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

Network Working Group
Request for Comments: 1035

P. Mockapetris
ISI

November 1987

Obsoletes: RFCs 882, 883, 973

域名 --- 实现及标准

目录

第 1 章 本备忘录状态

本 RFC 介绍域系统和协议细节，并假设读者熟悉在姊妹篇 RFC“域名 - 概念和设施”[RFC-1034]中讨论的概念。

域系统是正式协议的功能和数据类型的混合体，域系统是仍然处于试验中的功能和数据

类型的混合体。因为域系统有意做成可扩展的，应当总是希望新的数据类型和试验行为成为超出正式协议的系统的组成部分。正式协议部分包括标准查询、响应和互联网类 RR 数据格式(例如，主机地址)。自从以前的一些 RFC 发表以来，几个定义已经改变，所以某些以前的定义已被废除。

在这些 RFCs 中，清晰标出了试验的和废除的特征，应当谨慎使用这些信息。
读者应特别小心，不能依赖在目前或完成的例子中出现的值，因为这些举例的目的主要是教学。这个备忘录的分发不受限制。

第 2 章　序言

2-1　综述

域名的目标是采用这样一种方法(采用这种方法，名称可以用于不同主机、网络、协议族、互连网络和管理组织。)，提供命名资源的机制。

从用户的观点，域名作为参数，对于称为解析器的本地代理是有用的，解析器检索与域名关联的信息。于是，用户或许询问与特定域名关联的主机地址或邮件信息。为使用户能够请求特定类型的信息，适当的查询类型被传递给有该域名的解析器。对于用户来说，域树是单一信息空间；解析器负责隐藏来自用户的、名称服务器间的数据分布。

从解析器的观点，构成域空间的数据库在不同的名称服务器间分布。尽管特定的数据项被重复保存在两个或多个名称服务器中，域空间的不同部分保存在不同的名称服务器中。开始时解析器至少有一个名称服务器的知识。当解析器处理用户查询时，它针对用户查询的信息询问已知的名称服务器；作为回报，解析器或者收到想要的信息，或者收到去另一个名称服务器的转介。使用这些转介，解析器了解其他名称服务器的身份和内容。解析器负责处理域空间的分布，负责通过咨询在其他服务器中的冗余数据库，应对名称服务器失效的影响。

名称服务器管理两种数据。第一种是在设置中被约束的称为区域(zone)的数据；每个区域是域空间特定“被切断的(pruned)”子树的完整数据库。这个数据被称为权威的。名称服务器定期检验，以便确认它的区域是最新的，如果不是这样，从保存在本地或另一个名称服务器中的主文件中获得被更新区域的新副本。第二种数据是由本地解析器获得的缓存数据。这个数据可能是不完整的，但是当重复访问非本地数据时它可以改善检索处理的性能。缓存数据最终被超时机制抛弃。

这个功能性结构隔离了用户接口问题、失败恢复问题和在解析器中分发的问题，隔离了名称服务器中数据库更新问题和刷新问题。

2-2　一般配置

主机能够以多种方法分享域名服务器，取决于或者主机运行检索来自域系统的信息的程序、运行检索来自名称服务器(这些服务器回答来自其他主机的查询)信息的程序，或者主机运行两种程序功能的各种组合。最简单和或许也是最典型的配置如图 1 所示。

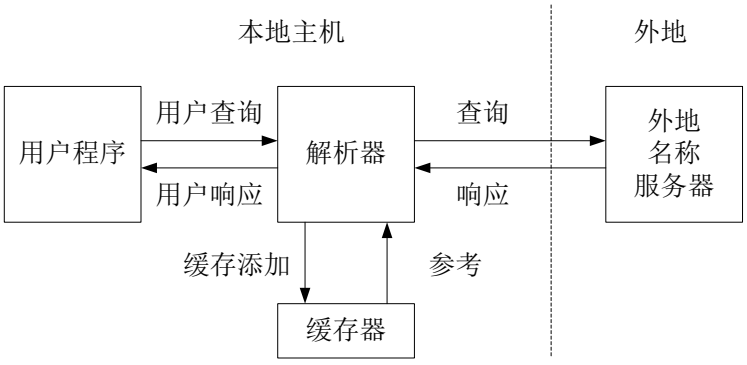

图 1　最简单和最典型的配置

用户程序通过解析器与名称空间互动;用户查询和响应格式是主机和主机操作系统特有的。用户查询一般是操作系统调用，解析器和它的缓存器是主机操作系统的一部分。能力较弱的主机可以选择将解析器作为子例行程序，该子例行程序被挂到每个需要它的服务的程序上。解析器用它们通过查询本地缓存器和外地名称服务器获得的信息回答用户查询。

注意，解析器或许必须对几个不同的外地名称服务器进行多个查询，以便回答特定用户的查询，因此，用户查询的解析可能涉及对几个网络的访问，和任意长的时间。对外地名称服务器的查询和相应的响应采用本备忘录中介绍的标准格式，可能是数据报。

根据名称服务器的能力，它可以是在专用计算机上的独立程序，或在大型分时主机上的一个或多个进程。一种简单配置或许如图 2 所示。

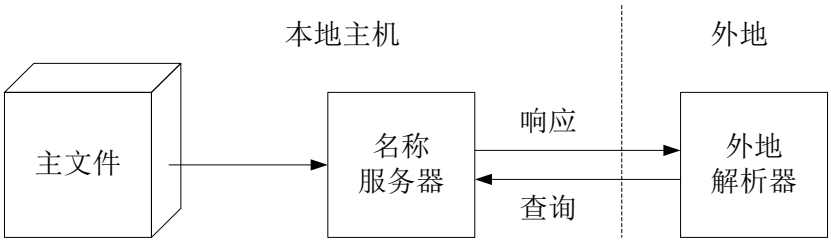

图 2　一种简单配置

这里，主名称服务器通过从它的本地文件系统读主文件，获得有关一个或多个区域的信息，并回答从外地解析器传来的有关那些区域的查询。

此 DNS 请求由不止一个名称服务器冗余支持的所有区域。指定的辅助服务器们可以获得区域，并使用 DNS 区域传送协议，根据主服务器，检查更新。这个配置如图 3 所示。

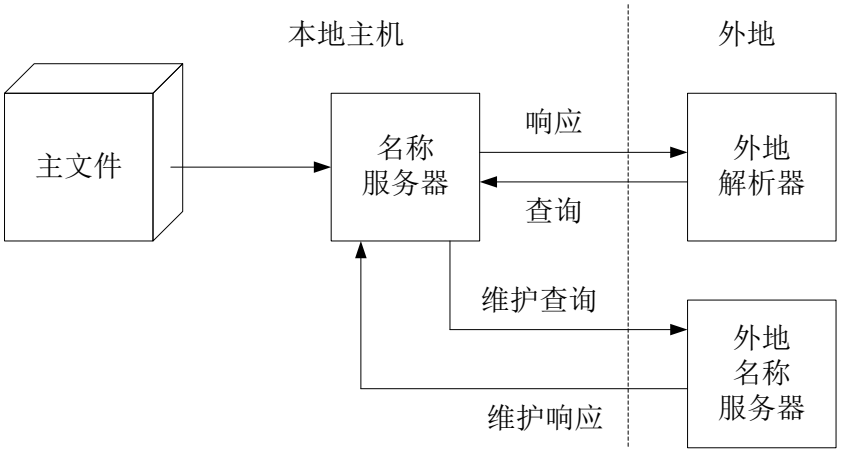

图 3　DNS 请求由多个名称服务器冗余支持的所有区域配置

在这个配置中，名称服务器定期建立到外地名称服务器的虚电路，以便获得区域的副本或证实现有的副本没有改变。发送用于这些维护行动的消息遵循与查询和响应使用的相同的格式，但是消息序列有所不同。

全面支持域名系统各个方面的主机中的信息流如图 4 所示。

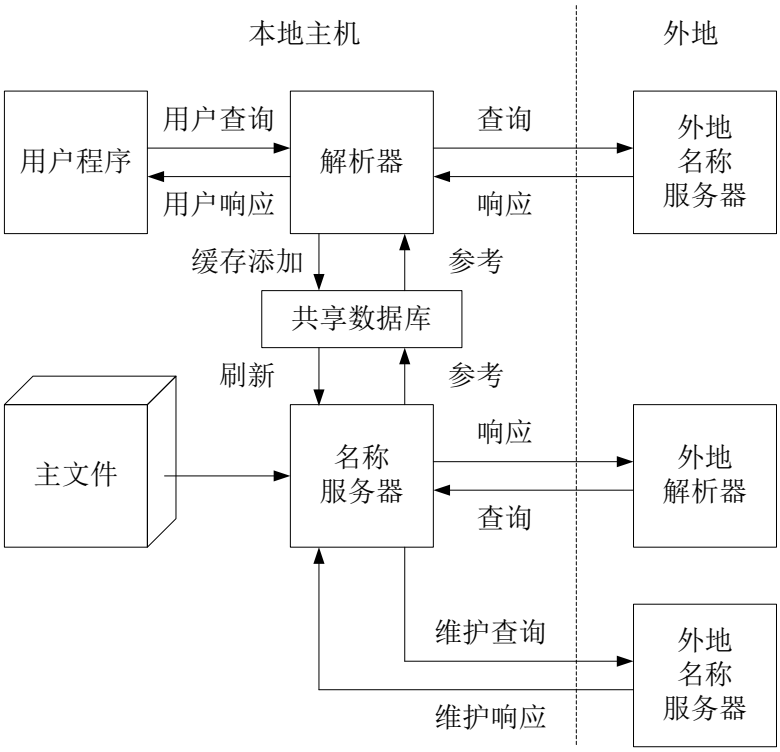

图 4　全面支持域名系统各个方面的主机中的信息流

共享数据库持有本地名称服务器和解析器的域空间数据。共享数据库的内容一般是由名称服务器定期更新操作维护的权威数据，和来自先前解析器请求的缓存数据的混合体。域数据的结构，以及名称服务器和解析器之间同步的必要性暗示这个数据库的一般特点，但是实际格式取决于本地实现者。

信息流也可能被裁减，以便一组主机共同进行优化行动。有时，这样做是为了给能力较弱主机卸载，以便它们不必执行整个解析器。这样做可能是适当的，尤其是对 PC 或希望尽量减小被要求的新网络代码的数量的主机。根据集中缓存有更高命中率的假设，这一方案也使得一组主机能够共享少量缓存器，而不是维护大量独立的缓存器。在这两种情况，在一个或多个已知执行那项业务的名称服务器中，解析器被用末梢解析器取代，末梢解析器充当到位于递归服务器的解析器的前端。带末梢解析器的信息流结构如图 5 所示。

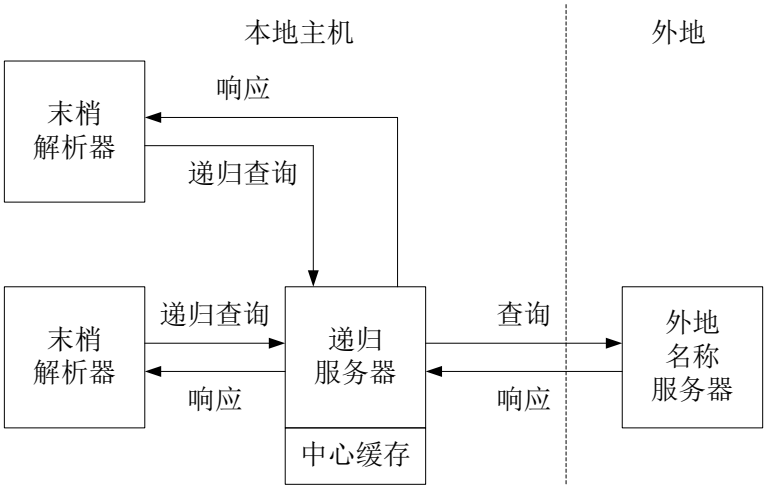

图 5　带末梢解析器的信息流结构

注意，在任何情况，考虑到可靠性，只要有可能，域成员总是被复制。

2-3 惯例

域系统有几个处理低层次(但是更基础)问题的惯例。虽然实现者**在他自己的系统内**有不遵守这些惯例的自由，然而，在**所有**可由其他主机观察到的行为中，他必须遵守这些惯例。

2-3-1 首选的名称句法

此 DNS 规范打算使构建的域名规则尽可能通用。这一想法的精髓是任何现有对象的名称，经过尽量小的改变，能够表述为域名。

然而，当为对象分配域名时，谨慎的用户将选择既满足域系统规则，又满足对象任何现有的规则的名称，无论这些规则是公开出版的还是由现有程序暗示的。

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。当设置新的主机名时，应当遵守旧的 HOSTS.TXT 规则。这可避免当旧软件被转换为用户域名时出现的问题。

就许多使用域名的应用(例如，邮件，TELNET)来说，下述句法产生的问题很少。

<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个字母表字符中的任何一个(A到Z的大写和a到z的小写)

<digit> ::= 0 到 9 十个数字中的任何一个

注意，尽管在域名中大小写字母是允许的，但是对大小写不做区别。即，两个名称有相同的拼写，但有不同的大小写，被看作是相同的。

标签必须遵守 ARPANET 主机名称规则。它们必须以字母开始，以字母或数字结束，内部字符仅可以是字母、数字和连字符号。对长度也有某些限制。标签必须是 63 个字符或更少。

例如，下述字符串等同于互联网中的主机：

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

2-3-2 数据传送顺序

在本文件中介绍的首部和数据的传送顺序被分解到八位位组层次。每当图表显示一组八位位组时，这些八位位组的发送顺序采用以英语阅读它们时通常采用的顺序。例如，在图 6 中，八位位组以它们编号的顺序发送。

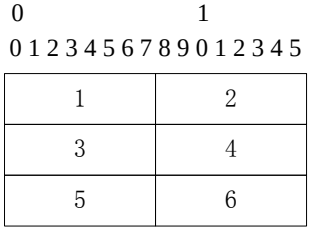

图 6 八位位组的发送顺序

每当八位位组代表数字量时，图中最左边位是高阶或最高有效位。即，标记为 0 的位是最高有效位。例如，图 7 是值 170(十进制)的二进制值表示。

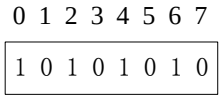

图 7　值 170(十进制)的二进制值表示

类似，每当多八位位组字段代表数字量时，整个字段的最左边位是最高有效位。当传送多八位位组量时，首先传送最高有效八位位组。

2-3-3　字符的大小写

作为此正式协议一部分的 DNS 的所有部分，字符串(例如，标签，域名，等)间的所有比较均不区分大小写。目前，这个规则在整个域系统执行，没有例外。然而，将来超出目前应用的添加，可能需要在名称中使用完全的二进制八位位组的能力，所以，尝试用 7 位 ASCII 码保存域名，或者尝试使用特殊字节终止标签，等，应当避免。

当数据进入域系统时，只要有可能，它的原始大小写应当被保留。在有些情况下这难以做到。例如，如果两个 RRs 被保存在数据库中，一个为 x.y，另一个为 X.Y，它们实际上被保存在该数据库的同一位置，因此，只有一种大小写被保存。基本原则是：仅当数据用于在数据库中定义结构时可以不顾及大小写，以及当以不区分大小写的方式比较时两个名称相同。

必须尽量不丢失区分大小写的数据。因此，当 x.y 和 X.Y 数据都可能在 x.y 或 X.Y 单一位置下保存时，决不能将 a.x 和 B.X 数据保存在 A.x、A.X、b.x 或 b.X 下。一般来说，这保存了域名的第一个标签的大小写，但是强制内部节点标签的标准化。

输入数据到域数据库的系统管理员应当务必指出他们以大小写一致的方式提供给域系统的数据，如果他们的系统是区分大小写的。在域系统中的数据分发系统将确保采用一致的表示法。

2-3-4　大小限制

DNS 中的各种对象和参数均有大小限制。列示它们如下。其中一些能够方便地改变，其他的更为基础。

标签	63 个八位位组或低于
名称	255 个八位位组或低于
TTL	有正负号的 32 位数的正值
UDP 消息	512 个八位位组或低于

第 3 章　域名空间和资源记录(RR)定义

3-1　名称空间定义

消息中的域名用标签序列来表示。每个标签被表示为一个八位位组长度字段，再加上那个八位位组的数目。因为每个域名用根的空标签结束，域名由零长度字节终结。每个长度八位位组的高阶 2 位必定是零，此长度字段的其余 6 位限制标签为 63 个八位位组或更少。

为了简化实现，域名的总长度(即，标签八位位组和标签长度八位位组)被限制在 255 个八位位组或更少。

虽然标签可以包括任何采用八位位组(这些八位位组构成标签)的 8 位值，强烈建议标签遵循在本备忘录其他部分介绍的首选句法，它与现有的主机命名约定兼容。名称服务器和解析器必须采用不区分大小写的方式比较标签(即，A=a)，假设采用零奇偶校验的 ASCII。非

字母代码必须严格匹配。

3-2　资源记录定义

3-2-1　格式

所有 RRs 有相同的顶层格式，如图 8 所示。

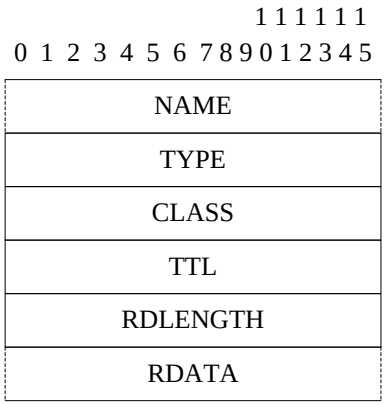

图 8　RRs 顶层格式

图 8 中：

NAME	所有者名称，即，这个资源记录匹配的节点的名称。
TYPE	包含 RR TYPE 代码之一的 2 个八位位组。
CLASS	包含 RR CLASS 代码之一的 2 个八位位组。
TTL	32 位有正负号整数，它规定应当再次咨询信息源之前此资源记录可以被缓存的时间间隔。零值被解释为该 RR 仅能用于正在进行的业务，不应当被缓存。例如，总是将零 TTL 分配给 SOA 记录，以便禁止缓存。零值也可以用于极短暂的数据。
RDLENGTH	无正负号 16 位整数，它规定以八位位组计的 RDATA 字段的长度。
RDATA	可变长度八位位组串，它描述资源。这个信息的格式按照资源记录的 TYPE 和 CLASS 改变。

3-2-2　TYPE 值

TYPE 字段用于资源记录。注意，这些类型是 QTYPEs 的子集。

TYPE	值和含意
A	1，主机地址
NS	2，权威名称服务器
MD	3，邮件目的地(被废弃，使用 MX)
MF	4，邮件转发器(被废弃，使用 MX)
CNAME	5，别名的正则名称
SOA	6，标记权威区域的开始
MB	7，邮箱域名(试验)
MG	8，邮件组成员(试验)
MR	9，邮件重新命名域名(试验)
NULL	10，空 RR(试验)
WKS	11，众所周知的业务描述
PTR	12，域名指针

HINFO 13，主机信息
MINFO 14，邮箱或邮件列表信息
MX 15，邮件交换
TXT 16，文本字符串

3-2-3 QTYPE 值

QTYPE 字段出现在查询的问题部分。QTYPE 是 TYPEs 的超集，因此所有 TYPEs 是合法的 QTYPEs。此外，定义有下述 QTYPEs：

AXFR 252，请求整个区域传送
MAILB 253，请求相关邮箱记录(MB、MG 或 MR)
MAILA 254，请求邮件代理 RRs(被废弃，参阅 MX)
* 255，请求所有记录

3-2-4 CLASS 值

CLASS 字段出现在资源记录中。定义有下述 CLASS 助记符和值：

IN 1，互联网
CS 2，CSNET 类(被废弃，仅在某些被废弃的 RFCs 中用于举例)
CH 3，CHAOS 类
HS 4，赫西奥德(Hesiod)[Dyer 87]

3-2-5 QCLASS 值

QCLASS 字段出现在查询的问题部分。QCLASS 值是 CLASS 值的超集；每一个 CLASS 都是合法的 QCLASS。除了 CLASS 值以外，定义有下述 QCLASSes：

* 5，任何类

3-3 标准 RRs

在所有类中，下述 RR 定义预期会出现，至少有可能。尤其是，NS、SOA、CNAME 和 PTR 将在所有类中使用，并且在所有类中有相同格式。因为它们的 RDATA 格式是已知的，在这些 RRs 的 RDATA 部分中的所有域名可以压缩。

<domain-name>是表示为一系列标签的域名，并被零长度标签终止。<character-string>是单一长度八位位组，再加上那些字符的数目。<character-string>被看作是二进制信息，其长度最多可达 256 个字符(包括长度八位位组)。

3-3-1 CNAME RDATA 格式

CNAME RDATA 格式如图 9 所示。

图 9 CNAME RDATA 格式

图 9 中：

CNAME 一个<domain-name>，它规定所有者的正则或主要名称。该所有者名称是别名。

CNAME RRs 不引起附加部分处理，但是在某些情况下，名称服务器可以选择以正则名称重新开始查询。更详细内容请参阅[RFC-1034]中有关名称服务器逻辑的介绍。

3-3-2 HINFO RDATA 格式

HINFO RDATA 格式如图 10 所示。

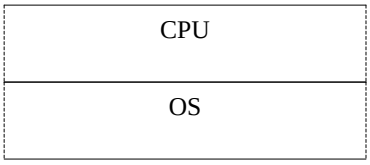

图 10 HINFO RDATA 格式

图 10 中：

CPU 一个<character-string>，它规定 CPU 的类型。

OS 一个<character-string>，它规定操作系统的类型。

CPU 和 OS 的标准值可在[RFC-1010]中找到。

HINFO 记录用于捕获关于主机的一般信息。主要用于诸如 FTP 等协议，当同样类型的计算机之间或操作系统之间交谈时，FTP 协议可以使用特殊的程序。

3-3-3 MB RDATA 格式(试验)

MB RDATA 格式(试验)如图 11 所示。

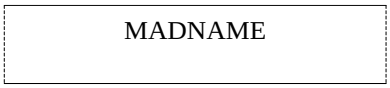

图 11 MB RDATA 格式

图 11 中：

MADNAME 一个<domain-name>，它规定有指定邮箱的主机。

MB 记录引起附加部分处理，该处理查找对应 MADNAME 的 A 类 RRs。

3-3-4 MD RDATA 格式(被废弃)

MD RDATA 格式(被废弃)如图 12 所示。

图 12 MD RDATA 格式(被废弃)

图 12 中：

MADNAME 一个<domain-name>，它规定有该域的邮件代理的主机，该邮件代理应当能够交付该域的邮件。

MD 记录引起附加部分处理，该处理查找对应 MADNAME 的 A 类记录。

MD 已经被废弃。新机制的细节请参阅 MX 的定义和[RFC-974]。处理主文件中发现的 MD RRs 的推荐策略是拒绝它们，或者将它们转换为具有优先权为 0 的 MX RRs。

3-3-5 MF RDATA 格式(被废弃)

MF RDATA 格式如图 13 所示。

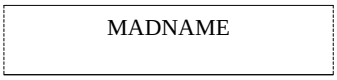

图 13　MF RDATA 格式(被废弃)

图 13 中：

MADNAME　　一个<domain-name>，它规定有该域的邮件代理的主机，该邮件代理将接收转发到该域的邮件。

MF 记录引起附加部分处理，该处理查找对应 MADNAME 的 A 类记录。

MF 被废弃。新机制的细节请参阅 MX 的定义和[RFC-974]。处理主文件中发现的 MD RRs 的推荐策略是拒绝它们，或者将它们转换为具有优先权为 10 的 MX RRs。

3-3-6　MG RDATA 格式(试验)

MG RDATA 格式如图 14 所示。

图 14　MG RDATA 格式

图 14 中：

MGMNAME　　一个<domain-name>，它规定邮箱，该邮箱是由域名规定的邮件组的成员。

MG 记录不引起附加部分处理。

3-3-7　MINFO RDATA 格式(试验)

MINFO RDATA 格式如图 15 所示。

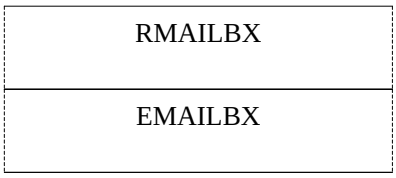

图 15　MINFO RDATA 格式

图 15 中：

RMAILBX　　一个<domain-name>，它规定负责邮件列表或邮箱的邮箱。如果这个域名命名根，MINFO RR 的所有者负责它自己。注意，许多现有的邮件列表对于邮件列表 X 的 RMAILBX 字段使用邮箱 X-请求，例如，Msgroup 的 Msgroup -请求。这个字段提供更多的通用机制。

EMAILBX　　一个<domain-name>，它规定邮箱，该邮箱将接收与邮件列表或由 MINFO RR 的所有者指定的邮箱有关的出错消息(类似 ERRORS-TO：已经被建议的字段)。如果这个域名命名根，应当将错误返回给消息发送者。

MINFO 记录不引起附加部分处理。虽然这些记录能够与简单的邮箱联系起来，通常用

邮件列表使用它们。

3-3-8　MR RDATA 格式(试验)

MR RDATA 格式如图 16 所示。

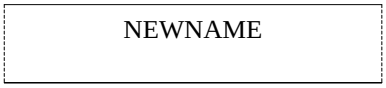

图 16　MR RDATA 格式

图 16 中：

NEWNAME　　一个<domain-name>，它规定邮箱，该邮箱是指定邮箱的适当重新命名。

MR 记录不引起附加部分处理。MR 的主要应用是作为用户(该用户已经移动到不同的邮箱)的转发条目。

3-3-9　MX RDATA 格式

MX RDATA 格式如图 17 所示。

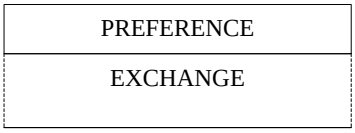

图 17　MX RDATA 格式

图 17 中：

PREFERENCE　　16 位整数，它规定在同一所有者内，众多 RR 间，给与这个 RR 的优先权。值越低优先权越高。

EXCHANGE　　一个<domain-name>，它指定愿意为所有者名称充当邮件交换的主机。

MX 记录引起针对由 EXCHANGE 指定的主机的、A 类型附加部分处理。MX RRs 的使用在[RFC-974]中详细介绍。

3-3-10　NULL RDATA 格式(试验)

NULL RDATA 格式如图 18 所示。

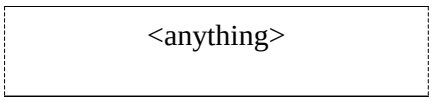

图 18　NULL RDATA 格式

在 RDATA 字段中可以放任何东西，只要它小于等于 65535 个八位位组。

NULL 记录不引起附加部分处理。在主文件中不允许存在 NULL RRs。在某些 DNS 的试验性扩展中，NULLs 用作占位符。

3-3-11　NS RDATA 格式

NS RDATA 格式如图 19 所示。

图 19　NS RDATA 格式

图 19 中：

NSDNAME　　一个<domain-name>，它规定主机，对于指定的类和域，该主机应当是权威的。

NS 记录不仅引起定位 A 类型记录的通常的附加部分处理，而且当在转介中使用时，引起特定区域(在该区域内，NS 记录作为胶信息驻留。)搜索。

NS RR 表明应当期盼命名的主机有以指定类的所有者名称开始的区域。注意，该类可以不指出应当用于与该主机通信的协议族，尽管该类一般是强烈暗示。例如，作为或者是互联网(IN)或者是 Hesiod(HS)类信息的名称服务器的主机，当它被查询时，一般使用 IN 类协议。

3-3-12　PTR RDATA 格式

PTR RDATA 格式如图 20 所示。

图 20　PTR RDATA 格式

图 20 中：

PTRDNAME　　一个<domain-name>，它在域名空间中指向某个位置。

PTR 记录不引起附加部分处理。这些 RRs 在特定的域中使用，以便在域空间中指向某个其他位置。这些记录是简单的数据，它们不暗示任何类似由 CNAME(它标识别名)执行的处理的特殊处理。具体举例，请参阅 IN-ADDR.ARPA 域介绍。

3-3-13　SOA RDATA 格式

SOA RDATA 格式如图 21 所示。

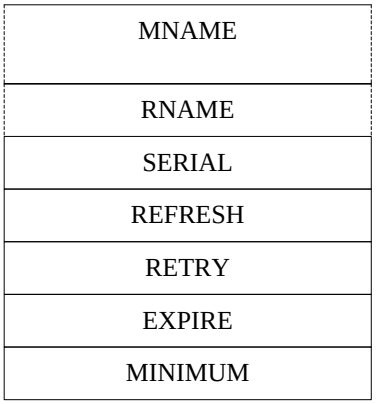

图 21　SOA RDATA 格式

图 21 中：

MNAME　　名称服务器的<domain-name>，该名称服务器是这个区域的

数据的起源或主要源。

RNAME	一个<domain-name>，它规定负责这个区域的个人的邮箱。
SERIAL	该区域的原始副本的无正负号 32 位版本号。区域传递保存这个值。这个值叠起(wrap)，并且应当使用系列空间算法比较这个值。
REFRESH	区域应当被刷新前的 32 位时间间隔。
RETRY	应当重试失败的刷新前，应当消逝的 32 位时间间隔。
EXPIRE	32 位时间值，它规定时间间隔(即，在区域不再是权威的之前可以消逝的时间间隔)的上限。
MINIMUM	无正负号 32 位最小值 TTL 字段，应当用来自这个区域的任何 RR 输出它。

SOA 记录不引起附加部分处理。

所有时间以秒为单位。

这些字段的大多数仅与名称服务器维护操作有关。然而，在所有查询操作(这些操作从区域中检索 RRs)中要使用 MINIMUM。每当在响应中发送 RR 到查询时，根据该 RR 和在适当 SOA 中 MINIMUM 字段，设置此 TTL 字段为 TTL 字段的最大值。于是，MINIMUM 是区域内所有 RRs 的 TTL 字段的下限。注意，当 RRs 被复制进响应时应当这样使用 MINIMUM，当从主文件或者通过区域传递加载区域时不应当这样使用 MINIMUM。这样规定的原因是使将来的动态更新功能可以用已知的语义改变 SOA RR。

3-3-14 TXT RDATA 格式

TXT RDATA 格式如图 22 所示。

图 22 TXT RDATA 格式

图 22 中：

TXT-DATA 一个或多个<character-string>s。

TXT RRs 用于保持描述性文本。文本的语义取决于文本被发现的域。

3-4 互联网特定 RRs

3-4-1 A RDATA 格式

RDATA 格式如图 23 所示。

图 23 RDATA 格式

图 23 中：

ADDRESS 32 位互联网地址。

有多个互联网地址的主机将有多个 A 记录。

A 记录不引起附加部分处理。主文件中的 A 线的 RDATA 部分表示为 4 个十进制数，这 4 个十进制数由圆点分开，没有任何嵌入的空格，是互联网地址(例如，“10.2.0.52”或“192.0.5.6”)。

3-4-2　WKS RDATA 格式

WKS RDATA 格式如图 24 所示。

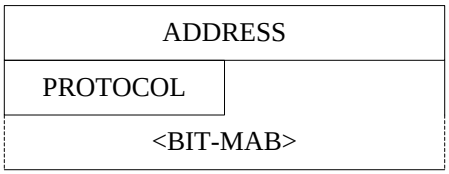

图 24　WKS RDATA 格式

图 24 中：

ADDRESS	32 位互联网地址
PROTOCOL	8 位 IP 协议号
<BIT MAP>	可变长度位图。该位图必须是 8 位长的倍数。

WKS 记录用于描述众所周知的业务，这些业务由特定互联网络地址上的特定协议支持。PROTOCOL 字段规定 IP 协议号，位图在每个规定协议的端口上有一位。第一位对应端口 0，第二位对应端口 1，等等。如果位图不包括感兴趣协议的位，假设那个位是零。端口和协议的适当值和助记符在[RFC-1010]中规定。

例如，如果 PROTOCOL=TCP(6)，第 26 位对应 TCP 端口 25(SMTP)。如果这个位被置 1，SMTP 服务器应当侦听 TCP 端口 25；如果置 0，在该指定地址上不支持 SMTP 业务。

WKS RRs 的作用是为 TCP 和 UDP 服务器提供可用信息。如果服务器既支持 TCP 又支持 UDP，或者有多个互联网地址，那么，使用多个 WKS RRs。

WKS RRs 不引起附加部分处理。

在主文件中，使用助记符或十进制数表示端口和协议。

3-5　IN-ADDR.ARPA 域

互联网使用专门的域支持网关定位和互联网地址到主机的映射。在其他域中，其他的类可以使用相似策略。这个域的目的是提供执行主机地址到主机名称映射的可靠方法，以及为找出互联网中特定网络上所有网关的查询提供方便。

注意，这两项业务都类似能由反向查询执行的功能；区别是域名空间的这个部分被按照地址构建，并因此能够保证不需要域空间穷举搜索就可以找到适当数据。

该域以 IN-ADDR.ARPA 开始，并且有遵循互联网寻址结构的子结构。

除了 IN-ADDR.ARPA 后缀以外，定义 IN-ADDR.ARPA 域中的域名有最多 4 个标签。每个标签代表互联网地址的一个八位位组，并且表示为范围在 0-255 内的十进制值字符串(除由单个 0 代表的零八位位组情况，省略起始的零)。

用域名代表主机地址，该域名有所有 4 个规定的标签。于是，互联网地址 10.2.0.52 的数据位于域名 52.0.2.10.IN-ADDR.ARPA 中。反过来，虽然难以理解，允许授权给这样的区域(这些区域恰好是地址空间的一个网络)。例如，10.IN-ADDR.ARPA 可以是包括 ARPANET 的数据的区域，而 26.IN-ADDR.ARPA 可以是 MILNET 的独立区域。地址节点用于保持到一般域空间中主要主机名称的指针。

在 IN-ADDR.ARPA 域中，网络编号对应在不同深度中的某些非终端节点，因为互联网编号可以是 1、2 或 3 个八位组。网络节点用于保持到附着的那个网络的网关们的主要主机名称的指针。因为由定义可知，网关位于不止一个网络上，网关一般有两个或多个指向它的网络节点。网关也有主机层次的指向它们的完全合格地址的指针。

不仅在网络节点中的网关指针，而且在完整地址节点中的一般主机指针都使用 PTR RR

反向指出相应主机的主要域名。

例如，IN-ADDR.ARPA 域包括网络 10 和网络 26 之间有关 ISI 网关的信息，从网络 10 到 MIT 的网络 18 有关 MIT 网关的信息，以及主机 A.ISI.EDU 和 MULTICS.MIT.EDU 的有关信息。假设 ISI 网关有地址 10.2.0.22 和 26.0.0.103，以及名称 MILNETGW.ISI.EDU，并且 MIT 网关有地址 10.0.0.77 和 18.10.0.4，以及名称 GW.LCS.MIT.EDU，域数据库将包括：

```
10.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
10.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
18.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
26.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
22.0.2.10.IN-ADDR.ARPA.       PTR MILNET-GW.ISI.EDU.
103.0.0.26.IN-ADDR.ARPA.      PTR MILNET-GW.ISI.EDU.
77.0.0.10.IN-ADDR.ARPA.       PTR GW.LCS.MIT.EDU.
4.0.10.18.IN-ADDR.ARPA.       PTR GW.LCS.MIT.EDU.
103.0.3.26.IN-ADDR.ARPA.      PTR A.ISI.EDU.
6.0.0.10.IN-ADDR.ARPA.        PTR MULTICS.MIT.EDU.
```

于是，希望找到在网络 10 上的网关的程序能够发起形如 QTYPE=PTR, QCLASS=IN, QNAME=10.IN-ADDR.ARPA 的查询。它将在响应中收到两个 RRs：

```
10.IN-ADDR.ARPA.              PTR MILNET-GW.ISI.EDU.
10.IN-ADDR.ARPA.              PTR GW.LCS.MIT.EDU.
```

接着，该程序能够为 MILNETGW.ISI.EDU.和 GW.LCS.MIT.EDU.发起 QTYPE=A, QCLASS=IN 查询，以便发现这些网关的互联网地址。

希望发现对应互联网主机地址 10.0.0.6 的主机名称的解析器将追踪形如 QTYPE=PTR, QCLASS=IN, QNAME=6.0.0.10.IN-ADDR.ARPA 的查询，并将收到：

```
6.0.0.10.IN-ADDR.ARPA.        PTR MULTICS.MIT.EDU.
```

对这些业务应用有几点告诫提请注意：

- 因为 IN-ADDR.ARPA 指定域和特定主机或网关的一般域位于不同区域，数据有可能不一致。
- 在分开的域中网关常常有两个名称，这两个名称中只有一个是主要的。
- 使用域数据库的系统，为了初始化它们的路由表，必须备足网关信息才能开始，以便保证这些系统能够访问适当的名称服务器。
- 网关数据仅以等同于目前 HOSTS.TXT 文件的方式反映网关的存在。它不能代替从 GGP 或 EGP 动态获得的信息。

3-6 定义新的类型、类和专用名称空间

前面定义的类型和类是到本备忘录发布时在使用的类型和类。应当预期会有新定义出现。这一节给出一些建议，供拟补充现有功能的设计者参考。邮件列表 NAMEDROPPERS@SRI-NIC.ARPA 是讨论一般设计问题的论坛。

一般讲，当添加新信息到有关现有对象的数据库，或我们需要某个全新对象的新数据格式时，新类型是适当的。设计者应当尝试定义通常适用所有类的类型和它们的 RDATA 格式，这将避免信息重复。当 DNS 用于新协议(等)，新协议需要新的专用类数据格式时，新的类是适当的；或者当希望得到现有名称空间的副本，但必须是独立的管理域时，新的类是适当的。

新类型和类需要主文件助记符；主文件的格式要求类型助记符和类助记符不相交。

TYPE 值和 CLASS 值必须分别是 QTYPEs 和 QCLASSes 的适当子集。

目前的系统使用多个 RRs 代表多个类型值，而不是在单一 RR 的 RDATA 部分保存多个值。对于大多数应用这样做效率不高，但这使得 RRs 较短。在某个关于动态更新方法的试验中，包括了多 RRs 假设。

目前的系统尝试尽量减少数据库中的数据复制，以便确保一致性。于是，为了发现邮件交换的主机的地址，您映射邮件域名到主机名称，接着从主机名称到地址，而不是直接映射到主机地址。这种方法是首选方法，其优点是它避免了产生不一致的可能。

在定义新的数据类型时，不应当用多个 RR 类型生成条目间的顺序，或用其表示等价绑定的不同格式，反之，这个信息应当在 RR 实体中使用单一类型携带。这个策略可避免与缓存多个类型和定义 QTYPEs 匹配多个类型有关的问题。

例如，邮件交换绑定的原始形式使用两个 RR 类型，一个代表“详尽的(closer)”交换(MD)，一个代表“较少详尽的”交换(MF)。困难在于缓存中一个 RR 类型的存在不传递有关其他 RR 类型的任何信息，因为获得该缓存信息的查询或许已经使用 MF、MD 或 MAILA(该 MAILA 匹配二者)的 QTYPE。重新设计的业务使用单一类型(MX)，该单一类型使用 RDATA 部分中的“优先权”值，该“优先权”值能够排序不同的 RRs。然而，如果在缓存中发现任何 MX RRs，那么，都应当在那里。

第 4 章　消息

4-1　格式

域协议内部的所有通信采用称作消息的单一格式携带。消息的顶层格式分成 5 个部分(在某些情况，它们中一些是空的)，如图 25 所示。

Header	
Question	向名称服务器提出的问题
Answer	回答问题的RRs
Authority	指向权威的RRs
Additional	保持附加信息的RRs

图 25　消息格式

首部部分总是存在。首部包括一些字段，这些字段规定其余部分中哪些部分存在，也规定消息是查询还是响应，是标准查询还是某个其他的操作码，等等。

首部后面部分的名称引申自在标准查询中它们的应用。问题部分包括一些字段，这些字段描述发送给名称服务器的问题。这些字段是查询类型(QTYPE)、查询等级(QCLASS)和查询域名(QNAME)。最后 3 个部分有相同格式：级联的资源记录(RRs)的可能空列表。回答部分包括回答问题的 RRs；权威部分包括指向权威名称服务器的 RRs；附加记录部分包括与查询有关的 RRs，但是这些 RRs 严格说不是问题的回答。

4-1-1　首部部分格式

首部包括如图 26 所示的字段。

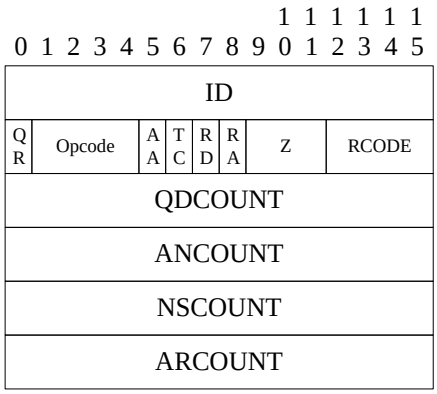

图 26　首部包括的字段

图 26 中：

ID　由程序分配的 16 位标识符，该程序产生任何种类的查询。这个标识符是被复制的相应响应，这个标识符可由请求者用于匹配未得到解决的查询的响应。

QR　1 位字段，该字段规定这个消息是查询(0)还是响应(1)。

OPCODE　4 位字段，该字段规定这个消息中查询的种类。这个值由查询的发起者设置，它被复制进响应中。这个值的具体取值是

- 0　标准查询(QUERY)
- 1　反向查询(IQUERY)
- 2　服务器状态请求(STATUS)
- 3-15　保留将来使用

AA　权威回答(Authoritative Answer) ---这个位在响应中有效，这个位规定进行回应的名称服务器是问题部分中域名的权威(名称服务器)。注意，由于别名，回答部分的内容可以有多个所有者名称。AA 位对应匹配查询名称的名称，或者对应回答部分中第一个所有者名称。

TC　截断(Truncation) ---表示这条消息由于长度大于传送通道上准许的长度而被截断。

RD　期望递归(Recursion Desired) ---在查询中这个位可以置 1，并且被复制进响应中。如果 RD 置 1，它引导名称服务器递归跟踪查询。支持递归查询是可选项。

RA　递归可用(Recursion Available) ---在响应中这个字段被置 1 或被清零，指示在名称服务器中是否支持递归查询。

Z　保留将来使用。在所有查询和响应中此位必须置 0。

RCODE　响应代码(Response code)---这个 4 位字段是响应的一部分。其取值有如下含意：

- 0　没有出错条件
- 1　格式错误---名称服务器不能解释查询。
- 2　服务器故障---由于与名称服务器有关的问题，名称服务器不能处理这个查询。
- 3　名称错误---仅对来自权威名称服务器的响应有意义，这个代码预示在该查询中被查询的域名不存

在。

4　　未实现---名称服务器不支持请求的查询种类。

5　　拒绝---由于策略原因名称服务器拒绝执行指定的操作。例如，名称服务器可能不希望提供信息给特定的请求者，或者名称服务器可能不希望为特定的数据执行特定的操作(例如，区域传递)。

6-15　　保留将来使用。

QDCOUNT　　无正负号 16 位整数，它规定问题部分中条目的数量。

ANCOUNT　　无正负号 16 位整数，它规定回答部分中资源记录的数量。

NSCOUNT　　无正负号 16 位整数，它规定权威记录部分中名称服务器资源记录的数量。

ARCOUNT　　无正负号 16 位整数，它规定附加记录部分中资源记录的数量。

4-1-2　问题部分格式

问题部分用于在大多数查询中携带“问题”，即，这些参数定义正在问什么。此部分包括 QDCOUNT(通常取 1)条目，具有如图 27 所示的格式。

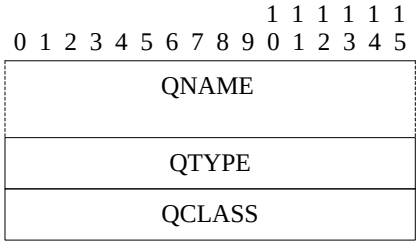

图 27　问题部分格式

图 27 中：

QNAME　　用标签序列表示的域名，在标签序列中每个标签包括长度八位位组，再加上那个八位位组的数量。对于根的空标签，此域名用零长度八位位组终止。注意，这个字段可以是奇数个八位位组；不使用填充。

QTYPE　　2 个八位位组代码，它规定查询的类型。这个字段的值包括所有适用于 TYPE 字段的代码，以及某些更一般的代码(这些代码可以匹配不止一个 RR 类型)。

QCLASS　　2 个八位位组代码，它规定查询的类。例如，对于互联网，QCLASS 字段是 IN。

4-1-3　资源记录格式

回答部分、权威部分和附加部分都共享相同的格式：可变数目资源记录，其中记录的数目在首部内相应计数字段中规定。每条资源记录的格式如图 28 所示。

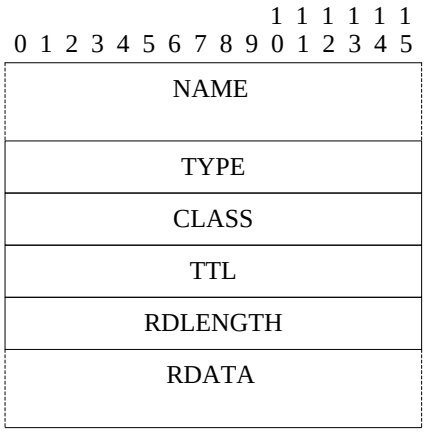

图 28　资源记录格式

图 28 中：

NAME	这个资源记录匹配的域名。
TYPE	包括 RR 类型代码之一的两个八位位组。这个字段规定 RDATA 字段中数据的含意。
CLASS	两个八位位组，它们规定 RDATA 字段中数据的类。
TTL	32 位无正负号整数，它规定时间间隔(单位为秒)，即资源记录应当被抛弃前，它可以被缓存的时间。零值的含意是 RR 仅能用于正在进行的业务，不能被缓存。
RDLENGTH	无正负号 16 位整数，它规定 RDATA 字段的长度，以八位位组为单位。
RDATA	可变长度八位位组串，它描述资源。这条信息的格式根据资源记录的 TYPE 和 CALSS 改变。例如，如果 TYPE 是 A 和 CALSS 是 IN，此 RDATA 字段是 4 个八位位组 ARPA 互联网地址。

4-1-4　消息压缩

为了减小消息大小，域系统使用去除消息中域名重复的压缩方案。在这个方案中，整个域名或在域名底部的标签列表被用一个指针取代，该指针指向同一名称的前一个具体值。

该指针取图 29 所示两个八位位组序列格式。

图 29　指针的两个八位位组序列格式

前两位是 1。这使得指针能够区分标签，因为由于标签被限制在小于等于 63 个八位位组，标签必须用两个 0 位开始。(10 和 01 这两个组合保留将来使用。) OFFSET 字段规定从消息开始的偏移(即，域首部中 ID 字段的第一个八位位组)。零偏移表示 ID 字段的第一个字节，等等。

此压缩方案使得消息中的域名能够表示为下述之一：

- 以零八位位组结束的标签序列
- 指针
- 用指针结束的标签序列

指针仅能用于这样的域名具体值，该域名中格式不是特定类。如果不是这种情况，将要

求名称服务器或解析器知道它所处理的所有 RRs 的格式。到目前为止没有这类情况，但是在将来的 RDATA 格式中可能出现。

如果消息(该消息受长度字段支配)的一部分包括域名(诸如 RR 的 RDATA 部分)，并且使用压缩，在长度计算中使用被压缩的名称的长度(而不是扩展名称的长度)。

在由程序生成的消息中是否避免使用指针由程序自主考虑，尽管这将降低数据报的能力，并且可能引起截断。然而，要求所有程序理解包括指针的到达消息。

例如，数据报或许需要使用域名 F.ISI.ARPA、FOO.F.ISI.ARPA、ARPA 和根。忽略消息的其他字段，这些域名或许如图 30 所示。

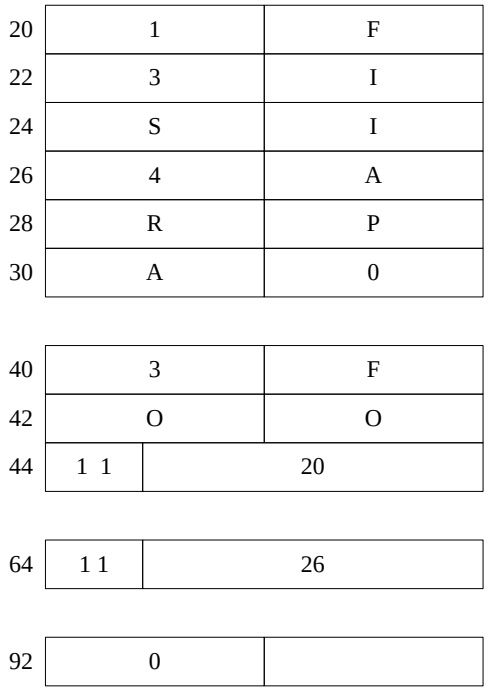

图 30　域名格式

F.ISI.ARPA 的域名以偏移 20 显示。FOO.F.ISI.ARPA 的域名以偏移 40 显示；这个定义使用了将 FOO 的标签级联到先前定义的 F.ISI.ARPA 的指针。使用以偏移 20 指向名称 F.ISI.ARPA 的 ARPA 分量的指针，以偏移 64 定义域名 ARPA；注意，这个指针依赖于 ARPA，该 ARPA 是在偏移 20 的字符串中的最后一个标签。根域名称由在偏移 92 的单一零八位位组定义；该根域名没有标签。

4-2　传送

DNS 假设消息将作为数据报传送，或采用字节流由虚电路传送。尽管虚电路能够用于任何 DNS 活动，数据报由于它们较低的门槛和较好的性能是查询的首选。区域刷新活动必须使用虚电路，因为需要可靠传输。

互联网支持在服务器端口 53(十进制)上使用 TCP[RFC-793]的名称服务器访问，以及支持在 UDP 端口 53(十进制)上使用 UDP[RFC-768]的数据报访问。

4-2-1　UDP 应用

使用 UDP 用户服务器端口 53(十进制)发送的消息。

由 UDP 携带的消息被限制在 512 字节(不包括 IP 首部或 UDP 首部)。较长的消息被截断，TC 位在首部中被置 1。

区域传递不适合使用 UDP，但是推荐的互联网中标准查询方法是 UDP。使用 UDP 发送的查询可能丢失，因此需要考虑重传策略。查询或查询的响应可能由网络重新排序，或者由名称服务器中的处理重新排序，所以解析器不能依赖按顺序返回的响应。

最优 UDP 重传策略将随互联网性能和客户端需求改变，然而给出下述建议：

- 在向服务器的特定地址重复查询前，客户端应当尝试其他服务器和服务器其他地址。
- 如果可能，重传间隔应当基于前面的统计量。过于频繁的重传一般会导致社区响应速度减慢。根据客户端连接到它期盼的服务器的畅通情况，最小重传间隔应当为 2 至 5 秒。

关于服务器选择和重传策略的更多建议请参阅本备忘录的解析器部分。

4-2-2 TCP 应用

在 TCP 连接上发送消息使用服务器端口 53(十进制)。该消息用两字节长度字段(该字段给出消息长度)做前缀，该消息不包括该两字节长度字段。这个长度字段使得在开始解析消息前低层处理能够组装好完整的消息。

推荐几种连接管理策略：

- 服务器不应当阻止等待 TCP 数据的其他活动。
- 服务器应当支持多连接。
- 服务器应当假设客户端将发起连接关闭，应当推迟关闭它的连接，直到所有未解决的客户端请求都被满足。
- 如果服务器需要关闭域连接以便回收资源，服务器应当等待，直到连接变为空闲大约两分钟后。尤其是，服务器应当允许在一次连接上实现 SOA 和 AXFR 请求序列(该序列启动刷新操作)。因为服务器有可能无论用什么方法也不能够回答查询，代替得体的关闭，服务器可能使用单方面关闭或重新设置。

第 5 章 主文件

主文件是文本文件，它包括文本形式的 RRs。因为区域的内容可以用 RRs 列表的形式表示，主文件多用于定义区域，尽管它可用于列出缓存器内容。因此，本节首先讨论主文件中 RRs 的格式，接着讨论当主文件用于在某个名称服务器中创立区域时的特殊考虑。

5-1 格式

这些文件的格式是一系列条目。条目主要是面向行的，尽管圆括号可用于跨行边界继续项目(item)列表，文本字面值可以在文本内部包括 CRLF。制表符和空格的任意组合在构成条目的独立项目之间充当定界符。在主文件中，任何行均可以由注释结束。注释由“;”(分号)开始。

定义下述条目：

<blank>[<comment>]

$ORIGIN <domain-name> [<comment>]

$INCLUDE <file-name> [<domain-name>] [<comment>]

<domain-name><rr> [<comment>]

<blank><rr> [<comment>]

在文件任何位置的空白行，有注释或没有注释，都是允许的。

定义两个控制条目：$ORIGIN 和$INCLUDE。跟在$ORIGIN 之后的是域名，$ORIGIN 重新设置相对域名目前的起点到规定的名称。$INCLUDE 将命名的文件插入到目前的文件，

并能够有选择地指定域名，该域名为包括的文件设置相对域名起点。$INCLUDE 也可以有注释。注意，无论在包括的文件中对相对起点做了什么改变，$INCLUDE 条目绝不改变父文件的相对起点。

最后两种形式表示 RRs。如果 RR 条目以空格开始，那么假设该 RR 由上次规定的所有者拥有。如果 RR 条目以<domain-name>开始，那么所有者名称被重新设置。

<rr>内容取下述形式之一：

[<TTL>] [<class>] <type> <RDATA>

[<class>] [<TTL>] <type> <RDATA>

RR 以可选的 TTL 字段和类字段开始，跟着是对应类型和类的类型和 RDATA 字段。类和类型使用标准助记符，TTL 是十进制整数。省略的类值和 TTL 值默认是上次显示指定的值。因为类型和类助记符不相交，解析是唯一的。(注意，这个顺序不同于在举例中以及在实际 RRs 中使用的顺序；此假设的顺序更方便解析和预先设置。)

在主文件中<domain-name>s 占数据的一大部分。域名中的标签被表示为字符串并用圆点分开。引用惯例使得任意字符能够在域名中保存。称以圆点结束的域名为绝对域名，并看作是完整的。称不以圆点结束的域名为相对域名；实际域名是带有起点的相对部分的级联，该起点在$ORIGIN、$INCLUD 中规定，实际域名或者作为对加载例行程序的主文件的内容简介。当没有起点可用时，出现相对名称表示出错。

用一种或两种方法表示<character-string>：作为没有内部空格的连续字符集合，或者作为以“and ending with a”开始的串。在“delimited string any character can occur, except for a”自身内，必须使用\(反斜杠)引用。

因为这些文件是文本文件，为了能够加载任意数据，需要几个特殊编码，尤其是：

	根的。
@	独立的@用于表示当前的起点。
\X	这里 X 是除了数字(0-9)以外的任何字符，用于引用那个字符，以致它的特殊含意不能应用。例如，“\.”可用于在标签中放置圆点字符。
\DDD	这里每个 D 都是数字，\DDD 是对应由 DDD 描述的十进制数的八位位组。假设此最终的八位位组是文本，并且此最终的八位位组没有被用于检查特殊含意。
( )	圆括号用于聚拢跨行边界的数据。实际上，在圆括号内不能识别行结束。
；	分号用于开始注释；该行的其余部分被忽略。

5-2　定义区域的主文件的应用

当主文件加载区域时，如果主文件中出现任何错误，此操作应当被制止。这样做的原因是单个错误可能产生广泛影响。例如，假如定义授权的 RRs 有语法错误；那么服务器将对该子区域内的所有名称返回权威名称错误(除了在这样的情况，即，那里，该子区域也在该服务器上)。

除了确保该文件在语法上是正确的以外，应当执行另外几个合法性检查：

1、文件中所有 RRs 应当有相同的类。

2、在该区域顶部应当恰好存在一个 SOA RR。

3、如果授权存在，并且要求胶信息，胶信息应当存在。

4、在区域中，存在于权威节点之外的信息应当是胶信息，而不是起点或类似错误的结果。

5-3　主文件举例

下面是文件举例，该文件或许用于定义 ISI.EDU 区域，并用 ISI.EDU 的起点加载：

```
@       IN      SOA     VENERA      Action\.domains (
                                                20 ; SERIAL
                                                7200 ; REFRESH
                                                600 ; RETRY
                                                3600000; EXPIRE
                                                60) ; MINIMUM
                NS      A.ISI.EDU.
                NS      VENERA
                NS      VAXA
                MX      10      VENERA
                MX      20      VAXA
A               A       26.3.0.103
VENERA          A       10.1.0.52
                A       128.9.0.32
VAXA            A       10.2.0.27
                A       128.9.0.33
$INCLUDE <SUBSYS>ISI-MAILBOXES.TXT
```

这里，文件<SUBSYS>ISI-MAILBOXES.TXT为：

```
MOE             MB          A.ISI.EDU.
LARRY           MB          A.ISI.EDU.
CURLEY          MB          A.ISI.EDU.
STOOGES         MG          MOE
                MG          LARRY
                MG          CURLEY
```

注意在指定负责人信箱“Action.domains@E.ISI.EDU”的 SOA RR 中\字符的使用。

第 6 章　名称服务器实现

6-1　架构

名称服务器的最佳架构将取决于主机操作系统，以及是否名称服务器与解析器操作集成在一起(即，或者是通过支持递归业务，或者是通过与解析器共享名称服务器的数据库)。本节讨论名称服务器的实现考虑，该名称服务器与解析器共享数据库，然而在任何名称服务器中这些关注大都具备。

6-1-1　控制

名称服务器必须采用多并发活动，无论多并发活动是在主机 OS 中作为单独任务实现，还是在单个名称服务器程序内作为多路复用实现。绝不应该做的是当名称服务器等待刷新 TCP 数据或等待查询活动时，它阻止 UDP 请求业务。类似，如果不以并行方式处理这些请求名称服务器不应当尝试提供递归业务，尽管它可以选择串行处理来自单个客户端的请求，或者选择把来自同一客户端的相同请求看作是多个副本。当名称服务器从主文件重新载入区域时，或当名称服务器将新刷新的区域加入到它的数据库时，名称服务器不应当显著延时请求。

6-1-2　数据库

虽然名称服务器的实现可以自由使用这些实现选择的任何内部数据结构，建议的结构由三个主要部分组成：

- “目录”数据结构，它列出可由这个服务器使用的区域，列出指向该区域数据结构的“指针”。这个结构的主要目的是为到来的标准查询发现最近的祖宗区域，如果有的话。
- 由名称服务器维持的每个区域的独立数据结构。
- 缓存数据的数据结构。(或者或许是不同的类的独立缓存)

所有这些数据结构能够采用相同的树状结构格式实现，这一树状结构格式在不同部分采用断开束缚节点的不同数据：在目录中数据是指向区域的指针，而在区域和缓存器数据结构中，数据是 RRs。在设计树状框架时，设计者应当认识到查询处理需要使用不区分大小写的标签比较来横贯整个树；认识到在实际数据中，少数节点有非常高的分支系数(100 至 1000 或更高)，但是绝大多数节点只有非常低的分支系数(0 至 1)。

解决大小写问题的一种方法是采用两部分保存每个节点的标签：标签的标准化大小写表示法(其中所有 ASCII 字符采用一种大小写)，同时采用位掩码，位掩码指出哪个字符实际上具有不同的大小写。分支系数差异可使用简单的节点链接表来处理，直到分支系数超过某个门限，超过该门限后，转变到哈希结构。在任何情况，用于保存树的各部分的哈希结构必须确保哈希函数和过程保持兼顾 NDS 惯例。

在以下因素推动下，数据库不同部分使用独立结构：

- 目录结构几乎可以是静态结构，仅当系统管理员改变由服务器支持的区域时它才需要改变。这个结构也可以保存用于控制刷新活动的参数。
- 区域的独自数据结构，使得通过简单改变目录中的指针就可以代换区域。区域刷新操作可以建立新的结构，并且当完成后，经简单的指针替代即可将新的结构粘贴入数据库。特别需要注意的是当区域被刷新后，查询不应当同时使用新的和旧的数据。
- 采用适当的搜索程序，区域中的权威数据将总是“隐藏”缓存的数据，因此优先于缓存的数据。
- 在区域定义中的错误(这些错误造成重叠区域，等)可以引起对查询的错误响应，但是问题确定被简化，一个“坏”区域的内容不会侵蚀另一个区域的内容。
- 因为缓存器更新非常频繁，在系统重新启动期间缓存器对于侵蚀的防范最为薄弱。缓存器也会变得充斥过期的 RR 数据。在这两种情况，过期的 RR 数据都能够简单地被抛弃，不会妨碍区域数据。

数据库设计的主要方面是选择结构，该结构使名称服务器能够应对名称服务器主机崩溃。在系统崩溃时名称服务器应当保存的状态信息包括目录结构(包含每个区域的刷新状态)和区域数据自身。

6-1-3　时间

RRs 的 TTL 数据和刷新活动的计时数据取决于 32 位计时器(以秒为单位)。在数据库内部，刷新计时器和缓存数据的 TTLs 在概念上“递减计数”，而区域中的数据不超过恒定的 TTLs。

推荐的实现策略是采用两种方法保存时间：作为相对增量和作为绝对时间。实现此的一种方法是对于一种类型使用正 32 位数，对于其他类型使用负数。区域中的 RRs 使用相对时间；刷新计时器和缓存数据使用绝对时间。根据某个已知的起点选取绝对数值，当放置到针对查询的响应中时转换成相对值。当转换到相对值后绝对 TTL 为负数时，该数据变成过期

数据，应该忽略。

6-2　标准查询处理

标准查询处理的主要算法在[RFC-1034]中介绍。

当采用 QCLASS=*，或某个其他 QCLASS(它匹配多个类)处理查询时，响应绝不可能是权威的，除非服务器能够保证该响应覆盖所有类。

当编辑响应时，可以将被插入到附加部分的 RRs (除非这些 RRs 是回答部分或权威部分中的副本 RRs)从附加部分忽略。

当响应太长，以至于需要截断时，截断应当在响应的结束处开始，截断应当在数据报中提前制定。因此，如果有任何权威部分的数据，将保证回答部分唯一。

在SOA中的MINIMUM值应当用于设置数据(该数据按照区域分发)的TTL的最低标准。当该数据被复制进响应时，应当确定此最低标准函数。这样一来，将来的动态升级协议能够无二义地改变 SOA MINIMUM 字段。

6-3　区域更新和重新加载处理

尽管服务器尽了最大努力，当存在语法错误等时，它还是不能从主文件中加载区域数据，或者不能在它的满期参数到期前更新区域。在这种情况，假如不能假设名称服务器拥有该区域，名称服务器应当回答查询。

如果主文件正在通过 AXFR 向外发送区域，在此发送期间生成了新的版本，如果可能主文件应当继续发送旧版本。在任何情况，主文件绝不能发送一种版本的一部分，发送另一种版本的一部分。如果不可能完成，主文件应当重新设置进行区域传递的连接。

6-4　反向重新(可选)

反向查询是 DNS 的可选部分。不要求名称服务器支持任何形式的反向查询。如果名称服务器收到它不支持的反向查询，它返回出错响应，该响应首部带有“Not Implemented”错误置 1。尽管反向查询是可选的，所有名称服务器必须至少能够返回出错响应。

6-4-1　反向查询和响应的内容

反向查询颠倒标准查询操作执行的映射；当标准查询映射域名到资源时，反向查询映射资源到域名。例如，标准查询或许绑定域名到主机地址；相应的反向查询绑定主机地址到域名。

在消息的回答部分中反向查询取单 RR 格式，问题部分为空。此查询 RR 的所有者名称和它的 TTL 没有意义。响应在问题部分携带问题，这些问题标记拥有该查询 RR(**名称服务器知道该查询 RR**)的所有名称。因为没有名称服务器知道所有域名空间，决不能假设响应是完整的。因此，反向查询主要用于数据库管理和调试操作。反向查询**不是**映射主机地址到主机名称的可接受方法；取而代之的是：使用 INADDR.ARPA 域。

在可能的情况下，名称服务器应当提供不区分大小写的反向查询比较。因此询问“Venera.isi.edu”的 MX RR 的反向查询应当得到与对于“VENERA.ISI.EDU”的反向查询相同的响应；对 HINFO RR“IBM-PC UNIX”的反向查询应当产生对“IBM-pc unix”的反向查询相同的结果。然而，不能保证做到此，因为名称服务器可以拥有包括字符串的 RRs，但是名称服务器不知道该数据是字符。

当名称服务器处理反向查询时，它返回下述二者之一：

1、在问题部分中，将零、1、或多个指定资源的域名作为 QNAMEs。

2、指出名称服务器不支持指定资源类型反向映射的出错代码。

当对反向查询的响应包括一个或多个 QNAMEs 时，在回答部分(该部分定义反向查询)中修改 RR 的所有者名称和 TTL，以便精确匹配在第一个 QNAME 中发现的 RR。

不能使用与标准查询的响应使用的相同机制缓存反向查询中返回的 RRs。原因之一是名称或许有相同类型的多个 RRs，并且仅一个会出现。例如，对多归属地主机的单个地址的反向查询或许产生仅存在一个地址的印象。

6-4-2　反向查询和响应举例

检索域名(该域名对应互联网地址 10.1.0.52)的反向查询的整体结构如图 31 所示。

首部	OPCODE=IQUERY, ID=997
问题	<空>
回答	<anyname>A IN 10.1.0.52
权威	<空>
附加	<空>

图 31　反向查询的整体结构

这个查询询问问题，该问题的回答是互联网风格地址 10.1.0.52。因为所有者名称不知道，任何域名可用作占位符(并被忽略)。通常使用单一零值八位位组(符号化根)，因为它可以尽量减小消息的长度。RR 的 TTL 没有意义。对这个查询的响应或许如图 32 所示。

首部	OPCODE=RESPONSE, ID=997
问题	QTYPE=A, QCLASS=IN, QNAME=VENERA.ISI.EDU
回答	VENERA.ISI.EDU A IN 10.1.0.52
权威	<空>
附加	<空>

图 32　反向查询举例中的响应

注意，在对反向查询的响应中 QTYPE 与该反向查询回答部分中的 TYPE 字段相同。当该反向查询不是唯一的时，对反向查询的响应可以包括多个问题。如果响应中问题部分不为空，那么，修改回答部分中的 RR，以便与在第一个 QNAME 中的 RR 的精确副本相一致。

6-4-3　反向查询处理

支持反向查询的名称服务器，通过穷举搜索它们的数据库便可支持这些操作，但当数据库规模增加时穷举搜索将变得难以实现。替代方法是按照搜索密钥反转数据库。

对于支持多区域和大量数据的名称服务器，推荐的方法是为每个区域单独反演。当刷新期间特定区域被改变时，仅需要重做该区域的反演。

将来的域系统版本可能支持这类反演转换，但是本版本不支持。

6-5　完整查询和响应

在 RFC-882 和 RFC-883 中介绍的可选的完整查询业务已经被删除。将来，对业务重新设计后或许可以应用。

第 7 章　解析器实现

推荐的解析器算法的顶层设计在[RFC-1034]中讨论。本节讨论实现细节，假设采用本备忘录名称服务器实现部分建议的数据库结构。

7-1　将用户请求转换为查询

解析器采取的第一个步骤是以指定名称(它匹配指定的 QTYPE 和 QCLASS)，采用适合本地 OS 的格式，将客户端请求转换为 RRs 的搜索规格。在可能的情况下，QTYPE 和 QCLASS 应当对应单一类型和单一类，因为这更方便缓存数据应用。之所以这样是因为缓存器中存在一种类型数据不能证实其他类型数据的存在或不存在，因此唯一保险的方法是咨询权威资源。如果使用 QCLASS=*，那么，将不能得到权威回答。

因为解析器必须能够复用多个请求，如果解析器高效执行它的功能，每个悬而未决的请求通常用某个状态信息块代表。这个状态块一般包括：

- 指出请求开始时间的时间戳。该时间戳决定数据库中的 RRs 是可以使用的还是已经过期的。这个时间戳使用前面讨论区域和缓存器中 RR 存储时谈及的绝对时间格式。注意，当 RRs TTL 指示相对时间时，该 RR 必须适时发生，因为它是区域的一部分。当 RR 有绝对时间时，该 RR 是缓存器的一部分，RR 的 TTL 与请求的开始时间戳比较。

  注意，使用时间戳优于使用当前时间，因为时间戳使得带零值 TTLs 的 RRs 能够以通常方式进入缓存器，但是仍然由目前的请求使用，即使由于系统加载、查询重发超时等造成许多秒的时间耽搁之后。
- 某种限制工作量的参数(将为这个请求执行这些参数)。

  必须限制解析器为响应客户端请求而执行的工作量，以便避免数据库中出错(诸如循环 CNAME 参考)，以及避免操作问题(诸如网络分区，网络分区阻止解析器访问它需要的名称服务器)。尽管对解析器重传特定查询到特定名称服务器地址的次数的本地限制是必不可少的，解析器应当有综合的预请求计数器(该计数器限制在单一请求上的工作量)。计数器应当被设置到某个初始值，并且只要解析器执行任何动作(重传超时，重传，等)就递减。如果此计数器过零，用临时错误终止此请求。

  注意，如果解析器结构允许一个请求以并行方式开始其他请求(诸如因为一个请求需要访问名称服务器从而引起对该名称服务器地址的并行解析)，引起的请求应当以较低的计数器开始。这样可以阻止数据库中的循环引用启动解析器动作的连锁反应。
- 对 SLIST 数据结构的讨论请参阅[RFC-1034]。

  如果请求必须等待来自外地名称服务器的回答，这个结构跟踪请求的状态。

7-2　发送请求

如[RFC-1034]所述，解析器的基本任务是系统说明将回答客户端请求的查询，以及引导该查询到能够提供信息的名称服务器。解析器通常仅有关于应该询问哪个服务器的强烈暗示(以 NS RRs 形式)，解析器或许必须根据 CNAMEs 修改查询，或者根据授权响应修改解析器正在询问的名称服务器的设置，这些授权响应告诉解析器哪些名称服务器更接近想要的信息。除了由客户端请求的信息以外，解析器或许必须请求它自己的业务，以便确定解析器希望访问的名称服务器的地址。

在任何情况，本备忘录中使用的模型假设解析器同时关注多个请求，这些请求有些来自

客户端，有些产生于内部。每个请求由某个状态信息代表，想要的行为是解析器传送请求到名称服务器，采用的方法应能尽量扩大请求被回答的可能性，尽量减小请求耗用的时间，以及避免过量传送。关键算法使用请求的状态信息，选择下一个将要查询的名称服务器地址，关键算法也计算如果响应没有达到将引起下一个动作的超时。下一个动作通常是传送到某个其他服务器，除非是出现到客户端的临时错误。

解析器总是用待查询服务器名称列表(SLIST)开始。这个列表是所有 NS RRs，这些 NS RRs 对应解析器知道的最近的祖宗区域。为了避免启动问题，如果解析器没有适当的通用 NS RRs，解析器应当有一组它将询问的默认服务器。接着，解析器将所有这些已知的名称服务器的地址添加到 SLIST，当解析器有这些名称服务器的名称但没有其地址时，解析器可以并行开始请求，以便获得它们的地址。

为了完成 SLIST 的初始化，解析器附加它有的无论什么历史信息到 SLIST 中的每个地址。历史信息通常包括地址响应时间的某种加权平均、地址的平均成功率(即，多少时间有一个真正响应该请求的地址)。注意，这条信息应当以每个地址保存，而不是以每个名称服务器保存，因为特定服务器的响应时间和平均成功率会随地址的不同有相当大的变化。也要注意，这条信息实际上是解析器地址/服务器地址对特有的，所以，有多个地址的解析器可以希望为它的每一个地址保存独立的历史信息。此步骤的一部分必须处理没有这类历史信息的地址；在这种情况，预期 5-10 秒的往返时间应当是最糟糕情况，对于同一本地网络等情况，往返时间可估计的低一些。

注意，只要授权跟随其后，此解析器算法重新初始化 SLIST。

此信息建立可用名称服务器地址的部分排序。每次选择一个地址，状态应当被改变，以便防止该地址被重复选择，直到所有其他地址都被尝试过。每次发送的超时应当 50%-100% 地大于平均预计值，以便考虑到响应中的差异。

一些细节考虑：

- 解析器可能遭遇这样的情况，那里，在 SLIST 中命名的任何名称服务器的地址都不能得到，并且在该列表中的服务器们恰好是通常用于查找它们自己的地址的服务器。当胶地址 RRs 的 TTL 比标记授权的 NS RRs 的 TTL 小时，或当解析器缓存 NS 搜索的结果时，这种情况一般会发生。解析器应当检测这个条件并在下一个祖宗区域重启搜索，或作为替代在根区域重启搜索。
- 如果解析器收到服务器出错或其它来自名称服务器的怪异响应，解析器应当从 SLIST 中将该服务器删除，解析器可能希望安排到下一个候选服务器地址的立即发送。

7-3　处理响应

对到达的响应数据报进行处理，第一步是分析该响应。这个流程应当包括：

- 检验首部的合理性。当期望响应到达时，到达的却是查询数据报，抛弃该数据报。
- 分析消息的各个部分，确信所有 RRs 具有正确的格式。
- 作为可选步骤，检验到达数据的 TTLs，寻找有过长 TTLs 的 RRs。如果 RR 有过长的 TTL，比如说大于一周，或者抛弃整个响应，或者限制该响应中的所有 TTLs 为一周。

下一步是将响应与当前的解析器请求相匹配。推荐的策略是使用域首部中的 ID 字段做初步匹配，接着验证问题部分与当前期望的信息是一致的。这要求发送算法把域 ID 字段的几个比特用在某种请求标识符上。这一步有几个细微之处需注意：

- 某些名称服务器是从不同地址发送它们的响应，而不是从用于接收查询的同一个地址发送。即，解析器不能认为：响应将来自与解析器发送相应查询去时服务器

使用的相同地址。这个名称服务器缺陷典型会在 UNIX 系统中遇到。

- 如果解析器重新发送特定请求到名称服务器，解析器应当能够使用来自任何发送的响应。然而，如果解析器正在使用该响应去取样访问该名称服务器的往返时间，解析器必须能够确定哪一次发送匹配该响应(并为每个外发消息维持发送次数)，或仅根据初始的发送计算往返时间。
- 名称服务器偶尔会没有区域的当前副本(按照某些 NS RRs 名称服务器应当有该副本)。解析器应当简单地从当前 SLIST 中删除该名称服务器，并继续。

7-4 使用缓存器

一般讲，我们希望解析器缓存它从响应中接收到的所有数据，因为将来回答客户端请求时这些数据可能有用。然而，下述几种类型的数据不应当缓存：

- 当可以得到特定所有者名称的几个相同类型 RRs 时，解析器应当或者都缓存它们，或者一个都不缓存。当响应被截断时，以及解析器不知道是否响应有完整的集合时，解析器不应当缓存可能的部分 RRs 集合。
- 在使用中，缓存的数据绝不应当优先于权威数据，因此，如果缓存将引起这种情况发生，不应当缓存该数据。
- 反向查询的结果不应当缓存。
- 标准查询的结果(其中 QNAME 包含“*”)标明是否数据或许被用于构建通配符。因为缓存器不一定必须包括现有的 RRs 或区域边界信息，该信息是限制通配符 RRs 应用所必须的。
- 可靠性有疑问的响应中的 RR 数据。当解析器收到不请自来的响应或不是所请求的 RR 数据时，它应当抛弃，不需要缓存。基本的思路是在缓存任何分组前应当执行所有关于该分组的完整性检验。

与此类似，当解析器在响应中有某个名称的一组 RRs 并想缓存该组 RRs 时，解析器应当针对已经存在的 RRs 检验它的缓存器。根据不同情况，或者在响应中的数据优先，或者在缓存器中的数据优先，但是二者绝不能混用。如果响应中的数据来自回答部分中的权威数据，它总是优先。

第 8 章 邮件支持

域名系统定义了映射邮箱到域名的标准，定义了使用邮箱信息推导出邮件路由信息的两种方法。第一种方法称为邮件交换绑定，另一种方法是邮箱绑定。邮箱编码标准和邮件交换绑定是 DNS 正式协议的一部分，是在互联网中实现邮件路由的推荐方法。邮箱绑定具有试验性质，仍在发展和变化中。

邮箱编码标准假设邮箱名称形如“<local-part>@<mail-domain>”。尽管在这些部分的每一部分中允许的语法在各种邮件的互连网络间变化很大，ARPA 互联网首选语法在[RFC-822]中介绍。

DNS将<local-part>编码为单一标签，将<mail-domain>编码为域名。来自<local-part>的单一标签位于来自<mail-domain>的域名的前面，形成对应该邮箱的域名。于是，邮箱HOSTMASTER@SRINIC.ARPA被映射成域名HOSTMASTER.SRI-NIC.ARPA。如果<local-part>包含圆点或其他专用字符，在主文件中它的表示法需要使用反斜杠引用，以便确保域名被适当编码。例如，邮箱Action.domains@ISI.EDU将被表示为Action\.domains.ISI.EDU。

8-1 邮件交换绑定

邮件交换绑定使用邮箱规格的<mail-domain>部分，该部分确定邮件应当发送到哪里。

甚至不咨询<local-part>。[RFC-974]详细介绍了这种方法，在试图使用邮件交换支持以前，应当查阅[RFC-974]。这个方法的优点之一是它以查询功能内间接的另一层为代价，使邮件目的地命名与用于支持邮件业务的主机分开。然而，在<local-part>中，附加层应当去除使用复杂的“%”、“！”等编码的需要。

此方法的精华是用<mail-domain>作域名，找出类型 MX RRs，该 MX RRs 列出愿意接收<mail-domain>邮件的主机，同时使用优先权值，该值按照由<mail-domain>的管理者规定的次序排序主机。

在本备忘录，许多举例中使用<mail-domain>ISI.EDU，连同主机 VENERA.ISI.EDU 和 VAXA.ISI.EDU，作为 ISI.EDU 的邮件交换。如果邮件收发程序有 Mockapetris@ISI.EDU 的消息，它将通过查询 ISI.EDU 的 MX RRs，路由该消息。在 ISI.EDU 中的 MX RRs 命名 VENERA.ISI.EDU 和 VAXA.ISI.EDU，A 类型查询可以发现该主机地址。

8-2　邮箱绑定(试验)

在邮箱绑定中，邮件收发程序使用完整的邮件目的地规格构建域名。邮箱的编码域名用作QTYPE=MAILB查询中的QNAME字段。

这个查询的几个可能的结果是：

1、查询能够返回名称错误，该名称错误指出该邮箱作为域名不存在。
从长远观点看，这将指出该特定邮箱不存在。然而，直到邮箱绑定普遍使用为止，这个出错条件应当被解释为由全球部分标识的组织不支持邮箱绑定。在此点，适当的程序将恢复到交换绑定。

2、查询能够返回 Mail Rename (MR) RR。
MR RR 在它的 RDATA 字段携带新邮箱规格。邮件发送程序应当用新邮箱代替旧邮箱，并重试操作。

3、查询能够返回 MB RR。
MB RR 在它的 RDATA 字段携带主机域名。邮件收发程序应当通过无论什么可用的协议(例如，b、SMTP)，交付消息到那个主机。

4、查询能够返回一个或多个 Mail Group (MG) RRs。
这个条件意味着该邮箱实际上是邮件列表或邮件组，不是单个邮箱。每个 MG RR 有 RDATA 字段，该字段标识是组成员的邮箱。邮件收发程序应当递交此消息的副本给每个成员。

5、查询能够返回 MB RR，以及一个或多个 MG RRs。
这个条件意味着邮箱实际上是邮件列表。邮件收发程序可以或者交付消息给由 MB RR 指定的主机(该主机依次将该消息交付给所有成员)，或者使用 MG RRs 自己做此扩展工作。

在这些情况中的任何一种情况，响应可以包括 Mail Information (MINFO) RR。这个 RR 通常与邮件组关联，但是合法地带有 MB。MINFO RR 标识两个邮箱。其中一个标识原始邮箱名称的负责人。这个邮箱应当用于添加到邮件组的请求等。在 MINFO RR 中的第 2 个邮箱名称标识应当接收邮件失败出错消息的邮箱。这特别适用于邮件列表，当成员名称中的错误应当报告给不是发送消息到该列表的人时。

将来，新的字段可以添加到这个 RR。

第 9 章　参考资料和书目

[Dyer 87]　S.Dyer, 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] J. Quarterman, 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.

[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. 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. 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