Network Working Group
Request for Comments: 1034
Obsoletes: RFCs 882, 883, 973

P. Mockapetris
ISI

November 1987

Доменные имена - концепции и возможности

DOMAIN NAMES - CONCEPTS AND FACILITIES

1. Статус документа

Данный RFC служит введением в систему доменных имен (DNS[1]) и не содержит многих деталей, которые описаны в другом документе серии RFC, «Доменные имена - реализация и спецификация» [RFC-1035]. Второй RFC предполагает знакомство читателя с основными концепциями, рассмотренными здесь.

Подмножество функций и типов данных DNS формирует официальный протокол. Этот протокол включает стандартные запросы и отклики на них, а также большинство форматов данных Internet (например, адресов хостов).

Однако система доменных имен умышленно делается расширяемой. Исследователи постоянно предлагают, реализуют и проверяют новые типы данных, запросов, классов, функций и т. п. Таким образом, хотя компоненты официального протокола предполагаются в целом неизменными и обеспечивающими эксплуатацию системы, следует постоянно быть готовыми к экспериментальному поведению расширений, выходящих за рамки протокола. Экспериментальные и устаревшие функции явно отмечены в этой серии RFC - применять такие возможности следует с осторожностью.

Следует специально предупредить читателя, что приведенные здесь примеры не являются полными и могут не соответствовать современному состоянию - они служат прежде всего учебным целям. Документ может распространяться без ограничений.

2. Введение

Данный RFC служит введением в систему доменных имен, ее использование для электронной почты Internet и для поддержки адресов хостов, а также описывает протоколы и серверы, применяемые для реализации DNS.

2.1. История доменных имен

Побудительным мотивом для разработки системы доменных имен послужил сети Internet.

- Имена хостов для отображения на адреса поддерживались Сетевым информационным центром (NIC[2]) в одном файле (HOSTS.TXT), который по протоколу FTP получали все хосты [RFC-952, RFC-953]. Общая полоса сети, расходуемая на распространение новой версии этого фала пропорциональна квадрату числа хостов в сети и даже при использовании многоуровневой системы серверов FTP нагрузка на FTP-сервер хоста NIC весьма значительна. Взрывной рост числа хостов в сети не сулит хорошего будущего.
- «Население» сети претерпевает качественные изменения. Хосты совместного использования, характерные для исходной сети ARPANET, меняются на локальные сети рабочих станций. Организации самостоятельно администрируют свои адреса и имена, но вынуждены ждать от NIC внесения изменений в файл HOSTS.TXT для обеспечения доступности внесенных изменений в сети Internet. Для организаций также желательна локальная структуризация пространства имен.
- Приложения Internet стали более сложными и требуют наличия службы имен общего пользования.

В результате перечисленных обстоятельств было предложено несколько идей по организации пространства имен и его поддержке [IEN-116, RFC-799, RFC-819, RFC-830]. Эти предложения отличались одно от другого, но все они использовали иерархическую схему пространства имен с построением иерархии в соответствии с организационной структурой и использованием символа точки «.» в качестве разделителя уровней иерархии имен. Схема с распределенной базой данных и обобщенными ресурсами была описана в [RFC-882, RFC-883]. На основе экспериментов с несколькими реализациями эта схема была развита в решение, описанное в данном документе.

Термины «домен» или «доменное имя» широко используются и за пределами контекста DNS, описанного здесь. Очень часто термин «доменное имя» используется для обозначения имен, содержащих в своей структуре символы точки, но не связанных с DNS. Особенно характерно такое применение для адресации электронной почты [Quarterman 86].

2.2. Цели создания DNS

Цели создания DNS влияют на структуру системы.

- Основной целью является создание согласованного пространства имен, которые будут использоваться для обращения к ресурсам. Дабы избежать проблем, связанных со специальным кодированием, не следует требовать в именах использования сетевых идентификаторов, адресов, маршрутов и другой информации подобного типа.
- Размер базы данных и частота ее обновления диктуют распределенный характер базы данных с локальным кэшированием в целях повышения производительности. Модели, пытающиеся собрать всю информацию в одном месте, будут становиться все более дорогими и сложными и, следовательно, их нужно избегать. Тот же

[1]Domain Name System.
[2]Network Information Center.

принцип относится и к структуре пространства имен — в частности, механизмы создания и удаления имен тоже должны быть распределенными.

- При возникновении конфликтов между «стоимостью» получения данных и точностью кэширования источник данных должен контролировать выбор.

- Затраты на внедрение такого механизма требуют, чтобы он был полезен для решения широкого класса задач, а не ограничивался одним приложением. Должна обеспечиваться возможность использования имен для отыскания адресов хостов, почтовых адресов и другой, пока не определенной информации. Все данные, связанные с именем, помечаются определенным типом и запросы могут ограничиваться одним типом данных.

- Поскольку мы хотим обеспечить полезность пространства имен для разных и непохожих сетей и приложений, мы обеспечиваем возможность использования одного пространства имен с разными протоколами и различным управлением. Например, форматы адресации хостов различаются между протоколами, хотя все протоколы имеют нотацию для адресов. DNS делит все данные на классы, помечая их соответствующими тегами, что позволяет использовать разные форматы данных.

- Мы хотим обеспечить независимость транзакций DNS от используемых для их передачи коммуникационных систем. Некоторые системы могут использовать дейтаграммы для передачи запросов и откликов, организуя виртуальные устройства (каналы) только для транзакций, которым нужна надежность (например, для обновления баз данных или продолжительных транзакций), а другим системам виртуальные устройства могут требоваться постоянно.

- Система должна быть полезной для широкого спектра возможностей хостов. Должна обеспечиваться возможность использования системы как для персональных компьютеров, так и для хостов коллективного пользования, хотя способы работы могут различаться.

2.3. Предположения об использовании

Организация системы доменных имен базируется на некоторых допущениях о потребностях и способах ее применения сообществом пользователей, а устройство системы должно предотвращать возникновение множества проблем, характерных для баз данных общего назначения.

Ниже перечислены эти допущения.

- Общий размер базы данных на начальном этапе пропорционален числу хостов, использующих систему, но будет расти пропорционально числу пользователей этих хостов за счет добавления в DNS информации о почтовых адресах и других данных.

- Большая часть данных в системе (например, почтовые адреса и адреса хостов) будет изменяться весьма редко, но система должна обеспечивать возможность работы с подмножеством достаточно быстро (за минуты и даже секунды) изменяющихся данных.

- Административные границы, используемые для распределения зон ответственности в базе данных, будут обычно соответствовать организациям, имеющим один или множество хостов. Каждая организация, отвечающая за некий набор доменов, будет обеспечивать резервные серверы имен на собственных хостах этой организации или других хостах, которыми организация может пользоваться.

- Клиентам DNS следует обеспечивать возможность идентификации доверенных серверов имен, которые являются предпочтительными для использования по сравнению с серверами имен, не входящими в число «доверенных».

- Доступность информации более важна, чем своевременность обновлений или гарантии согласованности. Следовательно, процесс обновления позволяет выполнять обновление через пользователей системы взамен гарантированного одновременного обновления всех копий. При недоступности обновлений по причине отказа на хосте или в сети обычной практикой является использование старых данных и продолжение попыток обновления. Общая модель заключается в распространении копий по тайм-ауту обновления. Распространяющая данные сторона устанавливает значение тайм-аута, а ответственность за своевременное обновление ложится на получателя информации. В специальных случаях могут задаваться очень короткие интервалы обновления или владелец может запрещать копирование.

- В любой системе с распределенной базой данных любой конкретный сервер имен может получить запрос, ответ на который может быть дан только неким другим сервером. Существует две модели решения этой проблемы - «рекурсивная», когда первый сервер транслирует (передает) запрос клиента другому серверу, и «итеративная», когда сервер указывает клиенту другой сервер, который способен ответить на запрос клиента. Обе модели имеют преимущества и недостатки, но рекурсивная модель предпочтительна при обслуживании запросов на базе дейтаграмм. Система DNS требует реализации итеративной модели, но позволяет использовать рекурсивную модель в качестве опции.

Система доменных имен предполагает, что все данные хранятся в мастер-файлах, хранящихся на хостах, которые используют эту систему. Эти файлы обновляются локальными администраторами. Файлы используют текстовый формат, считываются локальным сервером имен и, следовательно, доступны через сервер пользователям DNS. Пользовательские программы обращаются к службе доменных имен через специальные программы, называемые распознавателями (resolver).

Стандартный формат мастер-файлов позволяет осуществлять обмен этими файлами между хостами (по протоколу FTP, электронной почте или с помощью иных механизмов), что весьма полезно в тех случаях, когда организация хочет иметь свой домен, но не желает поддерживать свой сервер имен. Организация в таком случае может поддерживать мастер-файлы локально, передавая их копии на хост, используемый в качестве сервера имен, согласованным с администратором этого хоста способом.

Серверы имен и распознаватели каждого хоста настраиваются локальным сетевым администратором [RFC-1033]. Для сервера имен конфигурационные параметры включают имена локальных мастер-файлов и инструкции по загрузке нелокальных первичных файлов с внешних серверов. Сервер имен использует мастер-файлы или их копии для загрузки своих зон. Для распознавателей конфигурационные параметры указывают серверы имен, которые следует использовать в качестве первичных источников информации.

Система доменных имен определяет процедуры доступа к данным и ссылок на другие серверы имен. DNS также определяет процедуры кэширования данных и периодического обновления данных, определяемые системным администратором.

Системный администратор обеспечивает:

- определение границ зон;
- первичные файлы;
- идентификацию первичных файлов;
- заявление желаемой политики обновления.

DNS обеспечивает:

- стандартный формат данных о ресурсах;
- стандартные методы запросов к базе данных;
- стандартные методы позволяющие локальному серверу имен получать обновления от внешних серверов имен.

2.4. Элементы DNS

DNS состоит из трех основных компонент:

- **Пространство доменных имен** и **записи о ресурсах**, которые обеспечивают спецификации для пространства имен со свободной структурой и данных, связанных с именами. Концептуально, каждый узел и ветвь дерева пространства доменных имен именует набор информации и запросные операции спредставляют собой попытку получить заданные типы информации из конкретного набора. Запрос указывает интересующее доменное имя и описывает тип желательной информации о ресурсе. Например, в Internet некоторые доменные имена используются для идентификации хостов и запросы адресных ресурсов будут возвращать адреса хостов Internet.
- **Сервер имен** представляет собой прогрраммные компоненты, которые поддерживают информацию о структуре доменного дерева и набор информации. Сервер имен может кэшировать структуру или набор информации о любой части доменного дерева, но в общем случае конкретный сервер имеет полную информацию о подмножестве пространства домена и указатели на другие серверы имен, которые могут использоваться для получения информации о любой части доменного дерева. Серверы имен знают части доменного дерева, для которых они имеют полную полную информацию. Применительно к этим частям пространства имен сервер считается полномочным (AUTHORITY). Полномочная информация организуется в блоки, называемые зонами (ZONE) и эти зоны могут автоматически распространяться серверам имен, которые являются резервными для данных зон.
- **Распознаватель** (RESOLVER) представляет собой программу, получающую от сервера имен информацию в ответ на запрос клиента. Распознаватель должен обеспечивать доступ по крайней мере к одному серверу имен и использовать данные этого сервера для ответа на запросы напрямую или переадресации запроса другим серверам имен, указанных основным сервером. Распознаватель обычно является частью операционной системы, поэтому применения специальных протоколов для взаимодействия между распознавателем и пользовательскими программами не требуется.

Эти три компоненты приблизительно соответствуют трем уровням или представлениям (view) DNS:

- с точки зрения пользователя доступ к DNS осуществляется с помощью простой процедуры или обращения ОС к локальному преобразователю; пространство домена состоит из одного дерева и пользователь может запросить данные из любой части этого дерева;
- с точки зрения распознавателя DNS представляет собой неизвестное число серверов имен; каждый сервер поддерживает одну или множество частей дерева данных всего домена, с точки зрения распознавателя все эти базы данных являются статическими;
- с точки зрения сервера имен DNS состоит из раздельных наборов локальной информации, называемых зонами; сервер имен должен периодически обновлять свои зоны из мастер-копий, хранящихся в локальных файлах или на внешних серверах имен; сервер имен должен постоянно обрабатывать запросы, получаемые от распознавателей.

В интересах производительности реализации могут объединять в себе эти функции. Например, Распознаватель на той же машине, что и сервер имен, может совместно с сервером использовать базу данных, включающую зоны, которые обслуживает сервер, пользуясь в то же время данными из своего кэша.

3. Пространство доменных имен и записи о ресурсах

3.1. Спецификации пространства имен и терминология

Пространство имен имеет структуру дерева. Каждый узел и ветвь (лист) дерева соответствуют набору ресурсов (который может быть пустым). DNS не различает внутренние узлы и ветви и здесь термин узел относится к обоим.

Каждый узел имеет метку размером от 0 до 63 октетов. Братские узлы могут иметь разные метки и, в то же время, одна метка может использоваться узлами, которые не являются братскими. Метка null (нулевого размера) имеет специальное назначение и зарезервирована для корня.

Доменное имя узла представляет собой список меток на пути от узла до корня дерева. По соглашению метки, образующие доменное имя печатаются и считываются слева направо от наиболее специфичной (низшей, наиболее удаленной от корня) к наименее специфичной (верхней, ближайшей к корню).

В программах, имеющих дело с доменными именами, такие имена следует представлять в форме последовательности меток, каждая из которых имеет октет размера, за которым следует строка октетов. Поскольку все доменные имена заканчиваются у корня, меткой которого является null, внутреннее представление может использовать нулевое значение байта размера в качестве признака завершения доменного имени.

По соглашению доменные имена могут сохраняться в любом регистре (строчные или прописные буквы), все функции работы с доменными именами выполняются независимо от регистра для кодировки ASCII с нулевым значением старшего бита. Это означает, что вы можете создать узел с меткой «A» или «a», но не использовать символы разных регистров вперемешку. При получении доменного имени регистр символов следует сохранять. Это разумно делать потому, что в будущем может потребоваться поддержка в доменных именах расширенного набора символов и при сохранении регистра старые программы не придется менять.

Когда пользователю нужно напечатать доменное имя, размер меток опускается и для разделения меток используется символ точки «.». Поскольку полное доменное имя завершается корневой меткой, такая нотация ведет к появлению завершающей точки в доменных именах. Это свойство будет использоваться для того, чтобы различать:

- строки, представляющие полные доменные имена (например, poneria.ISI.EDU.), которые иногда называют «абсолютными»;
- строки, представляющие начальные метки доменного имени (например, poneria), которые будут дополняться до полного имени локальными программами, знающими имя своего локального домена (например, poneria.ISI.EDU.); такие имена часто называют «относительными».

Относительные имена привязываются к общеизвестному источнику или к списку доменов, заданных для поиска. Относительные имена используются в основном на уровне пользовательского интерфейса, где их интерпретация может меняться в разных реализациях, и в master-файлах, где эти имена привязываются к имени исходного домена (origin domain name). Типичная интерпретация использует корень «.» в качестве единственной исходной точки или в качестве одного из элементов списка поиска, поэтому относительные имена, содержащие более одной метки, зачастую используют там, где хочется опустить завершающую точку.

Для упрощения реализации общее число октетов, представляющих доменное имя (т. е., октетов меток и их размеров), ограничено значением 255.

Домен идентифицируется именем и включает ту часть пространства доменных имен, которая лежит ниже имени этого домена. Домен является субдоменом другого домена, если он содержится в том домене. Для проверки принадлежности достаточно убедиться, что в конце имени субдомена содержится имя домена. Например, субдоменами домена A.B.C.D являются B.C.D, C.D, D и « ».

3.2. Административное руководство по использованию

Технические спецификации DNS не задают какую-либо конкретную структуру дерева или правила выбора меток - спецификации сделаны максимально обобщенными, чтобы их можно было использовать для построения любый приложений. В частности, система разработана так, чтобы не возникало привязок к границам сетей, серверам имен и т. п. Причина этого заключается не в том, что пространству имен не следует иметь подразумеваемой семантики, а скорее в том, что выбор подразумеваемой семантики следует оставлять открытым, а также в возможности использования разной подразумеваемой семантики в разных частях дерева имен. Например, домен IN-ADDR.ARPA организован и распределен по адресам сетей и хостов, поскольку роль этого домена заключается в трансляции номеров сетей и хостов на имена; домены NetBIOS [RFC-1001, RFC-1002] являются плоскими, поскольку это лучше подходит для приложений.

Однако имеется ряд рекомендаций, применимых к «обычной» части пространства имен, используемой для хостов, почтовых ящиков и т. п., и позволяющих унифицировать использование пространства имен, обеспечить возможность роста и минимизировать проблемы перехода от старой таблицы хостов. Политика для верхних уровней дерева имен изначально рассмотрена в RFC 920. Современная политика для верхних уровней приведена в [RFC-1032]. Вопросы перехода для MILNET рассмотрены в [RFC-1031].

Нижележащие домены, которые, в конечном итоге, делятся на множество зон, должны обеспечивать ветвление на вершине домена так, чтобы была возможна декомпозиция без переименования. Метки узлов, в которых используются специальные символы, цифры в начале и т. п., будут нарушать работу старых программ, которым присущи большие ограничения.

3.3. Техническое руководство по использованию

Прежде, чем DNS станет возможно использовать для хранения имен того или иного типа объектов, должны быть разработаны:

- соглашение об отображении между именами объектов и доменными именами, описывающее доступ к информации об объекте;
- типы RR и форматы данных для описания объекта.

Эти правила могут оказаться и достаточно простыми и очень сложными. Очень часто разработчикам потребуется принимать во внимание существующие форматы и решать вопросы совместимости с используемыми решениями. Может потребоваться множество отображений или многоуровневое отображение.

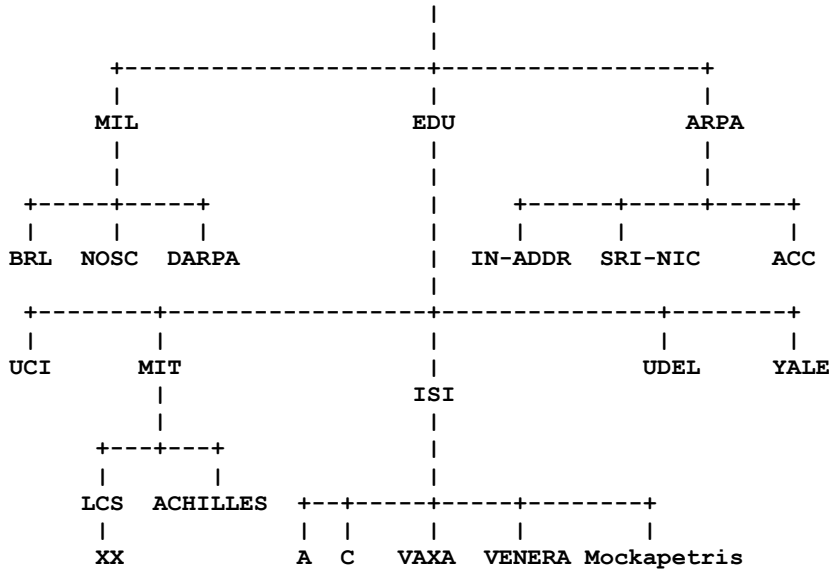
Для хостов отображение зависит от используемого синтаксиса имен хостов, который является подмножеством синтаксиса текстового представления доменных имен, а также форматов RR для описания адресов хостов и т. п. Поскольку нам требуется надежное обратное преобразование, определено также специальное отображение адресов на домен IN-ADDR.ARPA.

Для почтовых ящиков отображение несколько сложнее. Обычный почтовый адрес <local-part>@<mail-domain> отображается на доменное имя путем преобразования <local-part> в одну метку (независимо от наличия точек в локальной части), а <mail-domain> - в доменное имя с использованием текстового формата для доменных имен (точки служат границами меток) с последующим объединением, формирующим одно полное доменное имя. Таким образом, почтовый ящик HOSTMASTER@SRI-NIC.ARPA представляется доменным именем HOSTMASTER.SRI-NIC.ARPA. При выборе такого преобразования должна также приниматься во внимание схема почтового обмена [RFC-974].

Обычному пользователю нет нужды задумываться об определении этих правил, но следует понимать, что правила обычно являются компромиссом между требованиями совместимости со старыми решениями при взаимодействии между различными определениями объектов и неизбежным желанием расширить возможности при определении новых правил. Способ использования DNS для поддержки некоторых объектов зачастую может играть более важную роль, нежели присущие DNS ограничения.

3.4. Пример пространства имен

На рисунке показана часть современного пространства доменных имен. Эта часть пространства имен используется во многих примерах в данном RFC. Отметим, что показанное на рисунке дерево содержит лишь малую часть реального пространства.



В этом примере корневой домен включает три субдомена - MIL, EDU, ARPA. Домен LCS.MIT.EDU имеет один субдомен с именем XX.LCS.MIT.EDU. Все остальные ветви на рисунке также обозначают домены.

3.5. Синтаксис имен

Спецификация DNS пытается сохранить максимальный уровень обобщения для правил конструирования доменных имен. Идея заключается в том, чтобы имя любого существующего объекта можно было выразить в форме доменного имени с минимальными изменениями.

Однако при выделении доменного имени для объекта предусмотрительный пользователь будет выбирать такое имя, которое удовлетворяет как правилам DNS, так и существующим для объекта правилам, если эти правила опубликованы или вносятся используемыми программами.

Например, при именовании почтового объекта пользователю следует выполнить правила данного документа и RFC 822. При создании нового имени хоста следует выполнять правила для HOSTS.TXT. Это позволит предотвратить проблемы при переводе старых программ на работу с доменными именами.

Приведенный ниже синтаксис будет обеспечивать лишь незначительные проблемы для большинства приложений (например, почта, TELNET) при использовании доменных имен.

```
<domain>      ::= <subdomain> | " "
<subdomain>   ::= <label> | <subdomain> "." <label>
<label>       ::= <letter> [ [ <ldh-str> ] <let-dig> ]
<ldh-str>     ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>
<let-dig-hyp>::= <let-dig> | "-"
<let-dig>     ::= <letter> | <digit>
<letter>      ::= любая из 52 букв от A до Z (верхний регистр) или от a до z (нижний регистр)
<digit>       ::= любая из десятичных цифр от 0 до 9
```

Отметим, что строчные и прописные буквы в доменных именах не различаются, хотя можно использовать символы в обоих регистрах.

Метки должны следовать правилам для имен хостов в домене ARPANET. Метки должны начинаться с буквы, заканчиваться буквой или цифрой и могут включать только буквы, цифры и знак дефиса (-). Существует также ограничение на размер меток (не более 63 символов).

Например, показанные ниже имена могут использоваться для хостов Internet:

3.6. Записи о ресурсах

Доменное имя идентифицирует узел. Каждый узел имеет (возможно, пустой) набор информации о ресурсах. Набор данных, связанный с конкретным именем, состоит из отдельных записей о ресурсах (RR). Порядок RR в наборе не имеет значения и его не требуется представлять каждому серверу имен, преобразователю или иной компоненте DNS.

При обсуждении конкретной RR используются следующие категории:

owner владелец доменного имени, к которому относится RR.

type 16-битовое значение, задающее тип ресурса в данной записи; тип относится к абстрактному ресурсу.

В этом документе используются следующие типы:

A адрес хоста;

CNAME идентифицирует каноническое имя для псевдонима;

HINFO идентифицирует процессор (CPU) и операционную систему (OS) хоста;

MX идентифицирует узел почтового обмена для домена (см. [RFC-974);

NS полномочный сервер имен для домена;

PTR указатель на другую часть пространства доменных имен;

SOA идентифицирует начало зоны полномочий.

class 16-битовое значение, задающее семейство протоколов или экземпляр протокола.

В этом документе используются два класса:

IN Internet;

CH Chaos.

TTL время жизни RR - 32-битовое целое число, задающее число секунд; используется в основном кэширующими распознавателями. Значение TTL задает продолжительность нахождения записи RR в кэше.

RDATA тип или (иногда) класс в зависимости от данных, которые описывают ресурс:

A для класса IN это 32-битовый адрес IP;

для класса CH это доменное имя, за которым следует 16-битовый восьмеричный адрес Chaos;

CNAME доменное имя;

MX 16-битовое значение, задающее уровень предпочтения (меньшее значение предпочтительней). за которым следует имя хоста, обеспечивающего почтовый обмен для домена владельца;

NS имя хоста;

PTR доменное имя;

SOA несколько полей.

Имя владельца зачастую задается неявно вместо его включения в RR. Например, многие серверы имен формируют внутреннюю структуру дерева или хэш для пространства имен и связывают записи RR с узлами. Оставшаяся часть RR имеет фиксированный заголовок (тип, класс, TTL), который согласован для всех RR, а также переменную часть (RDATA), соответствующую задаче описания ресурса.

Поле TTL задает период времени, в течение которого RR может сохраняться в кэше. Этот предел не относится к полномочным данным в зоне - такие записи тоже стареют, но это происходит в соответствии с политикой обновления для зоны. Значение TTL выделяется администратором зоны, из которой происходят данные. Хотя малые значения TTL могут служить для минимизации времени кэширования, а TTL=0 - для запрета кэширования, в реальной практике Internet предполагается, что эти значения для обычного хоста задают время порядка нескольких дней. Если предполагается внесение изменений, значение TTL можно заранее уменьшить с целью минимизации срока распространения вносимых изменений, а по завершении обновления - восстановить прежнее время жизни.

Данные в разделе RDATA записи RR передаются в форме комбинации двоичных строк и доменных имен. Доменные имена часто используют в качестве «указателей» на друге данные DNS.

3.6.1. Текстовое представление RR

Записи RR представляются в пакетах протокола DNS в двоичной форме, а при хранении на серверах имен и распознавателях обычно используется кодирование. В данном документе для показа содержимого RR применяется стиль представления, подобный стилю, используемому в первичных файлах. В таком формате большинство RR представляются в форме одной строки, хотя могут использоваться и многострочные записи со скобками.

Начало строки задает владельца RR. Если строка начинается с пустого пространства, предполагается, что владелец записи совпадает с владельцем предыдущей RR. Для наглядности представления могут использоваться пустые строки.

После владельца указывается TTL, тип и класс RR. Для класса и типа используется определенная выше мнемоника, а TTL задается целым числом перед полем типа. Во избежание неоднозначностей при разборе мнемоника для типов и классов не связана между собой. Значения TTL являются целым числами, а мнемоника типа всегда указывается последней. Класс IN и значение TTL в примерах часто опускается в целях простоты и наглядности.

Данные о ресурсе или раздел RDATA записи RR задаются с использованием типичного представления данных.

Например, записи RR, передаваемые в сообщении, можно представить в виде:

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

В записях MX RR имеется раздел RDATA, включающий 16-битовое целое число, за которым следует доменное имя. Адресные RR используют стандартный формат адресов IP (32 бита).

В приведенном выше примере даны шесть RR (по две записи RR для каждого из трех доменных имен).

Аналогично, мы можем видеть:

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

В этом примере показаны два адреса разных классов для хоста XX.LCS.MIT.EDU.

3.6.2. Псевдонимы и канонические имена

В существующих системах хосты и другие ресурсы часто имеют по несколько имен, идентифицирующих один ресурс. Например, имена C.ISI.EDU и USC-ISIC.ARPA идентифицируют один хост. Аналогичная ситуация наблюдается для почтовых ящиков - например, Mockapetris@C.ISI.EDU, Mockapetris@B.ISI.EDU и PVM@ISI.EDU указывают на один почтовый ящик (хотя механизм в этом случае несколько сложнее).

Большинство из таких систем различают имена во множестве имеющихся - одно из имен является каноническим или основным, а все прочие - псевдонимами.

Доменная система представляет такую возможность за счет использования записей для канонических имен (CNAME). Запись CNAME RR идентифицирует имя своего владельца в качестве псевдонима и задает соответствующее каноническое имя в разделе RDATA записи RR. Если на узле присутствует CNAME RR, других данных не должно быть - это гарантирует совпадение данных для канонического имени и его псевдонимов. Это правило также обеспечивает возможность кэширования CNAME без проверки на полномочном сервере других типов RR.

Записи CNAME RR играют в программах DNS особую роль. Когда серверу имен не удается найти желаемую RR в наборе ресурсов, связанных с доменным именем, он проверяет наличие а наборе ресурсов записи CNAME соответствующего класса. Если такая запись имеется, сервер имен включает запись CNAME в отклик и повторяет запрос для доменного имени, заданного в поле данных записи CNAME. Единственным исключением из этого правила является то, что запросы, которые соответствуют типу CNAME, не повторяются.

Предположим в качестве примера, что сервер имен обрабатывает запрос для USC-ISIC.ARPA, пытаясь получить данные типа A, и нашел следующие записи о ресурсах:

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

Обе записи RR будут возвращены в отклике на запрос типа A, тогда как при запросе типа CNAME или * следует возвращать только CNAME.

Доменные имена в RR, которые указывают на другое имя, всегда должны ссылаться на первичное имя, а не на псевдоним. Это позволит избежать ненужных перенаправлений при доступе к данным. Например, для упомянутого выше хоста запись инверсного преобразования будет иметь вид:

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

а не указывать на псевдоним USC-ISIC.ARPA. Программам DNS в соответствии с принципами отказоустойчивости не следует давать сбои при возникновении цепочек или петель CNAME; цепочки CNAME следует разбирать, а при обнаружении петли - сообщать об ошибке.

3.7. Запросы

Запросы представляют собой сообщения, которые передаются серверам имен для инициирования отклика. В Internet запросы передаются с использованием дейтаграмм UDP или соединений TCP. Отклик сервера имен отвечает на заданные вопросы, указывает запрашивающей стороне другой набор серверов имен для запроса данных или сигнализирует о возникновении ошибки.

В общем случае пользователь не генерирует запросы самостоятельно - вместо этого он обращается к преобразователю (resolver), который, в свою очередь, передает запрос или множество запросов серверам имен и обеспечивает обработку сообщений об ошибках и ссылок на другие серверы. Естественно, вопросы, которые могут задаваться серверам имен, формируют тип сервиса, который может обеспечивать распознаватель имен.

Запросы и отклики DNS передаются в форме стандартных сообщений. Сообщение включает заголовок, содержащий множество фиксированных полей, и четыре раздела, которые могут передавать параметры запроса и RR.

Наиболее важным полем заголовка является 4-битовое поле, называемое кодом операции (opcode) и позволяющее разделять разнотипные запросы. Это поле позволяет задать 16 значений, одно из которых (стандартный запрос) является частью официального протокола, два (реверсный запрос и запрос состояния) являются опциями, одно (завершение) относится к устаревшим, а остальные пока не используются.

Четыре раздела сообщения включают:

Question вопрос к серверу имен и другие параметры запроса;

Answer записи RR, которые напрямую отвечают на запрос;

Authority записи RR, описывающие другие полномочные серверы имен (может опционально включать SOA RR для полномочных данных в разделе answer);

Additional записи RR, которые могут быть полезны при использовании RR из остальных разделов.

Отметим, что содержимое (но не формат) этих разделов меняется в зависимости от кода операции в заголовке.

3.7.1. Стандартные запросы

Стандартный запрос задает искомое доменное имя (QNAME), тип (QTYPE) и класс (QCLASS) запроса, а также запрашивает соответствующие записи RR. Запросы этого типа составляют основную часть запросов DNS и мы будем использовать термин «запрос» для обозначения стандартного запроса, если явно не указано иное. Поля QTYPE и QCLASS имеют размер 16 битов и являются надмножеством определенных типов и классов.

Поле QTYPE может содержать:

<any type> соответствует заданному типу (например, A, PTR);

AXFR QTYPE для переноса зоны (AXFR);

MAILB соответствует всем почтовым ящикам, связанных RR (например, MB и MG);

* соответствует всем типам RR.

Поле QCLASS может содержать:

<any class> соответствует заданному классу (например, IN, CH);

* соответствует всем классам RR.

Используя искомое доменное имя, QTYPE и QCLASS, сервер имен ищет соответствующие запросу записи RR. В дополнение к имеющим отношение к запросу записям сервер может возвращать RR, указывающие сервер имен, которые имеет желаемую информацию, или RR, которые могут быть полезны при интерпретации других записей в отклике. Например, сервер имен, у которого нет запрашиваемой информации. Может знать другой сервер имен, у которого такая информация имеется, - в этом случае сервер, который возвращает доменное имя в соответствующих RR, может также возвратить RR, которые связывают это доменное имя с адресом.

Например, почтовая программа, которой нужно отправить письмо по адресу Mockapetris@ISI.EDU, может запрашивать у распознавателя почтовую информацию о домене ISI.EDU с помощью запроса QNAME=ISI.EDU, QTYPE=MX, QCLASS=IN. Раздел answer в отклике может иметь вид:

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

А дополнительный раздел может иметь вид:

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

Сервер предполагает, что клиент, запрашивающий информацию для обмена электронной почтой, может вслед за этим запросить адрес почтового сервера.

Отметим, что конструкция QCLASS=* требует особой интерпретации в части полномочий. Поскольку конкретный сервер имен может не знать всех классов, доступных в DNS, он никогда не будет знать, является ли он полномочным для всех классов. Следовательно, запросы QCLASS=* не могут считаться аутентичными.

3.7.2. Инверсные запросы (необязательно)

Серверы имен могут также поддерживать инверсные запросы для отображения конкретных ресурсов на доменные имена или определения доменных имен заданного ресурса. Например, стандартный запрос может отображать доменное имя на SOA RR, а соответствующий инверсный запрос может выполнять обратное отображение SOA RR на доменное имя.

Реализация этого сервиса не является обязательной для сервера имен, но все серверы имен должны по крайней мере понимать инверсные запросы и возвращать сообщение о том, что данный сервис не поддерживается.

Система доменных имен не может гарантировать полноты и уникальности для инверсных запросов, поскольку система DNS организована по именам доменов, а не адресам хостов или иным типам ресурсов. Инверсные запросы полезны в основном для отладки и поддержки баз данных.

Инверсные запросы не могут возвращать надлежащее значение TTL и не указывают ситуации, когда найденная RR является одной из множества существующих (например, один адрес для хоста, имеющего множество адресов). Следовательно, возвращаемые по инверсным запросам RR никогда не должны кэшироваться.

Инверсные запросы **не** подходят для отображения адресов хостов на их имена - вместо этого должен использоваться специальный домен IN-ADDR.ARPA.

Подробное описание инверсных запросов приведено в [RFC-1035].

3.8. Запросы состояния (эксперимент)

Будут определены впоследствии.

3.9. Завершение запросов (отменено)

Опциональные услуги завершения, описанные в RFC 882 и 883 удалены. В будущем может появиться переработанный сервис или коды операций будут использованы для иных целей.

4. Серверы имен

4.1. Введение

Серверы имен являются хранилищами информации, которая формирует базу доменных имен. Эта база делится на зоны, которые распределены по серверам имен. Хотя серверы имен могут поддерживать дополнительные функции и источники данных, основной задачей таких серверов является обработка запросов с использованием данных из зон сервера. По своему устройству серверы имен могут отвечать на запросы в простой форме - ответ может быть порожден с использованием только локальных данных и содержать запрашиваемую информацию или ссылку на другие серверы имен, которые «ближе» к запрашиваемым данным.

Конкретная зона может быть доступна на нескольких серверах имен чтобы обеспечить доступность информации при возникновении отказов на хосте или в коммуникационном канале. Административные меры требуют размещения каждой зоны по крайней мере на двух серверах - увеличение числа избыточных зон повышает надежность.

Конкретный сервер имен обычно поддерживает одну или множество зон, но является полномочным лишь для малой части дерева доменных имен. Сервер может поддерживать также некоторые кэшированные (неполномочные) данные о других частях дерева имен. Сервер имен маркирует свои отклики на запросы, указывая, является отклик полномочным или нет.

4.2. Деление базы данных на зоны

База доменных имен делится двумя способами - по классам и по «срезам» (cut) в пространстве имен между узлами.

Деление по классам выполняется просто. База данных для любого класса организуется, делегируется и поддерживается для каждого класса отдельно. Таким образом, по соглашению, пространства имен одинаковы всех классов, а разные классы можно рассматривать, как массив «параллельных» деревьев пространств имен. Отметим, что данные, связанные с узлами, будут различаться для этих различных параллельных классов. Основной причиной создания новых классов является необходимость введения нового формата данных для существующих типов или желание создать раздельно управляемую версию существующего пространства имен.

Внутри класса «срез» пространства имен может быть выполнен между любой парой смежных узлов. После выполнения всех срезов каждая группа связанного пространства имен представляет собой отдельную зону. Зона считается полномочной для всех имен связанного с ней региона. Отметим, что «срезы» в пространстве имен могут выполняться в разных местах для разных классов, серверы имен могут различаться и т. п.

Эти правила означают, что каждая зона имеет по крайней мере один узел и, следовательно, доменное имя, для которого эта зона является полномочной, а все узлы в конкретной зоне связаны между собой. С учетом древовидной структуры каждая зона имеет верхний узел, расположенный ближе всего к корню. Имя этого узла зачастую используется для идентификации зоны.

Можно (хотя и не приносит практической пользы) разделить пространство доменных имен так, чтобы каждый домен являлся отдельной зоной, или так, чтобы все узлы входили в одну зону. Вместо этого база данных делится в точках, где отдельные организации хотят взять субдерево под свой контроль. Контролирующая зону организация может по своему усмотрению менять данные в зоне, создавать новые ветви дерева, соединенные с зоной, удалять существующие узлы или делегировать новые подзоны своей зоны.

Если организация имеет сложную внутреннюю структуру, она может разделить свою зону, используя внутреннее делегирование (передачу полномочий). В некоторых случаях такое деление выполняется просто для упрощения поддержки базы данных.

4.2.1. Технические вопросы

Описывающие зону данные состоят из 4 основных частей:

- полномочные данные для всех узлов зоны;
- данные, определяющие верхний узел зоны (можно считать частью полномочных данных);
- данные, описывающие делегированные субзоны (т. е., срезы в нижней части зоны);
- данные для доступа к серверам имен для субзон (их еще называют «склеивающими» данными).

Все эти данные выражаются в форме RR что позволяет полностью описать зону в форме набора RR. Зоны целиком могут передаваться между серверами имен путем передачи RR за счет последовательности сообщений или передачи текстового представления первичных файлов по протоколу FTP.

Полномочные данные для зоны представляют собой просто все записи RR, связанные со всеми узлами от вершины зоны до узлов ветвей или узлов над срезами на нижнем краю зоны.

Являясь логически частью полномочных данных, записи RR, описывающие верхний узел зоны, имеют особое значение для управления зоной. Эти записи бывают двух типов - RR серверов имен (по одной записи для каждого сервера зоны) и одна SOA RR, описывающая параметры управления зоной.

Записи RR, описывающие срезы на дне зоны, являются NS RR, которые указывают на серверы имен для субзон. Поскольку срезы выполняются между узлами, эти RR **не** являются частью полномочных данных зоны и должны совпадать с соответствующими RR на вершинах субзон. Так как серверы имен всегда связаны с границами зон, записи NS RR можно найти лишь на узлах, являющихся вершинами той или иной зоны. В составляющих зоны данных записи NS RR находятся на вершинных узлах зон (и являются полномочными) и на срезах в нижней части зон (здесь эти данные не являются полномочными), но никогда не могут размещаться в промежутке.

Одной из целей структуры зоны является обеспечение для любой зоны наличия полного набора данных, требуемых для организации связи с серверами имен любой из субзон. Т. е., родительские зоны имеют всю информацию, требуемую для доступа к серверам имен своих дочерних зон. Записей NS RR с именами серверов для субзон зачастую недостаточно, поскольку такие записи содержат имена серверов, но не дают их адресов. В частности, если имя

сервера имен само находится в субзоне, может возникнуть ситуация, когда записи NS RR скажут нам, что для получения адреса сервера имен нам следует обратиться к серверу имен, адрес которого мы хотим узнать. Для решения этой проблемы в зоны включаются «склеивающие» записи, которые не относятся к полномочным данным зоны и содержат RR с адресами серверов имен. Эти RR требуются только в тех случаях, когда имя сервера имен находится «ниже» среза и используются только, как часть отклика со ссылкой.

4.2.2. Административные вопросы

Если организация хочет самостоятельно управлять своим доменом, сначала ей нужно идентифицировать подходящую родительскую зону и получить согласие ее владельца на передачу полномочий (делегирование) по управлению. Каких-либо технических ограничений здесь не возникает, однако есть некоторые административные вопросы (см. [RFC-1032]), решаемые на верхнем уровне организации, а зоны среднего уровня могут создавать свои правила владения. Например, один университет может выбрать использование единой зоны, а другой решит организовать субзоны для своих подразделений. В [RFC-1033] приведен каталог доступных программ DNS и рассмотрены административные процедуры.

После выбора подходящего имени для новой субзоны от ее новых владельцев следует потребовать демонстрации наличия резервного сервера имен. Отметим, что серверы имен для зоны не обязаны иметь имя в том же домене. Во многих случаях зона может быть более доступной, если ее серверы распределены по разным местам, а не размещаются в точках, физически контролируемых поддерживающей зону организацией. Например, в современной структуре DNS один из серверов для Великобритании (домен UK) расположен в США (US). Это позволяет американским хостам получать данные домена UK без загрузки трансатлантических каналов с ограниченной полосой.

В качестве заключительного шага в родительскую зону следует добавить делегирующие NS RR и склеивающие записи, которые обеспечат передачу полномочий для субзоны. Администраторам обеих зон следует обеспечить согласованность записей NS RR и склеивающих записей по обе стороны среза.

4.3. Внутреннее устройство сервера имен

4.3.1. Запросы и отклики

Основной работой серверов имен являются ответы на стандартные запросы. Как запросы, так и отклики на них передаются в форме сообщений стандартного формата, описанного в [RFC-1035]. Запросы содержат поля QTYPE, QCLASS и QNAME, которые описывают типы и классы желаемой информации, а также задают интересующее имя.

Способ ответа сервером на запрос зависит от поддержки сервером рекурсивного режима работы.

- Простейшим режимом работы с точки зрения сервера является нерекурсивный, поскольку в этом режиме сервер может давать ответы на запросы, используя только локальную информацию - отклик содержит ответ, сообщение об ошибке или ссылку на другой сервер, расположенный «ближе» к ответу. Все серверы имен должны поддерживать нерекурсивные запросы.

- Простейшим режимом с точки зрения клиента является рекурсивный, поскольку в этом режиме сервер имен выполняет функции распознавателя имен и всегда возвращает ответ или сообщение об ошибке, но не ссылку на другой сервер. Такой режим для серверов имен является опциональным и сервер имен может ограничивать круг клиентов, для которых обеспечивается рекурсивный режим обработки запросов.

Рекурсивный сервис полезен в нескольких ситуациях:

- относительно простая реализация запрашивающей стороны, не обеспечивающая возможность использования чего-либо, кроме прямого ответа на вопрос;

- запрос, который требуется передать другому протоколу или через иную границу, но можно отправить серверу, действующему в качестве посредника;

- сеть, где мы хотим концентрировать кэшированные отклики вместо создания отдельного кэша для каждого клиента.

Нерекурсивный сервис подходит в тех случаях, когда запрашивающая сторона способна обратиться по ссылке, а запрашиваемая информация может быть полезна для будущих запросов.

Использование рекурсивного режима ограничено ситуациями, когда клиент и сервер согласны использовать такой режим. Согласие обеспечивается путем использования двух битов в сообщениях запроса и отклика.

- Бит поддержки рекурсии (RA) устанавливается или сбрасывается сервером имен во всех откликах. Этот бит устанавливается, если сервер имен согласен предоставлять клиенту рекурсивное обслуживание, независимо от того, запрашивал ли данный клиент рекурсию. Т. е., бит RA говорит о поддержке рекурсии, а не о ее применении.

- Запросы включают бит RD (нужна рекурсия). Этот бит указывает потребность запрашивающей стороны в поддержке рекурсии для данного запроса. Клиенты могут запрашивать рекурсивное обслуживание у любого сервера имен, хотя им следует делать это лишь для серверов, которые ранее установили бит RA в своих сообщениях или согласились поддерживать рекурсию на основе приватной договоренности или с помощью иных мер, не включенных в протокол DNS.

Рекурсивный режим возникает в тех случаях, когда запрос с установленным флагом RD поступает на сервер, который желает обеспечивать рекурсивный сервис; клиент может убедиться в использовании рекурсивного режима, проверяя наличие в отклике обоих флагов RA и RD. Отметим, что серверам имен не следует выполнять рекурсию, если таковая не запрошена путем установки бита RD, поскольку нарушение этого правила может существенно осложнить поиск неполадок в серверах имен и их базах данных.

Если запрошен рекурсивный сервис и он поддерживается сервером, рекурсивный отклик может принимать одну из приведенных ниже форм:

- ответ на запрос с возможной добавкой впереди одной или нескольких записей CNAME RR, указывающих псевдонимы, которые встретились при подготовке отклика;
- сообщение об ошибке, указывающее на то, что запрошенного имени не существует; такое сообщение может включать записи CNAME RR, показывающие, что имя в запросе является псевдонимом для несуществующего имени;
- индикация временной ошибки.

Если рекурсивный сервис не был запрошен или не доступен, передается нерекурсивный отклик в одной из форм:

- ошибка authoritative name, указывающая на то, что имени не существует;
- индикация временной ошибки;
- та или иная комбинация:

  записей RR, отвечающих на запрос, вместе с индикацией получения данных из зоны или кэша;

  ссылок на серверы имен, имеющих зоны, которые ближе к прародителю имени, нежели данный сервер;
- записи RR которые по мнению сервера имен будут полезны для запрашивающего.

4.3.2. Алгоритм

Используемый сервером имен алгоритм будет зависеть от его операционной системы и структур данных для хранения RR. В описанном ниже алгоритме предполагается, что записи RR организованы в нескольких структурах типа «дерево» - по одному для каждой зоны и отдельные структуры для кэша:

1. Устанавливается или сбрасывается флаг поддержки рекурсии в зависимости от того, желает ли сервер поддерживать рекурсивные запросы. Если рекурсия возможна и запрошена с помощью бита RD в запросе, переход к п. 5, иначе — п. 2.
2. Поиск доступных зон для зоны, которая является ближайшим предком QNAME. При нахождении такой зоны переход к п. 3, иначе — п. 4.
3. Начало поиска соответствия (метка за меткой) в зоне. Процесс поиска может прерываться в нескольких случаях:

   a) При полном соответствии QNAME узел считается найденным.

   Если данные на узле совпадают с CNAME, а QTYPE не соответствует CNAME, запись CNAME RR копируется в раздел ответов отклика, QNAME меняется на каноническое имя в CNAME RR и выполняется возврат к п 1.

   В противном случае в раздел ответов копируются все RR, соответствующие QTYPE, и выполняется переход к п. 6.

   b) Если соответствие обнаруживается за пределами полномочных данных, это говорит о ссылке. Такие происходит в случаях, когда мы сталкиваемся с узлом, записи NS RR которого размещаются на верхнем срезе зоны.

   Записи NS RR для субзоны копируются в полномочный раздел (authority section) отклика. Все доступные адреса помещаются в дополнительный раздел с использованием склеивающих записей, если адреса нет в полномочных данных или кэше. Переход к п. 4.

   c) Если для какой-либо метки сопоставление невозможно (т. е., соответствующей метки не существует), проверяется существование метки «*».

   Если метки «*» не существует, проверяется является ли искомое имя исходным QNAME в запросе или именем, полученным через CNAME. Если имя является исходным, в отклике указывается ошибка authoritative name и поиск завершается. В противном случае поиск заканчивается без констатации ошибки.

   Если метка «*» существует, записи RR на данном узле сопоставляются с QTYPE. При обнаружении соответствий записи копируются в раздел ответа, но в качестве владельца RR указывается QNAME, а не узел с меткой «*». Переход к п. 6.
4. Начало поиска соответствий в кэше. При нахождении в кэше QNAME все записи RR, присоединенные к нему и совпадающие по QTYPE, копируются в раздел ответов. Если нет делегирования от полномочных данных, отыскиваются наиболее подходящие данные в кэеше и помещаются в раздел authority. Переход к п. 6.
5. Используется локальный распознаватель (local resolver) или копия его алгоритма (см. 5. Распознаватели) для ответа на запрос. Все результаты, включая промежуточные CNAME, сохраняются в разделе answer отклика.
6. С использованием только локальных данных предпринимается попытка добавления других RR, которые могут быть полезны в дополнительный раздел (additional section). Выход.

4.3.3. Шаблоны

В предыдущем алгоритме использовалась специальная трактовка записей RR, у которых имена владельца начинаются с метки «*». Такие RR называют шаблонами. Шаблонные RR можно представлять, как инструкцию по синтезированию RR. При выполнении соответствующих условий сервер имен создает записи RR, для которых имя владельца совпадает с именем в запросе, а содержимое берется из шаблонных RR.

Это свойство может зачастую применяться для создания зон, которые будут служить для пересылки почты в другие почтовые системы. Общая идея заключается в том, что любое имя в такой зоне, представленной серверу в запросе, будет предполагаться существующим и имеющим некие свойства, пока не очевидно обратное. Отметим, что использование термина «зона» вместо термина «домен» в данном случае является преднамеренным — это

предотвращает выход используемых по умолчанию значений через границы зоны, хотя субзона может выбрать такое же поведение путем установки аналогичных значений по умолчанию.

Содержимое шаблонных RR следует обычным правилам и форматам для RR. Шаблоны в зоне имеют имя владельца, которое определяет соответствие именам в запросах. Имя владельца шаблонной RR имеет форму *.<anydomain>, где <anydomain> может представлять собой любое доменное имя. В <anydomain> не следует включать другие метки «*», и это имя должно входить в полномочные данные зоны. Шаблоны могут применяться к наследникам <anydomain>, но не к самому домену <anydomain>. Иными словами, метка «*» всегда соответствует по крайней мере одной полной метке и может соответствовать множеству меток (всегда полных).

Шаблоны RR не применимы в тех случаях, когда:

- запрос относится к другой зоне (т. е., делегирование отменяет шаблоные допущения);
- имя в запросе или имя между шаблонным доменом и именем в запросе определенно не существует; например, если шаблонная запись RR имеет владельца с именем *.X и зона содержит также записи RR, привязанные к B.X, шаблон будет применяться к запросам для имени Z.X (предполагается отсутствие явных данных для for Z.X), но не к B.X, A.B.X или X.

Метка «*» в запросе не вызывает каких-либо специальных эффектов, но может применяться для тестирования шаблонов в уполномоченной зоне; такие запросы являются единственным способом получить отклик, содержащий RR с именем владельца «*». Результаты таких запросов не следует кэшировать.

Отметим, что содержимое шаблонных RR не изменяется при из использовании для синтезирования записей RR.

В качестве иллюстрации применения шаблонных RR представим крупную компанию с большой сетью не-IP/TCP, для которой нужно организовать почтовый шлюз. Если компания называется X.COM, а поддерживающий IP/TCP шлюз называется A.X.COM, в зону COM можно добавить следующие RR:

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

Это приведет к тому, что на любой запрос MX, завершающийся в X.COM, будет возвращаться запись MX RR, указывающая на A.X.COM. Две шаблонных RR нужны для того, чтобы влияние шаблона *.X.COM подавлялось в субдереве A.X.COM явным указанием для A.X.COM. Отметим, что явные данные MX для X.COM и A.X.COM являются обязательными и ни одна из расположенных выше RR не будет соответствовать запросу имени XX.COM.

4.3.4. Кэширование негативных откликов (необязательно)

DNS обеспечивает дополнительный сервис, позволяющий серверам имен распространять, а распознавателям (resolver) кэшировать негативные результаты с TTL (время жизни). Например, сервер может распространять TTL вместе с индикацией ошибки имени (name error), а получивший такую информацию распознаватель может считать, что имя в действительности не существует, пока на завершится срок TTL, не запрашивая полномочных данных. Аналогично, распознаватель может сделать запрос с QTYPE, которому соответствует множество типов, и кэшировать факт отсутствия некоторых из этих типов.

Эта функция может оказаться весьма важной для систем, в которых реализовано сокращение имен с использованием списков поиска, поскольку для популярных сокращений могут возникать многочисленные ошибки имен.

Метод заключается в том, что сервер имен может добавить запись SOA RR в дополнительный раздел отклика когда данный отклик является полномочным. Запись SOA должна быть от той зоны, которая была источником полномочных данных в разделе ответов (answer section) или информации об ошибке имени. Поле MINIMUM в SOA определяет продолжительность периода, на который негативный результат может кэшироваться.

Отметим, что при некоторых обстоятельствах раздел ответа может содержать множество имен владельцев. В таких случаях механизм SOA следует использовать только для данных, которые соответствуют QNAME, поскольку полномочными в разделе будут только эти данные.

Серверам имен и распознавателям ни в коем случае не следует пытаться добавлять записи SOA в дополнительный раздел или предполагать результаты, которые не получены напрямую из полномочного отклика. Это обусловлено несколькими причинами, включая: недостаточность кэшированной информации для определения соответствия RR и имен зон, возможность кэширования SOA RR в результате прямых запросов SOA (серверы имен не обязаны включать SOA в раздел authority).

Эта функция является необязательной, хотя предполагается, что расширенная версия в будущем станет частью стандартного протокола. Серверы имен не обязаны добавлять SOA RR во все полномочные отклики, а распознаватели не обязаны кэшировать негативные результаты. Однако рекомендуется делать это. Все распознаватели и рекурсивные серверы имен должны быть способны по крайней мере игнорировать записи SOA RR, если таковые присутствуют в отклике.

Предложены некоторые эксперименты, использующие данную функцию. Идея заключается в том, что при наличии в кэше данных, которые происходят из конкретной зоны и получении полномочной копии SOA для этой зоны, в которой значение поля SERIAL не изменилось по сравнению с кэшированными данными, можно сбросить для кэшированных данных значение TTL до значения поля MINIMUM, если оно меньше. Такое подход рассматривается пока лишь как экспериментальный и в настоящее время еще не рекомендуется для всеобщего использования.

4.3.5. Обслуживание и перенос зон

Часть работы администратора зоны заключается в поддержке зон на всех серверах имен, которые являются полномочными для зоны. Когда требуемые изменения внесены, их нужно распространить по всем серверам имен. Хотя такое распространение возможно на основе протокола FTP или иных специальных процедур, предпочтительней использовать связанную с переносом зон часть протокола DNS.

Общая модель автоматического переноса или обновления зон строится на основе того, что один из серверов имен служит для зоны в качестве ведущего (master) или первичного (primary) для зоны. Связанные с зоной изменения обычно вносятся путем редактирования основного файла для зоны. После редактирования администратор дает серверу команду на загрузку измененной зоны. Другие серверы, которые служат для зоны вторичными (secondary) или ведомыми (non-master) периодически (с заданным интервалом) проверяют наличие изменений и при их обнаружении загружают новую копию зоны.

Для детектирования изменений вторичные серверы просто проверяют значение поля SERIAL в записи SOA для зоны. В дополнение к любым вносимым в зону изменениям значение поля SERIAL в SOA этой зоны следует каждый раз увеличивать. Увеличение может выполняться в форме простого добавления фиксированного значения, но обычно в качестве порядкового номера используют значение текущей даты[1], дополняемое порядковым номером изменений в течение текущих суток. Такое изменение порядкового номера позволяет убедиться, какая из двух копий зоны является более новой путем простого сравнения порядковых номеров. При увеличении и сравнении номеров используется арифметика порядковых номеров, поэтому число цифр порядкового номера определяет число возможных изменений файла зоны без достижения максимума. Обычно предыдущие копии зоны должны «умереть» до того, как порядковый номер достигнет половины своего максимального 32-битового значения. На практике проблемы могут возникать только при расположении номеров около максимально и минимально возможных значений диапазона.

Периодичность опроса со стороны вторичных серверов управляется параметрами записи SOA RR для зоны, которые устанавливают минимальное допустимое значение интервала опроса. Эти параметры называются REFRESH, RETRY и EXPIRE. Когда на вторичный сервер загружается новая зона, этот сервер ждет в течение REFRESH секунд, прежде чем проверять порядковый номер этой зоны и далее повторяет эти попытки через каждые RETRY секунд. Проверка заключается в простом запросе у первичного сервера записи SOA RR для зоны. Если поле порядкового номера зоны на вторичном сервере не меньше порядкового номера на первичном, это говорит об отсутствии изменений и таймер REFRESH запускается заново. Если в течение интервала EXPIRE вторичный сервер не смог проверить порядковый номер зоны, ему следует предположить свою копию зоны устаревшей и отбросить ее.

Когда опрос показывает изменение зоны, вторичный сервер должен запросить перенос зоны с помощью запроса AXFR для нее. Запрос AXFR может породить ошибку (например, отказ), но обычно в ответ на него приходит последовательность откликов. Первое и последнее сообщения этой цепочки должны содержать данные для верхнего полномочного узла зоны. В промежуточных сообщениях содержаться все остальные RR из зоны, включая полномочные и неполномочные RR. Этот поток сообщений позволяет вторичному серверу реконструировать копию зоны. Поскольку точность в этом случае важна, для запросов AXFR должен использоваться протокол TCP или иной с гарантией доставки.

Каждый вторичный сервер должен выполнять эти операции по отношению к первичным серверам, но может выполнять их и по отношению к другим вторичным серверам. Такая стратегия может повысить эффективность переноса зон в тех случаях, когда первичный сервер оказывается недоступным в результате проблем на хосте или в сети, а также в тех случаях, когда сервер имеет более эффективный доступ к «промежуточному» вторичному серверу, чем к основному.

5. Распознаватели

5.1. Введение

Распознаватель (Resolver) представляет собой программу, обеспечивающую интерфейс между пользовательскими приложениями и серверами доменных имен. В простейшем случае распознаватель получает запрос от пользовательской программы (например, почтового клиента, TELNET, FTP) в форме вызова подпрограммы, системного вызова и т. п., после чего возвращает запрошенную информацию в форме, совместимой с форматами данных локального хоста.

Распознаватель размещается на одной машине с запрашивающей его услуги программой, но ему могут потребоваться услуги размещающихся на других хостах серверов имен. Поскольку распознаватель может запрашивать данные у нескольких серверов имен или искать их в локальном кэше, время выполнения запросов может меняться в широких пределах, от миллисекунд до нескольких секунд.

Важной функцией распознавателей является снижение задержек в сети и нагрузки на серверы имен за счет кэширования информации, полученной при предшествующих запросах. Следовательно, кэш общего пользования, который доступен множеству процессов, пользователей, машин и т. п., обеспечивает более высокую эффективность, нежели индивидуальное кэширование.

5.2. Интерфейс между распознавателем и клиентом

5.2.1. Типовые функции

На клиентский интерфейс распознавателей влияет среда локального хоста, но типичный интерфейс распознавателя с пользовательскими программами включает три функции:

1. **Преобразование имен хостов в их адреса**.

   Эта функция служит заменой прежним функциям, основанным на работе с файлом HOSTS.TXT. Вызывающая функцию программа ждет получения одного или множества 32-битовых адресов IP по символьной строке с именем хоста. В рамках DNS текстовая трока преобразуется в запрос для записей типа A RR. Поскольку DNS не сохраняет порядок записей RR, эта функция по своему усмотрению может выполнять сортировку найденных адресов или выбирать «лучший» адрес, если клиенту нужен только один адрес. Отметим, что рекомендуется возвращать все множество полученных адресов, но для эмуляции предшествующих служб HOSTS.TXT может потребоваться возврат единственного адреса.

2. **Преобразование адресов хостов в их имена**.

   Эта функция по форме похожа на предшествующую и возвращает символьную строку имени хоста по 32-битовому адресу IP. Порядок октетов адреса IP меняется на обратный и выражается в форме символьной

[1] В формате ГГГГЧЧММ

строки (компонента имени), к которой добавляется суффикс IN-ADDR.ARPA. Для получения искомых записей делается запрос типа PTR для получения основного (primary) имени хоста. Например, запрос имени хоста с IP-адресом 1.2.3.4 будет выполняться, как поиск записей PTR RR для доменного имени 4.3.2.1.IN-ADDR.ARPA.

3. **Общие функции поиска**.

   Эти функции служат для получения от DNS другой информации, которая не поддерживалась в предшествующих службах. Вызывающая сторона передает значения QNAME, QTYPE, QCLASS и хочет получить в ответ все соответствующие им записи RR. Эта функция зачастую использует формат DNS для всех данных RR и возвращает все содержимое RR (например, TTL) вместо обработки в соответствии с локальными соглашениями данного хоста.

В результате выполнения перечисленных выше функций распознаватель обычно возвращает клиенту один из перечисленных ниже вариантов:

- Одна или множество записей RR с запрошенными данными.

  В этом случае распознаватель возвращает ответ в соответствующем формате.

- Сообщение «Ошибка в имени» (NE).

  Это сообщение возвращается при запросе данных для несуществующего имени (например, в результате опечатки пользователя).

- Сообщение об ошибке, связанной с тем, что данные не удалось найти.

  Такое сообщение возвращается в тех случаях, когда запрошенное имя существует, но данные соответствующего типа отсутствуют. Например, функция поиска адреса, примененная к символьному адресу электронной почты будет приводить к такому сообщению, поскольку имя существует но адресной записи RR для него нет.

Важно отметить, что функции преобразования имен хостов в адреса и обратно могут объединять сообщения «ошибка в имени» и «данные не найдены» в одно сообщение об ошибке, но общие функции поиска не должны этого делать. Одна из причин этого заключается в том, что приложения могут запросить сначала один тип информации от имени, после чего будет следовать запрос другого типа информации для того же имени. Если сообщения об ошибках смешать, это может привести к замедлению хоста из-за выполнения бесполезных запросов.

5.2.2. Псевдонимы

При попытке выполнить преобразование для конкретного запроса распознаватель может обнаружить, что запрашиваемое имя является псевдонимом. Например, распознаватель может определить, что имя хоста, указанное для преобразования в адрес, является псевдонимом, найдя соответствующую запись CNAME RR. По возможности информация об этом должна передаваться от распознавателя клиенту.

При обнаружении записи CNAME распознаватель обычно повторяет запрос для нового имени. Однако при выполнении функций общего назначения преобразователю не следует повторять запрос, если запись CNAME RR соответствует типу запроса. Это делает возможными запросы, позволяющие узнать о наличии псевдонимов имен. Например, если для запроса указан тип CNAME, пользователя интересует собственно CNAME RR, а не другие записи RR, на которые указывает имя.

При работе с псевдонимами возникает несколько особых случаев. Следует избегать многоуровневых псевдонимов в целях повышения производительности. Циклы из псевдонимов и псевдонимы, которым не соответствуют реальные имена, могут приводить к возникновению ошибок, возвращаемых клиенту.

5.2.3. Временные отказы

В реальном мире у каждого распознавателя время от времени могут возникать проблемы при выполнении того или иного запроса. Проблемы могут быть связаны с изоляцией распознавателя от остальной сети в результате неполадок в сети или на шлюзах, а также (менее вероятно) с недоступностью серверов для конкретного домена.

Важно в таких случаях не давать приложениям сообщения о постоянной ошибке, связанной с запрошенным именем или данными. Такие сообщения раздражают пользователей и могут вызывать ущерб для почтовых систем, использующих DNS.

Хотя в некоторых случаях возникновения временных проблем можно просто блокировать запросы на неопределенное время, это не является хорошим решением особенно в тех случаях, когда клиентом распознавателя является серверный процесс, который распространит эту проблему на другие задачи. Рекомендуемым решением является сигнализация о временных ошибках, хотя это может осложнить эмуляцию функций HOSTS.TXT.

5.3. Устройство распознавателей

В каждой реализации распознавателей могут применяться слегка различающиеся алгоритмы и, как правило, значительно больше усилий затрачивается на обработку различных типов ошибок, нежели на выполнение типовых функций. В этом разделе описана рекомендуемая стратегия работы распознавателей, а детальное описание дано в [RFC-1035].

5.3.1. Оконечные распознаватели

Одним из вариантов реализации распознавателя является перенос функций преобразования с локальной машины на сервер имен, который поддерживает обработку рекурсивных запросов. Это позволяет обеспечить простой способ поддержки служб DNS на ПК с недостаточными для выполнения функций преобразования ресурсами, а также обеспечить централизованное кэширование для локальной сети или организации.

Все что нужно от такого оконечного распознавателя, - это поддержка списка серверов имен, которые отвечают на рекурсивные запросы. Такой тип распознавателей возможно будет нуждаться в информации, хранящейся в

конфигурационном файле, поскольку у него не достает возможности поиска ее в базе данных о доменах. Пользователь также должен убедиться, что перечисленные в файле серверы имен поддерживают рекурсию, поскольку серверы могут отвергать такие запросы от некоторых или всех клиентов. Пользователю следует проконсультироваться с локальным администратором по вопросу выбора нужных серверов имен.

Этот тип сервиса имеет ряд недостатков. Поскольку выполнение рекурсивных запросов может занимать произвольное время, у оконечного распознавателя могут возникать сложности с выбором интервала повтора в случаях потери пакетов UDP или «мертвых» серверов имен; кроме того, сервер имен легко можно перегрузить запросами таких оконечных распознавателей, если сервер будет интерпретировать повторы, как новые запросы. Решением проблемы может служить использование протокола TCP, но это может привести к издержкам, сравнимым с организацией полнофункционального распознавателя.

5.3.2. Ресурсы

В дополнение к собственным ресурсам распознаватель может также использовать разделяемый доступ к зонам, поддерживаемым локальным сервером имен. Это позволяет преобразователю быстрее отвечать на запросы, но ему следует соблюдать осторожность и не позволять перезаписывать данные зоны с использованием кэшированной информации. Здесь термин «локальная информация» трактуется, как объединение кэша и разделяемых зон с пониманием того, что полномочные данные в любом случае предпочтительней кэшированных при наличии обоих.

Рассмотренный ниже алгоритм работы распознавателя базируется на допущении о том, что все функции преобразуются в общую функцию поиска, и использует для представления состояний обрабатываемого запроса следующие структуры данных:

SNAME доменное имя, для которого выполняется поиск.

STYPE значение QTYPE для поискового запроса.

SCLASS значение QCLASS для поискового запроса.

SLIST структура, описывающая серверы имен и зону, для которой распознаватель пытается выполнить запрос. В этой структуре сохраняется лучшие допущения распознавателя о том, какие серверы имеют желаемую информацию; структура обновляется по мере поступления новых данных, меняющих допущения. Эта структура включает эквивалент имени зоны, известные серверы имен для зоны и данные предыстории, которые могут использоваться для наиболее подходящего сервера при следующей попытке. Эквивалент имени зоны включает счетчик соответствия с числом меток от корня зоны вниз, для которых SNAME является общим с запрашиваемой зоной; это значение служит в качестве меры «близости» к SNAME.

SBELT структура safety belt (ремень безопасности) имеет такую же форму, как SLIST, которая инициализируется из конфигурационного файла и содержит список серверов, которые преобразователю следует использовать при отсутствии локальных данных по выбору серверов имен. Значение счетчика соответствий -1 указывает на отсутствие соответствующих меток.

CACHE структура, в которой сохраняются результаты из предыдущих откликов. Поскольку распознаватель отвечает за отбрасывание записей RR с истекшим временем TTL, большинство реализаций преобразует интервал, заданный в приходящих RR, в некое «абсолютное» время, в течение которого RR будет храниться в кэше. Вместо независимого уменьшения значений TTL распознаватель в этом случае может просто игнорировать или отбрасывать устаревшие RR в процессе поиска или отбрасывать их при периодическом просмотре для освобождения памяти от старых RR.

5.3.3. Алгоритм

Алгоритм верхнего уровня состоит из 4 шагов:

1. Проверка наличия локальной информации и, при обнаружении, возврат ее клиенту.
2. Определение подходящего сервера для запроса.
3. Передача запросов выбранному серверу до получения от него отклика.
4. Анализ отклика:
   a. если отклик отвечает на вопрос или указывает на ошибку в имени, данные кэшируются и возвращаются клиенту;
   b. если отклик указывает на делегирование другому серверу, кэеширование информации о делегировании и переход к п. 2.
   c. если отклик показывает запись CNAME, которая сама по себе не является ответом на вопрос, запись CNAME кэшируется, значение SNAME меняется на каноническое имя из CNAME RR и выполняется переход к п. 1.
   d. если отклик сообщает об отказе сервера или имеет странное содержимое, сервер удаляется из списка SLIST и выполняется переход к п. 3.

На этапе 1 выполняется поиск интересующих данных в кэше. Если эти данные найдены там, они предполагаются подходящими для использования. В некоторых распознавателях на уровне пользовательского интерфейса поддерживается опция, вынуждающая распознаватель игнорировать кэшированные данные и запрашивать полномочный сервер. Не рекомендуется включать эту опцию по умолчанию. Если распознаватель имеет прямой доступ к зонам сервера имен, ему следует проверить присутствие полномочных данных и, при их наличии, использовать полномочные данные вместо кэшированных.

На этапе 2 отыскивается сервер имен для запроса у него требуемых данных. Общая стратегия поиска сервера состоит в просмотре доступных локально RR серверов имен, начиная с SNAME, затем родительского домена SNAME, родителя родителя и так далее до корневого сервера. Таким образом, если SNAME указывает на Mockapetris.ISI.EDU, на этом этапе будут отыскиваться записи NS RR сначала для Mockapetris.ISI.EDU, затем ISI.EDU, EDU и в заключении - .

(корень). Эти записи NS RR перечисляют список хостов для зоны SNAME или выше. Имена копируются в SLIST. С использованием локальных данных определяются адреса хостов. Возможны ситуации, когда эти адреса не будут доступны. Распознаватель в таких случаях может использовать несколько вариантов, лучшим из которых является запуск параллельного процесса преобразования с запросом на определение адреса при одновременном продолжении работы с доступными адресами. Очевидно, что вопрос выбора и реализации опций является достаточно сложным и зависит от возможностей хоста. Ниже приведены рекомендации для разработчиков программ-распознавателей:

1. Ограничить объем работы (число переданных пакетов и запущенных процессов) так, чтобы запрос не мог породить бесконечного цикла или начать цепную реакцию запросов, даже если **какая-то из взаимодействующих реализаций имеет некорректную настройку**.
2. Возвращать отклик по возможности.
3. Избегать ненужных передач.
4. Давать ответ с максимально возможной скоростью.

Если поиск NS RR завершился отказом, распознаватель инициализирует SLIST, используя «ремень безопасности» SBELT. Идея заключается в том, что при отсутствии у распознавателя соображений относительно выбора серверов для запроса следует использовать информацию из конфигурационного файла, в котором перечислены несколько серверов, способных помочь. Хотя это не исчерпывает всех случаев, целесообразно указать в файле пару корневых серверов и пару серверов домена, к которому относится хост. Два сервера каждого типа обеспечивают резервирование. Пара корневых серверов поможет обслужить в конечном итоге любые запросы по всему пространству доменных имен, а пара локальных серверов позволит преобразователю работать с локальными сетевыми ресурсами даже в случае отказа шлюза или внешнего канала.

Структура данных SLIST с именами и адресами серверов может быть отсортирована в порядке предпочтений для обеспечения работы с более подходящим сервером или поочередного использования всех серверов из списка. Сортировка может быть основана на предпочтении локальных адресов или на статистике прошлых событий (таких, как время отклика и др.).

На этапе 3 передаются запросы до тех пор, пока не будет получен отклик. Стратегия заключается в циклическом переборе адресов всех серверов с использованием интервала между повторами передачи. На практике важно использовать все адреса многодомного хоста, а слишком агрессивная политика повтора будет реально замедлять получения отклика при использовании множеством распознавателей (а иногда и одним) одного и того же сервера имен. Структура данных SLIST обычно включает данные для выбора интервалов между повторами и сохранения данных о предшествующих передачах запросов.

Этап 4 включает анализ откликов. Преобразователю следует использовать параноидальный подход при разборе полученных откликов, а также следует проверять соответствие отклика запросу с использованием поля ID в отклике.

Идеальным случаем является ответ полномочного сервера, который возвращает требуемые данные или сообщает об ошибке имени. Данные передаются пользователю и помещаются в кэш для последующего использования, если значение TTL больше нуля.

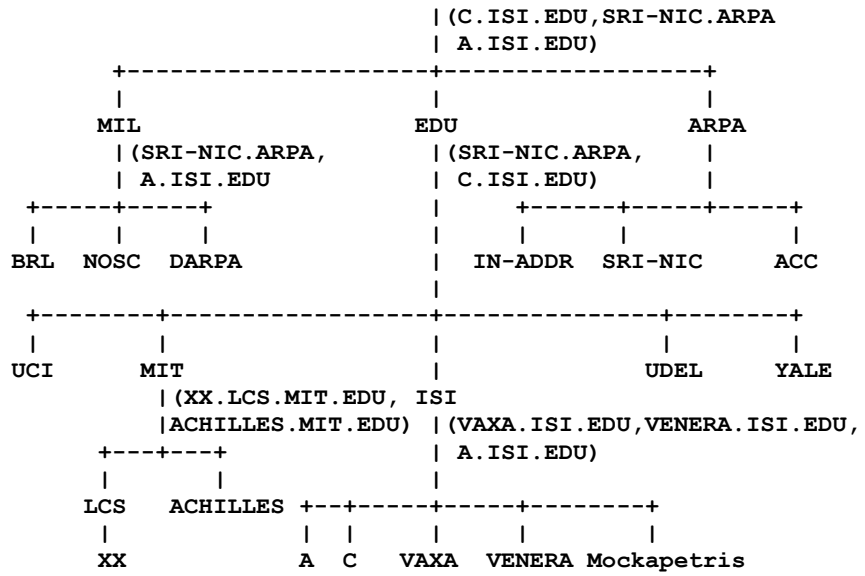
Если отклик показывает делегирование, преобразователю следует проверить, что это делегирование «ближе» к ответу, чем серверы из SLIST. Это можно сделать путем сравнения счетчика соответствий в SLIST со значением, рассчитанным из SNAME и записей NS RR для делегирования. Если делегирование не обеспечивает «приближения», следует считать его фальшивым и отбросить. Если делегирование корректно, записи NS и все адресные RR для серверов следует кэшировать. Серверы имен включаются в структуру SLIST и поиск повторяется.

Если отклик включает CNAME, поиск повторяется с использованием значения из CNAME, пока не будет получено каноническое имя, если само значение CNAME не является ответом.

Более подробные рекомендации для разработчиков приведены в [RFC-1035].

6. Сценарий

В нашем образце пространства доменных имен предположим, что нужно организовать отдельный административный контроля для корневой зоны и доменов MIL, EDU, MIT.EDU and ISI.EDU. Мы можем организовать иерархию доменов, как показано на рисунке.



В этом примере полномочные серверы имен показаны в круглых скобках на тех уровнях дерева доменов, где предполагается контроль.

Таким образом, корневыми серверами будут C.ISI.EDU, SRI-NIC.ARPA и A.ISI.EDU. Домен MIL обслуживается серверами SRI-NIC.ARPA и A.ISI.EDU, а домен EDU — серверами SRI-NIC.ARPA и C.ISI.EDU. Отметим, что серверы могут иметь зоны, которые являются непрерывными или фрагментарными (disjoint). В этом примере C.ISI.EDU имеет непрерывные зоны в корневом домене и EDU, сервер A.ISI.EDU имеет непрерывные зоны в корневом домене и MIL, а также фрагментарную зону в ISI.EDU.

6.1. Сервер имен C.ISI.EDU

C.ISI.EDU является сервером имен для корневого домена, а также доменов MIL и EDU в классе IN, поэтому будет включать зоны этих 3 доменов. Зона для корневого домена может иметь вид:

```
.       IN      SOA     SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                        870611          ;порядковый номер
                        1800            ;обновление каждые 30 минут
                        300             ;повтор каждые 5 минут
                        604800          ;срок действия истекает через неделю
                        86400)          ;минимум 1 день
                NS      A.ISI.EDU.
                NS      C.ISI.EDU.
                NS      SRI-NIC.ARPA.

MIL.    86400   NS      SRI-NIC.ARPA.
        86400   NS      A.ISI.EDU.

EDU.    86400   NS      SRI-NIC.ARPA.
        86400   NS      C.ISI.EDU.

SRI-NIC.ARPA.   A       26.0.0.73
                A       10.0.0.51
                MX      0 SRI-NIC.ARPA.
                HINFO   DEC-2060 TOPS20

ACC.ARPA.       A       26.6.0.65
                HINFO   PDP-11/70 UNIX
                MX      10 ACC.ARPA.

USC-ISIC.ARPA.  CNAME   C.ISI.EDU.

73.0.0.26.IN-ADDR.ARPA.     PTR     SRI-NIC.ARPA.
65.0.6.26.IN-ADDR.ARPA.     PTR     ACC.ARPA.
51.0.0.10.IN-ADDR.ARPA.     PTR     SRI-NIC.ARPA.
52.0.0.10.IN-ADDR.ARPA.     PTR     C.ISI.EDU.
103.0.3.26.IN-ADDR.ARPA.    PTR     A.ISI.EDU.

A.ISI.EDU. 86400        A       26.3.0.103
C.ISI.EDU. 86400        A       10.0.0.52
```

Данные здесь представлены в том виде, как они задаются в основном (master) файле. Большая часть RR имет вид одиночной строки, единственным исключением является запись SOA RR, в которой используются круглые «(« для обозначения начала многострочной RR и «)» - для завершения. Поскольку все RR в зоне должны относиться к одному классу, этот класс требуется указывать только в первой RR зоны. Когда сервер имен загружает зону, он устанавливает значение TTL для всех полномочных RR не менее значения поля MINIMUM в записи SOA (86400 секунд соответствует суткам). Записи NS RR обозначают делегирование для доменов MIL и EDU и вместе со склеивающими RR для адресов серверов не являются полномочными данными зоны и, следовательно, имеют явно заданные значения TTL.

К корневому узлу присоединены 4 записи RR: SOA с описанием корневой зоны и 3 NS RR со списком серверов имен для корневой зоны. Данные в SOA RR описывают управление зоной. Данные зоны поддерживаются на хосте SRI-NIC.ARPA и отвечает за зону лицо с адресом HOSTMASTER@SRI-NIC.ARPA. Ключевым элементом SOA является минимальное значение TTL 86400 секунд, которое показывает, что все полномочные данные зоны имеют TTL не меньше этого значения, хотя явно могут быть указаны и большие времена жизни.

Записи NS RR для доменов MIL и EDU указывают границу между корневой зоной и зонами MIL и EDU. Отметим, что в примере нижележащие зоны поддерживаются теми же серверами имен, что и корневая зона.

Основной файл зоны EDU может указываться относительно «корня» EDU. Файл данных домена EDU может иметь вид:

```
EDU.  IN SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                        870729      ;порядковый номер
                        1800            ;обновление каждые 30 минут
                        300             ;повтор каждые 5 минут
                        604800          ;срок действия истекает через неделю
                        86400)          ;минимум 1 день
                    NS      SRI-NIC.ARPA.
                    NS      C.ISI.EDU.

UCI         172800              NS      ICS.UCI
            172800              NS      ROME.UCI
ICS.UCI             172800      A       192.5.19.1
```

```
    ROME.UCI        172800       A      192.5.19.31
    ISI       172800             NS     VAXA.ISI
              172800             NS     A.ISI
       172800              NS     VENERA.ISI.EDU.
    VAXA.ISI  172800       A      10.2.0.27
              172800       A      128.9.0.33
    VENERA.ISI.EDU. 172800       A 10.1.0.52
                         172800       A 128.9.0.32
    A.ISI     172800       A      26.3.0.103

    UDEL.EDU.172800       NS     LOUIE.UDEL.EDU.
                    172800       NS     UMN-REI-UC.ARPA.
    LOUIE.UDEL.EDU. 172800 A      10.0.0.96
                         172800       A      192.5.39.3

    YALE.EDU.172800       NS     YALE.ARPA.
    YALE.EDU.       172800       NS     YALE-BULLDOG.ARPA.

    MIT.EDU.  43200  NS     XX.LCS.MIT.EDU.
                     43200  NS     ACHILLES.MIT.EDU.
    XX.LCS.MIT.EDU. 43200  A      10.0.0.44
    ACHILLES.MIT.EDU.      43200  A      18.72.0.8
```

Отметим здесь использование относительных имен. Имя владельца ISI.EDU указано в относительном формате, как и имена из серверов имен. Допускается произвольное смешение относительной и абсолютной формы имен.

6.2. Примеры стандартных запросов

Приведенные ниже примеры запросов и откликов иллюстрируют поведение сервера имен. Если явно не указано иное, запросы не указывают не желательность рекурсии (RD). Отметим, что отклики на нерекурсивные запросы зависят от запрашиваемого сервера, но не зависят от идентификации запрашивающего.

6.2.1. QNAME=SRI-NIC.ARPA, QTYPE=A

Запрос имеет вид:

```
            +---------------------------------------------------+
Заголовок   | OPCODE=SQUERY                                     |
            +---------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A           |
            +---------------------------------------------------+
Ответ       | <пусто>                                           |
            +---------------------------------------------------+
Полномочия  | <пусто>                                           |
            +---------------------------------------------------+
Дополнения  | <пусто>                                           |
            +---------------------------------------------------+
```

Отклик от C.ISI.EDU будет иметь вид:

```
            +---------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                       |
            +---------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A           |
            +---------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 86400 IN A 26.0.0.73                |
            |               86400 IN A 10.0.0.51                |
            +---------------------------------------------------+
Полномочия  | <пусто>                                           |
            +---------------------------------------------------+
Дополнения  | <пусто>                                           |
            +---------------------------------------------------+
```

Заголовки отклика и запроса похожи, за исключением установки бита RESPONSE, показывающего, что сообщение является откликом, а не запросом, и бита полномочного ответа (AA[1]), указывающего, что записи RR в разделе ответов получены из полномочного источника. Раздел вопроса в отклике соответствует одноименному разделу запроса.

Если тот или иной запрос бел передан некому серверу, который не был полномочным для SRI-NIC.ARPA, отклик может иметь вид:

```
            +---------------------------------------------------+
Заголовок   | OPCODE=SQUERY,RESPONSE                            |
            +---------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A           |
            +---------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 1777 IN A 10.0.0.51                 |
            |               1777 IN A 26.0.0.73                 |
            +---------------------------------------------------+
Полномочия  | <пусто>                                           |
            +---------------------------------------------------+
Дополнения  | <пусто>                                           |
```

[1]Authoritative Answer.

```
            +-------------------------------------------------+
```

Этот отклик отличается от предыдущего отсутствием бита AA в заголовке и значениями TTL. Это показывает, что данные были взяты из кэша, а не из зоны. Разница между полномочным TTL и TTL из примера заключается в том, что запись из кэша уже несколько устарела. Разница в порядке записей RR в разделе ответов не столь существенна.

*6.2.2. QNAME=SRI-NIC.ARPA, QTYPE=**

Запрос очень похож на предыдущий, но значение * в поле QTYPE приведет к получению от C.ISI.EDU отклика вида:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
            +-------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 86400 IN  A     26.0.0.73         |
            |                         A     10.0.0.51         |
            |                         MX    0 SRI-NIC.ARPA.   |
            |                         HINFO DEC-2060 TOPS20   |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Если подобный запрос был направлен двум серверам имен, которые не полномочны для SRI-NIC.ARPA, отклики могут иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE                         |
            +-------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
            +-------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 12345 IN     A       26.0.0.73    |
            |                            A       10.0.0.51    |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

и

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE                         |
            +-------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
            +-------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 1290 IN HINFO  DEC-2060 TOPS20    |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Ни в одном из этих откликов не установлен бит AA, поскольку они не содержат полномочных данных. Различие в содержимом и разные TTL говорят о том, что серверы кэшировали данные в разное время и первый сервер кэшировал отклик на запрос QTYPE=A, а второй — отклик на запрос HINFO.

6.2.3. QNAME=SRI-NIC.ARPA, QTYPE=MX

Такой тип запроса может появиться в результате попытки почтовой программы найти маршрутную информацию для доставки сообщения по адресу HOSTMASTER@SRI-NIC.ARPA.

Отклик C.ISI.EDU будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=MX        |
            +-------------------------------------------------+
Ответ       | SRI-NIC.ARPA. 86400 IN     MX      0 SRI-NIC.ARPA.|
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | SRI-NIC.ARPA. 86400 IN     A       26.0.0.73    |
            |                            A       10.0.0.51    |
            +-------------------------------------------------+
```

Этот отклик показывает запись MX RR в разделе ответов. Дополнительный раздел включает адресные RR, поскольку сервер имен C.ISI.EDU предполагает, что запрашивающему потребуются эти адреса для использования информации из записи MX.

6.2.4. QNAME=SRI-NIC.ARPA, QTYPE=NS

C.ISI.EDU будет отвечать на такой запрос откликом вида:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=NS        |
            +-------------------------------------------------+
Ответ       | <пусто>                                         |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Единственное различие между запросом и откликом заключается в том, что в заголовке отклика установлены биты AA и RESPONSE. Это означает, что отклик получен от полномочного сервера и запрошенное имя существует, но записи типа NS на сервере нет.

6.2.5. QNAME=SIR-NIC.ARPA, QTYPE=A

Такой запрос может возникать при ошибке пользователя в имени хоста. Ответ сервера C.ISI.EDU будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA, RCODE=NE           |
            +-------------------------------------------------+
Вопрос      | QNAME=SIR-NIC.ARPA., QCLASS=IN, QTYPE=A         |
            +-------------------------------------------------+
Ответ       | <пусто>                                         |
            +-------------------------------------------------+
Полномочия  | . SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA.    |
            |       870611 1800 300 604800 86400              |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Этот отклик говорит, что имени не существует. Это показано кодом отклика (RCODE) в заголовке.

Запись SOA RR в разделе полномочий содержит негативную информацию (она может кэшироваться). Позволяющую преобразователю использовать этот отклик для допущения об отсутствии имени в течение SOA MINIMUM (86400) секунд.

6.2.6. QNAME=BRL.MIL, QTYPE=A

Если такой запрос передать серверу C.ISI.EDU, отклик будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE                         |
            +-------------------------------------------------+
Вопрос      | QNAME=BRL.MIL, QCLASS=IN, QTYPE=A               |
            +-------------------------------------------------+
Ответ       | <пусто>                                         |
            +-------------------------------------------------+
Полномочия  | MIL.            86400 IN NS     SRI-NIC.ARPA.   |
            |                 86400    NS     A.ISI.EDU.      |
            +-------------------------------------------------+
Дополнения  | A.ISI.EDU.              A       26.3.0.103      |
            | SRI-NIC.ARPA.           A       26.0.0.73       |
            |                         A       10.0.0.51       |
            +-------------------------------------------------+
```

Раздел ответов отклика не содержит данных и отклик не является полномочным, поскольку он возвращает ссылку. Сервер имен C.ISI.EDU указывает, что он не является полномочным для домена MIL и указывает запрашивающему на серверы A.ISI.EDU и SRI-NIC.ARPA, которые имеют полномочные данные для домена MIL.

6.2.7. QNAME=USC-ISIC.ARPA, QTYPE=A

Отклик на такой запрос от сервера A.ISI.EDU будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A        |
            +-------------------------------------------------+
Ответ       | USC-ISIC.ARPA. 86400 IN CNAME   C.ISI.EDU.      |
            | C.ISI.EDU.     86400 IN A       10.0.0.52       |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Отметим, что бит AA в заголовке отклика гарантирует полномочность данный, соответствующих QNAME, но не говорит о том, что данные для C.ISI.EDU являются полномочными. Такой полный отклик возможен, поскольку A.ISI.EDU является полномочным для домена ARPA, где найден USC-ISIC.ARPA, и домена ISI.EDU, где найден C.ISI.EDU.

Если такой же запрос передаст серверу C.ISI.EDU, его отклик может оказаться таким же, если он имеет в кэше свой адрес или отклик будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A        |
            +-------------------------------------------------+
Ответ       | USC-ISIC.ARPA.   86400 IN CNAME   C.ISI.EDU.    |
            +-------------------------------------------------+
Полномочия  | ISI.EDU.        172800 IN NS      VAXA.ISI.EDU. |
            |                           NS      A.ISI.EDU.    |
            |                           NS      VENERA.ISI.EDU.|
            +-------------------------------------------------+
Дополнения  | VAXA.ISI.EDU.   172800    A       10.2.0.27     |
            |                 172800    A       128.9.0.33    |
            | VENERA.ISI.EDU. 172800    A       10.1.0.52     |
            |                 172800    A       128.9.0.32    |
            | A.ISI.EDU.      172800    A       26.3.0.103    |
            +-------------------------------------------------+
```

Это сообщение содержит полномочный отклик для псевдонима USC-ISIC.ARPA и ссылку на серверы имен для ISI.EDU. Такой тип откликов встречается нечасто для случая запроса имени хоста для сервера имен, которому передается запрос, но может встречаться достаточно часто для других псевдонимов.

6.2.8. QNAME=USC-ISIC.ARPA, QTYPE=CNAME

Если этот запрос отправить серверу A.ISI.EDU или C.ISI.EDU, отклик будет иметь вид:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Вопрос      | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=CNAME    |
            +-------------------------------------------------+
Ответ       | USC-ISIC.ARPA. 86400 IN CNAME     C.ISI.EDU.    |
            +-------------------------------------------------+
Полномочия  | <пусто>                                         |
            +-------------------------------------------------+
Дополнения  | <пусто>                                         |
            +-------------------------------------------------+
```

Поскольку QTYPE=CNAME, запись CNAME RR сама по себе является ответом на вопрос и сервер не будет пытаться найти что-либо для C.ISI.EDU (возможным исключением являются дополнения).

6.3. Пример преобразования

Следующие примеры иллюстрируют операции, которые распознаватель должен выполнять для своих клиентов. Будем предполагать, что распознаватель начинает свою работу без кэша (например, после перезагрузки системы). Далее предположим, что система не является одним из хостов, для которых есть данные, хост размещается где-то в сети 26, в структура SBELT включает следующую информацию:

```
    Счетчик соответствий = -1
    SRI-NIC.ARPA.   26.0.0.73       10.0.0.51
    A.ISI.EDU.      26.3.0.103
```

Эта информация показывает серверы, к которым можно обращаться, с их адресами и содержит значение счетчика совпадений -1, говорящее о том, что серверы недостаточно близки к цели. Отметим, что значение счетчика (-1) не рассматривается, как количественная мера удаленности.

Приведенные далее примеры показывают использование кэша, т. е. запросы в примерах предполагаются выполняющимися уже после других запросов.

6.3.1. Определение MX для ISI.EDU.

Предположим, что первый запрос к преобразователю делается от локальной почтовой программы, которой нужно отправить письмо по адресу PVM@ISI.EDU. Почтовая программа может запросить записи MX RR для домена ISI.EDU.

Распознаватель может поискать в кэше записи MX RR для ISI.EDU, но пустой кэш не поможет. После этого распознаватель поймет необходимость запроса данных у внешних серверов и попытается найти подходящий сервер. Поиск будет включать записи NS RR для доменов ISI.EDU, EDU и корня. Записей в кэше для них не будет. В качестве последнего шага распознаватель будет вынужден использовать данные из SBELT, копируя их в свою структуру SLIST.

На этом этапе преобразователю нужно выбрать один из трех доступных адресов. С учетом нахождения распознавателя в сети 26, для первой попытки разумно выбрать адрес 26.0.0.73 или 26.3.0.103. По выбранному адресу будет отправлен запрос вида:

```
            +-------------------------------------------------+
Заголовок   | OPCODE=SQUERY                                   |
            +-------------------------------------------------+
Вопрос      | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX             |
            +-------------------------------------------------+
```

```
Ответ       | <пусто>                                              |
            +------------------------------------------------------+
Полномочия  | <пусто>                                              |
            +------------------------------------------------------+
Дополнения  | <пусто>                                              |
            +------------------------------------------------------+
```

Распознаватель будет ждать отклика или тайм-аута. При возникновении тайм-аута будет предпринята попытка запроса к другому серверу, по другим адресам тех же серверов с повтором процедуры при отсутствии отклика. В конце концов может быть получен от сервера SRI-NIC.ARPA отклик вида:

```
            +------------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE                              |
            +------------------------------------------------------+
Вопрос      | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX                  |
            +------------------------------------------------------+
Ответ       | <пусто>                                              |
            +------------------------------------------------------+
Полномочия  | ISI.EDU.          172800 IN NS       VAXA.ISI.EDU.   |
            |                           NS       A.ISI.EDU.      |
            |                           NS       VENERA.ISI.EDU. |
            +------------------------------------------------------+
Дополнения  | VAXA.ISI.EDU.     172800    A        10.2.0.27       |
            |                   172800    A        128.9.0.33      |
            | VENERA.ISI.EDU.   172800    A        10.1.0.52       |
            |                   172800    A        128.9.0.32      |
            | A.ISI.EDU.        172800    A        26.3.0.103      |
            +------------------------------------------------------+
```

Распознаватель увидит, что информация в отклике дает более близкое к ISI.EDU делегирование по сравнению с SLIST (соответствие в 3 метки). После этого распознаватель примет решение о кэшировании данных из отклика и ее включение в новую структуру SLIST:

```
    Счетчик соответствий = 3
    A.ISI.EDU.      26.3.0.103
    VAXA.ISI.EDU.   10.2.0.27       128.9.0.33
    VENERA.ISI.EDU. 10.1.0.52       128.9.0.32
```

A.ISI.EDU присутствует в этом списке, как и в предыдущем, но это просто случайность. Распознаватель будет передавать новые запросы и ждать откликов. В конце концов он получит отклик вида:

```
            +------------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                          |
            +------------------------------------------------------+
Вопрос      | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX                  |
            +------------------------------------------------------+
Ответ       | ISI.EDU.                MX 10 VENERA.ISI.EDU.        |
            |                         MX 20 VAXA.ISI.EDU.          |
            +------------------------------------------------------+
Полномочия  | <пусто>                                              |
            +------------------------------------------------------+
Дополнения  | VAXA.ISI.EDU.     172800  A  10.2.0.27               |
            |                   172800  A  128.9.0.33              |
            | VENERA.ISI.EDU.   172800  A  10.1.0.52               |
            |                   172800  A  128.9.0.32              |
            +------------------------------------------------------+
```

Распознаватель поместит полученную информацию в свой кэш и вернет клиенту записи MX RR.

6.3.2. Определение имени хоста по адресу 26.6.0.65

Распознаватель будет транслировать это в запрос PTR RR для 65.0.6.26.IN-ADDR.ARPA. Эта информация не кэшируется, поэтому распознаватель будет искать внешние серверы для запроса. Подходящих серверов нет, поэтому снова будет использоваться SBELT (отметим, что серверы для ISI.EDU имеются в кэше, но ISI.EDU не является предком 65.0.6.26.IN-ADDR.ARPA, поэтому используется SBELT).

Поскольку запрос относится к полномочным данным для обоих серверов в SBELT, в конечном итоге возвращен может быть показанный ниже отклик.

```
            +------------------------------------------------------+
Заголовок   | OPCODE=SQUERY, RESPONSE, AA                          |
            +------------------------------------------------------+
Вопрос      | QNAME=65.0.6.26.IN-ADDR.ARPA.,QCLASS=IN,QTYPE=PTR    |
            +------------------------------------------------------+
Ответ       | 65.0.6.26.IN-ADDR.ARPA.     PTR     ACC.ARPA.        |
            +------------------------------------------------------+
Полномочия  | <пусто>                                              |
            +------------------------------------------------------+
Дополнения  | <пусто>                                              |
            +------------------------------------------------------+
```

6.3.3. Получение адреса хоста по имени poneria.ISI.EDU

Такие запросы будут транслироваться в запрос типа A для poneria.ISI.EDU. Распознаватель не найдет кэшированных данных, но сможет найти в кэше записи NS RR для ISI.EDU при поиске серверов для отправки запроса. С использованием этих данных он построит структуру SLIST вида:

```
Счетчик соответствий = 3
A.ISI.EDU.      26.3.0.103
VAXA.ISI.EDU.   10.2.0.27       128.9.0.33
VENERA.ISI.EDU. 10.1.0.52
```

Сервер A.ISI.EDU указан первым в предположении, что распознаватель упорядочиват список, а A.ISI.EDU находится в той же сети.

Один из включенных в список серверов ответит на такой запрос.

7. Литература

[Dyer 87] Dyer, S., and F. Hsu, "Hesiod", Project Athena Technical Plan - Name Service, April 1987, version 1.9.

Описывает основы службы имен Hesiod.

[IEN-116] J. Postel, "Internet Name Server", IEN-116, USC/Information Sciences Institute, August 1979.

Служба имен, замененная DNS, но еще используемая.

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

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

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

[RFC-793] J. Postel, "Transmission Control Protocol", RFC-793, USC/Information Sciences Institute, September 1981.

[RFC-799] D. Mills, "Internet Name Domains", RFC-799, COMSAT, September 1981.

Предлагает использование иерархического именования в Internet взамен плоской схемы имен.

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

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

Отменен. См. RFC-952.

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

Отменен. См. RFC-953.

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

[RFC-819] Z. Su, and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC-819, Network Information Center, SRI International, August 1982.

Предварительные соображения по организации системы доменных имен. Современная реализация совершенно иная.

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

[RFC-830] Z. Su, "A Distributed System for Internet Name Service", RFC-830, Network Information Center, SRI International, October 1982.

Предварительные соображения по организации системы доменных имен. Современная реализация совершенно иная.

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

Заменен настоящим документом.

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

Заменен настоящим документом.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC-920, USC/Information Sciences Institute October 1984.

Описание схемы именования для доменов верхнего уровня.

[RFC-952] K. Harrenstien, M. Stahl, E. Feinler, "DoD Internet Host Table Specification", RFC-952, SRI, October 1985.

Задает формат файлов HOSTS.TXT, которые использовались до DNS.

[RFC-953] K. Harrenstien, M. Stahl, E. Feinler, "HOSTNAME Server", RFC-953, SRI, October 1985.

В этом RFC содержится официальная спецификация протокола для сервера HOSTNAME, отмененного протоколом DNS. Основанная на TCP версия протокола обеспечивает доступ к информации в формате RFC-952 и служит для получения копии таблицы хостов.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC-973, USC/Information Sciences Institute, January 1986.

Описывает изменения, внесенные в RFC 882 и RFC 883, а также причины этих изменений. Отменен.

[RFC-974] C. Partridge, "Mail routing and the domain system", RFC-974, CSNET CIC BBN Labs, January 1986.

Описывает переход от адресации почты на базе файла HOSTS.TXT к более мощной системе на базе записей MX в DNS.

[RFC-1001] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and Methods", RFC-1001, March 1987.

Этот документ и RFC-1002 описывают предварительную реализацию протокола NETBIOS на базе TCP/IP, основанную на организации службы имен NetBIOS «поверх» DNS.

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

[RFC-1010] J. Reynolds and J. Postel, "Assigned Numbers", RFC-1010, USC/Information Sciences Institute, May 1987

Описывает номера сокетов и мнемонику имен хостов, операционных систем и т. п.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition", RFC-1031, November 1987.

Описывает план перехода от MILNET к DNS.

[RFC-1032] M. K. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC-1032, November 1987.

Описывает правила регистрации, используемые NIC для администрирования доменов верхнего уровня и делегирования субзон.

[RFC-1033] M. K. Lottor, "Domain Administrators Operations Guide", RFC-1033, November 1987.

«Поваренная книга» администратора домена.

[Solomon 82] M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server", Computer Networks, vol 6, nr 3, July 1982.

Описывает службу имен для CSNET, которая независима от DNS, а также использование DNS в CSNET.

Предметный указатель

Абсолютное имя 4
Доменное имя 1, 4
Завершение 8
Запись о ресурсе 5
Запрос состояния 8
Зона 3, 9
Инверсные запросы 8
Итеративная модель 2
Код операции 7
Кэширование негативных откликов 12
Метка 4
Относительное имя 4
Ошибка в имени 14
ошибка имени 12, 14pp.
Перенос зон 12
Полномочный сервер 3
Почтовый ящик 4
Преобразователь 3
Псевдоним 7, 14
Раздел 7
Регистр символов 4
Рекурсивная модель 2
Рекурсивный сервис 10
Сервер имен 2, 8
Склеивающие данные 9
Стандартный запрос 7
Шаблоны 11
A 6
AUTHORITY 3
AXFR 8
CH 6
CNAME 6
HINFO 6
IN 6
MX 6
NE 14
NS 6
Opcode 7
PTR 6
QCLASS 7
QNAME 7
QTYPE 7
RDATA 6
RESOLVER 3
RR 5
safety belt 15p., 21
SOA 6
TTL 6
ZONE 3

Перевод на русский язык

Николай Малых

nmalykh@protocols.ru