从理论到实践，全方位认识DNS（理论篇）

5 回复 275 查看

(https://www.shiyanlou.com/user/8490) 实验楼管理员 (https://www.shiyanlou.com/vip) 2015-12-18 17:47

技术分享 (https://www.shiyanlou.com/questions/?tag=技术分享)

对于 DNS(Domain Name System)大家肯定不陌生，不就是用来将一个网站的域名转换为对应的IP吗。当我们发现可以上QQ但不能浏览网页时，我们会想到可能是域名服务器挂掉了；当我们用别人提供的hosts文件浏览到一个“不存在”的网页时，我们会了解到域名解析系统的脆弱。

然而关于DNS还有一大堆故事值得我们去倾听，去思考。

全部回答

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)

(https://www.shiyanlou.com/user/8490)

一、DNS 源起

要想访问网络上的一台计算机，我们必须要知道它的IP地址，但是这些地址（比如243.185.187.39）只是一串数字，没有规律，因此我们很难记住。并且如果一台计算机变更IP后，它必须通知所有的人。

显然，直接使用IP地址是一个愚蠢的方案。于是人们想出了一个替代的方法，即为每一台计算机起一个名字，然后建立计算机名字到地址的一个映射关系。我们访问计算机的名字，剩下的名字到地址的转换过程则由计算机自动完成。

1.1 hosts映射

早期，名字到地址的转换过程十分简单。

每台计算机保存一个hosts文件，里面列出所有计算机名字和对应的IP地址，然后定期从一个维护此文件的站点更新里面的记录。

当我们访问某个计算机名字时，先在hosts文件找到对应的IP，然后就可以建立连接。

早期的ARPANET就是这样做的，但是随着网络规模的扩大，这种方法渐渐吃不消了。

主要有以下三个原因：

- hosts文件变得非常大；
- 主机名字会冲突；
- 集中的维护站点会不堪重负（需要给几百万机器提供hosts文件，想想就可怕）。

1.2 域名系统

为了解决上面的问题，1983年Paul Mockapetris提出了域名系统（DNS, Domain Name System)，这是一种层次的、基于域的命名方案，并且用一个分布式数据库系统加以实现。

当我们需要访问一个域名（其实就是前面说的计算机的名字）时，应用程序会向DNS服务器发起一个DNS请求，DNS服务器返回该域名对应的IP地址。

通过下面三种手段解决了上面的问题：

- 用户计算机上并没有存储所有的名字到IP的映射，这样避免了hosts文件过于庞大（现在各操作系统中hosts文件默认都是空的）。
- 规定了域名的命名规则，保证主机名字不会重复。
- DNS服务器不再是单一的一台机器，而是一个层次的、合理组织的服务器集群。

这样访问一个域名的过程可以简化为下图：

2015-12-18 17:48

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)
(https://www.shiyanlou.com/user/8490)

二、DNS 协议

那么如何具体实现这个所谓的域名系统呢，要知道管理一个超大型并且不断变化的域名到IP的映射集合可不是一个简单的事，况且还要去应付成千上万的DNS查询请求。

人们最终想出了一套不错的协议，规定如何来实现这个系统，下面我们一起来看看吧。

2.1 域名空间

首先我们需要制定一套命名规则，防止域名出现重复。DNS关于域名的规则和我们生活中的快递系统类似，使用层次的地址结构。

快递系统中要给某人邮寄物品，地址可能是这样：中国、广东省、广州市、番禺区、中山西路12号 XXX。

而一个域名看起来则是这样的groups.google.com（为什么不是com.google.groups？我猜可能和老外写地址的习惯有关）。

对于Internet来说，域名层次结构的顶级（相当于国际快递地址中的国家部分）由ICANN（互联网名称与数字地址分配机构）负责管理。

目前，已经有超过250个顶级域名，每个顶级域名可以进一步划为一些子域（二级域名），这些子域可被再次划分（三级域名），依此类推。

所有这些域名可以组织成一棵树，如下图所示（图片来自Computer Networks: 7-1 ）：

2.2 域名资源记录

DNS设计之初是用来建立域名到IP地址的映射，理论上对于每一个域名我们只需要在域名服务器上保存一条记录即可。

这里的记录一般叫作域名资源记录，它是一个五元组，可以用以下格式表示：

```
Domain_name Time_to_live Class Type Value
```

其中：

- `Domain_name`：指出这条记录适用于哪个域名；
- `Time_to_live`：用来表明记录的生存周期，也就是说最多可以缓存该记录多长时间（后面会讲到缓存机制）；
- `Class`：一般总是IN；
- `Type`：记录的类型；
- `Value`：记录的值，如果是A记录，则value是一个IPv4地址。

我们看到域名资源记录有一个Type字段，用来表明记录的类型。这是为什么呢？

因为对于一个域名来说，通常并非只记录其IP地址，还可能需要一些其他种类的记录，一些常见的记录类型如下：

记录类型	含义
A	主机的IPv4地址
AAAA	主机的IPv6地址
NS	该域名所在域的权威域名服务器
MX	接受特定域名电子邮件的服务器域名
CNAME	当前域名的一个别名

关于这些域名资源记录的实例我们将在下一篇文章（实践篇）看到。

2015-12-18 17:48

2.3 域名服务器

我们知道不能只用一台域名服务器来响应所有的DNS查询，因为没有一台机器能够给全球的用户提供查询服务，计算能力、存储、带宽都不允许。只能合理组织一个域名服务器集群，使他们协同工作，共同提供域名解析服务。

接下来首先要面对的一个问题是如何合理地将所有的域名资源记录存储到不同的域名服务器上。

前面说过域名的名字空间可以组织为一棵树，这里我们可以进一步将其划分为不重叠的区域（DNS zone），针对上图的域名空间，一种可能的域名划分如下图：

然后将每个区域与多个域名服务器（其中一个是master，其他slave服务器则用来提供数据备份、加快解析速度、保证服务可用性）关联起来，称这些域名服务器为该区域的权威域名服务器(Authoritative Name Servers)，它保存两类域名资源记录：

- 该区域内所有域名的域名资源记录。
- 父区域和子区域的域名服务器对应的域名资源记录（主要是NS记录）。

这样，所有的域名资源记录都保存在多个域名服务器中，并且所有的域名服务器也组成了一个层次的索引结构，便于我们后面进行域名解析。

下面以一个简化的域名空间为例子，说明域名资源记录是如何保存在域名服务器中的，如下图a：

图中域名空间划分为A, B, C,D,E,F,G七个DNS区域，每个DNS区域都有多个权威域名服务器，这些域名服务器里面保存了许多域名解析记录。

对于上图的NDS区域E来说，它的权威域名服务器里面保存的记录如图中表格所示。

仔细观察上图你可能会发现区域A、B并没有父区域，他们之间并没有一条路径连在一起。这将导致一个很麻烦的问题，那就是区域A的权威域名服务器可能根本不知道区域B的存在。

认识到这一点后，你可能会想出一个很自然的解决方案，就是在A中记录B域名服务器的地址，同时在B中记录A的，这样它们两个就联系起来了。但是考虑到我们有超过250个顶级域名，这样做并不是很恰当。

而我们使用的域名系统则采用了一种更加聪明的方法，那就是引入根域名服务器，它保存了所有顶级区域的权威域名服务器记录。

现在通过根域名服务器，我们可以找到所有的顶级区域的权威域名服务器，然后就可以往下一级一级找下去了。

下图为全球根域名服务器的分布图，可以在这里找到。

现在为止，我们的权威域名服务器和根域名服务器其实组成了一个树，树根为根域名服务器，下面每个节点都是一个区域的权威域名服务器。

对于图a中各个DNS区域的权威域名服务器，它们组成了下面这棵树（实际中，一个权威域名服务器可能保存有多个DNS区域的记录，因此权威域名服务器之间的联系并不构成一棵树。这部分的详细内容可以参考RFC 1034: 4. NAME SERVERS。下面为了容易理解，将其简化为一棵树）：

2015-12-18 17:49

实验楼管理员 (https://www.shiyanlou.com/user/8490) (https://www.shiyanlou.com/vip)

(https://www.shiyanlou.com/user/8490)

2.4 域名解析

我们已经有了一个域名服务器集群，该集群合理地保存了域名空间和域名资源记录的对应关系。

现在我们要做的就是发送一个DNS请求给域名服务器，然后坐等它返回正确的域名资源记录，这个过程叫作域名解析。

严格来说，域名解析的过程最早要追溯到建立网络连接。因为每当连接上网络之后，计算机会自动获得一个默认的DNS服务器，当然你也可以用自己信任的DNS服务器，比如8.8.8.8（DNS服务器也有信任不信任之分，是的，实践篇会讲到），我们把这个域名服务器也叫作本地域名服务器。

接下来当我们需要知道一个域名对应的资源记录时，会向本地域名服务器发起请求，如果该域名恰好在本地域名服务器所辖属的域名区域（DNS zone）内，那么可以直接返回记录。

如果在本地域名服务器没有发现该域名的资源记录，就需要在整个域名空间搜索该域名。而整个域名空间的资源记录存储在一个分层的、树状联系的一系列域名服务器上，所以本地域名服务器首先要从根域名服务器开始往下搜索。

这里有一个问题就是本地域名服务器如何找到根域名服务器在哪里呢？其实域名服务器启动的时候，就会加载一个配置文件，里面保存了根域名服务器的NS记录（要知道根域名服务器地址一般非常稳定，不会轻易改变，并且数量很少，所以这个配置文件会很小）。

找到根域名服务器之后，就可以一级一级地往下查找啦。

仍然以我们的图a为例，现在假设区域E内的某个用户想访问math.sysu.edu.cn，那么请求的过程如下：

用语言简单描述如下：

- 用户：喂，本地域名服务器，告诉我math.sysu.edu.cn的地址；
- 本地域名服务器：哎呀，我不知道啊，不在我的辖区，容我去问问老大哥吧。root老大，能告诉我math.sysu.edu.cn的地址吗；
- 根域名服务器：忙着呢，你去问B（.cn）；
- 本地域名服务器：喂，B，告诉我math.sysu.edu.cn的地址；
- B：你去问D（.edu.cn）；
- 本地域名服务器：喂，D，告诉我math.sysu.edu.cn的地址；
- D：你去问F（sysu.edu.cn）；
- 本地域名服务器：喂，F，告诉我math.sysu.edu.cn的地址；
- F：容老衲看看，哎呀，找到了，是X.X.X.X；
- 本地域名服务器：踏破铁鞋终于找到啦，喂用户，出来啊，我找到了，是X.X.X.X

仔细想想，这和我们邮寄快递实在是如出一辙啊，假设你从美国邮东西到广州市番禺区，首先快递送到中国（不过这里没有一个类似根域名服务器的中转站而已），然后往下到广东省，接下来是广州市，再往下是番禺了。

上面的是本地域名服务器的迭代解析过程，其实也可以递归查询，这里就不说了，道理差不多。

2015-12-18 17:49

三、缓存机制

现在整个域名系统已经可以为我们提供域名解析服务了，当我们输入域名，计算机发送DNS请求，然后DNS服务器返回给我们解析的结果，一切看起来很完美。然而是不是可以更完美呢？

回顾一下平时浏览网站的情况，我们会发现两个比较有意思的结论：

- 80%的时间我们都在看那些20%的网站，这就是大名鼎鼎的80/20 Rule；
- 我们会在一个网站的不同网页之间跳转，也就是不断地访问同一个域名，类似程序访问的局部性原理。

这两条结论很容易让我们联想到缓存机制。如果我们将已经访问过的那些域名的解析结果缓存在自己的计算机上，那么下次访问的时候可以直接读取结果，不用再次重复DNS查询过程，给自己和域名服务器都节省了麻烦。

当然，这样做的一个前提是要缓存的解析结果不会频繁更改，也就是说我十分钟后解析一个域名的结果和现在解析的结果是一样的。对大多数域名来说，这都是一个不争的事实。

但是难免有一些“善变”的域名，他们可能会频繁更改自己的解析结果。为了使缓存机制适应这两类情况，我们在域名资源记录里面添加一个Time_to_live字段，表明这条记录最多可以缓存多久。

对于那些“稳如泰山”的域名，给一个比较大的值，而那些“朝三暮四”的域名，则可以给定一个小的值。

我们既然可以在本机利用缓存，那么可不可以在域名服务器上也利用缓存机制呢，答案当然是可以的。

因为对于域名服务器来说，上面的两条有意思的结论仍然有效。所以，域名服务器可以将那些访问过的域名资源记录缓存，用户再次发起请求时，可以直接返回缓存结果，不用去迭代或者递归解析。

关于DNS理论部分，更多内容还可以参考这两个文本：

- RFC 1034: Domain Names - Concepts and Facilities (https://tools.ietf.org/html/rfc1034)

四、并没有结束

上面一大堆理论，看上去有点不明所以是吧，没事，接下来会结合实践来更加清晰地认识DNS这一最基础的系统。

其实不止是DNS，还有HTTPS、TCP、UDP这些很基础的协议，都值得我们静下心去好好认识它们。

因为，写DNS之前，我以为我已经完全搞明白了它，但是写的过程发现好多地方自己根本就不知道，之前完全是停留在一个很浮夸的层面上。

所以，是时候找时间好好把这些协议过一遍，用自己的语言，从解决问题的角度，记录下这些经典协议的故事了。

文章地址：http://selfboot.cn/2015/11/05/dns_theory/

2015-12-18 17:49

HTML5 (https://www.shiyanlou.com/questions/?tag=HTML5)　信息安全 (https://www.shiyanlou.com/questions/?tag=信息安全)

GO (https://www.shiyanlou.com/questions/?tag=GO)　NoSQL (https://www.shiyanlou.com/questions/?tag=NoSQL)

Android (https://www.shiyanlou.com/questions/?tag=Android)　Ruby (https://www.shiyanlou.com/questions/?tag=Ruby)

训练营 (https://www.shiyanlou.com/questions/?tag=训练营)　Perl (https://www.shiyanlou.com/questions/?tag=Perl)

相关问题

Web测试工具指南 (https://www.shiyanlou.com/questions/4442)

使用 Docker搭建本地 Hadoop集群 (https://www.shiyanlou.com/questions/4345)

每个程序员都应该收藏的算法复杂度速查表 (https://www.shiyanlou.com/questions/4426)

Python程序员最常犯的10个错误 (https://www.shiyanlou.com/questions/4327)

30+有用的CSS代码片段 (https://www.shiyanlou.com/questions/2747)

谷歌推荐的计算机科学学习路线 (https://www.shiyanlou.com/questions/4399)

提高Linux工作效率的十大bash技巧 (https://www.shiyanlou.com/questions/4297)

10个精妙的Java编码最佳实践 (https://www.shiyanlou.com/questions/4373)

深入理解DIP、IoC、DI以及IoC容器 (https://www.shiyanlou.com/questions/4369)

20 个 CSS 高级技巧汇总 (https://www.shiyanlou.com/questions/4364)