

Доменные имена — реализация и спецификация

DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

1. Статус документа

В этом RFC описаны детали и протокол системы доменных имен. Предполагается, что читатель знаком с концепциями, рассмотренными в связанном документе - Domain Names - Concepts and Facilities [RFC-1034].

Система доменных имен представляет собой комбинацию функций и типов данных, относящихся к официальному протоколу, а также функций и типов данных, остающихся экспериментальными. Поскольку система доменных имен непрерывно развивается, следует принимать во внимание, что новые типы данных и экспериментальное поведение всегда будут встречаться в компонентах системы, выходящих за пределы официального протокола. Компоненты официального протокола включают стандартные очереди, отклики и форматы данных RR¹ (например, адреса хостов). С момента выхода предшествующей серии RFC некоторые определения были изменены, а часть определений утратила силу.

Экспериментальные или устаревшие функции явно отмечены в этой серии RFC - использовать их следует с осторожностью.

Читателя следует специально предупредить о том, что приведенные в примерах значения не являются актуальными или полными - они приведены лишь для наглядности. Документ можно распространять свободно.

Оглавление

1. Статус документа.....	1
2. Введение.....	2
2.1. Обзор.....	2
2.2. Конфигурации общего назначения.....	3
2.3. Соглашения.....	4
2.3.1. Предпочтительный синтаксис имен.....	4
2.3.2. Порядок передачи данных.....	4
2.3.3. Регистр символов.....	5
2.3.4. Ограничение размера.....	5
3. Определения пространства доменных имен и RR.....	5
3.1. Определения пространства доменных имен.....	5
3.2. Определения RR.....	5
3.2.1. Формат.....	5
3.2.2. Значения типа.....	6
3.2.3. Значения QTYPE.....	6
3.2.4. Значения CLASS.....	6
3.2.5. Значения QCLASS.....	6
3.3. Стандартные RR.....	6
3.3.1. Формат CNAME RDATA.....	6
3.3.2. Формат HINFO RDATA.....	7
3.3.3. Формат MB RDATA (эксперим.).....	7
3.3.4. Формат MD RDATA (устаревш.).....	7
3.3.5. Формат MF RDATA (устаревш.).....	7
3.3.6. Формат MG RDATA (эксперим.).....	7
3.3.7. Формат MINFO RDATA (экспер.).....	7
3.3.8. Формат MR RDATA (эксперим.).....	7
3.3.9. Формат MX RDATA.....	8
3.3.10. Формат NULL RDATA (экспер.).....	8
3.3.11. Формат NS RDATA.....	8
3.3.12. Формат PTR RDATA.....	8
3.3.13. Формат SOA RDATA.....	8
3.3.14. Формат TXT RDATA.....	9
3.4. Internet RR.....	9
3.4.1. Формат A RDATA.....	9
3.4.2. Формат WKS RDATA.....	9
3.5. Домен IN-ADDR.ARPA.....	9
3.6. Определение новых типов, классов и специальных пространств имен.....	10
4. Сообщения.....	11
4.1. Формат.....	11

¹Resource Record - запись о ресурсе. Прим. перев.

- 4.1.1. Формат заголовочного раздела..... 11
- 4.1.2. Формат вопросительного раздела..... 12
- 4.1.3. Формат записи о ресурсе..... 12
- 4.1.4. Сжатие сообщений..... 12
- 4.2. Транспорт..... 13
 - 4.2.1. Использование UDP..... 13
 - 4.2.2. Использование TCP..... 13
- 5. Первичные файлы..... 14
 - 5.1. Формат..... 14
 - 5.2. Использование первичных файлов для определения зон..... 15
 - 5.3. Пример первичного файла..... 15
- 6. Реализация сервера имен..... 15
 - 6.1. Архитектура..... 15
 - 6.1.1. Управление..... 15
 - 6.1.2. База данных..... 16
 - 6.1.3. Время..... 16
 - 6.2. Обработка стандартных запросов..... 16
 - 6.3. Обновление и перезагрузка зоны..... 16
 - 6.4. Инверсные запросы (опция)..... 17
 - 6.4.1. Содержимое инверсных запросов и откликов..... 17
 - 6.4.2. Пример инверсного запроса и отклика..... 17
 - 6.4.3. Обработка инверсного запроса..... 17
 - 6.5. Завершение запросов и откликов..... 18
- 7. Реализация преобразователя имен..... 18
 - 7.1. Преобразование пользовательских запросов в запросы DNS..... 18
 - 7.2. Передача запросов..... 18
 - 7.3. Обработка откликов..... 19
 - 7.4. Использование кэша..... 19
- 8. Поддержка электронной почты..... 19
 - 8.1. Привязка почтового обмена..... 20
 - 8.2. Привязка почтового обмена (эксперим.)..... 20
- 9. Литература..... 20
- Предметный указатель..... 22

2. Введение

2.1. Обзор

Целью системы доменных имен является обеспечение механизма именования ресурсов таким способом, чтобы имена были бы приемлемы для разных хостов, сетей, семейств протоколов, межсетевое взаимодействия, административных организация.

С точки зрения пользователя доменные имена полезны в качестве аргументов для локального агента, называемого преобразователем (resolver), который восстанавливает информацию, связанную с доменным именем. Таким образом, пользователь может получить адрес хоста или почтовую информацию, связанную с конкретным доменным именем. Чтобы позволить пользователям запрашивать конкретный тип информации, запрос соответствующего типа передается преобразователю вместе с доменным именем. Для пользователя дерево доменов представляет собой единое информационное пространство - преобразователь отвечает за сокрытие от пользователя распределения данных между разными серверами имен.

С точки зрения преобразователя база данных, образующая пространство доменных имен, распределена по разным серверам имен. Различные части пространства имен хранятся на разных серверах, хотя отдельные элементы данных могут в целях резервирования размещаться на двух и более серверах имен. Преобразователь начинает с нахождения хотя бы одного сервера имен. Когда преобразователь обрабатывает пользовательский запрос, он запрашивает информацию у известного сервера имен. В ответ преобразователь получает желаемые данные или ссылку на другой сервер имен. Используя серверы, на которые ссылается известный сервер имен, преобразователь отождествляет другие серверы имен и хранящуюся на них информацию. Преобразователи отвечают за работу с распределенным пространством имен и при возникновении отказов используют резервные базы данных на других серверах.

Серверы имен работают с двумя типами данных. Первый тип хранится в наборах, называемых зонами (zone); каждая зона представляет собой полную базу данных для одного «вырезанного» суб-дерева пространства доменных имен. Этот тип данных называют полномочными (authoritative). Сервер имен периодически проверяет актуальность данных своих зон и, обнаружив несоответствие, получает обновленные данные зоны из первичных (master) файлов, хранящихся локально или на другом сервере имен. Ко второму типу относятся кэшированные данные, которые были получены локальным преобразователем. Эти данные могут быть неполными, но их наличие существенно повышает



домена и необходимость синхронизации между серверами имен и преобразователями косвенно задают характеристики базы данных, но выбор реального формата остается за разработчиком. Поток информации можно организовать так, чтобы группа хостов действовала совместно в целях оптимизации. Иногда это выполняется для разгрузки слабых хостов, чтобы на них не нужно было реализовать полнофункциональный преобразователь имен. Это может быть приемлемо для ПК и хостов, которые хотят минимизировать объем требуемого для работы сетевого кода. Такая схема также позволяет группе хостов совместно использовать незначительное число кэшей взамен организации многочисленных кэшей, исходя из того, что централизация кэшей будет увеличивать частоту запросов к тем или иным данным к кэше. В этом случае преобразователи заменяются «заглушками» (stub resolver), которые действуют в качестве «препроцессоров» для преобразователей, расположенных в рекурсивном сервере одного или нескольких серверов имен, про которые известно, что они предоставляют такие услуги (см. рисунок справа).

Отметим, что в любом случае компоненты домена всегда реплицируются для повышения надежности.

2.3. Соглашения

В системе доменных имен используются некоторые соглашения, связанные с вопросами низкого уровня, но имеющими фундаментальное значение. Хотя разработчики вправе отказаться от выполнения этих соглашений **внутри своей системы**, они **должны** следовать соглашениям в части **всех** аспектов поведения, доступных для других хостов.

2.3.1. Предпочтительный синтаксис имен

Спецификации DNS пытаются сохранить максимально доступный уровень обобщения для правил создания доменных имен. Идея заключается в том, чтобы любые существующие объекты могли быть представлены доменными именами с минимизацией требуемых изменений.

Однако при выделении доменного имени для объекта благоразумный пользователь будет стараться выбрать имя, которое будет соответствовать как правилам системы доменных имен, так и существующим для данного объекта правилам, которые опубликованы или подразумеваются существующими программами.

Например, при именовании почтового домена пользователь должен следовать правилам данного документа и RFC 822. При создании нового имени хоста следует выполнять старые правила для файла HOSTS.TXT. Это позволит избежать проблем со старыми программами при использовании ими доменных имен.

Показанный ниже синтаксис не будет вызывать существенных проблем для большинства программ, использующих доменные имена (например, электронной почты TELNET).

```
<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 А до Z и от а до z
<digit> ::= любая из десяти цифр от 0 до 9
```

Отметим, что в доменных именах могут использоваться строчные и прописные буквы, а регистр символов не имеет значения (т. е., строчные и прописные буквы в доменных именах не различаются).

Метки должны следовать правилам для имен хостов ARPANET. Имена должны начинаться с буквы, заканчиваться могут буквой или цифрой, а в середине могут включать буквы, цифры и знак дефиса. Существуют также некоторые ограничения на размер имени - метка не должна включать более 63 символов.

Например, приведенные ниже имена могут идентифицировать хосты Internet.

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

2.3.2. Порядок передачи данных

Порядок передачи заголовка и данных, описанных в настоящем документе, определяется на уровне октетов. is resolved to the octet level. Всякий раз, когда на диаграмме показана группа октетов, порядок передачи этих октетов совпадает с порядком их прочтения, принятом в английском языке¹. В примере, показанном на рисунке справа, октеты будут передаваться в порядке их нумерации на рисунке.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 | 1 1 1 1 1 1 1 1 | 2 2 2 2 2 2 2 2 |
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 | 3 3 3 3 3 3 3 3 | 4 4 4 4 4 4 4 4 |
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 | 5 5 5 5 5 5 5 5 | 6 6 6 6 6 6 6 6 |
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 | 0 1 0 1 0 1 0 1 |
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
 Когда октет представляет собой численное значение, расположенный слева на рисунке бит является старшим (наиболее значимым). Т. е., бит, помеченный номером 0 на рисунке слева, является старшим. В этом примере показано десятичное число 170.

Аналогично этому для случаев когда числовое значение представлено многооктетным полем левый бит является самым старшим для этого поля. При передаче многооктетного поля сначала передается старший октет.

¹Слева направо. Прим. перев.

2.3.3. Регистр символов

Для всех компонент DNS, которые являются частью официального протокола, сравнение символьных строк (меток, доменных имен и т. п.) осуществляется без учета регистра символов. В настоящее время это правило не имеет исключений в системе доменных имен. Однако в будущем может потребоваться использование всех битов октетов, поэтому следует избегать попыток использования для доменных имен 7-битовую кодировку ASCII или применения специальных октетов завершения меток и т. п.

При вводе данных в систему доменных имен по возможности следует сохранять регистр символов. Однако в некоторых случаях такое сохранение невозможно. Например, если две записи RR сохраняются в базе данных и одна представлена, как x.y, а вторая, - как X.Y, реально они будут храниться в одном месте базы и, следовательно, только в одном из регистров. Основное правило сохранения регистра символов для таких случаев «отбрасывается» лишь в тех ситуациях, когда данные служат для определения структуры в базе данных и два имени будут совпадать, если не принимать во внимание регистр символов.

Потеря чувствительно к регистру символов информации должна быть минимизирована. Таким образом, хотя данные для x.y и X.Y могут храниться в одном месте, как x.y или X.Y, данные для a.x и B.X не будут храниться в форме A.x, A.X, b.x или b.X. В общем случае такой подход сохраняет регистр первой метки доменного имени, но заставляет стандартизировать регистр меток внутренних узлов.

Системным администраторам при вводе данных в систему доменных имен следует быть осторожными с регистрами символов в доменных именах, если их системы различают строчные и прописные буквы. Система распространения данных в DNS будет гарантировать сохранение соответствующего представления.

2.3.4. Ограничение размера

Для объектов и параметров в DNS существуют ограничения на размеры, перечисленные здесь. Некоторые из этих ограничений легко могут быть изменены, а другие являются более фундаментальными.

- метки не более 63 октетов;
- имена не более 255 октетов;
- TTL положительные значения в формате 32-битового целого числа со знаком;
- Сообщения UDP не более 512 октетов

3. Определения пространства доменных имен и RR

3.1. Определения пространства доменных имен

Доменные имена в сообщениях представляются в форме последовательности меток.

Каждая метка представляется в форме 1-октетного поля размера, за которым следует соответствующее число октетов. Поскольку каждое доменное имя завершается null-меткой корня, завершающим октетом доменного имени является поле размера пустой метки со значением 0. Два старших бита каждого октета размера должны иметь значение 0, а оставшиеся 6 октетов позволяют создавать метки размером до 63 октетов.

Для упрощения реализации общий размер доменного имени (т. е., число октетов меток и октетов размера) ограничен значением 255 октетов.

Хотя метки могут содержать в своих октетах любые 8-битовые значения, настоятельно рекомендуется следовать предпочтительному синтаксису, описанному в этом документе и обеспечивающему совместимость с существующими соглашениями по именованию. Серверы имен и преобразователи должны сравнивать метки без учета регистра символов (т. е., A=a), предполагая использование кодировки ASCII с нулевым битом четности.

3.2. Определения RR

3.2.1. Формат

Все записи RR используют на верхнем уровне единый формат, показанный на рисунке справа.

		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
NAME	имя владельца (т. е., узла, которому принадлежит запись о ресурсе).	----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															
TYPE	два октета, содержащие код типа RR.	----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															
CLASS	два октета, содержащие код класса RR.	----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															
TTL	32-битовое целое число со знаком ¹ , задающее временной интервал, в течение которого запись может кэшироваться прежде, чем снова возникнет необходимость обращения к источнику данных.	----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															
RDLENGTH	Нулевое значение этого поля говорит о том, что RR может использоваться только для текущей транзакции и эту запись не следует кэшировать. Например, записи SOA всегда распространяются с нулевым значением TTL для запрета их кэширования. Нулевые значения используются также для постоянно обновляющихся данных.	----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															
		----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----															

RDLENGTH 16-битовое целое число без знака, задающее размер поля RDATA в октетах.

¹В параграфе 4.1.3 приведено определение поля TTL еще раз и оно трактуется, как целое число без знака. Найти разрешения этого конфликта в других RFC не удалось. Ошибка отмечена в [RFC Errata](#). Прим. перев.

RDATA строка октетов переменного размера, описывающая ресурс. Формат этой информации меняется в зависимости от типа (TYPE) и класса (CLASS) записи о ресурсе.

3.2.2. Значения типа

Поля TYPE используются в записях о ресурсах. Отметим, что эти типы являются подмножеством QTYPE.

Тип	Значение	Описание
A	1	Адрес хоста
NS	2	Полномочный сервер имен
MD	3	Назначение для электронной почты (устарело, используется MX)
MF	4	Пересылка для электронной почты (устарело, используется MX)
CNAME	5	Каноническое имя для псевдонима
SOA	6	Показывает начало зоны полномочий
MB	7	Доменное имя почтового ящика (эксперимент)
MG	8	Член почтовой группы (эксперимент)
MR	9	Доменное имя переименования почты (эксперимент)
NULL	10	Пустая RR (эксперимент)
WKS	11	Описание общеизвестного сервиса
PTR	12	Указатель доменного имени
HINFO	13	Информация о хосте
MINFO	14	Информация о почтовом ящике или почтовом списке
MX	15	Почтовый обмен
TXT	16	Текстовая строка

3.2.3. Значения QTYPE

Поля QTYPE (тип запроса) появляются в вопросительной части запросов. QTYPE является надмножеством TYPE и, следовательно, все значения типов корректны для поля QTYPE. В дополнение к этому определены несколько значений QTYPE.

AXFR	252	запрос на перенос зоны целиком;
MAILB	253	запрос почтовых записей (MB, MG или MR);
MAILA	254	запрос RR для почтового агента (устарел, см. MX);
*	255	запрос на все записи.

3.2.4. Значения CLASS

Поля класса¹ используются в записях о ресурсах. Ниже приведен список определенных значений класса.

IN	1	Internet;
CS	2	класс CSNET (устарел, используется лишь в качестве примера в некоторых устаревших RFC);
CH	3	класс CHAOS;
HS	4	Hesiod [Dyer 87];

3.2.5. Значения QCLASS

Поля QCLASS (класс запроса) появляются в вопросительной части запросов. Значения QCLASS являются надмножеством значений CLASS, поэтому для них применимы все определенные значения классов. В дополнение к этому определен 1 специальный класс запросов.

*	255	любой класс
---	-----	-------------

3.3. Стандартные RR

Приведенные ниже определения RR потенциально могут появляться во всех классах. В частности, записи NS, SOA, CNAME и PTR будут использоваться во всех классах в одинаковом формате. Поскольку формат RDATA для этих записей известен, все доменные имена в разделах RDATA этих RR могут сжиматься.

Строка <domain-name> указывает доменное имя в форме последовательности меток, завершаемое меткой нулевого размера. <character-string> представляет собой 1-октетное поле размера, за которым следует соответствующее число символов. <character-string> трактуется, как двоичные данные и может иметь размер до 256 октетов (с октетом размера).

3.3.1. Формат CNAME RDATA

CNAME	Строка	<domain-name>,	задающая	/		CNAME	/
		каноническое или основное имя владельца.	/				/
		Имя владельца является псевдонимом.					

¹Точнее было бы сказать «класса сети». *Прим. перев.*

Записи CNAME RR не требуют дополнительной обработки раздела, но сервер имен может выбирать в некоторых случаях повторный запрос для канонического имени. Логика работы серверов имен описана в [RFC-1034].

3.3.2. Формат HINFO RDATA

CPU	Строка	<character-string>	, задающая	тип	-----	CPU	/
	процессора.				-----		/
OS	Строка	<character-string>	, задающая	тип	-----	OS	/
	операционной системы.				-----		/

Стандартные значения поле CPU и OS можно найти в документе [RFC-1010].

Записи HINFO используются для получения общих сведений о хосте. Основное предназначение — протоколы типа FTP, которые могут использовать специальные процедуры при работе между однотипными машинами или операционными системами.

3.3.3. Формат MB RDATA (эксперим.)

MADNAME	Строка	<domain-name>	, задающая хост,	/	-----	MADNAME	/
	на котором расположен		указанный	/	-----		/
	почтовый ящик.				-----		/

Записи MB вызывают дополнительную обработку раздела, при которой определяются записи RR типа A, соответствующие MADNAME.

3.3.4. Формат MD RDATA (устаревш.)

MADNAME	Строка	<domain-name>	, задающая хост с	/	-----	MADNAME	/
	почтовым агентом для		домена, которому	/	-----		/
	следует обеспечивать		возможность		-----		/
	доставки почты в этот		домен.		-----		/

Записи MD вызывают дополнительную обработку раздела, при которой определяются записи RR типа A, соответствующие MADNAME.

Записи MD признаны устаревшими. Новая схема подробно описана в определении записи MX и [RFC-974]. При обнаружении MD RR в первичном файле запись следует отбрасывать или преобразовать в MX RR с приоритетом 0.

3.3.5. Формат MF RDATA (устаревш.)

MADNAME	Строка	<domain-name>	, задающая хост с	/	-----	MADNAME	/
	почтовым агентом для		домена, который	/	-----		/
	будет принимать почту		для пересылки в		-----		/
	домен.				-----		/

Записи MF вызывают дополнительную обработку раздела, при которой определяются записи RR типа A, соответствующие MADNAME.

Записи MF признаны устаревшими. Новая схема подробно описана в определении записи MX и [RFC-974]. При обнаружении MF RR в первичном файле запись следует отбрасывать или преобразовать в MX RR с приоритетом 10.

3.3.6. Формат MG RDATA (эксперим.)

MGMNAME	Строка	<domain-name>	, задающая	/	-----	MGMNAME	/
	почтовый ящик, который		является членом	/	-----		/
	почтовой группы, заданной		доменным		-----		/
	именем.				-----		/

Записи MG не вызывают дополнительной обработки.

3.3.7. Формат MINFO RDATA (экспер.)

RMAILBX	Строка	<domain-name>	, задающая	/	-----	RMAILBX	/
	почтовый ящик, который		отвечает за		-----		/
	список рассылки, или		почтовый ящик.	/	-----	EMAILBX	/
	Если доменное имя		указывает на корень,		-----		/
	владелец MINFO RR		отвечает сам за себя.		-----		/
	Отметим, что многие		из существующих		-----		/
	списков рассылки		используют почтовый		-----		/
	ящик X-request в		качестве значения		-----		/
	поля RMAILBX для		списка рассылки X		-----		/
	(например, Msggroup-		request для списка		-----		/
	Msggroup). Это поле		обеспечивает более		-----		/
	общий механизм.				-----		/

EMAILBX Строка <domain-name>, задающая почтовый ящик, который получает сообщения об ошибках, связанные со списком рассылки или почтовым ящиком, заданным владельцем MINFO RR (подобно полю ERRORS-TO:, которое было предложено). Если доменное имя указывает на корень, ошибки должны возвращаться отправителю сообщения.

Записи MINFO не вызывают дополнительной обработки. Хотя эти записи могут быть связаны с обычным почтовым ящиком, обычно их используют для списков рассылки.

3.3.8. Формат MR RDATA (эксперим.)

NEWNAME	Строка	<domain-name>	, задающая	/	-----	NEWNAME	/
	почтовый ящик, который		подобающим	/	-----		/
	образом переименовывает		почту		-----		/
	указанного почтового		ящика.		-----		/

Записи SOA не вызывают дополнительной обработки.

Все временные параметры задаются в секундах.

Большинство из этих полей имеет отношение только к операциям поддержки сервера имен. Однако поле MINIMUM используется для всех запросов, которые возвращают записи RR из зоны. Всякий раз, когда RR передается в ответ на запрос, в поле TTL устанавливается большее из значений поля TTL в RR и поля MINIMUM в соответствующей записи SOA. Значение поля MINIMUM является нижней границей значений TTL для всех RR в зоне. Отметим, что такое использование поля MINIMUM возникает, когда RR копируются в отклик, а не при загрузке зоны из первичного файла или при использовании переноса зоны. Причина такого поведения обусловлена желанием позволить будущим средствам динамического обновления менять SOA RR с известной семантикой.

3.3.14. Формат TXT RDATA

TXT-DATA Одна или множество строк <character-string>.

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/                               TXT-DATA                               /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Записи TXT RR используются для хранения описательной информации. Семантика текста зависит от домена, к которому относится запись.

3.4. Internet RR

3.4.1. Формат A RDATA

ADDRESS 32-битовый адрес Internet.

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               ADDRESS                               |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Хосты с множеством адресов Internet будут иметь множественные записи типа A.

Записи A не вызывают дополнительной обработки. Раздел RDATA в строке A первичного файла представляет собой адрес IP в форме четырех десятичных значений, разделенных точками без каких-либо пробельных символов (например, 10.2.0.52 или 192.0.5.6).

3.4.2. Формат WKS RDATA

ADDRESS 32-битовый адрес Internet.

PROTOCOL 8-битовый номер протокола IP.

<BIT MAP> Строка битов переменного размера (размер должен быть кратным 8).

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               ADDRESS                               |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               PROTOCOL                               |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                               <BIT MAP>                             |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Записи WKS¹ используются для описания общеизвестных услуг, поддерживаемых определенным протоколом на конкретном адресе IP. Поле PROTOCOL задает номер протокола IP, а строка битов показывает активность портов для заданного протокола. Первый бит строки соответствует порту 0, второй - порту 1 и т. д. Если бит для интересующего протокола не указан, его значение предполагается нулевым. Допустимые значения и мнемоника для портов и протоколов описаны в [RFC-1010].

Например, если PROTOCOL=TCP (6), 26-й бит будет соответствовать порту TCP с номером 25 (SMTP). Если этот бит установлен, серверу SMTP следует прослушивать порт 25 для протокола TCP, а нулевое значение бита говорит о том, что услуги SMTP не поддерживаются для указанного адреса.

Целью записей WKS RR является предоставление информации о доступности серверов TCP и UDP. Если сервер поддерживает оба протокола или имеет множество адресов Internet, используется множество записей WKS RR.

Записи WKS не вызывают дополнительной обработки.

В первичных файлах порты и протоколы указываются с использованием мнемоники или десятичных номеров.

3.5. Домен IN-ADDR.ARPA

В сети Internet используется специальный домен для поддержки местоположения шлюзов и отображения адресов Internet на имена хостов. В других классах могут использоваться аналогичные стратегии. Назначение специального домена заключается в обеспечении гарантированного метода отображения адресов хостов на их имена и обслуживания запросов на поиск всех шлюзов в конкретной сети, входящей в Internet.

Отметим, что оба эти сервиса похожи по функциям, которые могут выполняться в инверсных запросах - различие заключается в том, что эта часть пространства доменных имен структурирована по адресам и, следовательно, обеспечивается гарантия нахождения требуемой информации без полного просмотра пространства доменных имен.

Домен начинается от «корня» IN-ADDR.ARPA и имеет подструктуру, соответствующую структуре адресации Internet.

Имена в домене IN-ADDR.ARPA определяются так, чтобы они имели четыре метки в дополнение к суффиксу IN-ADDR.ARPA. Каждая метка представляет один октет адреса Internet и выражается строкой символов десятичного значения из диапазона 0-255 (нули перед значащими цифрами опускаются за исключением случаев, когда октет адреса имеет значение 0).

Адреса хостов представляются доменными именами, в которых заданы все 4 метки. Таким образом, данные для адреса Internet 10.2.0.52 размещаются в домене 52.0.2.10.IN-ADDR.ARPA. Обратный порядок указания октетов адреса позволяет делегировать зоны, представляющие собой в точности одну сеть в адресном пространстве. Например, 10.IN-ADDR.ARPA может быть зоной, содержащей данные для сети ARPANET, а 26.IN-ADDR.ARPA может быть отдельной зоной для сети MILNET. Адреса хостов используются для хранения указателей на основное имя хостов в обычном пространстве доменных имен.

¹Well known service.

Номера сетей соответствуют неким нетерминальным узлам на различных уровнях домена IN-ADDR.ARPA, поскольку номера сетей Internet включают 1, 2 или 3 октета. Сетевые узлы используются для хранения указателей на первичные имена хостов для шлюзов, подключенных этим сетям. Поскольку шлюз по определению подключается более, чем к одной сети, на него обычно будет указывать не менее двух сетевых узлов. Шлюзы будут иметь также указатели уровня хостов для своих полных адресов.

Оба указателя для шлюзов (полный и сетевой) используют записи PTR RR, указывающие на основное доменное имя соответствующих хостов.

Например, в домене IN-ADDR.ARPA будет содержаться информация о шлюзе ISI между сетями 10 и 26, шлюзе MIT между сетью 10 и принадлежащей MIT сети 18, а также хостах A.ISI.EDU и MULTICS.MIT.EDU. В предположении, что шлюз ISI имеет адреса 10.2.0.22, 26.0.0.103 и имя MILNET-GW.ISI.EDU, а шлюз MIT - адреса 10.0.0.77, 18.10.0.4 и имя GW.LCS.MIT.EDU, база данных домена будет содержать строки:

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

Таким образом, программа, которая хочет найти шлюзы в сети 10, будет генерировать запрос в форме QTYPE=PTR, QCLASS=IN, QNAME=10.IN-ADDR.ARPA. После обработки этого запроса в отклике будут получены две записи RR:

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

После этого программа может выдать запрос QTYPE=A, QCLASS=IN для MILNET-GW.ISI.EDU. и GW.LCS.MIT.EDU., чтобы определить адреса Internet для этих шлюзов.

Преобразователь имен, который хочет определить имя хоста с адресом Internet 10.0.0.6, будет генерировать запрос в форме QTYPE=PTR, QCLASS=IN, QNAME=6.0.0.10.IN-ADDR.ARPA и получит в ответ:

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

При использовании этих услуг следует принимать во внимание перечисленные ниже обстоятельства.

- Поскольку домен IN-ADDR.ARPA является специальным и обычная нотация для конкретного хоста или шлюза будет размещаться в другой зоне, существует возможность рассогласования данных.
- Шлюзы часто имеют два имени в разных доменах, но только одно имя может быть первичным.
- Системы, использующие базу данных DNS для инициализации своих таблиц маршрутизации, должны стартовать с обеспечением достаточной информации о шлюзах, чтобы гарантировать возможность доступа к соответствующим серверам имен.
- Данные шлюза отражают только существование шлюза в манере, эквивалентной современным файлам HOSTS.TXT. Это не является заменой динамического обмена информацией с помощью протоколов GGP и EGP.

3.6. Определение новых типов, классов и специальных пространств имен

К числу определенных ранее типов и классов относятся те, которые существовали к моменту выхода этого документа. Следует ожидать появления новых определений. В этом параграфе даны некоторые рекомендации разработчикам по добавлению новых определений. Общие вопросы ввода новых определений обсуждаются в почтовой конференции NAMEDROPPERS@SRI-NIC.ARPA.

В общем случае новые типы оправданы, когда в базу данных добавляется новая информация о существующем объекте или требуются новые форматы данных для совершенно нового объекта. Разработчикам следует пытаться определять типы и форматы RDATA для них так, чтобы они были применимы во всех классах и не возникало дублирования информации. Новые классы оправданы в тех случаях, когда DNS будет использоваться для новых протоколов, т. е., потребуются новые форматы данных специально для этого класса или желательно иметь копию существующего пространства имен, для которой требуется независимое управление.

Новые типы и классы потребуют новой мнемоники для первичных файлов - формат этих файлов требует, чтобы мнемоника для типов и классов была не была связанной.

Значения TYPE и CLASS должны быть корректным подмножеством QTYPE и QCLASS, соответственно.

Существующая система использует множество RR для представления многочисленных значений типа вместо хранения множества значений в разделе RDATA одной записи RR. Для многих приложений это менее эффективно, но позволяет уменьшить размер RR. Множество предложений в части RR включено в некоторые экспериментальные работы по методам динамического обновления.

Существующая система пытается минимизировать дублирование данных в базе для сохранения согласованности. Таким образом, для того, чтобы найти адрес хоста почтового обмена, почтовый домен отображается на имя хоста, а это имя далее отображается на адрес (вместо прямого отображения). Такой подход является более предпочтительным, поскольку он позволяет предотвратить возможную несогласованность.

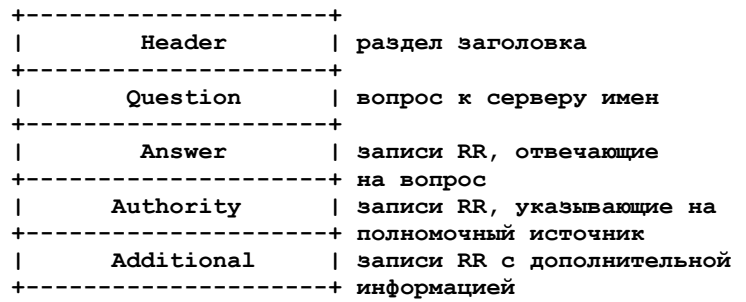
При определении нового типа данных не следует использовать множество типов RR для упорядочивания элементов или выражения различных форматов для эквивалентных привязок. Вместо этого следует передавать информацию в теле RR, используя один тип. Это правило позволяет избежать проблем с кэшированием множества типов и определением QTYPE для многочисленных типов.

Например, исходная форма привязки почтового обмена использует два типа RR - один для представления «ближайшего» (MD), а другой - для «менее близкого» (MF). Сложность заключается в том, что наличие одного типа RR в кэше не дает никакой информации о другом, поскольку в запросе, обеспечившем кэширование информации для QTYPE могло использоваться значение MF, MD или MAILA (соответствует обоим). Переработанный сервис импользует один тип MX с полем «приоритета» в разделе RDATA, которое позволяет упорядочить разные RR. Однако при наличии в кэше любой записи MX RR остальные тоже должны присутствовать там.

4. Сообщения

4.1. Формат

Весь обмен данными в рамках протокола DNS осуществляется с использованием единого формата, основанного на сообщениях. На верхнем уровне сообщение делится на 5 разделов (некоторые разделы могут оставаться пустыми в конкретном сообщении), как показано на рисунке справа.



Раздел заголовка присутствует в каждом сообщении. Заголовок включает поля, задающие присутствие остальных разделов, характер сообщения (запрос или отклик, стандартный запрос или дополнительная операция).

Имена разделов после заголовка выбраны с учетом их функциональности в стандартных запросах. Вопросительный раздел содержит поля, описывающие запрос к серверу имен. К таким поля относятся QTYPE (тип запроса), QCLASS (класс запроса) и QNAME (запрашиваемое доменное имя). Три последних раздела имеют одинаковый формат и содержат (возможно пустой) список «сцепленных» (конкатенация) записей о ресурсах (RR). Раздел ответа содержит отвечающие на вопрос записи RR, раздел полномочий содержит RR, указывающие полномочный сервер имен, а дополнительный раздел включает RR, связанные с запросом, но, строго говоря, не являющимися ответом на него.

4.1.1. Формат заголовочного раздела

Заголовок сообщения содержит перечисленные ниже и показанные на рисунке справа поля

		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
ID	16-битовое целое число, присваиваемое программой, которая генерирует запрос любого типа. Этот идентификатор копируется в соответствующий ответ и может использоваться запрашивающей стороной для сопоставления откликов с ожидающими выполнения запросами.	ID															
QR	Однобитовое поле, показывающее, является это сообщение запросом (0) или откликом (1).	QR	Opcode		AA	TC	RD	RA	Z	RCODE							
		QDCOUNT															
		ANCOUNT															
		NSCOUNT															
		ARCOUNT															
OPCODE	Четырехбитовое поле, задающее тип запроса в сообщении. Это значение устанавливается инициатором запроса и копируется в отклик																
	0	стандартный запрос (QUERY);															
	1	инверсный запрос (IQUERY);															
	2	запрос статуса сервера (STATUS);															
	3-15	резерв для использования в будущем.															
AA	Полномочный ответ (Authoritative Answer) - этот флаг допустим в откликах и указывает, что отвечающий сервер имен является полномочным для домена, указанного в вопросительном разделе запроса.																
		Отметим, что содержимое раздела ответов может включать множество имен владельцев в результате наличия псевдонимов. Бит AA относится к имени, которое соответствует имени в запросе, или первому имени владельца в разделе ответов.															
TC	Отсечка (TrunCation) - показывает, что часть сообщения была отсечена в результате превышения размера, допустимого для канала передачи.																
RD	Запрос рекурсии (Recursion Desired) - этот флаг может устанавливаться в запросе и копироваться в отклик. Установленный флаг RD инициирует рекурсивное выполнение запроса сервером имен. Поддержка рекурсивных запросов является необязательной.																
RA	Доступность рекурсии (Recursion Available) - этот флаг устанавливается или сбрасывается в отклике в соответствии с состоянием поддержки рекурсивных запросов сервером имен.																
Z	Зарезервировано для использования в будущем. Должно иметь значение 0 во всех запросах и откликах.																
RCODE	Код отклика - 4-битовое поле, являющееся частью ответа. Интерпретация значений показана ниже.																
	0	нет ошибок;															
	1	ошибка в формате - сервер имен не способен интерпретировать запрос;															
	2	отказ сервера - сервер имен не способен выполнить запрос в результате внутренних проблем;															
	3	ошибка в имени - имеет значение только для откликов от полномочного сервера имен; этот код означает, что представленного в запросе доменного имени не существует;															

Схема компресси позволяет представлять доменное имя в сообщении любым из трех способов:

- последовательность меток, завершающаяся нулевым октетом;
- указатель;
- последовательность меток, завершающаяся указателем.

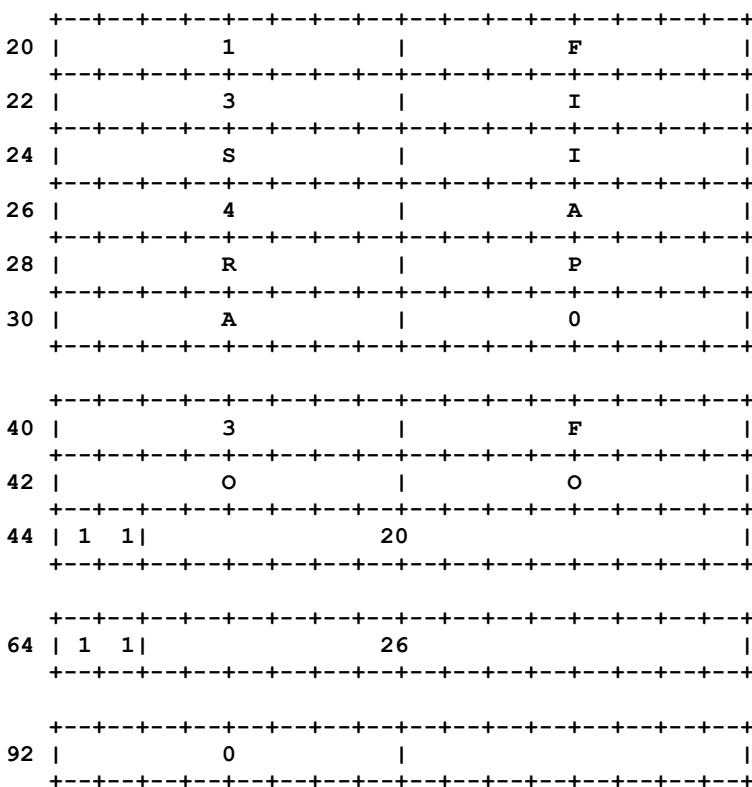
Указатели можно использовать только для тех входящих доменных имен, где формат не зависит от класса. Если это условие не выполняется, серверу имен и преобразователю потребуется знать форматы всех обслуживаемых RR. В настоящее время таких случаев не наблюдается, но они могут возникать для будущих форматов RDATA.

Если доменное имя содержится в части сообщения, учитываемой полем размера (например, RDATA в записи RR) и используется компрессия, при расчете размера следует использовать размеж сжатого представления имени, а не его исходную длину.

Программы могут отказаться от использования указателей в генерируемых сообщениях, но это будет снижать «емкость» дейтаграмм и может приводить к отсечке содержимого. Однако все программы должны понимать входящие сообщения, содержащие указатели.

Например, дейтаграмма может потребоваться для доменных имен F.ISI.ARPA, FOO.F.ISI.ARPA, ARPA и корня. Без учета остальных полей сообщения эти доменные имена можно представить, как показано на рисунке справа.

Доменное имя F.ISI.ARPA показано со смещением 20. Имя FOO.F.ISI.ARPA показано со смещением 40 и использует указатель на определенное ранее имя F.ISI.ARPA, объединенный с меткой для FOO. Имя ARPA определено со смещением 64 и использует указатель на компоненту ARPA имени F.ISI.ARPA (смещение 20); отметим, что этот указатель задает смещение именно метки ARPA, а не значение 20, по которому начинается полное имя. Имя корневого домена определено одним октетом (0) со смещением 92 (имя корневого домена не имеет метки).



4.2. Транспорт

DNS предполагает, что сообщения будут передаваться в форме дейтаграмм или в байтовом потоке через виртуальный канал. Хотя виртуальные каналы могут применяться для любых операций DNS, для запросов более предпочтительно использование дейтаграмм, поскольку в этом случае снижается объем служебной информации и повышается производительность. Операции переноса зон должны использовать виртуальные устройства, поскольку для таких операций требуется надежная доставка.

В сети Internet доступ к серверам имен поддерживается по протоколу TCP [RFC-793] через порт 53 (десятичный), а также с помощью дейтаграмм UDP [RFC-768] через порт с тем же номером.

4.2.1. Использование UDP

Сообщения, передаваемые по протоколу UDP, используют на сервере порт с десятичным номером 53.

Размер сообщений, передаваемых по протоколу UDP, ограничен 512 байтами (без учета заголовков IP и UDP). Более длинные сообщения укорачиваются с установкой флага TC в заголовке.

Протокол UDP не подходит для переноса зон, но его использование рекомендуется для стандартных запросов в Internet. Переданные по протоколу UDP сообщения могут теряться, поэтому требуется механизм повтора передачи. Порядок передачи запросов и откликов может изменяться в сети или при обработке запроса на сервере имен, поэтому преобразователи имен должны работать независимо от порядка доставки пакетов.

Оптимальная стратегия повтора передачи по протоколу UDP может меняться в зависимости от производительности Internet и потребностей клиента, но рекомендуется следовать приведенным ниже рекомендациям:

- клиенту следует попытаться воспользоваться другими серверами и другими адресами сервера прежде, чем повторять запрос по тому же адресу;
- по возможности интервал повтора следует выбирать на основе предшествующей статистики; слишком агрессивная политика повтора может замедлить отклики для всех клиентов сервера; в зависимости от способа соединения клиента с предполагаемым сервером следует устанавливать минимальный интервал повтора 2 — 5 сек.

Дополнительные рекомендации по выбору политики повтора приведены ниже в описании преобразователей имен.

4.2.2. Использование TCP

Сообщения, передаваемые по протоколу TCP, используют на сервере порт с десятичным номером 53. Сообщение имеет двухбайтовый префикс размера, в котором учитывается и размер этого поля. Это поле размера обеспечивает возможность обработки на низком уровне для сборки полного сообщения до начала его разбора.

Рекомендуется следовать приведенным ниже правилам управления соединениями:

- серверу не следует блокировать других операций во время ожидания данных TCP;
- серверу следует поддерживать множественные соединения;
- серверу следует предполагать, что каждый клиент будет инициировать завершение соединения и задерживать завершение соединения со своей стороны, пока все остающиеся запросы клиента не будут завершены;
- если серверу нужно закрыть бездействующее соединение в целях экономии ресурсов, ему следует дождаться момента, когда время бездействия соединения превысит примерно две минуты; в частности, серверу следует разрешать последовательности запросов SOA и AXFR (с которых начинаются операции обновления) в одном соединении; если сервер совсем не способен ответить на запрос, он может использовать односторонний разрыв или сброс соединения вместо аккуратного завершения.

5. Первичные файлы

Первичные файлы являются текстовыми и содержат записи RR в виде текста. Поскольку содержимое зоны может быть выражено в форме списка RR, первичные файлы используются в основном для определения зон, хотя их можно применять и для списков содержимого кэша. По этой причине в данном разделе обсуждается формат RR в первичном файле, а также специально рассматриваются случаи использования первичных файлов для определения зон.

5.1. Формат

Первичные файлы представляют собой последовательность записей. Записи представляют собой в основном полную строку, хотя могут использоваться скобки для создания многострочных записей, а в тексте могут содержаться последовательности CRLF¹. Все комбинации символов пробела и табуляции служат разделителями между отдельными элементами записей. Любая строка первичного файла может включать в конце комментарий. Началом комментария служит точка с запятой (;).

Определены следующие типы записей:

```
<blank>[<comment>]
$ORIGIN <domain-name> [<comment>]
$INCLUDE <file-name> [<domain-name>] [<comment>]
<domain-name><rr> [<comment>]
<blank><rr> [<comment>]
```

Пустые строки (с комментариями или без таковых) могут включаться в любом месте файла.

Объявлены две управляющих записи: \$ORIGIN и \$INCLUDE. Вслед за \$ORIGIN указывается доменное имя и эта запись служит в качестве индикатора завершения использования предыдущей точки отсчета (origin) для относительных имен и начала отсчета таких имен от указанного в записи доменного имени. Запись \$INCLUDE служит для включения указанного в ней файла в текущий файл и может опционально задавать доменное имя, которое служит точкой отсчета для относительных доменных имен во включенном файле. Запись \$INCLUDE может также включать некое дополнительное содержимое. Отметим, что запись \$INCLUDE никогда не меняет точки отсчета для родительского файла независимо от изменения таковой для включаемого файла.

Две последних формы служат для представления RR. Если запись для RR начинается с пустого пространства (blank), предполагается, что эта запись RR принадлежит указанному последним владельцу. Если RR начинается с <domain-name>, имя предыдущего владельца «сбрасывается».

Элемент <rr> может иметь одну из приведенных ниже форм:

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

Записи RR начинаются с необязательных полей TTL и класса, за которыми следуют поля типа и RDATA (в соответствии с типом и классом). Поля класса и типа используют стандартную мнемонику, поле TTL содержит десятичное целое число. Опущенные значения класса и TTL предполагаются совпадающими с последними явно заданными значениями этих полей. Поскольку мнемоника для типов и классов независима, разбор строки будет однозначным. Отметим, что этот порядок отличается от порядка, использованного в примерах, и порядка в реальных RR - он упрощает разбор и использование принятых по умолчанию значения.

Строки <domain-name> составляют значительную часть первичного файла. Метки в доменных именах представляются символьными строками, разделенными точками. Соглашения по кватированию позволяют использовать в доменных именах произвольные символы. Доменное имя, заканчивающееся точкой, называется абсолютным - это полное имя. Доменные имена без точки в конце называют относительными; полное доменное имя в этом случае образуется путем добавления строки, заданной полем \$ORIGIN, \$INCLUDE или иным аргументом программы загрузки первичного файла. При отсутствии точки отсчета относительные доменные имена приводят к возникновению ошибки.

<character-string> выражается в форме непрерывной последовательности символов без внутренних пробелов или в форме строки, заключенной в кавычки ("). Между символами кавычек могут включаться любые символы, за исключением символа кавычек. При необходимости включения кавычек перед символом кавычки используется знак \ (обратная дробная черта).

Поскольку первичные файлы являются текстовыми, требуется несколько вариантов специального кодирования, обеспечивающих возможность загрузки любых данных. В частности:

@ символ @ в отдельной строке обозначает текущее значение точки отсчета.

X где X может быть любым символом, за исключением цифр (0-9), используется для кватирования символа, имеющего специальное значение. Например, «\.» может использоваться для включения точки в метку.

¹Перевод строки, возврат каретки. Прим. перев.

\DDD (где каждый символ D представляет собой цифру) представляет собой октет, соответствующий десятичному числу, описываемому DDD. Получаемое в результате значение считается текстом и для него не применяется специальных трактовок.

() скобки используются для группировки данных, занимающих более одной строки. Символы перевода строки и возврата каретки внутри скобок игнорируются.

; точка с запятой указывает начало комментария - расположенная справа от нее часть строки игнорируется.

5.2. Использование первичных файлов для определения зон

При использовании первичного файла для загрузки зоны операцию следует прерывать в случае обнаружения любой ошибки в файле. Причина такого поведения обусловлена серьезностью последствий, которые может вызывать единственная ошибка. Например, при наличии синтаксической ошибки в RR, определяющих передачу полномочий, сервер будет возвращать ошибки полномочного имени для всех имен в подзоне (за исключением случая, когда субзона также присутствует на сервере).

Ниже перечислены некоторые проверки, которые следует выполнять для первичного файла в дополнение к проверке синтаксиса.

1. Все RR в одном файле должны относиться к одному классу.
2. На вершине зоны должна присутствовать единственная запись SOA RR.
3. Если присутствует делегирование и нужна склеивающая запись, эта запись должна присутствовать.
4. Информация, представленная за пределами полномочных узлов зоны, должна быть склеивающей информацией, а не результатом ошибки в точке отсчета или аналогичной ошибки.

5.3. Пример первичного файла

Ниже приведен пример первичного файла, который может определять зону ISI.EDU и загружается с точкой отсчета ISI.EDU.

```
@ IN SOA VENERA Action\.domains (
    20 ; SERIAL - порядковый номер
    7200 ; REFRESH - обновление
    600 ; RETRY - период повтора
    3600000 ; EXPIRE - срок жизни
    60) ; MINIMUM - минимальное время жизни

    NS A.ISI.EDU.
    NS VENERA
    NS VAXA
    MX 10 VENERA
    MX 20 VAXA

A A 26.3.0.103

VENERA A 10.1.0.52
A 128.9.0.32

VAXA A 10.2.0.27
A 128.9.0.33
```

\$INCLUDE <SUBSYS>ISI-MAILBOXES.TXT

Файл <SUBSYS>ISI-MAILBOXES.TXT содержит строки:

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

Отметим, что использование символа \ в записи SOA RR обусловлено наличием точки в почтовом адресе ответственного за зону лица Action.domains@E.ISI.EDU¹.

6. Реализация сервера имен

6.1. Архитектура

Реальная структура сервера имен будет зависеть от операционной системы хоста, интеграции функций с преобразователем имен, поддержки рекурсии и использования вместе с преобразователем общей базы данных. В этом разделе рассмотрены вопросы реализации сервера имен, разделяющего базу данных с преобразователем имен, но большая часть рассмотрения применима ко всем реализациям серверов имен.

6.1.1. Управление

Сервер имен должен обеспечивать возможность одновременного выполнения множества операций, независимо от того, реализуется такая обработка в виде загрузки разных задач в ОС хоста или в форме мультиплексирования в рамках одной программы сервера имен. Для сервера имен недопустимо блокировать запросы UDP по время ожидания

¹В исходном документе ошибочно указано «Action.domains@E.ISI.EDU» (см. [RFC Errata](#)). Прим. перев.

данных TCP для обновления или переноса зон. Аналогично, серверу имен не следует пытаться использовать рекурсию без параллельной обработки запроса, хотя сервер может создание очереди для запросов одного клиента или рассматривать идентичные запросы от одного клиента, как дубликаты. Серверу имен не следует задерживать обработку запросов в процессе перезагрузки зоны из первичного файла или встраивания обновленной зоны в свою базу данных.

6.1.2. База данных

Хотя реализации серверов имен могут использовать любые внутренние структуры данных, предлагаемая здесь структура состоит из трех частей:

- «каталог» данных, в котором перечислены зоны, поддерживаемые этим сервером и приведены «указатели» на структуры данных зон; основной целью этой структуры является нахождение ближайшей родительской зоны (если таковая имеется) для поступающих стандартных запросов;
- отдельные структуры данных для каждой из поддерживаемых сервером зон;
- структура для кэшированных данных (возможно с отдельными кэшами для каждого класса).

Все эти структуры данных могут быть реализованы в идентичных форматах с древовидной структурой с различными данными, «прицепленными» к узлам на разных путях - в каталоге данные указывают на зоны, а в зонах и записях кэша в качестве данных будут служить RR. При разработке структуры дерева следует принимать во внимание то, что при обработке запроса будет осуществляться перемещение по дереву с использованием чувствительного к регистру сравнения меток, а также то, что некоторые узлы будут иметь очень высокий коэффициент ветвления (100 - 1000 или более), но основная часть узлов будет иметь очень низкий коэффициент ветвления (0 или 1).

Одним из способов решения проблем с регистром символов является сохранение меток для каждого узла в двух частях - одна часть будет содержать собой стандартизованное по регистру представление метки, где все символы ASCII будут иметь одинаковый регистр, а вторая часть будет представлять собой битовую маску, обозначающие символы, которые заданы в другом регистре. Пока коэффициент ветвления не превышает некоего порога, для обработки ветвления на узле может использоваться простой связный список, а после превышения заданного порога - хэш-структура. В любом случае хэш-структуры, используемые для хранения частей дерева, должны обеспечивать соответствие процедур и функций хэширования соглашениям DNS в части сохранения регистров.

Использование отдельных структур для разных частей базы данных обусловлено несколькими факторами, перечисленными ниже.

- Для каталога может использоваться почти статическая структура, которую требуется изменять лишь в тех случаях, когда администратор меняет поддерживаемые сервером зоны. Эта структура может использоваться также для хранения параметров управления процессами обновления.
- Отдельные структуры данных для зон позволяют заменять зону путем простого изменения указателя в каталоге. Операции обновления зон могут строить новые структуры и по завершении обновления просто менять указатель в базе данных. Это очень важное обстоятельство, поскольку в процессе обновления зоны для запросов не следует использовать одновременно старую и новую копию данных.
- При правильной организации поиска полномочные данные в зоне всегда будут «спрятаны», отдавая преимущество кэшированным данным.
- Ошибки в определениях зон, которые могут приводить к перекрытию зон и т. п., могут приводить к возврату некорректных откликов на запросы, но определение таких проблем достаточно просто и содержимое «плохой» зоны не может повреждать другие зоны.
- Поскольку кэш обновляется наиболее часто, он более уязвим при перезапуске системы. Кэш может также полностью состоять из устаревших данных RR. В любом случае кэш легко отбросить без повреждения данных зоны.

Основным аспектом устройства базы данных является выбор структуры, которая позволит серверу имен преодолевать аварийные ситуации на хосте, где работает сервер имен. Информация о состоянии, которую серверу имен следует сохранять при авариях на хосте, включает структуру каталога (с данными по статусу обновления для каждой зоны) и собственно данные зон.

6.1.3. Время

Как данные TTL для RR, так и данные о времени операций обновления зависят от 32-битовых таймеров для отсчета секунд. Внутри базы данных таймеры обновления и значения TTL для кэшированных данных используют «обратный отсчет», а для данных в зоне сохраняются постоянные значения TTL.

Рекомендуемая стратегия реализации включает два варианта хранения значений времени - относительное приращение и абсолютное значение. Одним из вариантов реализации является использование положительных 32-битовых целых чисел для одного типа и отрицательных чисел - для другого. Записи RR в зонах используют относительное время, а таймеры обновления и кэширования - абсолютное. Абсолютные значения отсчитываются от некоего известного начального момента и преобразуются в относительные значения при включении в отклик на запрос. Когда абсолютное значение TTL становится отрицательным после преобразования в относительное, это говорит о том, что данные устарели и их следует игнорировать.

6.2. Обработка стандартных запросов

Основной алгоритм обработки стандартных запросов представлен в [RFC-1034].

При обработке запросов с QCLASS=* или иным значением QCLASS, которому соответствует множество классов, отклик не должен быть полномочным за исключением тех случаев, когда сервер может гарантировать, что отклик перекрывает все классы.

При создании отклика записи RR, которые будут включаться в дополнительный раздел, но являются дубликатами RR из раздела answer или authority, могут быть опущены в дополнительном разделе.

Если отклик слишком велик и требуется отсечка, начинать отсечку следует в конце отклика. Таким образом, при наличии каких-либо данных в разделе полномочий для раздела ответа гарантируется уникальность.

Значение MINIMUM в записи SOA следует использовать минимального значения TTL для распространяемых из зоны данных. Функцию установки значения времени жизни следует выполнять при копировании данных в отклик. Это позволит будущим динамическим протоколам обновления менять поле SOA MINIMUM без использования неоднозначной семантики.

6.3. Обновление и перезагрузка зоны

Несмотря на все усилия сервера, он может оказаться не способным загрузить данные зоны из первичного файла по причине синтаксических ошибок и т. п. или не сможет обновить зону в соответствии с заданными параметрами обновления. В таких случаях серверу следует отвечать на запросы, как будто он не владеет этой зоной.

Если ведущий сервер передает информацию зоны с использованием AXFR и в процессе передачи создается новая версия, серверу следует, по возможности, продолжать передачу старой версии. В любом случае серверу никогда не следует передавать часть зоны одной версии, а часть - другой. Если завершение передачи невозможно, ведущему серверу следует сбросить соединение, через которое осуществлялась передача зоны.

6.4. Инверсные запросы (опция)

Инверсные запросы являются необязательной частью DNS. От серверов имен не требуется поддержки той или иной формы инверсных запросов. Если сервер имен получает неподдерживаемый инверсный запрос, он возвращает сообщение об ошибке с установленным флагом Not Implemented (не реализовано) в заголовке. Хотя поддержка инверсных запросов не является обязательной, каждый сервер имен должен поддерживать, по крайней мере, возврат сообщений об ошибке.

6.4.1. Содержимое инверсных запросов и откликов

Инверсные запросы выполняют отображение, обратное тому, которое осуществляется при стандартных запросах - стандартный запрос служит для отображения доменного имени на ресурс, а инверсный отображает ресурс на доменное имя. Например, стандартный запрос может связать доменное имя с адресом хоста, а инверсных - адрес хоста с доменным именем.

Инверсные запросы включают одну запись RR в разделе ответов, а вопросительный раздел остается пустым. Имя владельца записи RR в запросе и значение TTL для нее значения не имеют. Отклик содержит в разделе question вопросы, которые идентифицируют все имена, владеющие указанной в запросе RR, **которые сервер может знать**. Поскольку ни один сервер имен не имеет информации обо всем пространстве имен, отклик никогда не считается полным. Таким образом, инверсные запросы полезны, прежде всего, для управления базами данных и отладки. Инверсные запросы **не** являются допустимым методом отображения адреса хоста на доменное имя, для этой цели используется специальный домен IN-ADDR.ARPA.

По возможности серверам имен следует обеспечивать независимое от регистра сравнение символов для инверсных запросов. Таким образом, инверсные запросы MX RR для доменных имен Venera.isi.edu и VENERA.ISI.EDU должны возвращать одинаковые результаты, равно, как инверсные запросы HINFO RR для имен IBM-PC UNIX и IBM-PC UNIX. Однако выполнение этого правила не гарантируется, поскольку сервер имен может владеть записями RR, содержащими строки символов, для которых сервер имен не знает, что эти данные являются символьными.

При обработке инверсного запроса сервер возвращает один из двух вариантов отклика:

- 0, 1 или множество доменных имен для указанного ресурса в форме значений QNAME в разделе вопросов;
- код ошибки, показывающий, что сервер имен не поддерживает обратное отображение для заданного типа ресурса.

Когда отклик на инверсный запрос содержит одно или несколько полей QNAME, имя владельца и значение TTL для записи RR в разделе ответов, который определяет инверсный запрос, меняется в точном соответствии с записью RR в первом поле QNAME.

Записи RR, возвращаемые в ответ на инверсные запросы, не могут кэшироваться с использованием того же механизма, который применяется для откликов на стандартные запросы. Одной из причин этого является то, что имя может иметь множество RR одного типа, а появляться будет только одна запись. Например, инверсный запрос для одного адреса многодомного хоста может создавать впечатление о существовании единственного адреса.

6.4.2. Пример инверсного запроса и отклика

Общая структура инверсного запроса для определения доменного имени, соответствующего адресу Internet, 10.1.0.52 показана на рисунке справа.

В этом запросе задается вопрос, ответом на который является Internet-адрес 10.1.0.52. Поскольку имя владельца неизвестно, в качестве «заполнителя» может быть указано любое доменное имя (это значение игнорируется). С целью минимизации размера сообщения обычно используется один октет со значением 0, указывающий на корень. Величина TTL для RR не имеет значения. Отклик на этот запрос показан на рисунке слева.

Header		OPCODE=IQUERY, ID=997	
Question		<пусто>	
Answer		<anyname> A IN 10.1.0.52	
Authority		<пусто>	
Additional		<пусто>	

Header	OPCODE=RESPONSE, ID=997, QR=1	Отметим, что значение QTYPE в отклике на инверсный запрос совпадает со значением поля TYPE в разделе ответов инверсного запроса.
Question	QTYPE=A, QCLASS=IN, QNAME=VENERA.ISI.EDU	Отклики на инверсные запросы могут включать множество разделов Questions, поскольку обратное преобразование не обязано быть уникальным. Если вопросительный раздел в отклике не пуст, RR в разделе ответа изменяется в соответствии с точной копией RR в первом QNAME.
Answer	VENERA.ISI.EDU A IN 10.1.0.52	
Authority	<пусто>	
Additional	<пусто>	

6.4.3. Обработка инверсных запросов

Серверы имен, поддерживающие инверсные запросы, могут обеспечивать выполнение таких операций путем полного просмотра своих баз данных, но это становится непрактичным по мере роста размера баз. Другим вариантом является инвертирование баз данных в соответствии с ключами поиска.

Для серверов имен, поддерживающих множество зон и большой объем данных, рекомендуется поддерживать инверсные данные отдельно для каждой зоны. При изменении отдельной зоны в процессе обновления потребуется вносить изменения только в одну инверсную зону.

Поддержка переноса инверсных зон может быть добавление в будущие версии DNS, а в данной версии спецификации отсутствует.

6.5. Завершение запросов и откликов

Оptionальные службы завершения, описанные в RFC 882 и RFC 883, были упразднены. В будущем могут быть восстановлены переработанные варианты этих служб.

7. Реализация преобразователя имен

Верхние уровни рекомендуемого алгоритма преобразования имен описаны в [RFC-1034]. В этом разделе рассматриваются детали реализации в предположении, что структура базы данных соответствует предложениям раздела реализации сервера имен настоящего документа.

7.1. Преобразование пользовательских запросов в запросы DNS

На первом этапе преобразователь конвертирует запрос клиента, выраженный в принятом локальной ОС формате, в спецификацию поиска записей RR для заданного имени в соответствии со значениями QTYPE и QCLASS. По возможности, значения QTYPE и QCLASS следует задавать соответствующими одному типу и одному классу, поскольку это значительно упрощает использование кэшированных данных. Причина этого заключается в том, что присутствие данных одного типа не позволяет сделать выводов о наличии или отсутствии данных других типов, следовательно, единственным способом получения достоверной информации остается обращение к полномочному источнику. Если используется QCLASS=*, ответы полномочного источника перестают быть доступными.

Поскольку для эффективной работы преобразователя требуется мультиплексирование множества запросов, каждый незавершенный запрос обычно представляется неким блоком данных о его состоянии. Такой блок обычно включает:

- Временная метка, показывающая момент начала запроса. Эта метка служит для принятия вопроса об актуальности записей RR в базе данных. Для метки используется абсолютный формат представления времени, описанный выше при рассмотрении вопроса хранения RR в зонах и кэше. Отметим, что в случаях, когда RR TTL указывает относительное время, запись RR должна быть актуальной, поскольку она является частью зоны. Когда RR имеет абсолютное время, это говорит о том, что запись является частью кэша и значение TTL записи RR сравнивается с временной меткой начала запроса.

Отметим, что использование временных меток обеспечивает преимущества по отношению к использованию текущего времени, поскольку в этом случае обеспечивается возможность обычного ввода в кэш записей RR с TTL=0 и продолжения их использования текущим запросом даже по истечении продолжительного интервала в результате значительной загрузки системы, повтора передачи запроса и т. п.

- Некоторый сорт параметров для ограничения объема работы при выполнении запроса.

Объем работы, выполняемой преобразователем имен в ответ на запрос клиента, должен быть ограничен для защиты от ошибок в базе данных типа циклических ссылок CNAME и эксплуатационных проблем типа «разделения», которые могут препятствовать требуемому доступу к серверам имен. Хотя локальные ограничения на число повторов преобразователем одного запроса к конкретному серверу имен играют достаточно важную роль, преобразователям имен следует поддерживать глобальный счетчик для каждого запроса, чтобы ограничить объем работы в рамках отдельного запроса. Для счетчика следует устанавливать некое начальное значение и уменьшать его при выполнении каждой операции (тайм-аут повтора, повтор передачи и т. п.). При достижении счетчиком нулевого значения запрос прерывается с возвратом сообщения о временной ошибке.

Отметим, что если структура преобразователя имен позволяет начинать запрос во время обработки других запросов, когда необходимость доступа к серверу имен для одного запроса вызывает параллельное преобразование для адреса сервера имен, порожденный запрос следует начинать с уменьшенным значением счетчика. Это предотвращает от проблем, связанных с циклическими ссылками в базе данных, которые могут вести к росту объема работы при преобразовании имен.

- Структуру данных SLIST, описанную в [RFC-1034].

Эта структура служит для отслеживания состояния запроса, для которого возникает необходимость ожидания ответов от внешних серверов имен.

7.2. Передача запросов

Как описано в [RFC-1034], основной задачей преобразователя имен является формулировка запроса, который будет давать ответ на запрос клиента, и передача этого запроса серверам имен, которые могут обеспечить информацию. Преобразователь обычно имеет лишь строгие рекомендации по выбору серверов для запроса информации в форме записей NS RR и может потребоваться пересмотр запроса с результате полученных значений CNAME или смены набора серверов имен, которые запрашивает преобразователь в ответ на отклики о передаче полномочий, указывающие преобразователю на серверы имен, более близкие к запрашиваемой информации. В дополнение к запрашиваемой клиентом информации преобразователю может потребоваться обращение к собственным функциям для определения сервера имен, с которым он желает контактировать.

В любом случае, используемая в этом документе модель предполагает, что преобразователь разделяет (мультиплексирует) внимание между множеством запросов, часть которых порождена клиентами, а остальные - самим преобразователем. Каждый запрос представляется некоей информацией о состоянии и желаемое поведение заключается в том, что преобразователь передает запросы серверам имен таким способом, который обеспечивает максимальную вероятность получения ответа, минимизирует время, затрачиваемое на выполнение запроса, и предотвращает избыточный обмен информацией. Ключевой алгоритм использует данные о состоянии запросов для выбора следующего сервера имен с целью адресации запроса ему, а также рассчитывает значение тайм-аута, по истечении которого следует выполнить следующее действие, если отклик не будет получен. Следующим действием обычно является передача запроса другому серверу, но возможна также генерация для клиента сообщения о временной ошибке.

Преобразователь всегда начинает выполнение запроса со списка серверов имен (SLIST). Этот список содержит все записи NS RR, соответствующие ближайшей родительской зоне, о которых известно преобразователю. Во избежание проблем при старте преобразователю следует иметь набор используемых по умолчанию серверов, к которым ему следует обращаться при отсутствии подходящих записей NS RR. Преобразователь добавляет в список SLIST все известные ему адреса серверов имен и может запускать параллельные запросы для получения адресов серверов, известных преобразователю только по именам.

Для завершения инициализации списка SLIST преобразователь добавляет к нему всю информацию об истории взаимодействия для каждого из адресов в SLIST. Эта информация обычно включает средневзвешенное время отклика от соответствующего адреса и усредненная «эффективность» для этого адреса (т. е., доля запросов, на которые были получены ответы). Отметим, что эту информацию следует хранить отдельно для каждого адреса, а не для каждого сервера имен, поскольку время отклика и средняя эффективность конкретного сервера может существенно меняться для разных адресов этого сервера. Отметим также, что эта информация обычно привязана к паре адресов преобразователя и сервера имен, поэтому преобразователям с множеством адресов следует отдельно сохранять историю для каждого из своих адресов. Для части адресов история может отсутствовать - в этом случае ожидаемое время кругового обхода следует принимать равным 5 - 10 сек. в худшем случае, уменьшая это значение для некоторых ЛВС и т. п. сетей.

Отметим, что при каждой передаче полномочий для зон алгоритм преобразователя заново инициализирует SLIST.

Эта информация обеспечивает некоторое ранжирование адресов доступных серверов имен. При каждом выборе адреса состояние для него следует менять для предотвращения выбора того же адреса при наличии других незанятых адресов. Тайм-аут для каждой передачи следует устанавливать на 50-100% больше среднего ожидаемого времени отклика с целью компенсации возможных вариаций при откликах.

Ниже рассмотрены некоторые тонкости.

- Преобразователь может столкнуться с ситуацией, когда недоступно ни одного адреса для серверов имен в списке SLIST, а включенные в этот список серверы в точности совпадают с теми, которые обычно будут использоваться для поиска их же адресов. Такие ситуации возникают обычно в тех случаях, когда склеивающая запись RR имеет меньшее значение TTL, чем в записи NS RR, указывающей передачу полномочий или когда преобразователь кэширует результат поиска NS. Преобразователю следует детектировать такие ситуации и повторять поиск в следующей родительской зоне или в корне.
- Если преобразователь получает от сервера имен информацию об ошибке или иной странной отклик, преобразователю следует удалить этот сервер из списка SLIST и можно также передавать следующие данные по адресу другого сервера-кандидата.

7.3. Обработка откликов

Первым этапом обработки поступившего отклика является его разбор. В процедуру разбора следует включить:

- Проверку обоснованности заголовка и отбрасывание дейтаграмм, содержащих запрос вместо ожидаемого отклика.
- Разбор разделов сообщения и проверку корректности форматирования всех записей RR.
- В качестве необязательного этапа — проверка значений TTL для прибывающих данных на предмет обнаружения RR с чересчур большими значениями TTL. Если RR имеет слишком большое значение TTL (скажем, более 1 недели) отбрасывается весь отклик целиком или для всех TTL в отклике устанавливается значение в 1 неделю.

Следующим шагом является проверка соответствия отклика текущему запросу преобразователя. Рекомендуется сначала проверить соответствие поля ID в заголовке домена, а потом убедиться в соответствии содержимого раздела вопросов желаемой информации. Это требует от алгоритма передачи выделения нескольких битов поля ID для сохранения того или иного идентификатора запроса. Этот этап имеет несколько тонкостей:

- Некоторые серверы имен передают свои отклики не с тех адресов, по которым они получают запросы. Т. е., преобразователь не может рассчитывать на получение ответа с того же адреса, по которому был направлен соответствующий запрос. Такая «ошибка» серверов имен обычно встречается в UNIX-системах.

- Если преобразователь повторно передает некий запрос серверу имен, он должен поддерживать возможность обработки отклика на любой из таких повторов. Однако при использовании отклика для определения времени кругового обхода при обращении к серверу имен преобразователь должен сопоставлять отклики с запросами (и сохранять время передачи для каждого исходящего сообщения) или рассчитывать время кругового обхода только на основе начальной передачи.
- Сервер имен в некоторых случаях может не иметь копии зоны, которая должна у него быть в соответствии с некоторыми записями NS RR. В таких случаях преобразователю следует просто удалять этот сервер из текущего списка SLIST и продолжать работу.

7.4. Использование кэша

В общем случае предполагается, что преобразователь кэширует все данные, которые он получает в откликах, поскольку эти данные могут быть полезны при обработке последующих клиентских запросов. Однако существуют несколько типов данных, которые не следует кэшировать:

- Когда для одного конкретного имени владельца доступны несколько однотипных RR, преобразователю следует кэшировать все такие записи или не кэшировать их совсем. При отсечке отклика, если преобразователь не знает, получил ли он полный набор записей, ему не следует кэшировать набор RR, который может быть неполон.
- Кэшированные данные никогда не следует предпочитать полномочным данным, поэтому кэширование, которое может приводить к возникновению такого предпочтения, не следует выполнять.
- Не следует кэшировать результаты инверсных запросов.
- Не следует кэшировать результаты стандартных запросов, где QNAME содержит метки *, если данные могут использоваться для создания шаблонов. Причина заключается в том, что в кэше не обязательно будут содержаться существующие записи RR или информация о границе зоны, которые потребуются для ограничения использования шаблонных RR.
- Надежность данных RR в отклике вызывает сомнения. Когда преобразователь получает незапрошенные отклики или данные RR, которые отличаются от запрошенных, ему следует отбросить такие данные без их кэширования. Основная идея заключается в том, что для пакета должны быть выполнены все проверки корректности прежде, чем данные из пакета будут кэшированы.

Аналогично в случае, когда преобразователь имеет в отклике множество RR для некоего имени и хочет кэшировать записи RR, ему следует проверить наличие в своем кэше этих RR. В зависимости от обстоятельств выбираются данные из отклика или из кэша, но эти данные ни в коем случае не следует комбинировать. Если данные в отклике относятся к полномочным данным из раздела ответа, всегда следует отдавать предпочтение таким данным.

8. Поддержка электронной почты

DNS определяет стандарт для отображения почтовых ящиков на доменные имена и два метода использования информации о почтовом ящике для получения данных о маршрутизации электронной почты. Стандарт представления почтовых ящиков и привязка ящиков к узлам обмена электронной почтой являются частью официального протокола DNS и рекомендуются в качестве метода маршрутизации электронной почты в Internet. Привязка почтовых ящиков пока остается экспериментальной функцией, которая продолжает развиваться и может быть изменена.

Стандарт представления почтовых ящиков предполагает, что имя почтового ящика имеет форму <локальная-часть>@<почтовый-домен>. Хотя разрешенной для каждой из частей синтаксис может сильно отличаться в разных почтовых системах, рекомендуется использовать синтаксис ARPA Internet, определенный в [RFC-822].

DNS представляет <local-part> в форме одной метки, а <mail-domain> - в форме доменного имени. Метка <local-part> служит префиксом для <mail-domain> и в результате формируется доменное имя, соответствующее почтовому ящику. Таким образом, почтовый ящик HOSTMASTER@SRI-NIC.ARPA отображается на доменное имя HOSTMASTER.SRI-NIC.ARPA. Если метка <local-part> содержит точки или другие специальные символы, для ее представления в первичном файле требуется использовать символ обратной дробной черты (\), чтобы обеспечить корректную интерпретацию доменного имени. Например, почтовый ящик Action.domains@ISI.EDU будет представляться в форме Action\domains.ISI.EDU.

8.1. Привязка почтового обмена

Для привязки почтового обмена используется компонента адреса <mail-domain>, позволяющая определить, куда следует отправлять почту. Компонента <local-part> в этом случае просто не рассматривается. В [RFC-974] приведена детальная спецификация этого метода и указанный документ следует внимательно прочесть прежде, чем реализовать поддержку почтового обмена.

Одним из преимуществ этого метода является независимость именования адресата от хостов, используемых для организации почтовой службы, за счет другого уровня «косвенности» в функции поиска. Однако при добавлении уровня следует устранить осложнения, связанные с представлением символов «%», «!» и т. п. в <local-part>.

Суть этого метода заключается в том, что компонента <mail-domain> используется в качестве доменного имени для поиска записей MX RR, которые указывают хосты, готовые принимать почту для домена <mail-domain>, с уровнем предпочтения для каждого из таких хостов, что позволяет ранжировать их в соответствии с порядком, заданным администраторами для <mail-domain>.

В этом документе ISI.EDU используется в качестве примера значения <mail-domain>, а хосты VENERA.ISI.EDU и VAXA.ISI.EDU служат примерами узлов почтового обмена для домена ISI.EDU. Если почтовая программа имеет сообщение для адресата Москва@ISI.EDU, она будет маршрутизировать это сообщение путем поиска записей MX RR для домена ISI.EDU. В число MX RR для ISI.EDU входят хосты с именами VENERA.ISI.EDU и VAXA.ISI.EDU, запросы типа A позволяют определить адреса этих хостов.

8.2. Привязка почтового ящика (эксперим.)

Для привязки почтового ящика почтовая программа использует полный адрес получателя при определении доменного имени. Найденное для почтового ящика доменное имя используется в качестве значения поля QNAME в запросе QTYPE=MAILB.

Для таких запросов возможно несколько вариантов исхода.

1. Запрос может вернуть сообщение об ошибке в имени, показывающее, что доменного имени для этого почтового ящика не существует.

В долгосрочной перспективе такая ошибка будет говорить об отсутствии указанного почтового ящика. Однако, пока использование почтовой привязки не стало повсеместным, это сообщение об ошибке следует трактовать, как отсутствие поддержки привязки почтовых ящиков в организации, указанной глобальной¹ частью адреса. В таких случаях целесообразно просто вернуться к использованию привязки почтового обмена.

2. Запрос может вернуть запись MR² RR.

Запись MR RR показывает новое имя почтового ящика в поле RDATA. Почтовой программе следует заменить старое имя ящика новым и повторить операцию.

3. Запрос может вернуть запись MB RR.

Запись MB RR содержит доменное имя хоста в своем поле RDATA. Почтовой программе следует доставить сообщение этому хосту с использованием доступного протокола (например, SMTP).

4. Запрос может вернуть одну или множество записей MG³ RR.

Этот результат означает, что почтовый ящик на самом деле относится к списку рассылки или почтовой группе, а не отдельному адресату. Каждая запись MG RR имеет поле RDATA, идентифицирующее почтовый ящик, относящийся к группе. Почтовой программе следует доставить копию сообщения каждому члену группы.

5. Запрос может вернуть запись MB RR, а также одну или множество записей MG RR.

Этот результат означает, что почтовый ящик на самом деле относится к списку рассылки. Почтовая программа может доставить сообщение хосту, указанному записью MB RR, который, в свою очередь, доставит почту членам списка, или использовать записи MG RR для определения всех участников списка.

В любом из этих случаев отклик может включать запись MINFO⁴ RR, которая обычно ассоциируется с почтовой группой, но корректна и для MB. запись MINFO RR идентифицирует два почтовых ящика. Один из них относится к лицу, ответственному за почту для исходного почтового ящика. Этот адрес следует использовать для добавления в почтовые группы и т. п. Второе имя почтового ящика в MINFO RR идентифицирует адрес, по которому следует отправлять сообщения об ошибках при возникновении отказов. Это, в частности, подходит для списков рассылки, где ошибки, связанные с адресами членов списка, следует направлять ответственному за список, а не отправителям сообщений, адресованных в список рассылки.

В будущем для описанных здесь RR могут быть введены новые поля.

9. Литература

- [Dyer 87] S. Dyer, 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, заменяемая DNS, но остающаяся в использовании.
- [Quarterman 86] J. Quarterman, and J. Hoskins, "Notable Computer Networks", Communications of the ACM, October 1986, volume 29, number 10.
- [RFC-742] K. Harrenstien, "NAME/FINGER", RFC 742⁵, Network Information Center, SRI International, December 1977.
- [RFC-768] J. Postel, "User Datagram Protocol", RFC 768⁶, USC/Information Sciences Institute, August 1980.
- [RFC-793] J. Postel, "Transmission Control Protocol", RFC 793⁶, USC/Information Sciences Institute, September 1981.
- [RFC-799] D. Mills, "Internet Name Domains", RFC 799, COMSAT, September 1981.
Вводит иерархическое пространство имен для Internet взамен плоского пространства.
- [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.
Устарел. См. RFC 952.
- [RFC-811] K. Harrenstien, V. White, and E. Feinler, "Hostnames Server", RFC 811, Network Information Center, SRI International, March 1982.
Устарел. См. RFC 953.

¹Доменным именем. *Прим. перев.*

²Mail Rename - переименование почты.

³Mail Group - почтовая группа.

⁴Mail Information - почтовая информация.

⁵Этот документ признан устаревшим и заменен RFC 1194, RFC 1196 и RFC 1288. *Прим. перев.*

⁶Перевод этого документа на русский язык имеется на сайте www.protocols.ru. *Прим. перев.*

- [RFC-812] K. Harrenstien, and V. White, "NICNAME/WHOIS", RFC 812¹, 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.
Начальные наброски по созданию системы доменных имен. Полностью отличаются от современной реализации.
- [RFC-821] J. Postel, "Simple Mail Transfer Protocol", RFC 821², 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.
Начальные наброски по созданию системы доменных имен. Полностью отличаются от современной реализации.
- [RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC 882, USC/Information Sciences Institute, November 1983.
Заменен данным документом.
- [RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC 883, USC/Information Sciences Institute, November 1983.
Заменен данным документом.
- [RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC 920, USC/Information Sciences Institute, October 1984.
Объясняет схему именования для доменов верхнего уровня.
- [RFC-952] K. Harrenstien, M. Stahl, E. Feinler, "DoD Internet Host Table Specification", RFC 952, SRI, October 1985.
Задаёт формат файла HOSTS.TXT - таблицы адресов и имен, заменённой DNS.
- [RFC-953] K. Harrenstien, M. Stahl, E. Feinler, "HOSTNAME Server", RFC 953, SRI, October 1985.
This RFC contains the official specification of the hostname server protocol, which is obsoleted by the DNS. This TCP based protocol accesses information stored in the RFC-952 format, and is used to obtain copies of the host table.
- [RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC 973³, USC/Information Sciences Institute, January 1986.
Описывает изменения документов RFC 882 и RFC 883, а также причины внесения этих изменений.
- [RFC-974] C. Partridge, "Mail routing and the domain system", RFC 974², CSNET CIC BBN Labs, January 1986.
Описывает переход от адресации почты с использованием файлов 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 вместе с RFC 1002 описывает предварительную реализацию сервиса NETBIOS на основе стека протоколов TCP/IP с организацией службы имен NetBIOS на базе 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 1010⁴, USC/Information Sciences Institute, May 1987.
Содержит список номеров сокетов и мнемонику для имен хостов, операционных систем и т. п.
- [RFC-1031] W. Lazeur, "MILNET Name Domain Transition", RFC 1031, November 1987.
Описывает план перехода сети MILNET на использование DNS.
- [RFC-1032] M. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC 1032, November 1987.
Описывает политику регистрации, используемую NIC для администрирования доменов верхнего уровня и делегирования субзон.
- [RFC-1033] M. Lottor, "Domain Administrators Operations Guide", RFC 1033⁵, 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 сети CSNET, независимую от DNS, а также применение DNS в сети CSNET.

¹Этот документ признан устаревшим и заменен RFC 954, который, в свою очередь заменен RFC 3912. *Прим. перев.*

²Этот документ признан устаревшим и заменен RFC 2821, который, в свою очередь заменен RFC 5321. Переводы документов имеются на сайте www.protocols.ru. *Прим. перев.*

³Этот документ признан устаревшим и заменен настоящим документом. *Прим. перев.*

⁴Этот документ неоднократно обновлялся, а в соответствии с RFC 3232 утратил силу совсем. База данных выделенных значений доступна сейчас на сайте www.iana.org. *Прим. перев.*

⁵Перевод этого документа на русский язык имеется на сайте www.protocols.ru. *Прим. перев.*

Предметный указатель

инверсный запрос	11	Hesiod	6, 8	NAME	5, 12	RDATA	6, 12
стандартный запрос	11	HINFO	6	NS	6, 8	RDLENGTH	12
;	2, 15	HS	6, 8	NSCOUNT	12	RDLENGTH	
@	14	ID	11	NSDNAME	8		5
*	6	IN	6, 8	NULL	6, 8	REFRESH	8
<character-string>	6	IN-ADDR.ARPA	8pp., 17	OFFSET	12	RETRY	8
<domain-name>	6pp., 14	IQUERY	11	OPCODE	11	RNAME	8
A	6, 9	MAILA	6, 11	PREFERENCE	8	RR	1
AA	11	MAILB	6, 21	PTR	6, 10	SERIAL	8
ANCOUNT	12	MB	6	QCLASS	6, 11p.	SLIST	18p.
ARCOUNT	12	MD	6, 11	QDCOUNT	12	SOA	6, 8, 14
AXFR	6, 14, 17	MF	6, 11	QNAME	11p.	STATUS	11
CH	6	MG	6	QR	11	TC	11
CLASS	5p., 12	MINFO	6	QTYPE	6, 11p.	TTL	5, 12
CNAME	6, 8, 18	MINIMUM	8	QUERY	11	TXT	6, 9
CS	6	MNAME	8	RA	11	TYPE	5p., 12, 18
EXCHANGE	8	MR	6	RCODE	11	WKS	6, 9
EXPIRE	8	MX	6, 8	RD	11	Z	11

Перевод на русский язык

Николай Малых

nmalykh@protocols.ru