Система доменных имен

(материалы книги П.Б. Храмцова)

1. Система доменных имен Internet. Немного истории и принципы построения иерархии имен.

Когда при деловом общении представители двух фирм обмениваются визитками, то в них (визитках) обязательно будут указаны адрес электронной почты и имя корпоративного Web-узла компании. При этом можно также услышать, как собеседники обмениваются "интернет-адресами" ("электронными адресами") компаний. Во всех выше перечисленных случаях так или иначе речь идет об использовании доменных имен.

В адресе электронной почты формально доменным именем можно считать то, что написано после символа коммерческого ат - "@". Например, в user@corp.ru доменное имя почтового узла - corp.ru.

Имя Web-узла - это доменное имя этого узла. Например, Web-узел компании Microsoft имеет доменное имя Microsoft.com.

В большинстве случаев при поиске информации в Сети мы перебираем доменные имена или следуем по ссылкам, в нотации которых опять же используются доменные имена.

Довольно часто наряду со словосочетанием "интернет-адрес" употребляют "доменный адрес". Вообще говоря, ни того, ни другого понятий в сетях TCP/IP не существует. Есть числовая адресация, которая опирается на IP-адреса, (группа из 4-ех чисел, разделенных символом ".") и Internet-сервис службы доменных имен (Domain Name System - DNS).

Числовая адресация удобна для компьютерной обработки таблиц маршрутов, но совершенно (здесь мы несколько утрируем) не приемлема для использования ее человеком. Запомнить наборы цифр гораздо труднее, чем мнемонические осмысленные имена.

Тем не менее, установка соединений для обмена информацией в Интернет осуществляется по IP-адресам. Символьные имена системы доменных имен - суть сервис, который помогает найти необходимые для установки соединения IP-адреса узлов сети.

Тем не менее, для многих пользователей именно доменное имя выступает в роли адреса информационного ресурса. В практике администрирования локальных сетей нередки ситуации, когда пользователи жалуются администратору сети на недоступность того или иного сайта или долгую загрузку страниц. Причина может крыться не в том, что сегмент сети потерял связь с остальной сетью, а в плохой работе DNS - нет IP-адреса, нет и соединения.

DNS существовала не с момента рождения TCP/IP сетей. Поначалу для облегчения взаимодействия с удаленными информационными ресурсами в Интернет стали использовать таблицы соответствия числовых адресов именам машин.

Авторство создания этих таблиц принадлежит доктору Постелю (Dr. Jon Postel - автор многих RFC - Request For Comments). Именно он первым поддерживал файл hosts.txt, который можно было получить по FTP.

Современные операционные системы тоже поддерживают таблицы соответствия IP-адреса и имени машины (точнее хоста) - это файлы с именем hosts. Если речь идет о системе типа Unix, то этот файл расположен в директории /etc и имеет следующий вид:

```
127.0.0.1 localhost
144.206.130.137 polyn Polyn polyn.net.kiae.su polyn.kiae.su
144.206.160.32 polyn Polyn polyn.net.kiae.su polyn.kiae.su
144.206.160.40 apollo Apollo www.polyn.kiae.su
```

Пользователь для обращения к машине может использовать как IP-адрес машины, так и ее имя или синоним (alias). Как видно из примера, синонимов может быть много, и, кроме того, для разных IP-адресов может быть указано одно и то же имя.

Напомним еще раз, что по самому мнемоническому имени никакого доступа к ресурсу получить нельзя. Процедура использования имени заключается в следующем:

- сначала по имени в файле hosts находят IP-адрес,
- затем по IP-адресу устанавливают соединение с удаленным информационным ресурсом.

Обращения, приведенные ниже аналогичны по своему результату - инициированию сеанса telnet с машиной Apollo:

```
telnet 144.206.160.40
```

или

```
telnet Apollo
```

или

```
telnet www.polyn.kiae.su
```

В локальных сетях файлы hosts используются достаточно успешно до сих пор. Практически все операционные системы от различных клонов Unix до Windows последних версий поддерживают эту систему соответствия IP-адресов именам хостов.

Однако такой способ использования символьных имен был хорош до тех пор, пока Интернет был маленьким. По мере роста Сети стало затруднительным держать большие согласованные списки имен на каждом компьютере. Главной проблемой стал даже не размер списка соответствий, сколько синхронизация его содержимого. Для того, что бы решить эту проблему, была придумана DNS.

DNS была описана Полом Мокапетрисом (Paul Mockapetris) в 1984. Это два документа: RFC-882 и RFC-883 (Позже эти документы были заменены на RFC-1034 и RFC-1035). Пол Мокапетрис написал и реализацию DNS - программу JEEVES для ОС Tops-20. Именно на нее в RFC-1031 предлагается перейти администраторам машин с ОС Tops-20 сети MILNET. Не будем подробно излагать содержание RFC-1034 и RFC-1035. Ограничимся только основными понятиями.

Роль имени (доменного имени) в процессе установки соединения осталось прежним. Это значит, что главное, для чего оно нужно, - получение IP адреса. Соответственно этой роли,

любая реализация DNS является прикладным процессом, который работает над стеком протоколов межсетевого обмена TCP/IP. Таким образом, базовым элементом адресации в сетях TCP/IP остался IP-адрес, а доменное именование (система доменных имен) выполняет роль вспомогательного сервиса.

Система доменных имен строится по иерархическому принципу. Точнее по принципу вложенных друг в друга множеств. Корень системы называется "root" (дословно переводится как "корень") и никак не обозначается (имеет пустое имя согласно RFC-1034).

Часто пишут, что обозначение корневого домена - символ ".", но это не так, точка - разделитель компонентов доменного имени, а т.к. у корневого домена нет обозначения, то полное доменное имя кончается точкой. Тем не менее символ "." достаточно прочно закрепился в литературе в качестве обозначения корневого домена. От части это вызвано тем, что в файлах конфигурации серверов DNS именно этот символ указывается в поле имени домена (поле NAME согласно RFC-1035) в записях описания ресурсов, когда речь идет о корневом домене.

Корень - это все множество хостов Интернет. Данное множество подразделяется на домены первого или верхнего уровня (top-level или TLD). Домен ru, например, соответствует множеству хостов российской части Интернет. Домены верхнего уровня дробятся на более мелкие домены, например, корпоративные.

В 80-е годы были определены первые домены первого уровня (top-level): gov, mil, edu, com, net. Позднее, когда сеть перешагнула национальные границы США появились национальные домены типа: uk, jp, au, ch, и т.п. Для СССР также был выделен домен su. После 1991 года, когда республики Союза стали суверенными, многие из них получили свои собственные домены: ua, ru, la, li, и т.п.

Однако Интернет не СССР, и просто так выбросить домен su из системы доменных имен нельзя. На основе доменных имен строятся адреса электронной почты и доступ ко многим другим информационным ресурсам Интернет. Поэтому гораздо проще оказалось ввести новый домен к существующему, чем заменить его.

Если быть более точным, то новых имен с расширением su в настоящее время ни один провайдер не выделяет (делегирует). Однако у многих существует желание возобновить процесс делегирования доменов в зоне SU.

Со списком доменов первого уровня (top-level) и их типами можно ознакомиться, например, в материале "Общая информация о системе доменных имен".

Как уже было сказано, вслед за доменами первого уровня(top-level) следуют домены, определяющие либо регионы (msk), либо организации (kiae). В настоящее время практически любая организация может получить свой собственный домен второго уровня. Для этого надо направить заявку провайдеру и получить уведомление о регистрации (см. "Как получить домен").

Далее идут следующие уровни иерархии, которые могут быть закреплены либо за небольшими организациями, либо за подразделениями больших организаций.

Часть дерева доменного именования можно представить следующим образом:

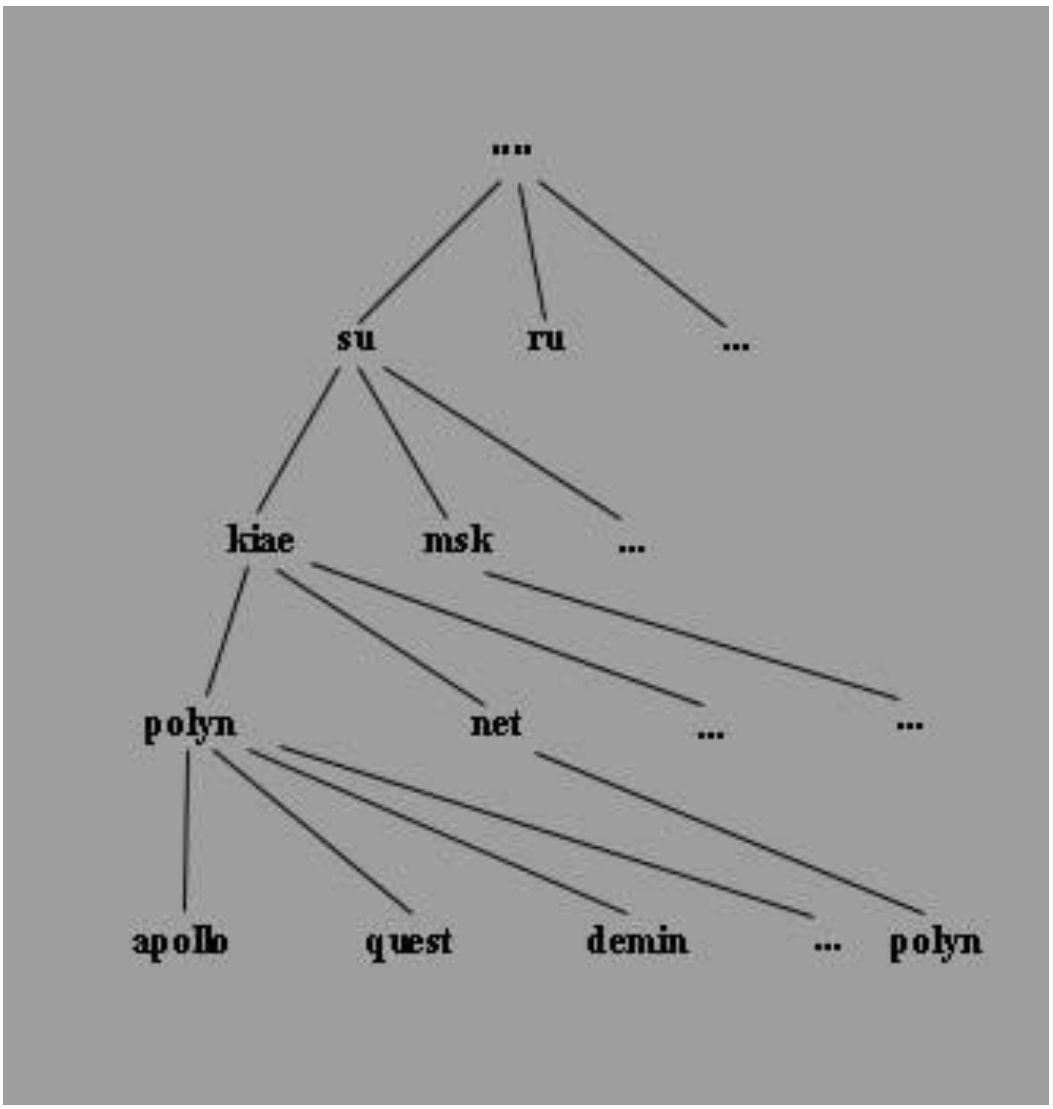

Рис.1. Пример части дерева доменных имен.

Корень дерева не имеет имени метки. Поэтому его обозначают как "". Остальные узлы дерева метки имеют. Каждый из узлов соответствует либо домену, либо хосту. Под хостом в этом дереве понимают лист, т.е. такой узел ниже которого нет других узлов.

Именовать хост можно либо частичным именем, либо полным именем. Полное имя хоста - это имя, в котором перечисляются слева направо имена всех промежуточных узлов между листом и корнем дерева доменного именования, при этом начинают с имени листа, а кончают корнем, например:

polyn.net.kiae.su.

Частичное имя - это имя, в котором перечислены не все, а только часть имен узлов, например:

polyn
apollo.polyn
quest.polyn.kiae

Обратите внимание на то, что в частичных (неполных именах) символ точки в конце имени не ставится. В реальной жизни программное обеспечение системы доменных имен расширяет неполные имена до полных прежде, чем обратиться к серверам доменных мен за IP-адресом.

Слово "Хост" не является в полном смысле синонимом имени компьютера, как это часто упрощенно представляется. Во-первых, у компьютера может быть множество IP-адресов, каждому из которых можно поставить в соответствие одно или несколько доменных имен. Во-вторых, одному доменному имени можно поставить в соответствие несколько разных IP-адресов, которые, в свою очередь могут быть закреплены за разными компьютерами.

Еще раз обратим внимание на то, что именование идет слева направо, от минимального имени хоста (от листа) к имени корневого домена. Разберем, например, полное доменное имя demin.polyn.kiae.su. Имя хоста - demin, имя домена, в который данный хост входит, - polyn, имя домена, который охватывает домен polyn, т.е. является более широким по отношению к polyn, - kiae, в свою очередь последний (kiae) входит в состав домена su.

Имя polyn.kiae.su - это уже имя домена. Под ним понимают имя множества хостов, у которых в их имени присутствует polyn.kiae.su. Вообще говоря, за именем polyn.kiae.su может быть закреплен и конкретный IP-адрес. В этом случае кроме имени домена данное имя будет обозначать и имя хоста. Такой прием довольно часто используется для обеспечения коротких и выразительных адресов в системе электронной почты.

Имена хоста и доменов отделяются друг от друга в этой нотации символом ".". Полное доменное имя должно оканчиваться символом ".", т.к. последняя точка отделяет пустое имя корневого домена от имени домена верхнего уровня. Часто в литературе и в приложениях эту точку при записи доменного имени опускают, используя нотацию неполного доменного имени даже в том случае, когда перечисляют все имена узлов от листа до корня доменного именования.

Следует иметь в виду, что доменные имена в реальной жизни достаточно причудливо отображаются на IP-адреса, а тем более на реальные физические объекты (компьютеры, маршрутизаторы, коммутаторы, принтеры и т.п.), которые подключены к сети.

Компьютер, физически установленный и подключенный к Сети в далекой Америке, может совершенно спокойно иметь имя из российского корпоративного домена, например, chalajva.ru, и наоборот, компьютер или маршрутизатор российского сегмента может иметь имя из домена com. Последнее, к слову сказать, встречается гораздо чаще.

Более того, один и тот же компьютер может иметь несколько доменных имен. Возможен вариант, когда за одним доменным именем может быть закреплено несколько IP-адресов, которые реально назначены различным серверам, обслуживающим однотипные запросы. Таким образом, соответствие между доменными именами и IP-адресами в рамках системы доменных имен не является взаимно однозначным, а строится по схеме "многие к многим".

Несколько последних замечаний были призваны обратить внимание читателя на тот факт, что иерархия системы доменных имен строго соблюдается только в самих именах и отображает только вложенность именования и зоны ответственности администраторов соответствующих доменов.

Следует также упомянуть о канонических доменных именах. Это понятие встречается в контексте описания конфигураций поддоменов и зон ответственности отдельных серверов доменных имен. С точки зрения дерева доменных имена не разделяют на канонические и неканонические, но с точки зрения администраторов, серверов и систем электронной почты такое разделение является существенным. Каноническое имя - это имя, которому в соответствие явно поставлен IP-адрес, и которое само явно поставлено в соответствие IP-адресу. Неканоническое имя - это синоним канонического имени. Более подробно см. "настройка BIND".

Наиболее популярной реализацией системы доменных имен является Berkeley Internet Name Domain (BIND). Но эта реализация не единственная. Так в системе Windows NT 4.0 есть свой сервер доменных имен, который поддерживает спецификацию DNS.

Тем не менее, даже администраторам Windows желательно знать принципы функционирования и правила настройки BIND, т.к. именно это программное обеспечение обслуживает систему доменных имен от корня до TLD (Top Level Domain).

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. W.Lazear. RFC-1031. MILNET NAME DOMAIN TRANSITION. 1987. (http://www.ietf.org/rfc/rfc1031.txt?number=1031)
4. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. http://www.dns.net/dnsrd/docs/ - коллекция ссылок на документы о системе доменных имен.
2. http://www.internic.net/faqs/authoritative-dns.html - коротенькое описание назначения системы доменных имен.
3. http://www.icann.org/ - сайт организации, которая в ответе за именование в Интернет.
4. http://www.ispras.ru/~grn/dns/index.html - Г.В. Ключников. Служба доменных имен (Domain Name System). 1999. На самом деле, это отличная компиляция приведенных в конце книжки первоисточников. Примеры взяты из этих же первоисточников. Очень качественный перевод и грамотно скомпонованный текст.
5. http://www.ibb.ru/articles/stat_3.phtml - из серии "DNS за пять минут" J, но в качестве введения в тему данный материал может пригодиться.
6. http://www.pi2.ru:8100/prof/techsupp/dns.htm - своеобразное описание системы доменных имен. Во всяком случае, самобытное. Но некоторые аспекты освещены довольно необычно.

2. Как работает система доменных имен.

Согласно руководящим материалам (RFC-1034, RFC-1035) система доменных имен состоит из трех основных частей:

- Всего множества доменных имен (domain name space)
- Серверов доменных имен (domain name servers)
- Клиенты DNS (Resolver-ы)

Множество доменных имен было подробно рассмотрено в материале "Доменная адресация. Немного истории и принципы построения", поэтому сосредоточимся на двух оставшихся компонентах серверах и resolver-ах.

Сервис системы доменных имен строится по схеме "клиент-сервер". В качестве клиентской части выступает прикладной процесс, который запрашивает информацию о соответствии имени адресу (или наоборот адреса имени). Это программное обеспечение и называют resolver. В качестве сервера выступает прикладная программа-сервер.

Чаще всего, Resolver не является какой-либо программой или системной компонентой. Это набор процедур из библиотеки прикладного программного обеспечения (например, из библиотеки libc), которые позволяют программе, отредактированной с ними, выполнять запросы к системе доменных имен и получать ответы на них. Эти процедуры обращаются к серверу доменных имен и, таким образом, обслуживает запросы прикладных программ пользователя.

Ряд производителей операционных систем, например, Sun или SGI, предлагали решения, в которых resolver являлся отдельным процессом, и прикладные программы через него реализовывали взаимодействие с DNS.

Другой пример реализации resolver-а - это браузеры Nescape некоторых версий, где для ускорения процесса получения ответов на запросы к DNS запускался отдельный процесс resolver-а.

Самостоятельный resolver может быть собран и в BIND версии 9. Это так называемый lightweight resolver. Он состоит из rosolver-демона и библиотеки взаимодействия с этим демоном, процедуры которой линкуются с прикладным ПО. Данный resolver позволяет не только посылать запросы к серверу доменных имен, но кэшировать соответствия между доменным именем и IP-адресом.

В качестве серверов доменных имен чаще всего используются различные версии BIND (Berkeley Internet Name Domain). Если сервер реализован на платформе Windows, то тогда используют решение от Microsoft, хотя для этой платформы также существуют версии BIND.

Общую схему взаимодействия различных компонентов системы доменных имен можно изобразить так, как это представлено на следующем рисунке:

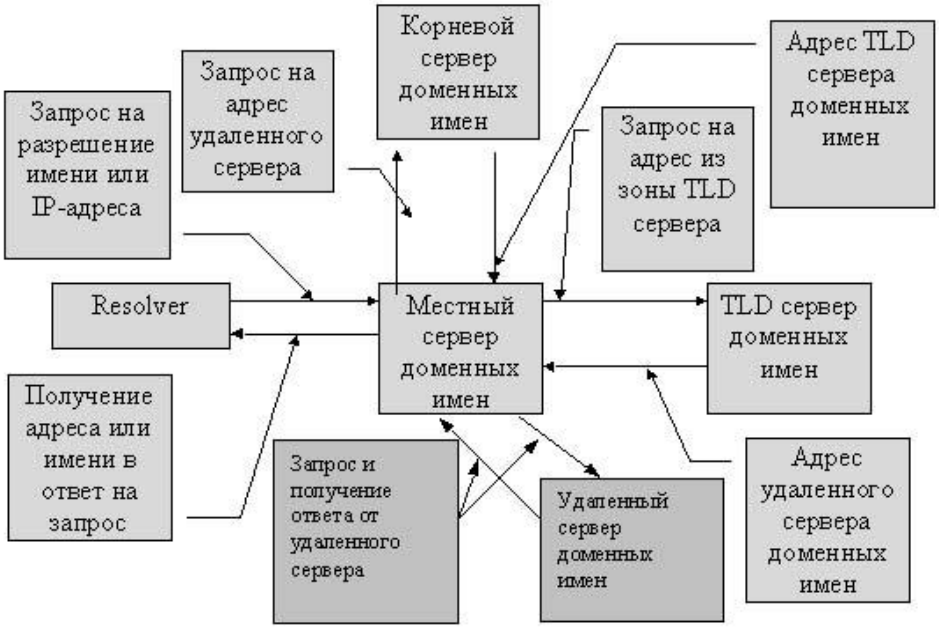

Рис1. Рекурсивный запрос resolver и нерекурсивная (итеративная) процедура на разрешение доменного имени сервером доменных имен.

Данная схема разрешения имени (установки соответствия между именем и IP-адресом) называют нерекурсивной (итеративной). Чем она отличается от рекурсивной мы обсудим позже.

Поясним приведенную схему нерекурсивной процедуры разрешения запроса:

1. Прикладная программа через resolver запрашивает IP-адрес по доменному имени у местного сервера (запрос resolver рекурсивный, т.е. resolver просит server найти ему адрес).
2. Местный сервер сообщает прикладной программе IP-адрес запрошенного имени, выполняя при этом нерекурсивный опрос серверов доменных имен. При этом:
    a. если адрес находится в зоне его (местного сервера) ответственности, сразу сообщает его resolver-у,
    b. если адрес находится в зоне ответственности другого сервера доменных имен, то обращается к корневому серверу системы доменных имен за адресом TLD-сервера (top-level domain server),
    c. обращается к TLD-серверу за адресом,
    d. получает от него адрес удаленного сервера,
    e. обращается к удаленному серверу за адресом,
    f. получает от удаленного сервера адрес,

В данном случае мы рассмотрели вложенность доменного имени второго порядка., т.е. хост имел имя похожее на quest.kuku.ru или даже kuku.ru.

Последнее важно понять, т.к. корпоративные почтовые адреса типа user@kuku.ru как раз и требуют от прикладного программного обеспечения обращения за IP-адресом хоста kuku.ru. TLD-сервер домена ru не обладает информацией о том, какому IP-адресу данное имя соответствует, но при этом он знает какой сервер отвечает за домен kuku.ru.

Если вложенность доменного имени будет большей, скажем третьего уровня (host.department.corp.ru), и этот уровень будет поддерживаться другим сервером доменных

имен, отличным от того, который поддерживает второй уровень вложенности, то тогда нашему локальному серверу удаленный сервер доменных имен передаст не адрес хоста, а адрес нового сервера доменных имен, в зоне ответственности которого находится запрашиваемое имя.

Как видно из приведенной схемы, получение информации из системы доменных имен - это многоходовой процесс, который не реализуется мгновенно. В следующем примере показано как на практике ощущается работа DNS.

При входе в режиме удаленного терминала на компьютер polyn.net.kiae.su по команде:

```
/usr/paul>telnet polyn.net.kiae.su
```

Мы получаем в ответ:

```
/usr/paul>telnet polyn.net.kiae.su
trying 144.206.130.137 ...
login:
.....
```

Строчка, в которой указан IP-адрес компьютера polyn.net.kiae.su, показывает, что к этому времени доменное имя было успешно разрешено сервером доменных имен и прикладная программа, в данном случае telnet получила на свой запрос IP-адрес. Таким образом, после ввода команды с консоли, и до появления IP-адреса на экране монитора прикладная программа осуществила запрос к серверу доменных имен и получила ответ на него.

Довольно часто можно столкнуться с ситуацией, когда после ввода команды довольно долго приходиться ждать ответа удаленной машины, но зато после первого ответа удаленный компьютер начинает реагировать на команды с такой же скоростью, как и ваш собственный персональный компьютер. В этом случае, скорее всего, в первоначальной задержке виноват сервис доменных имен.

Другой пример того же сорта - это программа traceroute. Здесь задержка на запросы к серверу доменных имен проявляется в том, что время ответов со шлюзов, на которых "умирают" ICMP пакеты, указанное в отчете, маленькое, а задержки с отображением каждой строчки отчета довольно большие.

Любопытно, что в системе Windows 3.1 некоторые программы трассировки, показывали сначала IP-адрес шлюза, а только потом, после разрешения "обратного" запроса, заменяли его на доменное имя шлюза. Если сервис доменных имен работал быстро, то эта подмена была практически незаметна, но если сервис работал медленно, то промежутки бывали довольно значительными.

Проверить на сколько "чувствительны" запросы traceroute с точки зрения отображения трассы к использованию сервера доменных имен можно задав поиск трассы с использованием сервера:

>traceroute www.w3.org

и без использования сервера:

>traceroute -n www.w3.org

Если в примере с telnet и ftp мы рассматривали, только "прямые" (forward) запросы к серверу доменных имен, то в примере с traceroute мы впервые упомянули "обратные"(reverse) запросы. В "прямом" запросе прикладная программа запрашивает у сервера доменных имен IP-адрес, сообщая ему доменное имя. При "обратном" запросе прикладная программа запрашивает доменное имя, сообщая серверу доменных имен IP-адрес.

Следует заметить, что скорость разрешения "прямых" и "обратных" запросов в общем случае будет разная. Все зависит от того, где описаны "прямые"(forward) и "обратные"(reverse) зоны в базах данных серверов доменных имен, обслуживающих соответствующие домены (прямой и обратный).

Более подробно этого вопроса мы коснемся при обсуждении файлов конфигурации программы named.

Однако вернемся к обсуждению схемы работы системы доменных имен. Собственно нерекурсивным рассмотренный выше запрос является только с точки зрения сервера. С точки зрения resolver-а процедура разрешения запроса является рекурсивной, так как resolver перепоручил локальному серверу доменных имен заниматься поиском необходимой информации. Согласно RFC-1035, resolver и сам может опрашивать удаленные серверы доменных имен и получать от них ответы на свои запросы.

В этом случае resolver обращается к локальному серверу доменных имен, если не получает от него адреса, то опрашивает сервер корневого домена, получает от него адрес удаленного сервера TLD, опрашивает этот сервер, получает адрес удаленного сервера, опрашивает удаленный сервер и получает IP-адрес, если он посылал, так называемый "прямой" запрос.

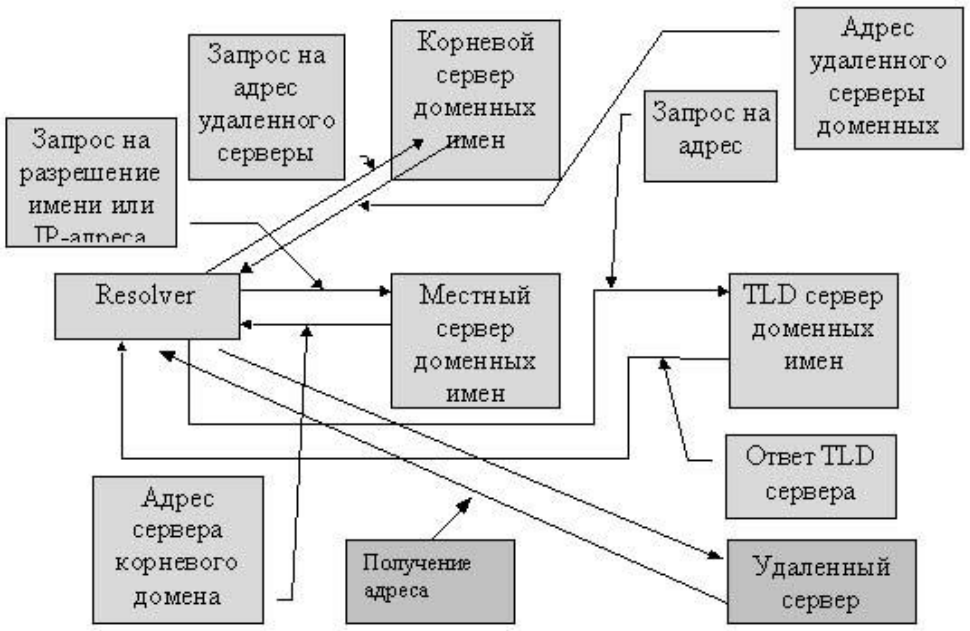

Рис.2. Нерекурсивный запрос resolver-а.

Как видно из этой схемы, resolver сам нашел нужный IP-адрес. Однако общая практика такова, что resolver не порождает нерекурсивных запросов, а переадресовывает их локальному серверу доменных имен.

Локальный сервер и resolver не все запросы выполняют по указанной процедуре. Дело в том, что существует кэш, который используется для хранения в нем полученной от удаленного сервера информации.

Самые умные resolver-ы, такие, например, как resolver Windows 2000 Server и BIND 9 умеют поддерживать кэш, в котором сохраняют не только удачно установленные соответствия имени и адреса (positive response), но и так называемые "негативные" отклики (negative response results) на запросы. Кроме того, эти resolver-ы упорядочивают отклики об адресах серверов в соответствии с принятыми в них (resolver-ах) алгоритмами выставления предпочтений, которые базируются на времени отклика сервера.

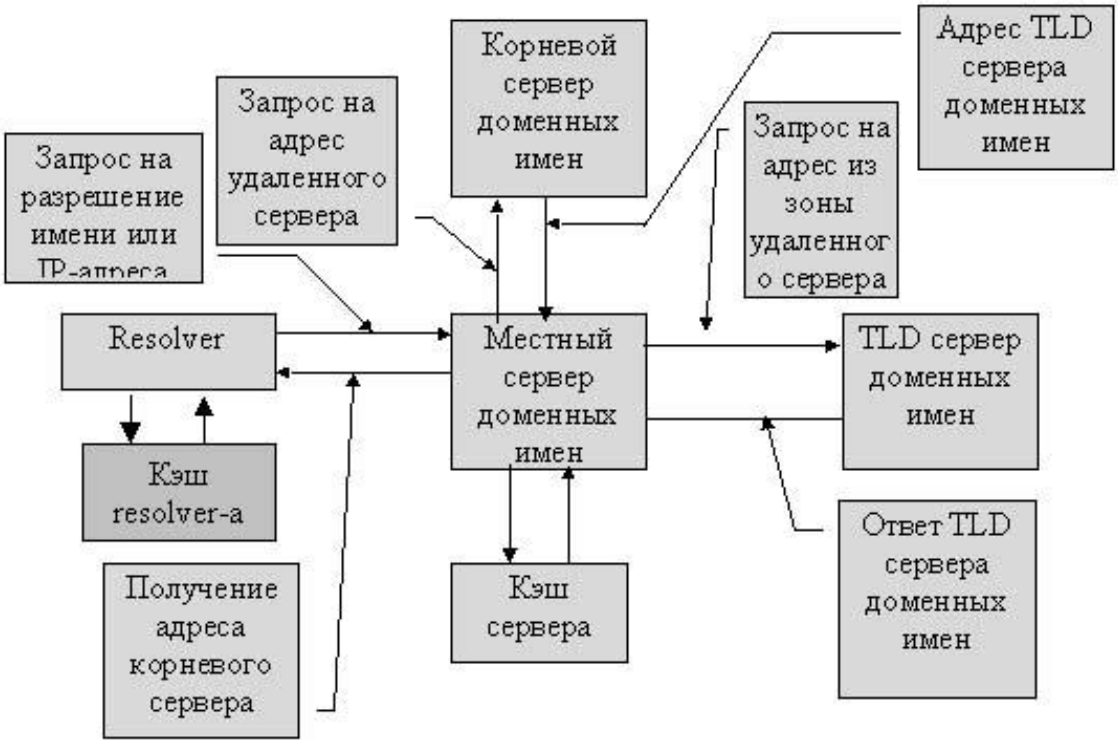

Рис.3. Схема разрешения запросов с кэшированием ответов.

Если пользователь обращается в течение короткого времени к одному и тому же ресурсу сети, то запрос на удаленный сервер не отправляется, а информация ищется в кэше.

Вообще говоря, прядок обработки запросов можно описать следующим образом:

1. поиск ответа в локальном кэше
2. поиск ответа на локальном сервере
3. поиск информации в сети.

При этом кэш может быть как у resolver-а, так и у сервера.

Прежде, чем мы двинемся дальше в нашем понимании работы системы доменных имен введем еще одно понятие - зону.

Между доменом и зоной существует разница, которую бывает трудно найти, но всегда нужно иметь в виду. Домен - это все множество машин, которые относятся к одному и тому же доменному имени. Например, все машины, которые в своем имени имеют

постфикс kiae.su относятся к домену kiae.su. Зона - это "зона ответственности" конкретного сервера доменных имен, т.е. понятие домена шире, чем понятие зоны. Если домен разбивается на поддомены, то у каждого из них может появиться свой сервер. При этом зоной ответственности сервера более высокого уровня будет только та часть описания домена, которая не делегирована другим серверам. Разбиение домена на поддомены и организация сервера для каждого из них называется делегирование прав управления зоной соответствующему серверу доменных имен, или просто делегированием зоны.

При настройке сервера, в его файлах конфигурации можно непосредственно прописать адреса серверов зон. В этом случае обращения к корневому серверу не производятся, т.к. местный сервер сам знает адреса удаленных серверов зон, которым он же и делегировал права управления этими зонами.

Существует еще и другой вариант работы сервера, когда он не запрашивает корневой сервер на предмет адреса удаленного сервера доменных имен. Это происходит тогда, когда незадолго до этого сервер уже разрешал задачу получения IP-адреса из того же домена. Если требуется получить IP-адрес удаленного сервера доменных имен, отвечающего за домен, то он будет просто извлечен из буфера сервера (кеша), т.к. в течение определенного времени, заданного в конфигурации описания зоны (Time To Live - TTL), этот адрес будет храниться в кэше сервера. А попал он туда как результат выполнения предыдущего запроса.

Кроме нерекурсивной процедуры разрешения имен возможна еще и рекурсивная процедура разрешения имен. Ее отличие от описанной выше нерекурсивной процедуры состоит в том, что удаленный сервер сам опрашивает свои серверы зон, а не сообщает их адреса местному серверу доменных имен. Рассмотрим этот случай более подробно.

На рисунке 4 продемонстрирована нерекурсивная процедура разрешения IP-адреса по доменном имени. Основная нагрузка в этом случае ложится на местный сервер доменных имен, который осуществляет опрос всех остальных серверов. Для того, чтобы сократить число таких обменов, если позволяет объем оперативной памяти, можно разрешить буферизацию (кэширование) адресов. В этом случае число обменов с удаленными серверами сократится.

На рисунке 5 удаленный сервер домена сам разрешает запрос на получение IP-адреса хоста своего домена по рекурсивному запросу, используя при этом нерекурсивный опрос своих серверов поддоменов.

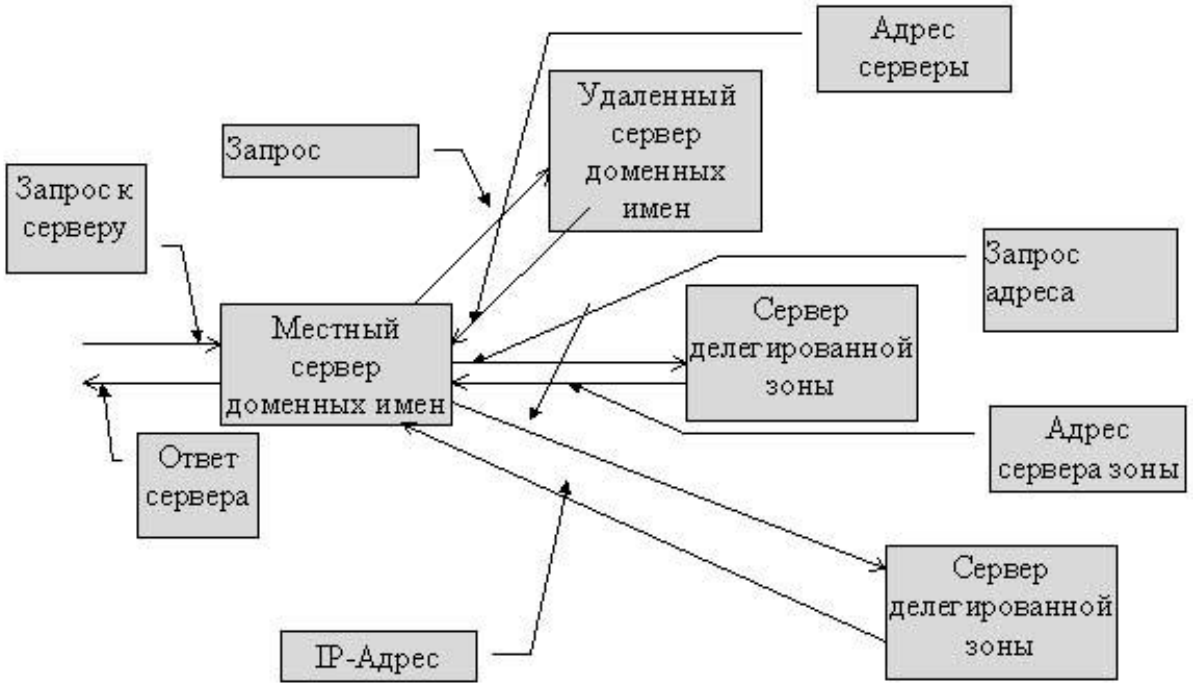

Рис 4. Нерекурсивная обработка запроса местным сервером доменных имен на получение IP-адреса по доменному имени.

Рис. 5. Рекурсивная (для местного сервера) и нерекурсивная (для удаленного сервера) процедуры разрешения адреса по IP-имени.

При этом локальный сервер сразу получает от удаленного адрес хоста, а не адреса серверов поддоменов. Удаленному серверу при этом должно быть разрешено

обслуживание рекурсивных запрос с соответствующего IP-адреса, местный сервер должен обратиться к удаленному с рекурсивным запросом.

Представленные здесь варианты работы системы доменных имен не являются исчерпывающими. За более подробной информацией следует обращаться к RFC-1034 и RFC-1035.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. http://www.microsoft.com/windows2000/en/server/help/sag_DNS_und_HowDnsWorks.htm?id=1945 - Документация Microsoft по Windows 2000 Server. Раздел посвящен принципам работы системы доменных имен. Описывает общие принципы построения DNS и взаимодействия resolver и серверов.
2. http://www.microsoft.com/windows2000/en/server/help/sag_DNS_ovr_ClientFeatures.htm?id=1942 - Документация Microsoft по Windows 2000 Server. Раздел посвящен принципам работы resolver.
3. http://www.nominum.com/resources/documentation/Bv9ARM.pdf - Документация по BIND 9. Справочное руководство системного администратора. Нужно ознакомиться с разделом, посвященном lightweight resolver.
4. http://www.igc.ru/cgi-bin/man-cgi?traceroute - описание команды traceroute, раз уж ее здесь упомянули.
5. http://www.menandmice.com/online_docs_and_faq/glossary/glossarytoc.htm - глоссарий терминов DNS.

3. Типы серверов доменных имен. Master, Slave, Cache, Stealth, Root.

В данном материале разбирается типизация DNS серверов по их назначению, рассматриваются основной функционал серверов каждого типа и его значение для надежной и эффективной работы системы доменных имен.

В данном материале разбирается типизация DNS серверов по их назначению, рассматриваются основной функционал серверов каждого типа и его значение для надежной и эффективной работы системы доменных имен. Прежде чем приступить к чтению данного материала, необходимо ознакомиться с материалом "Как работает система доменных имен".

Сразу оговоримся, что в данном материале речь не идет о серверах, как о программах, которые реализуют функцию сервера доменных имен, например named или сервер доменных имен Windows Server 2000. Мы будем рассматривать серверы, как процессы, которые должны выполнять определенные функции по обслуживанию DNS запросов.

В документах RFC-1034 и RFC-1035 выделяют несколько типов DNS-серверов. В соответствии с типами откликов на запрос к системе доменных имен серверы можно разделить на авторитативные (authoritative) и неавторитативные (non authoritative).

Авторитативный отклик (authoritative response) возвращают серверы, которые являются ответственными за зону, в которой описана информация, необходимая resolver-у (клиент DNS).

Неавторитативный отклик (non authoritative response) возвращают серверы, которые не отвечают за зону, содержащую необходимую resolver информацию.

Это значит, что если наш сервер доменных имен поддерживает домен kuku.ru, и наш resolver настроен таким образом, что посылает рекурсивные запросы к системе доменных имен через наш сервер доменных имен, и при этом мы хотим получить доступ к сайту www.kuku.ru, то мы получим от нашего сервера доменных имен авторитативный отклик, т.к. именно наш сервер отвечает на все запросы к зоне kuku.ru.

Если же мы захотим получить информацию о домене lenta.ru, то в этом случае мы получим неавторитативный отклик с нашего сервера, т.к. он не отвечает за зону lenta.ru и, соответсвенно, либо будет проводить нерекурсивный опрос серверов доменных имен, либо выдаст нам соответствие из своего кэша, т.к. существует большая вероятность того, что кто-то из наших коллег уже обращался с таким запросом к нему.

Авторитативный отклик могут, в свою очередь, вернуть либо master-сервер зоны (primary server), либо slave-сервер (secondary server) зоны. В русскоязычной литературе их называют основным и дублирующим серверами (или первичным и вторичным, соответственно).

Master-сервер (primary, первичный) доменных имен является ответственным (authoritative) за информацию о зоне в том смысле, что читает описание зоны с локального диска компьютера, на котором он функционирует и отвечает в соответствии с этим описанием на запросы resolver-ов. Описание зоны master-сервера является первичным, т.к. его создает вручную администратор зоны. Соответственно, вносить изменения в описание зоны может только администратор данного сервера. Все остальные серверы только

копируют информацию с master-сервера. Вообще говоря, такое определение несколько устарело, и позже будет ясно почему. Но при настройках реальных серверов мы будем использовать именно это определение.

Для зоны можно определить только один master-сервер, т.к. первоисточник может и должен быть только один.

Slave-сервер (secondary, вторичный, дублирующий) также является ответственным (authoritative) за зону. Его основное назначение заключается в том, чтобы подстраховать работу основного сервера доменных имен (master server), ответственного за зону, на случай его выхода из строя, а также для того, чтобы разгрузить основной сервер, приняв часть запросов на себя. Так, например, из 13 серверов, которые обслуживают корневую зону, 12 являются slave-серверами.

Администратор slave-сервера не прописывает данные описания зоны. Он только обеспечивает настройку своего сервера таким образом, чтобы его сервер копировал описание зоны с master-сервера, поддерживая его (описание зоны) в актуальном согласованном с master-сервером состоянии.

Обычно, время согласования описания зоны между slave-сервером и master-сервером задается администратором master-сервера в описании зоны. Slave-сервер в момент своего запуска копирует это описание и затем руководствуется им при обновлении информации о зоне. Slave-сервера периодически через заданный интервал времени опрашивают master-сервер на предмет изменения описания зоны. Если изменения имеют место быть, то описание зоны копируется на slave-сервер.

Спецификация DNS позволяет реализовать и другой механизм обновления информации - оповещение об изменениях. В этом случае инициатива обновления описаний зоны на slave-серверах принадлежит не этим серверам, а master-серверу. Последний оповещает slave-серверы от том, что в базу были внесены изменения, и необходимо эти изменения скопировать на slave-серверы.

Существует оговоренная практика резервирования серверов, которая описана в рекомендациях по ведению зон. Она заключается в том, что для домена второго уровня необходимо иметь как минимум два сервера, ответственных за зону, т.е. дающих авторитативные отклики на запросы. Один master-сервер и один slave-сервер. При этом эти серверы должны иметь независимые подключения к Интернет, чтобы обеспечить бесперебойное обслуживание запросов к зоне в случае потери связи с одним из сегментов сети, в котором находится один из серверов.

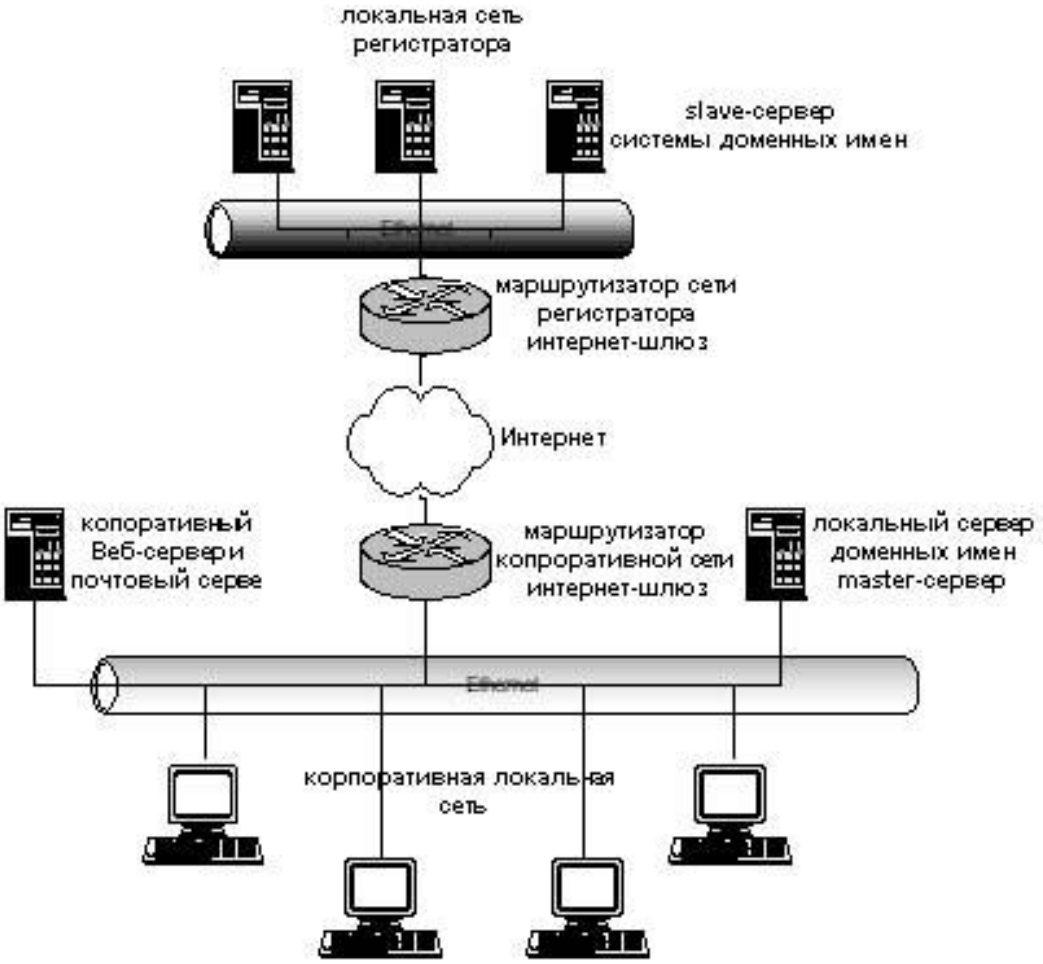

Рис.1. Схема подключения master и slave серверов зоны

Из рисунка 1 видно, что корпоративная локальная сеть и сеть регистратора имеют независимые подключения к Интернет, через разных канальных провайдеров. В частности резервирование серверов в ru-center (www.nic.ru) обеспечивает такую схему подключения, т.е. master-сервер может быть размещен на площадке корпоративной локальной сети, а slave на площадке ru-center.

Размещение обоих серверов на площадке регистратора, например, ru-center, также обеспечивает независимое подключение серверов, т.к. обычно регистратор имеет несколько точек подключения к Интернет.

Как уже было сказано, slave-сервер копирует описание зоны с master-сервера. В настоящее время существует два механизма копирования зоны: полное копирование (AXFR) и инкрементальное (incremental) копирование зоны (IXFR).

В современных RFC, которые расширяют толкование механизмов взаимодействия между участниками обмена данными в рамках DNS, типизацию серверов и их разделение на master-серверы и slave-серверы дают относительно процедур копирования зоны.

Так slave-сервером называют сервер, который использует механизм передачи зоны для получения копии зоны, а master-сервером называют сервер, с которого осуществляется копирование зоны. При этом зону можно скопировать с любого сервера, который является авторитативным. Т.е. зону можно скопировать и со slave-сервера, который относительно самой процедуры копирования зоны будет считаться master-сервером. Для того чтобы выделить сервер, который не копирует зоны ни с какого другого сервера, вводят понятие **Primary Master**. Этот сервер для зоны только один, и он находится в корне процедуры копирования описания зоны.

Типизация серверов относительно процедуры обмена описаниями зон связана с возможностями, которые не описаны в RFC-1034 и RFC-1035, и, соответственно, не поддерживались в ранних версиях программ-серверов доменных имен. Это механизм объявлений об изменениях в описании зоны (RFC-1996), динамическое обновление описания зоны (RFC-2136) и инкрементальное обновление описания зоны (RFC-1995).

Для большинства администраторов корпоративных доменов все эти новшества, возможно, любопытны, но в реальной жизни не слишком полезны, т.к. вся их мощь проявляется только при работе с динамичными описаниями зон, информация в которых подвержена постоянным изменениям. Тем не менее, не сказать о них нельзя, т.к. при работе современными версиями программ-серверов обязательно возникнут вопросы типа "а чтобы это значило".

Сначала о том, что такое уведомления об **изменениях описания зоны (DNS NOTIFY)**. Идея достаточно проста и проистекает из проблемы распространения изменений, внесенных в описание зоны при ее редактировании. Дело в том, что slave-серверы при традиционной работе опрашивают master-сервер (в данном контексте primary master) на предмет изменения в описании зоны через определенные промежутки времени, которые задаются в описании зоны. Возможна ситуация, когда изменения были внесены в зону сразу после ее копирования slave-сервером. В этом случае новая информация появится на всех slave-серверах только при следующем обновлении зоны.

Ситуация усугубляется еще и тем, что остальные не авторитативные серверы держат записи соответствия между именем домена и IP_адресом в своем кэше в течение времени TTL (Time To Live), которое также определяется в описании зоны. Обычно задержка между внесением изменений и стабильной работой с новым описанием зоны в российском сегменте Интернет составляет примерно 2 часа. Вот пример отчета программы nslookup для rambler.ru:

>set type=soa

>rambler.ru

Server: ns.rambler.ru

Address: 217.73.192.49

rambler.ru
origin = ns.rambler.ru
mail addr = bindmaster.rambler-co.ru
serial = 2002090601
refresh = 10800 (3H)
retry = 1800 (30M)
expire = 10800 (3H)
minimum ttl = 3600 (1H)

rambler.ru nameserver = ns.rambler.ru
rambler.ru nameserver = ns.park.rambler.ru
ns.rambler.ru internet address = 217.73.192.49
ns.park.rambler.ru internet address = 217.73.193.23
>

Позиция refresh говорит о том, что время опроса в стандартном режиме составляет 3 часа. А вот из другого отчета для зоны relarn.ru это время и того больше - сутки:

> relarn.ru
Server: ns.relarn.ru
Address: 194.226.65.3

relarn.ru
origin = ns.relarn.ru
mail addr = noc-dns.relarn.ru
serial = 650127367
refresh = 86400 (1D)
retry = 3600 (1H)
expire = 604800 (1W)
minimum ttl = 86400 (1D)
relarn.ru nameserver = ns.relarn.ru
relarn.ru nameserver = ns.ussr.EU.net
relarn.ru nameserver = ns.spb.su
relarn.ru nameserver = ns2.ripn.net
ns.relarn.ru internet address = 194.226.65.3
ns.ussr.EU.net internet address = 193.124.22.65
ns.spb.su internet address = 193.124.83.69
ns2.ripn.net internet address = 194.226.96.30
>

О том, что обозначают другие позиции этих отчетов, мы поговорим тогда, когда будем обсуждать описание зоны (запись SOA), а пока только заметим, что внесение изменений в описание зоны не приводит к появлению возможности мгновенного их использования.

Таким образом, механизм уведомления об изменениях в описании зоны необходим для ускорения введения в жизнь сделанных изменений.

Принцип работы DNS NOTIFY следующий:

1. В базу данных primary master вносятся изменения (руками с последующей перезагрузкой сервера или динамически по протоколу динамического обновления зоны).
2. Primary master оповещает свои slave о том, что произошли изменения, сообщая им номер версии описания зоны.
3. Slave запрашивает с primary master описание зоны и, если номера версии описаний зоны не совпадают (на primary master номер больше), то инициирует процесс обновления описания зоны.
4. Завершив обновление описания зоны, slave посылает оповещения на известные ему авторитативные сервера зоны.

На рисунке пояснется работы приведенной выше схемы:

Рис.2. Схема работы DNS NOTIFY

На рисунке 2 primary master является для обоих slave-серверов master-сервером с точки зрения процедуры передачи зоны. Но в принципе, для одного из серверов он может быть и не определен в качестве master-сервера. В этом случае появляется дополнительное звено в дереве зависимости обмена данными зоны. В этом случае изменения будут передаваться от сервера к серверу так, как это показано на рисунке 3:

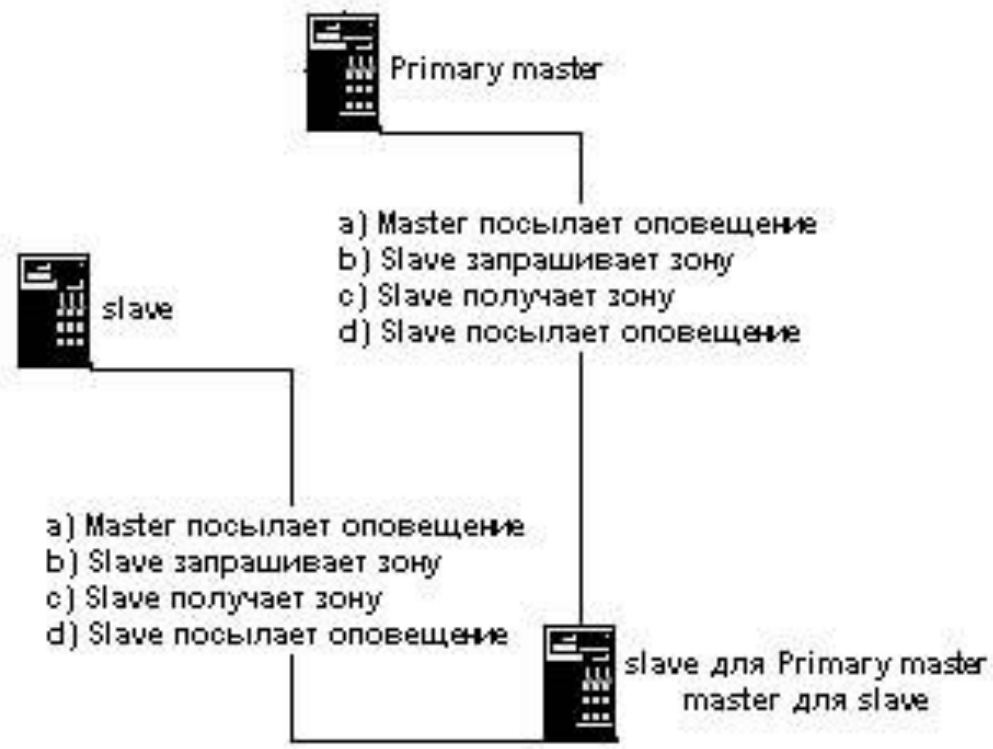

Рис.3. Схема работы DNS NOTIFY при двухзвенном оповещении об изменении описания зоны.

Теперь поясним механизм **динамического обновления описания зоны (Dynamic Updates или DNS UPDATE)**. Изначально описание зоны было, да и до сих пор в большинстве случаев остается, статическим. Это значит, что есть на Primary master файл описания зоны, в который администратор вручную или при помощи скриптов изменения содержания файла вносит изменения. Для того чтобы они стали актуальными, т.е. их увидели resolver-ы, необходима перезагрузка сервера. Идея динамического обновления описания зоны состоит в том, чтобы, во-первых, вносить изменения в описание зоны без перезагрузки сервера, а, во-вторых, делать это удаленно, т.е. администратору не нужно получать доступ к файлам описания зон ни для ручного редактирования, ни для скриптов. Тем самым класс клиентов в модели "клиент-сервер" в рамках DNS обмена расширяется на группу клиентов администрирования.

Когда актуально динамическое обновление зоны? Чаще всего необходимость динамического обновления связывают с работой по протоколу DHCP (Dynamic Host Configuration Protocol, RFC-1541). DHCP Позволяет динамически назначать компьютерам IP-адреса, маски, IP-адреса шлюзов, доменные имена и т.п., т.е. передавать хостам данные без которых невозможна работа в сетях TCP/IP.

DHCP существенно облегчает работу сетевого администратора, которому не нужно настраивать каждый компьютер, подключенный к сети, вручную. Динамическое именование влечет за собой постоянное изменение информации в описании зоны.

Динамическое обновление позволяет авторизованным клиентам посылать запросы на динамическое обновление описания зоны серверам, которые являются авторитативными для соответствующей зоны. Обратите внимание, что запросы могут направляться как master-серверам, т.к. slave-серверам.

Если сервер не является Primary master-сервером зоны, то он отправляет (ретранслирует) запрос на обновление своему master-серверу. Таким образом запрос на обнавление достигает Primary master-сервера зоны.

Как только Primary master-сервер совершит обновление описания зоны, он может по механизму DNS NOTIFY оповестить об этом slave-серверы, которые, в свою очередь, обновят свои описания зоны.

В рамках динамического обновления можно удалять и добавлять отдельные записи описания ресурсов в описания зоны, наборы записей описания ресурсов, выделенных по определенному признаку, скажем записи определенного типа, или записи связанные с определенным доменом. Возможно редактирование описания зоны по условию.

Для того, чтобы успешно выполнялись обмены со slave-серверами, при каждом изменении зоны изменяется и номер ее версии. Это позволяет slave-серверам поддерживать свои описания зоны в актуальном, согласованном с Primary master-сервером состоянии.

Динамическое обновление описания зоны порождает другую проблему - многократную передачу по сети описания зоны между master-серверами и slave-серверами. Если описание зоны большое, то и объем трафика, который передается по сети, будет немаленький.

При этом стоит отметить, что собственно сами изменения описания зоны не столь и велики, т.к., например, при DHCP при каждом изменении добавляется/удаляется одна-две записи описания ресурсов. Каждое такое изменение будет порождать обмен описанием зоны.

При традиционном обмене описанием зоны (AXFR) Между master-сервером и slave-сервером передается полное описание зоны.

Для того, чтобы не передавать всю зону, а передавать только изменения предназначен механизм **инкрементальной передачи описания зоны (IXFR, RFC-1995)**. В рамках обмена передаются номера версий описаний зон и записи, которые нужно добавить или удалить. Принцип простой - сначала идут номер старой версии и список записей, которые нужно удалить, а потом номер более свежей версии и записи, которые нужно добавить.

Мы более подробно остановимся на этом вопросе в разделе, посвященном настройкам BIND и файлам описания зон.

В контексте рассмотрения обмена описаниями зоны между master-серверами и slave-серверами следует упомянуть о **"невидимых" серверах (stealth, см. RFC-1996)**. Суть такого сервера в том, что он не упоминается в описании зоны. Таким образом, его никто не видит, т.к. в рамках DNS-обмена данными информацию о нем получить нельзя ни путем простых запросов, ни путем копирования описания зоны.

Тем не менее, существуют еще файлы статической настройки (конфигурации) серверов доменных имен, где такой сервер может быть прописан. Его можно прописать в качестве master-сервера для slave или сконфигурировать таким образом, что он будет работать в качестве slave для конкретной зоны.

Для чего нужен такой невидимый сервер. Например, для того, чтобы вносить обновления в зону, находясь под защитой firewall. В этом случае Primary master можно сделать

невидимым, а все остальные, в том числе и заявленные при регистрации домена, slave-серверами зоны. Это позволяет нейтрализовать атаки на зону, т.к. обновление всегда будет производиться с "невидимого" Primary master:

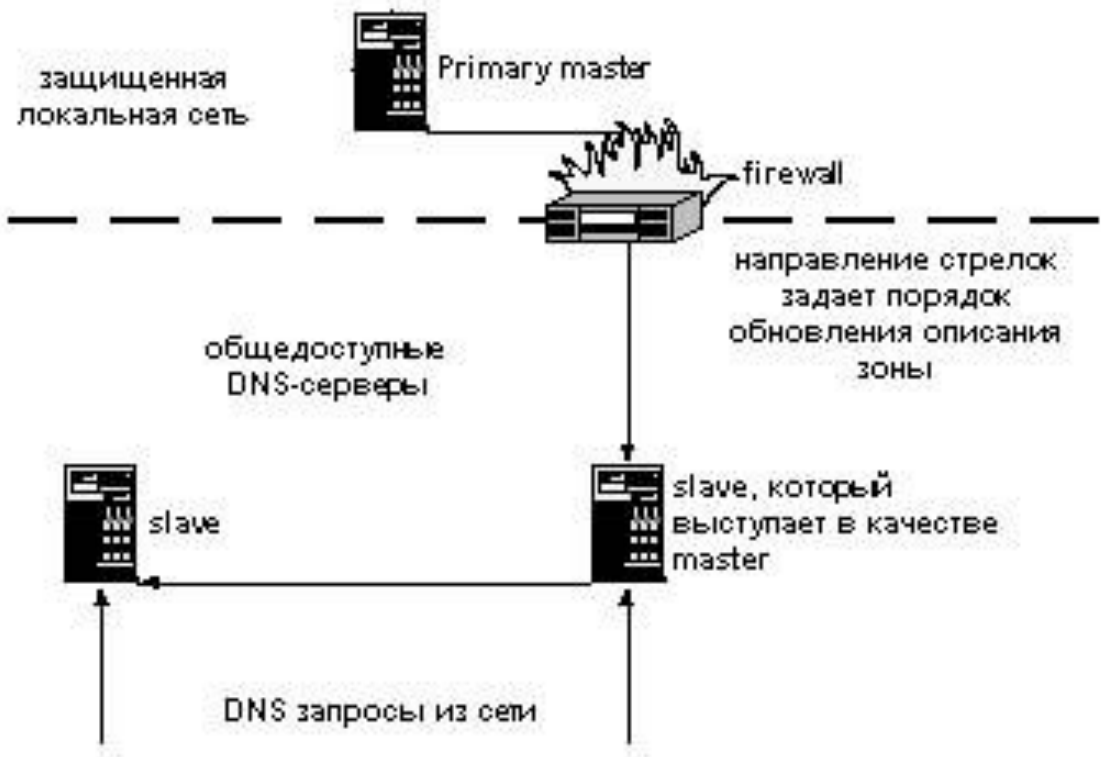

Рис.4. Схема организации DNS-серверов с невидимым primary master сервером

Другая причина создания "невидимых" серверов - разгрузка официально зарегистрированных серверов. В этом случае для обслуживания определенного класса клиентов, которые можно настроить на работу с "невидимым" сервером, создается один или несколько slave-серверов зоны. Они являются авторитативными, но неизвестными широкой интернет-общественности.

Рассмотрим еще один тип серверов, которые выделяют при описании системы доменных имен - **кэширующие (cache) серверы**. Этот тип серверов отличается от тех, что мы обсудили раньше, тем, что сервер данного типа не является авторитативным для какой-либо зоны.

Серверы данного вида используют для организации централизованного кеширования соответствий доменных имен и IP-адресов. Идея организации кэширующего сервера состоит в том, чтобы на искать соответствие доменного имени и IP-адреса в сети, а накапливать их в своем локальном кэше и обслуживать от туда запросы relover-ов.

Кэш-сервер не поддерживает описаний зон и, соответственно, не посылает resolver-ам авторитативных откликов:

> www.w3.org
Server: [144.206.192.60]
Address: 144.206.192.60

Non-authoritative answer:
Name: www.w3.org
Addresses: 18.29.1.35, 18.7.14.127, 18.29.1.34

>

Выше приведен типичный отклик кэширующего сервера на запрос nslookup об адресе www.w3.org. Если бы сервер был авторитативным, то строчки "Non-authoritative answer" в отклике мы бы не увидели.

Мы не обсудили еще один тип серверов - это **сервера, которые обслуживают корневую зону (Root servers)**. Их место в получении отклика на запрос к системе доменных имен ключевое. Именно к одному из корневых серверов обращается локальный сервер доменных имен, если не находит в зоне своей ответственности или в своем кэше соответствия между доменным именем и IP-адресом.

Список этих серверов можно получить достаточно просто. Ниже приведен отчет программы nslookup:

> .
Server: [144.206.192.60]
Address: 144.206.192.60

Non-authoritative answer:
(root)
origin = A.ROOT-SERVERS.NET
mail addr = NSTLD.VERISIGN-GRS.COM
serial = 2002091100
refresh = 1800 (30M)
retry = 900 (15M)
expire = 604800 (1W)
minimum ttl = 86400 (1D)

Authoritative answers can be found from:
(root) nameserver = A.ROOT-SERVERS.NET
(root) nameserver = B.ROOT-SERVERS.NET
(root) nameserver = C.ROOT-SERVERS.NET
(root) nameserver = D.ROOT-SERVERS.NET
(root) nameserver = E.ROOT-SERVERS.NET
(root) nameserver = F.ROOT-SERVERS.NET
(root) nameserver = G.ROOT-SERVERS.NET
(root) nameserver = H.ROOT-SERVERS.NET
(root) nameserver = I.ROOT-SERVERS.NET
(root) nameserver = J.ROOT-SERVERS.NET
(root) nameserver = K.ROOT-SERVERS.NET
(root) nameserver = L.ROOT-SERVERS.NET
(root) nameserver = M.ROOT-SERVERS.NET
A.ROOT-SERVERS.NET internet address = 198.41.0.4
B.ROOT-SERVERS.NET internet address = 128.9.0.107
C.ROOT-SERVERS.NET internet address = 192.33.4.12
D.ROOT-SERVERS.NET internet address = 128.8.10.90
E.ROOT-SERVERS.NET internet address = 192.203.230.10
F.ROOT-SERVERS.NET internet address = 192.5.5.241

G.ROOT-SERVERS.NET internet address = 192.112.36.4
H.ROOT-SERVERS.NET internet address = 128.63.2.53
I.ROOT-SERVERS.NET internet address = 192.36.148.17
J.ROOT-SERVERS.NET internet address = 198.41.0.10
K.ROOT-SERVERS.NET internet address = 193.0.14.129
L.ROOT-SERVERS.NET internet address = 198.32.64.12
M.ROOT-SERVERS.NET internet address = 202.12.27.33
>

Согласно этому отчету Primary master сервером для корневой зоны является сервер A.ROOT-SERVERS.NET. Именно он отвечает за все изменения в описании зоны. Остальные серверы относительно корневой зоны являются slave-серверами. Slave-серверы обновляют свои описания зоны каждые 30 минут. Если в течении недели Primary master будет неработоспособен, то по идее вся система доменных имен потеряет связность. В RFC-2870, где описаны требования к работе root серверов, содержатся общие слова о том, как не допустить такого развития событий, но там нет конкретного плана действий.

На самом деле каждый из 13 серверов - это не одна машина. Так, например, k.root-servers.net (европейский) состоит из k1, k2 и k3, m.root-servers.net (японский) состоит из двух основных серверов и двух серверов горячего резерва, которые подключены к трем независимым точкаv обмена трафиком, f.root-servers.net состоит из двух двухпроцессорных Alpha, по 4GB оперативной памяти в каждой, с балансировкой нагрузки через маршрутизатор Cisco.

И еще несколько слов о root серверах в заключении этого раздела. Существует такая организация - IEPG (Internet Operational Group), задача которой помогать Интернет Сервис Провайдерам взаимодействовать в Сети Интернет. В 1999 году на очередной встрече этой группы обсуждались проблемы DNS И приводилась некоторая статистика по запросам к root серверам:

Средние цифры таковы:

Исходящий трафик: 600 Kbps
Входящий трафик: 2.2 Kbps
Число пакетов за сутки: 26M

Распределение запросов по типам:

Поиск IP-адреса(A) 77%
Поиск сервера доменных имен (NS) 15%
Поиск почтового шлюза(MX) 5%

Статистика обслуживания запросов, которая названа ужасающей(Scary statistics):

- Только 26% правильных(valid) запросов к корневой зоне
- В 6% случаев на правильный запрос нельзя получить авторитативного отклика (.com, .net etc.)
- 66% запросов к несуществующим доменам верхнего уровня (non existent TLDs)

По поводу последней цифры автор BIND Paul Vixie предположил, что это обращения к группам Windows NT, и при отсутствие негативного кэширования в Windows эти запросы повторяются до 10 раз за секунду.

Приведенные цифры должны дать представление о степени нагрузки на root серверы и цену тиражировании программного обеспечения, содержащего неточности прireализации протоколов.

К слову сказать, в 2002 году на встрече IEGP также обсуждались проблемы DNS, а точнее масштаб ошибок при делегировании и описании зон.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. P. Vixie & others. RFC-2136. Dynamic Updates in the Domain Name System (DNS UPDATE). 1997. (http://www.ietf.org/rfc/rfc2136.txt?number=2136)
4. P. Vixie. RFC-1996. A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY). 1996. (http://www.faqs.org/rfcs/rfc1996.html)
5. P. Vixie. RFC-1995. Incremental Zone Transfer in DNS. 1996. (http://www.faqs.org/rfcs/rfc1995.html)
6. R. Bush. RFC-2870. Root Name Server Operational Requirements. 2000. (http://www.ietf.org/rfc/rfc2870.txt)
7. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. R. Droms. Dynamic Host Configuration Protocol. ISI, 1997. (ftp://ftp.isi.edu/in-notes/rfc2131.txt)
2. http://www.wia.org/pub/rootserv.html - схема расположения root servers системы доменных имен.
3. http://m.root-servers.org/ - статистика и краткое описание root серверов m.root-servers.org
4. http://k.root-servers.org/ - статистика и краткое описание root серверов k.root-servers.org
5. http://www.isc.org/iepg/notes-mar99.html - протокол встречи IEPG Оклахоме в марте 1999 года. Довольно любопытная статистика обращений к корневой зоне.

4. BIND (Berkeley Internet Name Domain). Общие сведения

В этом материале речь пойдет о том, что такое BIND, в чем его отличие от DNS, из каких компонентов он состоит, и как его получить и установить.

BIND или Berkeley Internet Name Domain - это пакет программного обеспечения для поддержки DNS, реализованный в университете Беркли. Он широко применяется в Сети. Основная масса серверов DNS - это серверы различных версий BIND.

Иногда название BIND (Berkeley Internet Name Domain) вводит новичков в заблуждение. Им кажется, что речь идет не о программе, а некой альтернативе другой системе доменных имен. Для того чтобы этого избежать, иногда вместо слова "domain" используют слово "daemon", обычно употребляющиеся для обозначения программ-демонов в системах Unix, которые находятся в памяти компьютера и выполняют служебные операции. Для того чтобы потом не повторяться, сразу поясним различия между DNS и BIND.

DNS - это совокупность понятий, архитектуры, спецификаций и протоколов, реализующих эти спецификации. Это изложение и детализация идеи иерархического именования всех хостов в Интернет и правил установки соответствия между именем и IP-адресом хоста.

BIND - это одна из реализаций идеи и спецификаций DNS. Самая популярная, одна из самых совершенных на данный момент, но не единственная. BIND обеспечивает поиск доменных имен и IP адресов для любого узла Сети, взаимодействие с системой электронной почты, поддержку программного обеспечения динамической настройки хостов и т.п. Именно BIND установлен на серверах обеспечивающих поддержку корневой зоны.

Пакет Berkeley Internet Name Domain был первоначально разработан в университете Беркли (Калифорния) в качестве работы на соискания ученой степени доктора философии (graduate student project) по гранту DARPA. Авторами первоначальной версии пакета были Дуглас Терри (Douglas Terry), Марк Пайтер (Mark Painter), Давид Райгл (David Riggle) и Сонгниан Зоу(Songnian Zhou).

В Беркли BIND "прожил" до версии 4.8.3. Версии 4.9 и 4.9.1 были разработаны в DEC, которая волшебным образом превратилась с некоторых пор в COMPAQ. В настоящее время пакет поддерживает Internet Software Consortium.

Существует множество версий пакета. До самого недавнего времени наиболее распространенной была версия 4 и ее модификации. В настоящее время все работы по ней приостановлены. Рекомендуемой к использованию является версия 9, которая существует в своей наиболее свежей ипостаси, как BIND 9.2.1. Часто используются и 8-ые версии пакета.

Автор BIND Paul Vixie рекомендует пользоваться 9-ой версией программы, которая является абсолютно новым продуктом, разработанным для функционирования в агрессивной среде современного Интернета. По его утверждению все версии BIND до 8-ой включительно опирались на код, который написали аспиранты Berkeley. Последние выпуски 8-ой версии исчерпали все возможности по улучшению кода, что и привело к

появлению 9-ой версии программы, которая была полностью переписана профессионалами с применением методов "контрактного" проектирования.

BIND состоит из трех компонентов:

- сервера доменных имен (Domain Name System Server) named;
- библиотеки ПО revolver;
- средств обеспечивающих правильную настройку и работу сервера.

Первые две позиции из этого списка обеспечивают заложенную в DNS архитектуру "клиент-сервер", последняя позиция позволяет упростить настройку сервера и управление его функционированием.

Обычно, модули resolver-а находятся в библиотеке libc.a, если речь идет о системе UNIX, и редактируются вместе с программой, использующей вызовы gethostbyname и gethostbyaddr. BIND работает как со стандартной библиотекой resolver, которая поставляется с операционной системой, так и с собственной библиотекой.

Работа с его собственной библиотекой дает дополнительные преимущества, т.к., во-первых, можно будет работать с "умным" resolver, а, во-вторых, можно будет использовать новые относительно RFC-1034 и RFC-1035 возможности DNS.

BIND адаптирован для большинства современных компьютерных платформ, в том числе и для Windows. Практическая работа с системой доменных имен (администрирование серверов) тесно связана с практикой применения BIND.

Наиболее существенные отличия BIND 9.2.1 (последняя версия на момент написания этого материала) от предыдущих версий:

- Безопасность DNS (DNSSEC, TSIG)
- IPV6
- Улучшения в протоколе DNS (IXFR, DDNS, Notify, EDNSO)
- Более полное соответствие процедурам, описанным в протоколах
- Мультипроцессорная поддержка
- Улучшенная переносимость
- Поддержка подсхем пространства доменных имен

Дистрибутив BIND можно получить с сайта разработчиков - ftp://ftp.isc.org/isc/bind9/9.2.1/bind-9.2.1.tar.gz , либо с одного из официальных зеркал - http://www.isc.org/ISC/MIRRORS.html.

Выше указан дистрибутив пакета, который содержит исходный код. Его нужно предварительно собрать. Для Windows и Linux можно воспользоваться собранным кодом.

Впрочем, собрать BIND несложно. Для этого его нужно распаковать:

```
>tar -zxvf bind-9.2.1.tar.gz
```

После этого в каталоге, где размещен архив BIND, будет создан каталог bind-9.2.1. Следует перейти в этот каталог и выполнить команды:

>./configure

>make

Для установки BIND следует выполнить:

>make install

По умолчанию все будет установлено в /usr/local с соответствующей разбивкой по подкаталогам. Если нужна какая-то нестандартная установка, то лучше посмотреть внимательнее документацию, которая также содержится в дистрибутиве.

Для запуска сервера нужно создать файлы настройки и описания зон, если это необходимо, но об этом мы поговорим в разделе "Настройка сервера. Файлы конфигурации Named".

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind9.html - страничка BIND 9.2.1
2. http://www.linuxsecurity.com/feature_stories/conrad_vixie-1.html - интервью авторов BIND сетевому изданию linuxsecurity.com. Наиболее интересна та часть ответов, которые дает Paul Vixie, автор BIND до версии 8 и руководитель работ над версией 9.
3. http://www.osp.ru/os/1998/06/34.htm - описание принципов контрактного проектирования. Всегда полезно прочитать о том, как правильно делать свою работу :)

5. Resolver. Типовые настройки

В данном материале разбираются настройки и принципы функционирования resolver в различных версиях BIND. Обсуждаются также особенности resolver, поставляемого в дистрибутиве клонов Unix.

Как уже говорилось выше, система разрешения доменных имен IP-адресами и обратная процедура построены по схеме "клиент-сервер". Функции из библиотеки libc.a (или libc.so) выступают в качестве клиентов, а вся их совокупность носит название resolver. Для того, чтобы клиент мог обратиться к серверу, он должен знать следующее:

- Установлен ли вообще сервер доменных имен,
- Если сервер установлен, то где (IP-адрес сервера доменных имен).
- Если сервер установлен, то к какому домену относится машина.

Иногда название BIND (Berkeley Internet Name Domain) вводит новичков в заблуждение. Им кажется, что речь идет не о программе, а некой альтернативе другой системе доменных имен. Для того чтобы этого избежать, иногда вместо слова "domain" используют слово "daemon", обычно употребляющиеся для обозначения программ-демонов в системах Unix, которые находятся в памяти компьютера и выполняют служебные операции. Для того чтобы потом не повторяться, сразу поясним различия между DNS и BIND.

Знать это нужно для того, чтобы правильно формировать запросы к серверу/ам доменных имен и не перегружать систему лишними или некорректными запросами.

Главная задача resolver - принять запрос от прикладного программного обеспечения, провзаимодействовать с серверами DNS и вернуть в зависимости от запроса либо IP-адрес, либо доменное имя.

Очевидно, что можно это сделать двумя разными способами. Во-первых, путем итеративного опроса серверов, а, во-вторых, путем посылки рекурсивного запроса одному из серверов доменных имен, который этот запрос и обслужит.

Рис.1. Типовая схема взаимодействия Resolver с локальным сервером доменных имен.

На практике второй способ является наиболее распространенным. При этом согласно RFC-1034 речь идет о так называемом "усеченном" resolver (stub resolver).

В RFC-1034 указано также, что основной задачей resolver является обеспечение максимально быстрого обслуживания запросов прикладных программ к системе DNS. Основное место здесь отводится кэширующим resolver-ам. Однако, stub resolver таковым не является. Это значит, что каждый раз, когда прикладная программа запрашивает систему DNS resolver посылает запрос локальному серверу доменных имен.

Любопытно, что в стандартном resolver (см. man resolver - http://ddb.kharkov.ua/cgi-bin/bsdi-man?proto=1.1&query=resolver&msection=3&apropos=0, например) предусмотрена возможность итеративного опроса серверов доменных имен, но не предусмотрена возможность кэширования.

В Windows 2000 предусмотрен кэширующий resolver, который запускается в системе в качестве сервиса по умолчанию. Существует возможность остановки и повторного пуска этого сервиса, а также просмотра результатов его работы. Важно отметить, что данный resolver умеет осуществлять "отрицательное" кэширование (negative caching). Это значит что, если на какой-либо запрос поступает отрицательный ответ, например, отсутствие данного доменного имени, то этот ответ запоминается, и в следующий раз прикладная программа получит "отлуп" прямо из кэша resolver, а не после процедуры безуспешного поиска по серверам доменных имен. Естественно, что "отрицательный" ответ хранится в кэше resolver ограниченное время (Windows 2000 TCP/IP. DNS Resolver Cache Service.).

На самом деле, появление нового resolver в Windows 2000 позволило решить некоторые проблемы, которые windows-системы доставляли системе доменных имен (см. "Полезные ссылки" [2]). Впрочем всех проблем новый resolver Microsoft так и не решил.

В рамках пакета BIND также существует кэширующий resolver, который фактически реализует функции кэширующего сервера доменных имен. При этом он реализован как процесс-демон. О его настройках и использовании мы остановимся в конце данного раздела.

Рассуждения о настройке resolver мы будем вести в терминах Unix системы, если речь пойдет о другом операционном окружении, то это будет оговорено отдельно.

Традиционно вся совокупность параметров настройки resolver задается в файле /etc/resolv.conf. Приведем примера этого файла и разберем назначение каждой из команд, которая может быть использована в файле настройки resolver.

```
#
# Resolver config for paul.
#
domain polyn.kiae.su
nameserver 144.206.130.137
```

Строки, начинающиеся с символа "#" - это комментарии. Среди них следует выделить строку "nonameserver". В нашем примере она не влияет на работу системы разрешения доменных имен, но если в сети нет сервера доменных имен, то следует с этой строки снять символ комментария, а прочие команды напротив превратить в комментарии. При отсутствии сервера доменных имен система будет использовать файл /etc/hosts (см. "История и правила именования хостов").

По директиве domain resolver определяет, в каком домене он функционирует. Это локальное доменное имя. Локальным доменным именем называют имя домена, в который входит хост, где выполняется прикладная программа, использующая библиотеку resolver-а. Например, если хост имеет имя host.kuku.ru, то локальным доменным именем будет kuku.ru.

Узанать локальное доменное имя просто - нужно выполнить команду Hostname:

```
> hostname
generate.polyn.kiae.su
>
```

В данном случае локальное доменное имя - polyn.kiae.su.

Обычно, локальным доменным именем, которое указано после директивы domain, расширяются неполные имена хостов. Например, если обратиться к какой-либо машине с компьютера из предыдущего примера только по имени машины:

```
/usr/paul> telnet radleg
```

то система расширит имя radleg именем домена, и будет искать машину с именем radleg.polyn.kiae.su.

Указанный выше пример - это только частный случай процедуры расширения неполного имени, которая используется функциями resolver. В литературе этот процесс называется применением списка поиска.

Различают два алгоритма применения списка поиска. Ниже представлен алгоритм, который использует resolver стандартной библиотеки libc.a. Большинство клонов Unix (различные версии этой операционной системы) поставляются именно с таким алгоритмом. Обычно именно он применялся в 4-ых версиях пакета BIND.

В общем виде он выглядит следующим образом:

- Если в прикладной программе указано имя хоста с точкой на конце, то расширение имени не производится:

  ```
  /usr/paul>ping polyn.
  ```

  В данном случае имя "polyn." завершено точкой.

- Если имя указано без точки, расширение производится по следующим правилам: либо происходит добавление локального домена, либо происходит обращение к файлу синонимов. Вообще говоря, в различных системах это расширение может быть организовано различными способами. Например, HP-UX имя файла синонимов указывается в переменной окружения HOSTALIASES. Пример расширения доменным именем был приведен выше.
- Если в качестве имени указывается составное имя, то производится серия подстановок, при помощи которых пытаются получить IP-адрес хоста. Например, для выполнения команды Ping из ниже приведенного примера:

/usr/paul> ping paul.polyn

Будет выполнена подстановка по следующим правилам: если в качестве домена в resolv.conf указан домен polyn.kiae.su, то будет проверена следующая последовательность имен: paul.polyn; paul.polyn.polyn.kiae.su; paul.polyn.kiae.su. Последнее имя будет разрешено сервером доменных имен. Ниже приведен пример поиска IP-адреса по списку поиска для неполного имени paul.polyn:

> paul.polyn

Server: IRIS.polyn.kiae.su
Address: 144.206.192.10

;; res_nmkquery(QUERY, paul.polyn, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 53781, rcode = NXDOMAIN
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
paul.polyn, type = A, class = IN
AUTHORITY RECORDS:
-> (root)
ttl = 325 (5m25s)
origin = A.ROOT-SERVERS.NET
mail addr = NSTLD.VERISIGN-GRS.COM
serial = 2002091600
refresh = 1800 (30M)
retry = 900 (15M)
expire = 604800 (1W)
minimum ttl = 86400 (1D)

;; res_nmkquery(QUERY, paul.polyn.polyn.kiae.su, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 53782, rcode = NXDOMAIN
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
paul.polyn.polyn.kiae.su, type = A, class = IN
AUTHORITY RECORDS:
-> polyn.kiae.su
ttl = 3600 (1H)
origin = polyn.net.kiae.su
mail addr = paul.kiae.su
serial = 231
refresh = 3600 (1H)
retry = 300 (5M)

expire = 9999999 (16w3d17h46m39s)
minimum ttl = 3600 (1H)

;; res_nmkquery(QUERY, paul.polyn.kiae.su, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 53783, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 1, authority records = 3, additional = 3

QUESTIONS:
paul.polyn.kiae.su, type = A, class = IN
ANSWERS:
-> paul.polyn.kiae.su
internet address = 144.206.192.34
ttl = 3600 (1H)
AUTHORITY RECORDS:
-> polyn.kiae.su
nameserver = polyn.net.kiae.su
ttl = 3600 (1H)
-> polyn.kiae.su
nameserver = ns.spb.su
ttl = 3600 (1H)
-> polyn.kiae.su
nameserver = ns.ussr.eu.net
ttl = 3600 (1H)
ADDITIONAL RECORDS:
-> polyn.net.kiae.su
internet address = 144.206.160.32
ttl = 85775 (23h49m35s)
-> ns.spb.su
internet address = 193.124.83.69
ttl = 159806 (1d20h23m26s)
-> ns.ussr.eu.net
internet address = 193.124.22.65
ttl = 170465 (1d23h21m5s)

Name: paul.polyn.kiae.su
Address: 144.206.192.34

>

В этом примере адрес был успешно найден. А вот пример, в котором адрес не был найден:

> paul.polyn.kiae
Server: IRIS.polyn.kiae.su
Address: 144.206.192.10

;; res_nmkquery(QUERY, paul.polyn.kiae, IN, A)

Got answer:

HEADER:
opcode = QUERY, id = 53784, rcode = NXDOMAIN
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
paul.polyn.kiae, type = A, class = IN
AUTHORITY RECORDS:
-> (root)
ttl = 86400 (1D)
origin = A.ROOT-SERVERS.NET
mail addr = NSTLD.VERISIGN-GRS.COM
serial = 2002091600
refresh = 1800 (30M)
retry = 900 (15M)
expire = 604800 (1W)
minimum ttl = 86400 (1D)

;; res_nmkquery(QUERY, paul.polyn.kiae.polyn.kiae.su, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 53785, rcode = NXDOMAIN
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
paul.polyn.kiae.polyn.kiae.su, type = A, class = IN
AUTHORITY RECORDS:
-> polyn.kiae.su
ttl = 3600 (1H)
origin = polyn.net.kiae.su
mail addr = paul.kiae.su
serial = 231
refresh = 3600 (1H)
retry = 300 (5M)
expire = 9999999 (16w3d17h46m39s)
minimum ttl = 3600 (1H)

;; res_nmkquery(QUERY, paul.polyn.kiae.kiae.su, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 53786, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
paul.polyn.kiae.kiae.su, type = A, class = IN
AUTHORITY RECORDS:
-> kiae.su
ttl = 86400 (1D)

origin = ns.kiae.ru
mail addr = noc-dns.relarn.ru
serial = 650127450
refresh = 28800 (8H)
retry = 3600 (1H)
expire = 604800 (1W)
minimum ttl = 86400 (1D)

Name: paul.polyn.kiae.kiae.su

>

Запрашивался адрес для неполного имени paul.polyn.kiae, в resolv.conf в качестве локального имени домена было указано:

domain polyn.kiae.su

Почему же мы не нашли адреса для имени? Потому, что список поиска остановился на paul.polyn.kiae.kiae.su, т.е. при переборе имен локальное доменное имя можно усечь только до имени состоящего из меток двух доменов. В нашем случае - это kiae.su.

Если ваш resolver ведет себя приведенным выше способом, то следует проверить, что в директиве domain срока заканчивается не пробелом, а отображаемым символом. В противном случае возможны ошибки при разрешении имен.

Усечение до двухзвенного имени вызвано проблемой, которая описана в RFC-1535. Суть ее заключается в том, при полном поиске, т.е. при подстановке и однозвенного имени появлялась реальная возможность "улететь" совсем на другой хост.

Например, если зарегистрировать в com домен ru (ru.com), то для всех пользователей машин из домена com, которые хотят попасть в домен ru, появляется возможность попасть в приватный домен ru.com, т.к. обычно пользователи не указывают на конце символа ".". При этом у администратора домена ru.com появляется возможность по своему усмотрению накручивать, например, сайт www.ru.com.

К слову сказать, ru.com уже зарегистрирован.

Вот отчет whois:

Registrant:
Central Nic Ltd (RU70-DOM)
13 Bowerdean St Fulham
London, SW6 3TU
UK

Domain Name: RU.COM

Administrative Contact, Technical Contact:
Hostmaster (HO4073-ORG) hostmaster@CENTRALNIC.NET
CentralNic Ltd
163 New Kings Road

London
UK
+44 20 7384 3050Fax- +44 20 7736 9253
Fax- - +44 20 7736 9253

Record expires on 23-Jun-2010.
Record created on 23-Jun-2000.
Database last updated on 16-Sep-2002 10:56:25 EDT.

Domain servers in listed order:

NS0.CENTRALNIC.NET 195.82.125.70
NS1.CENTRALNIC.NET 212.18.224.66
LON-NS-2.CENTRALNIC.NET 195.149.39.141
ESS-NS-0.CENTRALNIC.NET 194.221.62.8
NS0.JML.NET 195.149.39.76

Кроме того, www.ru.com тоже уже занят.

Новые (с BIND 4.9) версии resolver, которые входят в состав BIND работают иначе:

- Если введено имя хоста без точек, то оно расширяется локальным доменным именем
- Если пункт 1 не дал результата, то введенное имя используется в качестве имени TLD.
- Если введенное имя содержит составное, но не полное (fully qualified domain name - FQDN), имя, то в этом случае сначала проверяется введенное имя без расширения его локальным доменным именем.
- Если пункт 3 не дал результата, то введенное имя расширяется локальным доменным именем.

Усечений локального доменного имени в этом алгоритме не предусмотрено.

Если по каким-либо причинам стандартный алгоритм применения списка поиска не устраивает, то тогда вместо директивы damin в resolv.conf применяют директиву search:

search corp.ru .

После директивы search через пробел или табуляцию указывают доменные имена, которые будут расширять введенное пользователем имя, если оно не относится к FQDN.

Следует заметить, что размер списка не безграничен. Всего он может содержать 256 символов.

При указании имени домена следует учитывать то, как будет в этом случае работать весь комплекс программного обеспечения, установленный на данном компьютере. Дело в том, что некоторые программы, sendmail, например, способны сами решать проблему определения IP-адресов по именам (то бишь использовать свои собственные функции при обращении к DNS, например, sm_gethostbyname()). В ряде случаев это может привести к противоречиям и ошибкам при функционировании такого сорта программ.

Директива nameserver определяет адрес сервера доменных имен, который будет выполнять рекурсивные запросы resolver. Возможно указание нескольких серверов. Обычно - это не более трех серверов доменных имен.

На самом деле, если собирать библиотеку resolver самостоятельно, то число сервер, которое можно указывать в resolv.conf может быть установлено произвольно Определяется переменной MAXNS в файле /usr/include/resolv.h.

Вообще говоря, многие ограничения resolver заданы переменными в этом файле. Например, максимальное число элементов списка поиска - MAXDNSRCH, минимальный уровень вложенности локального доменного имени - LOCALDOMAINPARTS и т.п..

Если нет указаний на сервер доменных имен, то предполагается, что существует локальный сервер доменных имен. В старых версиях предполагалось, что он размещен по адресу 127.0.01, в новых версиях предполагается, что локальный адрес 0.0.0.0. Изменение адреса связано с проблемами, которые возникали при работе хоста через коммутируемые соединения. Предполагалось, что адрес умолчания - это адрес полученный через ifconfig в момент начальной загрузки машины. В этом случае интерфейс должен быть поднят и активен. Все такого сорта адреса интерфейсов маршрутизировались через 127.0.0.1. Но если интерфейс не поднимался, то возникало "залипание" и машина висла на этапе начальной загрузки. Изменение адреса позволяет решить эту проблему в рамках самого пакета.

Общая рекомендация такова: всегда нужно создавать файл resolv.conf и указывать в нем адрес локального сервера доменных имен.

Если указано несколько серверов доменных имен, то адрес каждого сервера указывается отдельной строкой в файле resolv.conf:

```
# resolv.conf
search corp.ru .
nameserver 192.168.0.1
nameserver 192.168.0.2
nameserver 192.168.0.3
```

Опрос серверов осуществляется по порядку их указания в файле resolv.conf. Сначала будет опрашиваться 192.168.0.1, потом, если первый сервер не ответил, - 192.168.0.2, и последним, если не ответили первые два, - 192.168.0.3. Минимальный интервал времени между попытками по умолчанию (переменная RES_TIMEOUT в файле resolv.conf - 5 секунд). Обычно resolver делает 2 или 3 серии попыток (зависит от версии).

Resolver не упорядочивает сервера из директив nameserver по продолжительности времени отклика от них. Он их всегда опрашивает в том порядке, в котором они указаны в файле resolv.conf. Наиболее целесообразно первым указывать основной сервер доменных имен данного домена, а вторым дублирующий сервер доменных имен данного домена .

Обычно директивами domain, search, nameserer и ограничиваются при описании файла настройки resolv.conf. Более подробную информацию можно найти на страницах руководств системных администраторов операционных систем (ссылка на man Unix приведен в "полезных" ссылках).

И в заключении несколько слов об "умном" resolver пакета BIND 9. Он носит название lwresd и запускается в момент начальной загрузки системы как демон. Прикладные программы для работы с ним должны быть отредактированы с библиотекой вызовов этого resolver (BIND 9 lightweight resolver library).

Lwresd - это практически кеширующий локальный сервер доменных имен, который общается с клиентами не по протоколу DNS, а по своему собственному, посылая в сеть запросы. Если в resolv.conf указан сервер доменных имен, который может обрабатывать рекурсивные запросы, то lwresd будет пересылать ему запросы клиентов, если нет, то он может, используя встроенный список корневых серверов сам выполнять запросы клиентов.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind9.html - страничка BIND 9.2.1
2. http://www.linuxsecurity.com/feature_stories/conrad_vixie-1.html - интервью авторов BIND сетевому изданию linuxsecurity.com. Наиболее интересна та часть ответов, которые дает Paul Vixie, автор BIND до версии 8 и руководитель работ над версией 9.
3. http://www.osp.ru/os/1998/06/34.htm - описание принципов контрактного проектирования. Всегда полезно прочитать о том, как правильно делать свою работу :)

6. Настройка named (BIND/ name server). Кэширующий сервер, slave и master

В этом материале будут разобраны различные версии нотации файла конфигурации программы named. Мы опишем также назначение некоторых директив файлов конфигурации и разберем типовые примеры для кэширующего, master и slave серверов.

В этом материале будут разобраны различные версии нотации файла конфигурации программы named. Мы опишем также назначение некоторых директив файлов конфигурации и разберем типовые примеры для кэширующего, master и slave серверов.

Named - это сервер доменных имен пакета BIND. Named может реализовывать функции серверов любого типа: master, slave, cache (см. материал "Типы серверов доменных имен).

Программа named в момент запуска или перезапуска считывает данные из своего файла конфигурации и файлов описания зон, если они существуют, и таким образом настраивается.

Формат файла конфигурации для 4-ой версии программы отличается от того, который применяется в 8-ой и 9-ой версиях BIND. В силу целого ряда причин имеет смысл рассмотреть оба формата файла конфигурации.

Файл конфигурации BIND 4

Всего существует три типа файлов, которые использует named 4-ой версии. К ним относятся:

- файл начальной загрузки буфера (cache)
- конфигурации named (named.boot)
- файлы описания "прямых" и "обратных" зон

В литературе по named принято начинать с файла описания базы данных named - named.boot. Еще раз напомним, что более свежие версии named настраиваются при помощи файла конфигурации, имеющего другое имя и другой формат директив.

Файл named.boot в 4-ой версии используется программой named для своей настройки и первичной загрузки базы данных домена. Это первое, что ищет named при своем запуске.

Терминология при работе с named из BIND 4 отличается от той, которая принята в настоящее время. Master сервер зоны в данном случае будет называться primary сервером зоны, slave сервер зоны будет называться secondary сервером зоны.

В файле named.boot для описания настроек named используются следующие команды:

- directory - адрес директории в файловой системе компьютера, на котором запускается named.
- primary - определяет зону, для которой данный сервер является primary server, и имя файла базы данных этой зоны. Файл размещается в директории, которая указана в команде directory.

- secondary - определяет зону, для которой данный сервер является secondary server, а также определяет адреса других серверов для этой зоны, обычно primary server, и имя файла где будет вестись копия базы данных этой зоны. Файл размещается в директории, указанной в команде directory.
- cache - определяет имя кэш-файла. Обычно в кэш-файле описаны адреса и имена корневых серверов, которые данный сервер будет использовать для получения адресов удаленных серверов доменных имен.
- forwarders - данная команда определяет адреса серверов доменных имен куда следует отправлять рекурсивные запросы. Естественно, что на этих серверах для хоста, на котором установлен named, должна быть разрешена обработка рекурсивных запросов с этого хоста.
- slave - заставляет сервер отправлять все запросы на серверы, которые определены командой forwarders.

Прежде, чем рассматривать примеры файлов named.boot для различных типов серверов, хотелось бы обратить внимание читателя на две последние команды.

Фактически, именно они и определяют какая из процедур разрешения запроса (рекурсивная или нерекурсивная будут применяться на вашем сервере в каждом конкретном случае). Если в файле named.boot есть команда slave, то определена рекурсивная процедура разрешения запроса, т.к. в этом случае named будет отсылать все запросы на серверы указанные в команде forwarders и от них получать адреса или имена удаленных хостов. В этом случае серверы из forwarders будут опрашивать корневые серверы и удаленные серверы доменов. Если эти серверы также содержат команду slave, то поиск адреса или имени будут осуществлять серверы, которые определены в их файлах named.boot как forwarders.

Когда такая конфигурация сервера полезна? Обычно так настраивают кэширующие серверы подразделений внутри одной организации. В качестве сервера, на который осуществляется пересылка всех запросов, используется центральный сервер организации. При этом с корневыми серверами общается только этот сервер, серверы зон самостоятельно на корневые серверы запросов не отправляют. Если число машин за пределами данного домена, с которыми общаются пользователи этого домена, не очень велико, и все они компактно расположены в других доменах, то центральный сервер домена быстро заполнит свой кэш необходимой для разрешения запросов информацией. При этом такой кэш будет только один, а работать он будет с той же скоростью, как если бы каждый из серверов зон создавал его у себя.

Приведем теперь пример файла named.boot, в котором проиллюстрируем использование указанных выше команд:

```
;An Example of the named.boot
;
directory namedb
primary 0.0.127.in-addr.arpa localhost
primary vega.ru vega
primary 43.226.194.in-addr.arpa vega.rev
secondary polyn.kiae.su 144.206.130.137 144.206.192.34 polyn
```

secondary 160.206.144.in-addr.arpa 144.206.130.137 144.206.192.34 polyn.rev

cache . named.root

Пример 1. Содержание файла named.boot.

Обычно, файл named.boot размещается в директории /etc. От этой директории затем идет отсчет места компонентов базы данных named. В нашем случае эту базу данных можно представить в виде графа (рис.1), у которого в качестве корня выступает директория /etc. В /etc существует поддиректория namedb, в которой располагаются все остальные файлы.

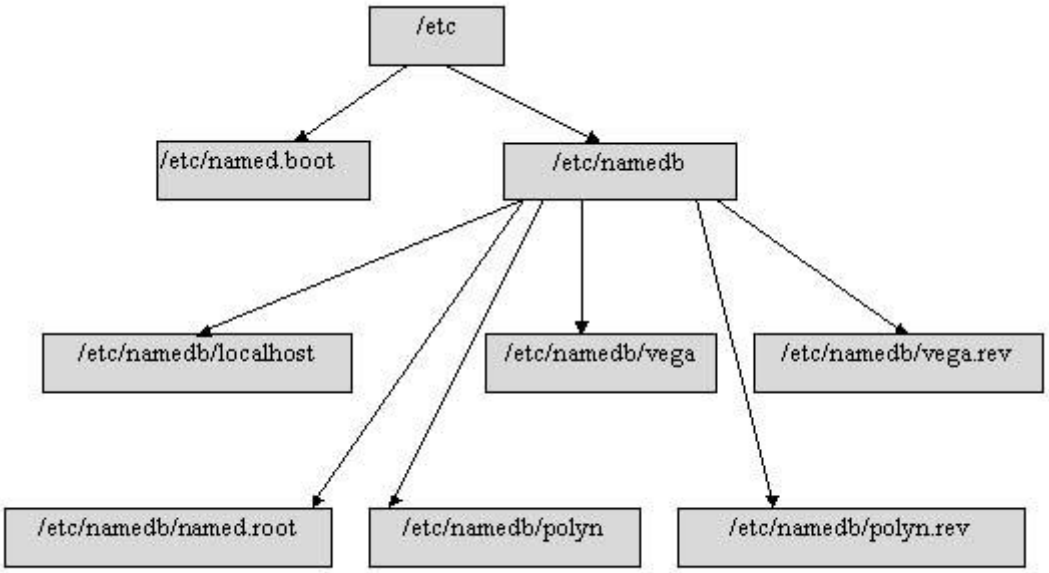

Рис.1. Структура размещения файлов базы данных named из примера 1.

Сопоставив рисунок 1 и описание из примера 1, легко догадаться, что последняя колонка в каждой из команд описания настроек named заканчивается именем соответствующего файла или директории. Рассмотрим формат каждой команды более подробно.

Команда directory определяет директорию namedb как директорию размещения файлов базы данных named (файлов описания зон). Вообще говоря, вовсе не так уж обязательно размещать все файлы в одной директории. Можно создать несколько директорий, например, по количеству зон. Например, для каждой зоны можно использовать отдельную поддиректорию в корневой директории named. Если придерживаться этой логики размещения файлов, то

;An Example of the named.boot
;
directory namedb
primary 0.0.127.in-addr.arpa localhost
primary vega.ru v/vega
primary 43.226.194.in-addr.arpa v/vega.rev
secondary polyn.kiae.su 144.206.130.137 144.206.192.34 p/polyn

secondary 160.206.144.in-addr.arpa 144.206.130.137 144.206.192.34 p/polyn.rev
cache . named.root

Пример 2. Содержание файла named.boot при размещении файлов описания зон для primary и secondary серверов по разным директориям.

В примере 2 в директории namedb определены две поддиректории v и p. В директории v размещаются файлы зон vega.ru и 43.226.194.in-addr.arpa, для которых данный сервер является primary. В свою очередь в директории p размещаются файлы описания зон polyn.kiae.su и 160.206.144.in-addr.arpa, для которых данный сервер является secondary сервером. Если представить структуру файлов в виде графа, то получится граф с рисунка 2.

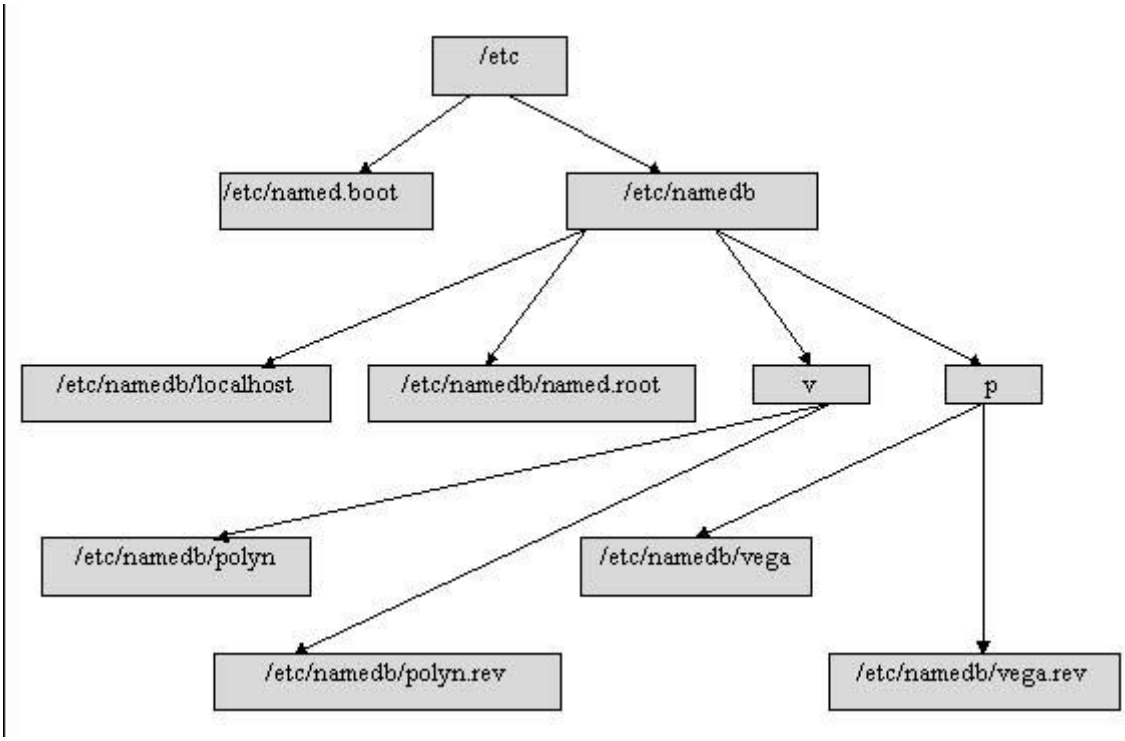

Рис.2. Дерево файлов описания базы данных named из примера 2.

Вообще говоря, корнем описания файлов базы данных named может быть директория отличная от /etc. если программу named запустить с флагом "-f", то в качестве параметра можно указать полный путь к файлу named.boot или к другому файлу, который используется вместо named.boot:

/usr/paul>named -f /usr/paul/named.par

В этом случае в качестве корневой директории для файлов описания базы данных named будет использоваться директория /usr/paul, а в качестве файла named.boot файл named.par из этой директории.

Кроме того, в команде directory, как это определено в наших примерах, используется, так называемый, неполный путь к директории, где расположены файлы базы данных named. Однако можно указывать и полный путь, что дает возможность расположить эти файлы где угодно в файловой системе сервера. Например, если файлы расположены в

директории /usr/local/etc/namedb, то в файле named.boot следует указать следующую команду directory:

directory /usr/local/etc/namedb

Команда primary используется для указания зоны, которую данный сервер обслуживает в качестве master сервера, и указания имени файла, где она (зона) описана. Формат команды primary можно описать следующим образом:

primary <имя зоны> <имя файла описания зоны>

В примерах 2 и 3 используется три команды primary. Первая описывает "обратную" зону 0.0.127.in-addr.arpa, вторая команда описывает "прямую" зону vega.ru и третья команда описывает "обратную" зону 43.226.194.in-addr.arpa. Наличие двух "обратных" зон вызвано тем, что прямая зона определена на двух сетях - 194.226.43.0 и 127.0.0.0.

Сеть 127.0.0.0 - это особая сеть, поэтому первая команда должна быть на любом сервере доменных имен, т.к. она служит для определения обратной зоны 0.0.127.in-addr.arpa, которая закреплена за любой машиной, на которой установлен стек протоколов TCP/IP.

Особое назначение этого домена следует из особого значение IP-адресов, которые закреплены за ним. Они обозначают "петлю", т.е. при отправке пакетов по адресу, например, 127.0.0.1 пакеты не выходят за пределы одного компьютера и таким образом не попадают в реальную сеть. В некоторых реализациях стека определенное значение имеют и другие адреса сети 127, например, 127.0.0.2 и 127.0.0.3 в HP-UX.

Очень часто в примерах описания файла named.boot можно встретить повторение имени зоны в названии файла:

primary 0.0.127.in-addr.arpa 0.0.127.in-addr.arpa

Это повторение означает только то, что в директории файлов описания базы данных named должен быть файл с таким именем. Для тех, кто привык к тому, что в файловой системе MS-DOS были разрешены только трехбуквенные расширения имени файла, подобное имя выглядит довольно странно, но у энтузиастов Unix и пользователей современных версий Windows это не должно вызывать затруднений. Использование имен зон в качестве имен файлов - это общепринятая практика. Так гораздо проще ориентироваться среди файлов описания базы данных named.

Часто вместо 0.0.127.in-addr.arpa указывают 127.in-addr.arpa. Сеть 127 - это сеть класса A (Для тех, кто привык к нотациям CIDR рекомендуем прочитать RFC-790 и 791). Для этой сети что 127, что 127.0, что 127.0.0 - суть одно и тоже. Так как при указании обратной зоны числа в IP-адресах указываются в обратной последовательности, то 0.0.127.in-addr.arpa и 127.in-addr.arpa также означают одно и тоже. Вообще говоря, обработка этой обратной зоны согласно RFC-1035 производится особым способом.

Среди специалистов по named нет единства мнений о стиле определения прямых и обратных зон, в том числе, и о зоны 0.0.127.in-addr.arpa. Некоторые предлагали ввести "прямую" зону, которая бы соответствовала "обратной" зоне 0.0.127.in-addr.arpa. Связано это с доменным именем, которое ставится в соответствие адресу 127.0.0.1.

Команда secondary используется для описания зон, где данный сервер выполняет функции slave сервера, и имеет следующий формат:

secondary <имя зоны> <список IP-адресов серверов зоны> <имя файла описания зоны>

В наших примерах описано две зоны, для которых данный сервер является secondary сервером - polyn.kiae.su и 160.206.144.in-addr.arpa. В примерах для каждой из этих зон указано по два IP-адреса серверов, поддерживающих зону.

Вообще говоря, достаточно указывать адрес только primary сервера зоны. С точки зрения актуальности состояния базы данных зоны, для которой создается secondary сервер, указание одного только primary наиболее правильное решение, т.к. только primary сервер содержит наиболее актуальную базу данных зоны. Однако в ряде случаев, имеет смысл указать несколько серверов, primary и secondary, например. В случае указания нескольких серверов, база копируется с того, который указан первым, если он доступен. Если сервер не доступен, то опрашивается следующий в списке, до первого доступного сервера.

Файл, который указан последним аргументом в команде secondary, создается named при запуске и постоянно обновляется в соответствии с описанием взаимодействия master и slave серверов. Редактировать вручную этот файл не имеет смысла, т.к. это калька с master сервера, и через постоянные промежутки времени этот файл обновляется программой named. Таким образом, если кто-либо и внесет в него изменения, то named, создавая новую копию базы данных зоны, затрет эти изменения.

В отличии от master сервера slave сервер не способен поддерживать разрешение запросов сколь угодно долго. Как только истечет время актуальности данных в его базе данных, он пытается создать новую копию базы с базы данных master сервера. Если это не удается, то сервер через некоторое время перестанет обслуживать зону (см. описание записи описания ресурсов SOA).

Главное назначение slave сервера - это повышение надежности службы доменных имен. Описание зоны named копирует с серверов, указанных в качестве аргумента команды secondary. Там указаны не только primary master сервер, но и другие серверы, которые относительно primary master являются slave серверами.

Зона копируется с того сервера, который доступен. Это значит, что на данном сервере может оказаться копия с другого slave сервера.

Команда cache служит для определения файла с начальными данными для запуска named. Для того, чтобы начать отвечать на запросы named должна знать адреса других серверов доменных имен, к которым можно было бы обратиться с запросами на разрешение IP-адреса по имени или имени по IP-адресу. Формат команды выглядит следующим образом:

cache <имя зоны> <имя файла cache>

Обычно обсуждение cache сводится к обсуждению того, какие корневые серверы, должны быть указаны в файле cache и как поддерживать актуальность этого файла. Прежде, чем обратиться к формату команды, заметим, что не только корневые серверы могут указываться в файле cache, но также и другие серверы, которые часто используются для разрешения запросов в вашем домене.

Согласно Крегу Ричмонду (Craig Richmond) различные версии named по разному используют файл, указанный в команде named. Одни программы загрузив данные из этого файла больше его не используют, другие наоборот постоянно вносят в него изменения.

В случае стабильного файла администратор системы должен сам заботится о его актуальности. Для этого он должен регулярно проверять соответствие между его файлом и файлом, который поддерживается в ns.internic.net. Получить копию файла можно либо, с ftp-сервера ftp.rs.internic.net, либо по команде:

/usr/paul>dig @ns.internic.net.ns > root.cache

Том Ягер (Tom Yager) рекомендует другой источник получения cache - ftp://nic.ddn.mil/netinfo/root-servers.txt.

На самом деле cache - это описание корневой зоны доменных имен, которую, собственно, и обслуживают root серверы. А куда еще прикажите обращаться при запуске named?

Если в файл cache необходимо внести другую информацию, то это делается аналогично описанию корневых серверов. В новых версиях named (8-9ая) нет кэш-файла, но есть описание корневой зоны. По этой причине не стоит вносить в него изменения, которые не сделали на primary master корневой зоны.

Команда forwarders определяет IP-адреса серверов, на которые следует отправлять рекурсивные запросы. Команда имеет следующий вид:

forwarders <список IP-адресов серверов>

Случай организации рекурсивной процедуры разрешения имени с использованием этой команды был рассмотрен ранее. Однако этим случаем не ограничивается круг использования команды forwarders. При регистрации домена некоторое время внешний относительно этого домена мир не подозревает о существовании домена. Должно пройти некоторое время, прежде чем будет закончена процедура регистрации домена и обновления баз данных вышестоящего в иерархии DNS домена на всех серверах как master, так и slave. Однако внутри домена все работает нормально, т.к. локальный сервер запускается до официальной регистрации и способен обслуживать машины домена. Однако, он может и не знать информации о всех доменах Internet. Для этого нужно самому запрашивать root-серверы. Но можно ведь воспользоваться и чужим кэшем. По этой причине всегда полезно указать команду forwarders на сервер домена вышестоящего относительно данного домена. Как правило, на нем разрешено обслуживать рекурсивные запросы хостов из поддоменов.

Обычно указывают не один, а несколько IP-адресов серверов, которые в состоянии ответить на запросы клиентов. Например, для сервера зоны vega.ru, который запущен, но еще не зарегистрирован, можно указать два сервера:

forwarders 144.206.136.1 144.206.130.137

Команда slave указывается тогда, когда сервер общается с внешним миром чрез серверы, указанные в команде forwarders. Параметров у данной команды нет. Файл named.boot для того сервера, если он еще и primary сервер для зон vega.ru и 43.226.194.in-addr.arpa будет выглядеть следующим образом:

```
;An Example of the named.boot
;
directory namedb
primary 0.0.127.in-addr.arpa localhost
primary vega.ru vega
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
forwarders 144.206.130.137 144.206.136.1
slave
```

Пример 3. Подчиненный сервер, работающий по рекурсивной процедуре разрешения запросов от resolver.

Фактически, команда slave позволяет организовать, в некотором смысле, "интеллектуальный" resolver.

Настройка BIND версий 8 и 9.

Named из пакета BIND 8-ой или 9-ой версий в своей настройке и управлении имеет ряд отличий от версии 4. Во-первых, файл конфигурации носит название named.conf и располагается по умолчанию либо в /etc (версия 9), либо в /etc/namedb (версия 8). Во-вторых, у named отсутствует хранимый на диске cache, описание корневой зоны вынесено в отдельный файл, и его нужно прописывать в настройках named. В-третьих, стало возможным дистанционно управлять named.

При запуске программа named читает файл named.conf и таким образом настраивается. Ранее были разобраны формат, структура и содержание файла настройки программы named версий 4.x. Учитывая тот факт, что "четверка" - это уже история, сосредоточимся теперь на файле настройки более свежих версий named.

Основные изменения, которые произошли с named, касаются повышения устойчивости сервера к различного рода атакам, что и отразилось в настройках. Администратор сервера получил возможность управлять копированием зон, обслуживанием запросов на разрешение имен ip-адресами (в данном случае слово "разрешение" употребляется в смысле "установка соответствия", а не в смысле "разрешить что-то делать"). Собственно, любая рекомендация перехода на новые версии named аргументируется более высокой защищенностью программы.

Внешним наиболее заметным отличием файла настройки версий 8.x и 9.x стал иной синтаксис. Он стал C-подобным (если бы новую версию программы начали делать сейчас, а не в 1997 году, то файл настройки получил бы наверное XML-синтаксис J).

Рассматривать все варианты конструкций в файле настройки named.conf, видимо не имеет смысла, а потому сосредоточимся только на наиболее часто встречающихся вариантах и наиболее привлекательных особенностях. При этом многие вопросы, связанные с безопасностью, динамическим обновлением и другими новыми возможностями оставим для отдельного обсуждения.

Прежде, чем приступить к обсуждению named.conf следует обратить внимание на место его расположения в файловой системе. Это важно, т.к. при установке named из портов (ports, т.е. установки двоичных, откомпилированных под определенную платформу версий

named утилитами (скриптами) установки) место расположения этого файла может отличаться от того, которое выбирается по умолчанию, при установке из исходных кодов.

По умолчанию named.conf (установка из исходных кодов) располагается в каталоге /etc/namedb/ (версия 8) или /etc (версия 9). Как любая прикладная программа, named позволяет переопределить место положение своего файла настройки при помощи флага b или c в командной строке при своем запуске:

```
>named -c /etc/named.conf
```

Если Вы работаете с Unix-системой, то нужно внимательно посмотреть файлы конфигурации скриптов начальной загрузки и сами скрипты (обычно что-то типа rc.*, в разных клонах могут быть расположены в различных местах, но корнем пути к которым (местам), обычно, является каталог /etc), чтобы убедиться, что установки умолчания для named не переопределены.

И последнее замечание прежде, чем приступить к описанию примеров. Во всех примерах BIND 8-9 используются адреса, так называемых, немаршрутизируемых сетей. По этой причине будьте внимательны при копировании отдельных фрагментов примеров, если, конечно, Вы сочтете такое копирование полезным J.

Кеширующий сервер (Cache server)

Как уже отмечалось ранее (см. "Типы серверов доменных имен") это сервер, который не отвечает ни за одну из зон, но используется для исполнения запросов resolver-ов. Он выполняет функции локального сервера доменных имен, т.е. выполняет рекурсивные запросы от прикладных программ к системе DNS. При этом в его кэш накапливается информация о соответствиях между доменными именами и IP-адресами, что позволяет существенно повысить скорость обработки запросов и разгрузить другие серверы доменных имен.

В соответствии со своими функциями кэширующий сервер будет иметь всего три файла настройки: файл named.conf, файл с описанием серверов обслуживающих корневую зону и файл описания обратной зоны для зоны 0.0.127.in-addr.arpa.

Формат, структуру и содержание двух последних файлов обсудим позже, а сейчас сосредоточимся на named.conf. Возьмем вариант из руководства по администрированию BIND версии 9:

```
// Two corporate subnets we wish to allow queries from.
acl "corpnets" {192.168.4.0/24; 192.168.7.0/24 };
options {
directory "/etc/namedb"; // Working directory
pid-file "named.pid"; // Put pid file in working
allow-query {"corpnets";};
};
// Root server hints
zone "." {
type hint;
file "root.hint";
};
// Provide a reverse mapping for the loopback address 127.0.0.1
zone "0.0.127.in-addr.arpa" {
```

```
type master;
file "0.0.127.in-addr.arpa";
notify no;
};
```

В данном примере описана настройка кэширующего сервера для двух подсетей. Они перечислены в access control list (директива acl) и названы как "corpnet". Теперь в любом месте файла настройки можно ссылаться на этот список просто как на "corpnet", что, собственно и сделано в директиве options.

В директиве "options" вначале указан рабочий каталог named (опция directory). В нем располагаются все файлы, которые программа использует при своей работе, в том числе и файлы описания зон. Эта опция аналогична команде directory из файла настроек bind версий 4.x.

Затем опция pid-file указывает имя файла в которые будет помещен идентификатор процесса named. Этот файл будет расположен в рабочем каталоге named (т.е. /etc/namedb)

Последняя опция директивы options - allow-query. Она определяет список IP-адресов, для которых разрешено обращаться с запросами к серверу. Другие хосты обслуживаться данным сервером доменных имен не будут. Еще раз обращаем внимание на то, что любые другие хосты с любыми запросами не будут обслуживаться, т.е. не будут обслуживаться как рекурсивные, так и нерекурсивные запросы.

Директива "zone" позволяет описать местоположение и опции для обслуживания зоны. Две зоны обычно всегда описывают. Это зона "." (корневая зона) и обратная зона для адреса 127.0.0.1.

Описание зоны "." (корня дерева доменных имен) необходимо для того, чтобы сервер мог обращаться к "корневым" серверам, с запросами на получение справки о том, где искать "ответственного" за зону, из которой клиент хочет получить информацию (IP-адрес или доменное имя). По этой причине тип зоны определен как "hint", т.е. в буквальном переводе - "справочный", "ссылочный", "наводка на", "намек на", "совет" и т.п..

Само описание "корневых" серверов находится в файле root.hint. Файл поставляется вместе с дистрибутивом, но администратор должен следить за обновлениями этого файла. О том, как это делать мы рассмотрели подробно при описании настроек BIND 4 (cache файл).

Описание обратной зоны для 127.0.0.1 необходимо для того, чтобы можно было локально разрешать (получать соответствие между IP-адресом и именем) при обращениях с реверсивными запросами к зоне 0.0.127.in-addr.arpa. О важности реверсивных запросов будет сказано в разделе посвященном описанию обратной зоны для маршрутизируемой сети (подсети). Здесь мы только укажем, что описать данную зону нужно в целях соблюдения единообразия, отладки и аккуратной работы сервиса DNS.

Для зоны 0.0.127.in-addr.arpa наш сервер будет первичным (primary), поэтому его тип опцией type будет определен как master. В самом деле за эту зону отвечает только наш сервер. Адрес 127.0.0.1 за пределы хоста не маршрутизируется, поэтому у других серверов будет своя зона 0.0.127.in-addr.arpa.

Обратите внимание, что речь идет о кэширующем сервере, а определяем мы его и как master для одной из зон. В данном случае термин "кэширующий" говорит о том, что наш сервер не поддерживает ни одну из реально существующих зон, т.е. ему не делегировано прав на такое обслуживание.

Описание обратной зоны для 0.0.127.in-addr.arpa находится в файле "0.0.127.in-addr.arpa". Вообще говоря, файл может иметь любое имя, которое допускается файловой системой. На самом деле в документации по BIND 9.x для описания зоны используется имя "localhost.rev". Изменить имя в примере было нужно для того, чтобы отразить часто встречающуюся практику именования файлов описания зон именами самих зон. Еще раз подчеркнем, что имя может быть любым допустимым в контексте конкретной файловой системы именем.

Последняя опция "notify" позволяет реализовать режим оповещения об изменениях другие сервера, которые поддерживают данную зону, обычно, вторичные (резервные, secondary, slave). Для нашей обратной зоны это в принципе не нужно, поэтому опция установлена в значение "no". Если же говорить вообще, то не все серверы поддерживают этот режим. Общая практика заключается в том, что обновления "расползаются" по сети в соответствии с параметрами записи SOA из описания зоны.

Официальный (Authoritative) сервер зоны

В руководстве по BIND данная конфигурация обозначена как "Authoritative-only server". Смысл ее заключается в том, что демонстрируется настройка сервера, который обслуживает запросы от любого хоста Сети, но только к той зоне, за которую он официально отвечает. В терминологии BIND 4.x такой сервер именовался как "primary" для зоны, а в терминологии BIND 8.x- 9.x он именуется как "master". Его файл настройки будет выглядеть следующем образом:

```
options {
directory "/etc/namedb"; // Working directory
pid-file "named.pid"; // Put pid file in working
allow-query { any; }; // This is default
recursion no; // Do not provide recursion service
};

zone "." {
type hint;
file "root.hint";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "0.0.127.in-addr.arpa";
notify no;
};

zone "example.com" {
type master;
file "example.com";
allow-transfer { 192.168.4.14; 192.168.5.53; };
};
```

Сначала обратим внимание на отличие в описании опций директивы "options". Во-первых, с запросами к данному серверу позволено обращаться любому хосту Сети, что логично, т.к. никто другой кроме официального сервера в полном объеме за данную зону не отвечает (опция "allow-query"). Есть, конечно, вспомогательные сервера, но они только дублируют master сервер. Вносить изменения в описание зоны можно только на primary master сервере. Именно поэтому при выходе из строя primary master сервера время обслуживания запросов вспомогательными серверами ограничено. Предполагается, что при отказе primary master данные вспомогательных серверов не будут соответствовать исходному описанию зоны, а потому обслуживание запросов лучше прекратить.

Во-вторых, данный сервер не обслуживает запросы рекурсивно. Он только отвечает на запросы к своей зоне (опция "recursion"). Последнее означает, что в отличии от кэширующего сервера, который принимает запросы от клиентов (resolver-ов), опрашивает серверы доменных имен и потом отвечает клиентам, наш сервер запросы клиентов, которые не касаются зоны его ответственности обслуживать не будет.

Описание корневых (root) серверов и обратной зоны для 127.0.0.1 такое же, как и для кэширующего сервера.

При описании зоны ответственности (директива "zone "example.com") в качестве первого параметра указано имя зоны ("example.com") в фигурных скобках определены опции: тип сервера - master, т.е. официальный сервер зоны; файл описания зоны - file "example.com"; список вспомогательных серверов - "allow-transfer { 192.168.4.14; 192.168.5.53 };".

Собственно, опция "allow-transfer" задает список серверов, которым разрешено копировать зону. Официальными вспомогательными серверами они станут только в том случае, если они таковыми были определены в заявке (для доменов второго уровня - корпоративных доменов, например), либо приписаны таковыми при делегировании зоны более глубокого уровня.

Если не установить ограничения на копирование зоны, или указать "any", то любой сервер может скопировать зону, и не только сервер. Из соображений безопасности настоятельно рекомендуется прописывать адреса серверов, которым можно копировать зону.

Вспомогательный сервер (secondary, slave)

В документации по BIND описание master сервера доменных имен и slave сервера совпадают. Но методически правильнее их разнести, что здесь и сделано.

```
options {
directory "/etc/namedb"; // Working directory
pid-file "named.pid"; // Put pid file in working
allow-query { any; }; // This is default
recursion no; // Do not provide recursion service
};

zone "." {
type hint;
file "root.hint";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "0.0.127.in-addr.arpa";
```

```
notify no;
};

zone "eng.example.com" {
type slave;
file "eng.example.com";
masters { 192.168.4.12;};
};
```

То, что мы имеет дело с вспомогательным сервером для зоны "eng.example.com" определено в соответствующей директиве "zone". В качестве типа сервера (type) указано значение "slave", что и определяет сервер как вспомогательный. В опции "masters" определяется список официальных серверов, с которых вспомогательный сервер может списывать зону в файл "eng.example.com". В данном случае указан только один - 192.168.4.12.

Мы рассмотрели типовые настройки файла конфигурации named.Для того, чтобы двигаться дальше, нужно рассмотреть файлы описания зон.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
3. J. Postel . ASSIGNED NUMBERS. RFC 790. (http://www.ietf.org/rfc/rfc0790.txt?number=790)
4. J. Postel .INTERNET PROTOCOL. RFC 791. (http://www.ietf.org/rfc/rfc0791.txt?number=791)
5. P. Mockapetris. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. RFC 1035. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)

Полезные ссылки:

1. http://www.ludd.luth.se/~kavli/BIND-FAQ.html - FAQ первоначально написанные Craig Richmond и в настоящее время поддерживаемые Ronny H. Kavli. Предназеачены главным образом для BIND версии 4.
2. http://www.die.net/doc/linux/man/man5/named.conf.5.html - страницы документации по named.conf для любителей Linux.
3. http://www.gsp.com/cgi-bin/man.cgi?section=5&topic=named.cont - то же самое, что и пункт 2, но только для любителей FreeBSD.

7. Описание зоны. Формат записи описания ресурсов (RR).

В данном материале речь пойдет о формате master файла (файла описания зоны) сервера доменных имен, о записях описания ресурсов и их основных типах.

Основная информация, ради которой, собственно, и затевалась система доменных имен, - это соответствия между IP-адресами и именами. Эта информация содержится в файлах описания зон (master files) ответственности серверов, или просто - описаниях зон.

Еще раз обратим внимание на то, что зона и домен - это не одно и тоже. Домен - это все поддерево дерева доменных имен. Зона - это тот же домен, но за вычетом делегированных другим серверам частей домена, которые в свою очередь будут называться делегированными зонами. Подробнее смотри в материале "Как работает система доменных имен".

Часто информацию о зонах, которые поддерживает сервер доменных имен (т.е. информацию о тех зонах, для которых он является авторитативным) называют информацией из базы данных сервера доменных имен. Не вдаваясь в особенности организации хранения описаний зон в оперативной памяти (ОП) программой named, можно сказать, что вся информация о зонах хранится в файлах описания зон, а при запуске сервера она загружается в ОП. По этой причине базой данных сервера называют всю совокупность файлов описания зон сервера.

Формат записей описания зон определен "по простому" в RFC-1033 и более академично в RFC-1035. К слову сказать, именно поэтому файлы описания зон во всех версиях BIND одинаковые, и если вы решили сменить BIND 4 на BIND 9, то придется переписать только файл конфигурации named, но не файлы описания зон.

Согласно RFC-1035 (секция 5 "Master Files") в файле описания зоны можно использовать следующие директивы:

- [<comment>]
- $ORIGIN [<comment>]
- $INCLUDE [] [<comment>]
- [<comment>]
- [<comment>]

Соответственно, это: комментарий, определение имени текущего домена, вставка внешнего файла и два формата записи описания ресурсов.

Каждая директива занимает ровно одну строку, но если применить круглые скобки "( )", то текст директивы может быть распространен на несколько строк. Любая непустая комбинация пробелов и символов табуляции рассматривается как разделитель между элементами директивы.

В приведенной выше нотации в квадратные скобки ([ ]) заключены необязательные параметры, а в угольные скобки (< >) - сущности. Например, <comment> определяет комментарий. В свою очередь, комментарий - это символ ";" с последующим за ним текстом до конца строки. В следующем фрагменте файла описания зоны:

```
$ORIGIN kyky.ru.
;
; Zone kyky.ru
;
ns IN A 192.168.0.1 ; name server
www IN A 192.168.0.2 ; web server
```

Строки со второй по четвертую включительно будут строками комментария. Пятая и шестая строки заканчиваются комментариями.

Все директивы можно разделить на два типа: директивы управления (control entries) и записи описания ресурсов (resource records - RR). Существует две директивы управления ($ORIGIN и $INCLUDE) и множество RR, большая часть из которых по различным причинам не используется в реальной жизни. Даже тот список записей, что перечислен в RFC-1033, является несколько избыточным.

Вообще-то, директив управления в современных пакетах BIND 8-ой и 9-ой версий больше. К стандартным $ORIGIN и $INCLUDE следует добавить $TTL и $GENERATE.

Формат записи описания ресурса.

Записи типа RR в master файле (файле описания зоны) имеют формат описанный в RFC-1033 и уточненный в RFC-1035. Приведем его по RFC-1033 как более простому, а отличия от RFC-1035 обсудим дополнительно ниже:

```
[] []
```

Как и прежде "[ ]" - это опционный, т.е. необязательный, параметр, а "< >" обозначают сущность. Отличие этой формы определения от RFC-1035 заключается в том, что тоже может быть опционным параметром, т.е. его можно опускать в RR, а также и можно менять местами. Более того, если первые три параметра опущены, то действует значение параметра, определенное в предыдущих записях файла, например:

```
kyky.ru. IN NS kyky.ru.
A 192.168.0.1
MX 10 192.168.0.1
```

В данном случае значение поля name (kyky.ru), значение поля ttl, которое в данном контексте не определено, значение поля class (IN, что означает Интернет) наследуются из записи определения сервера доменных имен(NS RR) адресной записью (A RR) и записью определения почтового посредника (MX RR).

Определим поля записи описания ресурса:

Поле **name** - это имя объекта. Объектом может быть хост, домен и поддомен. Существуют специальные правила именования объектов, которые базируются на понятии текущего имени домена.

Если в качестве значения поля name указан символ "@", то текущим именем домена является имя, указанное в директивах primary, secondary или cache файла named.boot (BIND 4) или zone файла named.conf (BIND 8 или 9), в зависимости от того, о каком файле

базы данных named идет речь. В противном случае для определения текущего имени должна быть указана директива $ORIGIN.

Если имя в записи описания ресурса опущено, то ресурс относится к текущему имени домена. Если имя указано без точки на конце, то оно расширяется текущим именем домена. Для изменения текущего имени домена следует ввести либо команду $ORIGIN, либо указать имя записи ресурса с точкой на конце. Например:

Файл конфигурации named:

```
zone "kyky.ru" in {
type master;
file "kyky.ru";
};

Файл "kyky.ru":

@ IN SOA ns.kyky.ru. user.kuku.ru. (
1 3h 1h 1w 1h )
IN NS ns.kyky.ru.
ns IN A 192.168.0.1
$ORIGIN kuku.ru.
ns IN A 192.168.0.1
www.kyky.ru. IN A 192.168.0.2
```

В выше приведенном пример мы предполагаем, что речь идет о master файле зоны kyky.ru. Именно она у казана в файле конфигурации named (версия BIND 9). Символ "@" в первой позиции первой строки (запись SOA) указывает на то, что текущим именем домена является имя kyky.ru. Для записи описания сервера доменных имен это имя наследуется из записи SOA. Запись описания сервера доменных имен (NS), следовательно относится к домену kyky.ru, т.е. авторитативный сервер доменных для домена kyky.ru называется ns.kyky.ru.

Далее определяется адрес хоста, который имеет имя ns.kuku.ru. При этом вовсе не обязательно указывать имя целиком, т.к. у ns на конце нет символа ".", и, следовательно, данное имя является не полным, и будет расширено до ns.kyky.ru.

Последняя строка определяет адрес для хоста с именем www.kyky.ru, т.к. доменное имя в этом случае заканчивается точкой, то оно не будет расширено значением текущего доменного имени.

Теперь мы переопределяем текущее имя домена при помощи директивы $ORIGIN. Текущее имя домена теперь - kuku.ru. Следовательно, следующая строка будет определять адрес машины ns.kuku.ru.

Если в качестве имени указана одна точка(".") или две точки("..") то такая запись описывает домен root, т.е. корневой домен.

Если в качестве имени указана одна точка(".") или две точки("..") то такая запись описывает домен root, т.е. корневой домен.

Если в имени записи встречается символ "*", то это он означает что вместо него можно подразумевать любую разрешенную последовательность символов. В англоязычных источниках это называют "wildcard character". Для пользователей любой операционной системы употредление этого символа хорошо знакомо по командам dir (MS-DOS) или ls (Unix). Например, при необходимости получить список файлов, оканчивающихся расширением "bak", и только их выдается команда:

```
/usr/paul>ls *.bak
```

Точно также используется этот символ и в имени записи базы данных описания домена. Если в поле имени указан только символ "*", то это предполагает "*.<имя текущего домена>. Если за "звездочкой" следует имя, то оно ограничивает расширение имен "звездочкой". Например, "*.polyn.kiae.su." определяет все имена из домена polyn.kiae.su, в том числе и имена машин в поддоменах, а не только имена машин в корне домена и имена самих поддоменов. Наиболее часто "*" используется в записях MX, которые регулируют обмен электронной почтой между Интернет-почтой и другими почтовыми службами.

Не следует, однако, думать, что символ "*" может быть использован где угодно. Место использования маски строго определено - это может быть только первый символ в поле имени текущего домена, отделенный от остальных символом ".".

Если в поле имени указан только символ "*", то он маскирует доменные имена хостов текущего домена. Однако, если для какого либо хоста существуют RR записи, то маска на этот хост не распространяется:

```
*.kyky.ru. IN MX 10 mail.kyky.ru.
first IN A 192.168.0.1
sub.kyky.ru. IN NS second.kyky.ru.
```

В приведенном выше примере, для всех хостов из зоны kyky.ru в качестве почтового посредника должен использоваться хост mail.kyky.ru. Однако, для first.kyky.ru это не так - в описании зоны есть запись описания ресурса для first.kyky.ru.

Кроме того, для делегированной зоны sub.kyky.ru хост mail.kyky.ru тоже не является почтовым посредником, т.к. для делегированных зон согласно RFC-1034 "делегирование прерывает действие маски".

При назначении имен следует иметь в виду, что не любой символ может быть использован в доменном имени. При этом существует некоторое рассогласование в документах регламентирующих DNS по поводу того, какие символы можно использовать в доменном имени, а какие использовать нельзя.

Так, например, в RFC-1033 прямо указано, что в имени можно использовать любые восьмибитовые отображаемые символы. При этом, правда сделана ссылка на то, что в других протоколах существуют ограничения, а потому рекомендованы символы "A-Z", "a-z", "0-9", а также символы "-" и "_".

Однако в RFC-1034 принято более жесткое определение того, что такое доменное имя. Из разрешенного списка исключили символ подчеркивания и постановили, что система не различает заглавные и прописные буквы, т.е. имена kuku, KUKU, KuKu - это одно и тоже имя.

Любопытно, что не во всех версиях BIND проверяется правильность доменного имени, а в последних версиях можно управлять проверкой правильности. Но уповать на то, что любые отображаемые символы будут правильно распознаны системой DNS, не стоит.

Кроме того, проблема именования доменов в настоящее время упирается в именование доменов не только символами английского алфавита. Но это уже совсем другая история.

Поле **ttl (Time To Live)** - это поле определяет время, в секундах, в течение которого данная запись сохраняется в кэше. Максимальное значение поля TTL 4294967295. Ровно такое максимальное положительное число может быть задано при помощи 32 битов (RFC-1035, секция 6.1.3).

В реальной жизни все гораздо менее масштабно. Записи обычно хранятся в кэше часа или около того, что составляет 3600 секунд. Правда для записей корневой зоны TTL умолчания составляет 1 день или 86400 секунд (см. распечатку корневых серверов в материале "Типы серверов доменных имен. Master, Slave, Cache, Stealth, Root.".

Если это поле оставлено пустым, то по умолчанию принемается значение, указанное в параметре minimum поля данных (data) записи SOA для данной зоны(см. описание записи SOA).

Написав предыдущий абзац, мы остались в эпохе до 1998 года, когда вышло RFC-2308 (Отрицательное кэширование DNS запросов). Тем не менее, не стоит его вымарывать их своей памяти, т.к. до версии 8 BIND минимальное время TTL определял согласно RFC-1035, т.е. в записи SOA. Точнее говоря, это было не минимальное время хранения в кэше, а время умолчания.

RFC-2308 ввел два новых понятия: время негативного кэширования и директиву управления $TTL.

Негативное кэширование - это кэширование запросов, которые доставляют информацию о том, что установить соответствие между доменным именем и IP-адресом нельзя по разным причинам.

Если не использовать негативного кэширования, то локальный сервер доменных имен, который обслуживает рекурсивный запрос resolver, либо сам resolver, если он "умный", будут совершать от 2 до 3 попыток разрешить (установить соответствие между доменным именем и IP-адресом) доменное имя, как только какая-либо из прикладных программ, данное имя запросит. При чем повторяться это будет каждый раз, как только новый запрос поступит, т.к. система не помнит результатов предыдущего выполнения запроса.

Согласно новым правилам, теперь время кеширования по умолчанию для запросов к системе доменных имен, на которые возможно получить позитивный отклик (можно установить соответствие), устанавливается в директиве управления $TTL, а время негативного кэширования устанавливается последним параметром записи SOA.

Если в поле TTL указано 0 значение, то тем самым запрещают кэширование такой записи. Например, запись SOA всегда передают со значением 0 в поле TTL, для того, чтобы запретить кэширование.

Поле **class** определяет класс записи описания ресурса. В Internet используется только один класс записей - класс IN. В принципе существуют еще классы HS(Hessiod) и CH(Chaos),

но в рамках нашего контекста - системы доменных имен Internet - они не рассматриваются.

Некоторые авторы, например, Ronny H.Kavli, который написал комментарии к FAQ Kaig Richmond-a, считают, что наличие этого поля только "замутняет чистый лик" базы данных описания ресурсов DNS. В любом случае все записи из базы данных named имеют вид:

IN

Поле type определяет тип записи описания ресурсов. К таким типам относятся: SOA (Start Of Authority), NS(Name Server), A(Address), MX(Mail eXchanger) CNAME(Canonical NAME), WKS(Well Known Services), PTR(PoinTeR), HINFO(Host INFOrmation) и др.

Перечисленные выше типы записей составляют набор стандартных записей, которые могут встретиться в базе данных описания домена. Кроме этих типов существуют еще и экспериментальные типы.

Следует заметить, что не все стандартные записи используются в базах данных описания доменов. Так администраторы редко пользуются записями типа HINFO и WKS. Некоторые администраторы считают, что вместо CNAME лучше назначить еще один адрес через запись типа A, существуют и другие предпочтения.

В поле data указываются данные для каждой записи описания ресурсов. Для различных типов записей формат данных также различный и соответствует назначению записи описания ресурсов.

При описании элементов записи описания ресурса можно применять маскирование символов. Маскирование символов применяется в том, случае, если необходимо использовать символ, который имеет особое значение для записей описания ресурсов. Например, символ "@". Для этого используется символ обратного слэша "". Аналогично языку программирования C символ можно указывать и цифрами. Только в этом случае используется десятичное число, которое соответствует коду ASCII, например, "40" также обозначает символ "@".

Настоятельно рекомендуем не использовать маскирования символов в файлах описания зон.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
3. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
4. M. Lottor. RFC-1033. DOMAIN OPERATIONS GUIDE November 1987. (http://www.ietf.org/rfc/rfc1033.txt?number=1033)
5. B. Manning & R. Colella. RFC 1637. DNS NSAP Resource Records. 1994. (http://rfc.sunsite.dk/rfc/rfc1637.html)
6. Everhart, Mamakos, Ullmann & Mockapetris. RFC 1183. New DNS RR Definitions. October 1990. (http://www.ietf.org/rfc/rfc1183.txt?number=1183)

7. M. Andrews. RFC 2308. Negative Caching of DNS Queries (DNS NCACHE). 1998.(http://www.ietf.org/rfc/rfc2308.txt?number=2308)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind9.html - страничка BIND 9.2.1
2. http://www.ludd.luth.se/~kavli/BIND-FAQ.html - DNS FAQ. Ответы на большинство вопросов, начинающих и "продвинутых" администраторов. Есть только одно большое НО! Этот материал посвящен BIND версии 4. Но все, что касается описания зоны вполне подходит и для более поздних версий BIND.

7.a. Запись "Start Of Authority"

В данном материале мы обсудим одну из самых важных записей описания ресурсов "Start of Authority"(SOA). Именно эта запись определяет ту зону, к которой относятся все описания ресурсов - прочие Resource Records.

В данном материале мы обсудим одну из самых важных записей описания ресурсов "Start of Authority"(SOA). Именно эта запись определяет ту зону, к которой относятся все описания ресурсов - прочие Resource Records.

Описание зоны по традиции начинают с записи SOA (Start Of Authority). На самом деле, после 1998 года, когда появился документ RFC 2308, описание зоны начинают с директивы управления $TTL, которая задает время хранения соответствий в кэше resolver или сервера доменных имен.

Запись SOA отмечает начало описания зоны. Обычно, это первая запись описания ресурсов (Resource Record - RR). Настоятельно рекомендуется наличие одной и только одной записи типа SOA в каждом файле описания зоны, который указан в записи primary файла named.boot или в директиве zone файла named.conf.

Формат записи SOA можно представить как:

[zone] [ttl] IN SOA origin contact (serial refresh retry expire minimum)

В этой записи каждое из полей обозначает следующее:

Поле **zone** - имя зоны. Если речь идет о зоне, описанной в записи primary файла named.boot, то в качестве имени употребляется символ коммерческого эт - "@". В этом случае в качестве имени текущей зоны берется имя, указанное в качестве первого аргумента команды primary из файла named.boot, или имя зоны, указанное в директиве zone файла named.conf.

Поле зоны обязательно должно быть указано, иначе named не сможет привязать следующие за данной записью описания ресурсов к имени зоны. Пустое имя зоны не является допустимым.

Можно, конечно, указывать и полное имя зоны, не забывая при этом ставить на конце имени ".":

kyky.ru. IN SOA ns.kyky.ru adm.kyky.ru (2 8h 30m 2w 2h)

Поле **ttl** в записи SOA всегда пустое. Дело в том, что время кэширования для записей описания зоны задается либо последним аргументом данных записи SOA (версии BIND до 8.2.), либо директивой управления $TTL. Запрет на кэширование SOA определен в RFC 1035.

Вообще говоря, данное жесткое ограничение (наличие 0 в поле ttl) было снято в 1997 году (RFC 2181). Связано это было с тем, что реально требование наличия нуля в поле ttl записи SOA нигде не использовалось и не проверялось. С тех пор записи SOA могут содержать значения в поле ttl.

Поле **origin** - это доменное имя primary master сервера зоны. В случае описания зоны kyky.ru в качестве сервера используется машина ns.kyky.ru. Данное доменное имя и должно быть указано в поле origin.

Очень часто в этом поле можно встретить имена, которые начинаются с "ns", например, ns.kiae.su или ns.relarn.ru. В данном случае это означает только то, что primary master сервер зоны размещен на машине с таким именем. Никого специального зарезервированного имени для указания в поле origin нет. Использование, указанных выше имен обосновано тем, что их просто легче запомнить, т.к. "ns" означает "name server".

Довольно часто администраторы зон, которые делегируют части своих зон другим организациям, в ультимативной форме требуют, чтобы имя primary master не совпадало с именем зоны.

Elz и Bush в RFC-2181 отмечали, что это требование повсеместно нарушается и практически является бесполезным. Кроме того, существует документ (ripe-203), в котором написано, что данное требование (отличие имени primary master сервера от имени зоны в SOA) справедливо, за исключением случая, когда доменное имя зоны связано адресной записью с IP-адресом primary master этой зоны. Для небольших зон это случается сплошь и рядом, т.к. и primary master зоны и почтовый транспортный агент и прочие сервисы в мелких организациях устанавливаются на одной и той же машине.

Требование, однако, справедливо при заполнении интерактивных форм регистрации домена, т.к. система в момент регистрации не имеет ни малейшего понятия о том, что администратор зоны напишет в файле описания зоны, т.е. гарантии того, что имя зоны и имя primary master сервера совпадают, в момент регистрации домена нет.

Поле **contact** определяет почтовый адрес лица, осуществляющего администрирование зоны. Данный адрес должен совпадать со значением адреса указанным в заявке на домен. Есть, однако, одна особенность при указании этого адреса. Так как символ "@" имеет особый смысл при описании зоны, то вместо этого символа в почтовом адресе используется символ ".".

Например, если ваш администратор домена имеет почтовый адрес adm@kyky.ru, то в поле contact следует писать не adm@kyky.ru, а adm.kyky.ru.

Если в имени пользователя есть какие-либо особые символы, имеющие специальные значения при описании зоны, то они должны маскироваться символом обратного слэша - "\".

Типичный пример - почтовый адрес типа user.name@kuku.ru. В этом случае точку следует замаскировать и написать в поле contact: user.name.kuku.ru. Поступая таким образом, мы исключили особое значение точки как разделителя поддоменов и обеспечили интерпретацию имени как единой символьной строки.

Вообще говоря, почтовые адреса приведенного выше типа не являются редкостью, но у администраторов компьютерных систем встречаются очень редко. Кроме того, если следовать рекомендациям (RFC 2142), то лучше всего, чтобы администратор DNS сервера имел адрес hostmaster@kuku.ru.

Поле данных в записи SOA разбито на аргументы, которые определяют порядок работы сервера с записями описания зоны. Как правило, все аргументы располагают на другой строке или, для лучшего отображения, каждый на своей строке, что заставляет записывать их внутри скобок. Напомним, что скобки позволяют продлить запись описания ресурса на несколько строк, а символы табуляции и пробелы разделяют поля записи описания ресурса между собой.

Атрибут **serial** - определяет серийный номер файла зоны. Если говорить проще, то в этом поле ведется учет изменений файла описания зоны. Serial - это 32-битное целое число, и ограничение по числу цифр, которое можно встретить в руководствах по BIND, на самом деле условно.

В принципе это могут быть любые числа, но чаще всего администраторы используют в качестве серийного номера год (4 позиции), месяц (две позиции), день (две позиции) и версию внесения изменений в файл описания зоны (две позиции). Таким образом эта нотация будет выглядеть как:

ГГГГММДДВВ

Итого получается 10 символов. В старых руководствах по BIND указывают максимальное значение длины этого поля 8 символов.

Важность серийного номера определяется тем, что когда вторичный(secondary) сервер обращается к первичному(primary) серверу для обновления информации о зоне, то он сравнивает серийный номер из своего кэша с серийным номером из базы данных первичного сервера. Если серийный номер из primary сервер больше, то secondary сервер обращается к primary и копирует описание всей зоны целиком, если нет, то он не вносит изменений в свою базу данных. Таким образом, когда производятся изменения в базе данных primary сервера, то значение атрибута serial в поле данных записи SOA для зоны, описание которой было изменено, должно быть увеличено. Неизменение номера - это типичная ошибка, которую совершают администраторы зон. По этой причине лучше пользоваться средствами автоматического обновления зоны.

А что делать при достижении максимально возможного порядкового номера? Вопрос гипотетический, т.к., если даже использовать в качестве серийного номера приведенную выше нотацию, то мы достигнем предельных значений серийного номера только в 4294 году (цитируем по man named для FreeBSD 4.2 -RELEASE), но все же?

На самом деле все просто: нужно начинать сначала. До BIND 9 можно было просто указать 0-ую версию описания зоны в любой момент, и потом снова начать ее увеличивать.

Любителям математики следует прочитать раздел "Цикл порядкового номера" в книге Пола Альбитца и Крикета Ли "DNS и BIND" на странице 194 (Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.). Там подробно объяснена концепция непрерывного арифметического пространства и способ перехода к младшим целым номерам от старших за два шага изменения серийного номера описания зоны. Кроме того, арифметике серийного номера посвящен отдельный документ - RFC 1982.

Атрибут **refresh** определяет интервал времени, после которого slave сервер обязан обратиться к master серверу с запросом на верификацию своего описания зоны. Как уже было сказано в разделе "Типы серверов доменных имен", при этом проверяется серийный номер описания зоны.

Описание зоны загружается slave сервером каждый раз, когда он стартует или перегружается. Однако, при стабильной работе может пройти достаточно большой интервал времени, пока эти события не произойдут в действительности. Кроме того, большинство систем, которые поддерживают сервис доменных имен, работают круглосуточно. Следовательно, необходим механизм синхронизации баз данных master и slave серверов.

Поле **refresh** задает интервал, после которого эта синхронизация автоматически выполняется. Длительность интервала указывается в секундах. В поле refresh можно указать любое целое 32-битное число. Указание маленького значения приведет к неоправданной загрузке сети, ведь в большинстве случаев описание зон довольно стабильно, но иногда, когда зона только создается, можно указать и небольшой интервал.

Атрибут **retry** начинает играть роль тогда, когда master сервер по какой-либо причине не способен удовлетворить запрос slave сервера за время определенное атрибутом refresh. А если говорить точнее, то в момент наступления времени синхронизации описания зоны master сервер по какой-либо причине не отвечает на запросы slave сервера.

Атрибут retry определяет интервал времени, после которого slave сервер должен повторить попытку синхронизовать описание зоны с master сервером. Ограничения на значение этого атрибута те же, что и для атрибута refresh.

При установке этого значения во внимание следует принимать несколько факторов. Во-первых, если master сервер не отвечает, то, скорее всего, произошло что-то серьезное (отделочники вырезали часть сети, т.к. мешала красить стены, экскаватор перекопал магистраль или отключили питание в сети). Такая причина не может быть быстро устранена, поэтому установка слишком малого времени опроса просто зря нагружает сеть.

Во-вторых, если на master сервере прописано много зон, и он обслуживает большое количество запросов, то сервер может просто не успеть ответить на запрос, а очень частые запросы от slave сервера просто "подливают масло в огонь", ухудшая и без того медленную работу сервера.

Атрибут **expire** определяет интервал времени, после которого slave должен прекратить обслуживание запросов к зоне, если он не смог в течение этого времени верифицировать описание зоны, используя информацию с master сервера.

Обычно это интервал делают достаточно большим. Если сделать маленький интервал, то какой смысл в slave сервере? Он просто скоропостижно скончается после отключения master сервера.

В течение всего этого времени, т.е. до достижения конца интервала expire, slave сервер будет пытаться установить контакт с master сервером и обслуживать запросы к зоне.

Правда все эти соображения хороши для случая, когда просто отключится или ломается машина, на которой стоит master сервер. Если же отключится вся сеть, которая описана в зоне, а для большинства организаций так оно и есть, то slave сервер становится подобным космическому аппарату "Пионер", который путешествует во Вселенной, давно потеряв всякую связь с Землей.

Тем не менее, существует несколько причин, по которым его наличие полезно. Во-первых, различные приложения по разному реагируют на ситуацию отсутствия соединения или невозможности найти соответствие доменному имени. Во-вторых, если зона доступна, а хост в ней недоступен, то приложение гораздо быстрее прерывает свою работу. В-третьих, время положительно кэширования гораздо больше времени негативного кэширования, а это ускоряет процесс прекращения попыток соединения с неработающим хостом, т.к. не нужно находить его адрес. В-четвертых, в отключенной сети все хосты не доступны, а они принадлежат, как правило, к одному домену, а адрес его slave сервера доменных имен закэширован и сам сервер доступен, что ускоряет, по крайней мере не замедляет, проверку доступности хостов домена (подробнее см. RFC 2182).

Последний атрибут из поля данных записи SOA - minimum. В разных версиях BIND он определяет совершенно разные понятия. До 8.2 этот параметр определял время кэширования по умолчанию ответов на запросы к DNS. Еще раньше он определял время кэширования по умолчанию только положительных ответов, т.е. тех, которые устанавливали наличие соответствия между IP-адресом и доменным именем. Теперь этот параметр обозначает время негативного кэширования (negative caching), т.е. время кэширования ответов, которые утверждают, что установить соответствие между доменным именем и IP-адресом нельзя.

Последнее обстоятельство заставило ввести в описание зоны новую директиву управления $TTL, которая и определяет время кэширования по умолчанию положительных ответов от системы DNS.

Существуют рекомендации по установки параметров записи SOA. Например, RFC 1537 рекомендует следующие установки для серверов, отличных от тех, которые поддерживают домены верхнего уровня:

28800 ; Refresh 8 hours
7200 ; Retry 2 hours
604800; Expire 7 days
86400 ; Minimum TTL 1 day

На самом деле установить в современных серверах minimum больше 3 часов нельзя (в том смысле, что написать-то можно, работать параметр не будет). Это максимальное время негативного кэширования, согласно документации на BIND 9.

Для TLD в том же документе рекомендованы:

86400 ; Refresh 24 hours
7200 ; Retry 2 hours
2592000; Expire 30 days
345600 ; Minimum TTL 4 days

Конечно, это не догма. В этом легко убедиться, посмотрев соответствующие параметры в SOA для зоны ru программой nslookup:

```
> set type=soa
> ru.
Server: ns.ripn.net
Address: 194.85.119.1

ru
origin = ns.ripn.net
mail addr = hostmaster.ripn.net
serial = 4005176
refresh = 7200 (2H)
retry = 900 (15M)
expire = 2592000 (4w2d)
minimum ttl = 345600 (4D)
ru nameserver = ns.ripn.net
ru nameserver = ns1.relcom.ru
ru nameserver = ns.uu.net
ru nameserver = sunic.sunet.se
ru nameserver = ns2.nic.fr
ru nameserver = ns2.ripn.net
ns.ripn.net internet address = 194.85.119.1
ns1.relcom.ru internet address = 193.125.152.3
ns2.ripn.net internet address = 194.226.96.30
>
```

Для зоны com информация несколько иная:

```
> com.
Server: [192.5.6.30]
Address: 192.5.6.30

com
origin = A.GTLD-SERVERS.NET
mail addr = NSTLD.VERISIGN-GRS.com
serial = 2002092401
refresh = 1800 (30M)
retry = 900 (15M)
expire = 604800 (1W)
minimum ttl = 86400 (1D)
```

Для полноты картины укажем и на более поздние рекомендации. Так, например, для небольших стабильных зон в 1999 году (ripe-203) рекомендовались следующие параметры SOA:

```
example.com. 3600 SOA dns.example.com. hostmaster.example.com. (
1999022301 ; serial YYYYMMDDnn
86400 ; refresh ( 24 hours)
7200 ; retry ( 2 hours)
3600000 ; expire (1000 hours)
172800 ) ; minimum ( 2 days)
```

При этом автор (Peter Koch) подробно описывает причины, по которым установлены эти значения.

Например, для refresh интервал в сутки выставлен потому, что существует динамическое обновление зоны и оповещение slave серверов о произошедших изменениях. По этой причине нет смысла ставить маленький интервал обновления базы данных slave сервера. Если оповещение отключено или используются старые версии BIND, то, видимо, нужно спуститься с небес на землю и поставить интервал поменьше, скажем часов 8 (RFC 1537) или еще меньше.

Для атрибута expire обычно указывают одну или две недели. Однако считается, что за это время серьезные проблемы с primary master не решить, следовательно, период должен быть большим, что-то порядка месяца-двух.

Peter Koch написал в 2001 году более свежие рекомендации по установке значений в записи SOA (RIPE DNS GUIDE). По факту (значениям, перечисленным в примере записи SOA) они ничем не отличаются от приведенного выше примера.

В новых рекомендациях существует важное замечание относительно значения параметра minimum. Интерпретация его значения зависит от того, поддерживает ли конкретная реализация BIND негативное кэширование. Если поддерживает и mininum - это время негативного кэширования, то следует указывать значение 3600, если не поддерживает, то - 172800.

Обратите внимание на то, что все атрибуты записи SOA определяют порядок взаимодействия master сервера и slave серверов. Исключение составляет только атрибут minimum. В данном случае речь идет о кэшировании адресов не только сервером, но системой "умного" resolver.

Дело в том, что в качестве resolver на локальной вычислительной установке выступают функции из стандартной библиотеки, которые направляют рекурсивные запросы к локальному серверу доменных имен и сами не осуществляют кэширования соответствий между именами и IP-адресами. В этом случае кэширование осуществляется локальным сервером доменных имен.

Если на машине запускается свой кэширующий сервер, который не описывает никакой зоны, а используется только для обслуживания запросов к сервису доменных имен, то именно он и является той частью resolver, которая кэширует соответствия.

Таким образом, когда речь идет об атрибуте minimum, то чаще всего его описывают в контексте программы named, которая может использоваться не только как master или slave сервер, но и как кэширующий сервер. В этом случае поле ttl записи описания ресурса, директива управления $TTL и атрибут minimum задают время хранения записи описания ресурса в кэше.

Приведем еще один пример записи типа SOA:

```
example.com. 3600 SOA dns.example.com. hostmaster.example.com. (
1999022301 ; serial YYYYMMDDnn
1d ; refresh
2h ; retry
```

30d ; expire
1H) ; minimum

В данном примере имя зоны указано непосредственно - example.com. Время ttl для самой записи указано равным часу, хотя записи SOA и не хранятся в кэше. Primary master зоны указан как dns.example.com, и его имя отличается от имени домена. Адрес электронной почты администратора соответствует всем существующим рекомендациям - hostmaster@example.com. Серийный номер версии описания зоны занимает 10 позиций и соответствует шаблону даты. Время обновления базы данных описания зоны на slave серверах установлено равным 1 суткам, время повторения попыток обновления установлено равным 2 часам, период "умирания" зоны равен одному месяцу, а время негативного кэширования установлен равным одному часу.

Обратите внимание, что интервалы заданы не в секундах, а в более понятной нотации (минутах (m), часах (h), днях (d), неделях (w)).

И еще одно замечание в завершении этого материала. Мы много говорили о параметрах негативного кэширования, но вопроса о необходимости этого самого кэширования не обсудили. Мы сделаем это отдельно, т.к. кэширование отрицательных ответов вызвано не только теоретическими соображениями, но и суровой прозой жизни.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. M. Andrews. RFC 2308. Negative Caching of DNS Queries (DNS NCACHE). 1998.(http://www.ietf.org/rfc/rfc2308.txt?number=2308)
4. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
5. P. Beertema. RFC 1537. Common DNS Data File Configuration Errors. 1993.(http://www.ietf.org/rfc/rfc1537.txt?number=1537)
6. D. Crocker. RFC 2142. MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS. 1997.(http://www.ietf.org/rfc/rfc2142.txt?number=2142)
7. Peter Koch. RIPE-203. Recommendations for DNS SOA Values. 1999. (http://www.ripe.net/docs/ripe-203.html)
8. R. Elz, R. Bush. RFC 2181. Clarifications to the DNS Specification. 1997. (http://www.ietf.org/rfc/rfc2181.txt?number=2181)
9. R. Elz, R. Bush. RFC 2182. Selection and Operation of Secondary DNS Servers. 1997. (http://www.ietf.org/rfc/rfc2182.txt?number=2182)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind9.html - страничка BIND 9.2.1
2. http://www.ludd.luth.se/~kavli/BIND-FAQ.html - DNS FAQ. Ответы на большинство вопросов, начинающих и "продвинутых" администраторов. Есть только одно большое НО! Этот материал посвящен BIND версии 4. Но все, что касается описания зоны вполне подходит и для более поздних версий BIND.
3. Peter Koch. INTERNET-DRAFT. RIPE DNS WG Guide To Setting Up a DNS Server. 2001. (http://www.techfak.uni-bielefeld.de/~pk/dns/draft-koch-ripe-dns-setup-guide-01.txt.gz)- это рекомендации по установке параметров записи SOA и описание применения других записей RR.

4. R. Elz, R. Bush. RFC 1982. Serial Number Arithmetic. 1996.(http://www.ietf.org/rfc/rfc1982.txt?number=1982)- этот документ для особо дотошных читателей. В нем подробно описана концепция непрерывного арифметического пространства серийного номера и приведены примеры изменения номера и сравнения номеров.

7. b. Запись "Name Server"(NS), проблема некорректного делегирования зон (lame delegation), метрика RTT (round trip time) и ее применение в BIND

Здесь мы обсуждаем вопросы применения записи описания ресурсов NS (Name Server). Подробно разбираем вопрос некорректного делегирования зон, а также алгоритм выбора наиболее подходящего авторитативного сервера для обслуживания запросов к зоне.

Запись NS используется для обозначения сервера, который поддерживает описание зоны. Собственно, когда resolver обращается к системе доменных имен за IP-адресом или доменным именем, то, скорее всего, он получит для начала NS-запись, которая будет указывать на сервер доменных имен, который знает где можно получить необходимую информацию. Именно этот отклик и называют рефералом или отсылкой.

Кроме того, запись NS позволяет делегировать поддомены, поддерживая тем самым иерархию доменов системы доменных имен Internet. Для того, чтобы домен был виден в сети в зоне описания домена старшего уровня должна быть указана запись типа NS для этого поддомена. Формат записи NS можно записать в виде:

[domain][ttl] IN NS [server]

В поле domain указывается имя домена, для которого сервер, указанный последним аргументом записи NS, поддерживает описание зоны. Значение поля ttl обычно оставляют пустым, тем самым, заставляя named устанавливать значение по умолчанию (поле minimum из записи SOA для данной зоны в старых версиях BIND или значение директивы управления $TTL).

При указании имени сервера следует иметь в виду следующие обстоятельства: если в качестве сервера указывается сервер из текущей зоны, то можно обойтись только неполным именем хоста, не указывая полное доменное имя, т.к. имя может быть расширено именем домена, но это скорее исключение, чем правило.

Записи NS указывают как на master, так и на slave серверы. Последние, обычно, принадлежат другим зонам, и для них следует указывать полное доменное имя сервера с указанием имени машины и имени зоны. Например, для сервера из домена polyn.kiae.su с именем ns следует писать полностью ns.polyn.kiae.su. Для того чтобы придерживаться какого-то единого стиля и во избежание ошибок, рекомендуется писать всегда полное имя сервера.

Какой либо разницы между master и slave в NS не указывается. Обычно primary master записывают первым, а резервные серверы указывают вслед за ним.

Приведем пример записи описывающей сервер доменных имен для поддомена:

```
$ORIGIN vega.ru.
zone1 IN NS ns.zone1.vega.ru.
```

В данном примере в качестве первой строки используется команда $ORIGIN, которая определяет имя текущей зоны (см. материал "Файлы описания зоны. Директивы управления"). Здесь она введена только для того, чтобы определить текущую зону.

В поле имени зоны указано имя zone1 без точки на конце. В данном контексте это означает делегирование зоны в домене vega.ru, полное имя которой будет zone1.vega.ru.

В качестве имени сервера, который управляет делегированной зоной указано имя ns.zone.vega.ru, т.е. полное имя машины в делегированном домене. В принципе, в данном случае можно было бы написать и по другому:

```
$ORIGIN vega.ru.
zone1 IN NS ns.zone1
```

т.к. имя принадлежит текущему домену и может быть расширено его именем.

В этом месте, видимо, следует обратить внимание на тот факт, что для записи доменных имен и имен зон существуют не только синтаксические ограничения, о которых упоминалось в материале "Правила именования хостов", а также при описании записи SOA, но и численные ограничения.

Длина метки узла дерева доменных имен должна находится в пределах от 1 до 63 октетов (RFC 2181). Любопытно, что написано не "символов", а октетов, то бишь байтов.

При этом длина полного имени (Full Qualified Doman Name -FQDN) не должна превышать 255 октетов, включая разделители (те самые символы ".", которые ставятся между именами меток узлов). Полное имя узла - это имя от узла до корня дерева доменных имен.

Таким образом, максимальное имя корпоративного домена в зоне ru (что-то типа "corporative-name.ru") не может быть больше 63 символов, а вместе с двумя точками и "ru" не должно превышать 67 символов.

Теперь вернемся к NS записям. NS-записи обычно следуют сразу за записью SOA в файле описания зоны и указывают на серверы, которые ответственны за эту зону:

```
$ORIGIN ru.
Webstatistics 3600 IN SOA ns.webstatistics.ru. hostmaster.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
```

В данном случае описана зона webstatistics.ru и для нее вслед за SOA указано два сервера доменных имен: ns.webstatistics.ru - primary master и ns4.nic.ru - slave.

Такого простого указания на серверы доменных имен зоны для нормальной работы недостаточно. Нужно еще знать IP-адреса этих серверов. Для slave его можно найти в зоне nic.ru, а вот для ns.webstatistics.ru нужно использовать "приклеенную" адресную запись (см. подробнее материал "Адресная запись "Address". Балансировка нагрузки. Round Robin. Учет топологии сетей.").

Рассмотрим еще одну причину использования NS-записей в описаниях зон - делегирование зоны.

До сих пор мы рассматривали NS запись только для объявления сервера доменных имен для той зоны, которую описываем, например, zone.ru:

```
@ IN SOA ns.zone.ru. hostmaster.zone.ru. (
2002092602 1d 3h 4w 3h )
IN NS ns.zone.ru.
IN NS ns.provider.ru.
```

Но это только часть функций NS записи. Теперь делегируем право ответственности за часть зоны другому администратору и выделяем в домене zone.ru зону subzone1. Этот факт мы должны отобразить в описании зоны zone.ru:

```
@ IN SOA ns.zone.ru. hostmaster.zone.ru. (
2002092602 1d 3h 4w 3h )
IN NS ns.zone.ru.
IN NS ns.provider.ru.
ns IN A 192.168.0.1
;
subzone1 IN NS ns.subzone1.zone.ru.
IN NS ns.zone.ru.
ns.subzone1 IN A 192.168.1.1
```

В данном случае имя subzone1 будет расширено именем текущей зоны (zone.ru), т.к. оно не завершается символом ".", а текущим является zone.ru, которое берется из файла конфигурации named потому, что указан символ "@" в записи SOA описания зоны.

На то, что subzone1 - это имя поддомена, указывает запись NS, потому, как у нее первое поле принимает значение имени домена (зоны).

При этом для зоны subzone1.zone.ru определено два авторитативных сервера: ns.subzone1.zone.ru и ns.zone.ru. Теперь, если к нашему серверу обратятся за адресом хоста из subzone1.zone.ru, то отошлем их к другим серверам, которые ответственны за эту зону.

Фраза "отошлем к другим серверам" означает то, что наш сервер вернет запрашивающему его клиенту или серверу, который выполняет рекурсивный запрос, имена и, если в зоне они прописаны, IP-адреса серверов доменных имен, ответственных за зону, где находится хост с заданным клиентом именем или адресом.

Понятно, что кроме нас адреса сервера домена, расположенного ниже в иерархии доменных имен, чем наш домен, никто не знает, поэтому в нашей зоне и содержится "приклеенная" адресная запись для соответствующего сервера:

```
ns.subzone1 IN A 192.168.1.1
```

Если внимательно присмотреться к описанию этого примера, то становится ясно, что наш сервер никуда никого отправлять не будет, т.к. он сам является авторитативным для зоны subzone1.zone.ru. Скорее всего, он является slave-ом, а ns.subzone1.zone.ru - это master.

Просто мы имеем еще один файл описания зоны, который копируем с ns.subzone1.zone.ru, и где находится описание зоны subzone1.zone.ru.

Подобная ситуация довольно типична, если клиенты провайдера используют его же сервер доменных имен в качестве slave, или если внутри организации происходит делегирование зон отдельным подразделениям. В первом случае выполняется условие независимого подключения, а во втором, т.к. домен находится ниже второго (корпоративного уровня), требования независимого подключения соблюдать хоть и желательно, но вовсе необязательно.

Теперь, когда понятна процедура делегирования зоны, вернемся к NS записям, описывающим серверы нашей зоны в нашем же описании зоны, а не там, где нам эту зону (нашу зону) делегировали.

С нашим сервером проблем не возникает. Мы сами его администрируем. А вот с администраторами серверов, которые (серверы) мы прописываем в зоне в качестве авторитативных, следует сначала договориться. Не нужно прописывать имена серверов "от балды". Эти серверы должны быть действительно slave для вашей зоны. Как правило, при регистрации домена проверяются все серверы, указанные в заявке.

Вообще говоря, по идее, неправильное указание имени сервера авторитативного за зону внутри описания самой этой зоны не должно влиять на работу других серверов. BIND не проверяет соответствие серверов в родительской зоне и в зоне-потомке. Другое дело неправильное делегирование (lame delegation), когда сервер родительской зоны перенаправляет клиента к серверу, который не является авторитативным для зоны-потомка.

В нашем случае, например, сервер ns.zone.ru может быть не настроен в качестве slave для зоны subzone1.zone.ru. Тогда ряд запросов может быть отвергнут, что, конечно, плохо.

Еще раз подитожим, что под некорректным делегированием понимают:

- наличие NS-записи, в которой указано имя сервера доменных имен, чей адрес невозможно найти;
- наличие NS-записи, в которой указано имя сервера доменных имен, не отвечающего на запросы;
- наличие NS-записи, в которой указано имя сервера доменных имен, негативно отвечающего на запросы.

На самом деле проблема некорректного делегирования достаточно серьезна. Так, например, Men&Mice в августе 2002 провела исследование 5000 случайно выбранных доменов в зоне com. Некорректное делегирование зоны было выявлено в 20% случаев. Это на один процент лучше, чем было в мае 2002 года, но все равно достаточно много. Любопытно, что в 14% случаев информация о делегировании зоны не совпадала с данными, указанными в самой зоне, а 17% случаев не удалось найти авторитативных серверов зоны.

Вообще говоря, проблема некорректного делегирования имеет решение, если сервер родительской зоны поддерживает механизм зон-заглушек (stub zone). Этот реализован в современных версиях BIND. Он позволяет серверу родительской зоны отслеживать изменения NS-записей делегированной зоны.

Для создания stub зоны следует в файл конфигурации named добавить:

```
zone "webstatistics.ru" {
type stub;
file "webstatistics.ru";
masters {144.206.192.60;};
}
```

В этом случае сервер, на котором организована stub зона будет срисовывать в соответствующий файл запись SOA и NS-записи описания зоны.

Пусть описание зоны будет иметь вид:

```
$ORIGIN ru.
Webstatistics 3600 IN SOA webstatistics.ru. paul.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
$ORIGIN nic.ru.
ns4 IN A 194.226.96.8
$ORIGIN webstatistics.ru.
www 3 IN A 144.206.192.60
3 IN A 144.206.192.61
3 IN A 144.206.192.62
3 IN A 144.206.160.32
```

Содержание файла типа stub на для зоны webstatistics.ru будет иметь следующий вид:

```
; BIND version named 8.2.2-P5-NOESW Mon Mar 20 20:39:05 GMT 2000
; BIND version root@monster.cdrom.com:/usr/obj/usr/src/libexec/named-xfer
; zone 'webstatistics.ru' first transfer
; from 144.206.192.60:53 (local 144.206.160.32) using AXFR at Mon Oct 7 19:31:1
; 7 2002
$ORIGIN ru.
Webstatistics 3600 IN SOA ns.webstatistics.ru. paul.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
; Ignoring info about ns4.nic.ru, not in zone webstatistics.ru.
; $ORIGIN nic.ru.
; ns4 60575 IN A 194.226.96.8
```

Из этого примера хорошо видно, что в зону были скопированы записи SOA, NS и "приклеенные" адресные записи. При этом адресная запись для slave сервера была проигнорирована, т.к. она принадлежит другой зоне и в ней нет необходимости (подробнее см. материал "Адресная запись "Address". Балансировка нагрузки. Round Robin. Учет топологии сетей.").

На самом деле есть одна загвоздка, которая делает указанный выше метод не достаточно удобным. Дело в том, что серийный номер родительской зоны не изменяется и, следовательно, slave-ы родительской зоны об изменениях в делегировании ничего знать не будут, если на них не настроить поддержку stub зоны.

В документации по BIND версии 9 не рекомендовано использовать зоны-заглушки для вновь создаваемых зон. Разработчики BIND советуют просто строго следовать правилам делегирования зоны.

При делегировании зоны нужно четко отдавать себе отчет в том, что администраторы делегированного домена не обязаны сообщать нам адрес primary master, т.е. того сервера, на котором происходит реальное редактирование информации о зоне. Они (администраторы) могут просто сообщить нам адреса своих slave серверов, которые тоже являются авторитативными для зоны. В этом случае мы будем иметь невидимый primary master сервер. Делается это, главным образом, из соображений безопасности.

На самом деле, остался не выясненным еще один вопрос, который тесно увязан с авторитативными серверами доменных имен. Это вопрос о порядке их опроса локальным сервером доменных имен, который выполняет рекурсивный запрос resolver-а.

На самом деле все достаточно просто. Когда сервер получает рекурсивный запрос, и в итоге опроса серверов доменных имен в конце концов получает список авторитативных серверов требуемой зоны, он для каждого из них включает метрику RRT (round trip time). Эта метрика в миллисекундах измеряет время отклика от каждого из серверов. В начальный момент времени, т.е. при первом запросе, она устанавливается в 0.

Сначала берется первый сервер, и отклик получается от него. Метрика этого сервера увеличивается, следовательно при повторном обращении к авторитативным серверам зоны наш сервер, который выполняет рекурсивный запрос, обратится уже к другому серверу зоны. После опроса всех серверов, наш сервер будет обращаться к тому, у которого метрика RTT меньше.

Наиболее эффективен этот алгоритм при работе с корневыми серверами,т.к. к ним приходится обращаться постоянно.

Следует заметить, что серверы, отличные от BIND могу реализовывать другие алгоритмы выбора сервера, т.к. в спецификации DNS этот вопрос не прояснен.

Любопытно, что измерения RTT для корневых серверов проведенные в 2000 году, хорошо совпали с RTT команды ping. Не обобщая этот вывод на все случае жизни, тем не менее можно сказать, что доступность DNS сервера можно оценивать по скорости отклика ping.

Вообще говоря, описанный алгоритм выбора аторитативного сервера на основе RTT не догма и его эффективность постоянно подвергается оценке. Основной вопрос, на который хотят получить ответ службы поддержки корневых серверов- это почему основная нагрузка падает на A сервер, в то время как ближайший к нему M сервер имеет нагрузку почти на четверть меньшую.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)

2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
4. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
5. 5. R. Elz, R. Bush. RFC-2181. Clarifications to the DNS Specification. 1997. (http://www.ietf.org/rfc/rfc2181.txt?number=2181)

Полезные ссылки:

1. http://www.menandmice.com/6000/61_recent_survey.html - результаты исследования Men&Mice по проблемам делегирования зон. Август 2002 года.
2. http://www.dns.net/dnsrd/trick.html#which-server-queried - FAQ. Описание алгоритма выбора сервера среди авторитативных серверов зоны на основе RTT.
3. http://www.caida.org/projects/dns-analysis/#logfile - анализ нагрузки корневых серверов и исследование эффективности алгоритма выбора сервера.
4. http://www2.auckland.ac.nz/net/Internet/rtfm/meetings/47-adelaide/pp-dist/ - материал 2000 года, в котором подмечено совпадение RTT DNS и ping.

7. c. Адресная запись "Address". Балансировка нагрузки. Round Robin. Учет топологии сетей.

В этом материале подробно разбирается формат, назначение и способы применения адресной записи описания ресурсов системы DNS. Особое внимание уделено также вопросам перебора адресных записей с целью баланс нагрузки на сервисы Internet.

Основное назначение адресной записи - установить соответствие между доменным именем машины и IP-адресом. Собственно, это главная задача всей системы доменных имен. По этой причине адресная запись описания ресурса является одной из ключевых записей описания зоны.

Вообще говоря, перед прикладным программным обеспечением, которое работает с DNS, всегда стоит две основных задачи. Во-первых, зная доменное имя хоста, найти его IP-адрес, а, во-вторых, зная IP-адрес, установить доменное имя. Первую задачу будем называть "прямой", а вторую "обратной".

При описании зон используется та же терминология. "Прямая" зона служит для решения "прямой задачи", а "обратная" зона - для решения "обратной" задачи. Решение как прямой, так и обратной задачи не являются тривиальными, т.к. каждому доменному имени можно поставить в соответствие несколько IP-адресов и, наоборот, каждому IP-адресу можно поставить в соответствие несколько доменных имен.

Обе задачи система DNS решает посредством так называемых стандартных запросов (standard queries), в которых указывается доменное имя (QNAME), тип записи описания ресурсов(QTYPE) и класс записи описания ресурсов (QCLASS).

Как при помощи этих трех параметров решается "обратная" задача мы рассмотрим в материале, посвященном записи описания ресурсов типа PTR (Pointer, указатель). "Прямую" задачу решают при помощи адресной записи описания ресурсов, которая имеет тип "A".

Адресная запись - это основа файла описания "прямой" зоны. Запись имеет следующий формат:

[host][ttl] IN A [address]

Поле **host** определяет доменное имя хоста (например, компьютера, на котором установлена и производит обслуживание программа, реализующая один из сетевых сервисов, например, web-сервер).

Чаще всего, это имя указывается относительно имени текущей зоны, поэтому на конце имени не проставляется символ ".", т.е. задается неполное имя хоста.

Если же надо описать машину из другой зоны, то следует указать ее полное доменное имя (fully qualified name) и не забыть в конце этого имени символ ".". Например, "quest.polyn.kiae.su.", указанное в поле host ссылается на машину с именем quest.polyn.kiae.su. Можно также применить директиву управления $ORIGIN и переопределить имя текущей зоны.

Поле ttl обычно оставляют пустым. Поле ttl определяет время хранения адресной записи в кэше сервера доменных имен или кэше "умного" resolver-a. Если значение поле ttl не задано, то оно берется из директивы управления $TTL для BIND 8 и 9 версий, или из поля minimum записи SOA для старых версий BIND.

Приемлемым временем кэширования можно принять значение "3 часа", что обычно записывается в виде числа секунд - 10800:

My-host 10800 IN A 192.168.0.1

В поле address указывают IP-адрес машины. В этом адресе точку на конце ставить не надо.

При делегировании доменов и в самом начале описания зоны (поле primary master сервера записи SOA и NS) возникает проблема описания серверов, поддерживающих описания зоны и описания поддоменов. В этих записях указываются доменные имена серверов, но не указывается их IP-адресов.

DNS - это только информационный сервис, который сам опирается на TCP/IP. При маршрутизации пакетов по сети нужно знать IP-адреса, а их-то как раз и не хватает для обращения к серверам поддоменов и к серверу зоны, если он расположен в той же зоне.

Для решения этой проблемы используются "приклеенные" (буквально "glue") записи типа "A". При этом нет разницы что будет указано в описании зоны раньше - использование доменного имени или определение соответствия этого имени IP-адресу.

Теперь приведем пример записи типа A и использование "приклеенных" адресных записей. Во-первых, рассмотрим простую запись типа A:

```
$ORIGIN vega.ru.
vega-gw IN A 194.226.43.1
```

В данном случае в зоне vega.ru определяется доменное имя для машины с IP-адресом 194.226.43.1. Для каждой машины зоны будет указана такая же запись.
Таким образом мы назначаем каждому хосту в локальной сети доменное имя.

Теперь рассмотрим использование "приклеенной" записи для сервера зоны, который указан в записи SOA:

```
$ORIGIN vega.ru.
@ IN SOA vega-gw.vega.ru paul.kiae.su (
101 ; serial number
86400 ; refresh within a day
3600 ; retry every hour
3888000 ; expire after 45 days
10800 ) ; negative caching
IN NS vega-gw.vega.ru.
IN A 194.226.43.1
;
vega-gw IN A 194.226.43.1
```

В данном примере при назначении сервера зоны в записи NS IP-адрес этого сервера еще не определен, но это и не суть важно, главное, чтобы он был определен далее в одной из адресных записей описания зоны.

Кроме этого, для обслуживания почтовых адресов типа:

```
/usr/paul>mail paul@vega.ru
```

"приклеена" еще одна адресная запись, которая назначает текущему домену IP-арес 194.226.43.1.

Формально эта запись определяет адрес для запросов по неполному имени. В данном случае для всех запросов, которые используют имя зоны, определен адрес шлюза, на котором установлено обслуживание всех сервисов, например, http://vega.ru/ или gopher://vega.ru/. Обычно такой подход используют в маленьких сетях для упрощения работы с сервисами этих сетей.

На самом деле, адресная запись для доменного имени зоны и доменного имени сервера vega-gw.vega.ru "приклеенными" не являются. "Приклеенными" называют только те записи, которые необходимы для поиска IP-адреса, который в контексте запроса невозможно найти путем обращения к авторитативному серверу соответствующей зоны. В нашем же случае мы уже добрались до описания нашей зоны, а в ней есть адресная запись соответствия имени зоны некоторому IP-адресу. Скорее всего, это произошло благодаря настоящей "приклеенной" записи адреса нашего сервера в родительской зоне.

Вообще говоря, правильнее проблему "приклеенных" записей рассматривать с точки зрения выполнения запросов к DNS. В откликах на запросы эти записи помещаются в специальную дополнительную секцию, откуда и извлекаются resolver-ом. Собственно, по этой причине их и называют "приклеенными", т.е. дополнительными.

Примером настоящей "приклеенной" записи может служить делегирование зон внутри домена. Если сервер поддомена будет находиться в самом этом поддомене, то кроме как через приклеенную адресную запись мы не сможем узнать его IP-адрес. Он является авторитативным за зону и в описании зоны, которая хранится у него, задано соответствие его собственного доменного имени и IP-адреса. Но как до него добраться? При помощи "приклеенных записей" в описании родительской зоны!

Пусть мы создаем два поддомена zone1 и zone2, серверы которых также расположены в этих же поддоменах:

```
$ORIGIN vega.ru.
@ IN SOA vega-gw.vega.ru paul.kiae.su (
101 ; serial number
86400 ; refresh within a day
3600 ; retry every hour
3888000 ; expire after 45 days
10800 ) ; negative caching
IN NS vega-gw.vega.ru.
IN A 194.226.43.1
;
vega-gw IN A 194.226.43.1
.....
; Glue address records.
zone1-gw.zone1 IN A 194.226.43.2
zone2-gw.zone2 IN A 194.226.43.3
; Subdomain delegation.
zone1 IN NS zone1-gw.zone1.vega.ru.
zone2 IN NS zone2-gw.zone2.vega.ru.
```

Сначала мы определили IP-адреса хостов в рамках текущего домена, а потом определили их в качестве серверов соответствующих зон нашего домена. Записи, определяющие IP-адреса хостов и будут "приклеенными".

Всегда существует соблазн задать лишние "приклеенные" записи. Например, адресную запись для сервера доменных имен провайдера, на котором поддерживается slave для вашей зоны. Такая практика встречалась раньше довольно часто, встречается она и сейчас. На самом деле, такая запись лишняя. Соответствие имени адресу можно получить непосредственно из зоны провайдера.

Размещать ненужные "приклеенные" записи в своей зоне не стоит по целому ряду причин. Во-первых, IP-адрес сервера провайдера может измениться, а Вы этого не узнаете, соответственно, возникнет коллизия некорректного делегирования зоны, и отвечать на запросы будет только один ваш сервер. Во-вторых, современные версии BIND все равно проигнорируют ваши некорректно "приклеенные" записи J.

Рассмотрим пример описания зоны с некорректно "приклеенной" записью:

```
$ORIGIN ru.
webstatistics 3600 IN SOA ns.webstatistics.ru. hostmaster.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
```

$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
$ORIGIN nic.ru.
ns4 IN A 194.226.96.8 ; некорекктно "приклеенная запись

Запись сообщения named при запуске сервера об игнорировании некорректных приклеенных записей будет выглядеть следующим образом:

Oct 7 12:15:34 generate named[136]:
dns_master_load: webstatistics.ru:10: ignoring out-of-zone data (ns4.nic.ru)

Некорректное "приклеивание" адресных записей приведет к увеличению трафика DNS на вашем сегменте сети и нагрузке на ваш сервер доменных имен. С точки зрения пользователей это тоже может вызвать определенные неудобства. Обычно сервер провайдера располагается на канале с большей пропускной способностью, т.е. время отклика у пользователя из-за некорректно приклеенной адресной записи может увеличиться.

Если у вас возникнут проблемы с доступностью сети, то ваша зона вообще может лишиться авторитативных серверов, что тоже является нежелательным, т.к. приводит к увеличению трафика в системе доменных имен, а также к негативному кэшированию отказов.

Рассмотрим теперь случай, когда одному и тому же доменному имени присваивается несколько IP-адресов. Ситуация эта не такая уж и редкая. Пример, который лежит на поверхности - это корневые серверы доменных имен, т.е. серверы, которые обслуживают корневую зону.

Мы знаем, что их 13 и, что за каждым именем корневого сервера скрывается много машинный комплекс. У каждого из этих хостов свой собственный IP-адрес, следовательно, доменное имя соответствует нескольким IP-адресам.

Другой пример - хост выполняет функции шлюза. У него одно доменное имя, но каждый из интерфейсов имеет свой собственный IP-адрес, следовательно, одному имени будет соответствовать несколько IP-адресов.

Еще один пример - балансировка нагрузки на Web-серверах. Балансировка нагрузки через DNS - это, видимо, не самое оптимальное решение, но в качестве первого приближения решения проблемы оно проходит.

Предыдущее замечание относится главным образом к тому, что называют Round Robin алгоритмом. Существуют более оптимальные решения, построенные на основе DNS, например, RFC 1794, но они не реализованы в виде стандартных опций BIND.

Смысл Round Robin алгоритма в DNS легко понять, опираясь на содержание сообщения отклика сервера доменных имен. Дело в том, что в ответ на стандартный запрос сервер отправляет список записей описания ресурсов, которые отвечают требованиям запроса.

Когда запрашиваются записи адресного типа, то клиенту (resolver или локальный сервер) возвращается список записей в том порядке, как они встретились в описании зоны. Именно в этом порядке прикладная программа и начинает проверять адреса с целью установки соединения.

Понятно, что в этом случае первый адрес будет использоваться наиболее часто. Если реализовать циклическую перестановку адресов в отклике сервера доменных имен, то тогда все IP-адреса будут задействованы в относительно равномерно.

Рассмотрим пример:

```
$ORIGIN ru.
Webstatistics 3600 IN SOA ns.webstatistics.ru. hostmaster.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
;
www 3 IN A 144.206.192.60
3 IN A 144.206.192.61
3 IN A 144.206.192.62
```

Мы задали три адреса для одного и того же доменного имени. Теперь посмотрим при помощи nslookup, как будет сервер реагировать на наши запросы:

```
> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.192.62, 144.206.192.60, 144.206.192.61

> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.192.61, 144.206.192.62, 144.206.192.60

> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.192.60, 144.206.192.61, 144.206.192.62

>
```

и далее по кругу. Вот это и есть Round Robin алгоритм.

Когда время обработки запросов одинаковое, то нагрузка на все адреса будет тоже примерно одинаковая, но если запросы осуществляются к разнородным ресурсам, то равномерной балансировки нагрузки на серверы, которым назначены соответствующие IP-адреса, добиться не удастся.

Впервые "тасовать" адреса (Shuffle Addresses) предложил Брайн Бичер, но его идея требовала введения дополнительных записей описания ресурсов, поэтому прошло решение Маршала Розе, которое, собственно, и называется Round Robin код.

Round Robin код разрабатывался для кластеров, на которых функционировали серверы доменных имен. Именно это обстоятельство позволяло не заботится о реальной нагрузке на хост, т.к. ее (нагрузку) учитывала кластерная архитектура вычислительной установки и соответствующая операционная среда.

Следует заметить, что современные resolver-ы позволяют игнорировать при соответствующей настройке упорядочивание записей в отклике сервера и опираться на топологию сети, т.е. обращаться к наиболее предпочтительному с их точки зрения IP-адресу. Таким resolver-ом является, например, resolver операционной системы Windows2000.

В последних версиях сервера доменных имен компании Microsoft (для Windows 2000 и NT) Round Robin по умолчанию не используется, а используются предпочтения, основанные на так называемой эвристике LocalNetPriority. Это значит, что в локальной сети всегда предпочтение отдается локальным адресам.

Естественно, что при применении Round Robin алгоритма время кэширования адресных записей следует уменьшить. Встречаются рекомендации, которые советуют установить TTL в 300 секунд. В примере задан интервал в 3 секунды (фактически кэширование отменено) только для того, чтобы кэширование не влияло на сообщения nslookup.

Вообще говоря, 8-я версия BIND позволяет настраивать "тасование" записей. Записи в откликах могут переставляться не циклически, а случайным образом. Для этого в файл конфигурации named в директиву options следует включить нечто похожее на следующий блок:

```
rrset-order {
class IN type A name "www.kyky.ru" order random;
order cyclic;
}
```

В данном случае адресные записи для хоста www.kyky.ru будут "тасоваться" случайным образом, а все остальные записи - циклически. При этом Round Robin будет применяться не только к адресным записям.

К сожалению, в 9-ой версии BIND заказать тип "тасования" нельзя. В этой версии применяется только random-cycling, т.е. начальная точка циклической перестановки выбирается случайным образом. Более того, он применяется по умолчанию, как только встретиться подходящий набор записей (RRset).

Если в вашей конфигурации встретиться опция rrset-order, а сервер эту опцию не поддерживает, то в логах появится запись вида:

```
Oct 7 12:44:47 generate named[136]:
/etc/named.conf:7: option 'rrset-order' is not implemented
```

Мы написали "к сожалению", но на самом деле существуют веские причины, по которым случайный выбор порядка адресов тоже не является удачным решением. Обычно это

происходит из-за короткого списка IP-адресов, что не позволяет реально балансировать нагрузку.

Перестановка адресов в купе с негативным кэшированием также не является приятной особенностью системы DNS. Если вдруг из списка IP-адресов один из адресов окажется не доступен, то закэшированный отклик будет постоянно мешать установке соединения с сервисом, у которого остальные IP-адреса функционируют нормально.

В BIND существуют и другие возможности сортировки отклика, состоящего из нескольких записей описания ресурсов - это директивы sortlist и topology. Первая принудительно задает порядок адресов в отклике, например для запросов из локальной сети на первое место в отклике будут ставиться адреса из локальной сети, если они существуют.

Такая ситуация может возникнуть, например, при обращении к удаленному ресурсу, чье зеркало находится в локальном сегменте сети.

Приведем пример использования директивы sortlist. Сначала посмотрим на файл конфигурации named.conf:

```
options {
directory "/etc/namedb";
pid-file "named.pid";
allow-query {any;};
allow-recursion { "kiae"; };
sortlist {
{ 144.206.192/24; { 144.206.192/24; 144.206.160/24;};};
{ 144.206.160/24; { 144.206.160/24; 144.206.192/24;};};
};
};
```

В нем мы добавили в директиву options опцию sortlist. Она задает порядок отображения записей описания ресурсов при доступе из подсетей 144.206.192/24 и 144.206.160/24. Суть его (порядка) заключается в том, чтобы в списке отклика первыми указывались записи, указывающие на ресурсы соответствующей подсети.

При этом сам файл описания зоны будет содержать следующую информацию:

```
$ORIGIN ru.
Webstatistics 3600 IN SOA ns.webstatistics.ru. hostmaster.webstatistics.ru. (
1 3600 600 86400 3600 )
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
$ORIGIN nic.ru.
ns4 IN A 194.226.96.8
$ORIGIN webstatistics.ru.
www 3 IN A 144.206.192.60
3 IN A 144.206.192.61
3 IN A 144.206.192.62
3 IN A 144.206.160.32
```

Для адреса www.webstatistics.ru определены адресные записи из двух перечисленных в файле конфигурации подсетей. Посмотрим с хостов, расположенных в каждой из этих подсетей, что увидит программа nslookup.

Для хоста из 144.206.160.32 получим:

```
> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.160.32, 144.206.192.60, 144.206.192.61, 144.206.192.62

>
```

А для хоста 144.206.192.2 получим:

```
> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.192.60, 144.206.192.61, 144.206.192.62, 144.206.160.32

> www.webstatistics.ru
Server: [144.206.192.60]
Address: 144.206.192.60

Name: www.webstatistics.ru
Addresses: 144.206.192.60, 144.206.192.61, 144.206.192.62, 144.206.160.32

>
```

Из последнего примера видно, что алгоритм "тасования" записей при использовании sortlist не активируется.

Вторая директива (topology) имеет ограниченное применение, т.к. в BIND 9 не поддерживается.

Рассмотрим теперь обратную ситуацию, когда одному IP-адресу можно назначить несколько доменных имен. В принципе, в рамках DNS существует запись описания ресурса CNAME, которая позволяет поддерживать доменные имена синонимы для главного (канонического) имени, за которым и закреплен IP-адрес хоста.

Однако, использование CNAME, во-первых, порождает цикл обращений к серверу для получения сначала канонического имени, а потом уже IP-адреса, а, во-вторых, на использование CNAME наложен целый ряд ограничений.

Ни один документ не запрещает нам использовать несколько адресных записей, которые ставят разные доменные имена в соответствии с одним и тем же IP-адресом. При этом имена могут принадлежать совершенно разным доменам.

Примером такого сорта могут служить одна из разновидностей виртуальных web-серверов (software web server). В этом случае IP-адрес хоста остается постоянным, а вот доменное имя, по которому на сервер попадают запросы, может изменяться, при чем сам web-сервер, зная содержание URL, имеет возможность откликаться на запросы разными страницами.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. M. Andrews. RFC 2308. Negative Caching of DNS Queries (DNS NCACHE). 1998.(http://www.ietf.org/rfc/rfc2308.txt?number=2308)
4. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
5. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

Полезные ссылки:

1. http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q177883& - это материал службы технической поддержки Microsoft, в котором поясняются детали настройки Round Robin.
2. http://www.ietf.org/rfc/rfc1794.txt?number=1794 - T. Brisco. RFC 1794. DNS Support for Load Balancing. Предложения по использованию DNS для баланса нагрузки. В реальной жизни не используется, но почитать интересно.

7.d. Записи типа "Pointer". Домен IN-ADDR.ARPA. Делегирование "обратных" зон. Инверсные запросы.

В этом материале мы рассмотрим, как решается в DNS задача поиска доменного имени по IP-адресу, что из себя представляет загадочный домен IN-ADDR.ARPA, какие проблемы возникают при делегировании "обратных" зон.

Задача поиска доменного имени по IP-адресу является обратной к прямой задаче - поиску IP-адреса по доменному имени. Как было рассмотрено в материале "Адресная запись "Address"…" прямая задача решается в DNS при помощи записей типа A (Address). Обратная же задача решается при помощи записей-указателей типа PTR (Pointer), которые совместно с записями SOA и NS составляют описание так называемой "обратной" зоны.

На самом деле в DNS решение задачи поиска доменного имени по IP-адресу несколько необычно. Казалось бы, что для решения этой задачи можно использовать описание "прямой" зоны. В ней ведь вся информация есть!

На самом деле решать "обратную" задачу по "прямой" зоне неудобно. Поиск сведется к полному перебору всех зон, т.к. удобного аппарата рефералов (отсылок по NS записям) для IP-адресов в прямых зонах нет.

Если IP-адрес, для которого ищется доменное имя, попадет в зону ответственности сервера доменных имен, или нужное соответствие окажется закэшированным, то

доменное имя сервер найдет без труда. Если IP-адрес не окажется в зоне ответственности сервера, то тогда воспользоваться стандартной процедурой поиска сервер не сможет.

Ведь первое, что делает сервер в выше описанной ситуации - это обращение к корневому серверу, а он-то откуда знает, кому принадлежит запрашиваемый IP-адрес? В системе доменных имен поддерживается иерархия доменных имен, но не IP-адресов.

Вот здесь и возникает довольно логичное и простое решение, которое позволяет использовать стандартный механизм поиска доменного имени для решения "обратной" задачи, - специальный домен, структура которого совпадает со структурой IP-адресов. Называется этот домен IN-ADDR.ARPA.

Сначала немного истории. В то время когда затевалась система доменных имен (примерно 1983 год - год выхода RFC 882 и 883) под Internet подразумевалась сеть ARPA, а кроме нее еще обсуждалась возможность построения единого адресного пространства с CSNET. По этой причине кроме класса записей описания ресурсов IN - INTERNET (в то время ARPA) был еще класс описания ресурсов CS - CSNET (Computer Science Network).

В рамках этой концепции в сети ARPA вводился домен для адресов интернет (IN-ADDR - Internet ADDRess). При этом отдельно оговаривалось, что для других сетей алгоритм поиска доменного имени по IP-адресу может отличаться от того, который предложен для адресного пространства сети ARPA.

При этом доменные адреса в ARPA оканчивались словом "ARPA", вот пример из RFC 883:

.. 9999999 IN NS B.ISI.ARPA
9999999 CS NS UDEL.CSNET
B.ISI.ARPA 9999999 IN A 10.3.0.52
UDEL.CSNET 9999999 CS A 302-555-0000

В нем мы видим, что у записей класса IN доменное имя оканчивается на ARPA, а у всех записей класса CS - на CSNET. Таким образом домен IN-ADDR был доменном верхнего уровня в рамках Internet (ARPA) в то время.

Сейчас уже нет ни CSNET, ни более поздних CH (CHAOS) и HS(Hesiod), которые можно найти в спецификации RFC 1035. Была реорганизована и исчезла ARPA. Но задуманный для адресного пространства ARPA домен IN-ADDR.ARPA остался.

На самом деле, в апреле 2000 года между DARPA (бывшей ARPA) и ICAAN было заключено соглашение о том, что домен верхнего уровня (TLD) ARPA будет использоваться для целей поддержки инфраструктуры Интернет.

Кроме того, само слово "ARPA" следует расшифровывать как "Address and Routing Parameter Area Domain (ARPA)", и не следует его ассоциировать с сетью ARPANET.

Основное назначение домена ARPA - обеспечивать отображение численных величин, определяемых протоколами межсетевого обмена, в пространство имен.

Делегирование поддоменов в домене ARPA возложено на IAB (Internet Architecture Board). В настоящее время в ARPA выделено три поддомена:

- in-addr.arpa для отображения IP-адресов IPv4 в пространство доменных имен;
- ip6.arpa для отображения IP-адресов IPv6 в пространство доменных имен;
- e164.arpa для отображения телефонных номеров формата E.164.

Сама зона ARPA поддерживается корневыми серверами и серверами TLD зон, хотя в соответствии с рекомендациями RFC 2870 для ARPA желательно выделение отдельных серверов.

Имена в домене IN-ADDR.ARPA образуют иерархию цифр, которые соответствуют IP-адресам. Правда, записываются эти имена в обратном порядке относительно написания IP-адреса.

Например, машина vega-gw.vega.ru, которая имеет адрес 194.226.43.1 должна быть описана в домене in-addr.arpa как 1.43.226.194.in-addr.arpa, т.е. адрес записывается в обратном порядке.

Так как речь идет о доменной адресации, то разбиение сети, на подсети в данном случае значения не имеет. Имена обрабатываются точно так же, как и обычные доменные имена. Это значит, что систему доменных имен машин этого домена можно представить, например, в виде:

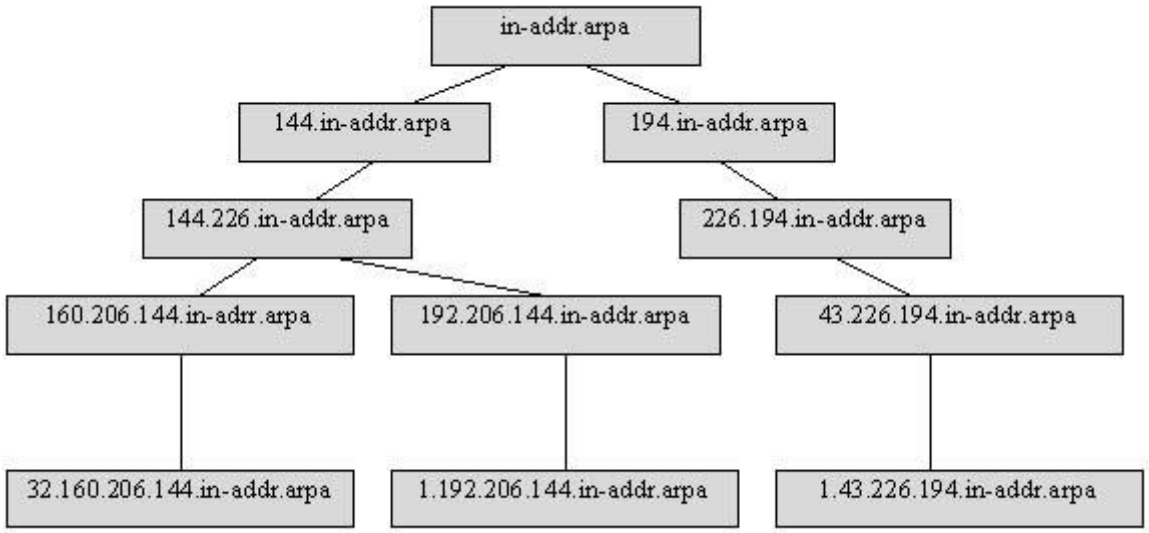

Рис.1. Пример структуры части домена in-addr.arpa

Как видно из рисунка, в данном случае не играет ни какой роли тип сети или разбиение на подсети. Согласно правилам описания доменов и зон в этих доменах, разделителем в имени, который определяет подчиненное значение зоны в домене, является только символ ".".

Когда мы написали, что разбиение на сети и подсети не имеет в данном случае никакого значения, мы имели в виду, только то, что для поиска доменного имени имеет значение только иерархия имен, доменов и хостов.

Однако на самом деле сама идея инверсной записи IP-адреса опирается на принципы построения этого адреса. Понятно, что первая группа цифр, ближайших к корню IN-ADDR.ARPA, задает номер сети, а остальные номер хоста. Принимая во внимание тот

факт, что первоначально при создании доменной адресации в ARPA речь шла о сетях класса A (первый байт-октет определяет номер сети), то первая цифра в доменном имени в зоне IN-ADDR.ARPA - это номер сети, а все последующие - это номер хоста:

1.0.0.10.IN-ADDR.ARPA

Пример определяет первый хост в 10-ой сети.

Поиск этого адреса занял бы поиск сервера, ответственного за 10.IN-ADDR.ARPA, а тот бы ответил именем соответствующим адресу хоста.

В настоящее время, когда адреса раздаются, главным образом, из сетей класса C, и сама классификация сетей подменена спецификацией CIDR, цепочка обращений к серверам доменных имен значительно длиннее, но тем не менее, работает она также исправно, как и вся система DNS.

Теперь, прежде чем обсуждать проблемы поиска доменных имен в IN-ADDR.ARPA, необходимо вернуться к формату описания зон этого домена, которые принято называть "обратными".

Обратная зона состоит, главным образом из записей типа "Pointer" или PTR-записей. Формат записи имеет следующий вид:

[name][ttl] IN PTR [host]

Поле имя задает номер. Это, как уже обсуждалось, не реальный IP-адрес машины, а имя в специальном домене in-addr.arpa или в одной из его зон.Приведем пример PTR записи:

$ORIGIN 43.226.194.in-addr.arpa.
1 IN PTR vega-gw.vega.ru.

По всей видимости провайдер выделили организации сетку класса C 194.226.43.0 (В нотации CIDR 194.226.43.0/24). При этом организации было также делегировано право ведения "обратной" зоны 43.226.194.in-addr.arpa. Вот в этой зоне администратор зоны и начал описывать "обратные" соответствия.

Следует признать, что соответствие обратных зон сетям - это общая практика описания "обратных" зон. Здесь точно также надо делегировать зоны из "обратного" домена. А делается это, как правило, на основе разбиения сети на подсети или сети.

В качестве примера для нашей учебной зоны в файле named.boot (BIND 4 или named.conf для BIND 8 - 9) определена "обратная" зона 43.226.194.in-addr.arpa. Ее описание будет выглядеть примерно так:

$TTL 3600
@ IN SOA vega-gw.vega.ru hostmaster.vega.ru (
101 ; serial number
86400 ; refresh within a day
3600 ; retry every hour
3888000 ; expire after 45 days
3600) ; negative caching
IN NS vega-gw.vega.ru.

```
IN NS ns.relarn.ru.
;
1 IN PTR vega-gw.vega.ru.
4 IN PTR dos1.vega.ru.
5 IN PTR dos2.vega.ru
2 IN PTR zone1-gw.zone1.vega.ru.
3 IN PTR zone2-gw.zone2.vega.ru.
```

Из этого примера видно, что для обратной зоны указаны те же записи описания зоны, что и для "прямой". Кроме того, обратную зону вовсе не обязательно разбивать в соответствии с разделением прямой зоны на другие зоны. Речь в данном случае идет о зонах zone1 и zone2. С точки зрения домена in-addr.apra все машины принадлежат одной зоне - 43.226.194.in-addr.arpa.

Другое дело машина с IP-адресом 127.0.0.1 (localhost). С точки зрения домена in-addr.arpa она принадлежит другой ветке иерархии, отличной от той, которой принадлежат все остальные хосты. Поэтому для домена 127.in-addr.arpa на любой машине есть отдельный файл обратной зоны, хотя localhost, которому соответствует этот IP-адрес, обычно описывается в прямой зоне домена вместе с другими хостами. Вот пример описания этой зоны:

```
; From: @(#)localhost.rev 5.1 (Berkeley) 6/30/90
; $FreeBSD: src/etc/namedb/PROTO.localhost.rev,v 1.6 2000/01/10
; 15:31:40 peter Exp $
;
; This file is automatically edited by the `make-localhost' script in
; the /etc/namedb directory.
;
$TTL 3600
@ IN SOA generate.polyn.kiae.su. root.generate.polyn.kiae.su. (
00000001 ; Serial
3600 ; Refresh
900 ; Retry
3600000 ; Expire
3600 ) ; Minimum
IN NS generate.polyn.kiae.su.
1 IN PTR localhost.polyn.kiae.su.
>
```

Как следует из комментария этого файла для его генерации можно использовать специальный sh-скрипт make-localhost, который входит в комплект поставки BIND. Он берет в качестве основы файл прототипа PROTO.localhost.rev подставляет в него имя хоста и имя домена.

Любопытно, что вместо пользователя hostmaster, что рекомендовано в RFC2142, в данном случае указывается пользователь root. Оно и понятно. Пользователя hostmaster может и не быть, ведь не все читают все подряд RFC, а пользователь root есть обязательно. Тем более, что это скорее всего и есть администратор локального сервера доменных имен.

Следует при этом помнить, что в файле конфигурации named.conf должен быть блок, который указывает на эту "обратную" зону:

```
zone "0.0.127.IN-ADDR.ARPA" {
type master;
file "localhost.rev";
};
```

На самом деле мы опустили один интересный вопрос, а как прямой IP-адрес формата IPv4 преобразуется в доменное имя, которое имеет инверсную запись цифр.

Эти функции возложены на resolver. Resolver преобразует 32-битовый адрес в текстовую строку, размещая октеты в обратном порядке, разделяя их символами ".". К полученному имени resolver через разделитель "." добавляет суфикс "in-addr.arpa". После этого формирует PTR запрос к системе доменных имен.

Вопросы делегирования зон в IN-ADDR.ARPA не столь просты, как может показаться на первый взгляд. Эта проблема тесно связана с получением пулов IP-адресов. Кроме получения адреса, нужно еще получить право на ведение обратного домена, соответствующего вашему адресному пулу.

На самом деле, никто кроме владельца адресного пула не знает, как распределено адресное пространство. Отдать кому-то ведение "обратной" зоны - это значит, во-первых, не обеспечить оперативность управления этой зоной, если только нет средств удаленного управления, как, например, в nic.ru, а, во-вторых, такая услуга стоит денег. По этим причинам обратную зону лучше вести самим.

Кто отвечает за родительскую зону вашего обратного домена, определяется размером вашего адресного пула и тем, как этот адресный пул был получен.

В рамках обсуждения проблем DNS, и, в частности, делегирования "обратных" зон, не очень хочется касаться вопросов выделения IP-адресов и подключения к сети. Все-таки это совершенно другой предмет, тем не менее, мы вскользь пройдем по организационным вопросам получения блока IP-адресов, т.к. это существенным образом влияет на делегирования "обратных" зон.

Прежде всего, следует знать, что первоначально все адреса, которые мы используем в RuNet, получают RIPE Network Coordination Centre (RIPE - это Reseaux IP Europeens), который является одним из трех региональных интернет регистраторов поддерживающих глобальную инфраструктуру Интернет.

Из предыдущего абзаца важно только то, что все блоки адресов в нотации CIDR xxxx.xxxx.xxxx.xxxx/16 и xxxx.xxxx.xxxx.xxxx/24 выдаются этой организацией локальным интернет регистраторам (Local Internet Registres - LIR). А вот уж они и распределяют эти адреса между своими клиентами.

В обычной классификации сетей Интернет это означает, что RIPE NCC отвечает за делегирование в зоне IN-ADDR.ARPA не только сетей класса B, но и сетей класса C. Сети класса B сейчас почти не выдают, поэтому смело можно остановиться на сетях класса C.

Если Вы не являетесь LIR, а Вам выделена сетка класса C, то направить заявку на делегирование обратной зоны в RIPE NCC Вы напрямую все равно не сможете. Сделать это сможет только провайдер, имеющий статус LIR, который, собственно, эту сетку в RIPE NCC получил. Следовательно, нужно связываться с ним и выяснять все вопросы делегирования обратной зоны.

Если Вам выделен пул адресов из сетки класса B, то все вопросы по делегированию обратной зоны нужно также направлять своему провайдеру, т.к. именно ему делегировано право управления обратной зоны этой сети.

Если Вы сами имеете статус LIR, то лучше всего не читать это краткое описание, а обратиться к первоисточникам (http://www.ripe.net/ripencc/mem-services/rigistration/reverse/howtoreverse.html или http://www.ripe.net/.c/ripencc/mem-services/training/material/c.refbooknew.pdf.txt)J.

Тем не менее следует знать, что для делегирования зоны необходимо три вещи:

1. получить адресный блок;
2. обеспечить поддержку зоны /16 или /24 в соответствии с требованиями RIPE NCC;
3. направить заявку в RIPE NCC на делегирование домена.

Адресный блок получают в соответствии с документом RIPE "IPv4 Address Allocation and Assignment Policies in the RIPE NCC Service Region".

Требования по поддержке зоны заключаются в том, что параметры записи SOA зоны должны соответствовать RFC 1912. при этом серийный номер зоны следует записывать в виде YYYYMMDDnn.

Для зон /24 LIR должен организовать два сервера (один у себя, а другой желательно у другого провайдера, во всяком случае, должно быть обеспечено независимое подключение). Для зон /16 должно быть обеспечено 3 сервера, причем один из них дожен быть сервер RIPE NCC ns.ripe.net. Он должен выполнять функцию slave (secondary) для зоны.

Заявка на делегирование зоны, которая посылается в RIPE NCC роботу регистрации на адрес auto-inaddr@ripe.net, должна выглядеть примерно так:

domain: 65.35.80.in-addr.arpa
descr: Reverse delegation for Bluelight 2nd /24
admin-c: JJ231-RIPE
tech-c: JAJA1-RIPE
zone-c: WF2121-RIPE
nserver: ns.bluelight.nl
nserver: ns2.bluelight.nl
mnt-by: BLUELIGHT-MNT
changed: jan@bluelight.nl 19991110
source: RIPE

О назначении полей и правилах их заполнения лучше всего справиться у вашего LIR, либо в nic.ru, либо в RIPE.

На самом деле, общий порядок при создании своей собственной "обратной" зоны точно такой же, как и при создании "прямой". Сначала создается ее описание, и запускаются серверы, которые будут ее обслуживать, а потом заполняются заявки и начинается работа с провайдером.

Отдельно обратим внимание на то, что для каждой "обратной" зоны, которая соответствует сети класса C (адреса в CIDR xxxx.xxxx.xxxx/24) нужно свое собственное

описание зоны. Нельзя в один файл мешать несколько зон данного типа. Мешать их может только в файле, описывающем родительскую зону типа xxxx.xxxx/16, например, в зоне 206.144.in-addr.arpa могут быть записи типа PTR для зон 160.206.144.in-addr.arpa и 192.206.144.in-addr.arpa, и то, только при условии, что данные зоны не делегированы на другой сервер.

Действительно, NS запись для делегирования будет находиться в том же файле описания зоны, что записи типа PTR. BIND отловит эту коллизию и проигнорирует соответствующие PTR записи.

Таким образом, вопрос зон, соответствующих целым сетям или пулам адресов, которые кончаются на границах октетов, доставит их администраторам гораздо больше проблем в области организационных мероприятий, нежели станет технической проблемой.

Но на самом деле для небольших компаний гораздо более серьезной проблемой становится делегирование обратных зон для пулов адресов, меньших, чем сеть класса C. Рассмотрим эту проблему более подробно. Все примеры будем брать из RFC 2317, т.к. именно оно рекомендовано RIPE NCC для решения нашей проблемы. Именно этот способ поддерживается в самом RIPE NCC.

Сначала о существе проблемы. Провайдер "нашинковал" в сети класса C (пул адресов в нотации CIDR /24) на несколько частей. Таким образом границы пулов адресов не приходятся на границы октетов.

Пусть мы имеем дело с сетью 192.0.2.0/24. Провайдер разделил адресное пространство следующим образом:

192.0.0/25 - организация A
192.0.128/26 - организация Б
192.0.192/26 - организация C

Классическое описание обратной зоны может выглядеть примерно так:

```
$ORIGIN 2.0.192.in-addr.arpa.
;
1 PTR host1.a.ru.
2 PTR host2.a.ru.
3 PTR host3.a.ru.
;
129 PTR host1.b.ru.
130 PTR host2.b.ru.
131 PTR host3.b.ru.
;
193 PTR host1.c.ru.
194 PTR host2.c.ru.
195 PTR host3.c.ru.
```

Делегировать поддержку такой зоны можно только кому-то одному. Все остальные будут зависимы от владельца зоны и все изменения должны будут вносить через него.

По этой причине предлагается довольно оригинальное решение. Для того, чтобы не завязываться на кого-то одного, владелец пула адресов (в нашем случае провайдер,

который выделяет их организациям) создает обратную зона из записей синонимов CNAME, переправляя, таким образом, запросы на другие серверы доменных имен, которые уже будут поддерживать те самые организации, что получили от него (провайдера) соответствующие блоки:

```
$ORIGIN 2.0.192.in-addr.arpa.
@ IN SOA ns.provider.ru. hostmaster.provider.ru. (…)
;
; 0-127 /25
;
0/25 IN NS ns.a.ru.
0/25 IN NS ns.slave.ru.
;
1 IN CNAME 1.0/25.2.0.192.in-addr.arpa.
2 IN CNAME 2.0/25.2.0.192.in-addr.arpa.
3 IN CNAME 3.0/25.2.0.192.in-addr.arpa.
;
; 129-191 /26
;
128/26 IN NS ns.b.ru.
128/26 IN NS ns.slave.ru.
;
129 IN CNAME 129.128/26.2.0.192.in-addr.arpa.
130 IN CNAME 130.128/26.2.0.192.in-addr.arpa.
131 IN CNAME 131.128/26.2.0.192.in-addr.arpa.
;
; 193-255 /26
;
192/26 IN NS ns.c.ru.
192/26 IN NS ns.slave.ru.
;
193 IN CNAME 193.192/26.2.0.192.in-addr.arpa.
194 IN CNAME 194.192/26.2.0.192.in-addr.arpa.
195 IN CNAME 195.192/26.2.0.192.in-addr.arpa.
```

Сами организации при это обязаны зависти зоны следующего содержания:

```
$ORIGIN 0/25.2.0.192.in-addr.arpa.
@ IN SOA ns.a.ru hostmaster.a.ru (…)
IN NS ns.a.ru.
IN NS ns.slave.ru.
;
1 IN PTR host1.a.ru.
2 IN PTR host2.a.ru.
3 IN PTR host3.a.ru.
```

Для блока 128/26 соответственно последовательность "0/25" следует заменить на "128/26", а сервер ns.a.ru на сервер ns.b.ru. Для блока 192/26 нужно также будет выполнить соответствующие замены.

Идея данного метода заключается в том, что, попадая в зону 2.0.192.in-addr.arpa, локальный сервер доменных имен или resolver на свой запрос доменного имени получают CNAME, т.е. сервер зоны говорит, что имя, которое ему было сообщено не является

каноническим именем хоста, указанным в адресной записи. Это лишь синонимом канонического имени. При этом каноническое имя сообщается в качестве ответа на запрос. Обращаясь по каноническому имени, локальный сервер доменных имен или resolver получают в ответ ссылку на другой сервер, т.к. в нашей зоне указан NS для поддержки зон с каноническими именами. Переходя на новый сервер, локальный сервер доменных имен или resolver получают уже обычную PTR запись ресурсов и могут сообщить доменное имя приложению, которое инициировало запрос к обратной зоне.

Может показаться накладным набивать большое количество CNAME в зонах типа 2.0.192.in-addr.arpa. Но здесь можно воспользоваться директивой управления $GENERATE, которая существенно сократит число набираемых вручную записей описания ресурсов.

Следует также обратить внимание на то, что имена зон в 2.0.192.in-addr.arpa могут быть любыми допустимыми именами, а не только "0/25" или "192/26", как это указано в примере. Более того, имена могут указывать на зоны из совершенно других доменов, не входящих в in-addr.arpa. Правда, зоны должны содержать соответствующие PTR записи описания ресурсов.

Мы в самом начале этого материала упоминали об инверсных запросах, которые не следует путать со стандартными запросами к серверам домена IN-ADDR.ARPA. Какова их судьба?

В принципе, поддержка инверсных запросов в серверах доменных имен возможна. Во всяком случае, они обязаны такие запросы распознавать и отвечать на них, либо списками доменных имен, либо 0, либо сообщением, о том, что данную опцию сервер не поддерживает.

При этом реальная область применения такого сорта запросов ограничивается рамками одного сервера.

И в заключении о том, зачем вообще нужно искать доменные имена по IP_адресам. Этому есть несколько причин.

Во-первых, многие сетевые сервисы блокируют доступ, если не могут отобразить IP-адрес хоста, с которого к ним обращаются, в доменное имя. Так поступают, например, ftp-серверы и серверы электронной почты.

Во-вторых, большое число запросов к "обратным" зонам возникает при работе систем электронной почты и веб-серверов. В электронной почте рассылка почты почтовыми транспортными агентами (ПТА) основана на процедуре канонизации адресов отправителя и получателя. А эта процедура подразумевает обращение к обратной зоне. Кроме того, ПТА блокируют пересылку почты, основываясь, в том числе, и на информации из "обратных" зон. При работе веб-серверов обращение к "обратной" зоне используется для сбора статистики.

В-третьих, при отсутствии "обратных" зон объем трафика, который генерируется прикладным программным обеспечением, гораздо больший, чем при наличии правильно сконфигурированных "обратных" зон.

Любопытно, что на 1999 год по данным RIPE NCC из 163124 зон, которые могли бы быть делегированы (зарегистрированые IP-адреса), реально было делегировано только 85352 или примерно 52%.

И еще одно замечание. Мы здесь вообще не рассматривали "обратное" отображение в рамках такой "экзотики", как адресное пространство IPv6. На самом деле современные версии BIND прекрасно справляются с этой задачей, но об этом как-нибудь в другой раз.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. H.Eidnes, G. de Groot, P. Vixie. RFC-2317. Classless IN-ADDR.ARPA delegation. 1998. (http://www.ietf.org/rfc/rfc2317.txt?number=2317)
4. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
5. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

Полезные ссылки:

1. P. Mockapetris. RFC-883. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc883.txt?number=883) - документ интересен с точки зрения исторического развития домена IN-ADDR.ARPA как домена "обратного" соответствия.
2. R. Bush, D. Karrenberg, M. Kosters, R. Plzak. RFC 2870. Root Name Server Operational Requirements. 2000. (http://www.ietf.org/rfc/rfc2870.txt?number=2870)
3. D. Crocker. RFC 2142. MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS. 1997. (http://www.ietf.org/rfc/rfc2142.txt?number=2142)
4. D. Barr. RFC 1912. Common DNS Operational and Configuration Errors. 1996. (http://www.ietf.org/rfc/rfc1912.txt?number=1912)
5. Policy for Reverse Address Delegation under in-addr.arpa in the RIPE NCC Service Region (http://www.ripe.net/ripe/docs/ripe-224.html) - очень коротенькое описание политики делегирования обратных зон в зоне ответственности RIPE NCC.
6. Reverse Delegation. IPv4 Request Procedure (reverse /24, /16) (http://www.ripe.net/ripencc/mem-services/registration/reverse/howtoreverse.html) - документ в стиле "делай раз, делай два, …". Все просто и доступно, но нужно иметь некоторое представление об устройстве сервисов RIPE NCC.
7. http://www.ripe.net/.c/ripencc/mem-services/training/material/c.refbooknew.pdf.txt - материалы учебного курса для LIR, которые зарегистрированы в RIPE. Интересен раздел 9 - "Reverse Delegation". Достаточно сжато и подробно описывается процедура делегирования "обратных" зон из зоны ответственности RIPE NCC.
8. G.Huston. RFC-3172. Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa"). 2001. (http://www.ietf.org/rfc/rfc3172.txt?number=3172) - статус домена ARPA, начиная с конца апреля 2000 года. Документ фиксирует состояние домена ARPA, ответственность ICAAN и IAB за управление этим доменом, его статус, а также правила делегирования поддоменов.

9. http://www.ripe.net/ripencc/pub-services/stats/revdns/assign/19990426.html - статистика соответствия регистрации пулов адресов и "обратных" зон на 1999 год в RIPE NCC.

7. e. Запись Mail exchanger (MX). Электронная почта и DNS

В этом разделе разбираются вопросы взаимодействия системы электронной почты интернет и системы доменных имен. Подробно разбирается механизм рассылки и пересылки почтовых сообщений, который опирается на информацию, полученную из службы доменных имен.

Если просуммировать все записи описания ресурсов, которые предлагалось указывать в описании зоны, то посвященных электронной почте RR-ов (Resource Records) было бы большинство.

В реальной жизни (лучше, наверное, подошло бы слово "виртуальной") используется запись одного типа - MX (Mail exchanger - почтовый шлюз). Смысл использования этой записи заключается в том, чтобы определить хост, который отвечает за доставку почты в определенный домен или знает, как это делается.

Запись MX может быть определена как для всей зоны, так и для отдельно взятой машины. MX RR имеет следующий формат:

[name] [ttl] IN MX [prference] [host]

Поле name определяет имя машины или домена, на который может отправляться почта. Это может быть как полностью определенное имя, так и частичное имя машины. Поле ttl обычно оставляют пустым. В поле preference указывается приоритет почтового сервера, имя которого указано последним аргументом в поле данных MX-записи.

Прежде чем перейти к подробному обсуждению назначения и практики применения MX записей следует понять основные принципы построения обмена электронной почтой в Интернет.

В своих рассуждениях мы будем опираться на концепцию, которая изложена в RFC 2821. Данный документ заменил старожилов - RFC 821 и RFC 974.

RFC 821 определяло протокол доставки почты - SMTP, а второй документ - взаимодействие электронной почты с системой доменных имен. Новый документ обобщил опыт применения первых двух на практике (например, отменил требование проверки наличия WKS записи для хоста, указанного в качестве шлюза в MX записи, которое все равно никто не соблюдал) и прояснил некоторые недостаточно четко сформулированные требования.

Во-первых, следует остановиться на самом понятии почтового шлюза. Наиболее распространенным в этом контексте были термин MTA (Mail Transfer Agent) и термин MUA (Mail User Agent). Данные термины применяли для того, чтобы подчеркнуть некоторое отличие MTA от почтового сервера, который являлся конечным получателем почты и от клиентского программного обеспечения (MUA), которое обеспечивало только "первую и последнюю мили" пути почтовго сообщения.

Сейчас предпочтение отдано терминологии "клиент-сервер" и термины MTA MUA следует употреблять с осторожностью.

Все дело в том, что ситуация в почтовой службе несколько изменилась. Раньше (во времена существования UUCP, которому посвящено не меньше места в старых RFC) наиболее распространенной процедурой доставки почты была многозвенная доставка с использованием большого числа промежуточных узлов. При этом почта передавалась из одних сетей в другие, в том числе, и с использованием сетей посредников.

При такой технологии работа электронной почты напоминала работу обычной почты, основанную на отделениях связи. Существовали и существуют до сих пор многочисленные узлы-шлюзы. С их списком можно ознакомиться по адресу http://www.faqs.org/faqs/mail/inter-network-guide/.

В настоящее время, когда доминирует SMPT, в понятие шлюза как бы расщепилось на relay (пересылка по SMTP) и gateway (шлюз в другую почтовую систему). В контексте DNS разницы между этими понятиями нет. MX запись указывает на любой хост, на который следует направлять почту, чтобы она попала в требуемый домен.

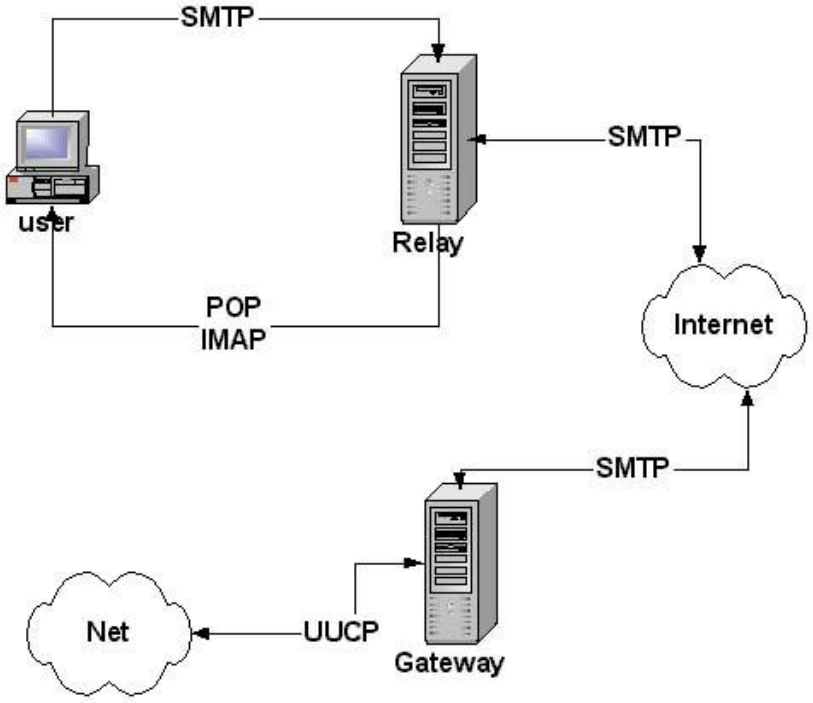

Рисунок.1. Схема доставки и отправки почты.

Кроме того, в почте от первых двух названных (relay и gateway) еще отличают хост, который, собственно, является хранителем почтовых ящиков. Локальная доставка писем адресатам.

Все попытки внедрить в обиход подобное разделение и для системы DNS пока ни к чему не привели. Записи типа MB (Mail Box) в описаниях зон Вы не найдете "днем с огнем, а ночью ощупью".

Поэтому в контексте системы DNS мы все почтовые хосты, для которых устанавливается запись MX, будем называть почтовыми шлюзами.

Таким образом, работа почтового шлюза заключается в том, чтобы принять почту от отправителя и переслать ее получателю, либо хосту, который знает, как эту почту получателю доставить.

Ключевым элементом в почтовом сообщении являются адреса отправителя и получателя. Адреса принято записывать в виде:

local-part@domain

Нас с точки зрения DNS интересует только та часть, которая называется "domain". На самом деле, вовсе не обязательно, что domain представляет из себя правильное доменное имя. Просто название этой части адреса созвучно DNS доменам.

Все обмены данными между шлюзам в рамках интернет производятся по протоколу SMTP (Simple Mail Transfer Protocol). Наверное, каждый пользователь, а уж тем паче администратор, имеет опыт настройки программ чтения и отправки почтовых сообщений на работу с таким шлюзом или, как их еще называют, relay-ем.

Что же делает шлюз, когда он получает почту от клиента для пересылки? Опуская подробности, связанные с ограничениями, накладываемыми администраторами почтовых шлюзов на процедуру пересылки, рассмотрим алгоритм взаимодействия шлюза с системой DNS.

Во-первых, в рамках SMTP-обмена в качестве доменных имен допускаются только полные доменные имена (fully-qualified domain names), которые можно разрешить при помощи поиска MX или A записей в системе DNS. Т.е. в поле domain почтового адреса должно быть имя, для которого есть MX или A запись. В принципе, допускаются и синонимы, т.е. имена для которых в DNS можно найти записи CNAME, перенаправляющие на MX или A записи. На самом деле многие шлюзы настроены таким образом, что CNAME записи игнорируются.

Для того чтобы этого добиться, все почтовые шлюзы прибегают к процедуре канонизации имени, т.е. преобразовывают его к такому виду, который был бы допустимым для системы доменных имен.

По этому имени шлюз производит поиск MX записей. Если он находит CNAME, то заменяет исходное имя каноническим и снова повторяет поиск (мы уже писали, что довольно часто эта возможность отключена в реальной жизни).

Если нет MX записей, но есть адресная запись, то делается попытка доставить почту по этому адресу. Если найден список MX записей, то адресная запись игнорируется.

Более того, если в последующем ни одна из MX записей не подойдет для отправки почты, то попытка доставить почту по IP-адресу, взятому из адресной записи, не будет

предпринята (на самом деле эта опция может быть настроена в конкретном программном обеспечении обмена почтой).

Если шлюз нашел список MX записей, то он сортирует его в порядке возрастания значений поля preference. Записи с меньшим значением этого поля считаются более предпочтительными. Затем шлюз пытается установить SMTP соединение и отправить почту, перебирая MX записи в порядке выставленных предпочтений.

Если MX записи имеют одинаковые предпочтения, то на практике шлюз выбирает наиболее предпочтительную из них случайным образом (во всяком случае, так делают Sendmail и Postfix).

Как только почта успешно отправлена, перебор шлюзов-получателей прекращается.

Приведем пример использования записей MX в описании зоны:

```
$TTL 3600
$ORIGIN vega.ru.
@ IN SOA vega-gw.vega.ru paul.kiae.su (
101 ; serial number
86400 ; refresh within a day
3600 ; retry every hour
3888000 ; expire after 45 days
3600 ) ; negative caching
IN NS vega-gw.vega.ru.
IN NS ns.relarn.ru.
IN NS polyn.net.kiae.su.
IN A 194.226.43.1
IN MX 10 vega-gw.vega.ru.
IN MX 20 ns.relarn.ru.
IN MX 30 relay.relarn.ru.
;
vega-gw IN A 194.226.43.1
IN MX 0 vega-gw
IN MX 10 ns.relarn.ru.
IN MX 20 relay.relarn.ru.
;
www IN CNAME vega.ru.
```

В данном примере мы определили несколько записей типа MX для почтовой рассылки.

Записи в описании зоны, сразу после A-записи, следующей за записью SOA, определяют пути поступления почты на хост-тезку данного домена. Для получения почты по имени домена эта адресная запись не нужна. Ее используют для других интернет-сервисов, скажем для доступа к web-сайту домена не только по доменному имени www.vega.ru, но и по доменному имени vega.ru.

Самый высокий приоритет здесь имеет прямая отправка почты на шлюз vega-gw.vega.ru. Если отправить почту на эту машину не удастся, то почтовый агент должен попробовать путь через почтовый сервер ns.relarn.ru . Если и этот путь окажется невозможным, то почту можно попытаться отправить на пересыльный пункт relay.relarn.ru . Такой порядок опроса почтовых серверов определяется полем preference - чем меньше значение поля, тем выше приоритет сервера в записи MX.

В нашем домене, который мы использует в качестве примера, большинство почтовых ящиков пользователей расположено на машине vega-gw.vega.ru. Для этой машины также указаны несколько записей MX. Наибольший приоритет среди них имеет запись с именем почтового сервера vega-gw. Обратите внимание на то, что данное имя не оканчивается точкой. Это значит, что оно расширяется именем текущей зоны. Все же другие имена серверов - это полностью определенные доменные имена (fully-qualified).

Одной из главных проблем пересылки почты является зацикливание при невозможности отправки непосредственно адресату. Представим ситуацию, когда хост vega-gw.vega.ru недоступен.

Тогда согласно процедуре, описанной выше, почта должна быть отправлена на ns.relarn.ru, он, в свою очередь, не может отправить почту на vega-gw.vega.ru. Себе он отправлять почту тоже не будет, поэтому отправит на relay.relarn.ru, а тот снова на ns.relarn.ru, и так по кругу.

На самом деле ничего подобного не произойдет, т.к. шлюзы обязаны исключать из списка MX записи, которые имеют больший или такой же приоритет, как и они сами. В нашем случае почта будет "отлеживаться" на ns.relarn.ru.

На самом деле связка "DNS-почта" - это очень тонкий инструмент. Программа почтового шлюза имеет множество самостоятельных настроек, и может быть сконфигурирована на игнорирование описанных выше процедур.

MX записи - это один из немногих случаев, где действительно имеет смысл применять wildcard в доменном имени. Если в описании зоны поместить запись вида:

*.domain.ru. IN MX 10 host.domain.ru.

то для любого хоста из зоны domain.ru, у которого не будет своих MX записей, почта будет отправляться на host.domain.ru, а почтовая программа-шлюз этого хоста сама разберется куда дальше посылать почту.

В настоящее время среди прочих коммерческих услуг достаточно популярной стала услуга перенаправления почты - mail forwarding. Ее идея заключается в том, что регистрируется домен с коротким легкозапоминающемся именем и почтовые адреса указываются относительно этого домена. При этом реальные почтовые ящики могут находиться где угодно.

Перенаправление почты осуществляется за счет добавления MX записи в описание зоны почтового домена, которая будет указывать на хост, который знает, как почту в этот домен доставлять.

Естественно, что на самом хосте соответствующим образом должен быть настроен и почтовый шлюз. Типичный пример можно найти в ru-rucenter (http://www.nic.ru/dns/service/no_primary.html)

И напоследок несколько цифр характеризующих проблему обмена электронной почтой с точки зрения DNS. Согласно отчету 2002 года компании men&mice в TLD com из 5000 обследованных доменов не имеют MX записей 18.68% зон. В эти зоны нельзя отправить почту. В 3.3% случаев MX записи указывают на CNAME, т.е. на синонимы, что для некоторых почтовых программ недопустимо. В 4.16%-ах случаев почтовые серверы

соответствующих доменов не работают с DNS правильно, т.е. согласно принятым спецификациям.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
4. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
5. 4. J.Klensin. RFC 2821. Simple Mail Transfer Protocol. 2001. (http://www.ietf.org/rfc/rfc2821.txt?number=2821)

Полезные ссылки:

1. Inter-Network Mail Guide (http://www.faqs.org/faqs/mail/inter-network-guide/) - руководство по пересылке почты между различными почтовыми системами.
2. Craig Partridge. RFC 974. MAIL ROUTING AND THE DOMAIN SYSTEM. 1986. (http://www.ietf.org/rfc/rfc974.txt?number=974) - стандарт интернет, который использовался до RFC 2821.
3. http://www.menandmice.com/infobase/mennmys/vefsidur.nsf/index/5.1 - статистика проблем электронной почты, связанных с неточностью настройки DNS

7. f. Запись назначения синонима каноническому имени "Canonical Name". Особенности использования синонимов при работе с NS и MX записями.

Здесь рассматриваются особенности применения синонимов доменных имен, которые задаются записью описания ресурсов CNAME. Разбираются основные ошибки при комбинировании записи определения синонимов с записями MX и NS.

Здесь рассматриваются особенности применения синонимов доменных имен, которые задаются записью описания ресурсов CNAME. Разбираются основные ошибки при комбинировании записи определения синонимов с записями MX и NS.

Обсуждение записей описания ресурсов очень похоже на движение по ленте Мебиуса. В принципе, можно начинать с любого места (с любой записи), к которому все равно вернешься.

Запись CNAME больше всего подходит для точки начала и окончания, т.к. больше всего ошибок при настройке описания зон связано с применением этой записи в сочетании с другими записями описания зоны.

Запись CNAME определяет синонимы для реального (канонического) доменного имени машины, которое определено в записи типа A. Имя в записи типа A называют

каноническим именем машины. Формат записи CNAME можно определить следующим образом:

[nickname] [ttl] IN CNAME [host]

Поле nickname определяет синоним для канонического имени, которое задается в поле host.

Запись CNAME чаще всего используется для определения имен информационных сервисов Internet, которые установлены на хосте. Так следующий пример определяет синонимы, для компьютера, на котором установлены серверы протоколов http и gopher:

$ORIGIN polyn.kiae.su.
olga IN A 144.206.192.2
www IN CNAME olga.polyn.kiae.su.
gopher IN CNAME olga.polyn.kiae.su.

Директива управления $ORIGIN введена здесь только для определения текущего имени зоны. Две записи типа CNAME позволяют организовать доступ к серверам, используя имена, характерные для соответствующих Интернет-сервисов.

Именно такие синонимы и используются в описаниях зон при обращении к таким системам как Yahoo(www.yahoo.com) или Altavista (www.altavista.digital.com).

Ниже приведен пример обращения к www.netscape.com:

> www.netscape.com
Server: IRIS.polyn.kiae.su
Address: 144.206.192.10

;; res_nmkquery(QUERY, www.netscape.com, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 12635, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 5, authority records = 3, additional = 3

QUESTIONS:
www.netscape.com, type = A, class = IN
ANSWERS:
-> www.netscape.com
canonical name = netscape.com
ttl = 900 (15M)
-> netscape.com
internet address = 64.12.180.19
ttl = 900 (15M)
-> netscape.com
internet address = 64.12.180.22
ttl = 900 (15M)
-> netscape.com
internet address = 64.12.151.211
ttl = 900 (15M)

-> netscape.com
internet address = 64.12.151.215
ttl = 900 (15M)
AUTHORITY RECORDS:
-> netscape.com
nameserver = ns.netscape.com
ttl = 86400 (1D)
-> netscape.com
nameserver = ns1.netscape.com
ttl = 86400 (1D)
-> netscape.com
nameserver = ns2.netscape.com
ttl = 86400 (1D)
ADDITIONAL RECORDS:
-> ns.netscape.com
internet address = 198.95.251.10
ttl = 3600 (1H)
-> ns1.netscape.com
internet address = 149.174.213.7
ttl = 3600 (1H)
-> ns2.netscape.com
internet address = 207.200.73.80
ttl = 3600 (1H)

Name: netscape.com
Addresses: 64.12.180.19, 64.12.180.22, 64.12.151.211, 64.12.151.215
Aliases: www.netscape.com

>

Из отчета nslookup видно, что www.netscape.com является синонимом netscape.com, которая, в свою очередь, имеет несколько адресных записей. В авторитативной секции отклика указаны доменные имена серверов зоны netscape.com, а в дополнительной секции их IP-адреса.

Правда, при обращении к серверу протокола http, который является базовым для системы World Wide Web, изменение имени машины, с которой осуществляют обслуживание? может быть вызвано многими причинами: перенаправлением запроса на другой сервер, использование виртуального сервера и т.п.

При использовании записей типа CNAME следует проявлять осторожность. Некоторые администраторы рекомендуют вообще от нее отказаться и если нужно еще одно имя, то лучше указать еще одну A-запись для того же самого IP-адреса.

На самом деле, применение записей CNAME строго регламентировано в RFC 1034 и RFC 1912. Смысл применения CNAME - назначение синонима для канонического имени. Описание ресурсов для канонического имени и для синонима должны совпадать.

По этой причине для имени, определенного как синоним не должно быть никаких других записей описания ресурсов (На самом деле существуют отдельные исключения, связанные с поддержкой безопасности DNS). Более того, имя синоним никогда не должно появляться в правой части записей описания ресурсов. Нужно также понимать, что синоним должен

быть определен только один раз. Последнее требование соблюдается далеко не всеми реализациями серверов доменных имен.

Администратору зоны следует понимать, как DNS работает с CNAME записями описания зоны. Поиск CNAME осуществляется только в том случае, если запросы другого типа, скажем поиск адресной записи, не дали положительного отклика. Т.е. сервер получает запрос к своей зоне на адресную запись; он не находит такой записи, но в описании зоны есть CNAME запись, которая соответствует запрашиваемому доменному имени; сервер на запрос возвращает эту запись и адресную запись, если последняя находится в описании этой же зоны. Если же CNAME ссылается на каноническое имя из другой зоны, то в отклике будет только CNAME.

Вот часть отчета nslookup, где указан запрос и ответ на него без секции данных секции авторитативности отклика сервера и дополнительной секции:

```
------------
Got answer:
HEADER:
opcode = QUERY, id = 51763, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 2, authority records = 2, additional = 2

QUESTIONS:
user.webstatistics.ru, type = A, class = IN
ANSWERS:
-> user.webstatistics.ru
canonical name = host.webstatistics.ru
ttl = 3 (3S)
-> host.webstatistics.ru
internet address = 144.206.192.63
ttl = 3 (3S)
```

Из этого отчета следует, что мы искали адресную запись для user.webstatistics.ru, но ее в описании зоны нет, поэтому сервер нашел CNAME и адрес для канонического имени синонима.

При этом в RFC 1034 есть строгое указание, что если CNAME будет указывать не на каноническое имя, а опять на синоним, в этом случае сервер должен вернуть негативный отклик, а не CNAME. При появлении такой цепочки в одной зоне определить цепочку можно, но при наличии CNAME цепочки через границы зон выявить ее практически невозможно.

Последнее проверить очень трудно, т.к. клиент при поиске в поле типа записи запроса укажет тип A (адресная запись). Новый сервер не будет знать о результатах предыдущего поиска и снова будет при отрицательном результате искать CNAME.

Вот пример из той же зоны, что и раньше, но мы в зоне реализовали цепочку CNAME:

```
;; res_nmkquery(QUERY, user1.webstatistics.ru, IN, A)
------------
Got answer:
HEADER:
opcode = QUERY, id = 47112, rcode = NOERROR
```

header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 3, authority records = 2, additional = 2

QUESTIONS:
user1.webstatistics.ru, type = A, class = IN
ANSWERS:
-> user1.webstatistics.ru
canonical name = user.webstatistics.ru
ttl = 3 (3S)
-> user.webstatistics.ru
canonical name = host.webstatistics.ru
ttl = 3 (3S)
-> host.webstatistics.ru
internet address = 144.206.192.63
ttl = 3 (3S)
AUTHORITY RECORDS:
-> webstatistics.ru
nameserver = ns.webstatistics.ru
ttl = 3600 (1H)
-> webstatistics.ru
nameserver = ns4.nic.ru
ttl = 3600 (1H)
ADDITIONAL RECORDS:
-> ns.webstatistics.ru
internet address = 144.206.192.60
ttl = 3600 (1H)
-> ns4.nic.ru
internet address = 194.226.96.8
ttl = 19465 (5h24m25s)

Name: host.webstatistics.ru
Address: 144.206.192.63
Aliases: user1.webstatistics.ru, user.webstatistics.ru

Сервер (BIND версии 9) вернул нам правильный IP-адрес, перебрав цепочку CNAME.

На самом деле многие "глупые"(stub) клиенты не умеют работать с цепочками CNAME, и по это причине информационный ресурс, к которому обращаются по имени-синониму, будет не доступен.

Таким образом, не смотря на то, что в спецификации цепочки синонимов запрещены явным образом, в реальной жизни они могут появляться и появляются.

Наш пример показывает цепочку в одной зоне, но гораздо хуже межзонные цепочки. Межзонная цепочка CNAME плоха тем, что не отслеживает исчезновение адресной записи в другой зоне, за которую сервер, содержащий в своей зоне CNAME, не отвечает. Как итог - появление "подвешенных" CNAME записей.

Теперь вернемся к вопросу о том, что для доменного имени, которое является синонимом, может существовать только одна запись описания ресурса и эта запись - CNAME. Что происходит, когда это правило нарушается?

Первая распространенная ошибка, на которую указывают и все RFC и FAQ - использование CNAME в совокупности с NS записями. Рассмотрим пример (RFC 1912):

```
podunk.xx. IN NS ns1
IN NS ns2
IN CNAME mary
mary IN A 1.2.3.4
```

В данном случае для домена Podunk.xx. определено два сервера доменных имен ns1 и ns2, но одновременно указано, что Podunk.xx. - это синоним для канонического имени mary. Mary, ns1 и ns2 - это не полные имена. В нашем случае данное обстоятельство значения не имеет. BIND, встретив CNAME, обе записи NS проигнорирует, т.к. будет следовать соответствующим стандартам и рекомендациям.

На самом деле, можно усмотреть противоречие между тем, что было описано в алгоритме обработки CNAME записей и тем, что описано абзацем выше. Ведь при поиске NS мы найдем NS записи, и, следовательно, нам не будет выдана CNAME запись.

Все правильно, если NS записи будут загружены в память сервером доменных имен при его запуске. Но сервер проверяет при своем запуске корректность описания зоны. Он просто проигнорирует все записи, отличные от CNAME, которые содержат синоним в первом поле записи описания ресурсов.

Вот пример сообщения сервера для этого случая (BIND 9):

```
Oct 14 12:55:51 generate named[136]: dns_master_load: webstatistics.ru:21: zone.
webstatistics.ru: CNAME and other data
Oct 14 12:55:51 generate named[136]: zone webstatistics.ru/IN: loading master fi
le webstatistics.ru: CNAME and other data
```

А вот фрагмент описания зоны, на которую он ругается:

```
zone IN NS ns.wenstatistics.ru.
IN CNAME subzone
subzone IN A 144.206.192.61
```

BIND 9, найдя такую ошибку, в зоне вообще обслуживать ее не будет. Точнее говоря, если он ее уже обслуживает, то при перезагрузке (restart) он старое описание на новое в своих кэшах не заменить, а при начальном запуске его не загрузит.

На самом деле ситуация "CNAME на NS" часто встречается в том случае, когда хотят определить IP адрес для доменного имени зоны. Если быть более точным, то администратор хочет некоторому хосту в зоне присвоить то же имя, что и всему домену. Например, это нужно для обеспечения доступа к веб-серверу по именам www.kyky.ru и kyky.ru. В этом случае описание вида:

```
$TTL 3600
@ IN SOA ns.kyky.ru hostmaster.kyky.ru (
20021013 3h 30m 30d 3600 )
IN NS ns.kyky.ru.
IN NS ns.provider.ru.
IN CNAME server.kyky.ru.
server IN A 192.168.0.1
```

www IN CNAME server.kyky.ru.
ns IN CNAME server.kyky.ru.

будет ошибочным. Мы потеряем не только NS записи зоны kyky.ru, но и вообще все записи этой зоны. Правильным был бы следующий вариант:

$TTL 3600
@ IN SOA ns.kyky.ru hostmaster.kyky.ru (
20021013 3h 30m 30d 3600)
IN NS ns.kyky.ru.
IN NS ns.provider.ru.
IN A 192.168.0.1
server IN A 192.168.0.1
ns IN A 192.168.0.1
www IN CNAME kyky.ru.

В принципе, вместо "kyky.ru." в CNAME можно было бы указать и символ @.

Раз уж мы заговорили о символе "@", то следует заметить, что этот символ в поле nickname применяться не должен, т.к., фактически, тем самым утверждается, что текущее имя зоны - это синоним другого имени, и, следовательно, описание этой зоны следует проигнорировать.

Скорее всего, идея об использовании символа "@" в CNAME возникла при желании назначить имя зоны конкретному хосту, который имеет IP-адрес. Движение мысли в этом случае понятно: хост - это синоним зоны, поэтому мы и назначаем зоне IP-адрес через имя хоста. Если вдуматься, то такая логика неверна. Хост и домен - это совершенно разные понятия. Если вы хотите хосту, а точнее IP-адресу поставить в соответствие еще одно имя, то делать это следует посредством адресной записи, как в приведенном выше примере. При этом совершенно не имеет значения тот факт, что назначаемое имя - это имя зоны.

На самом деле, при исправлении описания зоны мы поправили еще одну запись. В первоначальном варианте имя ns.kyky.ru является синонимом для server.kyky.ru. Таким образом, в первоначальном варианте существовала недопустимая цепочка в рамках описания зоны, которую сервер может сам обнаружить. В новом варианте мы запись CNAME заменили на адресную запись.

Вообще говоря, немотивированные запреты всегда провоцируют на попытки попробовать программное обеспечение на соответствие этим запретам. Ведь, в конечном счете, для нашей зоны в рамках описания зоны существует адресная запись, и сервер может ее успешно найти по цепочке CNAME (см. пример в начале этого материала).

Косвенно понятно, почему введен запрет. В конце концов, число серверов корневой зоны ограничено числом 13 по одной простой причине - размеру пакета UDP. На самом деле цепочки без конца и края также упрутся в то же ограничение, ведь в пакете нужно передавать и CNAME-ы и IP-адрес. Если изменять логику обработки запросов и заставлять клиента итеративно обращаться к серверу по CNAME цепочке, то где предел такого цикла обращений.

На самом деле есть еще одна причина запрета на использование CNAME для NS и MX, но прежде, чем ее назвать и обсудить, мы рассмотрим проблемы, связанные с совместным использованием MX и CNAME.

Трудность поиска ошибок этого сорта в том, что приходится работать с двумя информационными сервисами одновременно. Ошибки описания зоны проявляются не при поиске IP-адресов или доменных имен, а при доставке электронной почты.

Ошибки совместного применения CNAME и MX заключается в том, что в записи MX синоним используют как в первом поле MX, так и в последнем поле MX. Ни первое, ни второе делать не следует.

Первый вариант:

```
$ORIGIN kyky.ru
@ IN A 192.168.0.2
kuku IN CNAME kyky.ru.
IN MX 10 kyky.ru.
```

В данном случае, мы хотим отправлять почту на kuku.kyky.ru через kyky.ru. Правило запрета существования других записей описания ресурса для синонима кроме единственной CNAME записи, которая этот синоним и вводит, заставляет сервер проигнорировать MX.

Правильным бы было:

```
$ORIGIN kyky.ru
@ IN A 192.168.0.2
IN MX 10 kyky.ru.
kuku IN CNAME kyky.ru.
```

В данном случае, если почта отправляется на kuku.kyky.ru, то по CNAME сервер доменных имен возвращает kyky.ru адресную запись для kyky.ru. Почтовый клиент соединяется с этим IP-адресом и отправляет на него почту. Другое дело настройки почтового шлюза на kyky.ru. Если он не распознает имя kuku.kyky.ru как свое собственное, либо, если у него нет правил пересылки для kuku.kyky.ru, то возникнут проблемы с доставкой почты, но это уже не проблема DNS.

Теперь другой вариант. Синоним используется в последнем поле записи MX:

```
$ORIGIN kyky.ru
@ IN A 192.168.0.2
IN MX 10 kuku.kyky.ru.
kuku IN CNAME kyky.ru.
```

Этот случай аналогичен случаю использования в качестве имени сервера в NS записи синонима, заданного через CNAME. По этой причине мы сейчас возобновим обсуждение проблемы, начатое в части, касающейся NS записей.

В основе запрета на использование синонима в правой части записей MX и NS лежит процедура обработки запросов на MX и NS. Все дело в том, что в качестве ответа на запрос типа NS или MX сервер в основной секции данных отклика возвращает доменной имя сервера доменных имен, либо, если запрос на запись MX, доменное имя почтового шлюза, а в дополнительной секции отклика соответствующие адресные записи. CNAME запись никогда не может появиться в дополнительной секции отклика сервера доменных имен.

Следовательно, клиент, который попадает при поиске MX или NS на синоним Должен инициировать дополнительный запрос адресной записи. Особенность почтовых систем такова, что они такого запроса в большинстве случаев не инициируют. Как следствие - почта не может быть доставлена, если речь идет о MX записи.

Вот пример с записью NS:

```
> set debug
> set q=ns
> webstatistics.ru.
Server: [144.206.192.60]
Address: 144.206.192.60

;; res_nmkquery(QUERY, webstatistics.ru, IN, NS)
------------
Got answer:
HEADER:
opcode = QUERY, id = 41793, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 2, authority records = 0, additional = 1

QUESTIONS:
webstatistics.ru, type = NS, class = IN
ANSWERS:
-> webstatistics.ru
nameserver = ns.webstatistics.ru
ttl = 3600 (1H)
-> webstatistics.ru
nameserver = ns4.nic.ru
ttl = 3600 (1H)
ADDITIONAL RECORDS:
-> ns4.nic.ru
internet address = 194.226.96.8
ttl = 86297 (23h58m17s)

------------
webstatistics.ru
nameserver = ns.webstatistics.ru
ttl = 3600 (1H)
webstatistics.ru
nameserver = ns4.nic.ru
ttl = 3600 (1H)
ns4.nic.ru
internet address = 194.226.96.8
ttl = 86297 (23h58m17s)
>
```

Мы запрашиваем NS для зоны webstatistics.ru. В качестве ответа получаем доменные имена серверов доменных имен, но ns.webstatistics.ru - это синоним для webstatistics.ru, поэтому его адресной записи в дополнительной секции отклика сервера доменных имен нет. Ниже представлен пример описания этой зоны:

$ORIGIN ru.
webstatistics 3600 IN SOA ns.webstatistics.ru. hostmaster.webstatistics.ru. (
1 3600 600 86400 3600)
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
$ORIGIN webstatistics.ru.
ns IN CNAME @
$ORIGIN nic.ru.
ns4 IN A 194.226.96.8

Любопытно, что при запуске BIND 9-ой версии не сообщил о некорректном применении CNAME.

Считается, что гораздо проще заставить администраторов соблюдать правила использования CNAME записей, чем пытаться исправить ошибки администратора за счет "умного" программного обеспечения.

На самом деле ошибка может проистекать из-за обычной невнимательности администратора.

Приведем пример правильного и неправильного использования записи CNAME:

$ORIGIN polyn.net.kiae.su.
olga IN A 144.206.192.2
IN MX 0 olga
IN MX 10 ns.polyn.kiae.su.
www IN CNAME olga.polyn.kiae.su.
gopher IN CNAME olga.polyn.kiae.su.

В данном случае записи типа CNAME указаны правильно, т.к. не мешают переадресации почты на машину olga.polyn.kiae.su. Но если их поменять местами с записями типа MX, то скорее всего почта работать не будет:

$ORIGIN polyn.net.kiae.su.
olga IN A 144.206.192.2
www IN CNAME olga.polyn.kiae.su.
gopher IN CNAME olga.polyn.kiae.su.
IN MX 0 olga
IN MX 10 ns.polyn.kiae.su.

В данном случае можно предположить, что при редактировании зоны администратор просто неаккуратно скопировал блоки записей. Результат - неправильное применение CNAME.

Теперь от грустного - ошибок, перейдем к особенностям применения CNAME. Случай, когда CNAME используется для простого определения синонимов, мы уже рассматривали. Теперь коснемся такого приема, как Round Robin алгоритм.

Обычно циклическую перестановку адресов рассматривают в контексте адресных записей и позиционируют как один из способов распределения нагрузки. На самом деле BIND можно настроить таким образом, что он позволит "тасовать" не только адресные записи. В

частности это разрешено будет делать и для CNAME записей. При этом следует помнить, что строго говоря такое применение CNAME запрещено спецификациями.

Вот пример множественных адресных записей:

www.domain.ru. IN A 192.168.0.1
www.domain.ru. IN A 192.168.0.2
www.domain.ru. IN A 192.168.0.3

А вот его аналог для записей типа CNAME:

Server1.domain.ru. IN A 192.168.0.1
Server1.domain.ru. IN A 192.168.0.2
Server1.domain.ru. IN A 192.168.0.1
www.domain.ru. IN CNAME server1.domain.ru.
www.domain.ru. IN CNAME server2.domain.ru.
www.domain.ru. IN CNAME server3.domain.ru.

На первый взгляд большой разницы в том, что тасовать (адресные записи или CNAME записи), нет. И в том и в другом случае адреса ресурса циклически переставляются. На один запрос мы получаем один адрес, на другой - второй, и так далее по кругу.

Разница заключается в том, что на каждый запрос в первом случае выдается список IP-адресов, где они все перечисляются, только порядок их следования в отклике меняется. В этом случае клиент может теоретически перебрать их все, используя только одно обращение к DNS.

Во втором случае в качестве отклика возвращаются CNAME записи. Это значит, что клиент в конечном итоге получает не список IP-адресов, а только один IP-адрес.

На самом деле рекомендуется все-таки не использовать "тасование" CNAME, тем паче, что в BIND 9 на выше приведенный пример будет получен при запуске сервера следующий отклик в логах:

Oct 13 18:03:15 generate named[136]: dns_master_load: webstatistics.ru:19: www.domain.ru: multiple RRs of singleton type

Любопытно, что обслуживаться запросы по синониму будут. Просто из всех записей одного синонима будет выбрана последняя. Если же мы имеем дело со множественными адресными записями, и на их доменное имя существует синоним, то для запроса на этот синоним будет выдан список всех IP-адресов.

Ниже приведем пример описанного только что случая:

$ORIGIN ru.
Webstatistics 3600 IN SOA webstatistics.ru. paul.webstatistics.ru. (
1 3600 600 86400 3600)
3600 IN NS ns.webstatistics.ru.
3600 IN NS ns4.nic.ru.
IN A 144.206.192.60
$ORIGIN webstatistics.ru.
ns 3600 IN A 144.206.192.60
$ORIGIN nic.ru.

ns4 IN A 194.226.96.8
$ORIGIN webstatistics.ru.
www 3 IN A 144.206.192.60
www 3 IN A 144.206.192.61
www 3 IN A 144.206.192.62
www 3 IN A 144.206.160.32
www1 IN CNAME ns.webstatistics.ru.
www1 IN CNAME www

В данном случае BIND версии 9 игнорирует первую запись CNAME, выдает выше приведенное сообщение в логе и поддерживает только последний CNAME.

Если теперь обратиться за IP-адресом www1.webstatistics.ru, то мы получим следующее (тестирование программой nslookup):

> set debug
> www1.webstatistics.ru.
Server: [144.206.192.60]
Address: 144.206.192.60

;; res_nmkquery(QUERY, www1.webstatistics.ru, IN, A)

Got answer:
HEADER:
opcode = QUERY, id = 33822, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 5, authority records = 2, additional = 2

QUESTIONS:
www1.webstatistics.ru, type = A, class = IN
ANSWERS:
-> www1.webstatistics.ru
canonical name = www.webstatistics.ru
ttl = 3 (3S)
-> www.webstatistics.ru
internet address = 144.206.192.60
ttl = 3 (3S)
-> www.webstatistics.ru
internet address = 144.206.192.61
ttl = 3 (3S)
-> www.webstatistics.ru
internet address = 144.206.192.62
ttl = 3 (3S)
-> www.webstatistics.ru
internet address = 144.206.160.32
ttl = 3 (3S)
AUTHORITY RECORDS:
-> webstatistics.ru
nameserver = ns.webstatistics.ru
ttl = 3600 (1H)
-> webstatistics.ru
nameserver = ns4.nic.ru
ttl = 3600 (1H)

ADDITIONAL RECORDS:
-> ns.webstatistics.ru
internet address = 144.206.192.60
ttl = 3600 (1H)
-> ns4.nic.ru
internet address = 194.226.96.8
ttl = 79606 (22h6m46s)

Name: www.webstatistics.ru
Addresses: 144.206.192.60, 144.206.192.61, 144.206.192.62, 144.206.160.32
Aliases: www1.webstatistics.ru

>

На запрос адреса мы в основной секции отклика получаем CNAME и все адресные записи, в авторитативной секции получаем доменные имена серверов зоны (записи NS), а в дополнительной секции IP-адреса этих серверов доменных имен.

И в заключении о реальном масштабе ошибок, связанных с применением CNAME в системе доменных имен. Согласно исследованиям компании Men & Mice проведенным на 5000 случайно выбранных зонах домена com в августе 2002 года в 3.3%-ах случаев была зафиксирована ссылка в MX записи на синоним, т.е. неправильное совместное применение CNAME и MX.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
4. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
5. D. Barr. RFC 1912. Common DNS Operational and Configuration Errors. 1996. (http://www.ietf.org/rfc/rfc1912.txt?number=1912)
6. R. Elz, R. Bush. RFC 2181. Clarifications to the DNS Specificatrion. 1997. (http://www.ietf.org/rfc/rfc2181.txt?number=2181)

Полезные ссылки:

1. http://www.rscott.org/dns/cname.html - о способах верификации ошибок в описании зоны, относящихся к записям CNAME.
2. http://www.adminschoice.com/docs/dns_trouble_shooting.htm - пример правильного и неправильного употребления NS и CNAME.
3. http://support.microsoft.com/default.aspx?scid=KB;EN-US;q159310& - проблемы совмести dns.exe и BIND при обработке откликов NS и CNAME.
4. http://cr.yp.to/im/canme.html - обсуждается проблема работы с DNS программ sendmail и qmail. Основное внимание уделено работе с некорректными CNAME - записями.

5. http://www.acmebw.com/askmrdns/ - FAQ по системе доменных имен. На этот архив ссылаются и разработчики BIND. Вопросам некорректного использования CNAME посвящено около десятка страниц.
6. http://www.menandmice.com/6000/61_recent_survey.html - статистика ошибок при конфигурации серверов системы доменных имен.

7. g. Файлы описания зоны. Директивы управления.

В этом материале мы разберем особенности применения директив управления при описании зон в master файлах. Кроме стандартных директив $ORIGIN и $INCLUDE будут также рассмотрены $TTL и $GENERATE.

Наиболее используемая директива управления - **$ORIGIN**. Она позволяет определять текущее имя домена. Поясним что это такое. При описании файла конфигурации named в качестве первого параметра директив primary, secondary (файл конфигурации BIND 4) и директивы zone (файл конфигурации BIND 8 или 9) указывается имя зоны, например:

```
#
# файл настройки для BIND 4
#
directory /etc/namedb
primary kyky.ru kuku.ry
secondary first.kyky.ru 192.168.0.1 first.kuku.ru

или:

/*
файл настройки для BIND 9
*/
options {
directory "/etc/namedb";
};
zone "kyky.ru" in {
type master;
file "kyky.ru";
};
zone "first.kyky.ru" in {
type slave;
file "first.kyky.ru";
masters {192.168.0.1;};
};
```

В обоих примерах в качестве зоны ответственности для master сервера указана зона kyky.ru, а для slave - first.kyky.ru.

Теперь, если перейти к master файлу зоны kyky.ru, т.е. файлу описания зоны kyky.ru, то по умолчанию в качестве текущего доменного имени будет предполагаться имя kyky.ru. Если быть абсолютно точным, то это имя будет содержаться в переменной @. Именно поэтому файл описания зоны kyky.ru начинается записью вида:

```
@ IN SOA ns.kyky.ru. adm.kuku.ru. (
1 2h 30m 2d 30m )
IN NS ns.kyky.ru.
ns IN A 192.168.0.2
```

Таким образом, записи SOA и NS в данном случае относятся к зоне kyky.ru. А адресная запись определяет IP-адрес для хоста ns.kyky.ru.

Что делать, если нужно в описание зоны включить запись описания хоста из другой зоны, например из зоны делегированной данным сервером другому серверу:

```
@ IN SOA ns.kyky.ru. adm.kuku.ru. (
1 2h 30m 2d 30m )
IN NS ns.kyky.ru.
ns IN A 192.168.0.2
sub IN NS first.kyky.ru.
first IN A 192.168.0.2
$ORIGIN sub.kyky.ru.
ns IN A 192.168.0.45
```

В этом примере директива $ORIGIN переопределяет имя текущего домена с kyky.ru на sub.kyky.ru.

Можно было бы просто указать полное доменное имя для хоста с адресом 192.168.0.45 - ns.sub.kyky.ru, но это хорошо тогда, когда мы определяем один хост, но довольно утомительно набивать полные имена, если хостов будет много. При этом всегда нужно помнить о точке на конце полного имени J.

Вообще говоря, основное назначение $ORIGIN - это упрощение содержания файла описания зоны при определении поддоменов в той же зоне, где описан и выше стоящий домен:

```
$ORIGIN kyky.ru.
ns IN A 192.168.0.1
www IN A 192.168.0.2
$ORIGIN sub1.kyky.ru.
host1 IN A 192.168.0.3
host2 IN A 192.168.0.4
$ORIGIN sub2.kyky.ru.
host1 IN A 192.168.0.5
host2 IN A 192.168.0.6
```

В приведенном выше примере мы завели два поддомена в домене kyky.ru, при чем именование машин в них одинаковое, но полные доменные имена различаются, что и не удивительно.

Еще несколько лет назад довольно часто можно было видеть в файлах описания зон следующую картину:

```
@ IN SOA ns.kyky.ru. adm.kuku.ru. (
1 2h 30m 2d 30m )
IN NS ns.kyky.ru.
IN NS ns.provider.ru.
```

ns IN A 192.168.0.2
$ORIGIN provider.ru.
ns IN A 192.168.10.5

Что тут не так? Формально все верно. Для зоны kyky.ru должно быть определено два сервера доменных имен с независимым подключением к сети, т.к. речь идет о корпоративном домене.

Первый сервер определен в сети компании и имеет имя ns.kyky.ru, а второй сервер - это сервер провайдера, который обеспечивает дублирование, т.е. ns.kyky.ru - это master, а ns.provider.ru - это slave.

Вся штука в том, что имя ns.provider.ru не принадлежит домену kyky.ru. На самом деле наш сервер не является авторитативным для зоны provider.ru, поэтому он не вправе отвечать на запросы к этой зоне.

Тем не менее, существует процедура делегирования зон ответственности, и там возникает ситуация, когда IP-адрес сервера доменных имен нельзя получить иначе, как из зоны, в которой описано делегирование.

В этом случае отклик нашего сервера называют рефералом (refferal) и он содержит IP-адреса серверов доменных имен в качестве дополнительной информации. Эти адресные записи принято называть присоединенными (glue - буквально "приклеенные").

Так вот, присоединять адресную запись для ns.provider.ru нет смысла, т.к. ее можно найти в другой зоне. Более того, там она может оказаться более свежей, чем у нас, и указывающей совсем на другой адрес. Присоединяют только те записи, информацию о которых иначе получить просто нельзя, например, адреса серверов доменных имен, чьи имена находятся в делегируемой нами зоне.

Новые версии BIND (4.9 и выше) отслеживают этот момент, а вот более старые версии такой проверки не делают. Администраторы системы доменных имен включали такие "приклеенные" записи для того, чтобы сэкономить на трафике (нет нужды опрашивать серверы доменных имен).

О "приклеенных" записях подробно поговорим позже при описании адресных и NS записей.

Директива **$INCLUDE** используется для того, чтобы в файл описания зоны можно было включить содержание другого файла. Так рекомендуется поступать при описании больших зон, разбивая их на небольшие фрагменты. При этом имя включаемого файла должно быть описано либо полностью от корня файловой системы, либо оно будет привязано к директории, указанной в файле named.boot или named.conf.

$INCLUDE my_zone.dsc

В этом случае используется файл /etc/namedb/my_zone.dsc, т.к. обычно директивы directory (BIND 4) или directory (BIND 8-9) определяют именно эту директорию для хранения файлов базы данных named.

$INCLUDE /etc/my_zone

В данном случае указан полный путь, и именно он будет использоваться при доступе к файлу.

Рассмотрим теперь еще две директивы управления, которые сравнительно недавно появились в файлах описания зон: $TTL и $GENERATE.

Директива **$TTL** определяет время хранения записи описания ресурсов в кэше resolver, а если resolver "глупый" (stub), т.е. не имеет кэша и посылает только рекурсивные запросы, то $TTL будет определять время хранения RR в кэше локального сервера доменных имен, который выполняет запросы "глупого" resolver.

Данная директива была введена в RFC-2308 для того, чтобы освободить последний параметр записи SOA для времени кэширования отрицательных ответов (negative caching).

Директиву $TTL следует указывать прямо перед записью SOA в файле описания зоны. $TTL связана с определением 32-битового значения, поэтому может принимать значения от 0 до 2147483647(RFC 2181).

Вообще говоря, администратор серверов BIND версий 8 или 9 имеет возможность определять максимальное время кэширования для записи описания ресурса. Делается это в файле конфигурации named.conf. По умолчанию это время равно одной неделе.

В материале "Описание зоны. Формат записи описания ресурсов (RR)" было указано, что при переходе со старых версий BIND на новые не нужно изменять файлы описания зон, т.к. формат RR не изменился. Как теперь понятно, это не совсем так. Во всяком случае, изменилось назначение одного из атрибутов SOA и, как следствие стал необходима директива управления $TTL.

Конечно, разработчики BIND понимают, что в сети стоит большое число серверов, о которых администраторы просто забыли, а поэтому в BIND заложены разумные значения времена кэширования (TTL) по умолчанию.

Директива $GENERATE призвана упростить процесс создания фала описания зоны. Ее главное назначение заключается в том, чтобы сократить число регулярных записей, которые отличаются друг от друга одним или несколькими символами. Например, если стандартный набор записей в фале описания зоны выглядит, как:

```
host1 IN A 192.168.0.1
host2 IN A 192.168.0.2
…
host10 IN A 192.168.0.10
```

То запись с использованием $GENERATE будет выглядеть примерно так:

```
$GENERATE 1-10 $ IN A 192.168.0.$
```

Таким образом, 10 записей мы заменили одной.

Очень полезна данная запись в случае, когда нужно определять обратную зону для сети класса C (в нотации CIDR маскированы первые 24 бита), которая разбита провайдером на подсети, и эти подсети розданы различным клиентам (см. RFC 2317). Оставим саму проблему за кадром до того момента, когда нам действительно понадобится делегировать

подобного рода обратные зоны, а здесь приведем только ту часть файла описания "обратной" зоны, в которой применяется директива управления $GENERATE:

> *$GENERATE 1-63 $.0.168.192.in-addr.arpa IN CNAME $.0-63.0.168.192.in-addr.arpa.*
> *0-63.0.168.192.in-addr.arpa. IN NS ns.kyky.ru.*
>
> *$GENERATE 65-127 $.0.168.192.in-addr.arpa IN CNAME $.64-128.0.168.192.in-addr.arpa.*
> *64-128.0.168.192.in-addr.arpa. IN NS ns.corp.ru.*

В данном случае $GENERATE используется для производства однотипных RR.

Следует отметить, что $GENERATE это не стандартная директива, а расширение директив управления пакета BIND.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. H. Eidnes, G. de Groot, P. Vixie. RFC 2317. Classless IN-ADDR.ARPA delegation. 1998. (http://www.ietf.org/rfc/rfc2317.txt?number=2317)
4. M. Andrews. RFC 2308. Negative Caching of DNS Queries (DNS NCACHE). 1998.(http://www.ietf.org/rfc/rfc2308.txt?number=2308)
5. R. Elz, R. Bush. RFC 2181. Clarifications to the DNS Specification. 1997. (http://www.ietf.org/rfc/rfc2181.txt?number=2181)
6. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. http://www.ispras.ru/~grn/dns/index.html - Г.В. Ключников. Служба доменных имен (Domain Name System). 1999. На самом деле, это отличная компиляция приведенных в конце книжки первоисточников. Примеры взяты из этих же первоисточников. Очень качественный перевод и грамотно скомпонованный текст.
2. http://public.pacbell.net/dedicated/cidr.html - Classless Inter-Domain Routing (CIDR) Overview. Этот обзор позволяет получить представление о CIDR - концепции, которая пришла на смену маршрутизации на основе классов IP-сетей.

8. Типовые примеры описания зон и файлов конфигурации BIND

В данном материале мы даем короткий обзор наиболее типичных случаев, которые встречаются при организации работы с системой доменных имен. При этом мы опираемся и на потребности обычных пользователей, и на задачи, которые решают администраторы зон системы DNS.

Архитектура "клиент-сервер", положенная в основу системы доменных имен заставляет сетевых администраторов решать задачи взаимодействия с DNS как со стороны обычных пользователей, так и с обратной стороны - точки зрения администраторов DNS.

Когда речь заходит об организации поиска информационных ресурсов по их доменному имени, то сетевой администратор должен обеспечить для своих пользователей быстрый поиск IP-адресов по доменным именам в "прямых зонах" и поиск доменных имен по IP-адресам в "обратных" зонах.

Типовым решением данной задачи является организация кэширующего сервера доменных имен, который должен обслуживать рекурсивные запросы к DNS с группы корпоративных хостов. Настройки данного типа сервера мы рассмотрим в материале "Настройка кэширующего локального сервера доменных имен".

Современная особенность решения этой задачи состоит в том, чтобы сократить до необходимого минимума число обслуживаемых хостов и не порождать избыточного DNS трафика, а также избежать проблем с безопасностью.

Вообще говоря, под "корпоративными хостами" мы понимаем некоторую группу хостов, которую обслуживает администратор DNS. Это могут быть хосты компании, вуза, министерства, отдела научно-исследовательского института и т.п. Эти хосты мы называем "корпоративными" по аналогии с "корпоративными доменами". Последнее словосочетание принято применять для доменов второго уровня, принадлежащим организациям.

Второй круг проблем возникает в том случае, когда сетевой администратор реально отвечает за сервер, который является авторитативным для зоны. Это означает, что мы имеем дело либо с master сервером, либо со slave сервером зоны.

В большинстве случаев настройка master сервера зоны просто расширяет настройки кэширующего сервера. Строго говоря, это справедливо только для BIND, да и то только в тех случаях, когда безопасность системы не ставится на первое место.

Простейший переход от кэширующего сервера к master серверу зоны мы разберем в материале "Небольшой корпоративный домен в зоне ru". При этом мы постараемся указать на возможность повышения безопасности эксплуатации сервера.

Поддержка slave сервера еще проще, чем поддержка master. В этом случае не нужно самому прописывать информацию в файлы описания зоны. Зона просто скачивается с master сервера. Этот вариант настройки сервера мы рассмотрим в материале "Настройка slave сервера для корпоративного домена".

Кроме перечисленных случаев интерес представляют случай размещения зоны на двух сетях класса C (в нотации CIDR x.x.x.x/24) и размещение зоны на подсети класса C.

Особенности работы с такого сорта доменами заключены в организации "обратных" зон. Этим вопросам будут посвящены материалы: "Описание "прямой" и "обратной" зон для поддомена определенного на двух подсетях типа /24" и "Делегирование обратных зон для подсетей сетей сети класса С (/24)"

Отдельно следует рассмотреть вопрос делегирования поддоменов (материал "Делегирование поддомена внутри корпоративного домена"). Это связано с тем, что некорректное делегирование является одной из самых распространенных ошибок, которые встречаются в системе DNS.

Проблемы с вязанные с повышением безопасности, организацией "расщепленных" доменов, работа из-под firewall (защищенных сегментов сети), настройкой динамического обновления описания зон и уведомлений об этих изменениях мы рассмотри несколько позже.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. http://www.fima.net/bind-8.html - это очень коротко про настройку BIND 8.x.x. На самом деле, прежде, чем это читать, нужно попрактиковаться с named. Но, в принципе С.Минаков снабдил всех желающих исчерповующей справочной информацией.

8. а. Настройка кэширующего сервера доменных имен. Примеры описания зон и файлов конфигурации BIND. Запуск и проверка работоспособности.

В данном материале мы рассмотрим настройку сервера доменных имен, который выполняет рекурсивные запросы и функции кэширования записей описания ресурсов DNS для группы хостов. При этом данный сервер не будет поддерживать никакой зоны, т.е. он не будет авторитативным для какой-либо зоны пространства доменных имен Интернет.

Если рассматривать функции кэширующего сервера с точки зрения архитектуры DNS, описанной в RFC-1034, то это "умный" resolver. А точнее, "умный" независимый resolver, который выполняет рекурсивные запросы прикладных программ.

Очевидно, что при такой постановке вопроса ответственность за какую-либо зону на наш сервер не возлагается. Его главное назначение - принимать запросы от клиентов и опрашивать другие (авторитативные) серверы доменных имен на предмет поиска необходимой клиентам информации.

Второй основной задачей такого сервера является хранение найденной информации (записей описания ресурсов) к своем кэше в течение времени жизни этой информации. При этом в кэш в соответствии с RFC-1034 должна попадать не только информация о конечном соответствии IP-адреса и доменного имени, но и информация найденная в процессе поиска конечного соответствия, например, информация об авторитативных серверах зон доменных имен.

Мы будем рассматривать настройку такого типа сервера на примере программного обеспечения BIND, т.к. большинство серверов, которые в настоящее время установлены в сети - это серверы BIND, хотя эта точка зрения и горячо оспаривается профессором Бернштейном (автор djbdns) во всех конференциях и списках, касающихся системы DNS. По крайней мере, BIND поставляется в комплекте со свободно распространяемыми клонами Unix, что делает знакомство с ним необходимым для большинства сетевых администраторов.

Для того, чтобы настроить BIND в качестве кэширующего сервера необходимо создать и заполнить соответствующей информацией следующие файлы:

- named.boot (для BIND 4.x);
- named.conf (для BIND 8.x и BIND 9.x);
- root.cache (для BIND 4.x);
- описание серверов корневой зоны (зона типа hint для BIND 8.x и BIND 9.x);
- описание зоны 127.in-addr.arpa.

Будем рассматривать эти настройки по версиям BIND, и начнем с BIND 4.x, хотя она и является очевидным атавизмом.

Настройка кеширующего сервера в BIND 4.x

Сначала следует прописать правильным образом информацию в файле настройки самого сервера - named.boot, который располагается в каталоге /etc:

```
directory /etc/namedb
primary 127.in-addr.arpa 127.in-addr.arpa
cache . named.root
```

Первая строка этого файла определяет каталог, в котором хранятся описания зон. На первый взгляд нам эти описания не нужны, ведь мы настраиваем кэширующий сервер доменных имен, но при более детальном рассмотрении каталог нужен, т.к. наш сервер будет авторитативным для одной обратной зоны - 127.in-addr.arpa.

Именно этот факт и определяет вторая строка файла конфигурации сервера, в которой указана директива primary, определяющая, что сервер является master (в терминологии BIND 4.x - primary) для зоны 127.in-addr.arpa. Имя зоны задано вторым параметром директивы primary. Третий параметр этой директивы определяет имя файла в каталоге /etc/namedb, где описана соответствующая директиве зона.

Имя файла описания этой зоны может быть любым. Например, в последнее время модно называть его localhost.rev. Собственно, это имя генерирует скрипт make-localhost, который автоматически составляет описания зоны локального хоста.

Последняя директива (cache) определяет список серверов, обслуживающих корневой домен. В настоящее время никакой другой функции у этого файла нет, хотя по названию можно было бы предположить, что его назначение гораздо шире. В настоящее время этот файл используется только для получения адресов корневых серверов и загрузки с одного из них (обычно тот, что быстрее откликнется, а откликается первым a-сервер, т.к. его первым спрашивают) свежего списка серверов, ответственных за корневую зону.

Теперь посмотрим на содержание файлов описанных в named.boot и расположенных по умолчанию в каталоге /etc/namedb. Если они расположены в другом месте, то это указывается в директиве directory файла named.boot.

Содержание 127.in-addr.arpa:

```
; From: @(#)localhost.rev 5.1 (Berkeley) 6/30/90
; $Id: PROTO.localhost.rev,v 1.1 1995/03/21 16:33:44 wollman Exp $
;
; This file is automatically edited by the `make-localhost' script in
; the /etc/namedb directory.
;

@ IN SOA iris.polyn.kiae.su. root.iris.polyn.kiae.su. (
961015 ; Serial
3600 ; Refresh
300 ; Retry
3600000 ; Expire
3600 ) ; Minimum
IN NS iris.polyn.kiae.su.
1 IN PTR localhost.polyn.kiae.su.
```

Все, что начинается с ";" - это комментарии. Имя авторитативного сервера, указанное сразу после SOA берется скриптом генерации этого файла из команды hostname. Почтовый адрес - это адрес администратора сервера. NS запись снова ссылается на имя хоста, на котором запущен сервер и последняя запись - это PTR (pointer) на доменное имя localhost.polyn.kiae.su. Естественно, что, если в зоне polyn.kiae.su есть еще один хост с сервером доменных имен, то там тоже будет такая же обратная зона, с точно такой же записью, но ссылаться в реальности такая запись будет совсем на другой физический хост.

Вообще-то, в "прямой" зоне можно "в лоб" прописать некоторый IP-адрес для localhost.polyn.kiae.su. Это совершенно не повлияет работоспособность системы. Но в данном тексте мы "прямые" зоны не рассматриваем, т.к. описываем только кэширующий сервер.

Содержание файла named.root:

```
; This file holds the information on root name servers needed to
; initialize cache of Internet domain name servers
; (e.g. reference this file in the "cache . "
; configuration file of BIND domain name servers).
;
; This file is made available by InterNIC registration services
; under anonymous FTP as
; file /domain/named.root
; on server FTP.RS.INTERNIC.NET
```

```
; -OR- under Gopher at RS.INTERNIC.NET
; under menu InterNIC Registration Services (NSI)
; submenu InterNIC Registration Archives
; file named.root
;
; last update: Nov 8, 1995
; related version of root zone: 1995110800
;
;
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
;
; formerly C.PSI.NET
;
. 3600000 NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 A 192.33.4.12
;
; formerly TERP.UMD.EDU
;
. 3600000 NS D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. 3600000 A 128.8.10.90
;
; formerly NS.NASA.GOV
;
. 3600000 NS E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET. 3600000 A 192.203.230.10
;
; formerly NS.ISC.ORG
;
. 3600000 NS F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET. 3600000 A 192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
. 3600000 NS G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET. 3600000 A 192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
. 3600000 NS H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET. 3600000 A 128.63.2.53
;
; formerly NIC.NORDU.NET
;
. 3600000 NS I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET. 3600000 A 192.36.148.17
; End of File
```

Мы специально включили именно эту версию файла named.root, которая шла с дистрибутивом BIND 4.9.6 для операционной системы IRIX 6.5. При реальной работе с BIND следует и версию named поставить поновее и файл описания корневой зоны получить "свежий".

Теперь мы имеем все необходимые файлы для запуска кэширующего сервера. Запускаем сервер:

```
>named
```

и все должно работать.

Настройка кэширующего сервера в BIND 8.x и BIND 9.x

Отличие данного случая от случая BIND 4.x, описанного выше, заключается в том, что вместо named.boot мы должны создать файл named.conf, который имеет совсем другой формат управляющих директив. Вот пример его содержания:

```
options {
directory "/etc/namedb";
};
zone "." in {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
type master;
file "127.in-addr.arpa";
};
```

Это содержание файла named.conf полностью соответствует конфигурации BIND 4.x, которую мы разбирали выше.

Директива options определяет каталог, в котором хранятся файлы описания зон, директива zone определяет зоны, которые поддерживает сервер.

Зона "." сервером не поддерживается. Это корневая зона. Поэтому она имеет тип hint, т.е. "подсказка" на то, где описаны серверы корневой зоны.

Зона "0.0.127.in-addr.arpa" имеет тип master, т.к. данный сервер действительно является мастером для этой зоны.

Заметим, что по умолчанию в BIND 8.x и BIND 9.x файл named.conf расположен в разных каталогах. В BIND 8.x в каталоге /etc/namedb, а в BIND 9.x в каталоге /etc. Но, в принципе, используя параметры командной строки named этот порядок можно переиграть. Тем более его можно переиначить при сборке сервера из исходных текстов.

Теперь можно запускать named:

```
>named
```

Для BIND 8 сервер действительно запуститься, а вот для BIND 9.x этого скорее всего не произойдет. В файле системного журнала будет получено сообщение вида:

```
Nov 20 13:49:36 quest named[34123]: starting BIND 9.2.1
Nov 20 13:49:36 quest named[34123]: none:0: open: /etc/rndc.key: file not found
Nov 20 13:49:36 quest named[34123]: couldn't add command channel 127.0.0.1#953:
file not found
```

Для того, чтобы такое сообщение не появлялось нужно отменить удаленное управление сервером, которое обычно осуществляют через программу rndc:

```
options {
directory "/etc/namedb";
};
controls {};
zone "." in {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
type master;
file "127.in-addr.arpa";
};
```

Мы просто вставили директиву controls с пустым содержанием.

Вообще говоря, лучше создать требуемые файлы ключей и работать через программу rndc, но в контексте данного материала, когда мы рассматриваем последовательно BIND 4.x - BIND 8.x - BIND 9.x, отмена контроля выглядит вполне логично.

На самом деле, конечно, такая простая конфигурация сервера не является хорошим стилем. Во-первых, мы создаем кэширующий сервер не для всех желающих, а только для хостов, которые мы обслуживаем, во-вторых, желательно не сообщать о том, что за версия сервера у нас установлена, в-третьих, существует еще несколько мелочей, которые желательно указать в файле настройки named.

Для того, чтобы отделить весь мир от "наших" хостов нужно создать список доступа:

```
acl "our_folks" { 192.168.1.0/24; };
options {
directory "/etc/namedb";
allow-recursion { "our_folks"; };
allow-query {"our_folks"}
version "unknown";
fake-iquery no;
};
controls {};
zone "." in {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
type master;
```

```
file "127.in-addr.arpa";
};
```

В данном файле конфигурации мы разрешили рекурсию (рекурсивные запросы) с нашей локальной сети, кроме того, разрешили какие-либо запросы (не только рекурсивные) тоже только с нашей локальной сети, скрыли версию named и отменили явно инверсные запросы (fake-iquery).

Вообще говоря, в данном случае можно было не вводить ограничения рекурсию, а ограничить только саму возможность обработки запросов (allow-query). Рекурсивные запросы - это только одно из подмножеств запросов и оно автоматически попадает на наложенное на обработку запросов ограничение.

Запрет на инверсные запросы - это тоже избыточная директива. Современные серверы этот тип запросов не поддерживают, а только корректно сообщают о таком своем поведении.

На самом деле в рекомендациях CERT (Computer Emergency Responds Team) есть еще несколько моментов, которые позволяют ограничить себя от разного сорта неприятностей с кэширующим сервером доменных имен.

В первую очередь это касается спуфинга, т.е. подмены отклика авторитативного сервера доменных имен поддельными пакетами в ответ на реальный запрос. Для этого необходимо, чтобы идентификаторы пакетов генерировались случайным образом.

В BIND 9.x это реализовано по умолчанию, а в BIND 8.x для этой цели нужно вставить опцию use-id-pool:

```
acl "our_folks" { 192.168.1.0/24; };
options {
directory "/etc/namedb";
allow-recursion { "our_folks"; };
allow-query {"our_folks"}
version "unknown";
fake-iquery no;
use-id-pool yes;
};
controls {};
zone "." in {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" in {
type master;
file "127.in-addr.arpa";
};
```

Ограничение на исполнение рекурсивных запросов также повышает безопасность работы, т.к. не позволяет провоцировать сервер на запросы к зонам, находящимся под контролем злоумышленников, которые могут пытаться проделывать всякие предосудительные действия, например, "отравление" кэша вашего сервера.

В пакете BIND 9.x в комплект поставки входит легковесный resolver. Его функциональность не соответствует функциональности кэширующего сервера. Легковесный resolver может выполнять только рекурсивные запросы программного обеспечения, которое функционирует на том же самом хосте, где этот resolver запущен. Он не может обслуживать группу хостов.

Проверка работоспособности кэширующего named.

Первое, что нужно выполнить после запуска сервера, посмотреть, появился ли он в списке исполняемых процессов:

```
polyn: {26} ps -ax | grep named
74541 ?? Ss 0:13.95 /usr/sbin/named
polyn: {27}
```

Если процесса нет, то нужно обратиться к файлу системного журнала и посмотреть, почему сервер не запустился:

```
Nov 20 13:48:09 quest named[34110]: starting BIND 9.2.1
Nov 20 13:48:09 quest named[34110]: /etc/named.conf:4: missing ';' before 'zone'
Nov 20 13:48:09 quest named[34110]: loading configuration: failure
Nov 20 13:48:09 quest named[34110]: exiting (due to fatal error)
```

Например, в данном случае при запуске были обнаружены ошибки в синтаксисе файла named.conf. По этой причине BIND 9.2.1 не запустился.

Если сервер запустился, то можно обратиться к нему при помощи команды dig или другой команды проверки работоспособности DNS:

```
generate# dig @localhost -t txt -c chaos version.bind.

; <<>> DiG 8.3 <<>> @localhost -t -c version.bind.
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;; version.bind, type = TXT, class = CHAOS

;; ANSWER SECTION:
version.bind. 0S CHAOS TXT "unknown"

;; Total query time: 1 msec
;; FROM: generate.polyn.kiae.su to SERVER: localhost 127.0.0.1
;; WHEN: Wed Nov 20 19:29:17 2002
;; MSG SIZE sent: 30 rcvd: 51

generate#
```

Можно также попросить статистику его работы. Для сервера BIND 4.x в его работу нужно просто вмешаться сигналом ABRT:

```
IRIS 1# ps -ae | grep named
204 ? 36:00 named
IRIS 2# kill -ABRT 204
IRIS 3#more /var/tmp/named.stats
+++ Statistics Dump +++ (1037807339) Wed Nov 20 18:48:59 2002
2434023 time since boot (secs)
2434023 time since reset (secs)
12427 Unknown query types
518901 A queries
147 NS queries
33 SOA queries
177315 PTR queries
47964 MX queries
92 TXT queries
190 AAAA queries
60 type 33 queries
112 type 38 queries
24 AXFR queries
99118 ANY queries
++ Name Server Statistics ++
(Legend)
RR RNXD RFwdR RDupR RFail
RFErr RErr RAXFR RLame ROpts
SSysQ SAns SFwdQ SDupQ SErr
(Global)
…
```

Здесь только начало отчета файла статистики. Он довольно большой. Кроме того, каждое прерывание сервера (на самом деле мы выполняем прерывание, только сервер его перехватывает и обрабатывает) таким образом приводит к дописыванию статистики в конец файла, что только увеличивает его размер. Если сервер работает не долго (только запущен), то цифры отчета будут не большими J:

```
quest# more /etc/namedb/named.stats
+++ Statistics Dump +++ (1037800585)
success 0
referral 0
nxrrset 0
nxdomain 0
recursion 0
failure 0
--- Statistics Dump --- (1037800585)
quest#
```

Для того, чтобы появилась хоть какая-то статистика нужно его о чем-нибудь спросить:

```
quest# /usr/local/bin/dig @localhost www.ru +short
194.87.0.50
quest#
```

Теперь если посмотреть статистику обращений, то получим:

```
quest# more /etc/namedb/named.stats
+++ Statistics Dump +++ (1037800853)
success 3
referral 0
nxrrset 0
nxdomain 0
recursion 1
failure 0
--- Statistics Dump --- (1037800853)
quest#
```

На самом деле для того, чтобы "прочувствовать" на нашем сервере кэширование, мы спросили сервер не один, а три раза. Именно это и отражено в отчете (success). А рекурсивный запрос был выполнен только один (recursion).

Для BIND 8.x статистику получают командой ndc. Записывается статистика при этом не в /var/tmp, а уже в директорию файлов описания зоны. При этом мы запускаем ndc в интерактивном режиме:

```
polyn: {1} ndc
Type help -or- /h if you need help.
ndc> stats
Statistics dump initiated.
ndc> /exit
polyn: {2}
```

При помощи ndc можно не только собирать статистику, но и управлять сервером.

На самом деле приведенные примеры файлов статистики (кроме первого) сделаны при помощи команды rndc для сервера BIND 9.2.1. Для того, чтобы их получить, настройка сервера без удаленного контроля не годится. Нужно включить удаленный контроль.

Для того, чтобы его включить, нужно сгенерировать ключ. Это можно сделать любой из программ: dnskeygen из пакета BIND 8 или dnssec-keygen. На самом деле, вторая программка на некоторых системах не работает из-за ошибки взаимодействия с /dev/random, поэтому мы воспользовались первой.

```
quest# dnskeygen -H 128 -h -n rndc-key
** Adding dot to the name to make it fully qualified domain name**
Generating 128 bit HMAC-MD5 Key for rndc-key.

Generated 128 bit Key for rndc-key. id=0 alg=157 flags=513

quest#
```

Теперь нужно поправить named.conf:

```
quest# more named.conf
options {
directory "/etc/namedb";
};
```

```
zone "." {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "localhost.rev";
};
key "rndc-key" {
algorithm hmac-md5;
secret "0V5661f2iotUKMKVDXNydQ==";
};
controls { inet 127.0.0.1 allow { localhost; } keys {"rndc-key"; }; };

quest#
```

Появилась директива key и изменилась директива controls. Собственно, нам нужен был только ключ, который записан в опции secret. Он был сгенерирован программой dnskeygen по имени rndc-key и помещен в файл Krndc-key.+157+00000.private в том же каталоге, из которого запускалась программа генерации ключа. На самом деле там появился еще один файл Krndc-key.+157+00000.key с KEY записью описания ресурсов. Ключ можно взять и оттуда.

Само слово "rndc-key" в директиве key файла named.conf - это просто имя ключа, на которое мы ссылаемся в директиве controls.

Теперь нужно создать файл настройки rndc - rndc.conf в каталоге /etc:

```
options {
default-server localhost;
default-key "rndc-key";
};
key "rndc-key" {
algorithm hmac-md5;
secret "0V5661f2iotUKMKVDXNydQ==";
};
```

Директива key в точности повторяет директиву из файла настройки named. Теперь запускаем named:

```
>named
```

Все должно работать. При этом можно выполнять команду rndc.

На самом деле существует более простой путь. Нужно было просто создать файл rndc.key в каталоге /etc, который содержал бы описание ключа:

```
key "rndc-key" {
algorithm hmac-md5;
secret "0V5661f2iotUKMKVDXNydQ==";
};
```

При этом из файла конфигурации named можно вообще убрать директивы controls и key:

```
options {
directory "/etc/namedb";
};
zone "." {
type hint;
file "named.root";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "localhost.rev";
};
```

BIND 9.2.1 сам прочитает rndc.key и запуститься с каналом управления для локальной машины:

```
Nov 20 17:39:46 quest named[34430]: starting BIND 9.2.1
Nov 20 17:39:46 quest named[34430]: command channel listening on 127.0.0.1#953
```

Программа rndc при этом также не нуждается в файле настройки rndc.conf, а использует файл ключа rndc.key.

На этом закончим обсуждение настроек кэширующего сервера. Заметим только, что использование кэширующего сервера - это стандартный прием, к которому прибегают при обслуживании локальных сетей. Он позволяет сократить DNS трафик и, если сеть защищена, его (трафик) контролировать.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
4. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
5. BIND Configuration File Guide (8.3.4) (ftp://ftp.isc.org/isc/bind/src/8.3.4/bind-doc.tar.gz)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind8.html - страничка BIND 8.
2. http://www.isc.org/products/BIND/bind4.html - страничка BIND 4.9.11
3. http://www.acmebw.com/resources/papers/securing.pdf - Securing an Internet Name Server. CERT Coordination Center. Carnegie Mellon University. 2002. Довольно подробный и обстоятельный обзор возможных проблем безопасности серверов доменных имен с рекомендациями по их (серверов) конфигурации.
4. bind9-users-bounce@isc.org - список рассылки и форум пользователей BIND9. Очень забавное чтение. Особенно полемика разработчиков BIND и профессора Бернштейна.

8. b. Небольшой корпоративный домен в домене ru. Описание "прямых" зон. Описание "обратных" зон. Настройки BIND.

В данном материале мы рассмотрим конфигурацию программы named и файлы описания зон, необходимые для поддержки простого корпоративного домена в зоне ru.Кроме того, мы разберем настройку named для обслуживания запросов к этому корпоративному домену и рекурсивных запросов хостов корпоративной сети через тот же сервер доменных имен.

В первую очередь определимся с термином "корпоративный домен". Мы его используем здесь для обозначения домена второго уровня, т.е. все, что имеет имя типа name.ru - это корпоративный домен.

Строго говоря, это неправильно. Например, msk.ru - это географический домен, а pp.ru - это домен персональных доменов, он относится к типу доменов общего пользования (лучше было бы сказать "общего назначения"). Корпоративный домен - это домен компании.

Если бы домены государственных учреждений или высших учебных заведений размещались бы в доменах типа gov.ru и edu.ru, то можно было бы отдельно говорить как о тех, так и о других, хотя принципиального различия между процедурами поддержки таких доменов нет. Однако, наши учреждения регистрируют свои домены, как домены второго уровня, т.е. не используют домены общего пользования.

Поэтому, обобщив прилагательное "корпоративный" на все имена предшествующие ru, в контексте данного материала, мы будем вести речь о настройке и поддержке зоны корпоративного домена.

Мы оставляем также за пределами данного материала саму процедуру регистрации домена. Она достаточно подробно описана на сайтах независимых регистраторов, например, http://www.nic.ru/dns/service/how.html. Заметим только, что до того, как начать регистрацию домена нужно сконфигурировать и запустить все необходимые для поддержки домена серверы доменных имен. Это нужно сделать для того, чтобы при делегировании домена регистратор мог проверить их работоспособность.

Разбирать настройку сервера BIND (named) мы будем на примере домена vega.ru, который охватывает сеть класса C (194.226.43.0/24). При этом master сервер зоны мы разместим в этой же IP-сети, а поповоду slave сервера мы заключим соответствующее соглашение с администрацией сервера ns.relarn.ru.

Согласно правилам регистрации доменов второго уровня, как минимум, два сервера доменных имен авторитативных для зоны vega.ru нам необходимо иметь. При чем серверы должны подключаться к Интернет независимо.

Описание "прямой" и "обратных" зон

Сначала разберемся с файлами описания зон, которые будет поддерживать наш сервер. Они идентичны за небольшим исключением (параметры записи SOA и $TTL) для различных версий BIND.

Прямая зона, ради которой, собственно, и затевался весь сыр-бор, будет выглядеть следующим образом:

```
$TTL 3600
@ IN SOA vega-gw.vega.ru. hostmaster.vega.ru. (
101 ;serial number
86400 ;refresh
3600 ;retry
3888000 ;expire
3600 ;minimum
)
; Name server
IN NS vega-gw.vega.ru.
IN NS ns.relarn.ru.
IN A 194.226.43.1
IN MX 0 vega-gw.vega.ru.
IN MX 10 ns.relarn.ru.
;
vega-gw IN A 194.226.43.1
IN MX 0 vega-gw
IN MX 10 ns.relarn.ru.
www IN CNAME vega-gw
ftp IN CNAME vega-gw
gopher IN CNAME vega-gw
;
dos1 IN A 194.226.43.2
dos2 IN A 194.226.43.3
; The rest of the net should be presented below.
```

Символ "@" ссылается на имя зоны (vega.ru) из файла конфигурации named. Директива управления $TTL в BIND 4.x не применяется, и ее лучше для этой версии сервера не использовать. Время жизни записей описания ресурсов в кэше сервера будет определяться в BIND 4.x не директивой управления $TTL, а параметром minimum в записи SOA. В BIND 8.x и 9.x этот параметр используется для определения времени кэширования негативных ответов других серверов, поэтому для времени кэширования записей описания ресурсов по умолчанию нужно задавать директиву управления $TTL.

Из записи SOA следует, что master сервером домена является хост vega-gw.vega.ru, а администратор сервера имеет адрес hostmaster.vega.ru. Кроме того, среди серверов доменных имен, авторитативных для зоны vega.ru (записи NS), указан сервер ns.relarn.ru. Однако, адресной записи для этого доменного имени в описании зоны нет, т.к. его можно найти в зоне relarn.ru.

В BIND 4.x имело смысл прописывать адресную запись ns.relarn.ru, т.к. сервер вернул бы ее в дополнительной секции отклика. BIND 8.x и 9.x игнорируют такого сорта записи, поэтому мы в нашем примере ее не указываем. Собственно, мы поступаем в соответствие с рекомендациями RFC 1912.

Наш домен небольшой. Предполагается, что в нем совсем мало машин. Администрация сети не может себе позволить разносить сервисы по разным хостам. По этой причине почтовые ящики для vega.ru будут располагаться на сервере доменных имен. Точнее этот хост будет знать, как почту доставлять, т.к. он является почтовым шлюзом с наибольшим

приоритетом для домена vega.ru. Об этом говорят записи MX, которые следуют сразу за адресной записью в начале описания зоны.

Сама адресная запись в начале описания зоны призвана назначить для имени зоны IP-адрес, чтобы такие сервисы, как World Wide Web, например, приводили на корпоративную страничку не только по www.vega.ru, но и по vega.ru.

Далее следует адресная запись, которая определяет IP-адрес для master сервера зоны. За ней определены почтовые шлюзы, которые знают, как на этот сервер доставлять почту, и синонимы сервисов (ftp,www,gopher), которые размещены на этом же хосте. Вслед за сервисами размещаются записи адресов для всех остальных хостов зоны.

Здесь следует заметить, что довольно часто вместо CNAME для описания сервисов используют адресные записи. В принципе, это позволяет найти адрес для имени сразу, но, если будет проверяться обратное соответствие, а в "обратной" зоне будет описано каноническое имя, то имена не совпадут, и сервис может отказать клиенту, который запрашивает сервис, в обслуживании.

Еще одно важное замечание касается начала описания зоны. В нашем описании у SOA записи в качестве имени зоны стоит символ "@". Он позволяет сослаться на имя зоны из файла конфигурации named.

Имя зоны в SOA всегда должно содержать какое-либо имя. Иначе все остальные записи окажутся просто бесполезными. Описание зоны можно было начать и по другому:

```
$TTL 3600
$ORIGIN ru.
vega IN SOA vega-gw.vega.ru. hostmaster.vega.ru. (
101 ;serial number
86400 ;refresh
3600 ;retryz
3888000 ;expire
3600 ;minimum
)
```

Это описание также будет указывать на зону vega.ru. Слово "vega" будет расширяться до "vera.ru" по умолчанию.

Существует еще одно имя, которое стоит обсудить в контексте описания прямой зоны. Это имя localhost. В нашем случае - это localhost.vega.ru.

Рассмотрим сначала такой пример:

```
> host localhost.ru.
localhost.ru has address 195.68.136.26
localhost.ru mail is handled (pri=100) by mx2.elcomsoft.com
localhost.ru mail is handled (pri=200) by mx1.elcomsoft.com
>
```

Из примера следует, что в зоне ru есть хост с адресом localhost.ru. На самом деле это побочное явление того, что существует зона localhost:

```
> host -t soa localhost.ru.
localhost.ru start of authority ns1.localhost.ru noc.elcomsoft.com(
2001040300 ;serial (version)
14400 ;refresh period
3600 ;retry refresh this often
604800 ;expiration period
3600 ;minimum TTL
)
>
```

Если теперь посмотреть адрес localhost.nic.ru, например, то мы получим локальную петлю:

```
> host localhost.nic.ru.
localhost.nic.ru has address 127.0.0.1
>
```

Такая настройка "прямой" зоны является типовой, а точнее являлась типовой:

```
> host localhost.demos.ru.
localhost.demos.ru has address 127.0.0.1
> host localhost.relcom.ru.
localhost.relcom.ru has address 127.0.0.1
> host localhost.msk.ru.
localhost.msk.ru has address 127.0.0.1
> host localhost.spb.ru.
localhost.spb.ru has address 127.0.0.1
> host localhost.rambler.ru.
localhost.rambler.ru has address 127.0.0.1
> host localhost.yandex.ru.
localhost.yandex.ru has address 127.0.0.1
>
```

Современные провайдеры уже поступают иначе:

```
> host localhost.mail.ru.
Host not found.
> host localhost.localhost.ru.
Host not found.
> host localhost.lenta.ru.
Host not found.
> host localhost.runet.ru.
Host not found.
>
```

Приведенный пример показывает, что в соответствующих зонах хоста с именем localhost просто нет. При этом сами зоны прекрасно функционируют. Если бы запросить зоны vega.ru на предмет наличия в ней соответствия IP-адреса доменному имени localhost.vega.ru, то отклик программы host был бы таким же, как и в последнем примере.

На самом деле, это отголоски дискуссии по поводу имени localhost, которая велась в конце 90-х. В конце концов, в 1996 году вышел документ RFC 1912, в котором этот вопрос был прояснен. Более того, в RFC 2606 для localhost был зарезервирован специальный домен верхнего уровня localhost.

И все-таки, для чего и каким образом используется имя localhost при обращении к системе доменных имен? Ответ на этот вопрос является определяющим при конфигурации локального сервера доменных имен для работы с этим именем.

Имя localhost указывает на "петлю" - адрес 127.0.0.1, который закреплен за хостом исполняющим программу. По какой либо причине эта программа обращается к хосту с именем localhost. Система доменных имен должна вернуть в качестве отклика адрес 127.0.0.1.

Обратиться к системе DNS можно, используя два типа имен: полностью определенные имена (FQDN) и частично определенные имена. В первом случае имя кончается символом ".", т.к. у корневого домена имени нет. Во втором случае имя точкой не кончается.

Если программа обращается к приложению на той же машине, где она сама исполняется, то в качестве имени машины может быть указано FQDN, т.е. "localhost.", или "localhost", т.е. неполное имя.

На данном этапе в дело вступает библиотека resolver. Все будет теперь определяться тем алгоритмом, который реализует эта библиотека при построении FQDN.

С полностью определенным именем все понятно. Локальный сервер начнет искать домен верхнего уровня localhost. Согласно RFC 2606 такой домен должен существовать, и состоит из одной записи, которая имени "localhost." ставит в соответствие адрес 127.0.0.1.

А вот с неполным именем такой ясности нет. Если файл resolv.conf на хосте, где исполняется прикладная программа, составлен стандартным образом, т.е. там только указан сервер доменных имен и имя домена, в который входит хост, то сначала имя localhost будет расширено именем локального домена из resolv.conf, а уж только после этого определено, как имя "localhost." (см. материал "Resolver. Типовые настройки.").

Если на хосте в resolv.conf вместо директивы domain применяется директива search, то процедура поиска может увеличиться на несколько дополнительных подстановок.

Совершенно очевидно, что хочется получить адрес 127.0.0.1 как можно быстрее, и это возможно, если имя типа localhost.domain.ru будет тоже указывать на 127.0.0.1. Быстрее будет потому, что соответствие будет найдено сразу в локальном домене, и обращение к корневым серверам не потребуется. Конечно, это возможно организовать только в том случае, когда имя типа localhost.domain.ru не используется по каком-либо другому назначению.

Таким образом, для ускорения поиска в прямую зону принято вводить запись вида:

```
$TTL 3600
@ IN SOA vega-gw.vega.ru hostmaster.vega.ru (
101 ;serial number
86400 ;refresh
3600 ;retry
3888000 ;expire
3600 ;minimum
)
; Name server
IN NS vega-gw.vega.ru.
```

```
IN NS ns.relarn.ru.
IN A 194.226.43.1
IN MX 0 vega-gw.vega.ru.
IN MX 10 ns.relarn.ru.
;
localhost IN A 127.0.0.1
;
vega-gw IN A 194.226.43.1
```

Мы представили пример с окружением, чтобы было понятно, куда обычно вставляют эту адресную запись в прямой зоне.

Вообще говоря, это нерекомендованная практика. RFC 1912 Рекомендует создать отдельно зону localhost:

```
$TTL 1D
localhost. IN SOA vega-gw.vega.ru. hostmaster.vega.ru. (
101 ;serial number
86400 ;refresh
3600 ;retry
3888000 ;expire
3600 ;minimum
)
localhost. IN NS vega-gw.vega.ru.
localhost. IN A 127.0.0.1
```

Соответственно, эта зона должна быть прописана и в файле настройки named.

Идея с наличием этой зоны достаточно прозрачна. Сервер при обращении за адресом хоста "localhost." не будет обращаться к корневым серверам, т.к. сам знает этот адрес.

Однако, если зона localhost не будет прописана на локальном сервере, который выполняет рекурсивные запросы прикладных программ, корневой сервер должен вернуть правильный адрес, т.к. в DNS, согласно RFC 2606, эта зона должна поддерживаться, как домен верхнего уровня localhost.

Если же допустить, что по какой-либо причине корневые серверы недоступны, а зона localhost не прописана, то тогда программа все равно получит правильный адрес, т.к. в этом случае система перейдет от поиска адреса в DNS к поиску адреса в файле hosts, а там по умолчанию 127.0.0.1 закреплен за именем localhost.

Теперь обратим внимание на зону 0.0.127.in-addr.arpa. она является обратной для зоны localhost. Эта зона всегда прописывается на серверах доменных имен:

```
$TTL 1D
0.0.127.in-addr.arpa. IN SOA vega-gw.vega.ru paul.vega.ru (
101 ;serial number
86400 ;refresh
3600 ;retry
3888000 ;expire
3600 ;minimum
)
IN NS vega-gw.vega.ru.
```

```
; Localhost declaration
1 IN PTR localhost.
```

Назначение этой зоны заключается в ответах на запросы поиска доменного имени для IP-адреса 127.0.0.1. Согласно RFC 1912 имя это должно быть "localhost.". Довольно часто на "старых" доменах можно встретить указание на имя типа localhost.domain.ru.

В RFC 1912 определены еще две обратных зоны, которые рекомендуется иметь на сервере доменных имен, но которые обычно не настраивают. О них даже нет упоминания в широко распространенных по сети рекомендациях по настройке серверов доменных имен. Это зоны 255.in-addr.arpa и 0.in-addr.arpa.

Назначение этих зон состоит в том, чтобы избежать трансляции случайных запросов имен соответствующих IP-адресов на серверы, обслуживающие корневую зону.

Содержание файлов описания этих зон одинаковое: только SOA и NS записи:

```
$TTL 1D
255.in-addr.arpa. IN SOA vega-gw.vega.ru paul.vega.ru (
101 ;serial number
86400 ;refresh
3600 ;retry
3888000 ;expire
3600 ;minimum
)
IN NS vega-gw.vega.ru.
```

Для 0.in-addr.arpa в поле имени зоны нужно "255" заменить на "0".

На самом деле, эффективность с точки зрения времени поиска адресов и имен в зонах "петли" может быть очень высокой, если установить большое время в управляющей директиве $TTL. В этом случае адрес попадает в кэш и может жить там "вечно".

На самом деле все зоны кроме зоны vega.ru должны быть настроены на любом сервере доменных имен, в том числе и на сервере, выполняющем только функции кэширующего сервера. Обычно же на кэширующем сервере ограничиваются только описанием зоны 0.0.127.in-addr.arpa (см. матерал "Настройка кэширующего сервера доменных имен. Примеры описания зон и файлов конфигурации BIND. Запуск и проверка работоспособности."), что оправдано, т.к. зона localhost должна быть прописана на корневых серверах. Кроме того, существует еще файл hosts.

Мы написали "должна быть прописана" не зря. На самом деле на момент написания этого текста она там прописана не была:

```
> /usr/local/bin/dig localhost.

; <<>> DiG 9.2.1 <<>> localhost.
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 57298
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;localhost. IN A

;; AUTHORITY SECTION:
. 538 IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISIGN-GRS.COM.
2002112700 1800 900 604800 86400

;; Query time: 2 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Wed Nov 27 22:16:56 2002
;; MSG SIZE rcvd: 102

>
```

И это, не смотря на то, что документ RFC 2606, в котором декларируется организация такой зоны, вышел в 1999 году. Таким образом, зону localhost прописывать все же надо.

На самом деле мы не описали еще одну зону, которую необходимо иметь для запуска named - зону описания корневых серверов. Она описывается обычно в файле named.root, который совпадает в точности с аналогичным файлом, описанном в материале "Настройка кэширующего сервера доменных имен. Примеры описания зон и файлов конфигурации BIND. Запуск и проверка работоспособности.".

Теперь перейдем к рассмотрению файлов конфигурации named. Будем рассматривать настройку named по версиям программного обеспечения. Сначала рассмотрим BIND 4.x, а потом BIND 8.x и BIND 9.x.

Настройка BIND 4.x для поддержки небольшого корпоративного домена

Сначала мы рассмотрим случай, когда наш сервер должен будет только обслуживать запросы к зоне своей ответственности vega.ru. Это означает, что он не должен обрабатывать рекурсивные запросы.

Программа named, установленная на машине шлюзе будет иметь следующую конфигурацию, которая задается файлом named.boot:

```
;
; Zone vega.ru data base files.
;
directory /etc/namedb
primary vega.ru vega.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
```

В этом файле директивой directory определено, что директория расположения описания файлов зоны vega.ru - /etc/namedb. Это стандартная директория для BIND. Первая директива primary определяет файл описания "прямой" зоны домена vega.ru - файл vega.ru,

размещенный в директории /etc/namedb (/etc/namedb/vega.ru). Вторая директива primary определяет описание зоны localhost.

Кроме "прямых" зон описаны еще две обратных зоны 127.in-adrr.arpa и 43.226.194.in-addr.arpa. Первая определяет обратное соответствие между адресом 127.0.0.1 и именем localhost, а вторая множество обратных соответствий для сети 194.226.43.0.

Последней директивой указан файл начальной загрузки cache. Точка, символ "." в качестве первого аргумента указывает на описание корневой зоны.

Теперь нам нужно отключить рекурсию. Это делается при помощи директивы options:

```
;
; Zone vega.ru data base files.
;
directory /etc/namedb
primary vega.ru vega.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
options no-recursion
```

Вообще говоря, опция no-recursion обычно употребляется вместе с опцией no-fetch-glue. Это позволяет запретить серверу искать адреса серверов доменных имен при конструировании отклика с секцией дополнительных данных. Например, искать и заносить в эту секцию адресные записи для серверов доменных имен, которые посылаются в ответах (refferal) на запросы к зонам, для которых данный сервер не является авторитативным. Это позволяет избежать лишнего трафика, а также "отравления" кэша нашего сервера.

```
;
; Zone vega.ru data base files.
;
directory /etc/namedb
primary vega.ru vega.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
options no-recursion no-fetch-glue fake-iquery
```

В примере кроме no-fetch-glue мы добавили еще одну опцию - fake-iquery. Она позволяет правильно обрабатывать инверсные запросы (не путать с запросами к обратным зонам). Инверсные запросы считаются атавизмом и не поддерживаются в современных версиях DNS серверов, но обрабатывать их нужно корректно. Если не указывать опцию обработки этих запросов, то сервер будет сообщать об ошибке, что неправильно.

На самом деле мы забыли еще об одной вещи. Для нашей зоны существует еще и slave сервер. В нашей конфигурации он никак не упомянут. Это никак его не ограничивает, и он может благополучно копировать описание зоны с нашего сервера. Но кроме него это могут делать и другие серверы и программы. При современных веяниях это нехорошо. Интернет слишком агрессивная среда для такой открытости.

Следует ограничить число хостов, которым дозволено копировать зону:

```
;
; Zone vega.ru data base files.
;
directory /etc/namedb
primary vega.ru vega.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
xfrnets 194.226.65.0
cache . named.root
options no-recursion no-fetch-glue fake-iquery
```

В данном случае мы определили возможность копирования зоны только для хостов из сети, где расположен наш slave сервер. В принципе, можно определить хост точно. Для этого следует применить маску:

```
xfrnets 194.226.65.3&255.255.255.255
```

Сразу видно, что синтаксис "старый". Сейчас написали бы 194.226.65.3/32.

Когда мы говорили только об обслуживании запросов к зоне, то мы действовали в строгом соответствии с рекомендациями CERT, где указывается на желательность разнесения функций кэширующего сервера и сервера, отвечающего за зону.

В большинстве случаев применения BIND эти функции совмещают. Севрер является авторитативным сервером корпоративного домена и одновременно выполняет функции кэширующего сервера для хостов корпоративной сети.

Очевидным изменением файла настройки named в этом случае будет отказ от запрета на рекурсию:

```
;
; Zone vega.ru data base files.
;
directory /etc/namedb
primary vega.ru vega.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
xfrnets 194.226.65.0
cache . named.root
forwarders 194.226.65.3
options fake-iquery
```

Кроме отказа от рекурсии в в этом файле конфигурации появилась еще одна директива - forwarders. Она инструктирует сервер в том смысле, чтобы он отправлял запросы, на которые не может ответить сам, на сервер 144.226.65.3.

Таким образом, наш сервер не только обрабатывает рекурсивные запросы, но и сам их порождает, пытаясь воспользоваться "знаниями" другого сервера доменных имен. Естественно, что на том сервере нам должны разрешить себя вести подобным образом.

Теперь рассмотрим как настраивается сервер версий BIND 8.x и 9.x для выполнения точно таких же функций.

Настройка BIND 8.x и BIND 9.x для поддержки небольшого корпоративного домена

Во-первых, здесь мы будем иметь дело с файлом named.conf, который располагается по умолчанию либо в каталоге /etc (BIND 9.x), либо /etc/namedb (BIND 8.x). Во-вторых, формат директив этого файла конфигурации совершенно иной, чем в BIND 4.x.

Сначала рассмотрим сервер, который не поддерживает рекурсивных запросов, но обслуживает запросы к зоне своей ответственности:

```
options {
directory "/etc/namedb";
allow-query {any;};
recursion no;
fake-iquery yes;
fetch-glue no;
use-id-pool yes;
};
//Root server hints
zone "." { type hint; file "named.root"; };
// Forward Loopback
zone "localhost" {
type master;
file "localhost";
};
// Reverse Loopback
zone "0.0.127.in-addr.arpa" {
type master;
file "localhosr.rev";
};
// Corporative domain
zone "vega.ru" {
type master;
file "vega.ru";
allow-transfer { 194.226.65.3; };
};
// Reverse corporative domain
zone "43.226.194.in-addr.arpa" {
type master;
file "43.226.194.in-addr.arpa";
allow-transfer { 194.226.65.3; };
};
```

Директива options задает опции, значение которых распространяется на все зоны, поддерживаемые данным сервером. Опция directory определяет место расположения файлов описания зон. Опция allow-query разрешает обслуживать запросы от всех клиентов Интернет. Такое значение в данном случае позволяет отвечать на запросы к зонам ответственности сервера. Опция recursion отменяет обслуживание любых рекурсивных

запросов. Далее следуют опции имитации обслуживания инверсных запросов (fake-iquery), отмены поиска дополнительной информации для ответов рефералов (referrals, опция fetch-glue, в BIND 9.x не нужна, т.к. реализована по умолчанию), и опция противодействия спуфингу (use-id-pool в BIND 9.x не нужна, т.к. реализована по умолчанию).

Далее идет описание корневой зоны. Тип зоны указан как hint (буквально "подсказка"). Зона содержит информацию об именах и адресах серверов,обслуживающих корневую зону DNS.

За описанием корневой зоны следуют описания зон "петли" (localhost и 0.0.127.in-addr.arpa). Сервер определен для этих зон как master зоны. Ни каких дополнительных опций в описании конфигурации этих зон нет.

За ними следует описание настроек named для управления корпоративной зоной (vega.ru). Здесь мы ограничили число хостов, которые могут копировать зону slave сервером зоны. Если необходимо, то здесь можно расположить и другие хосты, которым будет разрешено копировать зону из соображений разгрузки master сервера, например.

Последней указаны настройки named для работы с "обратной" корпоративной зоной. Эти настройки ничем не отличаются от настроек "прямой" зоны.

Теперь мы расширим функциональность нашего сервера и разрешим ему обслуживать рекурсивные запросы. Но только пусть эти запросы будут исходить из корпоративной сети. Если мы просто разрешим рекурсию в директиве options:

```
options {
directory "/etc/namedb";
allow-query {any;};
recursion yes;
fake-iquery yes;
fetch-glue no;
use-id-pool yes;
};
```

то мы своей цели не добьемся. Сервер будет обрабатывать любые рекурсивные запросы, откуда бы они не исходили. В принципе для такого поведения вообще не нужно указывать опцию recursion. Рекурсия включается по умолчанию.

Для того, чтобы разрешить рекурсию только "своим" хостам, следует воспользоваться директивой allow-recursion:

```
options {
directory "/etc/namedb";
allow-query {any;};
recursion no;
fake-iquery yes;
fetch-glue no;
use-id-pool yes;
allow-reqursion { 194.226.43/24;};
};
```

В данном случае рекурсия разрешена только хостам из сети 194.226.43.0. Для всех остальных она запрещена опцией recursion.

Вообще говоря, из всех зон, которыми управляет наш сервер, для всего остального мира интересны только корпоративные зоны: прямая зона vega.ru и "обратная" зона 43.226.194.in-addr.arpa. Поэтому, если строго следовать назначению нашего сервера - обслуживать по максимуму корпоративную сеть и по необходимому минимуму весь остальной мир, мы должны несколько изменить настройки.

В первую очередь разделим весь мир на "своих" и "чужих". Это делается при помощи директивы acl (access control list):

```
acl "vega-net" {
194.226.43/24;
};
acl "vega-friend" {
194.226.65/24;
};
options {
directory "/etc/namedb";
allow-query {vega-net;};
recursion no;
fake-iquery yes;
fetch-glue no;
use-id-pool yes;
};
```

Два списка контроля доступа содержат пулы адресов корпоративной сети и сети slave сервера. Первый список мы применяем в директиве options. Разрешаем серверу обрабатывать запросы только от "наших" хостов (allow-query).

Опции директивы options распространяются на все зоны нашего сервера, поэтому, если мы хотим какую-либо из них обслуживать иначе, то должны в ее конфигурацию ввести изменения. Иначе мы обслуживаем зоны vega.ru и 43.226.194.in-addr.arpa:

```
// Corporative domain
zone "vega.ru" {
type master;
file "vega.ru";
allow-query {any; };
allow-transfer { vega-friend; };
};
// Reverse corporative domain
zone "43.226.194.in-addr.arpa" {
type master;
file "43.226.194.in-addr.arpa";
allow-query {any; };
allow-transfer { vega-friend; };
};
```

В описании этих зон мы применили список доступа vega-friend и только для этих зон разрешили обслуживание любых нерекурсивных запросов.

В принципе, можно было обойтись и без списков доступа, но в данном случае мы просто преследовали цель продемонстрировать способ их применения. Корпоративная сеть может быть развернута не на одной сети класса С, для дружественных сетей можно разрешить не только копирование зоны, но и удаленное ее (зоны) обновление. В этом случае списки увеличатся в объеме, и их копирование в разные части файла конфигурации может привести к нежелательным ошибкам.

Кроме того, при изменениях корпоративной сети и дружественных сетей вносить изменения придется только в списки управления доступом, а не в разбросанные по всему файлу описания. Использование списков управления (контроля) доступа - это более правильный стиль.

В BIND 9.x есть еще один механизм для определения различного поведения сервера при обслуживании запросов - "views". В материалах CERT для управления рекурсией приведена, например, такая конфигурация:

```
// Corporative domain
view "internalview" {
match-clients { internal; };
recursion yes;
};
// Internet
view "externalview" {
match-clients { any; };
recursion no;
};
```

Слово "internal" - это имя списка доступа, а слово "any" обозначение любого хоста Интернет.

На этом, пожалуй, можно и остановиться. Дальнейшее усложнение конфигурации сервера будет связано с вопросами безопасности и управления обновлениями зоны, но эти вопросы логичнее рассмотреть в других материалах, специально посвященных этим проблемам.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
3. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
4. D. Eastlake, A. Pantiz. RFC 2606. Reserved Top Level DNS Names. 1999.(http://www.ietf.org/rfc/rfc2606.txt)
5. D. Barr. RFC 1912. Common DNS Operational and Configuration Errors. 1996. (http://www.ietf.org/rfc/rfc1912.txt?number=1912)
6. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
7. BIND Configuration File Guide (8.3.4) (ftp://ftp.isc.org/isc/bind/src/8.3.4/bind-doc.tar.gz)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind8.html - страничка BIND 8.
2. http://www.isc.org/products/BIND/bind4.html - страничка BIND 4.9.11
3. http://www.acmebw.com/resources/papers/securing.pdf - Securing an Internet Name Server. CERT Coordination Center. Carnegie Mellon University. 2002. Довольно подробный и обстоятельный обзор возможных проблем безопасности серверов доменных имен с рекомендациями по их (серверов) конфигурации.
4. http://www.ietf.org/proceedings/02jul/I-D/draft-ietf-dnsop-dontpublish-unreachable-03.txt - хотя на документы этого типа и не принято ссылаться, т.к. они имеют силу только в течение полугода, но в этом файле содержится вразумительное описание мотивации создания зоны localhost и адресных записей в зоне корпоративного домена типа localhost.domain.ru

8. c. Slave сервер для корпоративного домена. Настройки BIND.

В данном материале мы рассмотрим конфигурацию программы named при организации slave сервера. Основное внимание будет уделено вопросам контроля работы сервера и управления копированием зон с master сервера зоны.

Наличие slave сервера у корпоративного домена не менее важно, чем организация master сервера. При регистрации домена надежность slave сервера проверяется точно таким же способом, что и надежность master сервера. По этой причине подходить к выбору места размещения slave сервера следует также тщательно, как и к выбору хоста, на котором будет размещен master сервер.

При организации slave сервера не нужно создавать никаких файлов описания зон. Они сами создаются при выполнении копирования зоны. Необходимо только правильно настроить named, т.е. прописать все необходимые опции в файле конфигурации программы.

Slave сервер выполняет процедуру автоматической синхронизации описания зоны на авторитативных серверах, которая в общем виде описана в разделе 4.3.5 RFC 1034.

Согласно этому документу обмен данными между серверами рекомендовано производить по TCP протоколу, используя тип запроса AXFR. По этому запросу за одно TCP соединение должна передаваться вся зона целиком (RFC 1035).

Важным моментом при обмене данными зоны являются настройки как master, так и slave серверов. Рассмотрим эти настройки более подробно, разбив рассмотрение по версиям BIND.

Настройка slave сервера в BIND 4.x.

Мы будем рассматривать настройку сервера в качестве slave сервера в контексте домена vega.ru, т.е. сервер этого домена будет одновременно выполнять функции slave сервера для другого домена:

```
directory /etc/namedb
primary vega.ru vega.ru
secondary corp.ru 192.168.0.1 10.0.0.1 corp.ru
```

primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root

В данном случае наш сервер размещает файлы описания зон в /etc/namedb. Именно там и появится файл зоны, которую этот сервер с копирует с одного из серверов домена corp.ru.

Сами master серверы прописаны в директиве secondary вслед за именем домена. Всего можно перечислить до 10 IP-адресов серверов, с которых наш сервер может скопировать зону.

Последний аргумент директивы secondary - это имя файла, куда записываются данные зоны. В нашем случае мы используем тот же принцип наименования файлов описания зон, что и в случае "прямых" зон: имя файла совпадает с именем зоны.

На самом деле, для slave сервера действуют все те же принципы ограничения доступа к описанию зоны и защите его от повреждения, что и для "прямой зоны".

Во-первых, можно ограничить обслуживание рекурсивных запросов:

directory /etc/namedb
primary vega.ru vega.ru
secondary corp.ru 192.168.0.1 10.0.0.1 corp.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
options no-recursion

Во-вторых, применить опции снижающие опасность спуфинга сервера:

directory /etc/namedb
primary vega.ru vega.ru
secondary corp.ru 192.168.0.1 10.0.0.1 corp.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
cache . named.root
options no-recursion no-fetch-glue fake-iquery

В данном случае отменяется формирование дополнительной секции данных и объявляется корректная обработка инверсных запросов.

Теперь мы можем ограничить число хостов, которым сами разрешаем копировать зону:

directory /etc/namedb
primary vega.ru vega.ru
secondary corp.ru 192.168.0.1 10.0.0.1 corp.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 43.226.194.in-addr.arpa vega.rev
xfrnets 194.226.65.1&255.255.255.255

```
cache . named.root
options no-recursion no-fetch-glue fake-iquery
```

В данном случае число таких хостов уменьшили до одного.

На самом деле, файл, который мы получаем при копировании зоны на slave сервер несколько отличается от исходного. Например, там нет "приклеенных" адресных записей, которые не относятся к нашей зоне.

Настройка slave сервера в BIND 8.x и 9.x.

Настройка BIND 8.x и 9.x отличается синтаксисом и более широкими возможностями по управлению конфигурацией сервера. Для того, чтобы поддерживать slave сервер нам нужно в файл конфигурации вставить код вида:

```
zone "corp.ru" {
type slave;
file "corp.ru";
masters { 192.168.0.1; 10.0.0.1;};
};
```

Многие настройки в BIND 8.x и 9.x в файле конфигурации собраны в директиве options. Поэтому их указание в описание зоны не требуется. А вот разрешить, или нет копирование зоны, лучше указывать внутри директивы zone:

```
zone "43.226.194.in-addr.arpa" {
type master;
file "43.226.194.in-addr.arpa";
allow-query {any; };
allow-transfer { none; };
};
```

В данном случае, с нашего сервера вообще никто не сможет скопировать зону, т.к. мы указали в списке разрешения на копирование ключевое слово "none". Такая настройка вполне оправдана, т.к. обычно slave является последним сервером в дереве серверов поддерживающих зону.

Вообще говоря, прежде, чем запускать сервер, желательно проверить сааму возможность копирования зоны с master сервера на slave сервер. Для этого можно воспользоваться программой dig, например,:

```
> dig @polyn.kiae.su bard.kiae.ru. axfr

; <<>> DiG 8.3 <<>> @polyn.kiae.su bard.kiae.ru. axfr
; (1 server found)
$ORIGIN bard.kiae.ru.
@ 1H IN SOA ns.polyn.kiae.ru. paul.polyn.kiae.su. (
7 ; serial
1H ; refresh
5M ; retry
```

```
16w3d17h46m39s ; expiry
1H ) ; minimum

1H IN NS polyn.kiae.ru.
1H IN NS iris.polyn.kiae.su.
1H IN MX 10 mail
1H IN MX 20 iris.polyn.kiae.ru.
1H IN A 144.206.192.32
mail 1H IN MX 10 mail
1H IN A 144.206.192.32
www 1H IN A 144.206.192.32
@ 1H IN SOA ns.polyn.kiae.ru. paul.polyn.kiae.su. (
7 ; serial
1H ; refresh
5M ; retry
16w3d17h46m39s ; expiry
1H ) ; minimum

;; Received 10 answers (10 records).
;; FROM: generate.polyn.kiae.su to SERVER: 144.206.160.32
;; WHEN: Fri Dec 6 16:37:57 2002
>
```

Но более правильным было бы использовать саму программу named, а точнее компоненту пакета BIND named-xfer, которая отвечает за копирование зоны на slave сервер:

```
> /usr/libexec/named-xfer -z bard.kiae.ru -f polyn.db polyn.kiae.su
named-xfer[8443]: send AXFR query 0 to 144.206.160.32
>
```

В данном случае мы копируем зону (ее описание) bard.kiae.ru в файл polyn.db с сервера polyn.kiae.su. По умолчанию программа использует запрос типа axfr.

Зона после копирования будет выглядеть примерно следующим образом:

```
; BIND version named 8.2.3-T6B Mon Nov 20 11:24:37 GMT 2000
; BIND version jkh@bento.FreeBSD.org:/usr/obj/usr/src/libexec/named-xfer
; zone 'bard.kiae.ru' first transfer
; from 144.206.160.32:53 (local 144.206.192.55) using AXFR at
; Fri Dec 6 16:47:3 2 2002
$ORIGIN kiae.ru.
bard 3600 IN SOA ns.polyn.kiae.ru. paul.polyn.kiae.su. (
7 3600 300 9999999 3600 )
3600 IN NS polyn.kiae.ru.
3600 IN NS iris.polyn.kiae.su.
3600 IN MX 10 mail.bard.kiae.ru.
3600 IN MX 20 iris.polyn.kiae.ru.
3600 IN A 144.206.192.32
$ORIGIN bard.kiae.ru.
mail 3600 IN MX 10 mail.bard.kiae.ru.
3600 IN A 144.206.192.32
www 3600 IN A 144.206.192.32
>
```

В этом примере стоит обратить внимание на комментарий в начале описания зоны, который программа named-xfer прибавляет от себя. В конце комментария содержится дата копирования зоны.

Named-xfer - это вспомогательный агент (как написано в его документации), который используется в BIND до версии 9.x. В новых серверах этого агента нет. Сервер теперь многопоточный и сам может порождать нить копирования зоны, не мешая обслуживанию запросов к этой зоне.

Распараллеливание процедуры обслуживания обычных запросов к зоне и процедуры копирования зоны было той причиной, по которой появилась программа named-xfer. Речь в данном случае идет о slave сервере. Он при помощи named-xfer запрашивает описание зоны и сохраняет его в файле описания. В этот момент старая копия зоны остается в памяти самого сервера, и он благополучно обслуживает запросы к ней. После получения описания зоны сервер перезагружает описание зоны в своей памяти.

В BIND 9 named-xfer нет. Разработчики рекомендуют использовать dig, если очень нужно получить зону по axfr запросу. Есть только один момент. Программы типа dig, например, host, в точности следуют спецификациям и копируют все что получают. А в качестве ответа на запрос типа axfr запись SOA передается в начале и в конце отклика. Естественно, что конечную запись желательно из файла удалить.

В принципе, slave сервер может не хранить копию зоны у себя в файловой системе. Эта копия нужна только в момент старта. Ее основное назначение - сокращение объем DNS трафика.

Копия не создается, если в директиве secondary BIND 4.x не задано имя файла, или в BIND 8.x и 9.x не задана опция file, которая выполняет аналогичное назначение.

Общие рекомендации, содержащиеся в документации по BIND, советуют все-таки копию описания зоны в файловой системе slave сервера создавать.

В заключении разберем вопрос о проверке работоспособности нашего сервера. Проверить это очень просто: после запуска сервера в рабочем каталоге named должен появиться соответствующий файл.

Можно, конечно, посмотреть работу сервера и при помощи программы dig - отклик должен быть авторитативный.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
3. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
4. D. Barr. RFC 1912. Common DNS Operational and Configuration Errors. 1996. (http://www.ietf.org/rfc/rfc1912.txt?number=1912)
5. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

6. BIND Configuration File Guide (8.3.4) (ftp://ftp.isc.org/isc/bind/src/8.3.4/bind-doc.tar.gz)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind8.html - страничка BIND 8.
2. http://www.isc.org/products/BIND/bind4.html - страничка BIND 4.9.11
3. http://www.acmebw.com/resources/papers/securing.pdf - Securing an Internet Name Server. CERT Coordination Center. Carnegie Mellon University. 2002. Довольно подробный и обстоятельный обзор возможных проблем безопасности серверов доменных имен с рекомендациями по их (серверов) конфигурации.

8. d. "Прямая" и "обратная" зоны для домена, определенного на двух адресных пулах типа x.x.x.x/24 или организация обратных зон на "суперсетях" класса C

В данном материале мы рассмотрим конфигурацию программы named при организации сервера домена, чьи хосты распределены по двум физическим IP-сетям класса C (в нотации CIDR x.x.x.x/24). Основное внимание будет уделено "обратным" зонам, т.к. "прямая" зона в этом случае не имеет существенных отличий от зоны, рассмотренной при описании небольшого корпоративного домена.

Ситуация, рассматриваемая в данном случае, характерна для организаций, которые имеют более одной сети класса C, где необходимо развернуть корпоративный домен. Если быть более точным, то речь идет не только о сетях класса C.

Предположим, что адресные пулы, которые выделены организации и ее подразделениям, представляют из себя не единое адресное пространство, а нарезку из нескольких блоков адресов. При этом эти блоки нарезаны таким образом, что адреса оказываются из разных областей, если рассматривать адресное пространство с точки зрения нотации CIDR x.x.x.x/24. Например:

192.168.0.0/24 и 192.168.10.0/24

С точки зрения описания соответствия между доменным именем и IP-адресом в "прямой" зоне здесь проблем нет:

```
$ORIGIN ru.
test IN SOA ns.test.ru. hostmaster.test.ru (
233 3600 300 9999999 3600 )
;
IN NS ns.test.ru.
IN NS ns.privider.ru.
IN A 192.168.10.1
IN MX 10 mail.test.ru.
IN MX 20 mail.provider.ru.
;
ns IN A 192.168.10.1
```

mail IN A 192.168.0.1
www IN A 192.168.10.2
ftp IN A 192.168.0.2

Из примера видно, что адресные записи могут прекрасно "перемешиваться" в описании зоны. Таким образом, прямую зону можно определить на любом множестве наборов адресов, которые могут быть, как угодно разбросаны по адресному пространству.

Конечно, есть адреса, которые нельзя мешать. Например, нельзя мешать маршрутизируемые и немаршрутизируемые адреса. Собственно, в примере мы используем именно последние (подробнее о немаршрутизируемых адресах см. RFC 1918).

Если запросить из Интернет IP-адрес по доменному имени и в ответ получить адрес из немаршрутизируемого пула, то не понятно, что с ним делать. Даже если вы сами находитесь внутри немаршрутизируемой сети полученный снаружи адрес из этой же сети, скорее всего, не является искомым адресом.

На самом деле, один и тот же сервер доменных имен может поддерживать как маршрутизируемые соответствия, так и немаршрутизируемые, но этот случай для простоты изложения лучше оставить до отдельного разбора в другом материале.

И так, в "прямой" зоне мы можем "мешать" адреса, но вот как поддерживать обратные соответствия? Ведь в случае "обратных" зон мы имеем дело тоже с доменными именами, хотя они и образованы инверсией IP-адресов. Разделителем в иерархии именования доменов является символ ".", следовательно, границы байтов в адресе будут соответствовать границам вложенности доменов.

Выход простой - создать две обратных зоны 0.168.192.in-addr.arpa и 10.168.192.in-addr.arpa. Выглядеть это будет примерно так:

$TTL 3600
$ORIGIN 168.192.in-addr.arpa.
10 IN SOA ns.test.ru. hostmaster.test.ru. (
75 3600 300 9999999 3600)
IN NS ns.test.ru.
IN NS ns.privider.ru.
1 IN PTR ns.test.ru.
2 IN PTR www.test.ru.

И для 0.168.192.in-addr.arpa. соответственно:

$TTL 3600
$ORIGIN 168.192.in-addr.arpa.
0 IN SOA ns.test.ru. hostmaster.test.ru. (
75 3600 300 9999999 3600)
IN NS ns.test.ru.
IN NS ns.privider.ru.
1 IN PTR ns.test.ru.
2 IN PTR www.test.ru.

На master сервере должно быть объявлено две "обратных" зоны. В BIND 4.x в файле named.boot это будет выглядеть примерно так:

```
directory /etc/namedb
primary test.ru test.ru
primary localhost localhost
primary 127.in-addr.arpa localhost.rev
primary 10.168.192.in-addr.arpa 10.168.192.in-addr.arpa
primary 0.168.192.in-addr.arpa 0.168.192.in-addr.arpa
xfrnets 192.168.20.1&255.255.255.255
cache . named.root
options no-recursion no-fetch-glue fake-iquery
```

Собственно, важным с точки зрения контекста данного материала являтся наличие среди директив управления двух директив primary для соответствующих обратных зон.

Здесь стоит только пояснить, что в данном случае адрес 192.168.20.1 - это адрес slave сервера, которому разрешено копировать зону. Назначение остальных директив управления подробно рассмотрено в "Небольшой корпоративный домен в домене ru. Описание "прямых" зон. Описание "обратных" зон. Настройки BIND.".

Что же касается файла named.conf версий BIND 8.x и 9.x, то его содержание будет выглядеть примерно так:

```
options {
directory "/etc/namedb";
allow-query {any;};
recursion no;
fake-iquery yes;
fetch-glue no;
use-id-pool yes;
};
//Root server hints
zone "." { type hint; file "named.root"; };
// Forward Loopback
zone "localhost" {
type master;
file "localhost";
};
// Reverse Loopback
zone "0.0.127.in-addr.arpa" {
type master;
file "localhosr.rev";
};
// Corporative domain
zone "test.ru" {
type master;
file "test.ru";
allow-transfer { 192.168.20.1; };
};
// Reverse corporative domain
zone "0.168.192.in-addr.arpa" {
type master;
file "0.168.192.in-addr.arpa";
allow-transfer { 192.168.20.1; };
};
```

```
// Reverse corporative domain
zone "10.168.192.in-addr.arpa" {
type master;
file "10.168.192.in-addr.arpa";
allow-transfer { 192.168.20.1; };
};
```

Это описание также содержит две директивы для обратных зон, на которые отображаются имена. Описание несколько более длинное, чем для BIND 4.x в силу иного формата файла конфигурации, но суть его та же.

Здесь следует отметить, что несколько обратных зон появляются, например, и для сетей типа x.x.x.x/23. Вся штука в том, что, адресный пул, например, 192.168.0.0.23, объединяет два смежных блока 192.168.0.0/24 и 192.168.1.0/24. Соответствующих обратных зон, следовательно, будет две: 0.168.192.in-addr.arpa и 1.168.192.in-addr.arpa. Объединить их стандартным образом можно только на уровне 168.192.in-addr.arpa, но никак не ниже.

Из выше сказанного, следует, что владелец зоны 168.192.in-addr.arpa должен делегировать ответственность за управления двумя обратными зонами своему клиенту, если не хочет управлять ими самостоятельно.

Аналогичные замечания справедливы и для адресных пулов x.x.x.x/16 и для адресных пулов x.x.x.x.8, т.е. сетей классов B и A соответственно. Пространство доменных имен "обратных" зон построено с учетом старой классификации адресов, в то время, когда нотация CIDR широко еще не использовалась.

В документе RFC 1519 подробно разбирается отображение адресного пространства CIDR на "суперсети" сетей класса C, т.е. пулов адресов, которые составлены из подсетей сетей класса B и A. Провайдер в этом случае должен делегировать соответствующие обратные зоны клиентам, а те обеспечить их поддержку способом, похожим на случай 192.168.0.0/23, рассмотренный выше.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
3. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
4. BIND 9 Administrator Reference Manual. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)
5. BIND Configuration File Guide (8.3.4) (ftp://ftp.isc.org/isc/bind/src/8.3.4/bind-doc.tar.gz)
6. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot. RFC 1918. Address Allocation for Private Internets. 1996. (http://www.ietf.org/rfc/rfc1918.txt?number=1918)
7. Y. Rekhter, T. Li. RFC 1518. An Architecture for IP Address Allocation with CIDR. 1993. (http://www.ietf.org/rfc/rfc1518.txt?number=1518)
8. V. Fuller, T. Li, J. Yu, K. Varadhan. RFC 1519. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. 1993. (http://www.ietf.org/rfc/rfc1519.txt?number=1519)

Полезные ссылки:

1. http://www.isc.org/products/BIND/bind8.html - страничка BIND 8.
2. http://www.isc.org/products/BIND/bind4.html - страничка BIND 4.9.11
3. http://www.acmebw.com/resources/papers/securing.pdf - Securing an Internet Name Server. CERT Coordination Center. Carnegie Mellon University. 2002. Довольно подробный и обстоятельный обзор возможных проблем безопасности серверов доменных имен с рекомендациями по их (серверов) конфигурации.

9. Тестирование серверов и системы доменных имен

Этот материал пытается дать ответ на вопрос - "за чем нужно, вообще, тестировать систему доменных имен и серверы доменных имен". Названы типовые запросы к системе доменных имен и к серверам системы доменных имен.

В тот момент, когда получен корпоративный домен, осуществлено делегирование зоны корпоративного домена, настроены и запущены серверы зоны, наступает следующий этап работы администратора домена - этап эксплуатации.

Главным на этапе эксплуатации становится контроль работы серверов доменных имен зоны и "видимость" зоны (как совокупности хостов) в пространстве доменных имен. При этом, как правило, администратор решает несколько стандартных задач.

Первая стандартная задача - это поиск IP-адреса хоста по доменному имени. При этом следует иметь в виду что, если мы ищем свой собственный только что добавленный в зону хост, то он может быть виден при использовании локального сервера доменных имен, т.к. последний отвечает за зону, в которую вносятся изменения, но не виден для всех остальных пользователей сети.

Это происходит по той простой причине, что slave серверы к моменту проведения поиска еще не обновили зону. Поэтому нужно попытаться использовать для поиска какой-либо "чужой" сервер для исполнения вашего рекурсивного запроса. Хост должен стать "видимым" после обновления описаний зон на всех серверах, ответственных за зону.

Вторая стандартная задача - поиск доменного имени хоста по IP-адресу. Довольно часто внося изменения в "прямую" зону, забывают внести изменения в "обратную" зону. Иногда это делают преднамеренно, если IP-адреса назначаются динамически. В последнее время часто используют поддержку динамических обновлений для DHCP. В этом случае также нужно убедиться в аккуратности ведения обратной зоны и внесения обновлений.

Третья стандартная задача, которая возникает в контексте почтовых проблем, - это перебор имен resolver-ом. На самом деле, здесь существует масса подводных камней связанных и с почтовым программным обеспечением и с версией пакета BIND.

Четвертая стандартная задача - передача описания зоны. Она возникает при выявлении рассогласования между master сервером зоны и slave серверами. Кроме того, "скачивание" всей зоны, как правило, разрешено далеко не всем, что тоже требует проверки.

Пятая стандартная задача - это выявление типовых ошибок при управлении зоной, к которым обычно относят: некорректное делегирование зон, проблемы установки синонимов на канонические имена в контексте MX и NS записей описания зоны, "проброс" запросов на несуществующие корпоративные домены различным прикладным ПО.

Шестая стандартная задача - получение отладочных отчетов о работе системы доменных имен и выполнении рекурсивных запросов вашим сервером доменных имен. Кроме того, всегда хочется иметь надежную информацию об исполнении динамических обновлений описания зоны или об уведомлениях об изменении зоны, которые ваш сервер рассылает.

Седьмая стандартная задача - это анализ запросов к вашему серверу на предмет его загруженности и безопасности. Оба этих вопроса тесно связаны. Чрезмерная загруженность сервера может сделать его "невидимым" в Интернет, что в свою очередь является в большинстве случаев основной целью злоумышленников.

Мы не претендуем на полноту предложенного выше списка. Тем не менее, на наш взгляд основные проблемы, на которые следует обратить внимание, в нем перечислены. Современные средства тестирования DNS позволяют тем или иным способом решить все задачи, порожденные этими проблемами.

Наиболее известными средствами тестирования работы системы DNS и серверов среди администраторов системы DNS и пользователей сети являются: nslookup, host и dig.

Nslookup является старшей из всех перечисленных средств тестирования DNS. Ее основное преимущество - наличие практически во всех операционных системах (В Windows 9x, ее, правда, нет, но на серверных платформах Microsoft nslookup присутствует). Программа работает в интерактивном и неинтерактивном режимах. Для ее применения, в прочем, как и для host с dig, нужно быть готовым к работе в командной строке.

Основной недостаток nslookup - невозможность работы с новыми возможностями серверов (ixfr, например). Однако, для исполнения основных функций администратора домена она вполне пригодна.

Про dig можно, наверное, сказать, что это основное средство отладки BIND. Программа поставляется в одном дистрибутиве с серверами BIND. Единственная программа из нашей тройки, которая продолжает развиваться и совершенствоваться. Однако, очевиден ее уклон сторону тестирования named. В отчетах много важной с точки зрения отладки ПО информации, в частности возможность контроля флагов заголовков DNS сообщений.

Host интересна своей направленностью на удовлетворение нужд администраторов зон, а не разработчиков серверов. В ней реализованы дополнительные возможности по составлению отчетов по зонам. RIPE, например, использует Host для сбора и опубликования статистики по зонам своей ответственности. К сожалению, с более свежую версию программы, чем версия от 2000 года в сети найти, видимо, не удастся. Во всяком случае, у нас это не получилось.

Существует еще несколько средств тестирования DNS для разных платформ, направленных на облегчение жизни администраторов, но все они менее популярны, чем nslookup, host и dig.

Рекомендованная литература:

1. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
2. A. Romao. RFC 1713. Tools for DNS debugging. 1994. (http://www.ietf.org/rfc/rfc1714.txt?number=1713)

Полезные ссылки:

1. http://ciisa.isa.utl.pt/cgi-bin/man2html?dig+1 - справочное руководство по dig в стиле Unix manual. Следует иметь в виду, что версия с которой вы работаете можно довольно сильно отличаться от этого описания. По этой причине лучше использовать тот manual, который поставляется с дистрибутивом named и dig.
2. http://tom.imm.uran.ru/cgi-bin/man?nslookup(8) - страница man по nslookup.
3. http://ciisa.isa.utl.pt/cgi-bin/man2html?host+1 - страница man host для операционной системы Linux. Это не полный список ключей, но кое что есть
4. Salamon, Andr‡s. "Tools to manage DNS." 1999.
   URL: http://www.dns.net/dnsrd/tools.html

9. а. Программа интерактивного тестирования серверов DNS - nslookup

В данном материале рассматриваются различные режимы работы программы nslookup - одного из основных средств тестирования работы сервера доменных имен.

Среди средств тестирования системы доменных имен nslookup подвергается наибольшей критике. Во всяком случае, авторы bind рекомендуют использовать dig вместо nslookup, как написано в руководстве по BIND 9 - "из-за загадочности пользовательского интерфейса и часто противоречивого поведения" последней. Тем не менее, nslookup - это одно из самых популярных средств тестирования DNS. Эта программа есть в большинстве версий Unix-систем.

Программа позволяет работать пользователю в двух режимах: интерактивном и режиме неинтерактивном режиме. В первом случае пользователь, попав в командную строку nslookup, имеет возможность исполнять команды nslookup, во втором случае отчеты получают, задавая аргументы командной строки интерпретатора (shell или любого другого командного интерпретатора) nslookup.

Сначала рассмотрим неинтерактивный режим работы. Как уже было сказано, он используется для получения информации из системы DNS непосредственно из командной строки интерпретатора. В аргументах командной строки nslookup можно указывать значения опций, имя или адрес хоста для которого производится поиск, и сервер доменных имен.

```
% nslookup www.ru
Server: polyn.net.kiae.su
Address: 144.206.160.32

Non-authoritative answer:
Name: www.ru
Address: 194.87.0.50

%
```

В данном случае мы просто используем сервер доменных имен из resolv.conf для поиска IP-адреса www.ru. Из отчета видим, что этот сервер не является авторитативным для искомого имени.

Для того, чтобы понять выполняет ли nslookup перебор доменных имен, который характерен для библиотеки resolver, зададим имя несуществующего хоста:

```
% nslookup gaga.ru
Server: polyn.net.kiae.su
Address: 144.206.160.32

Name: gaga.ru.net.kiae.su

%
```

За нашим именем появилась вставка net.kiae.su, которая взята из resolv.conf. Вот его (resolv.conf) содержание:

```
% more /etc/resolv.conf
nameserver 144.206.160.32
nameserver 144.206.192.10
search net.kiae.su polyn.kiae.su .
%
```

На самом деле можно получить более полный отчет, если включить режим отладки:

```
% nslookup -debug gaga.ru
;; res_mkquery(0, 32.160.206.144.in-addr.arpa, 1, 12)
------------
Got answer:
HEADER:
opcode = QUERY, id = 22021, rcode = NOERROR
header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 1, authority records = 3, additional = 3

QUESTIONS:
32.160.206.144.in-addr.arpa, type = PTR, class = IN
ANSWERS:
-> 32.160.206.144.in-addr.arpa
name = polyn.net.kiae.su
ttl = 3600 (1 hour)
AUTHORITY RECORDS:
-> 160.206.144.in-addr.arpa
nameserver = polyn.net.kiae.su
ttl = 3600 (1 hour)
-> 160.206.144.in-addr.arpa
nameserver = ns.spb.su
ttl = 3600 (1 hour)
-> 160.206.144.in-addr.arpa
nameserver = ns.ussr.eu.net
ttl = 3600 (1 hour)
ADDITIONAL RECORDS:
-> polyn.net.kiae.su
internet address = 144.206.160.32
ttl = 49698 (13 hours 48 mins 18 secs)
-> ns.spb.su
internet address = 193.124.83.69
```

ttl = 49556 (13 hours 45 mins 56 secs)
-> ns.ussr.eu.net
internet address = 193.125.152.3
ttl = 3083 (51 mins 23 secs)

Server: polyn.net.kiae.su
Address: 144.206.160.32

;; res_mkquery(0, gaga.ru, 1, 1)

Got answer:
HEADER:
opcode = QUERY, id = 22022, rcode = NXDOMAIN
header flags: response, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
gaga.ru, type = A, class = IN
AUTHORITY RECORDS:
-> ru
ttl = 10530 (2 hours 55 mins 30 secs)
origin = ns.ripn.net
mail addr = hostmaster.ripn.net
serial = 4005295
refresh = 7200 (2 hours)
retry = 900 (15 mins)
expire = 2592000 (30 days)
minimum ttl = 345600 (4 days)

;; res_mkquery(0, gaga.ru.net.kiae.su, 1, 1)

Got answer:
HEADER:
opcode = QUERY, id = 22023, rcode = NOERROR
header flags: response, want recursion, recursion avail.
questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
gaga.ru.net.kiae.su, type = A, class = IN
AUTHORITY RECORDS:
-> kiae.su
ttl = 10530 (2 hours 55 mins 30 secs)
origin = ns.kiae.ru
mail addr = noc-dns.relarn.ru
serial = 650127450
refresh = 28800 (8 hours)
retry = 3600 (1 hour)
expire = 604800 (7 days)
minimum ttl = 86400 (1 day)

```
------------
Name: gaga.ru.net.kiae.su

%
```

О чем говорит этот отчет. Сначала производится поиск по обратной зоне имени сервера доменных имен, который имеет IP-адрес 144.206.160.32. Затем производится поиск gaga.ru, а затем gaga.ru.net.kiae.su. По идее, следовало бы выполнить и другие подстановки, но nslookup этого не сделала. И вообще, в данном случае программа ведет себя несколько не так, как это предписано для системы resolver.

В данном случае аргумент -debug - это опция командной строки, которая позволяет управлять объемом и типом информации в отчетах nslookup при неинтерактивном режиме работы.

Вот еще один пример применения опционного аргумента:

```
% nslookup -type=SOA ru.
Server: polyn.net.kiae.su
Address: 144.206.160.32

Non-authoritative answer:
ru
origin = ns.ripn.net
mail addr = hostmaster.ripn.net
serial = 4005295
refresh = 7200 (2 hours)
retry = 900 (15 mins)
expire = 2592000 (30 days)
minimum ttl = 345600 (4 days)

Authoritative answers can be found from:
ru nameserver = NS2.NIC.FR
ru nameserver = ns.ripn.net
ru nameserver = NS2.ripn.net
ru nameserver = SUNIC.SUNET.SE
ru nameserver = NS.UU.net
ru nameserver = NS1.RELCOM.ru
NS2.NIC.FR internet address = 192.93.0.4
ns.ripn.net internet address = 194.85.119.1
NS2.ripn.net internet address = 194.226.96.30
SUNIC.SUNET.SE internet address = 192.36.125.2
NS.UU.net internet address = 137.39.1.3
NS1.RELCOM.ru internet address = 193.125.152.3
%
```

В данном случае мы просим доставить нам информацию о записи SOA зоны ru. Точка в конце имени зоны указана для определенности. Сервер выдает не авторитативный ответ и сообщает, где можно получить авторитативные ответы.

Опции всегда указываются перед именем или IP-адресом хоста, для которого ищется информация. На самом деле опции - это аргументы команды set интерактивного режима работы, о котором речь пойдет несколько позже.

Опции можно также указать в файле .nslookuprc, который может быть размещен в домашнем каталоге пользователя. Каждая опция должна занимать отдельную строку. Ниже приведен пример содержания этого файла:

```
% more .nslookuprc
type=NS
db2
%
```

Первая строка задает тип записей описания зоны, которые ищутся. В данном случае это записи NS. А вторая запись задает уровень подробности отчета (отладка - debug второго уровня).

До сих пор мы рассматривали работу nslookup с сервером доменных имен, который определен в resolv.conf. Имя сервера доменных имен, который будет выполнять рекурсивные запросы, можно задать в качестве последнего аргумента командной строки nslookup:

```
% nslookup -type=A www.ru. ns.relarn.ru.
Server: ns.relarn.ru
Address: 194.226.65.3

Non-authoritative answer:
Name: www.ru
Address: 194.87.0.50

%
```

В данном случае мы посылаем рекурсивный запрос не серверу умолчания из resolv.conf, а ns.relarn.ru, который присылает нам неавторитативный ответ.

Теперь разберем интерактивный режим работы nslookup. В него попадают двумя способами. Во-первых, просто введя nslookup в командной строке интерпретатора:

```
% nslookup
Default Server: polyn.net.kiae.su
Address: 144.206.160.32

>
```

В данном случае, в качестве сервера доменных имен программа использует сервер доменных имен умолчания, который указывается в файле настройки resolver (resolv.conf).

Во-вторых, в интерактивный режим можно попасть, задав nslookup с двумя аргументами: символом "-" и именем или IP-адресом сервера доменных имен, который будет использоваться для выполнения рекурсивных запросов:

```
% nslookup - 144.206.192.10
Default Server: IRIS.polyn.kiae.su
Address: 144.206.192.10

>
```

Для того, чтобы покинуть интерактивный режим nslookup и вернуться в командную строку интерпретатора следует выполнить команду exit:

```
% nslookup
Default Server: polyn.net.kiae.su
Address: 144.206.160.32
> exit
%
```

Вообще говоря, у nslookup довольно большое множество команд интерактивного режима, но мы их все рассматривать не будем. Для этого существуют руководства и система man. Рассмотрим только наиболее часто используемые случаи.

Самый простой из них - это поиск IP-адреса по доменному имени:

```
> quest.polyn.kiae.su.
Server: polyn.net.kiae.su
Address: 144.206.160.32

Name: quest.polyn.kiae.su
Address: 144.206.192.2

>
```

Лучше всего задавать на конце доменного имени символ ".". По умолчанию nslookup включает перебор доменных имен, а работает он не совсем корректно, как мы уже убедились выше.

Теперь попробуем найти обратное соответствие:

```
> 144.206.192.11
Server: polyn.net.kiae.su
Address: 144.206.160.32

Name: www.kiae.ru
Address: 144.206.192.11

>
```

Как мы видим из полученного отчета, этому адресу соответствует имя www.kiae.ru. Увеличим подробность отчета:

```
> set debug
> 144.206.192.11
Server: polyn.net.kiae.su
Address: 144.206.160.32

;; res_mkquery(0, 11.192.206.144.in-addr.arpa, 1, 12)
------------
Got answer:
HEADER:
opcode = QUERY, id = 31436, rcode = NOERROR
```

header flags: response, auth. answer, want recursion, recursion avail.
questions = 1, answers = 2, authority records = 3, additional = 2

QUESTIONS:
11.192.206.144.in-addr.arpa, type = PTR, class = IN
ANSWERS:
-> 11.192.206.144.in-addr.arpa
name = www.kiae.ru
ttl = 3600 (1 hour)
-> 11.192.206.144.in-addr.arpa
name = kiae.polyn.kiae.su
ttl = 3600 (1 hour)
AUTHORITY RECORDS:
-> 192.206.144.in-addr.arpa
nameserver = polyn.net.kiae.su
ttl = 3600 (1 hour)
-> 192.206.144.in-addr.arpa
nameserver = ns.spb.su
ttl = 3600 (1 hour)
-> 192.206.144.in-addr.arpa
nameserver = ns.ussr.eu.net
ttl = 3600 (1 hour)
ADDITIONAL RECORDS:
-> polyn.net.kiae.su
internet address = 144.206.160.32
ttl = 46562 (12 hours 56 mins 2 secs)
-> ns.spb.su
internet address = 193.124.83.69
ttl = 46420 (12 hours 53 mins 40 secs)

Name: www.kiae.ru
Address: 144.206.192.11

>

Подробность отчета мы увеличиваем командой set debug. Из полученного отчета видно, что nslookup производит поиск в обратной зоне, предварительно преобразовав соответствующим образом введенную нами строку IP-адреса.

Обратим внимание еще на один момент. Если при поиске IP-адреса тип запроса был A, то, как видно из выше приведенного примера, в случае поиска доменного имени по IP-адресу мы посылаем запрос на указатель (PTR).

Среди ответов мы находим два имени, которые соответствуют IP-адресу: www.kiae.ru и kiae.polyn.kiae.ru. Это отражает тот факт, что в описании обратной зоны 192.206.144.in-addr.arpa у нас есть что-то похожее на:

11 IN PTR www.kiae.ru.
IN PTR kiae.polyn.kiae.su.

Если теперь мы захотим получить параметры SOA записи для этой обратной зоны, то нам следует заказать именно эту запись описания зоны:

```
> set type=soa
> 192.206.144.in-addr.arpa.
Server: polyn.net.kiae.su
Address: 144.206.160.32

192.206.144.in-addr.arpa
origin = polyn.net.kiae.su
mail addr = paul.kiae.su
serial = 80
refresh = 3600 (1 hour)
retry = 300 (5 mins)
expire = 9999999 (115 days 17 hours 46 mins 39 secs)
minimum ttl = 3600 (1 hour)
192.206.144.in-addr.arpa nameserver = polyn.net.kiae.su
192.206.144.in-addr.arpa nameserver = ns.spb.su
192.206.144.in-addr.arpa nameserver = ns.ussr.eu.net
polyn.net.kiae.su internet address = 144.206.160.32
ns.spb.su internet address = 193.124.83.69
ns.ussr.eu.net internet address = 193.125.152.3
>
```

На самом деле мы получили не только параметры записи SOA, но и информацию о серверах доменных имен, которые эту зону поддерживают.

Приведенный выше отчет ни чем не отличается от того, который мы получили бы, если вместо обратной зоны использовали имя прямой зоны, например, polyn.kiae.su:

```
> set type=soa
> polyn.kiae.su.
Server: polyn.net.kiae.su
Address: 144.206.160.32

polyn.kiae.su
origin = polyn.net.kiae.su
mail addr = paul.kiae.su
serial = 80
refresh = 3600 (1 hour)
retry = 300 (5 mins)
expire = 9999999 (115 days 17 hours 46 mins 39 secs)
minimum ttl = 3600 (1 hour)
192.206.144.in-addr.arpa nameserver = polyn.net.kiae.su
192.206.144.in-addr.arpa nameserver = ns.spb.su
192.206.144.in-addr.arpa nameserver = ns.ussr.eu.net
polyn.net.kiae.su internet address = 144.206.160.32
ns.spb.su internet address = 193.124.83.69
ns.ussr.eu.net internet address = 193.125.152.3
>
```

Внимательный читатель обнаружит в приведенных выше примерах чудовищно большое время автономной работы для slave серверов для обеих зон (Устанавливается в SOA записи). Лучше всего следовать рекомендациям RIPE-203.

Рассмотрим еще два параметра интерактивного режима: server и lserver. Они нужны для изменения сервера доменных имен, которому nslookup посылает рекурсивные запросы (текущий сервер):

```
origin 16% nslookup
Default Server: IRIS.polyn.kiae.su
Address: 144.206.192.10

>
```

В данном случае в nslookup работает в качестве клиента и посылает рекурсивные запросы на сервер IRIS.polyn.kiae.su (144.206.192.10). Мы можем изменить этот сервер при помощи команд server или lserver:

```
> server 144.206.160.32
Default Server: polyn.net.kiae.su
Address: 144.206.160.32

>
```

В данном случае мы меняем адрес текущего сервера, используя текущий же сервер. Но текущий сервер может не отвечать:

```
> server 144.206.130.137
Default Server: [144.206.130.137]
Address: 144.206.130.137

> 144.206.160.1
Server: [144.206.130.137]
Address: 144.206.130.137
```

В этом месте мы "залипли", т.к. по адресу 144.206.130.137 до сервера доменных имен мы достучаться не можем (он там когда-то был, но сейчас его уже там нет). Нам нужно поменять текущий сервер, т.е. тот, которые мы используем для рекурсивных запросов:

```
> lserver 144.206.160.32
Default Server: polyn.net.kiae.su
Address: 144.206.160.32

> 144.206.160.1
Server: polyn.net.kiae.su
Address: 144.206.160.32

*** polyn.net.kiae.su can't find 144.206.160.1: Non-existent host/domain
>
```

Перед тем, как выдать команду мы вернулись в режим командной строки (^c) и потом выдали команду на изменение текущего сервера доменных имен.

Следует признать, что отклик nslookup на наш запрос не очень информативен. На самом деле просто в обратной зоне нет записи, которая бы устанавливала соответствие между IP адресом 144.206.160.1 и доменным именем. Впрочем, и более "свежие" программы тестирования DNS не более информативны. Вот пример отчета команды host:

```
> host 144.206.160.32
32.160.206.144.IN-ADDR.ARPA domain name pointer polyn.net.kiae.su
> host 144.206.160.1
Host not found.
>
```

В случае присутствия PTR записи, она выдается, при ее отсутствии - "хост не найден".

Теперь рассмотрим еще один важный момент тестирования - передачу зоны. В этом случаи nslookup выступает как slave сервер, которые "срисовывает" зону с master сервера. Делается это при помощи команды интерактивного режима ls:

```
> ls bard.kiae.ru
[IRIS.polyn.kiae.su]
$ORIGIN bard.kiae.ru.
@ 1H IN A 144.206.192.32
mail 1H IN A 144.206.192.32
www 1H IN A 144.206.192.32
>
```

На самом деле по команде ls в данном случае мы получили в отчете только адресные записи. Для того, чтобы получить все записи нужно указать либо флаг "-d", либо "-t any":

```
> ls -d bard.kiae.ru
[IRIS.polyn.kiae.su]
$ORIGIN bard.kiae.ru.
@ 1H IN SOA ns.polyn.kiae.ru. paul.polyn.kiae.su. (
7 ; serial
1H ; refresh
5M ; retry
16w3d17h46m39s ; expiry
1H ) ; minimum

1H IN NS polyn.kiae.ru.
1H IN NS iris.polyn.kiae.su.
1H IN MX 10 mail
1H IN MX 20 iris.polyn.kiae.ru.
1H IN A 144.206.192.32
mail 1H IN A 144.206.192.32
1H IN MX 10 mail
www 1H IN A 144.206.192.32
@ 1H IN SOA ns.polyn.kiae.ru. paul.polyn.kiae.su. (
7 ; serial
1H ; refresh
5M ; retry
16w3d17h46m39s ; expiry
1H ) ; minimum

>
```

Для того, чтобы списать зону, необходимо разрешение. Это значит, что администратор зоны должен разрешить списывание зоны с той машины, на которой Вы выполняете nslookup.

Следует еще раз напомнить, что nslookup считается не самым лучшим средством тестирования работы системы DNS и настройки серверов. Тем не менее, эта программа есть на любой Unix-платформе, и по этой причине ее применяют достаточно часто.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. Peter Koch. Ripe-203. Recommendations for DNS SOA Values. 1999. (http://www.ripe.net/docs/ripe-203.html)
2. http://tom.imm.uran.ru/cgi-bin/man?nslookup(8) - страница man по nslookup.

9. b. Программа тестирования системы доменных имен - host

В данном материале рассматривается применение различных опций программы host - одного из основных средств тестирования работы системы доменных имен.

Программа host среди трех наиболее используемых средств тестирования DNS (nslookup, dig, host) самая "свежая". В ней учтены нарекания пользователей, которые вызывают ее аналоги - nslookup и dig, а также реализованы возможности диагностирования ошибок и подсчета статистики.

Host не является средством отладки серверов доменных имен, каковым по сути является dig. Host не имеет интерактивного режима работы, который присутствует в nslookup. Зато host проста в использовании, не перегружена опциями и атрибутами, имеет ясный и лаконичный синтаксис и отчетность. Кроме того, ее любят применять в качестве компонента скриптов, которые кроме всего прочего призваны проверять наличие или отсутствие описания хостов в системе DNS.

Самый простой пример использования Host - это поиск IP-адреса по доменному имени:

```
> host quest
quest.polyn.kiae.su has address 144.206.192.2
quest.polyn.kiae.su mail is handled (pri=10) by quest.polyn.kiae.su
quest.polyn.kiae.su mail is handled (pri=20) by relay1.relcom.ru
>
```

В данном случае мы задали короткое имя хоста. Оно было расширено именем домена, в котором находится хост, с которого осуществляется запрос к системе доменных имен (polyn.kiae.su). В качестве отклика мы получили не только IP-адрес, но и информацию о том, как на хост polyn.kiae.su доставляется почта, т.е. какие хосты являются почтовыми шлюзами.

При решении обратной задачи, поиска имени по IP-адресу можно также воспользоваться программой host:

```
> host 144.206.192.2
2.192.206.144.IN-ADDR.ARPA domain name pointer quest.polyn.kiae.su
>
```

Как мы видим из этого примера, отчет host гораздо лаконичнее, чем отчет nslookup, например, - это просто сообщение о наличии записи в обратной зоне. Если бы с данным адресом, а точнее с именем 2.192.206.144.in-addr.arpa, в обратной зоне было бы связано более одной записи, то host сообщила бы о наличии всех этих записей:

```
> host 144.206.192.11
11.192.206.144.IN-ADDR.ARPA domain name pointer www.kiae.ru
11.192.206.144.IN-ADDR.ARPA domain name pointer kiae.polyn.kiae.su
>
```

Host принимает в качестве аргументов не только короткие имена, но и более длинные последовательности, которые могут быть как полными именами (FQDN), так и неполными именами:

```
> host quest.polyn
quest.polyn.kiae.su mail is handled (pri=20) by relay1.relcom.ru
quest.polyn.kiae.su mail is handled (pri=10) by quest.polyn.kiae.su
>
```

В контексте этого примера интересно рассмотреть, как host перебирает доменные имена и, вообще, что реально происходит. Для этого воспользуемся опцией отладки "-d":

```
> host -d quest.polyn
;; res_nmkquery(QUERY, quest.polyn, IN, A)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56773
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; quest.polyn, type = A, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 56773
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; quest.polyn, type = A, class = IN
. 9m48s IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISIGN-GRS.C
OM. (
2002110200 ; serial
30M ; refresh
15M ; retry
1W ; expiry
1D ) ; minimum

rcode = 3 (Non-existent domain), ancount=0
;; res_nmkquery(QUERY, quest.polyn, IN, MX)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56774
```

;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; quest.polyn, type = MX, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 56774
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; quest.polyn, type = MX, class = IN
. 9m48s IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISIGN-GRS.C
OM. (
2002110200 ; serial
30M ; refresh
15M ; retry
1W ; expiry
1D) ; minimum

rcode = 3 (Non-existent domain), ancount=0
;; res_nmkquery(QUERY, quest.polyn.polyn.kiae.su, IN, MX)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56775
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; quest.polyn.polyn.kiae.su, type = MX, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 56775
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; quest.polyn.polyn.kiae.su, type = MX, class = IN
polyn.kiae.su. 1H IN SOA polyn.net.kiae.su. paul.kiae.su. (
233 ; serial
1H ; refresh
5M ; retry
16w3d17h46m39s ; expiry
1H) ; minimum

rcode = 3 (Non-existent domain), ancount=0
;; res_nmkquery(QUERY, quest.polyn.kiae.su, IN, MX)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56776
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; quest.polyn.kiae.su, type = MX, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56776
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5
;; quest.polyn.kiae.su, type = MX, class = IN
quest.polyn.kiae.su. 1H IN MX 10 quest.polyn.kiae.su.
quest.polyn.kiae.su. 1H IN MX 20 relay1.relcom.ru.
polyn.kiae.su. 1H IN NS polyn.net.kiae.su.
polyn.kiae.su. 1H IN NS ns.spb.su.
polyn.kiae.su. 1H IN NS ns.ussr.eu.net.
quest.polyn.kiae.su. 1H IN A 144.206.192.2
relay1.relcom.ru. 21h21m56s IN A 193.125.152.57

polyn.net.kiae.su. 18h51m35s IN A 144.206.160.32
ns.spb.su. 20h28m28s IN A 193.124.83.69
ns.ussr.eu.net. 20h20m50s IN A 193.124.22.65
rcode = 0 (Success), ancount=2
quest.polyn.kiae.su mail is handled (pri=10) by quest.polyn.kiae.su
quest.polyn.kiae.su mail is handled (pri=20) by relay1.relcom.ru
>

Как видно из этого примера, сначала host просто запрашивает IP-адрес для имени quest.polyn у сервера 144.206.192.10. Получает ответ, что такого домена (polyn) нет. Запрос host посылала рекурсивный, поэтому опрос корневого сервера, ответ которого приведен в примере, был получен сервером 144.206.192.10.

Затем host формирует запрос на поиск MX записи по тому же неполному имени. Естественным образом получаем также отрицательный ответ от корневого сервера об отсутствии домена polyn.

Теперь host расширяет наше неполное имя именем домена по умолчанию. Берет его из resolv.conf. Домена polyn.polyn.kiae.su также не существует.

Теперь host хост расширяет неполное имя только частью имени домена kiae.su. Тип записи при этом не модифицируется и остается равным MX, хотя по умолчанию сначала использовался тип A.

Хост quest в зоне polyn.kiae.su существует и для него есть MX записи, о чем нас с радастью и информируют. Кроме того, мы получаем информацию о том, какие серверы доменных имен являются авторитативными для зоны polyn.kiae.su.

Мягко говоря мы получили не совсем то, что хотели. Для того чтобы получить IP-адрес в данном случае, нужно явно задать тип записи A - "-t a":

> host -t a quest.polyn
quest.polyn.kiae.su has address 144.206.192.2
>

Если попросить host выдать более подробную информацию, то можно получить следующее:

> host -v -t a quest.polyn
Trying null domain
rcode = 3 (Non-existent domain), ancount=0
Trying domain "polyn.kiae.su"
rcode = 3 (Non-existent domain), ancount=0
Trying domain "kiae.su"
rcode = 0 (Success), ancount=1
The following answer is not verified as authentic by the server:
quest.polyn.kiae.su 3600 IN A 144.206.192.2
For authoritative answers, see:
polyn.kiae.su 3600 IN NS ns.spb.su
polyn.kiae.su 3600 IN NS ns.ussr.eu.net
polyn.kiae.su 3600 IN NS polyn.net.kiae.su
Additional information:
ns.spb.su 59905 IN A 193.124.83.69

```
ns.ussr.eu.net 59447 IN A 193.124.22.65
polyn.net.kiae.su 54092 IN A 144.206.160.32
>
```

Таким образом, мы видим ту же самую последовательность проверки зон, но только теперь host не запрашивает MX записи, а пытается получить только адресные записи (A).

Внимательный читатель уже заметил, что вместо флага "-d" мы в последнем примере использовали флаг "-v". Флаг "-d" - это флаг отладочного отчета. В нем (отладочном отчете) расшифрованы значения флагов заголовка сообщений протокола DNS и содержание самих запросов. При флаге "-v" мы просто получаем подробный отчет вместо обычного укороченного отчета.

Теперь посмотрим еще один пример:

```
> host kuku.polyn.kiae.su
kuku.polyn.kiae.su.kiae.su mail is handled (pri=80) by relay1.kiae.su
kuku.polyn.kiae.su.kiae.su mail is handled (pri=90) by relay2.kiae.su
> host -v kuku.polyn.kiae.su.
rcode = 3 (Non-existent domain), ancount=0
Host not found.
> host -v kuku.polyn.kiae.su
Trying null domain
rcode = 3 (Non-existent domain), ancount=0
Trying domain "polyn.kiae.su"
rcode = 3 (Non-existent domain), ancount=0
Trying domain "kiae.su"
rcode = 0 (Success), ancount=0
For authoritative answers, see:
kiae.su 86400 IN SOA ns.kiae.ru noc-dns.relarn.ru(
650127450 ;serial (version)
28800 ;refresh period
3600 ;retry refresh this often
604800 ;expiration period
86400 ;minimum TTL
)
>
```

Сначала мы запросили адреса несуществующего имени. В домене polyn.kiae.su нет хоста с именем kuku. Во всяком случае, в описании зоны его нет точно. Host, тем не менее, в коротком отчете выдает MX записи для этого имени.

Мы задаем на конце имени символ ".", чтобы избежать перебора доменных имен. В этом случае мы получаем ответ, который и должны были получить. Теперь нам интересно, а почему в первом случае появились MX записи. Повторяем расширенный отчет, но без символа "." на конце имени хоста. Мы получаем положительный отклик, который говорит нам, что домен kiae.su существует. Но откуда адресные записи? Смотрим отладку, точнее только последнюю ее часть, которая касается домена kiae.su:

```
;; res_nmkquery(QUERY, kuku.polyn.kiae.su.kiae.su, IN, MX)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28015
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; kuku.polyn.kiae.su.kiae.su, type = MX, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28015
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 6
;; kuku.polyn.kiae.su.kiae.su, type = MX, class = IN
kuku.polyn.kiae.su.kiae.su. 23h52m13s IN MX 90 relay2.kiae.su.
kuku.polyn.kiae.su.kiae.su. 23h52m13s IN MX 80 relay1.kiae.su.
kiae.su. 23h52m13s IN NS ns.kiae.ru.
kiae.su. 23h52m13s IN NS ns2.ripn.net.
kiae.su. 23h52m13s IN NS ns.spb.su.
kiae.su. 23h52m13s IN NS ns.ussr.eu.net.
relay2.kiae.su. 23h52m13s IN A 193.125.152.57
relay1.kiae.su. 23h52m13s IN A 193.125.152.57
ns.kiae.ru. 19h41m6s IN A 194.226.65.4
ns2.ripn.net. 1d14h31m54s IN A 194.226.96.30
ns.spb.su. 16h21m18s IN A 193.124.83.69
ns.ussr.eu.net. 16h13m40s IN A 193.124.22.65
rcode = 0 (Success), ancount=2
kuku.polyn.kiae.su.kiae.su mail is handled (pri=90) by relay2.kiae.su
kuku.polyn.kiae.su.kiae.su mail is handled (pri=80) by relay1.kiae.su
>
```

Теперь понятно, что наш запрос в процессе обработки превратился в запрос MX записи, а для домена kiae.su установлены два почтовых шлюза. Мораль проста - нужно заказывать точный тип записи, и задавать полное доменное имя хоста с магической "." на конце.

До этого момента мы использовали host в качестве имитатора клиента DNS. Теперь попробуем изобразить сервер доменных имен, а точнее slave, который копирует зону:

```
> host -d -l vega.ru.
;; res_nmkquery(QUERY, vega.ru, IN, NS)
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64141
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; vega.ru, type = NS, class = IN
;; Querying server (# 1) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64141
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 3
;; vega.ru, type = NS, class = IN
vega.ru. 19h45m17s IN NS ns2.macomnet.ru.
vega.ru. 19h45m17s IN NS ns.macomnet.ru.
vega.ru. 19h45m17s IN NS ns1.psychol.ras.ru.
ns2.macomnet.ru. 19h45m17s IN A 212.5.66.14
ns.macomnet.ru. 19h45m17s IN A 195.128.64.3
ns1.psychol.ras.ru. 19h14m24s IN A 62.117.88.14
rcode = 0 (Success), ancount=3
Found 1 addresses for ns2.macomnet.ru
Found 1 addresses for ns.macomnet.ru
Found 1 addresses for ns1.psychol.ras.ru
```

```
;; res_nmkquery(QUERY, vega.ru, IN, AXFR)
Trying 212.5.66.14
Server failed, trying next server: Query refused
;; res_nmkquery(QUERY, vega.ru, IN, AXFR)
Trying 195.128.64.3
Server failed, trying next server: Query refused
;; res_nmkquery(QUERY, vega.ru, IN, AXFR)
Trying 62.117.88.14
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
vega.ru name server ns1.psychol.ras.ru
rcode = 0 (Success), ancount=1
vega.ru name server ns.macomnet.ru
rcode = 0 (Success), ancount=1
vega.ru name server ns2.macomnet.ru
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
vega.ru has address 194.67.107.8
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
vega-gw.vega.ru has address 194.67.107.8
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
localhost.vega.ru has address 127.0.0.1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
belti.vega.ru has address 213.59.3.161
rcode = 0 (Success), ancount=1
www.belti.vega.ru has address 213.59.3.161
rcode = 0 (Success), ancount=1
ns1.vega.ru has address 194.67.107.1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
rcode = 0 (Success), ancount=1
ns2.vega.ru has address 194.67.107.1
rcode = 0 (Success), ancount=1
>
```

Из отладочного отчета мы видим, что сначала определяются авторитативные сервера, а потом с них запрашивается передача зоны (AXFR). Но серверы macomnet.ru не дают нам списать зону. Наш запрос ими "отражен". Однако сервер ns1.psychol.ras.ru нам разрешает списать зону.

Любопытно, что именно он и является master домена vega.ru:

```
> host -t soa vega.ru
vega.ru start of authority ns1.psychol.ras.ru serge.psychol.ras.ru(
2001121202 ;serial (version)
7200 ;refresh period
1800 ;retry refresh this often
3600000 ;expiration period
107800 ;minimum TTL
)
>
```

Если бы мы не включали режим отладочного отчета, то получили бы только:

```
> host -l vega.ru
vega.ru name server ns1.psychol.ras.ru
vega.ru name server ns.macomnet.ru
vega.ru name server ns2.macomnet.ru
vega.ru has address 194.67.107.8
vega-gw.vega.ru has address 194.67.107.8
localhost.vega.ru has address 127.0.0.1
belti.vega.ru has address 213.59.3.161
www.belti.vega.ru has address 213.59.3.161
ns1.vega.ru has address 194.67.107.1
ns2.vega.ru has address 194.67.107.1
>
```

На самом деле мы всю зону не скопировали, а точнее host нам не показал всей информации, которая есть в файле описания зоны. Для того чтобы получить эту информацию, нужно задать больше параметров:

```
> host -l -t any vega.ru.
vega.ru start of authority ns1.psychol.ras.ru serge.psychol.ras.ru(
2001121202 ;serial (version)
7200 ;refresh period
1800 ;retry refresh this often
3600000 ;expiration period
107800 ;minimum TTL
)
vega.ru name server ns1.psychol.ras.ru
vega.ru name server ns.macomnet.ru
vega.ru name server ns2.macomnet.ru
vega.ru mail is handled (pri=0) by vega-gw.vega.ru
vega.ru mail is handled (pri=10) by new.psychol.ras.ru
vega.ru has address 194.67.107.8
smtp.vega.ru is a nickname for vega.ru
nntp.vega.ru is a nickname for vega.ru
news.vega.ru is a nickname for vega.ru
gopher.vega.ru is a nickname for vega.ru
vega-gw.vega.ru mail is handled (pri=0) by vega-gw.vega.ru
vega-gw.vega.ru mail is handled (pri=10) by new.psychol.ras.ru
vega-gw.vega.ru mail is handled (pri=30) by polyn.net.kiae.su
vega-gw.vega.ru has address 194.67.107.8
mail.vega.ru is a nickname for vega.ru
localhost.vega.ru has address 127.0.0.1
```

www.vega.ru is a nickname for vega.ru
belti.vega.ru mail is handled (pri=10) by pms.belti.ru
belti.vega.ru mail is handled (pri=20) by mail.belti.ru
belti.vega.ru has address 213.59.3.161
www.belti.vega.ru has address 213.59.3.161
ns1.vega.ru has address 194.67.107.1
ftp.vega.ru is a nickname for vega.ru
ns.vega.ru is a nickname for vega.ru
ns2.vega.ru has address 194.67.107.1
vega.ru start of authority ns1.psychol.ras.ru serge.psychol.ras.ru(
2001121202 ;serial (version)
7200 ;refresh period
1800 ;retry refresh this often
3600000 ;expiration period
107800 ;minimum TTL
)
>

Как это не выглядит странным, но расширенный отчет, в котором для сокращения его объема мы удалили несколько записей, выглядит гораздо более привычно, чем стандартный отчет host. Во всяком случае, в нем читаются обычные записи описания ресурсов для зоны vega.ru:

> host -l -v -t any vega.ru
rcode = 0 (Success), ancount=3
Found 1 addresses for ns.macomnet.ru
Found 1 addresses for ns1.psychol.ras.ru
Found 1 addresses for ns2.macomnet.ru
Trying 195.128.64.3
Server failed, trying next server: Query refused
Trying 62.117.88.14
vega.ru 107800 IN SOA ns1.psychol.ras.ru serge.psychol.ras.ru(
2001121202 ;serial (version)
7200 ;refresh period
1800 ;retry refresh this often
3600000 ;expiration period
107800 ;minimum TTL
)
vega.ru 107800 IN NS ns1.psychol.ras.ru
vega.ru 107800 IN NS ns.macomnet.ru
vega.ru 107800 IN NS ns2.macomnet.ru
vega.ru 107800 IN MX 0 vega-gw.vega.ru
vega.ru 107800 IN MX 10 new.psychol.ras.ru
vega.ru 107800 IN A 194.67.107.8
smtp.vega.ru 107800 IN CNAME vega.ru
nntp.vega.ru 107800 IN CNAME vega.ru
news.vega.ru 107800 IN CNAME vega.ru
gopher.vega.ru 107800 IN CNAME vega.ru
vega-gw.vega.ru 107800 IN A 194.67.107.8
mail.vega.ru 107800 IN CNAME vega.ru
www.vega.ru 107800 IN CNAME vega.ru
ftp.vega.ru 107800 IN CNAME vega.ru
ns.vega.ru 107800 IN CNAME vega.ru
ns2.vega.ru 107800 IN A 194.67.107.1

```
vega.ru 107800 IN SOA ns1.psychol.ras.ru serge.psychol.ras.ru(
2001121202 ;serial (version)
7200 ;refresh period
1800 ;retry refresh this often
3600000 ;expiration period
107800 ;minimum TTL
)
>
```

Любопытно, что SOA запись повторяется дважды, но это не какая-то особенность host. Сами разработчики host пишут о том, что это происходит по таинственной причине ("arcane reasons"). На самом деле в RFC-1034 черным по белому написано, что при передаче зоны по AXFR запросу, а именно им и пользуются при копировании зоны, в качестве первого и последнего сообщений в потоке обмена данными передается узел начала описания зоны (top authoritative node of the zone), а это, собственно, и есть SOA запись. На самом деле так себя ведут все программы, если они правильно соблюдают рекомендации RFC.

Пока в своих запросах мы использовали только серверы умолчания, которые указаны в resolv.conf. Host позволяет использовать в качестве сервера доменных имен для клиента, который хочет получить обслуживание свих рекурсивных запросов, любой сервер сети. Точнее, мы можем попытаться использовать "чужой" сервер таким образом:

```
> host -t a polyn.kiae.su ns.spb.su
Using domain server:
Name: ns.spb.su
Address: 193.124.83.69
Aliases:

polyn.kiae.su has address 144.206.160.32
>
```

Вообще говоря, не любой сервер будет выполнять рекурсивный запрос. Например, корневые серверы его не выполняют:

```
> host -t a polyn.kiae.su A.ROOT-SERVERS.NET.
Using domain server:
Name: A.ROOT-SERVERS.NET
Address: 198.41.0.4
Aliases:


>
```

Любой администратор может ограничить использование своего сервера хостами сети для обслуживания рекурсивных запросов.

Теперь несколько замечаний по поводу работы с серверами. Во-первых, в качестве корневого сервера host всегда использует A сервер, а, во-вторых, при работе с серверами доменных имен умолчания, хост не выбирает лучший из них, а просто пользуется первым, если он работает, а если не работает, то берет второй из списка:

```
generate# host -d demin
;; res_nmkquery(QUERY, demin.polyn.kiae.su, IN, A)
```

```
;; res_send()
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56137
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; demin.polyn.kiae.su, type = A, class = IN
;; Querying server (# 1) address = 144.206.192.1
;; new DG socket
res_send: recvfrom: Connection refused
;; Querying server (# 2) address = 144.206.192.10
;; new DG socket
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56137
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; demin.polyn.kiae.su, type = A, class = IN
demin.polyn.kiae.su. 1H IN A 144.206.192.4
```

Это не весь отладочный отчет, а только его фрагмент, который позволяет зафиксировать перебор серверов. Если мы снова обратимся к услугам host, то перебор серверов будет выполнен вновь.

На самом деле, вовсе не факт, что в вашей системе стоит программа host, а если даже она установлена, то, скорее всего, в ней нет множества полезных функций, из-за которых ее предпочитают другим системам тестирования DNS. Просто версия старая. По этой причине лучше всего ее (программу Host) скопировать с ftp сервера ftp://ftp.nikhef.nl/pub/network/host.tar.Z. На момент написания этого материала наиболее свежей версией была версия от 2000 года.

Что полезного можно найти в свежих версиях программы. Например, проверку записи SOA:

```
> ./host -C webstatistics.ru. 144.206.192.55
webstatistics.ru (generate.polyn.kiae.su)
ns.webstatistics.ru paul.webstatistics.ru (1 3600 600 86400 3600)
!!! webstatistics.ru SOA expire is less than 1 week (1 day)
>
```

Host в данном случае указала на слишком короткий срок отключения slave сервера при отказе основного сервера доменных имен.

А вот пример проверки записей NS и теста серверов доменных имен:

```
generate# host -t ns webstatistics.ru. 144.206.192.55
webstatistics.ru NS ns.webstatistics.ru
!!! webstatistics.ru NS host ns.webstatistics.ru is not canonical
webstatistics.ru NS ns4.nic.ru
generate# /stats/archive/host/host -C webstatistics.ru. ns4.nic.ru
webstatistics.ru (ns4.nic.ru)
webstatistics.ru SOA record currently not present at ns4.nic.ru
generate#
```

Две последовательно примененные команды host позволяют выявить отсутствие описания зоны на сервере ns4.nic.ru.

Вот еще один пример применения host:

```
generate# ./host -L 1 webstatistics.ru. 144.206.192.55
webstatistics.ru. NS ns.webstatistics.ru.
!!! webstatistics.ru NS host ns.webstatistics.ru is not canonical
```

В данном случае host сообщает нам, что имя сервера ns.webstatistics.ru - это синоним, а не каноническое имя. Параметр "-L 1" позволяет управлять рекурсий перебора синонимов, т.е. несколько раз обращаться системе DNS для поиска канонического имени при наличии цепочки синонимов.

Host позволяет реализовать множество полезных отчетов. Именно по этой причине его применяют для получения статистики в RIPE. Вот в заключении фрагмент отчета по зоне ru:

```
generate# ./host -H ru.
ru AXFR record query refused by NS2.RIPN.NET
!!! WINUP.ru NS host hw.prometeus.nsc.ru is not canonical
!!! AK.ru NS host ns4.piter.net does not exist
AK.ru has lame delegation to ns4.piter.net
!!! DC.ru NS host dc.netway.ru does not exist
DC.ru has lame delegation to dc.netway.ru
…
```

Речь идет о некорректном делегировании. Для указанных в NS записях имен серверов нет IP-адресов.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.

Полезные ссылки:

1. Peter Koch. Ripe-203. Recommendations for DNS SOA Values. 1999. (http://www.ripe.net/docs/ripe-203.html)
2. http://ciisa.isa.utl.pt/cgi-bin/man2html?host+1 - страница man host для операционной системы Linux. Это не полный список ключей, но кое что есть.

9. c. Программа тестирования системы доменных имен - dig

В данном материале рассматривается применение различных опций программы dig - одного из основных средств тестирования работы системы доменных имен, поставляемого вместе с BIND сервером системы доменных имен.

Программа dig входит в комплект поставки BIND. Ее основное назначение - проверка правильности работы сервера доменных имен. При исполнении программы она сообщает свою версию, которая, как правило, совпадает с версией BIND.

Для того, чтобы просто получить IP-адрес по имени хоста достаточно выполнить:

```
> /usr/local/bin/dig quest.polyn.kiae.su.

; <<>> DiG 9.2.1 <<>> quest.polyn.kiae.su.
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39541
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;quest.polyn.kiae.su. IN A

;; ANSWER SECTION:
quest.polyn.kiae.su. 3600 IN A 144.206.192.2

;; AUTHORITY SECTION:
polyn.kiae.su. 3600 IN NS ns.spb.su.
polyn.kiae.su. 3600 IN NS ns.ussr.eu.net.
polyn.kiae.su. 3600 IN NS polyn.net.kiae.su.

;; ADDITIONAL SECTION:
ns.spb.su. 160259 IN A 193.124.83.69
ns.ussr.eu.net. 166860 IN A 193.124.22.65
polyn.net.kiae.su. 71012 IN A 144.206.160.32

;; Query time: 3 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Tue Nov 5 20:17:23 2002
;; MSG SIZE rcvd: 187

>
```

На самом деле, это довольно пространный ответ, если сравнивать его с отчетом host, например. Он больше похож на отладочные сообщения системы тестирования нового сервера доменных имен. Видимо, так оно и есть.

Еще одной особенностью dig является отключенный по умолчания список поиска (подстановка имен доменов по умолчанию):

```
/usr/local/bin/dig quest
; <<>> DiG 9.2.1 <<>> quest
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 50813
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;quest. IN A

;; AUTHORITY SECTION:
. 185 IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISI
GN-GRS.COM. 2002110500 1800 900 604800 86400
```

```
;; Query time: 2 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Tue Nov 5 20:21:11 2002
;; MSG SIZE rcvd: 98

>
```

Как видно из этого примера имя quest не расширяется именем домена. В конце имени хоста просто ставится символ ".", и запрос через локальный сервер доменных имен, который выполняет рекурсивные запросы, переправляется на корневой сервер A.

Оба предыдущих примера хорошо демонстрируют тот факт, что dig просто распечатывает секции DNS сообщения и его заголовок, т.е. формат отчета dig предполагает определенную "подкованность" пользователя в форматах и спецификациях системы доменных имен.

В отличие от других средств отладки DNS dig при получении доменного имени по IP-адресу требует указания соответствующей опции для поиска в обратных зонах, сам адрес по умолчанию не "выворачивает":

```
>/p>

> /usr/local/bin/dig 144.206.192.1

; <<>> DiG 9.2.1 <<>> 144.206.192.1
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 54475
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;144.206.192.1. IN A

;; AUTHORITY SECTION:
. 574 IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISI
GN-GRS.COM. 2002110500 1800 900 604800 86400

;; Query time: 2 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Tue Nov 5 20:29:20 2002
;; MSG SIZE rcvd: 106

>
```

В данном случае адрес воспринят просто как доменное имя и вместо записи PTR dig ищет адресную запись. Для того чтобы адрес инвертировался нужно указать опцию "-x":

```
> /usr/local/bin/dig -x 144.206.192.2

; <<>> DiG 9.2.1 <<>> -x 144.206.192.2
;; global options: printcmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30502
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;2.192.206.144.in-addr.arpa. IN PTR

;; ANSWER SECTION:
2.192.206.144.in-addr.arpa. 3014 IN PTR quest.polyn.kiae.su.

;; AUTHORITY SECTION:
192.206.144.in-addr.arpa. 3138 IN NS polyn.net.kiae.su.
192.206.144.in-addr.arpa. 3138 IN NS ns.ussr.eu.net.
192.206.144.in-addr.arpa. 3138 IN NS ns.spb.su.

;; ADDITIONAL SECTION:
polyn.net.kiae.su. 68707 IN A 144.206.160.32
ns.ussr.eu.net. 164555 IN A 193.124.22.65
ns.spb.su. 157954 IN A 193.124.83.69

;; Query time: 3 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Tue Nov 5 20:55:48 2002
;; MSG SIZE rcvd: 222

>
```

Вот теперь все правильно, и имя преобразовано в имя домена зоны in-addr.arpa, и запрашивается запись типа PTR.

На самом деле можно и самому попробовать поискать "в лоб", задавая имя хоста в зоне in-addr.arpa и тип записи PTR, но это будет многоступенчатый процесс. Сначала dig отправит вас на корневые сервера, потом, на сервера зоны 144.in-addr.arpa и т.д., до тех пор, пока не откликнется авторитативный сервер зоны 192.206.144.in-addr.arpa:

```
> dig @ns.spb.su. 2.192.206.144.in-addr.arpa. -t ptr -c IN

; <<>> DiG 8.3 <<>> @ns.spb.su. 2.192.206.144.in-addr.arpa. -t -c
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; QUERY SECTION:
;; 2.192.206.144.in-addr.arpa, type = PTR, class = IN

;; ANSWER SECTION:
2.192.206.144.in-addr.arpa. 1H IN PTR quest.polyn.kiae.su.

;; AUTHORITY SECTION:
192.206.144.in-addr.arpa. 1H IN NS polyn.net.kiae.su.
192.206.144.in-addr.arpa. 1H IN NS ns.spb.su.
192.206.144.in-addr.arpa. 1H IN NS ns.ussr.eu.net.
```

```
;; ADDITIONAL SECTION:
polyn.net.kiae.su. 1D IN A 144.206.160.32
ns.spb.su. 1D IN A 193.124.83.69
ns.ussr.eu.net. 1D IN A 193.124.22.65

;; Total query time: 15 msec
;; FROM: generate.polyn.kiae.su to SERVER: ns.spb.su. 193.124.83.69
;; WHEN: Tue Nov 5 20:50:14 2002
;; MSG SIZE sent: 44 rcvd: 198

>
```

На самом деле, для интереса можно вручную проделать весь этот путь, для того, чтобы оценить полезность опции "-x" J.

Если объем информации по умолчанию вас раздражает, и Вы хотите получить более краткие отчеты, то можно применить опцию запроса "+short":

```
> /usr/local/bin/dig -x 144.206.192.2 +short
quest.polyn.kiae.su.
>
```

т.е. ничего лишнего. Получили только имя хоста. Естественно, что эту опцию можно применять к любому запросу dig, если она (опция) не противоречит логике запроса.

Например, мы хотим получить в отчете сам запрос (опция +qr). При этом ясно, что получение краткого отчета противоречит нашему желанию. Если мы зададим опцию краткого отчета, то запроса в отчете не будет:

```
> /usr/local/bin/dig -x 144.206.192.2 +qr +short
quest.polyn.kiae.su.
>
```

Если убрать "+sort", запрос в отчете dig появится (от ";;Sending:" до ";;Got answer:"):

```
> /usr/local/bin/dig -x 144.206.192.2 +qr

; <<>> DiG 9.2.1 <<>> -x 144.206.192.2 +qr
;; global options: printcmd
;; Sending:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17703
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;2.192.206.144.in-addr.arpa. IN PTR

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17703
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;2.192.206.144.in-addr.arpa. IN PTR
```

```
;; ANSWER SECTION:
2.192.206.144.in-addr.arpa. 1385 IN PTR quest.polyn.kiae.su.

;; AUTHORITY SECTION:
192.206.144.in-addr.arpa. 1509 IN NS polyn.net.kiae.su.
192.206.144.in-addr.arpa. 1509 IN NS ns.ussr.eu.net.
192.206.144.in-addr.arpa. 1509 IN NS ns.spb.su.

;; ADDITIONAL SECTION:
polyn.net.kiae.su. 67078 IN A 144.206.160.32
ns.ussr.eu.net. 162926 IN A 193.124.22.65
ns.spb.su. 156325 IN A 193.124.83.69

;; Query time: 2 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Tue Nov 5 21:22:57 2002
;; MSG SIZE rcvd: 222

>
```

Теперь вернемся к вопросу применения списка поиска, который по умолчанию отключен в dig. Для включения необходимо указать опцию запроса "+search":

```
> /usr/local/bin/dig quest +search +short
144.206.192.2
>
```

На самом деле, мы получили то, что запрашивали. Вот только полного имени хоста нам так и не сообщили. К какому домену принадлежит имя quest. На первый взгляд мы и так знаем, в каком домене находимся, но ведь список поиска может состоять из нескольких доменных имен. Если /etc/resolv.conf будет иметь следующее содержание:

```
domain polyn.kiae.su
nameserver 144.206.192.10
nameserver 144.206.160.32
```

то при поиске IP-адреса для хоста polyn мы ничего не получим, т.к. хоста polyn.polyn.kiae.su в описании зоны нет:

```
generate# /usr/local/bin/dig polyn +search +nocomments

; <<>> DiG 9.2.1 <<>> polyn +search +nocomments
;; global options: printcmd
;polyn. IN A
. 251 IN SOA A.ROOT-SERVERS.NET. NSTLD.VERISI
GN-GRS.COM. 2002110501 1800 900 604800 86400
;; Query time: 2 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Wed Nov 6 13:11:53 2002
;; MSG SIZE rcvd: 98

generate#
```

В данном отчете мы отключили печать комментариев (+nocomments), поэтому он получился более компактным, чем стандартный отчет.

Теперь изменим resolv.conf:

```
nameserver 144.206.192.10
nameserver 144.206.160.32
search polyn.kiae.su kiae.su .
```

Результат будет совсем иным:

```
generate# /usr/local/bin/dig polyn +search +nocomments

; <<>> DiG 9.2.1 <<>> polyn +search +nocomments
;; global options: printcmd
;polyn.kiae.su. IN A
polyn.kiae.su. 3600 IN A 144.206.160.32
polyn.kiae.su. 3600 IN NS ns.spb.su.
polyn.kiae.su. 3600 IN NS ns.ussr.eu.net.
polyn.kiae.su. 3600 IN NS polyn.net.kiae.su.
ns.spb.su. 98951 IN A 193.124.83.69
ns.ussr.eu.net. 105552 IN A 193.124.22.65
polyn.net.kiae.su. 9704 IN A 144.206.160.32
;; Query time: 3 msec
;; SERVER: 144.206.192.10#53(144.206.192.10)
;; WHEN: Wed Nov 6 13:19:07 2002
;; MSG SIZE rcvd: 175

generate#
```

Dig начала перебор доменных имен из resolv.conf. При этом, если бы мы попросили короткий отчет, то не имели бы представления о полном имени хоста:

```
generate# /usr/local/bin/dig polyn +search +nocomments +short
144.206.160.32
generate#
```

Для демонстрации трассы поиска информации в системе доменных имен, в dig предусмотрена опция "+trace":

```
generate# /usr/local/bin/dig www.ru +trace

; <<>> DiG 9.2.1 <<>> www.ru +trace
;; global options: printcmd
. 352895 IN NS F.ROOT-SERVERS.NET.
. 352895 IN NS B.ROOT-SERVERS.NET.
. 352895 IN NS J.ROOT-SERVERS.NET.
. 352895 IN NS K.ROOT-SERVERS.NET.
. 352895 IN NS L.ROOT-SERVERS.NET.
. 352895 IN NS M.ROOT-SERVERS.NET.
. 352895 IN NS I.ROOT-SERVERS.NET.
. 352895 IN NS E.ROOT-SERVERS.NET.
. 352895 IN NS D.ROOT-SERVERS.NET.
```

. 352895 IN NS A.ROOT-SERVERS.NET.
. 352895 IN NS H.ROOT-SERVERS.NET.
. 352895 IN NS C.ROOT-SERVERS.NET.
. 352895 IN NS G.ROOT-SERVERS.NET.
;; Received 436 bytes from 144.206.192.10#53(144.206.192.10) in 3 ms

ru. 172800 IN NS NS2.NIC.FR.
ru. 172800 IN NS NS.RIPN.NET.
ru. 172800 IN NS NS2.RIPN.NET.
ru. 172800 IN NS SUNIC.SUNET.SE.
ru. 172800 IN NS NS.UU.NET.
ru. 172800 IN NS NS1.RELCOM.ru.
;; Received 260 bytes from 192.5.5.241#53(F.ROOT-SERVERS.NET) in 242 ms

www.ru. 86400 IN NS ns.demos.su.
www.ru. 86400 IN NS ns1.demos.net.
;; Received 76 bytes from 192.93.0.4#53(NS2.NIC.FR) in 170 ms

www.ru. 86400 IN A 194.87.0.50
www.ru. 86400 IN NS ns.demos.su.
www.ru. 86400 IN NS ns1.demos.net.
;; Received 140 bytes from 194.87.0.9#53(ns.demos.su) in 7 ms

generate#

В данном случае dig в отчете показывает отклики от серверов, которые она опрашивает. Для начала dig сообщила нам, что локальный сервер 144.206.192.10 за зону ru не отвечает и, что он отправил нас к корневым серверам. Те, в свою очередь, отправили нас на серверы зоны ru. Серверы зоны ru отправили нас на серверы зоны www.ru. Первый из этих серверов (ns.demos.ru) сообщил нам искомый адрес. Еще раз обратите внимание на то, что поиск адреса осуществляла сама dig, а не локальный сервер доменных имен. По этой причине проверять этой опцией работу локального сервера по обработке рекурсивных запросов не стоит.

Dig обладает еще одной интересной особенностью - выполнением нескольких команд за один раз. При чем, это не выборка нескольких опций из файла, а исполнение нескольких запросов, указанных в командной строке:

generate# /usr/local/bin/dig news.ru +short -x 144.206.160.32 www.ru ns
194.87.0.23
polyn.net.kiae.su.
ns.demos.su.
ns1.demos.net.
generate#

В данном случае мы хотим получить IP адрес news.ru, имя для хоста 144.206.160.32 и все NS записи для зоны www.ru . Dig прекрасно справляется с поставленной задачей, мешаю, правда, все ответы в одну кучу.

На самом деле у dig есть еще один режим, который позволяет выполнять несколько запросов за раз. Это пакетная обработка файлов запросов. Пусть у нас будет иметься в наличии файл comd.txt следующего содержания:

quest.polyn.kiae.su. +short
-x 144.206.192.2 +short
www.rambler.su. +trace +noshort

Первая строка - это запрос на короткий отчет, который должен нам сообщить IP-адрес хоста auest.polyn.kiae.su, вторая срока запрашивает короткий отчет при поиске имени хоста с IP-адресом 144.206.192.2, в третьей строке мы запрашиваем трассу опроса серверов для имени www.rambler.ru и отменяем короткий отчет. Таким образом, каждый запрос dig - это отдельная строка файла. При запуске dig мы получим следующий отчет:

generate# /usr/local/bin/dig -f comd.txt
144.206.192.2
quest.polyn.kiae.su.
. 332654 IN NS A.ROOT-SERVERS.NET.
. 332654 IN NS H.ROOT-SERVERS.NET.
. 332654 IN NS C.ROOT-SERVERS.NET.
. 332654 IN NS G.ROOT-SERVERS.NET.
. 332654 IN NS F.ROOT-SERVERS.NET.
. 332654 IN NS B.ROOT-SERVERS.NET.
. 332654 IN NS J.ROOT-SERVERS.NET.
. 332654 IN NS K.ROOT-SERVERS.NET.
. 332654 IN NS L.ROOT-SERVERS.NET.
. 332654 IN NS M.ROOT-SERVERS.NET.
. 332654 IN NS I.ROOT-SERVERS.NET.
. 332654 IN NS E.ROOT-SERVERS.NET.
. 332654 IN NS D.ROOT-SERVERS.NET.
;; Received 436 bytes from 144.206.192.10#53(144.206.192.10) in 4 ms

ru. 172800 IN NS NS2.NIC.FR.
ru. 172800 IN NS NS.RIPN.NET.
ru. 172800 IN NS NS2.RIPN.NET.
ru. 172800 IN NS SUNIC.SUNET.SE.
ru. 172800 IN NS NS.UU.NET.
ru. 172800 IN NS NS1.RELCOM.ru.
;; Received 268 bytes from 198.41.0.4#53(A.ROOT-SERVERS.NET) in 139 ms

rambler.ru. 86400 IN NS ns.park.rambler.ru.
rambler.ru. 86400 IN NS ns.rambler.ru.
;; Received 103 bytes from 192.93.0.4#53(NS2.NIC.FR) in 170 ms

www.rambler.ru. 3600 IN A 81.19.66.131
www.rambler.ru. 3600 IN A 81.19.66.50
rambler.ru. 3600 IN NS ns.rambler.ru.
rambler.ru. 3600 IN NS ns.park.rambler.ru.
;; Received 135 bytes from 217.73.193.23#53(ns.park.rambler.ru) in 4 ms

generate#

Теперь от запросов к серверу, как рекурсивных, так и нерекурсивных, перейдем к копированию зоны. Для этого в качестве типа запроса нужно задать тип AXFR. При этом мы будем иммитировать работу slave сервера системы доменных имен, который пытается скопировать зону:

```
generate# /usr/local/bin/dig @localhost webstatistics.ru. axfr +multiline

; <<>> DiG 9.2.1 <<>> @localhost webstatistics.ru. axfr +multiline
;; global options: printcmd
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.webstatistics.ru.
(
2 ; serial
3600 ; refresh (1 hour)
600 ; retry (10 minutes)
86400 ; expire (1 day)
3600 ; minimum (1 hour)
)
webstatistics.ru. 3600 IN NS ns.webstatistics.ru.
webstatistics.ru. 3600 IN NS ns4.nic.ru.
webstatistics.ru. 3600 IN A 144.206.192.60
host.webstatistics.ru. 3 IN A 144.206.192.63
ns.webstatistics.ru. 3600 IN CNAME webstatistics.ru.
user.webstatistics.ru. 3 IN CNAME host.webstatistics.ru.
user1.webstatistics.ru. 3 IN CNAME user.webstatistics.ru.
www.webstatistics.ru. 3 IN A 144.206.160.32
www.webstatistics.ru. 3 IN A 144.206.192.60
www.webstatistics.ru. 3 IN A 144.206.192.61
www.webstatistics.ru. 3 IN A 144.206.192.62
www1.webstatistics.ru. 3 IN CNAME www.webstatistics.ru.
zone.webstatistics.ru. 3 IN NS ns.webstatistics.ru.
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.webstatistics.ru.
(
2 ; serial
3600 ; refresh (1 hour)
600 ; retry (10 minutes)
86400 ; expire (1 day)
3600 ; minimum (1 hour)
)
;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(localhost)
;; WHEN: Wed Nov 6 19:13:48 2002
;; XFR size: 16 records

generate#
```

В этом примере мы применили сразу два атрибута командной строки, которые не применяли раньше. Во-первых, мы явным образом указали сервер (@loacalhost), с которого мы хотим списать зону, во-вторых, указали явным образом тип запроса (axfr). Последний параметр (+multilane) позволил нам в удобочитаемом виде расположить опции SOA записи. Как и положено (RFC-1034) при копировании зоны SOA запись появляется дважды - в начале и конце передачи данных.

Вообще говоря, dig позволяет проиммитировать и инкрементальный обмен данными - тип запроса ixfr:

```
generate# /usr/local/bin/dig @193.0.0.236 example.com ixfr=2002010300

; <<>> DiG 9.2.1 <<>> @193.0.0.236 example.com ixfr=2002010300
;; global options: printcmd
example.com. 172800 IN SOA dns1.icann.org. hostmaster.icann
.org. 2002020400 7200 3600 1209600 21600
example.com. 21600 IN NS a.iana-servers.net.
example.com. 21600 IN NS b.iana-servers.net.
example.com. 172800 IN A 192.0.34.72
www.example.com. 172800 IN A 192.0.34.72
example.com. 172800 IN SOA dns1.icann.org. hostmaster.icann
.org. 2002020400 7200 3600 1209600 21600
;; Query time: 62 msec
;; SERVER: 193.0.0.236#53(193.0.0.236)
;; WHEN: Wed Nov 6 19:30:16 2002
;; XFR size: 7 records

generate#
```

Номер, который задается в качестве параметра ixfr - это серийный номер описания зоны. На самом деле, из данного отчета мало что понятно, т.к. мы не знаем, когда проводились реальные обновления зоны, и в чем они состояли.

Для того, чтобы проверить при помощи dig работу динамического обновления зоны, необходимо это свойство named настроить. Для этого следует выполнить две вещи:

- разрешить динамическое обновление зоны в файле конфигурации named.conf
- настроить систему ведения журналов named таким образом, чтобы она фиксировала изменения в зоне

Разрешение динамического обновление настраивается следующим образом: в файл конфигурации named в правила, относящиеся к конкретной зоне добавляют опцию allow-update:

```
zone "webstatistics.ru" {
type master;
file "webstatistics.ru";
allow-update {
{ 144.206.192.55; 144.206.160.32; 127.0.0.1; };
};
};
```

Специалисты по безопасности в этом месте заклеймят нас позором, т.к. мы только перечислили IP-адреса, с которых можно проводить обновление, но не установили дополнительных условий типа подписей транзакций (TSIG - transaction signature). Но на наш взгляд в данном случае упоминание дополнительных "наворотов" не способствует прозрачности изложения материала, а потому вопросы безопасности мы пока отодвинем на второй план. Мы просто хотим убедиться в работе dig с ixfr.

Теперь нам нужно быть уверенными в том, что сервер действительно принимает запросы динамического обновления и реагирует на них. Для этого нужно настроить соответствующим образом журнал обращений к серверу доменных имен. За эту часть работы отвечает опция logging в файле конфигурации named:

```
logging {
channel "update_log"{
file "update.log";
severity debug 3;
};
category "default" { "default_syslog"; "default_debug"; };
category "update" { "update_log"; };
};
```

Мы создали новый канал для ведения журнала - update_log. Связали с этим каналом файл update.log, который будет размещаться в рабочем каталоге сервера. Серьезность сообщений, которые туда будут писаться, определили как отладку третьего уровня. Сам канал отнесли к категории Update (список категорий, по которым ведутся журналы в named, ограничен и приведен в руководстве системного администратора сервера).

Теперь перезапустим сервер и попробуем внести изменение в зону "на лету", воспользовавшись программой nsupdate:

```
generate# /usr/local/bin/nsupdate -d
> server 127.0.0.1
> zone webstatistics.ru
> update add newhost 3600 in cname host
>

Reply from update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id: 65532
;; flags: qr ra ; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0

>
```

Прямо скажем, что для обычного пользователя не очень информативно. Само обновление сразу в файл описания зоны не попадет, и поэтому не просто установить факт свершившегося обновления.

Смотрим в журнал динамических обновлений зоны сервера доменных имен:

```
generate# more namedb/update.log
client 144.206.192.55#3187: updating zone 'webstatistics.ru/IN': adding an RR
generate#
```

Наш уровень сообщений позволил нам узнать, что что-то обновилось, но вот что? Сначала запустим dig для того, чтобы узнать серийный номер зоны. Это нам даст первую информацию о том, было или не было обновление, если мы, конечно, помним предыдущий серийный номер:

```
generate# /usr/local/bin/dig @localhost webstatistics.ru soa +short
ns.webstatistics.ru. paul.webstatistics.ru. 7 3600 600 86400 3600
generate#
```

Вот теперь запускаем dig для тестирования состояний динамических обновлений:

```
generate# /usr/local/bin/dig @localhost webstatistics.ru ixfr=6 +nocomments
```

```
; <<>> DiG 9.2.1 <<>> @localhost webstatistics.ru ixfr=6 +nocomments
;; global options: printcmd
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.websta
tistics.ru. 7 3600 600 86400 3600
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.websta
tistics.ru. 6 3600 600 86400 3600
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.websta
tistics.ru. 7 3600 600 86400 3600
newhost.webstatistics.ru. 3600 IN CNAME host.webstatistics.ru.
webstatistics.ru. 3600 IN SOA ns.webstatistics.ru. paul.websta
tistics.ru. 7 3600 600 86400 3600
;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(localhost)
;; WHEN: Thu Nov 7 10:55:35 2002
;; XFR size: 5 records

generate#
```

Вполне прогнозируемый результат. Dig сообщил нам, что не только была добавлена запись CNAME, но и изменилась запись SOA. Таким образом, мы можем с удовлетворением отметить, что все прекрасно работает.

Если снова посмотреть на описание зоны, которое до этого момента мы правили руками, то оно изменится гораздо существеннее, чем можно было бы ожидать. Сервер при динамическом обновлении его переписывает, подгоняя под внешний вид и стиль описания, которого придерживаются сами разработчики сервера.

Вообще говоря, у нас в запасе есть еще один способ убедиться в том, что все работает прекрасно - это файл журнала обновлений, но не наш, а системный для зоны webstatistics.ru - webstatistics.ru.jnl. В этом файле должны сохраняться все динамические обновления зоны. Следует, правда, иметь в виду, что информация пишется туда не для чтения ее человеком, но бездушной машиной. Разобрать, конечно, кое-что можно, но dig удобнее.

В конце концов, если не очень торопиться, то изменения должны появиться и в описании зоны, но при этом нужно будет подождать некоторое время (примерно час). Понятно, что в этих условиях Dig позволит получить информацию быстрее.

Dig позволяет получать информацию не только об интернет зонах. Например, в BIND есть такая особенность: named дает возможность удаленному пользователю узнать версию пакета. На самом деле это не очень хорошо. Если данная версия обладает "дыркой" в системе безопасности, то сервер доменных имен только помогает определить инструмент для своего взлома.

```
generate# /usr/local/bin/dig -t txt -c chaos VERSION.BIND @localhost

; <<>> DiG 9.2.1 <<>> -t txt -c chaos VERSION.BIND @localhost
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14834
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;VERSION.BIND. CH TXT

;; ANSWER SECTION:
VERSION.BIND. 0 CH TXT "9.2.1"

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(localhost)
;; WHEN: Wed Nov 6 14:19:12 2002
;; MSG SIZE rcvd: 48

generate#
```

На самом деле, настраивая BIND, можно "прикинуться" другой версией пакета. Для этого в файл конфигурации named нужно указать:

```
zone "bind" chaos {
type master ;
file "bind";
allow-query {
localhost ;
} ;
allow-transfer {
none;
} ;
};
```

И создать описание зоны (файл bind):

```
$ORIGIN bind.
@ 1D CHAOS SOA localhost. root.localhost. (
1 ; serial
3H ; refresh
1H ; retry
1W ; expiry
1D ) ; minimum
CHAOS NS localhost.
```

Собственно, пример с запросом на версию BIND призван был показать тот факт, что dig, как и любая другая система тестирования DNS, может запрашивать записи не только из Internet (класс IN), но и из других поддерживаемых в DNS классах.

В новых версиях BIND можно поступить еще проще, вставив в options (в файле named.conf) следующий фрагмент:

```
options {
version "Ne znaju";
};
```

В этом случае текст у директивы version будет отображаться в качестве txt RR-записи в зоне version.bin класса chaos.

Рекомендованная литература:

1. P. Mockapetris. RFC-1034. DOMAIN NAMES - CONCEPTS AND FACILITIES. ISI, 1987. (http://www.ietf.org/rfc/rfc1034.txt?number=1034)
2. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)
3. Альбитц П., Ли К.. DNS и BIND. - Пер. с англ. - СПб: Символ-Плюс, 2002. - 696 с.
4. Документация по BIND 9. Справочное руководство системного администратора. (http://www.nominum.com/resources/documentation/Bv9ARM.pdf)

Полезные ссылки:

1. http://ciisa.isa.utl.pt/cgi-bin/man2html?dig+1 - справочное руководство по dig в стиле Unix manual. Следует иметь в виду, что версия с которой вы работаете можно довольно сильно отличаться от этого описания. По этой причине лучше использовать тот manual, который поставляется с дистрибутивом named и dig

9. d. Средства и методы тестирования работы системы доменных имен, отличные от nslookup, host и dig

В этом материале речь пойдет о программах, которые нечасто используются при работе с системой доменных имен, в том числе и о тех, которые имеют преданных и активных сторонников. Всех их можно подразделить на конкурентов BIND, программы тестирования серверов, программы тестирования DNS. Альтернативные BIND серверы доменных имен мы здесь рассматривать не будем.

Было бы странно, если при обсуждении проблем системы доменных имен мы не упомянули об авторе qmail профессоре Бернштейне (D J Bernstein aka DJB). Почтовый транспортный агент (в современной терминологии, после выхода в свет RFC - 2821, почтовый сервер) qmail имел и имеет до сих довольно много приверженцев и стойко удерживает второе место по числу установок в Сети.

Qmail - это не единственное детище DJB. В контексте нашей проблемы нас больше всего будут интересовать djbdns - набор средств работы с системой доменных имен. Будучи настоящим математиком, DJB сконцентрировал свои усилия при разработке djbdns на вопросе точного соблюдения спецификаций, безопасности и надежности кода.

В отличии от BIND djbdns состоит из нескольких независимых компонентов, которые запускаются в системе как независимые процессы. Более того, каждый процесс (демон) должен быть запущен с правами уникального пользователя.

DNScache - это кэширующий сервер, который выполняет рекурсивные запросы локальных клиентов. Tynidns - это сервер, который отвечает только на udp запросы к своей зоне. Axfrdns и axfr-get используются при обмене данными зоны по TCP.

При работе с djbdns не нужно самому руками править зону. Для этого есть несколько специальных утилит. Кроме самого djbdns на ваш Unix-клон придется поставить еще

парочку продуктов: daemonstools, ucspi-tcp. Эти продукты необходимы для работы djbdns и обеспечивают пакету приемлемую безопасность и быстродействие.

Вообще говоря, если решиться на замену BIND и прочего программного обеспечения на gjbdns, то следует приготовиться к маленькой революции в вашей Unix-системе, а также научиться без ошибок набирать тексты в командной строке.

Из всего множества программ djbdns мы остановимся только на программах получения информации из системы доменных имен, а также на программах тестирования работы серверов. При описании этих программ мы будем опираться только на описание DJB и свой собственный опыт, т.к. установленный нами пакет не содержал более подробной документации, чем так, которая приведена в списке полезных ссылок к данному материалу.

И так, для получения информации из системы DNS в djbdns используются: dnsip, dnsipq, dnsname, dnsmx, dnstxt.

Dnsip позволяет получать IP-адрес хоста, если задано полное (FQDN) доменное имя этого хоста. Если имя задано частично, то получаем пустую строку. Перебор имен доменов в данном случае не осуществляется.

```
generate# ./dnsip quest.polyn.kiae.su.
144.206.192.2
generate#
```

Мы попросили у dnsip IP-адрес хоста quest.polyn.kiae.su. Задавать символ точки на конце имени не обязательно. Имена можно перечислять в качестве аргументов командной строки. В последнем случае каждый ответ будет напечатан на отдельной строке:

```
generate# ./dnsip quest.polyn.kiae.su www.ru
144.206.192.2
194.87.0.50
generate#
```

Для того, чтобы получать адреса для имен, заданных частично, следует воспользоваться программой dnsipq:

```
generate# dnsipq quest cpuv1
quest.polyn.kiae.su 144.206.192.2
cpuv1.kiae.su 193.124.22.22
generate#
```

В первом случае произведена подстановка всего доменного имени, а во втором фрагмента доменного имени, состоящего из двух частей, что соответствует алгоритму подстановки resolver. Если теперь попросить более мелкий фрагмент доменного имени, то мы ничего не получим:

```
generate# dnsipq relarn
relarn.
generate#
```

Хоста relarn в доменах polyn.kiae.su и kiae.su нет, а просто ru не применяется в переборе resolver-a. Попутно заметим, что djbdns не использует resolver BIND или тот, который установлен по умолчанию.

Еще один интересный момент:

```
generate# dnsipq www www.polyn
www.polyn.kiae.su 144.206.160.32
www.polyn
generate#
```

Адрес для хоста www.polyn.kiae.su в принципе существует, но заданное частично начало имени не позволяет его найти.

Теперь от поиска IP-адресов по именам перейдем к поиску имен по IP-адресам. Для этого в djbdns служит программа dnsname:

```
generate# ./dnsname 144.206.192.55
generate.polyn.kiae.su
generate#
```

Все так же лаконично, как и в случае поиска адресов - задаем адрес и получаем имя. Поиск, естественно, осуществляется по "обратным" зонам, но преобразовывать IP-адрес в формат зоны in-addr.arpa не нужно.

Точно так же, как и в случае dnsip, в dnsname можно задавать запрос на поиск нескольких имен:

```
generate# ./dnsname 144.206.192.55 144.206.160.32
generate.polyn.kiae.su
polyn.net.kiae.su
generate#
```

Каждый ответ будет печататься в этом случае на отдельной строке.

Вообще говоря, в djbdns есть программа-фильтр, которая прямо так и называется dnsfilter. Ее главное назначение принимать из потока стандартного ввода IP-адреса, и выдавать в поток стандартного вывода доменные имена:

```
generate# ./dnsfilter
144.206.192.1
144.206.192.1
144.206.192.2
144.206.192.2=quest.polyn.kiae.su
144.206.192.55
144.206.192.55=generate.polyn.kiae.su
144.206.160.32
144.206.160.32=polyn.net.kiae.su
generate#
```

Как видно из примера, если соответствие существует, то выдается строка типа "IP-адрес=доменное_имя", если не существует, то просто на стандартный вывод копируется

IP-адрес. В примере dnsfilter использовался как простая интерактивная команда, хотя основное ее назначение работать в связке с другими командами в качестве фильтра.

На самом деле есть еще один интересный момент:

```
generate# ./dnsname 127.0.0.1.

generate#
```

Мы пытаемся найти имя для адреса 127.0.0.1. Система его, естественно, не находит, потому, что не знает к кому обращаться за ним. Для этой сети в in-adrr.arpa зона не поддерживается.

А вот dig нам ответ дает совершенно запросто:

```
generate# ./dig @localhost -x 127.0.0.1 +short
localhost.polyn.kiae.su.
generate#
```

Мы запрашиваем локальный сервер, а на нем обратная зона 0.0.127.in-addr.arpa определена.

Мы сейчас оставим за скобками правомерность такого запроса вообще. Пример нужен был нам для того, чтобы обратить внимание на аргумент, который в dnsname, dnsip и dnsipq мы не использовали - имя сервера, к которому мы обращаемся запросом. В конце концов, если не указывать имя сервера, то и dig ничего не вернет:

```
generate# ./dig -x 127.0.0.1 +short
generate#
```

Следующая программа - это dnsmx. Она позволяет получить MX записи для заданного доменного имени:

```
generate# ./dnsmx polyn.kiae.su
10 polyn.kiae.su
20 relay1.relcom.ru
generate#
```

Если для заданного имени нет MX записей, то возвращается искусственная несуществующая запись:

```
generate# ./dnsmx kuku.polyn.kiae.su
0 kuku.polyn.kiae.su
generate#
```

Тем самым dnsmx моделирует работу MTA.

Такое внимание к MX записям объясняется наличием ошибок при описании зоны в части прописывания пересылки почты внутри зоны. Например, наличие петель пересылки.

Другая команда, dnstxt, написана, видимо, по причине использования TXT записей для совершенно разных функций. Например, для ограничения доступа к зоне (secure_zone

TXT запись). Привести толковый пример использования TXT записи сложно. Перебирая провайдеров, мы наткнулись только на demos.ru:

```
generate# ./dnstxt demos.ru
$Id: demos.ru,v 1.3 2002/05/13 08:14:45 rvp Exp $
generate#
```

В качестве отклика отображается та часть записи, которая относится к полю DATA (см. "Описание зоны. Формат записи описания ресурсов RR."):

```
demos.ru. 86361 IN TXT "$Id: demos.ru,v 1.3 2002/05/13 08:14:45 rvp Exp $"
```

Теперь рассмотрим программы для тестирования работы собственно серверов: dnsqr, dnsq, dnstrace. Программу tinydns-get мы опустим, т.к. это компонента сервера и требует установки дополнительного программного обеспечения в отличие от других команд.

Программа dnsqr посылает рекурсивные запросы на получение записей определенного типа для доменного имени, которое задается последним аргументом командной строки:

```
generate# ./dnsqr any polyn.kiae.su
255 polyn.kiae.su:
338 bytes, 1+7+3+5 records, response, authoritative, noerror
query: 255 polyn.kiae.su
answer: polyn.kiae.su 3600 NS polyn.net.kiae.su
answer: polyn.kiae.su 3600 NS ns.spb.su
answer: polyn.kiae.su 3600 NS ns.ussr.eu.net
answer: polyn.kiae.su 3600 SOA polyn.net.kiae.su paul.kiae.su 233 3600 300 99999
99 3600
answer: polyn.kiae.su 3600 MX 10 polyn.kiae.su
answer: polyn.kiae.su 3600 MX 20 relay1.relcom.ru
answer: polyn.kiae.su 3600 A 144.206.160.32
authority: polyn.kiae.su 3600 NS polyn.net.kiae.su
authority: polyn.kiae.su 3600 NS ns.spb.su
authority: polyn.kiae.su 3600 NS ns.ussr.eu.net
additional: polyn.net.kiae.su 70984 A 144.206.160.32
additional: ns.spb.su 135020 A 193.124.83.69
additional: ns.ussr.eu.net 136323 A 193.124.22.65
additional: polyn.kiae.su 3600 A 144.206.160.32
additional: relay1.relcom.ru 60516 A 193.125.152.57
generate#
```

В данном случае мы попросили все записи для имени Polyn.kiae.su.

Сразу следует оговориться, что получить зону целиком при помощи команд отладки пакета djbdns нельзя. Для этого следует пользоваться компонентой сервера этого пакета - tinydns-get. Возможность ввести в качестве аргумента тип axfr не должна вводить в заблуждение:

```
generate# ./dnsqr axfr polyn.kiae.su
252 polyn.kiae.su:
temporary failure
generate#
```

Если необходимо протестировать какой либо сервер на предмет обслуживания нерекурсивных запросов, то в этом случае следует воспользоваться программой dnsq:

```
generate# ./dnsq mx quest.polyn.kiae.su 144.206.160.32
15 quest.polyn.kiae.su:
251 bytes, 1+2+3+5 records, response, authoritative, weird ra, noerror
query: 15 quest.polyn.kiae.su
answer: quest.polyn.kiae.su 3600 MX 10 quest.polyn.kiae.su
answer: quest.polyn.kiae.su 3600 MX 20 relay1.relcom.ru
authority: polyn.kiae.su 3600 NS polyn.net.kiae.su
authority: polyn.kiae.su 3600 NS ns.spb.su
authority: polyn.kiae.su 3600 NS ns.ussr.eu.net
additional: quest.polyn.kiae.su 3600 A 144.206.192.2
additional: relay1.relcom.ru 78029 A 193.125.152.57
additional: polyn.net.kiae.su 66089 A 144.206.160.32
additional: ns.spb.su 66066 A 193.124.83.69
additional: ns.ussr.eu.net 131676 A 193.124.22.65
generate#
```

В данном примере мы запрашиваем авторитативный сервер зоны polyn.kiae.su на предмет наличия MX записей для хоста quest.polyn.kiae.su. В качестве ответа получаем эти записи плюс информацию о серверах ответственных за зону polyn.kiae.su. Стоит отметить, что тип запроса (тип записи) был преобразован в цифровой код, который соответствует этому типу RR (число "15" перед именем хоста в первой строке отчета).

Теперь посмотрим, как работает программа трассировки поиска информации в системе DNS. Вообще говоря, это монстр, который может порождать тысячи запросов. Он позволяет проверить все пути в дереве DNS, которые приводят к получению ответа. При этом можно выявить медленные серверы, некорректное делегирование, неработающие серверы и другие проблемы. Например, в отчете для зоны ru. есть такие строчки:

```
1 ns.ussr.eu.net 194.85.119.1
ALERT: lame server; refers to .
```

Саму программу запускают следующим образом:

```
generate# ./dnstrace soa ru ns.ripn.net
. cache NS .
. cache A 194.85.119.1
6 ru 194.85.119.1
ru cache NS ns.ripn.net
ru cache NS ns1.relcom.ru
ru cache NS ns.uu.net
ru cache NS sunic.sunet.se
ru cache NS ns2.nic.fr
ru cache NS ns2.ripn.net
ns.ripn.net cache A 194.85.119.1
ns1.relcom.ru cache A 193.125.152.3
ns2.ripn.net cache A 194.226.96.30
ru 86400 SOA ns.ripn.net hostmaster.ripn.net
400 5371 7200 900 2592000 345600
…
```

Мы обрезали отчет из-за его непомерной длины. Вместо ns.ripn.net можно указать любой сервер, с которого начинается обращение к системе доменных имен. На самом деле начинать, конечно, нужно с корневых серверов, но все работает и в случае нашего примера.

Собственно, на этом примере можно завершить рассмотрение творения DJB. Следует заметить, что довольно часто djbdns рассматривают в качестве альтернативы BIND. Вызвано это, скорее всего, не тем, что сервер Бернштейна действительно хорош по своим потребительским и прочим качествам, а тем, что реальной альтернативы BIND пока действительно не видно.

Рассмотрим еще несколько программ тестирования зон системы DNS. Первой из них будет dnswalk. Программа написана на Perl (первоначально она представляла фильтр программы dig) с использованием модуля Net::DNS. Основное назначение программы поиск ошибок в описании зон и поиск некорректного делегирования.

Если вызвать программу без дополнительных ключей командной строки, то получим:

```
generate# dnswalk bard.kiae.ru.
Checking bard.kiae.ru.
Getting zone transfer of bard.kiae.ru. from iris.polyn.kiae.su...done.
SOA=ns.polyn.kiae.ru contact=paul.polyn.kiae.su
BAD: bard.kiae.ru NS polyn.kiae.ru: unknown host
WARN: bard.kiae.ru MX iris.polyn.kiae.ru: unknown host
WARN: bard.kiae.ru A 144.206.192.32: no PTR record
WARN: mail.bard.kiae.ru A 144.206.192.32: no PTR record
WARN: www.bard.kiae.ru A 144.206.192.32: no PTR record
0 failures, 4 warnings, 1 errors.
generate#
```

Из отчета видно, что сначала найден авторитативный сервер зоны (iris.polyn.kiae.su), далее выявлено некорректное делегирование (unknown host для NS записи) и список нефатальных ошибок, который сводится к отсутствию записей в "обратной зоне".

У dnswalk существует несколько флагов командной строки, которые можно использовать при получении отчетов. Мы не будем останавливаться на всех. Посмотрим только некоторые:

```
generate# dnswalk -l webstatistics.ru.
```

В данном случае мы ищем некорректное делегирование. Если есть подозрение на то, что оно существует, то dnswalk выдаст строку типа:

FAIL: Cannot get SOA record for webstatistics.ru from ns4.nic.ru (lame?):
no name servers

Можно попросить dnswalk рекурсивно просмотреть и все поддомены:

```
generate# dnswalk -r webstatistics.ru.
Checking webstatistics.ru.
Getting zone transfer of webstatistics.ru. from ns.webstatistics.ru...done.
SOA=ns.webstatistics.ru contact=paul.webstatistics.ru
BAD: webstatistics.ru NS ns.webstatistics.ru: CNAME (to webstatistics.ru)
```

BAD: webstatistics.ru NS ns4.nic.ru: unknown host
WARN: user1.webstatistics.ru CNAME user.webstatistics.ru: CNAME
(to host.webstatistics.ru)
BAD: zone.webstatistics.ru NS ns.webstatistics.ru: CNAME (to webstatistics.ru)
Checking zone.webstatistics.ru
BAD: SOA record not found for zone.webstatistics.ru
BAD: zone.webstatistics.ru has NO authoritative nameservers!
BAD: All zone transfer attempts of zone.webstatistics.ru failed!
0 failures, 6 warnings, 6 errors.
generate#

В данном случае в зоне делегируется поддомен zone.webstatistics.ru, для которого не удается найти авторитативных серверов.

Кроме того, в отчете сообщается и о других ошибках описания зоны, например, о синонимах в NS записях.

К сожалению, в dnswalk нельзя указать сервер, который мы собираемся тестировать. Сервер берется из resolv.conf. В конце концов, нужный нам сервер можно указать первым в resolv.conf и тогда dnswalk будет работать с ним.

До сих пор мы работали с программами, которые позволяют анализировать описание зон путем использования DNS серверов. Однако, проверить правильность описания зоны можно просто, прочитав файл зоны. Проверив его на типичные ошибки, можно сгенерировать соответствующий отчет. Для этой цели написана программа nslint:

generate# nslint
nslint: "cname" referenced by other "cname" or "mx": ns.webstatistics.ru.
nslint: name referenced without other records: generate.polyn.kiae.su.
nslint: missing "ptr": paul.webstatistics.ru. -> 144.206.192.50
nslint: "cname" referenced by other "cname" or "mx": user.webstatistics.ru.
nslint: missing "ptr": host.webstatistics.ru. -> 144.206.192.63
nslint: missing "a": localhost.polyn.kiae.su. -> 127.0.0.1
nslint: missing "ptr": webstatistics.ru. -> 144.206.192.60
nslint: name referenced without other records: ns4.nic.ru.
nslint: missing "ptr": www.webstatistics.ru. -> 144.206.160.32
nslint: missing "ptr": www.webstatistics.ru. -> 144.206.192.60
nslint: missing "ptr": www.webstatistics.ru. -> 144.206.192.61
nslint: missing "ptr": www.webstatistics.ru. -> 144.206.192.62
nslint: 144.206.192.60 in use by www.webstatistics.ru. and webstatistics.ru.
generate#

В данном случае вообще не нужно задавать никаких аргументов, хотя они у nslint есть. Программа читает файлы конфигурации BIND (распознает как новый так и старый форматы), находит описания зон на локальном сервере и проверяет их на наличие ошибок. Понятно, что nslint имеет смысл запускать только на том компьютере, где установлен BIND.

Кроме тестирования работы системы DNS и серверов доменных имен существуют еще утилиты генерации описания зон и проверки файлов конфигурации. В для этой цели можно, например, использовать пакет dnsutl от Peter Miller.

Существует еще одна возможность тестирования работы сервера доменных имен - анализ его log-файла. Для выполнения этих действий написан скрипт, который назывался lamers.sh. В последнее время его переписали и он работает с DOC, т.е. кроме анализа log-файла еще и тестирует сервер путем вызова программ из DOC.

Ниже приведен основной кусочек скрипта, который, собственно, и ищет сообщения о неправильном делегировании:

```
#------------------------------------------------------------------------
# See if there are any lamers
#------------------------------------------------------------------------
grep "Lame" $LOGFILE | tr A-Z a-z | grep -v "*" | awk '{
if (length($9) == 2)
next
print substr($9, 2, length($9) - 2), substr($11, 2, length($11) - 2) }' |
sort -u | awk '{
printf("%s %s
", $1, $2)
}' > $LAMERS
```

В BIND версии 9 можно настроить сервер таким образом, чтобы он сам собирал сообщения о некорректном делегировании:

```
channel "lammers_log" {
file "lammers.log";
severity info;
};
category "lame-servers" { "lammers_log"; };
```

Этот фрагмент нужно вставить в опцию logging в файле настройки named.conf.

Тогда при обнаружении некорректного делегирования в этом файле будут появляться строчки типа:

```
lame server resolving 'www.ietf.net' (in 'ietf.NET'?): 218.145.29.146#53
```

Мы уже упоминали DOC (domain obscenity control) в контексте lamers. На самом деле этот продукт можно использовать совершенно самостоятельно. К написанию первоначальной версии DOC приложил руку Paul Moskapetris (отец DNS). Одно время DOC входило в состав дистрибутива BIND. Сейчас программу можно найти среди дополнительного программного обеспечения (ports) для большинства Unix-клонов.

Краткий отчет программы DOC будет выглядеть следующим образом:

```
generate# /usr/local/bin/doc alpha.ru.
Doc-2.1.4: doc alpha.ru.
Doc-2.1.4: Starting test of alpha.ru. parent is ru.
Doc-2.1.4: Test date - Thu Nov 14 16:40:13 MSK 2002
;; res_nsend to server ns2.nic.fr. 192.93.0.4: Operation timed out
DIGERR (UNKNOWN): dig @ns2.nic.fr. for SOA of parent (ru.) failed
Summary:
ERRORS found for alpha.ru. (count: 2)
Done testing alpha.ru. Thu Nov 14 16:40:24 MSK 2002
```

```
generate#
```

На самом деле интересно, что же за ошибки существуют в описании зоны. Для этого следует обратиться к файлу log.alpha.ru., который генерирует программа. Информацию об ошибках можно получить и на стандартный вывод:

```
generate# /usr/local/bin/doc -e alpha.ru.
Doc-2.1.4: doc -e alpha.ru.
Doc-2.1.4: Starting test of alpha.ru. parent is ru.
Doc-2.1.4: Test date - Thu Nov 14 16:45:33 MSK 2002
ERROR: NS list from alpha.ru. authoritative servers does not
=== match NS list from parent (ru.) servers
ERROR: ns.alpha.ru. claims to be authoritative, but does not appear in
NS list from authoritative servers
Summary:
ERRORS found for alpha.ru. (count: 2)
Done testing alpha.ru. Thu Nov 14 16:45:35 MSK 2002

generate#
```

Файловый отчет гораздо подробнее. Он кроме всего прочего содержит сами записи, в которых была обнаружена ошибка.

DOC использует dig Для получения содержания зоны, поэтому просто так без dig он работать не будет.

Есть еще одна любопытная программа - nsping. Ее не следует путать с "тезкой", которая применяется для Lotus Notes и Domino. Nsping тестирует серверы методом "случайного тыка". Основная идея заключается в том, чтобы обойти кэширование записей описания ресурсов и таким образом получить правдоподобную информацию о работоспособности сервера.

Отчет nsping выглядит следующим образом:

```
generate# /usr/local/sbin/nsping -z kiae.ru localhost
NSPING localhost (127.0.0.1): Domain = "kiae.ru", Type = "IN A"
+ [ 0 ] 159 bytes from 127.0.0.1: 0.471 ms [ 0.000 san-avg ]
+ [ 1 ] 158 bytes from 127.0.0.1: 0.433 ms [ 0.452 san-avg ]
+ [ 2 ] 159 bytes from 127.0.0.1: 0.492 ms [ 0.465 san-avg ]
+ [ 3 ] 159 bytes from 127.0.0.1: 0.474 ms [ 0.467 san-avg ]
+ [ 4 ] 159 bytes from 127.0.0.1: 0.443 ms [ 0.463 san-avg ]
+ [ 5 ] 158 bytes from 127.0.0.1: 0.426 ms [ 0.456 san-avg ]
^C
Total Sent: [ 6 ] Total Received: [ 6 ] Missed: [ 0 ] Lagged [ 0 ]
Ave/Max/Min: 0.456 / 0.492 / 0.426
generate#
```

В качестве первого параметра мы заказали зону, из которой хотели бы получать адреса, а в качестве второго параметра сервер (в данном случае localhost), который обрабатывал бы наши рекурсивные запросы.

Можно попросить оценить время получения и другого типа записей, например MX-записей:

```
generate# /usr/local/sbin/nsping -z rambler -T mx localhost
NSPING localhost (127.0.0.1): Domain = "rambler", Type = "IN MX"
+ [ 0 ] 455 bytes from 127.0.0.1: 0.712 ms [ 0.000 san-avg ]
+ [ 1 ] 454 bytes from 127.0.0.1: 0.625 ms [ 0.668 san-avg ]
+ [ 2 ] 455 bytes from 127.0.0.1: 0.770 ms [ 0.702 san-avg ]
+ [ 3 ] 455 bytes from 127.0.0.1: 0.841 ms [ 0.737 san-avg ]
+ [ 4 ] 455 bytes from 127.0.0.1: 0.761 ms [ 0.742 san-avg ]
+ [ 5 ] 454 bytes from 127.0.0.1: 0.626 ms [ 0.723 san-avg ]
^C
Total Sent: [ 6 ] Total Received: [ 6 ] Missed: [ 0 ] Lagged [ 0 ]
Ave/Max/Min: 0.723 / 0.841 / 0.625
generate#
```

Любопытно сравнить время получения информации с различных серверов. Например, с локального и с авторитативного сервера домена Microsoft:

```
generate# /usr/local/sbin/nsping -c 1 -z microsoft.com. -T ns dns1.dc.msft.net.
NSPING dns1.dc.msft.net. (207.68.128.151): Domain = "microsoft.com.", Type = "IN
NS"
- [ 0 ] 113 bytes from 207.68.128.151: 126.201 ms [ 0.000 san-avg ]

Total Sent: [ 2 ] Total Received: [ 1 ] Missed: [ 1 ] Lagged [ 0 ]
Ave/Max/Min: 126.201 / 126.201 / 126.201
generate# /usr/local/sbin/nsping -c 1 -z microsoft.com. -T ns localhost
NSPING localhost (127.0.0.1): Domain = "microsoft.com.", Type = "IN NS"
+ [ 0 ] 266 bytes from 127.0.0.1: 0.610 ms [ 0.000 san-avg ]

Total Sent: [ 2 ] Total Received: [ 1 ] Missed: [ 1 ] Lagged [ 0 ]
Ave/Max/Min: 0.610 / 0.610 / 0.610
generate#
```

Из отчета ясно, что с авторитативного сервера оно на два порядка больше.

Если появилось желание попробовать nsping, то программку можно установить из набора готового программного обеспечения ports. Однако нужно быть готовым к тому, что этот пустячок потянет за собой установку компилятора ADA и много другой сопутствующей мелочевки, которая, по большому счету, никому на вашей машине не нужна.

В заключении оговоримся, что перечисленные и описанные нами программы - это далеко не полный перечень средств, которые существуют в природе для решения проблем, которые возникают при работе системы доменных имен. В реальной жизни далеко не все администраторы даже догадываются о существовании этого программного обеспечения, обходясь стандартным набором из дистрибутива BIND.

Рекомендованная литература:

1. P. Mockapetris. RFC-1035. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. ISI, 1987. (http://www.ietf.org/rfc/rfc1035.txt?number=1035)

Полезные ссылки:

1. http://cr.yp.to/djbdns.html -странички djbdns, написанные профессором Бернштейном. Это первоисточник.
2. http://djbdns.org/ - странички группы любителей djbdns. Довольно много информации по патеку, но гораздо больше его особенностей можно получить подписавшись на конференцию BIND. J
3. http://www.canb.auug.org.au/~millerp/dnsutl/dnsutl.html - сайт dnsutl - пакета генерации описания зон. Лучше все же писать зоны самому, а прибегать к автоматизации только в совершенно прозрачных случаях.
4. http://www.visi.com/~barr/dnswalk/ - страница dnswalk. Последня версия программы датируется 2000 годом. Новее найти не удалось.
5. http://www.ripe.net/ripe/mail-archives/dns-wg/1994/msg00013.html - это прототип RFC 1731. Здесь перечислены средства отладки описания зон и основные проблемы, которые могут быть решены при помощи этих средств.