本文翻译者：weicq2000 (2012-10-9，weicq2000@sina.com)

Network Working Group P. Mockapetris
Request for Comments: 1034 ISI
Obsoletes: RFCs 882, 883, 973 November 1987

域名 --- 概念和设施

目录

第1章　本备忘录状态

本备忘录介绍域名系统(Domain Name System, DNS)，文中忽略了许多细节，这些细节可在姊妹篇RFC“域名---实现和规范”[RFC-1035]中找到。RFC1035假设读者熟悉本备忘录中讨论的概念。

DNS功能和数据类型子集构成正式协议。正式协议包括标准查询和它们的响应，以及大多数互联网类数据格式(例如，主机地址)。

然而，域系统有意构建成可扩展的。研究者正在不断提出、实施和试验新的数据类型、查询类型、类、功能，等等。因此，尽管期盼正式协议各部分保持基本不变并作为生产业务运行，应当始终鼓励试验工作不断扩展正式协议。在这些RFCs中对试验的或作废的特性做了明显的标记，使用这类信息应当谨慎。

读者应特别注意，不要依赖本文(目前的和完成的)举例中出现的值，因为这些值的确定主要源于教学演示需要。本备忘录的分发不受限制。

第2章　序言

本RFC介绍域样式名称，它们在互联网邮件和主机地址支持中的应用，以及实现域名功能的协议和业务。

2-1　域名的历史

域系统是在互联网增长的推动下发展起来的：

- 主机名称到地址的映射，由单一文件(HOSTS.TXT)中的网络信息中心(Network

Information Center, NIC)维护，所有主机通过文件传输协议(File Transfer Protocol, FTP)获得该文件[RFC-952、RFC-953]。按此方法分发新版本消耗的总网络带宽正比于该网络中主机数的平方，即使当使用多层FTP时，在NIC主机上的出境FTP负载也是相当可观的。

主机数量的爆炸性增长预计将来也不会停止。

- 网络数量也在相应改变。构成原始ARPANET的分时主机正在被本地工作站网络替代。本地组织正在管理他们自己的名称和地址，但是必须等待NIC改变HOSTS.TXT，以便在互联网上普遍可以看到这些改变。组织们也希望在名称空间上实现某种本地结构。
- 在互联网上的应用正在变得更加复杂，正在产生对通用目的名称服务的需求。

结果产生几种有关名称空间和对它们进行管理的思路[IEN-116、RFC-799、RFC-819、RFC-830]。这些建议各不相同，但是共同的主线是等级名称空间思想，采用粗略对应组织结构的等级，使用由“.”作为标记等级层次间边界的记号的名称。使用分布式数据库和广义资源的设计在[RFC-882、RFC-883]中介绍。根据几方面实践的经验，系统最终演进为本备忘录中介绍的方案。

术语“域(domain)”或“域名(domain name)”在许多场景中使用，使用范围超出这里介绍的DNS。更一般情况，术语域名用于指具有由圆点表示的结构的名称，但是与DNS没有关系。在邮件寻址[Quarterman 86]中尤其是这样。

2-2 DNS设计目标

DNS的设计目标影响它的结构。这些目标是：

- 主要目标是一致性名称空间，这一名称空间用于指资源。为了避免特别是由编码引起的问题，名称应当不要求包括网络标识符、地址、路由，或作为该名称一部分的类似信息。
- 规模庞大和频繁更新要求数据库必须采用分布式方式维护，使用本地缓存改善性能。企图收集整个数据库的一致性复本的方法将变得越来越昂贵和困难，并且因此应当避免。同样的道理也适用于名称空间结构，以及尤其是生成和删除名称的机制；这些也应当是分布式的。
- 凡是需要在获得数据成本、更新速率和缓存精度间进行折中时，数据源应当控制此折中。
- 实现成本(诸如便利性)要求DNS通常普遍可用，不被限制在单一应用。我们应当能够使用名称来检索主机地址、邮箱数据和其他尚未确定的信息。所有与名称关联的数据用类型(type)标记，并且可以将查询限制在单一类型。
- 因为我们希望名称空间可用于不同的网络和应用，我们提供由不同协议族或管理使用相同 名称空间的能力。例如，尽管所有协议都有地址概念，协议间主机地址格式不同。DNS用类以及类型标记所有数据，因此我们可以允许并行使用不同格式类型地址的数据。
- 我们希望名称服务器业务独立于携带这些业务的通信系统。一些系统可能希望使用数据报进行查询和响应，并且仅为强调可靠性的业务(例如，数据库更新，长业务)建立虚拟电路；其他系统将排他性地使用虚拟电路。
- 系统应当能够适应广泛的主机能力。个人计算机和大型分时主机都应当能够使用此系统，尽管可能采用不同方法。

2-3 有关应用的假设

域系统的组织来源于某些假设，这些假设涉及域系统用户社区的需求和使用模式，域系统的设计致力于避免在通用数据库系统中发现的许多复杂问题。

这些假设是：

- 整个数据库的大小初期正比于使用此系统的主机数量，但是当邮箱和其他信息被添加到此域系统时，最终将发展为正比于这些主机上的用户数量。
- 系统中大多数数据变化非常缓慢(例如，邮箱绑定，主机地址)，但是系统应当能够处理变化更为迅速的子集(在秒或分钟等级上)。
- 用于分配数据库职责的管理边界通常对应有一台或多台主机的组织。每个对特定一组域有责任的组织将提供冗余的名称服务器，或者在组织自己的主机上提供，或者在由该组织安排使用的其他主机上提供。
- 域系统的客户端应当能够识别值得信任的名称服务器集合，在接受转介到这个“值得信任的”集合之外的名称服务器前，客户端优先使用这些名称服务器。
- 获取信息比即时更新或一致性担保更为关键。因此，更新处理使更新能够过滤出域系统的用户而不是保证所有副本同时更新。当由于网络或主机故障，更新不能进行时，通常的做法是相信旧信息同时继续努力更新它。一般模式是用刷新超时分发副本。分发者设置超时值，分发物的接收者负责执行刷新。在特殊情况，可能规定非常短的间隔，或者所有者可能禁止复制。
- 在任何有分布式数据库的系统中，特定名称服务器可能遇到仅可以由某个其他服务器回答的查询。处理此问题的两个常用方法其一是“递归”，在递归中，第一个服务器继续为客户端在另一个服务器上查询；其二是“迭代”，在迭代中，该服务器引导客户端寻找另一个服务器，并让该客户端继续查询。两种方法各有优缺点，但是在数据报形式的访问中优先使用迭代方法。域系统需要执行迭代方法，但是允许将递归方法作为选项。

域系统假设，所有数据起源于使用域系统的主机散发的主文件。这些主文件被本地系统管理员更新。主文件是文本文件，本地名称服务器读这些主文件，因此经名称服务器后，可被该域系统的用户得到。用户程序通过称为解析器的标准程序访问名称服务器。

主文件采用标准格式因而它们能够在主机间交换(通过FTP、邮件、或某种其他机制)；当组织想要域，但是不希望支持名称服务器时，这个特点是有用的。组织能够维护本地使用文本编辑器的主文件，将它们传送给运行名称服务器的外地主机，接着与那个名称服务器的系统管理者协商获得加载的文件。

本地系统管理者配置每个主机的名称服务器和解析器[RFC-1033]。对于名称服务器，这个配置数据包括本地主文件标识，和有关从外地服务器加载非本地主文件的指令。名称服务器使用主文件或主文件副本加载它的区域。对于解析器，此配置数据指出应当是主信息源的名称服务器。

域系统定义访问数据和转介到其他名称服务器的流程。域系统也定义由系统管理员确定的缓存检索数据的流程和数据周期刷新的流程。

系统管理者提供：

- 区域边界定义。
- 数据的主文件。
- 对主文件的更新。
- 陈述希望的刷新策略。

域系统提供：

- 资源数据的标准格式。
- 查询数据库的标准方法。

- 名称服务器刷新来自外地名称服务器的本地数据的标准方法。

2-4 DNS组成部分

DNS有三个主要部分：

- 域名空间和资源记录(DOMAIN NAME SPACE和RESOURCE RECORDS)，它们是树状结构名称空间和与这些名称关联的数据的规范。从概念上讲，每个节点和域名空间树的叶子命名一组信息，查询操作是打算从特定集合提取特定类型信息。查询命名感兴趣的域名和描述希望的资源信息类型。例如，互联网使用它的某个域名标识主机；地址资源的查询返回互联网主机地址。
- 名称服务器(NAME SERVERS)，它们是服务器程序，它们保存着有关域树的结构的信息和设置信息。名称服务器可以缓存有关域树的任何部分的结构和设置信息，但是一般而言，特定的名称服务器有关于该域空间的子集的完整信息，有指向其他名称服务器的指针，这些其他服务器可用于从域树的任何部分引导信息。名称服务器们知道域树的各个部分(它们对其有完整信息)；对于名称空间的这些部分，名称服务器被看作是权威(AUTHORITY)。权威信息被编排进称作区域(ZONEs)的单元，这些区域可以自动分配给为区域中的数据提供冗余服务的名称服务器。
- 解析器(RESOLVERS)，是为了响应客户端请求从名称服务器提取信息的程序。解析器必须能够访问至少一个名称服务器，并使用那个名称服务器的信息直接回答查询，或使用到其他名称服务器的转介，继续查询。解析器一般是系统例行程序，它可直接访问到用户程序；因此在解析器和用户程序间不需要协议。

这三个组成部分大致对应域系统的三层或三个视野：

- 从用户的观点看，通过简单的流程或OS调用本地解析器，访问域系统。域空间由单一树构成，用户可以从该树的任何部分请求信息。
- 从解析器的观点看，域系统由未知数量的名称服务器组成。每个名称服务器有整个域树的数据中的一条或多条数据，然而解析器将这些数据库的每一个看作基本上是静态的。
- 从名称服务器的观点看，域系统由分离的、称作区域的本地信息集合构成。名称服务器有某些区域的本地副本。名称服务器必须根据本地文件中的主副本或外地名称服务器，周期刷新它的区域。名称服务器必须并发处理来自多个解析器的查询。

从性能考虑，实现可以融合这些功能。例如，在作为名称服务器的同一计算机上的解析器或许共享由区域构成的数据库(区域由该名称服务器管理)，以及可以共享由该解析器管理的缓存器。

第3章 域名空间和资源记录

3-1 名称空间规范和术语

域名空间是树形结构。每个节点和树上的叶对应资源集合(资源集合可以为空)。域系统对内部节点使用和树叶使用不加区别，本备忘录使用术语“节点(node)”指代两者。

每个节点有标签，标签长度从0到63个八位位组。兄弟节点不可以有相同的标签，尽管相同标签可用于不是兄弟的节点。一个标签被保留，它是空(即零长度)标签，用于树根。

节点的域名是从节点到树根路径沿途各个标签的明细表。按照惯例，构成域名的标签从左到右书写或读出，从最具体的(最低，离树根最远)到最不具体的(最高，离树根最近)。

在内部，操控域名的程序应当把域名表示为标签序列，这里每个标签是长度八位位组，

再加上八位位组字符串。因为所有域名在树跟结束，树根有标签的空字符串，这些内部表示可以使用零长度字节来终止域名。

按照惯例，域名可以用任意大小写保存，但是对于所有现有的域功能，域名比较采用不区分大小写的方式进行，假设采用ASCII字符集，高阶零位。这意味着您可以自由使用标记“A”生成节点，或使用标记“a”生成节点，但是二者不能作为兄弟；您可以或者使用“a”表示节点，或者使用“A”表示节点。当您收到域名或标签时，您应当保存它的大小写。这样选择的理由是或许某一天我们需要为新的业务添加全二进制域名；现有的业务不改变。

当用户需要打印域名时，忽略每个标签的长度，标签被用圆点(“.”)隔开。因为完整的域名用根标签结束，这产生以圆点结束的书写格式。我们使用这个性质相互间区别：

- 代表完整域名的字符串(常称作“绝对”)。例如，“poneria.ISI.EDU.”
- 代表不完整域名的开始标签的字符串，应当使用本地域知识，由本地软件完善该字符串(常称作“相对”)。例如，使用在ISI.EDU域中的“poneria”。

选取相对名称，或者是相对于众所周知的起源选取，或者是相对于用作搜索列表的域的明细表选取。相对名称大多出现在用户接口中，在那里，由于实现不同对相对名称的翻译也各不相同，在主文件中，相对名称相对于单一起源域名。最通用的翻译使用根“.”作为或者是单一起源，或者是搜索列表的成员之一，所以多标签相对名称常常是这样的相对名称，在那里尾随的圆点因为要节省打印而被忽略掉。

为了简化实现，表示域名的八位位组的总数(即，所有标签八位位组和标签长度的和)被限制到255。

域由域名标识，域由域名空间的那个部分组成，那个部分位于该域名中或在该域名以下，该域名规定该域。域是另一个域的子域，如果该域被包含在另一个域中。通过观察是否子域的名称用包含域的名称结束，可以检验这种相互关系。例如，A.B.C.D是B.C.D，C.D，D和“”的子域。

3-2 有关应用的管理准则

作为一个策略，DNS技术规范没有规定具体的树状结构或选择标签的规则；这样做的目的是尽可能通用，以便它能够用来建立任意应用。尤其是，将系统设计成名称空间不必沿着网络边界线、名称服务器等构建。这样做的理由不是名称空间应当没有隐含语义，而是隐含语义的选择应当保持开放，以便应用于被考虑的问题，以及树的不同部分可以有不同的隐含语义。例如，IN-ADDR.ARPA域是根据网络和主机地址进行组织和分发的，因为它的作用是从网络号或主机号翻译到名称；NetBIOS域[RFC-1001，RFC-1002]是扁平的，因为对于那个应用这是适当的。

然而，存在一些准则，它们适合用于主机、邮箱等的名称空间的“一般”部分，这些准则将使得名称空间更加统一，有利于其成长，以及尽量减少因软件是从较旧的主机表转换而来引起的问题。关于树的顶层的策略决策在RFC-920中讨论。目前的顶层策略在[RFC-1032]中讨论。MILNET转换问题在[RFC-1031]中讨论。

最终将被插入多个区域的较低的那些域应当在该域的顶部提供分支，以便无需重新命名即可进行最终的分解。使用特定字符串、前置数字等的节点标签极有可能突破较旧的软件(旧软件更多依赖于严格的选择)。

3-3 有关应用的技术准则

DNS能够用于保持某种对象的命名信息之前，必须满足两点：

- 在对象名称和域名之间进行映射的约定。这描述如何访问有关对象的信息。
- 描述对象的RR类型和数据格式。

这些准则可能非常简单或十分复杂。常常，设计者必须顾及已有的格式，必须规划向上兼容已有的应用。可能需要多映射或映射层级。

对于主机，映射取决于已有的主机名称句法(主机名称是域名的常用文本表示法的子集)，以及描述主机地址的RR格式，等等。因为我们需要可靠的逆映射(从地址到主机名称)，也定义了进入IN-ADDR.ARPA域的地址的特殊映射。

对于邮箱，映射稍微复杂一些。通过把<local-part>转变为单一标签(忽略标签包含的圆点)，使用域名的常用文本格式把<mail-domain>转变为域名(圆点表示标签中断)，并串联两部分形成单一的域名，通常的邮件地址<local-part>@<mail-domain>被映射成域名。于是，邮箱HOSTMASTER@SRI-NIC.ARPA由HOSTMASTER.SRI-NIC.ARPA表示为域名。评价这种设计背后的原因也必须考虑邮件交换的方案[RFC-974]。

普通用户不关心定义这些规则，但是应当理解这些规则通常是下述方面大量互动的结果，即，向上兼容旧有应用愿望之间的妥协，不同的对象定义之间的相互影响，以及当定义这些规则时不可避免的添加新特征冲动。使用DNS支持某些对象的方法常常比在DNS中固有的限制更为关键。

3-4 名称空间举例

图1显示当前域名空间的一部分，该图在本RFC的许多举例中使用。注意，此树是实际名称空间的非常小的子集。

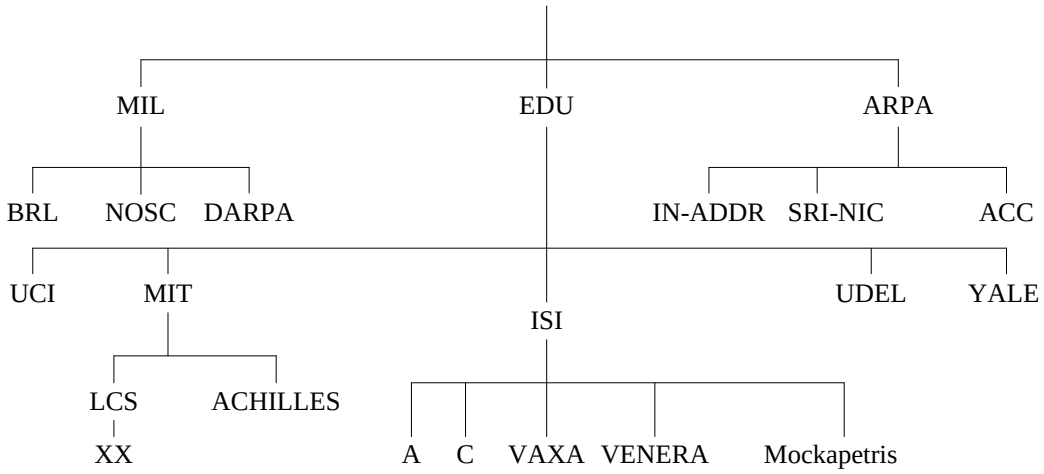

图1 域名空间一部分举例

在这个例子中，根域有三个直接子域：MIL、EDU和ARPA。LCS.MIT.EDU域有一个名为XX.LCS.MIT.EDU的直接子域。所有叶子也是域。

3-5 优先选用的名称句法

DNS规范力争在构建域名规则方面尽可能通用。具体思路是：任何已有对象的名称，能够以尽可能小的改动表示为域名。然而，当为对象分配域名时，谨慎的用户将选择既满足域名系统规则，又满足对象任何已有的规则的名称，无论这些规则已公开发表，还是隐含在已有程序中。

例如，当命名邮件域时，用户应当既满足本备忘录的规则，又满足RFC-822中的规则。当生成新的主机名称时，应当遵守旧的HOSTS.TXT规则。这可以避免当旧的软件转换为使用域名时可能出现的问题。

下述句法很少出现与使用域名的许多应用(例如：邮件，TELNET)有关的问题。

<domain> ::= <subdomain> | " "
<subdomain> ::= <label> | <subdomain> "." <label>
<label> ::= <letter> [[<ldh-str>] <let-dig>]
<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>
<let-dig-hyp> ::= <let-dig> | "-"
<let-dig> ::= <letter> | <digit>
< letter> ::= 52个字母字符，大写的A到Z和小写的a到z，中的任何一个。
< digit> ::= 十进制数字0到9中的任何一个。

注意，尽管在域名中大、小写字母都是允许的，但是区别大小写没有意义。即，两个有相同拼写，不同大小写的名称被看作同一名称。

标签必须遵循ARPANET主机名称规则。它们必须以字母开始，以字母或数字结束，中间的字符只能是字母、数字和连字符。对长度也有一些限制。标签必须少于等于63个字符。

例如，下述字符串标识互联网中的主机：

A.ISI.EDU XX.LCS.MIT.EDU SRI-NIC.ARPA

3-6 资源记录

域名标识节点。每个节点有一组资源信息，资源信息可以为空。与特定名称关联的资源信息集合由分开的资源记录(RRs)构成。在集合中RRs的次序没有意义，并且不需要由名称服务器、解析器、或DNS的其他部分保存。

当我们谈论特定RR时，我们假设它有下述内容：

所有者(owner) 它是域名，在该域名中RR被发现。

类型(type) 它是编码的16位值，该值规定在这个资源记录中的资源的类型。类型指抽象的资源。

本备忘录使用下述类型：

A	主机地址
CNAME	标记别名(alias)的正则(canonical)名称
HINFO	标记主机使用的CPU和OS
MX	标记域的邮件交换。更多内容参阅[RFC-974]。
NS	该域的权威名称服务器
PTR	到域名空间别的部分的指针
SOA	标识权威区域的开始

类(class) 它是编码的16位值，该值标识协议族或协议实例。

本备忘录使用下述类：

IN	互联网系统
CH	混沌(Chaos)系统

TTL 它是RR的生存时间。这个字段是32位整数，以秒为单位，主要由解析器缓存RRs时使用。TTL描述在RR应当被抛弃前，它能够被缓存多长时间。

RDATA 它可以是类型或类，取决于描述该资源的数据：

A	对于IN类，是32位IP地址。 对于CH类，是域名，再加上16位八进制Chaos地址。
CNAME	域名
MX	16位首选项值(越低越好。)，再加上愿意充当所有者域邮件交换的主机名称。

NS	主机名称
PTR	域名
SOA	几个字段

所有者名称常常是隐式的，不属于RR不可分割的一部分。例如，许多名称服务器在内部形成名称空间的树或哈希结构，束缚离开节点的RRs。其余RR部分是固定首部(类型、类、TTL)和可变部分(RDATA)，固定首部与所有RRs一致，RDATA适合正在被描述的资源的需要。

TTL字段的含义是关于RR能够被缓存多长时间的限制。这个限制对区域中的权威数据不适用；权威数据也会超时，但是使用该区域的刷新策略。管理员为数据起源区域分配TTL。虽然短TTLs能够用于尽量减少缓存，零TTL禁止缓存，互联网性能的现实暗示，对于典型的主机，这些时间应当设置为几天的数量级比较合适。如果能够预测到有改变，在改变前可以将TTL减少，以便尽量减小改变期间的不一致，然后随着改变，增加并返回到它的先前值。在RRs的RDATA部分中的数据被作为二进制串和域名的组合携带。域名被频繁用作指向DNS中其他数据的“指针”。

3-6-1 RRs的文本表示

在DNS协议的分组中，RRs用二进制形式表示，当存储在名称服务器或解析器中时，RRs通常用高度编码的形式表示。在本备忘录中，我们采用类似在主文件中使用的体裁，以便显示RRs的内容。在此格式中，大多数RRs在一行上显示，然而可以使用圆括号连续显示几行。在行的一开始给出RR的所有者。如果行以空格开始，那么，假设所有者与前一个RR的相同。为了方便阅读常常包括空行。

在所有者之后，我们列出RR的TTL、类型和类。类和类型使用上面定义的助记符，TTL是类型字段前的整数。为了避免分析上的歧义，类型助记符和类助记符不相交，TTLs是整数，类型助记符总是位于最后。在意思表达清晰的情况下，举例中常常忽略IN类值和TTL值。

使用数据的典型表示法知识，给出资源数据或RR的RDATA部分。

例如，我们或许将消息中携带的RRs显示为：

ISI.EDU.	MX	10 VENERA.ISI.EDU.
	MX	10 VAXA.ISI.EDU.
VENERA.ISI.EDU.	A	128.9.0.32
	A	10.1.0.52
VAXA.ISI.EDU.	A	10.2.0.27
	A	128.9.0.33

MX RRs有RDATA部分，该部分由16位数构成，再加上域名。地址RRs使用标准IP地址格式，包括32位网络互联地址。

此例显示6个RRs，3个域名中的每一个有2个RRs。

类似，我们或许看到：

XX.LCS.MIT.EDU.	IN	A	10.0.0.44
	CH	A	MIT.EDU. 2420

此例显示XX.LCS.MIT.EDU的2个地址，每个具有不同的类。

3-6-2 别名和正则名称

在现有系统中，主机和其他资源常常有几个名称，这些名称标识同一资源。例如，名称C.ISI.EDU 和USC-ISIC.ARPA都标识同一个主机。类似，在邮箱情况，许多组织提供多个名

称，这些名称实际上去相同邮箱；例如，Mockapetris@C.ISI.EDU、Mockapetris@B.ISI.EDU、和 PVM@ISI.EDU都去同一邮箱(尽管在此背后的机制有些复杂)。

这些系统中的大多数有这样的共识，即，等同的名称集合中的一个名称是正则的(canonical)或主要的名称，所有其他的都是别名。

域名系统提供使用正则名称(CNAME)RR的特征。CNAME RR标识它的所有者名称为别名，并在RR的RDATA部分规定相应的正则名称。如果节点中存在CNAME RR，不应当存在其他数据；这确保正则名称和它的别名的数据不会有不同。这个规则也确保缓存的CNAME可以不加检验就可以由其他RR类型的权威服务器使用。

在DNS软件中，CNAME RRs引起特殊行动。当名称服务器不能在与该域名关联的资源集合中发现期望的RR时，名称服务器检验是否资源集合与具有匹配类的CNAME记录一致。如果一致，名称服务器在响应中包括此CNAME记录，并且在此CNAME记录的数据字段中规定的域名重新开始查询。此规则的一个例外是：不重新开始匹配此CNAME类型的那些查询。

例如，假设名称服务器正在处理针对USCISIC.ARPA的查询，询问A类型信息，并有下述资源记录：

USC-ISIC.ARPA	IN	CNAME	C.ISI.EDU
C.ISI.EDU	IN	A	10.0.0.52

这两个RRs将在对A类型查询的响应中返回，而CANME类型或*查询应当仅返回CNAME。

在RRs中的域名(这些域名指向另一个名称)应当总是指向主(正则)名称，而不是指向别名。这可以避免访问信息中的额外迂回。例如，对于上面的主机，到名称RR的地址应当是：

52.0.0.10.IN-ADDR.ARPA IN PTR C.ISI.EDU

而不指向USC-ISIC.ARPA。当然，由稳健性原则，当存在CNAME链或环时，域软件不应当失败；应当接受CNAME链，CNAME环预示着出错。

3-7 查询

查询是消息，这些消息可以被发送到名称服务器，激发响应。在互联网中，查询由UDP数据报携带，或在TCP连接上传送。名称服务器的响应或者回答查询中提出的问题，引导提问者到另一组名称服务器，或者提示某个错误条件。

一般讲，用户不会直接生成查询，但是会向解析器提出请。解析器依次发送一个或多个查询给名称服务器，解析器处理错误条件和处理可能出现的转介。当然，在查询中能够被询问的可能问题就形成解析器能够提供的业务种类。

DNS查询和响应用标准的消息格式携带。此消息格式有包括众多固定字段(这些字段总是存在)的首部，和携带查询参数和RRs的4个部分。

首部中最重要的字段是称作操作码的字段，它有4位，用来区分不同的查询。具有16个可能值，1(标准查询)是正式协议的一部分，2(反向查询和状态查询)是选项，1(完整)是被废弃的，其余部分没有分配。

4个部分是：

问题	携带查询名称和其他查询参数。
回答	携带直接回答查询的RRs。
权威	携带描述其他权威服务器的RRs。可以选择在回答部分携带权威数据的SOA RR。
附加	携带在其他部分中使用RRs时，可能有帮助的RRs。

注意，这些部分中的内容，不是格式，会随首部操作码改变。

3-7-1　标准查询

标准查询规定目标域名(QNAME)、查询类型(QTYPE)和查询类(QCLASS)，以及询问匹配的RRs。查询的这个类型形成DNS查询的绝大多数，除非另有规定，我们使用术语“查询”指标准查询。QTYPE字段和QCLASS字段都是16位长，是定义的类型和定义的类的超集。

QTYPE字段可能包括：

< any type>	仅匹配那个类型(例如，A、PTR)
AXFR	传递QTYPE的特定区域
MAILB	匹配所有与RRs有关的邮箱(例如，MB和MG)
*	匹配所有RR类

QCLASS字段可能包括：

< any class>	仅匹配那个类(例如，IN、CH)
*	匹配所有RR类

使用查询域名、QTYPE和QCLASS，名称服务器查找匹配的RRs。除了相关的记录以外，名称服务器可能返回RRs，这些RRs指向有想要的信息的名称服务器，或者这些RRs被认为有助于解释相关RRs。例如，没有所请求信息的名称服务器可能知道某个名称服务器有该信息；在相关RR中返回域名的名称服务器也可能返回绑定那个域名到地址的RR。

例如，尝试发送邮件到Mockapetris@ISI.EDU的邮件收发程序或许向解析器询问关于ISI.EDU的邮件信息，这导致对QNAME=ISI.EDU、QTYPE=MX、QCLASS=IN的查询。该响应的回答部分或许是：

ISI.EDU.	MX	10 VENERA.ISI.EDU.
	MX	10 VAXA.ISI.EDU.

而附加部分或许是：

VAXA.ISI.EDU.	A	10.2.0.27
	A	128.9.0.33
VENERA.ISI.EDU.	A	10.1.0.52
	A	128.9.0.32

因为服务器假设，如果请求者想要邮件交换信息，请求者不久的将来或许也想要该邮件交换地址。

注意，QCLASS=*结构要求关于权威的特殊解释。因为特定名称服务器可能不知道域系统中所有可用的类，该特定名称服务器可能决不会知道是否对于所有类，它是权威的。因此，对QCLASS=*查询的响应可能绝不是权威的。

3-7-2　反向查询(可选)

名称服务器也可以支持反向查询，反向查询映射具体的资源到域名或一些域名(它们都有那个资源)。例如，当标准查询映射域名到SOA RR时，相应的反向查询映射SOA RR回到该域名。

这个业务的实现在名称服务器中是可选项，但是所有名称服务器必须至少能够理解反向查询消息，并返回说明不能实现的出错响应。

域系统不能保证反向查询的完整性或唯一性，因为域系统是由域名而不是由主机地址或其它任何资源类型构成的。反向查询主要用于调试和数据库维护。

反向查询可能不返回适当的TTL，反向查询不指出这样的情况，即，标识的RR是一组中的一个(例如，有多个地址的主机的一个地址)。因此，绝不应当缓存在反向查询中返回的RRs。

对于映射主机地址到主机域名，反向查询**不是**可接受的方法；使用IN-ADDR.ARPA域代替之。

有关反向查询的详细讨论参阅[RFC-1035]。

3-8 状态查询(试验中)

将要对其定义。

3-9 完整查询(放弃)

在RFC 882和RFC 883中介绍的可选的完整业务已经被删除。将来重新设计的该业务或许可以应用，或者收回操作码给其他业务使用。

第4章 名称服务器

4-1 序言

名称服务器是构成域数据库的信息仓库。域数据库被划分成称为区域(zone)的部分，区域在众多名称服务器间分配。虽然名称服务器可以有几个可选的功能和数据源，它的基本任务是：使用在它的区域中的数据回答查询。通过设计，名称服务器能够用简单的方法回答查询；响应总可以仅使用本地数据产生，响应或者包含对问题的回答，或者包含到其他“更接近”想要信息的名称服务器的转介。

可从几个名称服务器得到给定的区域，从而确保给定区域的可用性，无论主机或通信链路是否有故障。通过管理命令，我们要求每一个区域至少可在两台服务器上得到，许多区域有比此更高的冗余度。

给定的名称服务器典型支持一个或多个区域，但是这仅给予名称服务器有关域树的小部分的权威信息。名称服务器也可能有某些缓存的有关该树的其他部分的非权威数据。名称服务器标记它的查询响应，以便请求者能够识别响应来自权威数据，或不是。

4-2 怎样将数据库划分成区域

有两种方法划分域数据库：其一是按类，其二是在节点间的名称空间中进行“切割(cut)”。

按类划分简单。任何类数据库的组织、授权和维护独立于所有其他类。因为，按照惯例，名称空间对于所有类是相同的，分开的类可被视为并行名称空间树的阵列。注意，对于这些不同的并行类，附属于节点的数据是不同的。产生新类的最普通理由是现有类型的新数据格式所必须的，或对现有名称空间实行分开管理的愿望。

在类中，在名称空间内的“切割”可以在任何两个毗邻节点间进行。完成所有切割后，每个连接名称空间的组是独立的区域。对于连接的区间(region)内的所有名称，该区域被认为是权威的。注意，名称空间内的“切割”可以位于不同类的不同位置，名称服务器可以不同，等等。

这些规则意味着每个区域至少有一个节点，因此域名(对于域名区域是权威的)，以及特定区域内的所有节点被连接。假设，树状结构，每个区域有最高的节点(在该区域内该节点比任何其他节点都更靠近树根)。这个节点的名称常用来标识该区域。

有可能的是，虽然不是特别有用，划分名称空间造成每个域名位于独立的区域，或造成所有节点位于单一区域。反之，数据库被在一些点上划分，在这些点上特定的组织希望接管对子树的控制。一旦组织控制它自己的区域，组织能够单方面改变该区域中的数据，生长连接到该区域的新树部分，删除现有的节点，或在他的区域下授权新的子区域。

如果组织有子结构，组织可能希望做进一步的内部划分，以便实现名称空间控制的嵌套授权。在某些情况，这种划分纯粹是为了方便数据库维护。

4-2-1　技术上的考虑

描述区域的数据有4个主要部分：

- 区域内所有节点的权威数据。
- 定义区域的顶层节点的数据(可被视为权威数据的一部分)。
- 描述授权的子区域(即，靠近区域底部的切割)的数据。
- 允许访问子区域的名称服务器的数据(有时称为“胶(glue)”数据)。

这个数据的所有部分都以RRs形式表示，所以借助一组RRs能够完整描述区域。通过传送RRs，整个区域可以在名称服务器之间传送，或者是在一系列消息中携带，或者是通过FTP发送文本方式表示的主文件传送。

区域的权威数据就是附属于所有节点的所有RRs，这些节点包括从区域的顶层节点向下到叶节点，或者向下到靠近区域底部边缘的切割(cut)上面的节点。

虽然逻辑上是权威数据的一部分，描述区域顶层节点的RRs对于区域管理特别重要。这些RRs有两个类型：名称服务器RRs，它列出区域的所有服务器(每个RR列出一个)；单SOA RR，它描述区域管理参数。

描述靠近区域底部的切割的RRs是NS RRs，它命名子区域的服务器。因为切割位于节点之间，这些RRs不是区域权威数据的一部分，这些RRs应当与子区域顶层节点中相应RRs精确相同。因为名称服务器总是与区域边界关联，NS RRs仅在这样的节点出现，这些节点是某个区域的顶层节点。在形成区域的数据内，NS RRs在区域的顶层节点出现(并且是权威的)，以及出现在靠近区域的底部的切割中(在哪里，这些NS RRs不是权威的)，但是NS RRs绝不出现在二者之间。

区域结构的目标之一是：任何区域都有建立与任何子区域的名称服务器通信需要的所有数据。即，父区域有访问他们子区域的服务器所需要的所有信息。命名子区域服务器的NS RRs常常不足以完成这项任务，因为NS RRs命名这些服务器，但是不给出它们的地址。尤其是，如果子区域中名称服务器的名称是它自己时，我们可能面对这样的情况，那里NS RRs告诉我们，为了学习名称服务器的地址，我们应当使用我们希望学习的地址去联系该服务器。为了解决这个问题，区域包括“胶(glue)”RRs，它们不是权威数据的一部分，它们是那些服务器的地址RRs。如果名称服务器的名称“低于”切割时这些RRs才是必须的，并且这些RRs仅用作转介响应的一部分。

4-2-2　管理上的考虑

当某个组织希望控制它自己的域时，第一步要做的是标识适当的父区域，获得父区域所有者同意控制授权。虽然对在树中的哪里可以这样做没有特别的技术限制，在[RFC-1032]中讨论了一些管理上的分组，这些分组与顶层组织打交道，中层区域可自由创立它们自己的规则。例如，一所大学或许选择使用单一区域，而另一所大学或许选择专门针对各个系或学院的子区域进行组织。[RFC-1033]归类了可用的DNS软件，并讨论了管理流程。

一旦为新的子区域选择了适当的名称，应当要求新的所有者证实支持冗余名称服务器。注意，没有要求区域的服务器驻留在在那个域有域名的主机上。在许多情况，如果区域的服务器们广泛分布，而不是位于由管理该区域的同一组织控制的物理设备内，区域更容易方便地由互连网络访问。例如，在目前的DNS中，英国(或UK域)的名称服务器之一在美国出现。这使得美国主机不必使用有限的跨大西洋带宽，就能够获得UK数据。

作为最后的安装步骤，应当添加使授权生效必须的授权NS RRs和胶(glue)RRs到父区域。两个区域的管理者应当确保NS RRs和glue RRs(它们标记切割的两边)一致并保持一致。

4-3 名称服务器内部

4-3-1 查询和响应

名称服务器的主要活动是回答标准查询。采用[RFC-1035]中介绍的标准消息格式携带查询和查询的响应。查询包括QTYPE、QCLASS和QNAME，它们描述期望信息和感兴趣名称的类型和类。

名称服务器回答查询的方法取决于查询正在以递归模式运行，或者不是：

- 对服务器来说，最简单模式是非递归，因为服务器可以仅使用本地信息回答查询：响应包括出错、回答，或到某个“更接近”该回答的其他服务器的转介。所有名称服务器必须执行非递归查询。
- 对客户端来说，最简单模式是递归，因为在这种模式中，名称服务器充当解析器的角色，返回的内容或者是出错，或者是回答，但绝不是转介。在名称服务器中，这个业务是可选的，并且名称服务器也可能选择限制能够使用递归模式的客户端。

递归业务在以下几种情况是有用的：

- 相对简单的请求者，该请求者缺乏使用除了对问题的直接回答以外的任何东西的能力。
- 需要跨协议或跨其它边界的请求，以及能够被发送到可以充当中继的服务器的请求。
- 网络，在该网络中，我们希望集中缓存，而不是每个客户端有独立的缓存器。

如果请求者有能力跟踪转介，以及对有助于将来的请求的信息感兴趣，非递归业务是适当的。

递归模式的使用限定在一些场景，在那里，客户端和名称服务器一致同意使用递归模式。在查询消息和响应消息中，通过两个位的使用，对是否同意可以协商：

- RA位，由名称服务器在所有响应中置1，递归可以使用；或置0，递归不可以使用。如果名称服务器愿意为该客户端提供递归业务，此位为真，无论是否该客户端请求了递归业务。即，RA暗示可用性而不是使用。
- 查询包括称作希望递归(recursion desired)或RD的位。此位规定是否请求者希望这个查询使用递归业务。客户端可以请求来自任何名称服务器的递归业务，尽管客户端们应当仅相信从下述服务器接收的递归业务，这些服务器先前已经发送了RA，或者这些服务器已经同意通过私有协定或某些DNS协议以外的其他方法提供业务。

当带有RD置1的查询到达服务器(该服务器愿意提供递归业务)时，递归模式发生；通过检验在此响应中RA和RD都被置1，客户端可以证实使用了递归模式。注意，除非经RD要求，名称服务器绝不应执行递归业务，因为这会干扰名称服务器和它们的数据库的故障排查。如果要求递归业务并且可以使用，查询的递归响应将是下列之一：

- 对查询的回答，可能由一个或多个CNAME RRs开始，这些CNAME RRs规定在到达回答的路径上遇到的别名。
- 指出名称不存在的名称错误。它可能包括CNAME RRs，这些CNAME RRs指出起源的查询名称是不存在的名称的别名。
- 临时错误指示。

如果没有要求递归业务，或者不能获得递归业务，非递归响应将是下列之一：

- 权威名称错误，它指出该名称不存在。
- 临时错误指示。
- 下述的某个组合：

回答问题的RRs，连同是否数据来自区域或被缓存的指示。

到这样的名称服务器们的转介，与发送该响应的服务器相比，这些名称服务器有更接近于到该名称的祖宗们的区域。

- 名称服务器认为这些RRs将对请求者有用。

4-3-2 算法

名称服务器使用的实际算法取决于本地OS和存储RRs的数据结构。下述算法假设用几个树状结构组织RRs，一个树状结构用于每个区域，别的树状结构用于缓存：

1、根据名称服务器是否愿意提供递归业务，响应中递归可用的值被置1或置0。如果递归业务可用，并且在查询中经RD位请求递归业务，转到步骤5，否则到步骤2。

2、针对最接近到QNAME祖宗的区域，搜索可用区域，如果找到这样的区域，转到步骤3，否则到步骤4。

3、在该区域中，开始逐标签地向下匹配。此匹配处理可以终止下述几种情况：

a、如果整个QNAME匹配，我们找到该节点。

如果在该节点中的数据是CNAME，并且QTYPE不匹配CNAME，复制此CNAME RR进响应的回答部分，改变QNAME到在CNAME RR中的正则名称，返回到步骤1。

否则，将所有匹配QTYPE的RRs复制进回答部分并转到步骤6。

b、如果匹配从我们中除去权威数据，我们有转介。当我们遇到有NS RRs(该NS RRs标记沿区域底部的切割)的节点时，这种情况发生。

将此子区域的NS RRs复制进响应的权威部分。将无论什么样的可得到地址放进附加部分，如果这些地址不能从权威数据或缓存器中获得，使用胶RRs。转到步骤4。

c、如果在某个标签中，匹配是不可能的(即，相应的标签不存在)，查看是否“*”标签存在。

如果“*”标签不存在，验证或者我们正在寻找的名称是该查询中的起源QNAME，或者我们正在寻找的名称是我们基于CNAME跟踪的名称。如果该名称是起源的，在响应中设置权威名称错误并退出。否则，仅仅退出。

如果“*”标签存在，在那个节点中根据QTYPE匹配RRs。如果存在任何匹配，将它们复制进回答部分，但是设置RR的所有者是QNAME，并且不是带有“*”标签的节点。转到步骤6。

4、在缓存器中开始向下匹配。如果在缓存中找到QNAME，将所有附属于此QNAME的RRs(这些RRs匹配QTYPE)复制进回答部分。如果没有来自权威数据的授权，从缓存中查找最好的一个，并将其放置在权威部分。转到步骤6。

5、使用本地解析器或本地解析器的算法的副本(参阅本备忘录的解析器部分)，回答查询。在响应的回答部分保存结果，包括任何中间CNAMEs。

6、仅使用本地数据，尝试添加其他RRs，这些RRs对于查询的附加部分或许有用。退出。

4-3-3 通配符

在前面的算法中，具有以“*”标签开始的所有者名称的RRs被给与特殊待遇。这样的RRs称作通配符。可将通配符RRs看作是合成RRs的指令。当满足适当条件时，名称服务器生成具有所有者名称等于查询名称的RRs，以及生成取自该通配符RRs的内容。

这个便利常常大量用于生成这样的区域，该区域将用于从互联网到某个其他邮件系统转

发邮件。一般思路是，在那个区域(该区域存在于查询中的服务器)中的任何名称都将被假设存在，具有确定属性，除非有明确相反的证据存在。注意，这里故意使用术语区域(zone)，而不使用域(domain)；这样的默认不跨区域边界传播，尽管通过设置类似的默认子区域可以选择实现那个外部特性。

通配符RRs的内容遵循通常的RRs规则和格式。在区域中的通配符有所有者名称，该所有者名称控制通配符将匹配的查询名称。通配符RRs的所有者名称的格式为“*.<anydomain>”，这里，< anydomain>是任何域名。<anydomain>不应当包括其他*标签，并且应当位于区域权威数据中。通配符潜在用于<anydomain>的子孙，但是不用于< anydomain>自身。检查此的另一种方法是“*”标签总是匹配至少一个完整的标签，有时不止一个，但总是整个标签。

通配符RRs不用于：

- 当查询是在另一个区域中。即，授权取消通配符默认。
- 当查询名称或通配符域和查询名称之间的名称已知存在时。例如，如果通配符RR有“*.X”的所有者名称，并且该区域也包括附属于B.X的RRs，该通配符可用于Z.X名称的查询(假设没有Z.X的精确信息)，但是不用于B.X，A.B.X或X。

出现在查询名称中的*标签没有特殊作用，但是能够用于在权威区域中检测通配符；这样的查询是获得包括RRs(这些RRs具有所有者名称，这些所有者名称中带*)的响应的唯一方法。不应当缓存这样的查询结果。

注意，当用于合成RRs时，不修改通配符RRs的内容。

为了演示通配符RRs的应用，假设某大型公司采用大型、非IP/TCP、希望生成邮件网关的网络。如果称该公司为X.COM，称IP/TCP使能网关计算机为A.X.COM，或许下述RRs被输入进COM区域：

X.COM	MX	10	A.X.COM
*.X.COM	MX	10	A.X.COM
A.X.COM	A	1.2.3.4	
A.X.COM	MX	10	A.X.COM
*.A.X.COM	MX	10	A.X.COM

这将引起针对任何以X.COM结束的域名的任何MX查询，以便返回指向A.X.COM的MX RR。要求两个通配符RRs，因为通过明确的A.X.COM数据，在*.X.COM中的通配符的作用被限制在A.X.COM子树中。也要注意，在X.COM和A.X.COM中的明确的MX数据是要求的，以及上述RRs中将没有一个匹配XX.COM的查询名称。

4-3-4　否定响应缓存(可选)

DNS提供可选业务，该业务使名称服务器能够分发带TTLs的否定结果，使解析器能够缓存带TTLs的否定结果。例如，名称服务器能够随同名称错误指示分发TTL，接收这样信息的解析器被允许不咨询权威数据就假设：在TTL周期期间该名称不存在。类似，解析器可以用QTYPE(该QTYPE匹配多种类型)进行查询，并缓存某些类型不存在的事实。
这一特点在实现命名速记(命名速记使用搜索列表)的系统中特别重要，因为流行的速记(它正好需要接近搜索列表末尾的后缀)，只要它被使用，将产生多名称错误。

此方法是名称服务器可以添加SOA RR到响应的附加部分，当那个响应是权威的时。此SOA必须是区域的SOA，该区域是在回答部分中权威数据(或者是名称错误，如果适用)的源。此SOA的MINIMUM字段用于控制否定结果可以被缓存的时间长度。

注意，在某些场景，回答部分可以包括多个所有者名称。在此情况，SOA机制应当仅用于匹配QNAME的数据，该数据是这个部分中唯一的权威数据。

名称服务器和解析器绝不应当企图添加SOAs到非权威响应的附加部分，或企图推测不

是在权威响应中直接陈述的结果。这有几个原因，包括：缓存的信息常常不足以匹配RRs和RRs的区域名称，基于直接SOA查询SOA RRs可以被缓存，以及不要求名称服务器在权威部分中输出SOAs。这一特性是可选的，尽管预期将来修订的版本将变成标准协议的一部分。不要求名称服务器在所有权威响应中添加SOA RRs，也不要求解析器缓存否定的结果。对此二者仅是推荐。

要求当SOA RR在响应中出现时，所有解析器和递归名称服务器至少能够忽略SOA RR。也已经建议一些试验使用此特性。理想情况是如果已知缓存的数据来自特定区域，并且如果得到该区域的SOA的权威副本，以及如果该区域的SERIAL自从该数据被缓存后没有改变，那么，可以将此缓存数据的TTL重新设置到该区域的MINIMUM值，如果MINIMUM值更小。仅出于规划目的提及这种应用，尚不推荐这种应用。

4-3-5 区域维护和传送

区域管理者工作的一部分是在所有名称服务器(这些名称服务器对于该区域是权威的)中维护区域。当不可避免的改变发生时，这些改变必须被分发到所有名称服务器。虽然这些分发能够使用FTP或其他特定流程完成，优先考虑的方法是DNS协议的区域传送部分。

自动化区域传送或刷新有通用做法，此通用做法即名称服务器之一是该区域的主(或基本)名称服务器。要在主名称服务器中做出改变，典型通过编辑该区域的主文件实现。编辑之后，管理者指示主服务器加载新的区域。区域的其他非主服务器(或辅助服务器)们定期核查改变(以选定的时间间隔)，并获得新的区域副本，当改变已经发生时。

为了检测改变，辅助服务器仅检验区域SOA的SERIAL字段。除了做出无论什么其他改变之外，只要对区域做出任何改变，区域SOA中的SERIAL字段总是被提出。提出可以是简单的增加，或可以基于写数据和主文件的时间，等等。目的是通过比较序列号，能够确定区域的两个副本中的哪一个是较近期的。序列号的提出和比较使用序列空间算法，所以对区域能够多快更新存在理论上的限制，基本上在序列号覆盖它的32位范围的一半之前旧的副本必须抛弃。在实践中，唯一关注的是此比较操作适当处理围绕最正和最负32位数之间的边界的比较。

区域的SOA RR中的参数控制辅助服务器的定期轮询，这些参数设置最小可接受轮询间隔。这些参数被称作REFRESH、RETRY和EXPIRE。只要辅助服务器中加载新区域，在与新序列的主服务器一致前辅助服务器等待REFRESH秒。如果这个检验不能完成，每隔RETRY秒开始新的检验。此检验是对区域SOA RR的主服务器的简单查询。如果在辅助服务器的区域副本中的序列字段等于由主服务器返回的序列，那么，没有发生过改变，重新启动REFRESH间隔等待。如果辅助服务器发现不可能执行EXPIRE间隔的序列检验，辅助服务器必须假设它的区域副本是过时的，并且抛弃该副本。

当轮询显示区域已经改变，那么辅助服务器必须通过该区域的AXFR请求，请求区域传递。此AXFR可能引起错误，诸如拒绝，但是通常得到来自一系列响应消息的回答。第一个和最后一个消息必须包括区域的顶层权威节点的数据。中间的消息携带来自该区域的所有其他RRs，包括权威的和非权威的RRs。此消息流使得辅助服务器能够构建该区域的副本。因为精确性是根本，AXFR请求必须使用TCP或某种其他的可靠协议。

要求每个辅助服务器对照主服务器执行下述操作，但是也可以选择对照其他辅助服务器执行这些操作。当由于主机故障或网络问题使主服务器不能使用时，或者当辅助服务器访问“中间”辅助服务器的网络比访问主服务器的网络更好时，这个策略能够改善传递处理。

第5章 解析器

5-1 序言

解析器是连接用户程序到域名服务器的程序。在最简单的情况，解析器以子程序调用、系统呼叫等形式，接收来自用户程序的请求(例如，邮件程序，TELNET，FTP)，并用与本地主机数据格式兼容的格式返回期望的信息。

解析器和请求解析器服务的程序位于相同的机器内，但是解析器可能需要咨询在其他主机上的名称服务器。因为解析器可能需要咨询几个名称服务器，或可能有在本地缓存器中的请求信息，解析器完成解析需要的时间数量可能变化很大，从若干毫秒到几秒。解析器非常重要的目标是消除网络延时，和减轻源自大多数请求的名称服务器负载(方法是根据解析器对先前结果的缓存，回答这些请求)。因此，由多进程、多用户、多机器等共享的缓存比非共享的缓存效率更高。

5-2 客户端-解析器接口

5-2-1 典型功能

到解析器的客户端接口受本地主机的惯例影响，但是典型的解析器-客户端接口有3个功能：

1、主机名称到主机地址转换

这个功能常被定义为模仿先前的基于HOSTS.TXT的功能。给定字符串，调用者希望一个或多个32位IP地址。在DNS下，它转换成A类型RRs请求。因为DNS不保存RRs的次序，如果此业务仅可以返回一个选择给客户端，这个功能可以选择分类返回的地址或选择“最好的”地址。注意，推荐多地址返回，但是单地址可能是模拟先前的HOSTS.TXT业务的唯一方法。

2、主机地址到主机名称转换

这个功能常常遵循前面一些功能的格式。给定32位IP地址，调用者希望获得字符串。此IP地址的八位位组被颠倒，用作名称分量，用“IN-ADDR.ARPA”作后缀。PTR类型查询用于获得带主机的主名称的RR。例如，对于对应IP地址1.2.3.4的主机名称请求，查找域名“4.3.2.1.IN-ADDR.ARPA”的PTR RRs。

3、通用查找功能

这个功能检索来自DNS的任意信息，这个功能没有先前系统中的对应物。调用者提供QNAME、QTYPE和QCLASS，调用者希望得到所有匹配的RRs。这个功能常常使用所有RR数据的DNS格式，而不使用本地主机的DNS格式，这个功能返回所有RR内容(例如，TTL)，而不是返回采用本地引用惯例的处理格式。

当解析器执行指定的功能时，它通常向客户端传回下述结果之一：

- 一个或多个RRs给出被请求的数据。

  在这种情况，解析器以适当格式返回回答。

- 名称错误(NE)。

  当引用的名称不存在时发生此情况。例如，用户可能错误录入主机名。

- 数据没有找到错误。

  当引用的名称存在，但是却没有适当类型的数据时发生此情况。例如，应用于邮箱名称的主机地址功能将返回这个错误，因为该名称存在，但是没有地址RR。

注意到在主机名称和地址间转换的一些功能可以将“名称错误”条件和“数据没有发现”错误条件组合成单一类型错误返回是重要的，但是通用功能不应当组合。对此的原因之一是应用们可以首先询问关于名称的一类信息，继之以对于(某个其他类型信息的)相同名称的第二个请求；如果两个错误组合在一起，那么，无用的查询可能减慢应用。

5-2-2 别名

当尝试解析特定的请求时，解析器可能发现所涉及的名称是别名。例如，当解析器找到CNAME RR时，解析器或许发现针对主机名称到地址的转换给出的名称是别名。如果可能，别名条件应当从解析器传回给客户端。

在大多数情况，当解析器遭遇CNAME时，解析器简单以新的名称重新启动查询。然而，当执行通用功能时，解析器不应当推行别名，当CNAME RR匹配查询类型时。这允许询问别名是否存在的查询。例如，如果查询类型是CNAME，用户关注的是CNAME RR自身，不是CNAME RR指向的名称中的RRs。

几个特殊条件可随别名一起发生。因为别名效率低，应当避免多层别名，但是不应当将多层别名看作错误。应当捕获别名环回和指向不存在名称们的别名们，出错条件传回客户端。

5-2-3 临时故障

在不完善的环境，所有解析器偶尔会不能够解析具体的请求。如果由于链路故障或网关问题解析器变得与网络其他部分分开，或者较少发生的一致性失败或特定域的所有服务器都不可使用，这个条件会由解析器引起。

至关重要的是这种条件不应当看作是对于应用的名称或数据不存在错误。这种行为将令人烦恼，并且当邮件系统采用DNS时会引发巨大混乱。

尽管在某些情况通过无限期封锁请求可以应对这种临时问题，这通常不是好的选择，尤其是当客户端是能够转到其他任务上的服务器进程。推荐的解决方案是总是让临时失败作为解析器功能的可能结果之一，即使这样做会使现有HOSTS.TXT功能的仿真更为困难。

5-3 解析器内部

每一种解析器实现采用的算法稍有不同，一般情况比典型的occurances花费更多的逻辑运算来处理各种错误。本节简单讨论推荐的解析器运行基本策略，但是细节留给[RFC-1035]。

5-3-1 末梢解析器

实现解析器的一种选择是从本地机器中移出解析功能，并将其移进支持递归查询的名称服务器。这是在PC(该PC缺乏执行解析器功能的资源)中提供域服务的方便方法，这也能够集中整个本地网络或组织的缓存器。

末梢(解析器)需要的所有其余的东西是名称服务器地址列表，这些名称服务器将执行递归请求。这类解析器估计需要配置文件中的信息，因为这类解析器可能缺乏在域数据库中寻找该信息的经验。用户也需要验证列出的服务器将执行递归业务；名称服务器可以自主决定拒绝为任何或所有客户端执行递归业务。用户应当协商本地系统管理员，以便找到愿意执行这一业务的名称服务器。

这类业务存在某些不足。因为递归请求可能花费任意数量的执行时间，末梢(解析器)对于优化重传间隔，以便处理丢失的UDP分组和失效服务器，可能有困难；如果名称服务器把重新发送理解为新的请求，名称服务器会因太热情于末梢(解析器)很容易超载。使用TCP可能是一种选择，然而TCP很可能将负荷转移到主机能力上，这些能力类似真实解析器的能力。

5-3-2 资源

除了解析器自己的资源以外，解析器也可以共享对区域的访问，这些区域由本地名称服务器维护。这使解析器能够更快速地访问，但是解析器必须当心，绝不能让缓存的信息重写区域数据。在这里的讨论中，术语“本地信息”应该意味着缓存器和这些共享区域一致，应领会权威数据总是优先于缓存数据使用，当二者都存在时。

下述解析器算法假设所有功能已经转换到通用查找功能，并且在解析器中，使用下述数

据结构代表正在进行中的请求的状态：

SNAME	我们正在搜索的域名。
STYPE	该搜索请求的QTYPE。
SCLASS	该搜索请求的QCLASS。
SLIST	一种结构。该结构描述名称服务器和解析器目前正在尝试查询的区域。这个结构保持对解析器目前最好猜测的跟踪，该猜测是关于哪一个名称服务器握有想要信息的猜测；当到达的信息改变该猜测时，最好猜测被更新。这个结构包括区域名称的等价物、该区域已知的名称服务器们、这些名称服务器的已知地址，和历史信息(该历史信息可用于建议哪一个服务器极有可能是下一次尝试的最好服务器)。区域名称等价物是标签数量的匹配计数，此标签数量从根向下沿着SNAME有的、与正在被查询的区域相同的标签算起；这被用作解析器如何“靠近”SNAME的度量。
SBELT	与SLIST格式相同的“安全带(safety belt)”结构，根据配置文件对其初始化。当解析器没有任何指导名称服务器选择的本地信息时，“安全带”结构列出应当使用的服务器。匹配计数将为-1，指出已知没有匹配标签。
CACHE	一种结构，它保存以前响应的结果。因为解析器负责抛弃旧的RRs(TTL超期的RRs)，当在缓存中保存RR时，大多数实现把到达的RRs中规定的间隔转换成某种绝对时间。代替逐个向下计数TTLs，当解析器在搜索过程中碰见这些旧RRs时，解析器仅仅忽略或抛弃它们，或者在周期扫描期间抛弃它们，以便释放由旧RRs占用的存储资源。

5-3-3　算法

顶层算法分为4个步骤：

1、看看回答是否在本地信息中，如果是，将该回答返回给客户端。

2、发现将要向其询问的最好的服务器们。

3、向这些服务器发送查询，直到它们中的一个返回响应。

4、分析该响应，或者：

a、如果该响应回答了问题或者包含名称错误，缓存数据并将该响应返回给客户端。

b、如果该响应包含对其他服务器的更好授权，缓存授权信息，转到步骤2。

c、如果该响应显示CNAME，并且CNAME不是回答本身，缓存CNAME，改变SNAME到CNAME RR中的正则名称，转到步骤1。

d、如果该响应显示服务器故障或其它异常内容，从SLIST中删除该服务器，并转到步骤3。

步骤1为想要的数据搜索缓存器。如果缓存器中存有该数据，假设该数据对于正常使用足够好。某些解析器在用户接口中设有选择，该选择强制解析器忽略缓存的数据并咨询权威服务器。此做法没有作为默认推荐。如果解析器已经直接访问到名称服务器的区域，解析器应当查看是否想要的数据以权威格式存在，如果存在，使用权威数据优先于缓存的数据。

步骤2查找名称服务器，索求数据。一般策略是查找本地可用名称服务器RRs，以SNAME开始，接着是SNAME的父域名称，祖父，一直继续到根。于是，如果SNAME是Mockapetris.ISI.EDU，这一步骤将查找Mockapetris.ISI.EDU的NS RRs，接着是ISI.EDU，接

着是EDU，接着是…，(根)。

这些NS RRs为在SNAME中或在SNAME上的区域列出主机们的名称。复制这些名称到SLIST。使用本地数据设置它们的地址。可能出现地址不能得到情况。这里，解析器有多种选择：最好的选择是，当用可获得的地址继续向前时，开始采用并行解析器进程查找地址。很明显，设计选择和选项是复杂的，是随本地主机能力而变的。向解析器设计者推荐的优先次序是：

1、限制任务(分组发送，开始的并行进程)数量，以便请求不会陷入无限循环，或引起采用其它实现的请求或查询的连锁反应，**即使某个实现已经不正确地配置了某个数据**。

2、如果有可能，返回回答。

3、避免不必要的传递。

4、尽可能迅速地得到回答。

如果搜索NS RRs失败，那么，解析器根据安全带SBELT初始化SLIST。基本思路是，当解析器对应该问哪个服务器没有主意时，解析器应当使用来自配置文件的信息，该配置文件列出了几个预期是有帮助的服务器。虽然有一些特殊情况，通常的选择是根服务器中的两个和主机域的服务器中的两个。每种情况两个的原因是考虑冗余。根服务器将提供对所有域空间的最终访问。如果由于网关或链路故障，使得本地网络脱离互连网络，两个本地服务器将使解析器能够继续解析本地名称。

除了服务器的名称和地址以外，可以分类SLIST数据结构，以便首先使用最好的服务器，并确保以循环的方式使用所有服务器的所有地址。此分类可以是本地网络上(相比其他地址的)优先地址的简单函数，或者可以包括来自过去事件的统计量，诸如先前的响应时间和平均成功率。

步骤3送出查询，直到收到响应。策略是利用每次发送间的超时，围绕所有服务器的所有地址循环。实践中，重要的是使用多归属地主机的所有地址，当响应由多个竞争相同名称服务器的解析器(即使偶尔由单个解析器)使用时，过于激进的重传策略实际上延缓响应。SLIST一般包括控制超时的数据值，以及保持跟踪先前传输的数据值。

步骤4涉及分析响应。解析器在它的响应分析中应当保持高度质疑。解析器也应当使用响应中的ID字段验证响应匹配它发出的查询。理想的回答是来自该查询的服务器权威的回答，该回答或者给出请求的数据，或者给出名称错误。如果数据的TTL远大于0，数据被传回用户，并被输进缓存以备将来使用。

如果响应显示授权，解析器应当验证该授权比在SLIST中的服务器们“更接近于”回答。通过将SLIST中的匹配计数与根据该授权中的SNAME和NS RRs计算出的计数比较，可做到此。如果不是“更接近于”回答，该响应是虚假的，应当忽略。如果授权合法，应当缓存NS授权RRs和服务器们的任何地址RRs。将该名称服务器输入进SLIST，搜索重新开始。

如果响应包括CNAME，搜索在此CNAME重新开始，除非该响应有正则名称数据，或如果CNAME是回答本身。

细节和实现提示在[RFC-1035]中讨论。

第6章　场景

在我们的样本域空间中，假设我们希望对根、MIL、EDU、MIT.EDU和ISI.EDU区域进行分开的管理控制。我们或许分配名称服务器如图2所示。

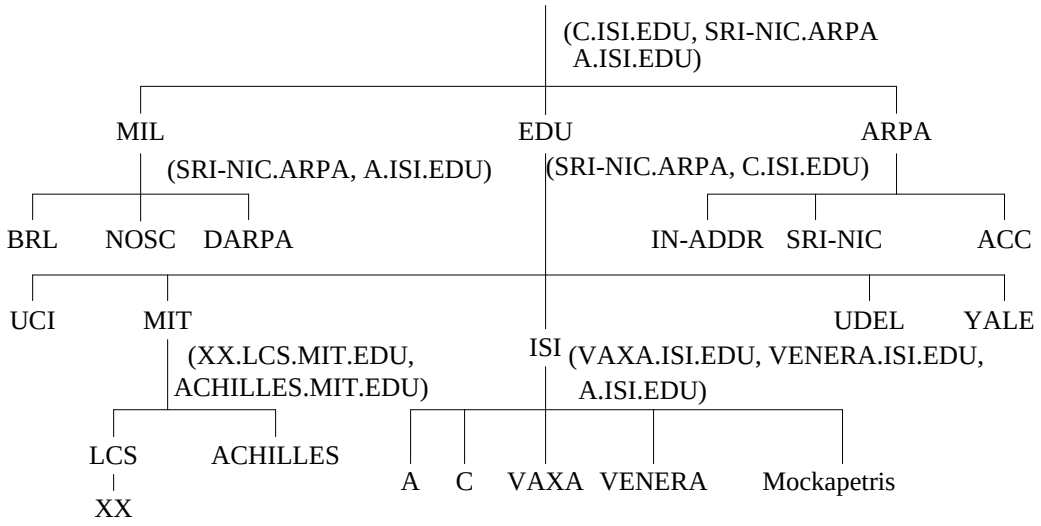

图2　名称服务器分配举例

在这个例子中，权威名称服务器显示在域树中的点旁边的圆括号内，在该点是假设控制。

于是，根名称服务器是在C.ISI.EDU、SRI-NIC.ARPA和A.ISI.EDU上。MIL域由SRI-NIC.ARPA 和A.ISI.EDU提供服务。EDU域由SRI-NIC.ARPA和C.ISI.EDU提供服务。注意，服务器们可以有连续的或不相交的区域。在本场景中，C.ISI.EDU有在根域和EDU域的连续区域。A.ISI.EDU有在根域和MIL域的连续区域，但是也有在ISI.EDU的非连续区域。

6-1　C.ISI.EDU名称服务器

C.ISI.EDU是IN类根域、MIL域和EDU域的名称服务器，C.ISI.EDU有这些域的区域。根域的区域数据或许是：

```
.           IN      SOA     SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                                870611          ; 顺次的
                                1800            ; 每30 min刷新
                                300             ; 每5 min重试
                                604800          ; 一周后到期
                                86400)          ; 一天的最小值
                        NS      A.ISI.EDU.
                        NS      C.ISI.EDU.
                        NS      SRI-NIC.ARPA.
MIL.        86400       NS      SRI-NIC.ARPA.
            86400       NS      A.ISI.EDU.
EDU.        86400       NS      SRI-NIC.ARPA.
            86400       NS      C.ISI.EDU
SRI-NIC.ARPA.           A       26.0.0.73
                        A       10.0.0.51
                        MX      0 SRI-NIC.ARPA.
                        HINFO   DEC-2060 TOPS20
ACC.ARPA.               A       26.6.0.65
                        HINFO   PDP-11/70 UNIX
                        MX      10 ACC.ARPA.
```

```
USC-ISIC.ARPA.    CNAME       C.ISI.EDU.
73.0.0.26.IN-ADDR.ARPA.       PTR       SRI-NIC.ARPA.
65.0.6.26.IN-ADDR.ARPA.       PTR       ACC.ARPA.
51.0.0.10.IN-ADDR.ARPA.       PTR       SRI-NIC.ARPA.
52.0.0.10.IN-ADDR.ARPA.       PTR       C.ISI.EDU.
103.0.3.26.IN-ADDR.ARPA.      PTR       A.ISI.EDU.
A.ISI.EDU. 86400     A          26.3.0.103
C.ISI.EDU. 86400     A          10.0.0.52
```

这个数据当它出现在主文件中时很有代表性。大多数RRs是单行条目；这里唯一的例外是SOA RR，它使用“(“to start a multi-line RR and”)”显示多行RR的结束。因为在区域中所有RRs的类必须相同，仅区域中的第1个RR需要规定类。当名称服务器加载区域时，它强制所有权威RRs的TTL至少是SOA的MINIMUM字段，这里是86400秒，或者是一天。标记MIL域和EDU域的授权的NS RRs，以及服务器主机地址的胶(glue)RRs，不是该区域中权威数据的一部分，并且因此有明确的TTLs。

根节点有4个RRs：包括描述此根区域的SOA和3个列出该根的名称服务器的NS RRs。在SOR RR中的数据描述区域管理。区域数据在SRI-NIC.ARPA主机上维护，区域的负责方是HOSTMASTER@SRI-NIC.ARPA。在SOA中的关键项是86400秒最小TTL，它意味着区域内的所有权威数据至少有该TTL，尽管可以明确规定更高的值。

MIL域和EDU域的NS RRs标记根区域及MIL区域和EDU区域间的边界。注意，在本例中，碰巧支持较低区域的名称服务器也支持根域区。

或许相对于原始EDU规定EDU区域的主文件。EDU域的区域数据或许是：

```
EDU.      IN SOA SRI-NIC.ARPA.      HOSTMASTER.SRI-NIC.ARPA. (
                                    870729      ; 顺次的
                                    1800        ; 每30min刷新
                                    300         ; 每5min重试
                                    604800      ; 一周后到期
                                    86400       ; 一天的最小值
                                    )
                          NS SRI-NIC.ARPA.
                          NS C.ISI.EDU.
UCI 172800 NS ICS.UCI
                          172800 NS ROME.UCI
ICS.UCI 172800 A 192.5.19.1
ROME.UCI 172800 A 192.5.19.31
ISI 172800 NS VAXA.ISI
                          172800 NS A.ISI
                          172800 NS VENERA.ISI.EDU.
VAXA.ISI 172800 A 10.2.0.27
                          172800 A 128.9.0.33
VENERA.ISI.EDU. 172800 A 10.1.0.52
                          172800 A 128.9.0.32
A.ISI 172800 A 26.3.0.103
UDEL.EDU. 172800 NS LOUIE.UDEL.EDU.
                          172800 NS UMN-REI-UC.ARPA.
```

LOUIE.UDEL.EDU. 172800 A 10.0.0.96
172800 A 192.5.39.3
YALE.EDU. 172800 NS YALE.ARPA.
YALE.EDU. 172800 NS YALE-BULLDOG.ARPA.
MIT.EDU. 43200 NS XX.LCS.MIT.EDU.
43200 NS ACHILLES.MIT.EDU.
XX.LCS.MIT.EDU. 43200 A 10.0.0.44
ACHILLES.MIT.EDU. 43200 A 18.72.0.8

注意，这里使用相对名称。使用相对名称规定ISI.EDU.的所有者名称，例如是名称服务器RR内容中的两个。在主文件中相对域名和绝对域名可以自由混用。

6-2 标准查询举例

下面的查询和响应演示了名称服务器的行为。除非另有说明，在首部中，查询不含期望递归(RD)。注意，对非递归查询的回答取决于被询问的服务器，但是不取决于提问者的身份。

6-2-1 QNAME=SRI-NIC.ARPA, QTYPE=A

查询或许如图3所示。

首部	OPCODE=SQUERY
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
回答	<空>
权威	<空>
附加	<空>

图3 QNAME=SRI-NIC.ARPA, QTYPE=A查询

来自C.ISI.EDU的响应或许如图4所示。

首部	OPCODE=SQUERY, RESPONSE, AA
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
回答	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 86400 IN A 10.0.0.51
权威	<空>
附加	<空>

图4 来自C.ISI.EDU的响应

此响应的首部看起来像查询的首部，只是RESPONSE位被置1，指出这是响应消息不是查询消息，以及权威回答(Authoritative Answer, AA)位被置1指出在回答部分中的地址RRs来自权威数据。响应的问题部分匹配查询的问题部分。

如果发送相同的查询到某个其他的服务器，该服务器不是SRI-NIC.ARPA的权威服务器，

响应或许如图5所示。

首部	OPCODE=SQUERY, RESPONSE
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
回答	SRI-NIC.ARPA. 1777 IN A 10.0.0.51 1777 IN A 26.0.0.73
权威	<空>
附加	<空>

图5　来自非权威服务器的响应

这个响应在两方面不同于前一个响应：首部没有AA置1，TTLs不同。由此可推断数据不是来自区域，而是来自缓存。权威TTL和这里的TTL间的不同是由于缓存中数据的迟滞。在回答部分中RRs顺序不同没有意义。

6-2-2　QNAME=SRI-NIC.ARPA, QTYPE=*

此查询类似前一个查询，但是使用*的QTYPE，将收到来自C.ISI.EDU的如图6所示的响应。

首部	OPCODE=SQUERY, RESPONSE, AA
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*
回答	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 A 10.0.0.51 MX 0 SRI-NIC.ARPA. HINFO DEC-2060 TOPS20
权威	<空>
附加	<空>

图6　来自C.ISI.EDU的响应(QTYPE=*)

如果类似的查询被定向到两个名称服务器，这两个名词服务器不是SRI-NIC.ARPA的权威服务器，响应或许如图7所示。

首部	OPCODE=SQUERY, RESPONSE
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*
回答	SRI-NIC.ARPA. 12345 IN A 26.0.0.73 A 10.0.0.51
官方	<空>
补充	<空>

图7　来自不是SRI-NIC.ARPA权威服务器的响应之一

和如图8所示。

首部	OPCODE=SQUERY, RESPONSE
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*
回答	SRI-NIC.ARPA. 1290 IN HINFO DEC-2060 TOPS20
权威	<空>
附加	<空>

图8　来自不是SRI-NIC.ARPA权威服务器的响应之二

这些回答没有一个有AA置1，所以，都不是来自权威数据的响应。不同的内容和不同的TTLs暗示这两个服务器在不同时刻缓存数据，并且第1个服务器缓存了给QTYPE=A查询的响应，第2个服务器缓存了给HINFO查询的响应。

6-2-3　QNAME=SRI-NIC.ARPA, QTYPE=MX

这类查询或许起因于邮件收发程序尝试查找邮件目的地HOSTMASTER@SRI-NIC.ARPA的路由信息。来自C.ISI.EDU的响应或许如图9所示。

首部	OPCODE=SQUERY, RESPONSE, AA
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=MX
回答	SRI-NIC.ARPA. 86400 IN MX 0 SRI-NIC.ARPA.
权威	<空>
附加	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 10.0.0.51

图9　来自C.ISI.EDU的对于邮件查找的响应

这个响应在响应的回答部分包括MX RR。附加部分包括地址RRs，因为在C.ISI.EDU的名称服务器猜测请求者将需要这些地址，以便适当地使用由MX携带的信息。

6-2-4　QNAME=SRI-NIC.ARPA, QTYPE=NS

C.ISI.EDU或许如图10所示回答这个查询。

首部	OPCODE=SQUERY, RESPONSE, AA
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=NS
回答	<空>
权威	<空>
附加	<空>

图10　C.ISI.EDU的回答

响应和查询间的唯一不同是首部中的AA位和RESPONSE位。这条响应可以翻译为该服务器是该名称的权威服务器，并且该名称存在，但是那里没有NS类型的RRs。

6-2-5　QNAME=SIR-NIC.ARPA, QTYPE=A

如果用户错误敲入主机名称，我们或许看到这类查询。

C.ISI.EDU将用如图11所示内容回答该查询。

首部	OPCODE=SQUERY, RESPONSE, AA, RCODE=NE
问题	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A
回答	<空>
权威	. SOA SRI-NIC.ARPA.　HOSTMASTER.SRI-NIC.ARPA. 870611　1800　300　604800　86400
附加	<空>

图11　C.ISI.EDU对QNAME=SIR-NIC.ARPA, QTYPE=A的响应

这个响应指出名称不存在。这个条件在首部的响应代码(RCODE)部分暗示。

在权威部分中的SOA RR是可选的否定缓存信息，它使得使用这个响应的解析器能够假设该名称在SOA MINIMUM (86400)秒内不存在。

6-2-6　QNAME=BRL.MIL, QTYPE=A

如果这个查询发送给C.ISI.EDU，响应将如图12所示。

首部	OPCODE=SQUERY, RESPONSE			
问题	QNAME=BRL.MIL, QCLASS=IN, QTYPE=A			
回答	<空>			
权威	MIL.	86400 IN NS	SRI-NIC.ARPA.	
		86400 NS	A.ISI.EDU.	
附加	A.ISI.EDU.	A	26.3.0.103	
	SRI-NIC.ARPA.	A	26.0.0.73	
		A	10.0.0.51	

图12　C.ISI.EDU对QNAME=BRL.MIL, QTYPE=A查询的响应

这个响应有空的回答部分，只是不是权威的，所以这个响应是转介。在C.ISI.EDU上的名称服务器，认出这个响应不是MIL域权威的，将此请求者转托给在A.ISI.EDU和SRI-NIC.ARPA上的服务器们，它知道它们是MIL域的权威。

6-2-7　QNAME=USC-ISIC.ARPA, QTYPE=A

对来自A.ISI.EDU的这个查询的响应将如图13所示。

首部	OPCODE=SQUERY, RESPONSE, AA			
问题	QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A			
回答	USC-ISIC.ARPA.,	86400 IN CNAME	C.ISI.EDU.	
	C.ISI.EDU.	86400 IN A	10.0.0.52	
权威	<空>			
附加	<空>			

图13　QNAME=USC-ISIC.ARPA, QTYPE=A查询的响应

注意，首部中的AA位保证匹配QNAME的数据是权威的，但对C.ISI.EDU的数据是否是权威的没做任何说明。这个完整的响应是可能的，因为A.ISI.EDU恰好对以下两个域是权威的，这两个域分别是发现USC-ISIC.ARPA的ARPA域，和发现C.ISI.EDU数据的ISI.EDU域。如果同样的查询发送给C.ISI.EDU，如果C.ISI.EDU在它的缓存中有它自己的地址，C.ISI.EDU的响应或许与上面显示的相同，但是也可能如图14所示。

首部	OPCODE=SQUERY, RESPONSE, AA			
问题	QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A			
回答	USC-ISIC.ARPA.	86400	IN CNAME	C.ISI.EDU.
权威	ISI.EDU.	172800	IN NS	VAXA.ISI.EDU.
			NS	A.ISI.EDU.
			NS	VENERA.ISI.EDU.
附加	VAXA.ISI.EDU.	172800	A	10.2.0.27
		172800	A	128.9.0.33
	VENERA.ISI.EDU.	172800	A	10.1.0.52
		172800	A	128.9.0.32
	A.ISI.EDU.	172800	A	26.3.0.103

图14　6-2-7中C.ISI.EDU的响应

这个响应包括USC-ISIC.ARPA别名的权威响应，加上到ISI.EDU名称服务器的转介。假如查询的是正在被询问的名称服务器的主机名称，这种响应不是非常合适，但对于其他别名可能是常见的。

6-2-8　QNAME=USC-ISIC.ARPA, QTYPE=CNAME

如果这个查询或者发送给A.ISI.EDU，或者发送给C.ISI.EDU，响应可能如图15所示。

首部	OPCODE=SQUERY, RESPONSE, AA			
问题	QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A			
回答	USC-ISIC.ARPA.	86400	IN CNAME	C.ISI.EDU.
权威	<空>			
附加	<空>			

图15　对QNAME=USC-ISIC.ARPA, QTYPE=CNAME查询的响应

因为QTYPE=CNAME，CNAME RR自己回答此查询，并且对于C.ISI.EDU，该名称服务器不尝试任何查找。(附加部分有可能除外。)

6-3　解析举例

下述各例演示解析器必须为它的客户端执行的操作。我们假设解析器一开始没有缓存，这或许是系统启动后的情况。我们进一步假设系统不位于数据中的主机之一，假设主机位于网络26上的某处，以及该系统的安全带(SBELT)数据结构有下述信息：

匹配计数　=　-1

SRI-NIC.ARPA.　26.0.0.73　10.0.0.51

A.ISI.EDU.　26.3.0.103

这个信息规定：尝试的服务器们、这些服务器的地址，以及值为-1的匹配计数，这个信息表明这些服务器不是非常靠近目标。注意，-1不被假设是精确的接近程度度量，仅仅是使算法的后期阶段能够工作的一个值。

下述举例演示缓冲器的使用，所以每个例子假设先前的请求已经完成。

6-3-1　解析ISI.EDU.的MX

假设到解析器的第1个请求来自本地邮件收发程序，它有PVM@ISI.EDU的邮件。接着该邮件收发程序或许询问ISI.EDU域名的MX RRs类型。解析器将在它自己的缓存中查找在ISI.EDU中的MX RRs，但是空的缓存没有帮助。解析器认识到它需要查询外地服务器，并试着确定那些最好的可供查询的服务器。这个搜索将查找ISI.EDU域、EDU域和根域的NS RRs。对缓存器的这些搜索也将失败。最后，解析器使用来自SBELT的信息，复制该信息到它的SLIST结构。

在此点，解析器需要从3个可得到的地址中捡出一个进行尝试。假设解析器是在网络26上，它应当或者是把26.0.0.73，或者是把26.3.0.103作为它的首次选择。它接着应当发出形如图16所示的查询。

首部	OPCODE=SQUERY
问题	QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX
回答	<空>
权威	<空>
附加	<空>

图16　6-3-1中解析器发出的查询

解析器接着将等待它的查询的响应或超时。如果发生超时，解析器将尝试不同的服务器，接着是同一服务器的不同地址，最后重新尝试已经试过的地址。最终，解析器或许收到来自SRI-NIC.ARPA的如图17所示的响应。

首部	OPCODE=SQUERY, RESPONSE				
问题	QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX				
回答	<空>				
权威	ISI.EDU.	172800	IN	NS	VAXA.ISI.EDU.
				NS	A.ISI.EDU.
				NS	VENERA.ISI.EDU.
附加	VAXA.ISI.EDU.	172800		A	10.2.0.27
		172800		A	128.9.0.33
	VENERA.ISI.EDU.	172800		A	10.1.0.52
		172800		A	128.9.0.32
	A.ISI.EDU.	172800		A	26.3.0.103

图17　来自SRI-NIC.ARPA的响应

解析器将注意到，在此响应中的信息给出了比它现有SLIST更近的到ISI.EDU的授权(因为这一信息匹配3个标签)。解析器接着将缓存这个响应中的信息，并用该信息建立新的SLIST：

匹配计数(Match count) = 3

A.ISI.EDU.	26.3.0.103	
VAXA.ISI.EDU.	10.2.0.27	128.9.0.33
VENERA.ISI.EDU.	10.1.0.52	128.9.0.32

A.ISI.EDU出现在这个列表中，也出现在前一个列表中，但这纯粹是巧合。解析器将再次启动发送并等待响应。最终，解析器将获得如图18所示的回答。

首部	OPCODE=SQUERY, RESPONSE,AA				
问题	QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX				
回答	ISI.EDU.			MX 10	VENERA.ISI.EDU.
				MX 20	VAXA.ISI.EDU.
权威	<空>				
附加	VAXA.ISI.EDU.	172800		A	10.2.0.27
		172800		A	128.9.0.33
	VENERA.ISI.EDU.	172800		A	10.1.0.52
		172800		A	128.9.0.32

图18　解析器最终获得的回答

解析器将把这个信息添加到它的缓存，并返回此MX RRs给它的客户端。

6-3-2　获得地址26.6.0.65的主机名

解析器将把“获得地址26.6.0.65的主机名”翻译成对于65.0.6.26.IN-ADDR.ARPA的PTR RRs的请求。这个信息不在缓存中，所以解析器将查找准备询问的外地服务器。没有服务器匹配，所以解析器将再次使用SBELT。(注意，ISI.EDU域的服务器们位于缓存中，但是ISI.EDU不是65.0.6.26.IN-ADDR.ARPA的祖宗，所以使用SBELT。)

因为这个请求位于SBELT内两个服务器的权威数据中，最终返回的一个响应将如图19所示。

首部	OPCODE=SQUERY, RESPONSE, AA
问题	QNAME=65.0.6.26.IN-ADDR.ARPA., QCLASS=IN, QTYPE=PTR
回答	65.0.6.26.IN-ADDR.ARPA. PTR ACC.ARPA.
权威	<空>
附加	<空>

图19　最终返回的一个响应

6-3-3　获得poneria.ISI.EDU的主机地址

这个请求将被翻译成poneria.ISI.EDU的A类型请求。解析器将不寻找这个名称的任何缓存数据，但是当它查找准备询问的外地服务器时，它将在缓存中发现ISI.EDU的NS RRs。使用这条数据，解析器将构建如下格式的SLIST：

匹配计数(Match count) = 3

A.ISI.EDU. 26.3.0.103

VAXA.ISI.EDU. 10.2.0.27 128.9.0.33

VENERA.ISI.EDU. 10.1.0.52

根据解析器按优先权排序它的选择的假设，首先列出A.ISI.EDU，并且A.ISI.EDU在同一网络上。

这些服务器之一将回答此查询。

第7章　参考文献和参考书目

[Dyer 87] Dyer, S., and F. Hsu, "Hesiod", Project Athena Technical Plan - Name Service, April 1987, version 1.9.
Describes the fundamentals of the Hesiod name service.

[IEN-116] J. Postel, "Internet Name Server", IEN-116, USC/Information Sciences Institute, August 1979.
A name service obsoleted by the Domain Name System, but still in use.

[Quarterman 86] Quarterman, J., and J. Hoskins, "Notable Computer Networks", Communications of the ACM, October 1986, volume 29, number 10.

[RFC-742] K. Harrenstien, "NAME/FINGER", RFC-742, Network Information Center, SRI International, December 1977.

[RFC-768] J. Postel, "User Datagram Protocol", RFC-768, USC/Information Sciences Institute, August 1980.

[RFC-793] J. Postel, "Transmission Control Protocol", RFC-793, USC/Information

Sciences Institute, September 1981.

[RFC-799] D. Mills, "Internet Name Domains", RFC-799, COMSAT, September 1981.
Suggests introduction of a hierarchy in place of a flat name space for the Internet.

[RFC-805] J. Postel, "Computer Mail Meeting Notes", RFC-805, USC/Information Sciences Institute, February 1982.

[RFC-810] E. Feinler, K. Harrenstien, Z. Su, and V. White, "DOD Internet Host Table Specification", RFC-810, Network Information Center, SRI International, March 1982.
Obsolete. See RFC-952.

[RFC-811] K. Harrenstien, V. White, and E. Feinler, "Hostnames Server", RFC-811, Network Information Center, SRI International, March 1982.
Obsolete. See RFC-953.

[RFC-812] K. Harrenstien, and V. White, "NICNAME/WHOIS", RFC-812, Network Information Center, SRI International, March 1982.

[RFC-819] Z. Su, and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC-819, Network Information Center, SRI International, August 1982.
Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-821] J. Postel, "Simple Mail Transfer Protocol", RFC-821, USC/Information Sciences Institute, August 1980.

[RFC-830] Z. Su, "A Distributed System for Internet Name Service", RFC-830, Network Information Center, SRI International, October 1982.
Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC-882, USC/Information Sciences Institute, November 1983. Superceeded by this memo.

[RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC-883, USC/Information Sciences Institute, November 1983.
Superceeded by this memo.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC-920, USC/Information Sciences Institute October 1984.
Explains the naming scheme for top level domains.

[RFC-952] K. Harrenstien, M. Stahl, E. Feinler, "DoD Internet Host Table Specification", RFC-952, SRI, October 1985.
Specifies the format of HOSTS.TXT, the host/address table replaced by the DNS.

[RFC-953] K. Harrenstien, M. Stahl, E. Feinler, "HOSTNAME Server", RFC-953, SRI, October 1985.
This RFC contains the official specification of the hostname server protocol, which is obsoleted by the DNS. This TCP based protocol accesses information stored in the RFC-952 format, and is used to

obtain copies of the host table.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC-973, USC/Information Sciences Institute, January 1986. Describes changes to RFC-882 and RFC-883 and reasons for them. Now obsolete.

[RFC-974] C. Partridge, "Mail routing and the domain system", RFC-974, CSNET CIC BBN Labs, January 1986. Describes the transition from HOSTS.TXT based mail addressing to the more powerful MX system used with the domain system.

[RFC-1001] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and Methods", RFC-1001, March 1987. This RFC and RFC-1002 are a preliminary design for NETBIOS on top of TCP/IP which proposes to base NetBIOS name service on top of the DNS.

[RFC-1002] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed Specifications", RFC-1002, March 1987.

[RFC-1010] J. Reynolds and J. Postel, "Assigned Numbers", RFC-1010, USC/Information Sciences Institute, May 1987 Contains socket numbers and mnemonics for host names, operating systems, etc.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition", RFC-1031, November 1987. Describes a plan for converting the MILNET to the DNS.

[RFC-1032] M. K. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC-1032, November 1987. Describes the registration policies used by the NIC to administer the top level domains and delegate subzones.

[RFC-1033] M. K. Lottor, "Domain Administrators Operations Guide", RFC-1033, November 1987. A cookbook for domain administrators.

[Solomon 82] M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server", Computer Networks, vol 6, nr 3, July 1982. Describes a name service for CSNET which is independent from the DNS and DNS use in the CSNET.

原文索引