器上装载信息，客户使用HTTP拉报文到浏览器，TCP连接由接受者发起。

SMTP是一个**推协议(push protocol)**,发送邮件的邮件服务器把文件推向邮件接收器。TCP连接由发送者发起。

1. SMTP需要把每个报文从首部行到实体体转换成7位的ASCII码，HTTP没有这种数据格式限制。
2. 当处理一个包含文本+图形的文档时，HTTP把每个对象封装在HTTP响应报文中，而SMTP把所有对象放在一个报文中。

#### 邮件访问协议(POP3, IMAP, HTTP)

图片包含 游戏机, 标志

描述已自动生成

当SMTP协议把发送者的邮件服务器中的邮件发送到接收方的邮件服务器中后，接收方需要运行其本地PC上的用户代理，获取位于某个ISP上邮件服务器的邮件。

所以接收方邮件服务器传输报文给接收方用户代理需要一些协议，包括**第三版的邮局协议POP3(Post office protocol-version 3), 因特网邮件访问协议(Internet Mail Access protocol) 和HTTP.**

##### POP3

POP3是一个简单协议.

用户代理打开一个连接到邮件服务器的，端口为110的TCP连接后，POP3开始工作。其三个过程由：**特许（authorization），事务处理和更新**。

**特许** :用户代理以明文向邮件服务器发送用户名和口令以鉴别用户身份。

**事务处理**：用户代理取得报文，并且可以对报文做删除标记，取消删除标记和查看统计数据

**更新**：用户代理向邮件服务器发送QUIT指令，之后邮件服务器对有删除标记的报文进行删除。

**事务处理过程**有两种方式：使用POP3的用户代理要么为用户配置“下载并删除”和“下载并保留”两种方式。

如果使用**“下载并删除”**，用户代理发出命令有list，retr和dele. 用户代理请求服务器列出所有请求报文的长度，接着从邮件服务器取回并且删除每一封邮件。但是这种方式不利于多个不同机器对邮件报文进行访问，因为一个主机访问后，邮件服务器的报文就已经被删除了。

在用户代理与邮件服务器之间的POP3会话期间，**POP3服务器保留了一些状态信息**，如**记录了哪些用户报文被标记为删除**。然而，**POP3服务器并不在POP3会话过程中携带状态信息**。

如在终端里： telnet pop.163.com 110可以尝试POP3协议

##### IMAP

**POP3协议没有给用户提供任何创建远程文件夹并为报文指派文件夹的方法，所以并不能让所有移动端的主机访问一个文件夹里的邮件报文。**而IMAP主要是为了这个而被创建。

IMAP相对来说是一个更复杂的协议。**IMAP把每个报文和一个INBOX文件夹联系。**

收件人能把邮件报文移动到一个新的用户创建的文件夹中进行阅读或者删除，IMAP协议为用户提供了创建文件夹和把邮件从一个文件夹移动到另一个的命令。用户也可以使用IMAP协议在远程文件夹中查询邮件，按照特定条件去匹配。

**另一个重要特性是：IMAP协议允许用户代理获取报文某些部分的命令。**比如一个用户代理可以只读取一个报文的首部，当用户代理和邮件服务器之间使用低带宽连接，用户可能并不想取回所有的邮件，特别是避免包含了音视频这些大邮件。

##### 基于Web的电子邮件（HTTP）

使用Web浏览器收发邮件，**用户代理就是普通的浏览器**，用户和他远程邮件服务器之间的通信就通过HTTP进行。所以在接收邮件的时候，使用的是**HTTP协议**而不是pop3或者IMAP；当需要发送邮件时，邮件报文从浏览器发送到邮件服务器，也是通过**HTTP协议**而不是SMTP。

但是邮件服务器之间的传输还是得通过SMTP。

### 2.3 DNS：因特网的目录服务

#### DNS的服务

好比人类需要多种方式标识，主机的一种标识方法是通过**主机名（host name）**。但是主机名包含的信息很少，比如[www.eurecom.fr](http://www.eurecom.fr)，只能看得到它位于法国。并且主机名是不定长的，使得路由器难以处理。所以主机也会使用I**P地址（IP address）**标识。

为了使主机名转换为IP地址，需要一个**域名系统（Domain Name Server，DNS）**。

DNS是：一个由分层的DNS协议实现的分布式数据库；一个使得主机能够查询分布式数据库的**应用层协议**（也是C/S服务模式，也使用下层端到端的通信）。**通常使用UDP，使用53号端口。**

**DNS服务器通常是运行Berkeley internet name domain的Unix服务器，它并不直接和用户打交道，而是为应用程序以及其他软件提供主机名到IP的转换。**所以DNS通常被其他协议使用，比如HTTP，SMTP，FTP。

一台主机尝试访问一个网站时，DNS的工作过程：

1. 这台主机上运行着DNS的客户端。
2. 浏览器抽出这个网站的主机名，传给DNS客户端。
3. DNS客户端向服务器发送一个包含主机名的请求报文。
4. DNS客户端收到响应报文，其中包含了网站IP地址。
5. 一旦客户端收到包含了该网站IP地址的响应报文，就可以向该IP地址的80端口的HTTP服务器发送TCP连接请求。

**如上，一个从主机名到IP地址的转换是需要额外时延的。**

DNS还有一些其他服务：

**主机别名（host aliasing）和邮件服务器别名（mail server aliasing）**：复杂主机名的主机可以有一个或者多个的别名。relay1. west-coast, enterprise.com的主机，可能还有两个别名为enter­prise. com 和 www. enterprise. Com.

这种情况下，relay1. west-coast, enterprise.为规范主机名canonical hostname，DNS可以通过主机名或者别名得到ip address。

负载分配（load distribution）：DNS也用于在冗余的服务器上进行负载分配。繁忙的站点被分布在多台服务器上，在不同的端系统的上对应了不同的IP地址。而DNS数据块储存了这些ip地址，当客户对映射到某地址集合的名字发送一个DNS请求时，服务器用整个ip地址的集合响应。但是在每个回答中循环ip地址的次序，因为客户总是使用排序在最前的ip地址。这样客户访问时是进入了同一个站点，但是其实对接的是不同的端系统。

#### 单点式和分布式的DNS以及查询方式

**DNS的一种设计是单点式的**，也就是全球的端系统都存有DNS的客户端，但是**DNS服务器只存在一个**。它需要面对全因特网的请求报文，而这种设计带来的弊端有：

1. 单点故障：一旦瘫痪，整个因特网瘫痪。
2. 通信容量：处理上亿次的DNS查询服务。
3. 远距离的集中式数据库：单个DNS服务器不可能临近所有的查询客户，如果查询跨越大片区域则需要通过低速和拥塞的链路。
4. 维护问题：单个DNS数据库如果存储了所有因特网上的IP地址，插入和更新新的地址会是问题。

分布式层次数据库：

为了处理拓展问题，DNS使用了大量的DNS服务器，大致可以分为三种：**根域名（root）服务器，顶级域名服务器(Top level domain, TLD)和权威域名服务器。**

当然其实还存在本地DNS服务器（local DNS server），它不属于服务器的层次结构。但它起着代理的作用：把主机的DNS请求转发到DNS服务器层次结构中。

图示

描述已自动生成

查询有两种方式：**递归查询（recursive query）**和**迭代查询（iterative query）。**

**递归查询**：本地DNS服务器收到主机查询请求后，向根域名服务器查询，跟域名服务器向顶级域名服务器查询，顶级域名服务器向权威DNS查询，得到信息后再传递回来。（一个个负责查询并且向前传递）

**迭代查询**：本地DNS服务器收到查询请求后，向根域名服务器查询，根域名服务器告诉本地DNS下一步向谁查询，接着本地DNS向其发送请求报文..重复这个动作直到从某一个DNS服务器收到了ip地址。（踢皮球）

#### DNS缓存

**简单来说就是类似于Web缓存（代理服务器），本地DNS服务器也能缓存来自于其他DNS服务器的响应报文信息（比如主机名到某ip地址的映射。）**

举一个例子，假定主机向本地DNS服务器查询主机名cnn. com的IP地址。

此后，假定过了几个小时，另外一台主机也向该本地DNS服务器查询相同的主机名。因为本地服务器有了缓存，该本地DNS服务器可以立即返回cnn. com的IP地址，而不必查询任何其他DNS服务器。

**本地DNS服务器也能够缓存TLD服务器的IP地址**，因而允 许本地DNS绕过查询链中的根DNS服务器。事实上，**因为缓存，除了少数DNS查询以外，根服务器被绕过了。**

因为主机的主机名和IP地址的映射不是永久的，所以 DNS服务器缓存并不是一直存在的，**通常在2天后丢弃缓存信息**。