Network Working Group
Request for Comments: 1123

Internet Engineering Task Force
R. Braden, Editor
October 1989

Требования к хостам Internet

Прикладные и служебные протоколы

Requirements for Internet Hosts -- Application and Support

Статус документа

В данном RFC содержатся официальные спецификации для сообщества Internet. Документ содержит ссылки, исправления и дополнения к первичным стандартам протоколов, имеющим отношение к хостам. Документ можно распространять свободно.

Аннотация

Этот документ является одним из двух RFC, посвященных определению и обсуждению требований к программам хостов Internet. Этот документ посвящен прикладным и служебным протоколам, а второй документ RFC 1122 - коммуникационным протоколам канального уровня, IP и транспортного уровня.

1. Введение

Этот документ является одним из пары RFC, определяющих и обсуждающих требования к реализации хост-систем на базе стека протоколов Internet. Документ посвящен протоколам прикладного уровня. Другой документ - RFC 1122 «Требования к хостам Internet - Коммуникационные уровни» [INTRO:1] - посвящен коммуникационным протоколам - канальный уровень, уровень IP (сетевой) и транспортный уровень. С этой парой также тесно связан документ RFC 1009 «Требования к шлюзам[1] Internet» [INTRO:2].

Документ предназначен для производителей и разработчиков, а также для пользователей коммуникационных программ Internet. Документ написан на основе обобщения опыта работы исследователей Internet и компаний-производителей.

В этом RFC рассмотрены стандарты протоколов, которые должны использоваться на хостах, подключенных к сети Internet, а также связанные с ними RFC и другие документы, описывающие текущие варианты спецификаций для таких протоколов. В документе исправлен ряд ошибок, встречающихся в упоминаемых документах, рассмотрены дополнительные вопросы и даны рекомендации по реализации протоколов.

Для каждого протокола в данном документе также приведен явный набор требований, рекомендаций и опций. Читатель должен понимать, что список приведенных в документе требований неполон - полные требования для хостов Internet содержатся в документах, определяющих спецификации протоколов. В данном RFC приведены поправки, уточнения и дополнения к стандартам протоколов.

Качественная реализация протоколов, подготовленная в результате внимательного прочтения этого RFC, работы в контакте с техническим сообществом Internet и учета позитивной практики разработки коммуникационных программ, не должна сильно отличаться от содержащихся в документе требований. Многие из требований данного RFC уже реализованы в стандартах или рассматриваются для включения в стандарты, поэтому их обсуждение в данном документе может показаться избыточным. Однако такие вопросы были включены в документ, поскольку некоторые из имеющихся реализаций содержат ошибки, приводящие к недостаточной интероперабельности, снижению производительности и возникновению иных проблем.

В документе обсуждается и разъясняется множество требований и рекомендаций. Ограничиваться простым списком требований было бы опасно по ряду причин:

- некоторые требуемые функции более важны, а ряд функций не относится к числу обязательных;
- существует множество причин, по которым продукция, разработанная для ограниченного контекста, может быть выбрана для использования в иных средах.

Нужно следовать приведенным в документе спецификациям для обеспечения интероперабельности хостов при взаимодействии через разнородные и сложные системы Internet. Хотя многие из числа имеющихся реализаций не соответствуют тем или иным требованиям, спецификации являются идеалом, к которому нужно стремиться.

Приведенные в документе требования основаны на современном уровне архитектуры Internet. Документ будет обновляться по мере развития спецификаций и технологий в тех областях, с которыми связан этот документ.

Вводная часть документа начинается с краткого обзора архитектуры Internet применительно к хостам, приведены также некоторые рекомендации по выбору программ для хостов. Кроме того, во введении даны некоторые рекомендации по работе с остальной частью документа и рассмотрена используемая терминология. Глава 2 описывает общие требования ко всем прикладным протоколам, главы 3 - 5 описывают требования к протоколам трех основных приложений - Telnet, обмена файлами (FTP) и электронной почты. Глава 6 посвящена служебным приложениям - DNS, инициализация и управление. В главе 7 приведены ссылки на другие источники информации.

1.1 Архитектура Internet

Краткое описание архитектуры Internet с точки зрения хостов приведено в параграфе 1.1 работы [INTRO:1]. Там же можно найти ссылки на другие публикации по архитектуре Internet.

1.2 Общие вопросы

Существует два важных аспекта, которые должны принимать во внимание все разработчики программ для хостов Internet, - постоянное развитие Сети и принцип устойчивости.

1.2.1 Постоянное развитие Internet

Непредсказуемо быстрый рост Internet порождает проблемы управления и масштабирования в гигантских системах передачи дейтаграмм. В результате решения таких проблем изменяются и спецификации, описываемые в данном документе. Изменения планируются и осуществляются под контролем с участием производителей сетевой продукции и организаций, ответственных за работу сетей.

Обновление и постоянное совершенствование являются неотъемлемыми чертами современных сетевых протоколов, хотя еще несколько лет назад такую ситуацию было трудно представить. Разработчики коммуникационных программ для стека протоколов Internet (или иных протоколов) должны поддерживать и обновлять свои программы с учетом изменяющихся спецификаций для того, чтобы не оставить в беде несчастных пользователей. Internet представляет собой большую коммуникационную сеть и пользователи постоянно контактируют с этой сетью. Опыт и знания разработчиков, реализованные в их программах, за короткое время становятся достоянием технического сообщества Internet.

1.2.2 Принцип устойчивости

Для протоколов всех уровней существует общее правило, которому приложения должны следовать во избежание проблем с устойчивостью и интероперабельностью: «будьте либеральны при приеме данных и консервативны при их передаче».

[1] Маршрутизаторам в современной терминологии.

Программы должны уметь обрабатывать все мыслимые ошибки; не имеет значения вероятность возникновения той или иной ошибки - раньше или позже пакет с любой возможной комбинацией ошибок и/или атрибутов будет получен и программа должна быть готова к обработке такого пакета. Если программа не может эффективно обрабатывать ошибки, она приведет прямой дорогой к хаосу. В общем случае лучше предположить, что сеть наводнена зловредными объектами, которые постоянно передают пакеты, предназначенные для нанесения максимального вреда. Такое предположение обеспечит высокий уровень защиты. Наиболее серьезные проблемы в Internet связаны с неисследованными механизмами, включающимися с малой вероятностью; намерения обычных злоумышленников никогда не могут принести такого вреда!

На всех уровнях программ хостов Internet должны быть реализованы средства адаптации. В качестве простого примера рассмотрим спецификацию протокола, использующего численные значения для какого-либо поля в заголовке (например, тип, номер порта или код ошибки) - при разработке следует предполагать, что используемая нумерация неполна. Если спецификация протокола предполагает четыре возможных кода ошибки, приложение должно уметь обрабатывать по крайней мере пять типов ошибок (4 заданных и все остальные). Появление не определенных в спецификации кодов должно протоколироваться (см. ниже), но не должно нарушать работу системы.

Почти так же важен еще один аспект - программы на других хостах могут содержать определения, которые могут толкнуть хост на неразумные, но допустимые с точки зрения протокола действия. Естественным решением будет «поиск неразумных хостов» - программы хоста должны быть готовы не просто переносить появление неразумных хостов, но и участвовать в процессе ограничения вредных воздействий, которые подобные хосты могут нанести сетевым объектам общего пользования.

1.2.3 Протоколирование ошибок

Сеть Internet включает огромное количество разнотипных хостов и шлюзов, в каждом из которых реализовано множество протоколов и уровней - некоторые из хостов и маршрутизаторов наверняка содержат ошибки в программах стека протоколов Internet. В результате сложности, разнотипности и использования распределенных функций диагностика Internet является весьма непростой задачей.

Упростить поиск проблем помогает использование хостами специальных средств протоколирования ошибок и странностей в поведении протоколов. При записи таких событий важно включать максимум диагностической информации. Часто бывает весьма полезно записывать заголовки пакетов, вызывающих ошибки. Однако следует знать меру - средства протоколирования ошибок не должны отнимать слишком много ресурсов хоста и снижать эффективность его работы.

При записи информации об ошибках существует вероятность получения журнальных файлов огромного размера - таких ситуаций можно избежать, используя «циклический» журнал или включая запись только для диагностики определенных сбоев. Полезно также фильтровать и считать повторяющиеся последовательные сообщения. Представляется привлекательной приведенная ниже стратегия:

(1) всегда считать аномальные события и делать содержимое счетчиков доступным с помощью средств управления сетью (см. параграф 6.3);

(2) поддерживать возможность выбора протоколируемых событий (например, записывать все ошибки, связанные с хостом X).

Отметим, что разные системы управления сетью могут придерживаться различных правил в части числа протоколируемых ошибок для каждого хоста. Некоторые системы исходят из принципа: «если это не имеет ко мне прямого отношения, не хочу об этом знать», тогда, как другие системы прилагают максимум усилий для обнаружения и устранения аномалий в поведении сетевых протоколов.

1.2.4 Конфигурационные параметры

Идеальной реализацией хоста будет та, на которой настройка стека протоколов Internet будет полностью автоматизирована (self-configuring). Это позволит реализовать весь стек в ПЗУ[1] или встроить в микросхемы оборудования - такое решение упростит организацию бездисковых станций, снимет значительную часть нагрузки с сетевых администраторов и упростит разработку систем. Однако этот идеал недостижим и на практике к нему не удается даже серьезно приблизиться.

В этом документе вы будете часто встречать требования, параметры которых являются опциями настройки. Существует несколько причин появления таких требований. В некоторых случаях это обусловлено отсутствием согласия по вопросу выбора наилучшего значения и в будущем может потребоваться установка иного значения параметра. В других случаях значение параметра может зависеть от внешних факторов (например, скорость и топология соседних сетей) и алгоритмы автоматической настройки могут отсутствовать или не обеспечивать полной настройки параметров. Причиной использования таких параметров могут послужить и административные требования.

Наконец, часть конфигурационных опций может потребоваться для обеспечения взаимодействия с устаревшими или содержащими ошибки реализациями протоколов, распространяемыми без исходных текстов (к несчастью, такое встречается в Internet достаточно часто). Для обеспечения корректного сосуществования с такими «сбойными» системами администраторам зачастую приходится «расстраивать» корректно работающие системы. Такие проблемы будут решаться сами собой по мере удаления сбойных систем, но разработчики не должны оставлять этот вопрос без внимания.

Когда мы говорим, что параметр требует настройки, это не означает требования читать значение параметра из конфигурационного файла при каждой загрузке. Мы рекомендуем разработчикам устанавливать для таких параметров используемые по умолчанию значения - тогда в конфигурационных файлах могут содержаться лишь те значения, которые не совпадают с принятыми по умолчанию. Таким образом, конфигурационные требования обеспечивают гарантию **возможности изменения** принятых по умолчанию значений параметров - это решение можно использовать даже при загрузке программного кода из ПЗУ.

В этом документе для ряда случаев указаны значения, которые следует использовать по умолчанию. Выбор принятых по умолчанию значений является достаточно важным вопросом для случаев использования вместе с существующими

[1]Постоянное запоминающее устройство. *Прим. перев.*

сбойными системами. По мере продвижения Internet к полной интероперабельности, принятые по умолчанию значения будут реализовать официальный протокол, а не «расстраивать» систему для обеспечения совместимости с плохо работающими реализациями. Хотя рынок заставляет некоторых производителей устанавливать по умолчанию значения для «расстройки», мы рекомендуем использовать по умолчанию значения, соответствующие стандартам.

В заключение отметим, что производители продукции должны предоставлять полную документацию по всем конфигурационным параметрам, указывая допустимые значения и описывая влияние параметров на работу системы.

1.3 Работа с документом

1.3.1 Структура документа

В общем случае все главы документа организованы в форме следующих параграфов:

(1) Введение

(2) Общие вопросы - рассматриваются документы со спецификациями протокола, приводятся исправления ошибок, перечисляются требования, которые могли быть нечетко или неточно сформулированы и приводятся дополнительные комментарии и разъяснения.

(3) Частные вопросы - рассматривается устройство протокола и вопросы реализации, не включенные в предыдущий раздел.

(4) Интерфейсы - обсуждаются услуги, предоставляемые вышележащему протоколу.

(5) Заключение - резюмируются требования данной главы.

Во многих темах документа встречаются параграфы, помеченные как **Обсуждение** или **Реализация**. Этот материал предназначен для разъяснения требований, рассмотренных ранее в документе. Такие параграфы также включают некоторые предложения по вариантам будущих разработок. Материалы о реализации содержат предложения, которые могут быть интересны для разработчиков.

Заключение служит в качестве краткого руководства и справочника к главе, но оно слишком кратко для использования в качестве информационного источника. Такие заключения никогда не должны использоваться отдельно от текста полного RFC.

1.3.2 Уровни требований

В этом документе модальные глаголы, служащие для формулировки требований, выделены жирным шрифтом и используются в оговоренном ниже смысле:

Требуется, должен (MUST)

Эти глаголы (или производные от них слова) используются для обозначения абсолютной необходимости.

Следует, рекомендуется (SHOULD)

Этот глагол (или причастие **рекомендуемый**) служит для обозначения тех случаев, когда могут существовать дополнительные факторы, позволяющие игнорировать данное требование; в таких случаях рекомендуется принимать взвешенное решение с учетом всех факторов.

Можно (MAY)

Этот глагол и прилагательное **необязательный** (опциональный) означает, что вопрос реализации требования отдается на откуп разработчику. Кто-то может реализовать требование с целью расширения возможностей, а другой разработчик может отказаться от реализации в целях упрощения.

Реализация считается несовместимой со стандартами, если в ней не выполнены обязательные требования (MUST). Реализация, в которой выполнены все требования MUST и SHOULD, считается безусловно совместимой. Если же выполняются все требования MUST, но проигнорирована часть требований SHOULD, реализация считается условно совместимой.

1.3.3 Терминология

Ряд используемых в документе технических терминов определен в данном параграфе.

Сегмент - Segment
Сегментом будем называть модуль[1] данных, используемый для сквозной передачи по протоколу TCP. Сегмент состоит из заголовка TCP, за которым следуют данные. Сегменты передаются с использованием инкапсуляции в дейтаграммы IP.

Сообщение - Message
Этот термин используется некоторыми протоколами прикладного уровня (в частности, SMTP) для обозначения модулей данных приложений.

Дейтаграмма - Datahram
Дейтаграмма [UDP] представляет собой модуль данных, используемый для передачи по протоколу UDP.

Многодомный - Multihomed
Хост называют многодомным (multihomed) если он имеет более одного адреса IP.

1.4 Благодарности

Этот документ включает результаты работы и комментарии большой группы специалистов по протоколам Internet, включая представителей университетов и исследовательских лабораторий, компаний-производителей и государственных агентств. Подготовка документа велась в рабочей группе IETF Host Requirements.

[1]Термином модуль (блок) будем обозначать неделимую при передаче на данном уровне совокупность данных. *Прим. перев.*

Редактор выражает особую благодарность за неустанную работу людям, принявшим участие в долгих конференциях и написавшим 3 мегабайта почтовых сообщений за 18 месяцев подготовки этого документа: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), James Van Bokkelen (FTP Software).

Кроме того, выражается благодарность всем, кто внес большой вклад в подготовку документа: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), Mike StJohns (DCA). Перечисленные ниже люди внесли значительный вклад в подготовку отдельных тем: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), Rayan Zachariassen (Toronto).

Благодарим также всех, кто так или иначе способствовал появлению этого документа.

2. Общие вопросы

В этом разделе рассматриваются требования, применимые ко всем протоколам прикладного уровня.

2.1 Имена хостов и числовые адреса

Синтаксис имен хостов Internet описан в RFC 952 [DNS:4]. Однако одно из требований к именам изменено настоящим документом - в качестве первого символа имени может использоваться буква латиницы или цифра[1]. Программы хостов **должны** следовать более либеральному требованию настоящего документа.

Программы хостов **должны** поддерживать имена длиной до 63 символов и **следует** поддерживать имена размером до 255 символов.

Следует обеспечивать возможность идентификации хостов с помощью (1) доменного имени хоста или (2) IP-адреса в десятичном формате с разделением точками (#.#.#.#). Для хостов **следует** синтаксически проверять введенный IP-адрес до обращения к DNS.

Обсуждение

Последнее требование не означает полную проверку синтаксиса введенного адреса - этот вопрос должен решаться на уровне пользовательского интерфейса. Например, адреса IP должны указываться в квадратных скобках [ ] для почтовых программ SMTP (см. 5.2.17). Такое обозначение может быть сделано универсальным в масштабе хоста, что позволяет упростить синтаксическую проверку вводимых адресов.

Если адреса можно указывать без ограничителей такого типа, требуется выполнять их полную синтаксическую проверку, поскольку имя хоста может начинаться с цифр и даже содержать только цифры (см. параграф 6.1.2.4). Однако корректное имя хоста никогда не может иметь вид корректного адреса IP в формате #.#.#.#, поскольку по крайней мере компонента (домен) верхнего уровня должна быть буквенной.

2.2 Использование службы доменных имен

Доменные имена хостов **должны** транслироваться в IP-адреса в соответствии с требованиями параграфа 6.1.

Приложения, использующие службу доменных имен, **должны** быть способны справиться с незначительными ошибками. Приложение **должно** ожидать в течение разумного периода перед выполнением повтора при некритичных ошибках. Кроме того, приложения **должны** допускать возможность того, что в результате сетевых проблем службы могут быть недоступными в течение часов и даже дней.

Для приложений **не следует** основываться на возможности обнаружения записи WKS, содержащей точный список всех служб для отдельного хоста, поскольку тип WKS RR нечасто используется на хостах Internet. Для проверки доступности сервиса нужно просто попытаться его использовать.

2.3 Приложения на многодомных хостах

Если удаленный хост является многодомным, трансляция имени будет возвращать список адресов IP. Как сказано в параграфе 6.1.3.4, этот список должен быть упорядочен по уменьшению приоритета адресов. Для реализаций прикладных протоколов **следует** быть готовым к попыткам использования нескольких адресов из полученного списка для достижения успеха. Более специфические требования для протокола SMTP рассмотрены в параграфе 5.3.4.

Когда локальный хост является многодомным, приложениям на основе запросов/откликов UDP **следует** передавать отклик с IP-адресом отправителя, который совпадает с адресом получателя в дейтаграмме запроса UDP. Термин «заданный адрес получателя» определен в параграфе 3.2.1.3 документа RFC 1122 [INTRO:1].

Подобно этому, серверным приложениям, открывающим множество соединений TCP с одним клиентом, **следует** использовать один локальный адрес IP для всех таких соединений.

2.4 Тип обслуживания

Приложение **должно** выбирать приемлемые значения TOS при обращении к службам транспортного уровня и эти значения **должны** быть настраиваемыми. Отметим, что значение TOS содержит 5 битов, из которых используются только три старших бита, а два оставшихся **должны** иметь нулевое значение.

[1] RFC 952 разрешает для первого символа только буквы английского алфавита. *Прим. перев.*

Обсуждение

По мере разработки алгоритмов поддержки TOS маршрутизаторами рекомендуемые значения типа обслуживания для различных прикладных протоколов могут изменяться. Кроме того, очевидно, что для различных комбинаций пользователей и путей Internet могут возникать потребности в использовании нестандартных значений TOS. Исходя из этих соображений, значения TOS должны быть настраиваемыми.

Рекомендуемые значения TOS для различных прикладных протоколов можно найти в документе «Assigned Numbers» [INTRO:5].

2.5 Требования общего плана

Функция	*Параграф*	*Требования*
Пользовательские интерфейсы:		
Имена хостов могут начинаться с цифры	2.1	Обязательно
Поддержка имен размером до 63[1] символов	2.1	Обязательно
Поддержка имен размером до 255 символов	2.1	Следует
Поддержка IP-адресов в десятичном формате	2.1	Следует
Выполнение сначала проверки синтаксиса для IP-адресов	2.1	Следует
Преобразование доменных имен в соответствии с параграфом 6.2	2.2	Обязательно
Устойчивость к некритичным ошибкам DNS	2.2	Обязательно
Повторение с разумным интервалом	2.2	Обязательно
Допустимость продолжительного отсутствия сервиса	2.2	Обязательно
Предположение о доступности записей WKS	2.2	Не следует
Попытки использования. различных. адресов для многодомного удаленного хоста	2.3	Следует
Адрес отправителя отклика UDP соответствует. адресу получателя запроса	2.3	Следует
Использование одного адреса IP для связанных соединений TCP	2.3	Следует
Задание приемлемых значений TOS	2.4	Обязательно
Возможность выбора TOS	2.4	Обязательно
Неиспользуемые биты (младшие 2 бита) равны 0	2.4	Обязательно

3. Удаленный доступ по протоколу TELNET

3.1 Введение

Telnet представляет собой стандартный протокол Internet для удаленного доступа в систему. Протокол обеспечивает правила кодирования для «связывания» клавиатуры и монитора на клиентском компьютере с командным интерпретатором удаленного сервера. Часть протокола Telnet включена также в другие протоколы прикладного уровня (например, FTP и SMTP).

Telnet использует одно соединение TCP и его нормальный поток данных (режим виртуального сетевого терминала - Network Virtual Terminal или NVT) представляет 7-битовые символы ASCII с escape-последовательностями, служащими для управления. Протокол Telnet обеспечивает возможность согласования множества дополнительных режимов и функций.

Основная спецификация Telnet содержится в RFC 854 [TELNET:1], а опции определены во множестве других RFC, перечисленных в разделе 7.

3.2 Общие вопросы

3.2.1 Согласование опций: RFC 854, стр. 2-3

Каждая реализация Telnet **должна** включать опцию согласования параметров с дополнительными механизмами [TELNET:2].

Хост **должен** аккуратно выполнять требования RFC 854, чтобы избежать возникновения петель при согласовании опций. Хост **должен** отказываться (т. е., давать отклик WONT/DONT на запросы DO/WILL) от использования не поддерживаемых им опций. **Следует** сохранять возможность согласования опций (даже при отказе от всех запросов) в течение всего срока существования соединения Telnet.

Если согласовать опции не удалось, реализация Telnet **должна** перейти в используемый по умолчанию режим NVT.

Обсуждение

Хотя поддержка более изощренных терминалов и опций стала нормой, люба реализация должна быть готова к поддержке режима NVT для любого соединения пользователя с сервером.

3.2.2 Функция Telnet Go-Ahead: RFC 854, стр. 5, RFC 858

На хосте, который никогда не передает команду Telnet GA[2], сервер Telnet **должен** попытаться согласовать опцию подавления этой команды - Suppress Go Ahead (т. е., передать WILL Suppress Go Ahead). Клиент и сервер Telnet **должны** всегда принимать согласование опции Suppress Go Ahead (подавление команды «идите прочь»).

При использовании режима полнодуплексного терминала, для которого GA не имеет смысла, реализация клиента Telnet **может** игнорировать команды GA.

Обсуждение

Полудуплексные (locked-keyboard) терминалы line-at-a-time (строка в один прием), для которых был разработан механизм Go-Ahead, сегодня уже практически не используются. Во многих операционных системах достаточно сложно реализовать передачу сигналов Go-Ahead, даже если эта ОС поддерживает полудуплексные терминалы. Эти трудности обычно связаны с тем, что программный код сервера Telnet не имеет доступа к информации о

[1] В исходном документе ошибочно указан размер 635. *Прим. перев.*
[2] Go Ahead - идите прочь.

блокировке пользовательского процесса в ожидании данных от соединения Telnet (т. е. невозможно надежно определить момент передачи команды GA). Следовательно, большинство хостов с серверами Telnet не поддерживает команду GA.

Смысл приведенных в этом параграфе правил состоит в том, что следует разрешать другой стороне соединения Telnet блокировать использование команд GA.

Существует класс полудуплексных терминалов, до сих пор находящихся в эксплуатации (терминалы ввода данных), которые работают в полноэкранном режиме. Однако поддержка таких терминалов при использовании ими протокола Telnet не требует сигналов Go Ahead (см. 3.3.2).

3.2.3 Функции управления: RFC 854, стр. 7-8

Список команд Telnet был расширен для включения команды EOR[1] с кодом 239 [TELNET:9].

Клиент и сервер Telnet **могут** поддерживать функции управления EOR, EC, EL, Break и **должны** поддерживать функции AO, AYT, DM, IP, NOP, SB, SE.

Хост **должен** быть способен принимать и игнорировать любые функции управления Telnet, которые он не поддерживает.

Обсуждение

Отметим, что сервер Telnet должен поддерживать функцию Telnet IP (Interrupt Process - прерывание), даже при наличии на хосте эквивалента этой функции (например, комбинация клавиш Control-C во многих системах). Функция Telnet IP может быть сильнее такой команды прерывания, поскольку использует срочные данные TCP.

Управляющая функция EOR может использоваться для задания границ потока. Важным применением этой функции является поддержка терминалов ввода данных (см; 3.3.2). Обратим внимание, что команда EOR не была определена в RFC 854, поэтому хост, не способный корректно игнорировать неизвестные команды Telnet, может «рухнуть» при получении EOR. Для защиты таких хостов была введена опция End-of-Record [TELNET:9], однако для корректно реализованных программ Telnet такая защита не требуется.

3.2.4 Сигнал Telnet Synch: RFC 854, стр. 8-10

При получении срочных данных TCP клиент или сервер Telnet **должен** отвергать все данные, за исключением команд Telnet, пока не будет получен сигнал DM (и завершатся срочные данные).

При передаче Telnet IP (Interrupt Process) клиенту Telnet **следует** сопровождать такое прерывание последовательностью Telnet Synch (т. е. передавать как срочные данные TCP последовательность IAC IP IAC DM). Указатель срочности TCP должен указывать на октет DM.

При получении команд Telnet IP сервер Telnet **может** передать последовательность Telnet Synch клиенту для отсечения выходного потока. Выбор решения связан с реакцией операционной системы сервера на локальное прерывание процесса пользователем.

При получении команды Telnet AO сервер Telnet **должен** передать пользователю последовательность Telnet Synch для отсечения выходного потока.

Клиенту Telnet **следует** поддерживать возможность отсечения вывода при передаче Telnet IP (см. также 3.4.5).

Обсуждение

Для клиента Telnet существует три способа отсечения выходного потока данных сервера:

(1) Передать команду AO после IP.

В результате хост сервера будет передавать сигнал flush-buffered-output (отбросить буферизованный вывод) своей операционной системе. Однако команда AO может не дать локального эффекта (т. е., не остановить терминальный вывод на стороне клиента Telnet), пока сервер Telnet получает и обрабатывает команду AO и передает обратно Synch.

(2) Передать DO TIMING-MARK [TELNET:7] после IP и локально отвергать весь вывод, пока не будет получена последовательность WILL/WONT TIMING-MARK от сервера Telnet.

Поскольку DO TIMING-MARK обрабатывается на сервере после IP, отклик на эту команду должен находиться в корректном месте выходного потока данных. Однако TIMING-MARK не будет посылать операционной системе сервера сигнал flush buffered output. Необходимость такого сигнала определяется операционной системой сервера Telnet.

(3) Использовать оба способа.

Однозначно выбрать лучший из двух вариантов нельзя, поскольку требуется адаптация к различным хостам с серверами Telnet, которые могут не соответствовать стандартам Telnet в различных вопросах. Наиболее безопасным решением будет поддержка выбираемой пользователем опции (1), (2) или (3).

3.2.5 Принтер и клавиатура в режиме NVT: RFC 854, стр. 11

В режиме NVT серверу Telnet **не следует** передавать символы с установленным старшим битом[2] и **недопустимо** использовать этот бит для контроля четности. Реализациям, передающим старший бит пользователю, **следует** согласовывать опцию бинарного[3] режима (см. 3.2.6).

Обсуждение

[1]End-of-Record - конец записи.
[2]Коды ASCII > 127. *Прим. перев.*
[3]Т. е. не текстового. *Прим. перев.*

Разработчики должны помнить, что RFC 854 позволяет клиентам и серверам, ожидающим NVT ASCII, игнорировать символы с установленным старшим битом. В общем случае предполагается использование бинарного режима для передачи расширенного (за пределы 7 битов) набора символов программам Telnet.

Однако, существуют приложения, которым реально требуется 8-битовое расширение режима NVT, не определенное в настоящее время, и такие приложения устанавливают старший бит на все или часть времени существования соединения Telnet. Отметим, что бинарный режим не совпадает с 8-битовым расширением NVT, поскольку в бинарном режиме отключается обработка символов завершения строки. С учетом сказанного требования к использованию старшего бита, включают глагол **следует**, а не **требуется**.

RFC 854 определяет минимальный набор свойств виртуального сетевого терминала (NVT); это не означает запрета на поддержку дополнительных функций в реальных терминалах. Соединения Telnet полностью прозрачны для всех 7-битовых символов ASCII, включая дополнительные коды управления ASCII.

Например, терминал может поддерживать полноэкранные команды, кодируемые как escape-последовательности ASCII; реализация Telnet будет передавать такие последовательности как неинтерпретируемые данные. Таким образом, режим NVT не следует трактовать, как терминал с сильно ограниченными возможностями.

3.2.6 Структура команд Telnet: RFC 854, стр. 13

Поскольку опции могут появляться в любой части потока данных, escape-символ Telnet (известный: как IAC, со значением 255), передаваемый как данные, **должен** дублироваться.

3.2.7 Опция Telnet Binary: RFC 856

При успешном согласовании опции Binary, разрешается использование 8-битовых символов. Однако, в потоке данных по-прежнему **должны** просматриваться символы IAC, **должны** выполняться все встроенные команды Telnet, а символы IAC в качестве данных **должны** дублироваться. Обработка других символов (например, замена CR на CR NUL или CR LF) **недопустима**. В частности, для бинарного режима не действует соглашение end-of-line (конец строки), обсуждаемое в параграфе 3.3.1.

Обсуждение

Опция Binary обычно согласуется для обоих направлений, чтобы перевести соединение Telnet из режима NVT в двоичный режим.

Последовательность IAC EOR может использоваться для обозначения границ блоков данных в бинарном потоке Telnet.

3.2.8 Опция типа терминала Telnet: RFC 1091

Опция Terminal-Type **должна** использовать названия типов терминалов, официально определенные в документе Assigned Numbers [INTRO:5], когда такие имена существуют для применяемых терминалов. Однако приниматься **должны** любые значения опции Terminal-Type.

Обсуждение

RFC 1091 [TELNET:10] содержит обновленное (по сравнению с RFC 930) определение опции Terminal-Type. Прежняя версия позволяла хосту сервера поддерживать множество типов терминалов для определения типа клиентского терминала на основе предположения, что каждый физический терминал имеет собственный тип. Однако сегодня терминал в большинстве случаев является на самом деле программой эмуляции терминала на базе ПК, которая зачастую может поддерживать различные типы терминалов. Следовательно, RFC 1091 расширяет спецификацию, позволяя более общее согласование типа терминала между клиентом и сервером Telnet.

3.3 Частные вопросы

3.3.1 Соглашение Telnet о завершении строки

Протокол Telnet определяет последовательность символов CR LF в качестве сигнала завершения строки. Для терминального ввода это соответствует завершению команды или нажатию клавиши перевода строки на пользовательском терминале (на терминалах ASCII это клавиша CR, которая может также называться Return или Enter).

Когда сервер Telnet принимает сигнал завершения строки CR LF, как ввод с удаленного терминала, эффект **должен** быть таким же, как от нажатия клавиши перевода строки на локальном терминале. На хостах, использующих ASCII, в частности, получение сервером Telnet последовательности CR LF должно сопровождаться таким же эффектом, как нажатие клавиши CR на локальном терминале. Таким образом, последовательности CR LF и CR NUL **должны** иметь одинаковый эффект на серверных хостах ASCII при получении ввода от соединений Telnet.

Клиент Telnet **должен** обеспечивать возможность передачи любой из последовательностей CR LF, CR NUL и LF. Клиентам Telnet на хостах ASCII **следует** обеспечивать пользователю возможность выбора последовательности CR LF или CR NUL при нажатии клавиши перевода строки (по умолчанию следует использовать последовательность CR LF).

Последовательность завершения строки Telnet CR LF **должна** использоваться для передачи данных Telnet, которые не относятся к типу «терминал-хост» (например, при передаче вывода от сервера Telnet или встраивании других прикладных протоколов в Telnet).

Обсуждение

Для обеспечения интероперабельности между различными серверами и клиентами Telnet в протоколе Telnet определено стандартное представление сигнала завершения строки. Поскольку ASCII не включает в явном виде символа завершения строки, могут использоваться различные варианты, в зависимости от системы (например, CR, LF, CR LF). В качестве стандартной протокол Telnet определяет последовательность CR LF.

К сожалению, спецификация протокола Telnet в RFC 854 [TELNET:1] не указывает однозначно символов, которые должны передаваться от клиента к серверу при нажатии клавиши перевода строки. В результате отсутствия однозначности постоянно возникают проблемы с интероперабельностью, порождаемой различными недостаточно корректными реализациями клиентов и серверов Telnet.

Хотя протокол Telnet основан на симметричной модели взаимодействия, в сеансах удаленного доступа роли пользовательского терминала и серверного хоста различаются. Например, RFC 854 определяет CR, LF и CR LF как выходные последовательности сервера, но не задает, какие символы должны передаваться со стороны клиента Telnet при нажатии на терминале клавиши завершения строки.

При нажатии пользователем клавиши завершения строки некоторые реализации клиентов Telnet передают последовательность CR LF, а другие - CR NUL (в результате различной интерпретации одного и того же предложения в тексте RFC 854). Эти последовательности будут эквиваленты для корректно реализованных серверов на хостах ASCII, как было показано выше. Для других серверов требуется выбор режима в клиентской реализации Telnet.

Существование клиентов Telnet, которые передают только CR NUL при нажатии клавиши CR, порождает дилемму для хостов, не поддерживающих ASCII - трактовать CR NUL на входе как эквивалент CR LF, препятствуя возможности ввода только CR, или полностью терять связь с сетью.

Предположим, что пользователь на хосте A применяет Telnet для доступа к серверу на хосте B и запуска на этом хосте (B) другого клиента Telnet для работы с сервером на хосте C. Для комбинации клиент/сервер Telnet на хосте B желательно обеспечить прозрачность (т. е. для хоста A такое подключение должно выглядеть, как прямое соединение с сервером C). В частности, корректная реализация будет обеспечивать прозрачность для Telnet-последовательностей завершения строки (за исключением преобразования CR LF в CR NUL и обратно).

Реализация

Чтобы разобраться с вопросами трактовки завершения строки в Telnet, нужно понять по крайней мере общую модель взаимодействия Telnet с локальной ОС. Серверный процесс Telnet обычно встроен в терминальный драйвер операционной системы как псевдотерминал. Последовательность завершения строки, принимаемая сервером Telnet, должна давать такой же эффект, как нажатие клавиши перевода строки на локально подключенном терминале.

Операционные системы, поддерживающие интерактивные приложения с посимвольным вводом (например, текстовые редакторы), обычно поддерживают два внутренних режима для своих терминалов ввода-вывода - форматированный режим, при котором к потоку данных применяется соглашение о завершении строки и другие правила форматирования, и режим необработанного текста (raw), при котором приложение имеет прямой доступ к символу сразу после его ввода. Серверы Telnet должны быть реализованы таким образом, чтобы оба режима имели одинаковый эффект для локальных и удаленных терминалов. Для примера предположим, что последовательность CR LF или CR NUL принимается сервером Telnet на хосте ASCII. В режиме raw передается приложению символ CR, а в форматированном режиме используется локальное соглашение о символах завершения строки.

3.3.2 Терминалы ввода данных

Обсуждение

В дополнение к строковым и символьным терминалам ASCII, для которых был разработан протокол Telnet, существует еще несколько семейств видео-терминалов, которые называют терминалами ввода данных или DET[1]. Семейство IBM 3270 является широко известным примером такого типа терминалов.

Для поддержки базовых типов DET были разработаны два протокола Internet - SUPDUP [TELNET:16, TELNET:17] и опция DET [TELNET:18, TELNET:19]. Опция DET обслуживает терминалы ввода данных через соединения Telnet с использованием (суб)согласования. SUPDUP представляет собой самостоятельный терминальный протокол, который может быть введен из Telnet путем согласования. Хотя протокол SUPDUP и опцию DET можно успешно использовать в отдельных типах сред, ни один из этих вариантов не обеспечивает универсальности.

Другим вариантом использования DET является реализация поддержки семейства терминалов IBM 3270 с помощью Telnet (это применимо к любым терминалам DET). Идея состоит в создании режима native DET, в котором оригинальные потоки ввода-вывода DET передаются, как двоичные данные. Команда Telnet EOR используется в этом случае для обозначения границ логических записей (например, «экранов») в двоичном потоке.

Реализация

Для активизации и отключения режима native DET используются следующие правила:

- сервер использует опцию Terminal-Type [TELNET:10] для определения принадлежности клиента к DET;
- обе стороны согласуют опцию EOR [TELNET:9] (это общепринято, но не обязательно);
- обе стороны согласуют опцию Binary [TELNET:3] для перехода в режим native DET;
- когда одна из сторон выходит из бинарного режима, другая сторона также должна сделать это, вернувшись в режим NVT.

3.3.3 Поддержка опций

Каждая реализация Telnet **должна** поддерживать опции Binary [TELNET:3] и Suppress Go Ahead [TELNET:5]; **следует** также поддерживать опции Echo [TELNET:4], Status [TELNET:6], End-of-Record [TELNET:9] и Extended Options List [TELNET:8].

Для клиентов и серверов Telnet **следует** поддерживать опцию Window Size [TELNET:12], если локальная ОС обеспечивает соответствующие возможности.

[1]Data entry terminal.

Обсуждение

Отметим, что опция End-of-Record (конец записи) означает лишь, возможность Telnet получать Telnet EOR без краха; следовательно, каждый модуль Telnet может попытаться согласовать опцию End-of-Record (см. 3.2.3).

3.3.4 Инициирование согласования опций

При использовании протокола Telnet в режиме клиент-сервер для сервера **следует** инициировать согласование режима взаимодействия с терминалом, который ожидает сервер.

Обсуждение

Протокол Telnet был определен как симметричный, но его реальное применение обычно бывает асимметричным. Удаленный вход в систему приводит к краху, если ни одна из сторон не инициирует согласования требуемых ей опций, которые не установлены по умолчанию. Обычно сервер определяет предпочтительный режим, поэтому сервер и должен инициировать согласование опций. Поскольку процесс согласования является симметричным, клиент также может быть инициатором.

Клиентам Telnet **следует** предоставлять пользователю возможность разрешить или запретить инициативу клиента по согласованию опций.

Обсуждение

Пользователям иногда требуется подключение к прикладным службам (например, FTP или SMTP), которые используют протокол Telnet для управления своими потоками данных, но не поддерживают опции Telnet. Для этих целей может использоваться клиент Telnet, если инициатива по согласованию опций для этого клиента запрещена.

3.3.5 Опция Telnet Linemode

Обсуждение

Важную роль играет новая опция Telnet LINEMODE [TELNET:12], обеспечивающая возможность согласования между клиентом и сервером обработки терминальных символов на стороне клиента, а не сервером. Когда клиент подготовит всю строку текста, он будет передавать ее серверу (обычно) в одном сегменте TCP. Эта опция позволяет существенно снизить число пакетов в сеансах Telnet и значительно ускорить отклик в нагруженных сетях или при значительных задержках в сети.

Опция LINEMODE позволяет динамически переключаться между локальной и удаленной обработкой символов. Например, соединение Telnet будет автоматически переходить в посимвольный режим при работе с полноэкранным редактором и возвращаться в строковый режим при завершении работы с редактором.

Предполагается, что после выпуска этого RFC хосты будут реализовать эту опцию для клиентских программ. Для корректной реализации опции на серверах последние должны обеспечивать возможность сказать локальной операционной системе, что не нужно обрабатывать символьный ввод, но следует запоминать текущее состояние терминала и уведомлять серверный процесс Telnet при изменении состояния. Это позволит обеспечить корректный вывод эхо-символов паролей и нормальную работу полноэкранных редакторов.

3.4 Пользовательский интерфейс TELNET

3.4.1 Прозрачность набора символов

Для реализаций клиентов Telnet **следует** обеспечивать возможность приема и передачи любых 7-битовых символов ASCII. **Следует** по возможности обходить все интерпретации специальных символов операционной системой пользовательского хоста, чтобы такие символы могли удобно приниматься и передаваться через соединение.

Какой-либо символ **должен** быть зарезервирован для перехода в командный режим (escape to command mode) с дублированием этого символа, когда он встречается в потоке данных. **Следует** предоставлять пользователю возможность выбора такого символа.

Для двоичных соединений клиент Telnet **может** обеспечивать escape-механизм для введения произвольных 8-битовых символов, если операционная система хоста не позволяет вводить такие символы непосредственно с клавиатуры.

Реализация

Вопросы прозрачности менее актуальны для серверов, но разработчики должны быть аккуратны и в этом случае - маскировать биты четности (передаются старыми, нестандартными клиентами) до того, как они будут переданы программам, ожидающим только NVT ASCII, и корректно обслуживать программы, запрашивающие 8-битовые потоки данных.

3.4.2 Команды Telnet

Клиент Telnet **должен** предоставлять пользователю возможность ввода управляющих функций Telnet IP, AO и AYT, **следует** также обеспечивать возможность ввода EC, EL и Break.

3.4.3 Ошибки соединений TCP

Клиенту Telnet **следует** сообщать пользователю о любых ошибках TCP, о которых сообщает транспортный уровень (см параграф «Интерфейс между TCP и прикладным уровнем» в работе [INTRO:1]).

3.4.4 Использование нестандартных портов

Для клиентов Telnet **следует** предоставлять пользователю возможность выбора нестандартного номера порта на хосте сервера Telnet.

3.4.5 Отсечение вывода

Для клиентов Telnet **следует** предоставлять пользователю возможность задавать режим отсечения вывода при передаче IP (см. 3.2.4).

Для любой схемы отсечения вывода, которая заставляет клиента Telnet отсекать вывод локально, пока не будет получен сигнал Telnet от сервера, **следует** обеспечить пользователю возможность вручную восстанавливать нормальный вывод, когда сервер не передает ожидаемый сигнал.

3.5. Требования к TELNET

Функция	*Параграф*	*Требования*
Согласование опций	3.2.1	Обязательно
Предотвращение петель при согласовании	3.2.1	Обязательно
Отказ от неподдерживаемых опций	3.2.1	Обязательно
Возможность согласования в течение всего сеанса	3.2.1	Следует
Использование по умолчанию режима NVT	3.2.1	Обязательно
Передача официальных названий в опции Term-Type	3.2.8	Обязательно
Восприятие любых имен в опции Term-Type	3.2.8	Обязательно
Поддержка опций Binary, Supress-GA	3.3.3	Обязательно
Опции Echo, Status, EOL, Ext-Opt-List	3.3.3	Следует
Реализация опции Window-Size при наличии поддержки в ОС	3.3.3	Следует
Сервер инициирует согласование режима	3.3.4	Следует
Пользователь может разрешить и запретить согласование единиц	3.3.4	Следует
Go-Ahead		
Сервер, не поддерживающий GA, согласует опцию Supress-GA	3.2.2	Обязательно
Клиент или сервер принимает опцию Supress-GA	3.2.2	Обязательно
Клиент Telnet игнорирует GA	3.2.2	Возможно
Функции управления		
Поддержка SE NOP DM IP AO AYT SB	3.2.3	Обязательно
Поддержка EOR EC EL Break	3.2.3	Возможно
Игнорирование неподдерживаемых функций управления	3.2.3	Обязательно
Клиент и сервер отбрасывают срочные данные вплоть до DM	3.2.4	Обязательно
Клиент передает Synch после IP, AO, AYT	3.2.4	Следует
Сервер отвечает Synch на IP	3.2.4	Возможно
Сервер отвечает Synch на AO	3.2.4	Обязательно
Клиент может отсекать вывод при передаче IP	3.2.4	Возможно
Кодировка символов		
Использование старшего бита в режиме NVT	3.2.5	Не следует
Использование старшего бита для контроля четности	3.2.5	Недопустимо
Согласование. двоичного режима при передаче старшего бита приложениям	3.2.5 3.2.6	Следует Обязательно
Дублирование IAC в потоке данных в любом режиме	3.2.7	Обязательно
Дублирование IAC в потоке данных при бинарном режиме	3.2.7	Обязательно
Следовать командам Telnet в бинарном режиме	3.2.7	Недопустимо
Завершение строки CR NUL в бинарном режиме		
Завершение строки		
EOL на сервере совпадает с локальным символом завершения строки	3.3.1	Обязательно
Сервер ASCII принимает CR LF и CR NUL как EOL	3.3.1	Обязательно
Клиент может передавать CR LF, CR NUL или LF	3.3.1	Обязательно
Клиент ASCII может выбирать между CR LF и CR NUL	3.3.1	Следует
Клиент по умолчанию использует CR LF	3.3.1	Следует
Неинтерактивное использование CR LF для EOL	3.3.1	Обязательно
Пользовательский интерфейс Telnet		
Ввод-вывод 7-битовых символов	3.4.1	Следует
Обход интерпретации символов локальной ОС	3.4.1	Следует
Escape-символ	3.4.1	Обязательно
Возможность пользовательского выбора	3.4.1	Следует
Escape для ввода 8-битовых символов	3.4.1	Возможно
Возможность ввода IP, AO, AYT	3.4.2	Обязательно
Возможность ввода EC, EL, Break	3.4.2	Следует
Передача пользователю сообщений об ошибках TCP	3.4.3	Следует
Возможность выбора нестандартного порта	3.4.4	Следует
Управление отсечением вывода при передаче IP	3.4.5	Следует
Возможность вручную восстановить режим вывода	3.4.5	Следует

4. Передача файлов

4.1 Протокол передачи файлов - FTP

4.1.1 Введение

Протокол передачи файлов FTP является основным средством обмена файлами в сети Internet. Текущая спецификация протокола содержится в RFC 959[1] [FTP:1].

FTP обеспечивает независимые одновременные соединения TCP для управления и передачи данных. Протокол FTP включает множество возможностей, из которых только часть относится к общеприменимым. Однако для каждой из

[1]В RFC 2228 (безопасность) и RFC 2640 (интернационализация) содержится ряд дополнений к этой спецификации. *Прим. перев.*

возможностей FTP существует, по крайней мере, одна реализация. Минимальная реализация, оговоренная в RFC 959, оказалась слишком примитивной, поэтому здесь определяются некоторые дополнительные требования.

Пользователи Internet в течение многих лет вынуждены были использовать дефективные реализации FTP. Разработчики приложений зачастую основывались на ошибочном предположении, что реализация FTP должна быть тривиальной. Это неверно, поскольку FTP имеет пользовательский интерфейс и программы должны работать (корректно) со всеми коммуникационными и операционными системами, обеспечивая устойчивость к их ошибкам.

4.1.2. Общие вопросы

4.1.2.1 Тип LOCAL: RFC 959, 3.1.1.4

Программы FTP **должны** поддерживать тип I (IMAGE или бинарный тип), а также тип L 8 (LOCAL - локальный с логическим размером байта 8). Машины, память которых организована в слова размерности m и значение m не кратно 8, **могут** также поддерживать тип L m.

Обсуждение

Команда TYPE L 8 часто требуется для передачи двоичных данных между машинами, память которых организована (например) в 36-битовые слова, и машинами с байтовой (8 битов) организацией памяти. Для машин с байтовой организацией памяти типы L 8 и IMAGE эквивалентны.

TYPE L m иногда используется программами FTP при обмене между двумя машинами с m-битовыми словами для обеспечения корректной передачи данных в естественном формате с одной машины на другую. Однако эта команда на таких машинах должна обеспечивать такой же эффект, как TYPE I.

4.1.2.2 Управление форматом Telnet: RFC 959, 3.1.1.5.2

Хостам, не делающим различий между TYPE N и TYPE T, **следует** реализовать тип T идентично типу N.

Обсуждение

Такое решение должно упрощать интероперабельность с хостами, которые различают эти типы.

Многие хосты используют для текстовых файлов внутреннее представление в виде строк символов ASCII, используя встроенные символы форматирования ASCII (LF, BS, FF, ...) для управления форматом при печати. Для таких хостов не существует разницы между пригодными для печати файлами и остальными типами файлов. Однако, системы, которые используют структурированные в виде записей файлы, требуют для печати использования специальных форматов (например, ASA для управления кареткой). Для таких хостов протокол FTP позволяет выбирать типе - TYPE N или TYPE T.

4.1.2.3 Структура страницы: RFC 959, 3.1.2.3 и Appendix I

Реализовать структуру страницы в общем случае **не следует**. Однако, если хосту не нужна реализация FTP для доступа к файлам типа random access (произвольный доступ) или holey («дырявый»), он **должен** использовать определенный формат структуры страницы вместо определения нового частного формата FTP.

4.1.2.4 Преобразования структуры данных: RFC 959, 3.1.2

Преобразования FTP между record-structure и file-structure **следует** делать обратимыми, чтобы расширить возможности использования файла на хосте получателе.

Обсуждение

RFC 959 требует обратимости преобразований структуры между record-structure и file-structure, но на практике вопросы эффективности и удобства зачастую препятствуют такой обратимости и требование остается невыполненным. Существует два различных подхода к передаче файлов - хост-получатель обрабатывает файлы или просто сохраняет их. Для варианта простого сохранения обратимость преобразований имеет важное значение. При обработке файлов получателем файл, создаваемый на приемной стороне, должен использовать формат, ожидаемый прикладной программой на этом хосте.

В качестве примера конфликта рассмотрим ориентированные на записи ОС, которые требуют, чтобы некоторые файлы данных использовали записи размером в точности 80 байтов. При сохранении файла на таком хосте сервер FTP должен обеспечивать возможность дополнения каждой строки или записи до требуемых 80 байтов; при последующем обращении к таким файлам исходное состояние не всегда можно восстановить (необратимое преобразование).

4.1.2.5 Управление соединением для передачи данных: RFC 959, 3.3

FTP-клиентам, которые используют потоковый режим (STREAM), **следует** посылать команду PORT для выделения нестандартного (non-default) порта для передачи данных по каждой из команд.

Обсуждение

Это требование обусловлено наличием значительной задержки между закрытием соединения TCP для пары сокетов и возможностью организации повторного соединения для этой же пары. Использование нестандартных портов позволяет организовать множество потоков данных в одном сеансе FTP. Передачи команды PORT можно избежать при использовании других режимов (не потокового), оставляя открытым соединение для передачи данных.

4.1.2.6 Команда PASV: RFC 959, 4.1.2

Сервер FTP **должен** поддерживать команду PASV.

Если в одной сессии организуется множество передач файлов различным клиентам, новые команды PASV должны вводиться перед каждой командой новой передачи для получения уникального номера порта.

Реализация

Формат отклика 227 на команду PASV не стандартизован должным образом. В частности, клиент FTP не может предполагать наличие круглых скобок, показанных на странице 40 в RFC 959 (и, фактически, опущенных на рисунке 3, стр. 43). Следовательно, клиент FTP, интерпретирующий отклик PASV, должен сканировать весь отклик для обнаружения первой цифры адреса хоста и номера порта.

Отметим, что h1,h2,h3,h4 задает IP-адрес серверного хоста, который передал отклик, а p1,p2 указывает нестандартный порт выделенный для передачи данных командой PASV.

4.1.2.7 Команды LIST и NLST: RFC 959, 4.1.3

Данные, возвращаемые командой NLST, **должны** содержать только простой список корректных параметров, которые сервер может непосредственно использовать, как аргументы в последующих командах передачи данных для отдельных файлов.

Для данных, возвращаемых командами LIST и NLST, рекомендуется использовать неявняй тип TYPE AN, если не задан в качестве текущего тип EBCDIC; если в качестве неявного задан тип TYPE EN, **следует** использовать этот тип.

Обсуждение

Многие клиенты FTP поддерживают макрокоманды, для загрузки (get) или выгрузки (put) файлов, соответствующих шаблону, с использованием команды NLST для получения списка имен. Расширение multiple-put является локальным для клиента, а multiple-get требует взаимодействия с сервером.

Для команд LIST и NLST разработан неявный тип, обеспечивающий совместимость с существующими клиентами FTP и поддерживающий групповые команды get.

4.1.2.8 Команда SITE: RFC 959, 4.1.3

Серверам FTP рекомендуется использовать команду SITE для поддержки нестандартных функций, взамен реализации фирменных нестандартных команд или нестандартного расширения существующих команд.

4.1.2.9 Команда STOU: RFC 959, 4.1.3

Команда STOU позволяет сохранять файлы с уникальными именами. При получении команды STOU сервер FTP **должен** возвратить актуальное имя файла в сообщении "125 Transfer Starting" или "150 Opening Data Connection", предшествующем передаче (код 250, указанный в RFC 959, некорректен для таких случаев). Точный формат сообщения показан ниже:

```
125 FILE: pppp
150 FILE: pppp
```

pppp представляет уникальное полное имя файла, который будет записан.

4.1.2.10 Код завершения строки Telnet: RFC 959, стр. 34

Недопустимо делать какие-либо предположения о соответствии между границами READ в управляющем соединении и последовательностями завершения строки Telnet EOL (CR LF).

Обсуждение

Таким образом, сервер или клиент FTP должен продолжать чтение символов из управляющего соединения, пока не будет получена полная последовательность Telnet EOL, прежде, чем начинать обработку команд (или откликов). Одна операция READ может получать из управляющего соединения несколько команд FTP.

4.1.2.11 Отклики FTP: RFC 959, 4.2, стр. 35

Сервер FTP **должен** передавать только корректно форматированные отклики в управляющее соединение. Отметим, что RFC 959 (в отличие от ранних спецификаций FTP) не содержит мер предосторожности для нестандартных откликов.

Серверам **следует** использовать коды откликов, определенные в RFC 959, всякий раз, когда это возможно. Однако сервер **может** использовать при необходимости иные коды откликов в соответствии с общими правилами (см. параграф 4.2). При выборе между кодами 4xx и 5xx серверам **следует** посылать коды 4xx (временный сбой), когда есть какая-то реальная возможность восстановления сервиса FTP в течение нескольких часов.

В общем случае клиентам **следует** использовать только старшую цифру 3-значного кода отклика для принятия решений - это позволяет избавиться от трудностей при получении откликов с нестандартными кодами.

Клиент FTP **должен** уметь работать с многострочными откликами. При получении откликов с превышающим допустимое значение числом строк клиент FTP **должен** сохранять нормальный режим работы (например, пропускать лишние строки до конца сообщения).

Клиентам FTP **не следует** специально интерпретировать код отклика 421[1], но **следует** детектировать закрытие сервером управляющего соединения.

Обсуждение

Реализации серверов с некорректными откликами часто приводят к зависанию клиентских программ FTP. Отметим, что RFC 959 устраняет неоднозначности в правилах передачи откликов, присутствовавшие в ранних спецификациях FTP.

Важно выбирать коды откликов FTP, которые позволяют отличать временные сбои от постоянных проблем, что позволяет успешно использовать клиентские демоны FTP. Работа таких программ зависит от кода отклика, на основе которого принимается решение о продолжении попыток. Использование кодов 5xx (постоянная проблема) для временных сбоев может приводить к некорректной работе демонов.

[1]Service not available, closing control connection - сервис недоступен, управляющее соединение закрывается.

Говоря о том, что отклики должны в точности соответствовать RFC 959, зачастую трактуют это соответствие, как дословную передачу. Однако, разработчикам серверов FTP следует выбирать (по возможности) тексты откликов, специфические для используемой ОС.

4.1.2.12 Соединения: RFC 959, 5.2

Слова "and the port used" во втором абзаце параграфа 5.2 RFC 959 являются (исторической) ошибкой и не должны приниматься во внимание.

На многодомных серверных хостах используемый по умолчанию порт передачи данных (L-1) **должен** связываться с тем же локальным адресом IP, который используется вместе с соответствующим портом управляющего соединения L.

Для клиентов FTP **недопустимо** передавать какие-либо коды управления Telnet, за исключением SYNCH и IP в управляющие соединения FTP. В частности, **недопустимо** для клиента согласовывать опции Telnet для управляющего соединения. Однако, сервер FTP **должен** быть способен воспринимать согласование опций Telnet и отказывать в таком согласовании (например, передавая DONT/WONT).

Обсуждение

Хотя в RFC сказано: «Server- and User-processes should follow the conventions for the Telnet protocol...[on the control connection][1]», это не имеет отношения к согласованию опций Telnet.

4.1.2.13 Минимальна реализация: RFC 959, 5.1

Перечисленные ниже команды и опции **должны** поддерживаться всеми клиентами и серверами FTP, за исключением случаев, когда файловая система или ОС не позволяют реализовать ту или иную возможность.

Типы: ASCII Non-print, IMAGE, LOCAL 8

Режимы: Stream

Структуры: File, Record[2]

Команды:

USER, PASS, ACCT,

PORT, PASV,

TYPE, MODE, STRU,

RETR, STOR, APPE,

RNFR, RNTO, DELE,

CWD, CDUP, RMD, MKD, PWD,

LIST, NLST,

SYST, STAT,

HELP, NOOP, QUIT.

Обсуждение

Поощряется реализация более широкого набора команд и опций. Например, в протоколе заложены важные средства обеспечения устойчивости (например, Restart, ABOR, блоковый режим), которые будут интересны некоторым пользователям Internet, но реализованы далеко не всегда.

Хост, который не использует структуры записей в своей файловой системе, может по-прежнему воспринимать файлы с STRU R, записывая байтовый поток «дословно».

4.1.3 Частные вопросы

4.1.3.1 Нестандартные команды

FTP позволяет использовать «экспериментальные» команды, имена которых начинаются с «X». При последующем включении команд в стандартные спецификации, они по-прежнему могут начинаться с X. Ниже представлен список экспериментальных команд:

RFC 959	Экспериментальные
MKD	XMKD
RMD	XRMD
PWD	XPWD
CDUP	XCUP
CWD	XCWD

Для всех реализаций FTP **следует** распознавать обе формы этих команд, просто просматривая дополнительные записи в таблице команд.

Реализация

Клиент FTP может обращаться к серверу, поддерживающему только X-команды, с помощью переключателя режима или за счет автоматического использования следующей процедуры - если стандартная (RFC 959) форма любой из

[1]сервер и клиент должны следовать соглашениям для протокола Telnet...[для управляющего соединения].
[2]Структуры Record **требуются** для тех хостов, чьи файловые системы поддерживают структуры записей.

перечисленных выше команд была отвергнута с кодом 500 или 502, следует попытаться использовать расширенную команду; все остальные отклики будут передаваться пользователю.

4.1.3.2 Время бездействия

Для серверных процессов FTP **следует** задавать время бездействия, по истечении которого процесс будет прерываться в случае отсутствия активности (нет команд или передачи данных). Значение тайм-аута **следует** делать настраиваемым и по умолчанию время бездействия должно составлять 5 минут.

Клиентскому процессу FTP (User-PI в RFC 959) тайм-аут на отклики нужен лишь в случаях вызова клиента из программ.

Обсуждение

Без использования тайм-аута сервер FTP может неограниченно долго оставаться в состоянии ожидания при крахе клиента без разрыва управляющего соединения.

4.1.3.3 Одновременное управление и передача данных

Обсуждение

При разработке протокола FTP ставилась задача обеспечения пользователю возможности передать в любой момент в процессе передачи данных команду STAT, на которую сервер будет незамедлительно возвращать информацию о состоянии (например, размер переданной части файла). Подобно этому команда ABOR должна обеспечивать возможность прерывания передачи данных в любой момент.

К несчастью, некоторые ОС (в небольших машинах) затрудняют программирование одновременных операций. Отдельные разработчики идут по пути наименьшего сопротивления, не поддерживая в своих реализациях возможности управления в процессе передачи данных. Но даже в минимальной реализации серверы должны быть готовы к восприятию и отсрочке команд STAT и ABOR, получаемых во время передачи данных.

4.1.3.4 Механизм FTP Restart

В описании отклика 110 на стр. 40-41 RFC 959 допущена ошибка, исправленная здесь. Сообщение restart reply, передаваемое через управляющее соединение от принимающего FTP клиенту FTP, имеет формат:

```
110 MARK ssss = rrrr
```

где:

- ssss - текстовая строка, которая появляется в Restart Marker в потоке данных и кодирует позицию в файловой системе отправителя;
- rrrr - кодирует соответствующую позицию в файловой системе получателя.

Кодирование зависит от используемой ОС и сетевой реализации и всегда генерируется и интерпретируется одной и той же системой (отправителем или получателем).

Когда FTP, реализующий рестарт, получает Restart Marker в потоке данных, **следует** форсировать запись данных до этой точки на стабильную среду для кодирования соответствующей позиции rrrr. Для FTP, передающего Restart Markers, **не допускается** предположение о возврате откликов 110 синхронно с данными (т. е., следует дождаться отклика 110 перед продолжением передачи данных).

Для сообщений об ошибках при рестарте передачи определяются два новых кода:

554 Requested action not taken: invalid REST parameter - запрошенное действие не выполнено: некорректный параметр REST.

Отклик 554 может быть результатом сервисной команды FTP, которой следует за командой REST. Отклик показывает, что существующий файл на сервере FTP невозможно репозиционировать в соответствии с командой REST.

555 Requested action not taken: type or stru mismatch - запрошенное действие не выполнено: несоответствие типа или stru.

Отклик 555 может быть результатом команды APPE или любой сервисной команды FTP, за которой следует команда REST. Этот код говорит о рассогласовании между текущими параметрами передачи (type и stru) и атрибутами существующего файла.

Обсуждение

Отметим, что механизм FTP Restart требует использования режима Block или Compressed для передачи данных, чтобы обеспечивалась возможность включения маркеров Restart Marker в поток данных. Частота передачи маркеров может быть достаточно низкой.

Restart Marker отмечает место в потоке данных, но получатель может выполнять некоторые преобразования данных при их сохранении в стабильной среде. В общем случае кодирование на приемной стороне должно включать любую информацию о состоянии, которая может потребоваться для возобновления передачи с любой точки потока данных FTP. Например, при передачах TYPE A некоторые принимающие хосты преобразуют последовательности CR LF в один символ LF при записи файла на диск. Если Restart Marker попадает между CR и LF, принимающая сторона должна указать в rrrr, что передача должна возобновляться в состоянии «CR has been seen and discarded» (получен и отброшен символ возврата каретки).

Отметим, что Restart Marker требуется обозначать, как строку печатных символов ASCII, независимо от типа данных.

RFC 959 говорит, что информация о возобновлении будет возвращаться пользователю. Это высказывание не следует понимать буквально. В общем случае клиенту FTP следует сохранять информацию о возобновлении (ssss,rrrr) на стабильной среде (например, дописывать ее в файл управления возобновлением передачи). Пустое поле управления рестартом следует создавать при начале передачи и автоматически удалять после ее успешного завершения. Предполагается, что имя такого файла будет связано с именем передаваемого файла и удаленного хоста (по типу имен для резервных копий редактируемых файлов в текстовых редакторах).

Возможны три варианта рестарта FTP.

(1) Передача от пользователя к серверу

Клиент FTP помещает маркеры Restart <ssss> в подходящие места потока данных. Когда сервер FTP получает маркер, он записывает все предшествующие данные на диск, кодируя позицию в своей файловой системе и состояние преобразования как rrrr, после чего возвращает отклик «110 MARK ssss = rrrr» через управляющее соединение. Клиент FTP дописывает пару (ssss,rrrr) в конец своего файла управления возобновлением передачи.

Для возобновления передачи FTP-клиент делает выборку последней пары (ssss,rrrr) из своего управляющего файла, меняет позицию в своей файловой системе и состояние, используя значение ssss, после чего передает серверу команду REST rrrr.

(2) Передача от сервера к пользователю

Сервер FTP помещает маркеры Restart <ssss> в подходящие места потока данных. Когда клиент FTP получает маркер, он записывает все предшествующие данные на диск, кодируя позицию в своей файловой системе и состояние преобразования как rrrr, после чего дописывает пару (rrrr,ssss) в конец своего файла управления рестартом.

Для возобновления передачи FTP-клиент делает выборку последней пары (ssss,rrrr) из своего управляющего файла, меняет позицию в своей файловой системе и состояние, используя значение ssss, после чего передает серверу команду REST ssss.

(3) Передача от сервера к серверу (Third-Party)

Передающий сервер помещает маркеры Restart <ssss> в подходящие места потока данных. Когда принимающий сервер получает маркер, он записывает все предшествующие данные на диск, кодируя позицию в своей файловой системе и состояние преобразования как rrrr, после чего возвращает отклик «110 MARK ssss = rrrr» через управляющее соединение. Клиент FTP дописывает пару (ssss,rrrr) в конег своего файла управления возобновлением передачи.

Для возобновления передачи FTP-клиент делает выборку последней пары (ssss,rrrr) из своего управляющего файла и отправляет сообщение REST ssss передающему серверу FTP и REST rrrr - принимающему серверу FTP.

4.1.4 Пользовательский интерфейс FTP

В этом разделе рассматривается пользовательский интерфейс FTP.

4.1.4.1 Спецификация Pathname

Поскольку протокол FTP предназначен для гетерогенных систем, реализация клиента **должна** поддерживать имена удаленных файлов как произвольные символьные строки, форма и содержание которые не ограничены соглашениями локальной ОС.

Обсуждение

В частности, имена удаленных файлов могут иметь произвольную длину и содержать любые печатаемые символы ASCII, включая пробелы (0x20). RFC 959 позволяет включать в имена все 7-битовые символы ASCII, за исключением CR и LF.

4.1.4.2 Команда QUOTE

Клиент FTP **должен** поддерживать команду QUOTE, которая будет передавать серверу произвольные символьные строки и выводить на консоль полученные на них отклики.

Чтобы команда QUOTE была полезна, клиентам FTP **следует** передавать серверу команды управления как ввод от пользователя, а не сохранять их для передачи серверу после того, как начнется передача данных.

Обсуждение

Команда QUOTE обеспечивает пользователям возможность доступа к серверам, которые требуют ввода специфических команд (например, SITE или ALLO), и позволяет реализовать возможности, не реализованные в стандартных клиентах FTP. В качестве примера использования QUOTE может служить команда TYPE A T для передачи файла на печать хостам, которые требуют различия типов, хотя клиент FTP не распознает эти типы.

4.1.4.3 Передача откликов пользователю

Для FTP-клиентов **следует** выводить для пользователя полный текст всех полученных сообщений об ошибках. **Следует** также поддерживать режим verbose, в котором выводятся полностью все переданные команды и полный текст откликов с кодом результата. Такая возможность позволяет упростить поиск проблем.

4.1.4.4 Поддержка синхронизации

Клиентам FTP **следует** быть устойчивыми к потере сообщений и неожиданным откликам, чтобы поддерживать синхронизацию команд с сервером.

4.1.5 Требования к FTP

Функция	*Параграф*	*Требования*
Если не делается различий, реализовать TYPE T, как TYPE N	4.1.2.2	Следует
Обратимое преобразование файлов/записей	4.1.2.4	Следует
Клиент FTP передает команду PORT для потокового режима	4.1.2.5	Следует
Реализация сервером FTP команды PASV PASV для каждой передачи отдельно	4.1.2.6 4.1.2.6	Обязательно Обязательно
Возможность использования отклика NLST в командах RETR	4.1.2.7	Обязательно

Функция	*Параграф*	*Требования*
Подразумеваемый тип для команд LIST и NLST	4.1.2.7	Следует
Команда SITE для нестандартных функций	4.1.2.8	Следует
Команда STOU возвращает указанное им файла (pathname)	4.1.2.9	Обязательно
Использование границ TCP READ на управляющем соединении	4.1.2.10	Недопустимо
Сервер передает отклики только в корректном формате	4.1.2.11	Обязательно
Сервер использует только стандартные отклики	4.1.2.11	Следует
Новые отклики в соответствии с требованиями параграфа 4.2	4.1.2.11	Возможно
Клиент использует только старшую цифру отклика	4.1.2.11	Следует
Клиент может работать с многострочными откликами	4.1.2.11	Обязательно
Специальная обработка клиентом откликов с кодом 421	4.1.2.11	Не следует
Порт данных по умолчанию использует тот же IP-адрес, что и порт управления	4.1.2.12	Обязательно
Клиент FTP передает команды Telnet (за исключением Synch, IP)	4.1.2.12	Недопустимо
Клиент FTP согласует опции Telnet	4.1.2.12	Недопустимо
Сервер FTP обслуживает опции Telnet	4.1.2.12	Обязательно
Поддержка «экспериментальных» команд	4.1.3.1	Следует
Тайм-аут по бездействию для серверов	4.1.3.2	Следует
Настраиваемое значение тайм-аута	4.1.3.2	Следует
Restart Marker как контрольная точка для приемной стороны	4.1.3.4	Следует
Отправитель предполагает синхронный характер откликов 110	4.1.3.4	Недопустимо
Поддержка типов (TYPE):		
ASCII - Non-Print (AN)	4.1.2.13	Обязательно
ASCII - Telnet (AT) - то же, что AN	4.1.2.2	Следует
ASCII - Carriage Control (AC)	959 3.1.1.5.2	Возможно
EBCDIC - (люба форма)	959 3.1.1.2	Возможно
IMAGE	4.1.2.1	Обязательно
LOCAL 8	4.1.2.1	Обязательно
LOCAL m[1]	4.1.2.1	Возможно
Поддержка режимов (MODE):		
Stream (поток)	4.1.2.13	Обязательно
Block (блок)	959 3.4.2	Возможно
Поддержка структур (STRUCTURE):		
File (файл)	4.1.2.13	Обязательно
Record (запись)	4.1.2.13	Обязательно[2]
Page (страница)	4.1.2.3	Не следует
Поддержка команд:		
USER	4.1.2.13	Обязательно
PASS	4.1.2.13	Обязательно
ACCT	4.1.2.13	Обязательно
CWD	4.1.2.13	Обязательно
CDUP	4.1.2.13	Обязательно
SMNT	959 5.3.1	Возможно
REIN	959 5.3.1	Возможно
QUIT	4.1.2.13	Обязательно
PORT	4.1.2.13	Обязательно
PASV	4.1.2.6	Обязательно
TYPE	4.1.2.13	Обязательно[3]
STRU	4.1.2.13	Обязательно[3]
MODE	4.1.2.13	Обязательно[3]
RETR	4.1.2.13	Обязательно
STOR	4.1.2.13	Обязательно
STOU	959 5.3.1	Следует
APPE	4.1.2.13	Обязательно
ALLO	959 5.3.1	Возможно
REST	959 5.3.1	Возможно
RNFR	4.1.2.13	Обязательно
RNTO	4.1.2.13	Обязательно
ABOR	959 5.3.1	Возможно
DELE	4.1.2.13	Обязательно
RMD	4.1.2.13	Обязательно
MKD	4.1.2.13	Обязательно
PWD	4.1.2.13	Обязательно
LIST	4.1.2.13	Обязательно
NLST	4.1.2.13	Обязательно
SITE	4.1.2.8	Возможно
STAT	4.1.2.13	Обязательно
SYST	4.1.2.13	Обязательно
HELP	4.1.2.13	Обязательно
NOOP	4.1.2.13	Обязательно

[1]Число m задает размер слова памяти в битах.
[2]Для хостов с файловой системой на основе записей; в остальных случаях необязательно.
[3]Для показанных раньше значений.

Функция	*Параграф*	*Требования*
Пользовательский интерфейс:		
Произвольные имена файлов (pathname)	4.1.4.1	Обязательно
Реализация команды QUOTE	4.1.4.2	Обязательно
Непосредственна передача команд управления	4.1.4.2	Следует
Вывод сообщений об ошибках на консоль пользователя	4.1.4.3	Следует
Режим Verbose	4.1.4.3	Следует
Поддержка синхронизации с сервером	4.1.4.4	Следует

4.2 Тривиальный протокол передачи файлов TFTP

4.2.1 Введение

Простой протокол передачи файлов (Trivial File Transfer Protocol - TFTP) определен в RFC 783 [TFTP:1].

TFTP своими средствами обеспечивает надежную доставку на базе транспортного протокола UDP, используя простую систему подтверждений stop-and-wait (остановиться и подождать). Поскольку TFTP работает только с одним окном размером 512 октетов, этот протокол может эффективно использоваться только на путях с небольшим значением произведения **задержка*полоса**. Интерфейс TFTP очень прост и не обеспечивает контроля доступа и безопасности.

Основным применением TFTP является стартовая загрузка (bootstrapping) хостов через локальную сеть, поскольку этот протокол достаточно прост и может быть легко реализован в EPROM [BOOT:1, BOOT:2]. Производителям оборудования просто требуется поддерживать TFTP для загрузки устройств.

4.2.2 Общие вопросы

Спецификация TFTP [TFTP:1] написана в открытом стиле и не определяет полностью многие части протокола.

4.2.2.1 Режимы передачи: RFC 783, стр. 3

Не следует не поддерживать режим передачи mail.

4.2.2.2 Заголовок UDP: RFC 783, стр. 17

Поле Length (длина) заголовка UDP определено некорректно; это поле включает размер заголовка UDP - 8 октетов.

4.2.3 Частные вопросы

4.2.3.1 Sorcerer's Apprentice Syndrome

В спецификации протокола содержится серьезна ошибка - «синдром ученика колдуна[1]», которая хоть и не приводит к некорректной передаче (файл всегда передается корректно, если передача завершена), но может привести к избыточным повторам передачи и тайм-аутам.

Во всех реализациях эта ошибка **должна** быть исправлена - отправитель (т. е., сторона, передающая пакеты данных) никогда не должен заново передавать текущий пакет данных (DATA) при получении дубликата подтверждения ACK.

Обсуждение

Ошибка связана с правилом, по которому люба сторона, получив старый дубликат, может заново передать текущую дейтаграмму. Если пакет задержан в сети, но успешно доставлен после того, как истекло время ожидания и другая сторона повторила передачу, может быть сгенерирован дубликат отклика. Если в ответ на такой дубликат передача будет повторена еще раз, передача всех остальных дейтаграмм будет дублироваться (если не будет утери дейтаграммы, которая прервет повтор). В дополнение к этому, поскольку задержка зачастую связана с насыщением сети, передача дубликатов обычно усиливает насыщение, что ведет к новым задержкам и т. д.

Для понимания проблемы может быть полезна приведенная ниже иллюстрация.

```
                TFTP A                          TFTP B
(1)             Прием ACK X-1
                Передача DATA X
(2)                                             Прием DATA X
                                                Передача ACK X
              (ACK X задерживается в сети и возникает тайм-аут):
(3)             Повтор передачи DATA X

(4)                                             Прием DATA X (повт.)
                                                Передача ACK X (повт.)
(5)             Прием (задерж.) ACK X
                Передача DATA X+1
(6)                                             Прием DATA X+1
                                                Передача ACK X+1
(7)             Прием ACK X (повт.)
                Передача DATA X+1 (повт.)
(8)                                             Прием DATA X+1 (повт.)
                                                Передача ACK X+1 (повт.)
(9)             Прием ACK X+1
                Передача DATA X+2
(10)                                            Прием DATA X+2
                                                Передача ACK X+3
(11)            Прием ACK X+1 (повт.)
                Передача DATA X+2 (повт.)
```

[1] Sorcerer's Apprentice Syndrome.

```
(12)                                    Прием DATA X+2 (повт.)
                                        Передача ACK X+3 (повт.)
```

Отметим, что после доставки задержанного подтверждения ACK протокол начинает дублировать все последующие пакеты (пп. 5-8 и 9-12). Проблема вызывается не тайм-аутом на любой из сторон, а повторной передачей обеими сторонами при получении дубликатов.

Для решения проблемы нужно разорвать возникшую петлю, как показано выше. Это аналогично поведению протокола TCP. Можно удалить таймер повторной передачи на приемной стороне, поскольку повторная передача подтверждения ACK не будет вызывать каких-либо действий; это упрощение TFTP особенно полезно при использовании протокола для стартовой загрузки. Сохранение таймера возможно и может оказаться полезным, если повторно переданное подтверждение ACK заменяет потерянное в сети. Для отправителя таймер повторной передачи остается необходимым.

4.2.3.2 Алгоритм определения тайм-аута

Реализация TFTP **должна** использовать адаптивный тайм-аут.

Реализация

Алгоритмы повторной передачи TCP обеспечивают полезный прототип. Необходимо реализовать по крайней мере экспоненциальное изменение тайм-аута повторной передачи.

4.2.3.3 Расширения

К протоколу TFTP было добавлено множество нестандартных расширений, включающих дополнительные режимы передачи и обеспечение безопасности (пароли). Ни одно из этих расширений не было стандартизовано.

4.2.3.4 Контроль доступа

При реализации сервера TFTP **следует** включать некоторые настраиваемые возможности контроля за счет задания полных имен файлов, которые допустимы при операциях TFTP.

4.2.3.5 Широковещательные запросы

Запрос TFTP, отправленный по широковещательному адресу, **следует** отбрасывать без уведомления.

Обсуждение

По причине слабого контроля доступа в TFTP, передача широковещательных запросов TFTP в чужие сети может пробить существенную брешь в безопасности.

4.2.4 Требования к TFTP

Функция	*Параграф*	*Требования*
Преодоление синдрома Sorcerer	4.2.3.1	Обязательно
Режимы передачи:		
netascii	RFC 783	Обязательно
octet (октет)	RFC 783	Обязательно
mail (почта)	4.2.2.1	Не следует
extensions (Расширения)	4.2.3.3	Возможно
Использование адаптивного тайм-аута	4.2.3.2	Обязательно
Настраиваемое управление доступом	4.2.3.4	Следует
Игнорирование широковещательных запросов	4.2.3.5	Следует

5. Электронная почта - SMTP и RFC 822

5.1 Введение

В стеке протоколов TCP/IP электронная почта в формате RFC 822 [SMTP:2] передается с помощью протокола SMTP[1], определенного в RFC 821 [SMTP:1].

Хотя протокол SMTP остается неизменным уже в течение многих лет, сообщество Internet внесло некоторые изменения в способы использования SMTP. В частности, переход к системе доменных имен (Domain Name System или DNS) потребовал изменения формата почтовых адресов и маршрутизации электронной почты. В этом разделе предполагается наличие у читателя базовых познаний в части DNS (см. параграф 6.1).

RFC 822 является стандартом Internet для форматов электронной почты. RFC 822 отменяет действие предшествующих стандартов, хотя RFC 733 может еще использоваться, несмотря на его отмену. Эти форматы для краткости обозначают номерами - 822 и 733.

RFC 822 используется также за пределами почтовой среды Internet с почтовыми протоколами, отличными от SMTP, да и протокол SMTP также адаптирован для использования в средах, отличных от Internet. Отметим, что данный документ содержит правила использования SMTP и RFC 822 только для среды Internet; в других почтовых средах, использующих эти протоколы, можно ожидать иных правил.

5.2 Общие вопросы

Этот раздел тесно связан с RFC 821 и RFC 822. Спецификация SMTP в RFC 821 описана четко и однозначно, а также содержит множество примеров, поэтому реализация не должна вызывать затруднений. В данном разделе просто рассмотрены некоторое важные аспекты RFC 821 и внесен ряд поправок.

RFC 822 - большой и сложный документ, содержащий синтаксические определения. К сожалению, неполные и некорректные реализации RFC 822 встречаются на каждом шагу. Сегодня используются практически все из множества форматов RFC 822, поэтому программы должны распознавать и корректно интерпретировать весь синтаксис RFC 822.

[1]Simple Mail Transfer Protocol - простой протокол передачи почты.

5.2.1 Модель SMTP: RFC 821, раздел 2

Обсуждение

Электронная почта передается с помощью серии транзакций запрос-отклик между клиентом (sender-SMTP) и сервером (receiver-SMTP). Эти транзакции проверяют (1) корректность сообщения, состоящего из заголовка и тела письма, а также (2) SMTP-адреса для отправителя и получателя (envelope - конверт).

Программы SMTP аналогичны агентам MTA[1] в среде X.400. Есть также другой уровень программ, расположенных ближе к пользователю, которые отвечают за сборку и анализ заголовков в сообщениях RFC 822; эта компонента называется пользовательским агентом (UA[2]) в среде X.400 и мы будем применять этот термин в данном документе. Существуют четкие различия между пользовательским агентом и реализацией SMTP, поскольку они работают на разных уровнях протокола. Отметим, однако, что это различие может неточно отражаться структурой типичных реализаций почтовых программ Internet. Очень часто программы, называемые mailer, реализуют функции SMTP и некоторые функции пользовательского агента; остальные функции UA включаются в пользовательский интерфейс, который служит для чтения и подготовки почтовых сообщений.

Конверт SMTP создается на стороне отправителя (обычно пользовательским агентом) при передаче сообщения в очередь для программы Sender-SMTP. Адрес для конверта может быть построен по адресу в заголовке сообщения, полученному от пользовательского интерфейса (например, для выполнения запроса bcc:), или найден в локальных конфигурационных параметрах (например, в списке рассылок). В общем случае конверт SMTP не может быть сгенерирован заново на более поздних этапах доставки почты, поэтому он передается отдельно от сообщения с использованием команд MAIL и RCPT протокола SMTP.

В RFC 821 предполагается, что почта доставляется отдельным пользователям на каждом хосте. С развитием доменной системы и началом маршрутизации почты на основе записей MX[3] почтовые программы должны обеспечивать доставку почты пользователям в домене, а не на отдельно взятом хосте. Однако это **не меняет** характера SMTP, который является протоколом обмена почтой между хостами.

5.2.2 Канонизация имен: RFC 821, 3.1

Имена доменов, которые Sender-SMTP передает в командах MAIL и RCPT, **должны** быть «канонизированы», т. е., должны быть полностью заданными именами (principal name или domain literal), а не кличками (nickname) или сокращениями. Канонизированное имя идентифицирует непосредственно хост или имя MX и не может быть CNAME (псевдоним имени).

5.2.3 Команды VRFY и EXPN: RFC 821, 3.3

Получатель SMTP **должен** поддерживать команду VRFY и **следует** также реализовать команду EXPN (это отличается от требований RFC 821). Однако **возможно** запрещать обработку команд VRFY и EXPN с помощью конфигурационных опций (более того, можно даже запрещать команду EXPN для отдельных списков).

Для команды VRFY здесь определяется новый код отклика:

252 Cannot VRFY user - невозможно проверить пользователя (например, отсутствует локальна информация), но будет предпринята попытка доставить почту этому пользователю.

Обсуждение

Пользователи и администраторы SMTP регулярно обращаются к этим командам для поиска проблем с доставкой почты. С ростом использования многоуровневых почтовых списков (зачастую, более 2 уровней), возрастает важность команды EXPN при определении возможных петель в списках доставки. С другой стороны, существует мнение, что команда EXPN представляет угрозу сохранению тайны личности и, возможно, безопасности[4].

5.2.4 Команды SEND, SOML, SAML: RFC 821, 3.4

SMTP **может** реализовать команды для передачи сообщений на пользовательский терминал: SEND, SOML, SAML.

Обсуждение

Предполагается, что трансляция почты (mail relaying) с помощью записей MX несовместима с использованием команды SEND для непосредственной доставки сообщений на пользовательский терминал. Однако принимающая программа SMTP, которая не способна писать непосредственно на пользовательский терминал, может передавать отклик "251 User Not Local" (нелокальный пользователь) на RCPT с последующей командой SEND для информирования оператора о возможности отложенной доставки.

5.2.5 Команда HELO: RFC 821, 3.5

Отправитель SMTP **должен** обеспечивать корректность параметра <domain> в команде HELO (полное доменное имя хоста) для клиентского хоста. В результате этого получателю SMTP не нужно будет выполнять преобразования MX для этого имени, чтобы проверить корректность параметра HELO.

Получатель HELO **может** проверить, что параметр HELO реально соответствует IP-адресу отправителя. Однако получатель **не имеет права** отказываться от восприятия сообщения даже при отрицательном результате проверки отправителя команды HELO.

Обсуждение

Проверка параметра HELO требует определения доменного имени и может, следовательно, потребовать значительного времени. Ниже предлагается другой способ определения подставных отправителей с помощью команды DATA.

---

[1] Message Transfer Agent - агент переноса сообщений.
[2] User Agent - пользовательский агент. *Прим. перев.*
[3] Mail-exchange - обмен почтой.
[4] Во многих современных руководствах по безопасности рекомендуется отключить команды VRFY и EXPN. *Прим. перев.*

Отметим также, что аргумент HELO все равно должен использовать корректный синтаксис <domain>, поскольку это имя будет появляться в строке Received: (при некорректном имени возникает ошибка 501).

Реализация

Когда проверка параметра HELO дает отрицательный результат, предлагается вставлять примечание о невозможности проверки отправителя в заголовок сообщения (например, в строку Received:).

5.2.6 Трансляция почты: RFC 821, 3.6

Различают три типа пересылки почты (возможно, с промежуточным сохранением):

(1) Простая программа пересылки (mail exchanger) рассылает сообщения с использованием частной информации о получателях (см. RFC 821, параграф 3.2).

(2) Транслятор SMTP (mail relay) пересылает сообщения в среде SMTP с использованием явно заданного отправителем маршрута (explicit source route), как определено в параграфе 3.6 RFC 821. Функции SMTP relay используют форму "@...:" для задания маршрута в соответствии с RFC 822 (см. 5.2.19 ниже).

(3) Почтовый шлюз (mail gateway) передает сообщения между различными средами. Правила работы почтовых шлюзов рассмотрены в параграфе 5.3.7.

Хостам Internet, пересылающим почту, но не являющимся шлюзами в другие почтовые среды (т. е., относящимся к типу (1) или (2)), **не следует** менять поля заголовков в сообщениях, хотя можно добавлять строку Received: в соответствии с требованиями параграфа 5.2.8.

Отправителям SMTP **не следует** передавать команду RCPT TO:, содержащую явный маршрут, с использованием адреса в формате "@...:". Таким образом, функции трансляции почты, определенные в параграфе 3.6 RFC 821, не следует использовать.

Обсуждение

Задача состоит в полном искоренении source routing и упразднении явного задания маршрутов для доставки почты в среде Internet. Задание маршрута не требуется и во всех случаях следует использовать простую форму адреса получателя - user@domain. Это является результатом принятия решения на уровне архитектуры почтовой среды об использовании универсального именования вместо явного задания маршрутов доставки почты. Таким образом, SMTP обеспечивает сквозную связность, а DNS - уникальные в масштабе планеты и не зависящие от местоположения имена. Для обработки случаев, когда может потребоваться задание маршрута используются записи MX.

Получатель SMTP **должен** воспринимать синтаксис явного задания маршрута в конверте, но он **может** реализовать функции трансляции в соответствии с параграфом 3.6 RFC 821. Если функция трансляции не реализована, получателю **следует** попробовать доставить сообщение напрямую хосту, указанному в адресе справа от знака @.

Обсуждение

Предположим для примера, что хост, не поддерживающий трансляции, получает сообщение с командой SMTP "RCPT TO:<@ALPHA,@BETA:joe@GAMMA>"(ALPHA, BETA и GAMMA представляют доменные имена). Вместо отказа с возвратом ошибки 550 (как предлагается на стр. 20 в RFC 821), хосту следует попытаться переслать сообщение напрямую в GAMMA с помощью команды RCPT TO:<joe@GAMMA>". Поскольку этот хост не поддерживает трансляции, ему не требуется обновлять путь возврата.

Некоторые считают, что задание маршрута может иногда потребоваться при отправке почты вручную в случаях наличия сбоев; однако реальность и важность такого применения весьма сомнительны. Использование явной трансляции SMTP для решения таких задач не представляется разумным и, фактически, не обеспечивает успеха, поскольку многие хосты не поддерживают явного задания маршрутов. В некоторых случаях для решения таких задач используется "%-hack" (см. параграф 5.2.16).

5.2.7 Команда RCPT: RFC 821, 4.1.1

Хост, поддерживающий функции SMTP-получателя, **должен** обеспечивать почтовый ящик Postmaster.

Получатель SMTP может проверять параметры RCPT при доставке; однако, отклики RCPT **недопустимо** задерживать сверх разумного времени (см. 5.3.2).

Следовательно, отклик 250 OK для RCPT не обязательно говорит о корректности указанного адреса получателя. Информация об ошибках, обнаруженных после восприятия сообщения, будет передаваться в виде почтовых сообщений в соответствующий адрес (см. 5.3.3).

Обсуждение

Набор условий, по которому параметр RCPT должен проверяться незамедлительно, был задан при разработке архитектуры. Передача уведомления об ошибке в адресе получателя для отправителя SMTP до отправки всего сообщения имеет достаточно важное значение, поскольку позволяет снизить расход времени и полосы канала, но это преимущество может быть утеряно при длительной проверке RCPT.

Например, получатель может проверить незамедлительно любую локальную ссылку (зарегистрированный локально почтовый ящик). С другой стороны, ограничение «разумным временем» в общем случае предполагает отложенную проверку для списков рассылки (пока сообщение не будет передано и воспринято), поскольку проверка большого числа адресов потребует продолжительного времени. Реализация программы может использовать или не использовать отложенную проверку адресов, которые не являются локальными и, следовательно, требуют обращения к DNS. Если используется DNS и при запросе обнаруживается некритичная ошибка (например, тайм-аут), адрес следует считать корректным.

5.2.8 Команда DATA: RFC 821, 4.1.1

Каждый получатель SMTP (не только тот, который принимает сообщения для трансляции или окончательной доставки [SMTP:1]), **должен** вставлять строку Received: в начале сообщения. В этой строке, которая в RFC 821 названа "time stamp line" (строка с временной меткой), указывается:

- в поле FROM **следует** включать (1) имя хоста-отправителя, представленное в команде HELO, и (2) доменное имя с адресом IP, определенным из соединения TCP;
- поле ID **может** содержать "@" (как предложено в RFC 822), но это необязательно;
- поле FOR **может** содержать список <path>, если было введено множество команд RCPT.

Для почтовых программ Internet **недопустимо** изменять строки Received:, добавленные в заголовок раньше.

Обсуждение

Включение имени хоста и IP-адреса отправителя в строку Received: может предоставить достаточно информации для обнаружения источников недозволенной почты и позволяет избавиться от необходимости явной проверки параметра HELO.

Строки Received: предназначены, прежде всего, для прослеживания (человеком) почтовых маршрутов, прежде всего в целях поиска проблем (см. также Обсуждение в параграфе 5.3.7).

Когда получатель SMTP выполняет окончательную доставку (final delivery) сообщения, он **должен** передать адрес MAIL FROM из конверта SMTP, связанного с данным сообщением, для использования в тех случаях, когда позднее требуется передать отправителю информацию об ошибках (см. 5.3.3). Это аналогично требованию к шлюзам при передаче почты из Internet в иную почтовую среду (см. 5.3.7).

Обсуждение

Отметим, что окончательный отклик на команду DATA зависит только от успеха при передаче и сохранении сообщения. Проблемы с адресом получателя могут привести (1) к сообщению об ошибке при вызове команды RCPT или (2) передаче последующего сообщения об ошибке в адрес отправителя.

Реализация

Информация MAIL FROM: может передаваться как параметр или строка Return-Path: в начале сообщения.

5.2.9 Синтаксис команд: RFC 821, 4.1.2

Синтаксис команды MAIL FROM: в RFC 821 не рассматривает случай пустой строки пути - MAIL FROM: <> (см. стр. 15 в RFC 821). Пустые пути возврата **должны** поддерживаться.

5.2.10 Отклики SMTP: RFC 821, 4.2

Получателю SMTP **следует** передавать только отклики с кодами, перечисленными в параграфе 4.2.2 RFC 821 или в данном документе. По возможности получателю SMTP **следует** использовать в откликах тексты, приведенные в примерах RFC 821.

Отправитель SMTP **должен** определять свои действия только на основе кода отклика, но не его текста (за исключением откликов 251 и 551); любой текст (или отсутствие такового) должен восприниматься нормально. Пробелы после кода отклика рассматриваются как часть текста. Отправителю SMTP **следует** проверять только первую цифру в коде отклика (см. Приложение E в RFC 821).

Обсуждение

Могут возникнуть проблемы с интероперабельностью при использовании кодов отклика, не указанных явно в параграфе 4.3 RFC 821, но корректных в соответствии с теорией откликов, рассмотренной в Приложении E (RFC 821).

5.2.11 Прозрачность: RFC 821, 4.5.2

Разработчики программ **должны** быть уверены, что их почтовые системы всегда добавляют и удаляют точки для обеспечения прозрачности сообщений.

5.2.12 Использование WKS при обработке MX: RFC 974, стр. 5

RFC 974 [SMTP:3] рекомендует запрашивать у DNS записи WKS[1], чтобы убедиться в поддержке SMTP каждым предложенным получателем. Однако опыт показывает, что поддержка WKS реализована не везде, поэтому WKS при обработке MX использовать **не следует**.

Далее приведены комментарии к RFC 822, организованные по разделам документа.

5.2.13 Спецификация сообщений: RFC 822, глава 4

Синтаксис строки Return-path не предусматривает возможности пустого пути возврата, которая используется для предотвращения петель при уведомлениях об ошибках (см. 5.3.3). Полный синтаксис имеет вид:

```
return = "Return-path"  ":" route-addr "Return-path"  ":" "<" ">"
```

Набор добавленных в последнее время полей заголовков включает поле Content-Type, определенное в RFC 1049 [SMTP:7]. Это поле позволяет программам для чтения почты идентифицировать тип структурированного тела сообщения и определить процесс для его корректного отображения [SMTP:7]. Пользовательский агент **может** поддерживать это поле.

[1]Well-Known Service - общеизвестный сервис.

5.2.14 Спецификации даты и времени: RFC 822, глава 5

Синтаксис дат был недавно изменен и теперь имеет вид:

```
date = 1*2DIGIT month 2*4DIGIT
```

Во всех почтовых сообщениях **следует** использовать 4 знака для обозначения года, чтобы упростить переход в следующее столетие.

Существует сильная тенденция в направлении использования цифровых обозначений часовых поясов и приложениям **следует** использовать цифровые обозначения вместо имен. Однако все реализации **должны** воспринимать оба типа обозначений. При использовании имен часовых поясов, эти имена **должны** в точности соответствовать RFC 822.

Военные часовые пояса в RFC 822 указаны некорректно - счет от UT ведется в обратном направлении. В результате военные часовые пояса в заголовках RFC 822 не несут полезной информации.

Наконец, отметим, что в определении "zone" при рассмотрении синтаксиса в Приложении D допущена опечатка; корректное определение приведено в главе 3 RFC 822.

5.2.15 Изменение синтаксиса: RFC 822, 6.1

Синтаксическое определение почтового ящика (mailbox) в RFC 822 недавно было заменено:

```
mailbox =  addr-spec            ; simple address
        / [phrase] route-addr   ; name & addr-spec
```

Т. е., фраза, предшествующая адресу маршрута (route address) сейчас является **необязательной**. Это изменение делает корректным приведенный ниже фрагмент заголовка:

```
From: <craig@nnsc.nsf.net>
```

5.2.16 Локальный путь: RFC 822, 6.2

Базовая спецификация адреса почтового ящика имеет форму: local-part@domain. Вместо local-part (локальная часть адреса) иногда используют термин left-hand side (левая часть адреса).

Хост, который пересылает сообщение, но не является его получателем, имеет дело с правой частью адреса - доменом. При пересылке сообщений **недопустимо** менять что-либо в локальной части адреса.

Когда почта передается через шлюз из почтовой среды Internet в инородную почтовую среду (см. 5.3.7), маршрутная информация для такой среды может быть вложена в локальную часть адреса. В этом случае шлюз будет интерпретировать локальную часть адреса в соответствии с требованиями чужой среды.

Обсуждение

Хотя задание маршрута отправителем **не следует** использовать для почты Internet (см. 5.2.6), существуют почтовые среды, в которых механизмы работы шлюзов основаны на таких маршрутах. Обычно маршруты для таких сред встраивают в локальную часть адреса при передаче почты через Internet. Когда почта приходит на нужный почтовый шлюз Internet, этот шлюз интерпретирует локальную часть адреса и строит адрес или маршрут для инородной почтовой среды.

Например, хост Internet может отправить почту по адресу a!b!c!user@gateway-domain. Сложная локальная часть a!b!c!user будет прозрачно передаваться через Internet, но указанный шлюз разберет эту часть и преобразует ее в корректный адрес другой почтовой среды.

Вложенные маршруты source route иногда помещаются в локальную часть адреса с использованием знака "%" в качестве правого оператора маршрутизации. Например, в адресе:

```
user%domain%relay3%relay2@relay1
```

знак % показывает, что почта маршрутизируется из relay1 через relay2 и relay3 для передачи пользователю user в домене domain. Такую нотацию часто называют %-hack. Предполагается, что % имеет меньший приоритет, нежели другие операторы маршрутизации (например, "!"), спрятанные в локальной части адреса. Например, a!b%c будет интерпретироваться как (a!b)%c.

Только хосту-получателю (в нашем случае, relay1) дозволено анализировать локальную часть user%domain %relay3%relay2.

5.2.17 Доменные имена: RFC 822, 6.2.3

Почтовая программа (mailer) **должна** воспринимать и разбирать доменные литералы Internet, контекст которых (dtext в RFC 822) содержит адрес хоста в десятичном формате с разделением точками. Это соответствует требованиям параграфа 2.1 для случая электронной почты.

Программа SMTP **должна** принимать и распознавать доменные литералы для любого из своих адресов IP.

5.2.18 Общие ошибки при форматировании адресов: RFC 822, 6.1

Ошибки при форматировании и анализе адресов формата 822 к сожалению встречаются постоянно. В этом параграфе рассматриваются лишь наиболее распространенные ошибки. Пользовательский агент **должен** воспринимать все корректные форматы адресов RFC 822; **недопустимо** генерирование адресов с некорректным синтаксисом.

- Общей ошибкой является сохранение точки с запятой (;) после идентификатора группы.
- Некоторые системы допускают ошибки при генерации полных имен в создаваемых сообщениях. Справа от знака @ в адресе заголовка **должно** размещаться полное доменное имя (FQDN[1]).

  Например, некоторые системы некорректно указывают доменное им в поле From:. В таких случаях возникают проблемы при попытке использования команды reply на приемной стороне.

[1]Fully-qualified domain name.

Обсуждение

Хотя RFC 822 допускает локальное (внутри домена) использование сокращенных доменных имен, применение RFC 822 для почты Internet не позволяет использовать такие сокращения. Для хостов Internet недопустимо передавать сообщения SMTP, заголовок которых содержит сокращенное доменное имя в поле адреса. Такие сокращения допустимы только для заголовков сообщений, которые не будут передаваться через Internet, как сказано в параграфе 5.2.6.

- Многие системы не умеют корректно разбирать заголовки с указанным маршрутом из нескольких частей:

```
@relay1,@relay2,@relay3:user@domain.
```

- Некоторые системы ошибочно добавляют точку в конце полного доменного имени в адресах и идентификаторах сообщений. Это является нарушением синтаксиса RFC 822.

5.2.19 Явное задание маршрута: RFC 822, 6.2.7

Программам хостов Internet **не следует** создавать заголовки RFC 822, содержащие адреса с явным маршрутом (explicit source route), но они **должны** воспринимать такие заголовки в целях совместимости.

Обсуждение

RFC 822 говорит: "The use of explicit source routing is discouraged" (рекомендуется избегать использования явно заданных маршрутов в адресах). На многих хостах поддержка маршрутов RFC 822 реализована некорректно, поэтому синтаксис на практике не обеспечивает однозначной трактовки. Многие пользователи считают этот синтаксис опасным. Явное задание маршрута в конверте не требуется для доставки (см. 5.2.6). В силу всего сказанного явное задание маршрутов с использованием синтаксиса RFC 822 не применяется в заголовках электронной почты Internet.

Как было сказано в параграфе 5.2.16, необходимо обеспечить возможность встраивания явных маршрутов в локальную часть адреса (например, за счет использования %-hack), чтобы позволить шлюзам передавать почту в инородные среды, требующие явного задания маршрута. Внимательный читатель заметит, что для пользовательских агентов не существует способа обнаружить и предотвратить использование таких неявных маршрутов при передаче почты в среде Internet. Мы можем только рекомендовать не применять задание маршрутов для почты Internet - это не нужно и нежелательно.

5.3 Частные вопросы

5.3.1 Стратегия очередей SMTP

Общая структура реализации SMTP на хосте включает пользовательские почтовые ящики, одну или несколько областей для организации очередей транзитных сообщений, а также один или несколько процессов-демонов для приема и передачи почты. Точная структура будет зависеть от потребностей пользователей, а также числа и размера поддерживаемых хостом списков рассылок. Здесь рассмотрены вопросы оптимизации, позволяющие повысить эффективность работы почтовых систем (в частности, систем с большим трафиком).

Любая стратегия работы с очередями **должна** включать:

- время ожидания (тайм-аут) для всех операций (см. 5.3.2);
- невозможность передачи сообщений об ошибке в ответ на сообщения об ошибке.

5.3.1.1 Стратеги передачи

Общая модель передающей стороны SMTP включает один или несколько процессов, периодически пытающихся передать исходящую почту. В типовой системе программы, готовящие почтовые сообщения, используют некий метод запроса немедленных действий для вновь созданного сообщений, однако почта не может быть отправлена незамедлительно, поэтому новые сообщения **должны** помещаться в очередь, к которой периодически обращается программа рассылки почты. Элемент почтовой очереди будет включать не только почтовое сообщение, но и связанный с ним конверт.

Отправитель **должен** задерживать попытки отправить почту по тому или иному адресу после связанной с этим адресом неудачи. В общем случае **следует** использовать интервал повтора не менее 30 минут, однако более изощренные и гибкие стратегии с определением причин неудачи являются более предпочтительными.

Попытки продолжаются, пока сообщение не будет передано или отправитель не откажется от дальнейших попыток (обычно отказ происходит через 4-5 дней). Параметры повторов передачи **должны** быть настраиваемыми.

Отправителю **следует** сохранять список хостов, с которыми не удается связаться и соответствующее время ожидания вместо простых попыток повтора передачи.

Обсуждение

Опыт показывает, что большинство отказов носит временный характер (например, перезагрузка хоста-получателя), поэтому рекомендуется делать две попытки передачи в течение первого часа пребывания сообщения в очереди и потом повторять попытки каждые 2 -3 часа.

Отправитель SMTP может сократить время нахождения сообщений в очереди за счет взаимодействия с принимающей стороной SMTP. В частности, если почта получена с конкретного адреса, очевидно, что доставка почты по этому адресу также возможна в данный момент. Дальнейшая оптимизация доставки может обеспечиваться путем учета множества почтовых адресов, связанных с хостом (см. 5.3.4), с учетом времени доставки и использованием ресурсов.

Отправитель SMTP может иметь большие очереди сообщений для каждого из недоступных хостов и при попытках передать все такие сообщения в каждом цикле повтора будет возникать излишняя загрузка, которая может привести к блокировке почтового демона на продолжительный период. Отметим, что SMTP в общем случае может

определить отказ только по истечении времени ожидания (минута или больше); минутный тайм-аут для соединения будет приводить к очень большим задержкам при повторении попыток для десятков и даже сотен сообщений из очереди.

Когда одно сообщение доставляется нескольким пользователям на одном хосте, **следует** передавать только одну копию. Т. е., отправителю SMTP рекомендуется использовать последовательность команд: RCPT, RCPT,... RCPT, DATA вместо последовательности: RCPT, DATA, RCPT, DATA,... RCPT, DATA. Реализация этого эффективного варианта настоятельно рекомендуется.

Подобно этому, отправитель SMTP **может** поддерживать множество одновременных исходящих почтовых транзакций для обеспечения быстрой доставки. Однако **следует** задавать некоторый предел для предотвращения излишнего расхода ресурсов на передачу почты.

Использование различных адресов на многодомных хостах рассматривается ниже.

5.3.1.2 Стратеги приема

На приемной стороне SMTP **следует** сохранять постоянное прослушивание порта SMTP. Это требуется для поддержки множества входящих TCP-соединений для SMTP. **Можно** ввести некоторые ограничения.

Реализация

Когда принимающая сторона SMTP получает почту от того или иного хоста, она может уведомить отправителя SMTP о возможности повтора для любой почты на данный хост, хранящейся в очереди.

5.3.2 Тайм-ауты SMTP

Существует два подхода при выборе времени ожидания для отправителей SMTP - (a) раздельно ограничивать время для каждой команды SMTP или (b) ограничивать время диалога SMTP в целом для каждого почтового сообщения. Для отправителей SMTP **следует** использовать вариант (a) - покомандные тайм-ауты. **Следует** также обеспечивать простой способ изменения времени ожидания, предпочтительно без перекомпиляции кода SMTP.

Обсуждение

Время ожидания является важным параметром реализации SMTP. Если тайм-аут слишком велик (или не задан вообще) отказы в соединениях Internet или ошибки в программах на приемной стороне SMTP могут привести процессы SMTP в состояние бесконечного ожидания. Если время ожидания слишком мало, это приведет к излишнему расходу ресурсов на попытки повторной передачи сообщений.

При использовании варианта (b) тайм-аут должен быть достаточно большим (например, час), чтобы можно было работать с очень большими списками рассылок. Может также потребоваться увеличение тайм-аута пропорционально размеру сообщения при работе с сообщениями большого размера. Использование большого тайм-аута с фиксированным значением может привести к двум проблемам - состояние отказа может сохраняться очень долго, а очень большие сообщения будут приводить к фиктивным тайм-аутам просто потому, что не хватило времени на передачу (это ведет к очень серьезным издержкам!).

При использовании рекомендуемого варианта (a) таймер устанавливается для каждой команды SMTP и каждого буфера передачи данных. Последнее означает, что общее время ожидания растет пропорционально размеру сообщения.

На основе опыта эксплуатации сильно загруженных почтовых хостов выработаны приведенные ниже правила, которые **следует** использовать при выборе времени ожидания:

- Изначальное сообщение 220: 5 минут

  Процесс-отправитель SMTP должен различать отказы соединений TCP от задержки при получении изначального отклика 220. Многие получатели SMTP будут воспринимать соединения TCP, но задерживать отклик 220 до тех пор, пока в системе не появится возможность обработки новой почты.

- Команда MAIL: 5 минут

- Команда RCPT: 5 минут

  Более продолжительный тайм-аут требуется при обработке списков рассылки и псевдонимов, если ее невозможно отложить, пока не будет воспринято сообщение.

- Инициирование команды DATA: 2 минуты

  Это время ожидания отклика 354 Start Input на команду DATA.

- Блок данных: 3 минуты

  Это время, в течение которого вызов TCP SEND передает блок данных.

- Прерывание команды DATA: 10 минут

  Время ожидания отклика 250 OK. Когда получатель переходит на этап завершения приема данных, он обычно выполняет операции по доставке сообщения в пользовательский почтовый ящик. Фиктивные тайм-ауты на этом этапе ведут к значительным издержкам, поскольку сообщение уже было передано целиком.

Для получателей SMTP **следует** устанавливать тайм-аут не менее 5 минут для ожидания следующей команды отправителя.

5.3.3 Надежное получение почты

Когда получатель SMTP принимает часть почты (передавая сообщение 250 OK в ответ на команду DATA), он берет на себя ответственность за доставку или трансляцию этого сообщения. Эта ответственность должна восприниматься

серьезно, т. е., **недопустимо** терять сообщения по незначительным причинам (например, в результате последующего краха хоста или предсказуемой нехватки ресурсов).

Если после восприятия сообщения возникают проблемы с его доставкой, получатель SMTP **должен** сформулировать и передать уведомление об этом. Такие уведомления **должны** передаваться с использованием пустого ("<>") пути возврата в конверте (см. параграф 3.6 в RFC 821). В качестве получателя такого уведомления **следует** указывать адрес из пути возврата в конверте или строки Return-Path:. Если этот адрес пуст ("<>"), для получателя SMTP **недопустима** передача уведомления о возникших проблемах. Если адрес содержит заданный явно маршрут, **следует** разобрать его до конечной точки.

Обсуждение

Предположим, что уведомление об ошибке должно быть передано для сообщения, принятого с "MAIL FROM:<@a,@b:user@d>". Уведомление в этом случае адресуется на: "RCPT TO:<user@d>".

Некоторые отказы при доставке после восприятия сообщения являются неизбежными. Например, причиной такого отказа может послужить невозможность проверки всех адресов доставки в команде RCPT в результате некритичной ошибки в домене или при отправке сообщения в адрес списка рассылок (см. обсуждение RCPT).

Во избежание дублирования сообщений в результате тайм-аутов получатель SMTP **должен** искать способ минимизации времени, требуемого для отклика на финальную точку, завершающую передачу сообщения. Обсуждение этой проблемы приведено в RFC 1047 [SMTP:4].

5.3.4 Надежная доставка почты

Для передачи сообщения отправитель SMTP определяет IP-адрес хоста-получателя по адресу получателя в конверте (преобразуется в адрес IP часть адреса получателя справа от знака @). При таком отображении или преобразовании могут возникать некритичные ошибки (soft error), в результате которых отправитель SMTP будет перестраивать почтовую очередь для последующей доставки сообщений, связанных с неудачной попыткой (см. 5.3.1.1).

При успешном преобразовании может быть возвращен список адресов доставки вместо единственного адреса (причиной этого может служить (a) наличие множества записей MX, (b) многодомный характер хоста или то и другое вместе). Для обеспечения надежной доставки почты отправитель SMTP **должен** предоставлять возможность попыток (и повторов) передачи по каждому из полученных адресов в соответствии с их порядком в списке, пока какая-либо из попыток не завершится успешно. Однако **могут** существовать конфигурационные ограничения на число альтернативных адресов, по которым могут осуществляться попытки доставки. В таких случаях хостам **следует** предпринять попытки хотя бы для двух адресов.

Для ранжирования адресов в списке можно использовать следующую информацию:

(1) Множество записей MX - значение записи можно использовать в качестве ключа сортировки. При существовании множества получателей с одинаковым значением и отсутствии других критериев (например, предпочтительный адрес) установки очередности отправителю SMTP **следует** использовать случайный выбор для распределения нагрузки между почтовыми серверами, обслуживающими указанную организацию (отметим, что в работе [DNS:3] предложен усовершенствованный вариант этой процедуры).

(2) Многодомный хост - хост-получатель (возможно определенный по записи MX с высшим приоритетом) может быть многодомным и в этом случае программа преобразования доменных имен будет возвращать список адресов IP. Упорядочивание адресов в списке (по уровню приоритета) является прерогативой программы-переобразователя адресов (см. параграф 6.1.3.4) и отправитель SMTP **должен** пытаться применять адреса в предложенном порядке.

Обсуждение

Хотя возможность работы с множеством альтернативных адресов является требованием, в некоторых обстоятельствах использование альтернативных адресов может быть ограничено или запрещено. Вопрос о возможности использования различных адресов многодомного хоста остается спорным. Главным аргументом в пользу работы с множеством адресов является повышение вероятности быстрой доставки и бесспорное повышение вероятности какой-либо доставки. Аргументом против такого использования является повышение расхода ресурсов[1].

5.3.5 Поддержка доменных имен

Реализации SMTP **должны** использовать механизм, определенный в параграфе 6.1 для преобразования доменных имен в IP-адреса и обратно. Это означает, что хост Internet SMTP **должен** включать поддержку Internet DNS.

В частности, отправитель SMTP **должен** поддерживать схему записей MX [SMTP:3]. Дополнительную информацию о преобразованиях доменных имен для SMTP можно найти в параграфе 7.4 работы [DNS:2].

5.3.6 Списки рассылок и псевдонимы

Использующим SMTP хостам рекомендуется поддерживать обе формы (списки - list и псевдонимы - alias) расширения адресов для организации групповой доставки. Когда сообщение доставляется или пересылается по каждому из адресов списка, адрес возврата в конверте (MAIL FROM:) **должен** заменяться на адрес администратора списка, но заголовок письма (в частности, поле From:) **должен** сохраняться неизменным.

Обсуждение

Важным вопросом для почтовой системы является доставка одного сообщения по множеству адресов, выполняемая путем преобразования или расширения (expanding) псевдо-адреса в список адресов реальных получателей. Когда сообщение передается по такому псевдоадресу (иногда такой почтовый ящик называют

[1]Отметим, что на затраты ресурсов при передаче значительное влияние оказывает выбор стратегии передачи (см. 5.3.1).

exploder - детонатор), копии письма отправляются по каждому из адресов, полученных путем преобразования. Такие псевдо-адреса делятся на псевдонимы (alias) и списки (list):

(a) Alias - псевдоним

Для расширения псевдонима принимающая программа просто заменяет псевдо-адрес в конверте на каждый включенный в псевдоним реальный адрес; остальная часть конверта и тело сообщения при этом не изменяются. Сообщение после этого доставляется или пересылается по каждому из адресов.

(b) List - список

Список адресов использует перераспределение (redistribution), а не пересылку (forwarding). Для расширения списка принимающая программа заменяет псевдоадрес в конверте реальными адресами из списка. Адрес возврата в конверте изменяется, поэтому все сообщения об ошибках доставляются администратору списка, а не отправителю сообщения, который зачастую не имеет возможности контроля списка адресов.

5.3.7 Почтовый шлюз

Передача почты между различными почтовыми средами, использующими разные форматы и протоколы, является сложной задачей, для которой еще нет должного уровня стандартизации (см. примеры в [SMTP:5a], [SMTP:5b]). Однако здесь приведены некоторые общие требования для почтовых шлюзов, обеспечивающих пересылку между Internet и другими почтовыми средами.

(A) Поля заголовков **могут** переписываться при необходимости в процессе обработки сообщений почтовыми шлюзами.

Обсуждение

Основным вопросом является интерпретация локальной части адреса, рассмотренная в параграфе 5.2.16.

Инородные почтовые системы при передаче сообщений в Internet обычно используют подмножество заголовков RFC 822, но некоторые из почтовых систем не имеют эквивалента конвертов SMTP. Следовательно, когда сообщение покидает среду Internet, может потребоваться включение информации из конверта SMTP в заголовок сообщения. Возможным решением является создание новых полей заголовка для передачи информации из конверта (например, X-SMTP-MAIL: и X-SMTP-RCPT:); однако такое решение может потребовать изменений в почтовых программах чужой среды.

(B) При пересылке сообщений в среду Internet или из нее, шлюз **должен** подготовить свою строку Received:, но **недопустимо** менять содержимое других строк Received: в полученном заголовке.

Обсуждение

Это требование является частью общих правил для строки Received:, рассмотренных в параграфе 5.2.8 и приведено здесь только для напоминания.

Поля Received: сообщений из другой среды могут не соответствовать в точности RFC 822. Однако, наиболее важным применением строк Received: является обнаружение почтовых отказов и такая отладка может быть испорчена шлюзами, которые пытаются править строки Received:.

Для шлюзов настоятельно рекомендуется указывать среду и протокол в предложениях "via" строки Received:, доставляемой шлюзом.

(C) Со стороны Internet шлюзу **следует** принимать все допустимые форматы адресов в командах SMTP и заголовках RFC 822, а также все допустимые сообщения RFC 822. Хотя шлюз должен воспринимать явно указанные маршруты RFC 822 (формат "@...:") в заголовке RFC 822 или в конверте, шлюз не обязан действовать на маршруте от отправителя (см. 5.2.6 и 5.2.19.

Обсуждение

Часто возникает искушение ограничить диапазон адресов, воспринимаемых почтовым шлюзом для упрощения трансляции адресов в форматы другой среды. Такая практика основывается на предположении, что пользователи почты имеют контроль над всеми адресами, по которым их почтовые программы шлют сообщения почтовому шлюзу. На практике, однако, пользователи имеют ограниченный контроль над адресами, которые они в конечном итоге используют, поскольку почтовые программы могут свободно менять адреса в любой допустимый формат RFC 822.

(D) Шлюз **должен** гарантировать, что все поля заголовков сообщений, пересылаемых в Internet, соответствуют почтовым требованиям Internet. На практике все адреса в полях From:, To:, Cc: и т. п. должны быть преобразованы (при необходимости) в соответствии с требованиями синтаксиса RFC 822.

(E) Алгоритму трансляции, используемому для преобразования почты Internet в другие почтовые системы, **следует** пытаться обеспечить гарантии доставки сообщений об ошибках из чужой среды по пути возврата из конверта SMTP, а не отправителю, указанному в поле From: сообщения RFC 822.

Обсуждение

Списки рассылок Internet обычно помещают адрес владельца списка в конверт, но указывают реального отправителя в поле From:. Этот подход представляется разумным - ответ на письмо приходит его реальному отправителю, а сообщения об ошибках - администратору списка, который может исправить связанные со списком ошибки.

(F) Подобно сказанному, при пересылке почты из чужой среды в Internet шлюзу **следует** установить в конверте путь возврата в соответствии с требованиями возврата сообщений об ошибках для инородной среды.

5.3.8 Максимальный размер сообщения

Почтовая программа **должна** обеспечивать возможность передачи и приема сообщений, размером не менее 64 кбайт (включая заголовок); желательно обеспечивать возможность работы с более крупными сообщениями.

Обсуждение

Хотя SMTP не задает максимальный размер сообщения, во многих системах этот размер ограничен.

Фактически в Internet обеспечивается гарантия передачи сообщений размером 64 кбайт. Однако электронная почта используется для решения различных задач и может потребоваться передача больших сообщений. Например, электронную почту зачастую используют взамен FTP для передачи ASCII-файлов, которые могут содержать целый документ. В результате сообщения размером 1 Мбайт и более не являются чем-то, из ряда вон выходящим.

5.4 Требования к SMTP

Функция	*Параграф*	*Требования*
Получатель SMTP:		
Реализация VRFY	5.2.3	Обязательно
Реализация EXPN	5.2.3	Следует
Возможность настройки EXPN, VRFY	5.2.3	Возможно
Реализация SEND, SOML, SAML	5.2.4	Возможно
Проверка параметра HELO	5.2.5	Возможно
Отбрасывание сообщений с некорректным HELO	5.2.5	Недопустимо
Допустимость явного синтаксиса source-route в среде	5.2.6	Обязательно
Поддержка postmaster	5.2.7	Обязательно
Обработка RCPT при получении (кроме списков)	5.2.7	Возможно
Значительна задержка откликов RCPT	5.2.7	Недопустимо
Добавление строки Received:	5.2.8	Обязательно
Строка Received: включает доменные литералы	5.2.8	Следует
Изменение предыдущей строки Received:	5.2.8	Недопустимо
Передача информации о пути возврата (Rerurn-Path)	5.2.8	Обязательно
Поддержка пустых обратных путей	5.2.9	Обязательно
Передача только официальных кодов отклика	5.2.10	Следует
Передача текста из RFC 822	5.2.10	Следует
Удаление *.* для прозрачности	5.2.11	Обязательно
Восприятие и распознавание своих доменных имен	5.2.17	Обязательно
Генерация сообщений об ошибках для сообщений об ошибках	5.3.1	Недопустимо
Сохранение состояния прослушивания для порта SMTP	5.3.1.2	Следует
Ограничение числа одновременно принимаемых сообщений	5.3.1.2	Возможно
Ожидание не менее 5 мин. перед следующей командой отправителя	5.3.2	Следует
Предотвращение сбоев при доставке после сообщений “250 OK”	5.3.3	Недопустимо
Передача уведомлений об ошибках после получения	5.3.3	Обязательно
Передача с использованием пустого пути возврата	5.3.3	Обязательно
Передача по пути возврата конверта	5.3.3	Следует
Передача по пустому адресу	5.3.3	Недопустимо
Вырезание заданного явно source-route	5.3.3	Следует
Минимизация задержки восприятия (RFC 1047)	5.3.3	Обязательно
Отправитель SMTP:		
Канонизированные доменные имена в MAIL, RCPT	5.2.2	Обязательно
Реализация SEND, SOML, SAML	5.2.4	Возможно
Передача корректного основного имени в HELO	5.2.5	Обязательно
Передача явного маршрута в RCPT TO:	5.2.6	Не следует
Использование для определения действия только кода отклика	5.2.10	Обязательно
Использование только старшей цифры кода отклика	5.2.10	Следует
Добавление *.* для прозрачности	5.2.11	Обязательно
Повторение передачи после некритичных ошибок	5.3.1.1	Обязательно
Задержка перед повтором	5.3.1.1	Обязательно
Настраиваемые параметры повторной передачи	5.3.1.1	Обязательно
Одна попытка для каждого хоста-получателя в очереди доставки	5.3.1.1	Следует
Множество RCPT для одной команды DATA	5.3.1.1	Следует
Поддержка одновременных транзакций	5.3.1.1	Возможно
Ограничение числа	5.3.1.1	Следует
Тайм-аут для всех операций	5.3.1	Обязательно
Тайм-аут для каждой команды независимо	5.3.2	Следует
Проста настройка времени ожидания	5.3.2	Следует
Рекомендуемые значения	5.3.2	Следует
Пробовать альтернативные адреса по порядку	5.3.4	Обязательно
Конфигурационные ограничения для числа адресов	5.3.4	Возможно
Пробовать по крайней мере два адреса	5.3.4	Следует
Распределение нагрузки при равных значениях MX	5.3.4	Следует
Использование DNS	5.3.5	Обязательно
Поддержка записей MX	5.3.5	Обязательно
Использование WKS при обработке MX	5.2.12	Не следует
Пересылка почты:		
Изменение существующих полей заголовка	5.2.6	Не следует
Реализация функций трансляции (RFC 821, параграф 3.6)	5.2.6	Возможно
Если нет, доставка в домен RHS	5.2.6	Следует
Интерпретация локальной части адреса (local-part)	5.2.16	Недопустимо
Списки рассылок и псевдонимы:		
Поддержка тех и других	5.3.6	Следует
Отчеты для локального администратора об ошибках в списке рассылки	5.3.6	Обязательно

Функция	*Параграф*	*Требования*
Почтовые шлюзы:		
Встраивание чужого почтового маршрута в локальную часть	5.2.16	Возможно
Переписывание при необходимости полей заголовка	5.3.7	Возможно
Вставка строки Received: в начало	5.3.7	Обязательно
Изменение существующих строк Received:	5.3.7	Недопустимо
Полное восприятие RFC 822 со стороны Internet	5.3.7	Следует
Работа на явном маршруте RFC 822	5.3.7	Возможно
Передача в сторону Internet только корректных RFC 822	5.3.7	Обязательно
Доставка сообщений об ошибках по адресу в конверте	5.3.7	Следует
Установка пути возврата в конверте из пути возврата ошибки	5.3.7	Следует
Пользовательский агент - RFC 822:		
Пользователь может вводить адрес <route>	5.2.6	Не следует
Поддержка пол Content Type (RFC 1049)	5.2.13	Возможно
Использование 4-значного года	5.2.14	Следует
Генерация временных зон в форме чисел	5.2.14	Следует
Восприятие всех временных зон	5.2.14	Обязательно
Использование нечисловых временных зон RFC 822	5.2.14	Обязательно
Опускание фразы перед route-addr	5.2.15	Возможно
Восприятие и разборка доменных имен dot.dec.	5.2.17	Обязательно
Восприятие всех форматов адреса RFC 822	5.2.18	Обязательно
Генерация адресов в некорректных форматах (RFC 822)	5.2.18	Недопустимо
Полные доменные имена в заголовке	5.2.18	Обязательно
Создание явного маршрута в заголовке	5.2.19	Не следует
Восприятие явного маршрута в заголовке	5.2.19	Обязательно
Прием/передача сообщений не менее 64 кбайт.	5.3.8	Обязательно

6. Служебные протоколы

6.1 Трансляция доменных имен

6.1.1 Введение

Каждый хост **должен** реализовать программу преобразования (resolver) для DNS и механизм использования этой программы для преобразования имен хостов в адреса IP и обратно [DNS:1, DNS:2].

В дополнение к DNS хост **может** поддерживать механизм преобразования имен на основе поиска в локальной таблице имен хостов Internet (см. параграф 6.1.3.8).

Обсуждение

Трансляция имен хостов Internet поначалу осуществлялась путем поиска в локальной копии списка **всех** хостов. Эти списки со временем стали слишком велики для обновления и распространения в прежней манере, поэтому была разработана служба доменных имен - DNS.

DNS использует распределенную базу данных, которая служит прежде всего для преобразования имен хостов в их адреса и наоборот. Требуется реализация программ DNS. Система доменных имен DNS состоит из двух различных частей - серверов имен и программ преобразования, которые иногда называют резольверами (resolver), хотя при реализации эти две части могут объединять в целях повышения эффективности [DNS:2].

Серверы доменных имен сохраняют полномочные (authoritative) данные для отдельных частей распределенной базы данных и отвечают на соответствующие запросы. Программы преобразования имен запрашивают у серверов данные по запросам пользовательских процессов. На каждом хосте, следовательно, должна быть программа преобразования (DNS resolver); а на некоторых хостах нужен также сервер имен. Поскольку ни один из серверов имен не содержит всей информации, в общем случае для преобразования имени в адрес (или наоборот) может потребоваться получение информации от нескольких серверов имен.

6.1.2 Общие вопросы

Разработчики должны внимательно ознакомиться с документами [DNS:1] и [DNS:2], содержащими описания теории, протоколов и реализации системы доменных имен с учетом реального опыта.

6.1.2.1 Записи RR с TTL=0: RFC 1035, 3.2.1

Серверы имен и преобразователи DNS **должны** корректно обрабатывать RR с нулевым значением TTL, возвращая клиенту запись RR, но не кэшируя ее.

Обсуждение

Нулевое значение TTL говорит о том, что запись RR можно использовать только для выполняемой транзакции и не следует кэшировать; это очень полезно для часто меняющихся данных.

6.1.2.2 Значения QCLASS: RFC 1035, 3.2.5

Запросы с QCLASS=* **не следует** использовать, если запрашивающая сторона не просматривает данные из нескольких классов. В частности, если запрашивающая сторона интересуется только типами данных Internet, **необходимо** использовать QCLASS=IN.

6.1.2.3 Неиспользуемые поля: RFC 1035, 4.1.1

Неиспользуемые поля запросов и откликов **должны** иметь нулевые значения.

6.1.2.4 Сжатие: RFC 1035, 4.1.4

Серверы имен **должны** использовать в откликах сжатие данных.

Обсуждение

Сжатие позволяет избавиться от лишних дейтаграмм UDP, как описано в параграфе 6.1.3.2.

6.1.2.5 Запрет на использование конфигурационных сведений: RFC 1035, 6.1.2

Рекурсивные серверы имен и полнофункциональные преобразователи используют некую конфигурационную информацию, содержащую сведения о расположении корневых и локальных серверов имен. Для реализаций программ **недопустимо** включение такой информации в отклики.

Обсуждение

Многие разработчики считают удобным сохранять такие данные, как будто они кэшируются, но иногда пренебрегают обеспечением запрета на включение этих «кэшируемых» данных в отклики. Некорректность такого рода информации может привести к серьезным проблемам в Internet.

6.1.3 Частные вопросы

6.1.3.1 Реализация программы преобразования

Для программ преобразования имен (name resolver) **следует** обеспечивать поддержку одновременных запросов, если хост поддерживает одновременные процессы.

При разработке программ преобразования **могут** выбраны различные модели - полнофункциональный преобразователь (full-service resolver) или оконечный (stub) преобразователь.

(A) Полнофункциональная программа

Полнофункциональный преобразователь обеспечивает полный сервис преобразования имен и может работать при коммуникационных сбоях, отказах отдельных серверов имен, а также способен определить корректный сервер для данного имени и т. д. К таким преобразователям предъявляются следующие требования:

- **должны** поддерживаться функции локального кэширования, позволяющие избавиться от лишних запросов к удаленным серверам при повторении идентичных запросов; для записей в локальном кэше **должно** задаваться время жизни;
- **следует** обеспечивать возможность настройки конфигурационных параметров при старте программы с помощью сведений, указывающих на разные корневые серверы и различные серверы имен для локального домена; это обеспечивает доступ преобразователя ко всему пространству имен в нормальном режиме и возможность преобразования локальных имен при отсутствии связи с Internet.

(B) Оконечная программа - боковик (Stub Resolver)

Работа оконечных преобразователей основана на обращениях к рекурсивным серверам имен в подключенной сети или соседней сети. Такая схема позволяет хосту передать нереализованные функции преобразования адресов серверу имен на другом хосте. Зачастую такое решение используется для небольших хостов (например, ПК) и рекомендуется также для случаев, когда хост является одной из нескольких рабочих станций локальной сети, поскольку такое решение позволяет всем станциям ЛВС использовать кэш рекурсивного сервера имен и, следовательно, снижает число запросов, экспортируемых локальной сетью.

В минимальном варианте stub-программа **должна** быть способна перенаправить свои запросы к резервным рекурсивным серверам имен. Отметим, что рекурсивные серверы имен разрешены для ограничения числа отправителей запросов, которые будут обслуживаться, поэтому администратор хоста должен убедиться в наличии сервиса. Оконечные преобразователи **могут** использовать кэширование и в таком случае **должны** задать время жизни информации в кэше.

6.1.3.2 Транспортные протоколы

Программы преобразования и рекурсивные серверы DNS **должны** поддерживать протокол UDP (**следует** также поддерживать TCP) для передачи запросов (не для переноса зон). Отметим также, что при передаче запроса, не относящегося к передаче зоны, сначала **должен** использоваться протокол UDP. Если раздел Answer (ответ) в отклике отсечен и запрашивающая программа поддерживает TCP, **следует** повторить запрос с использованием TCP.

Серверы DNS **должны** обслуживать запросы UDP; **следует** обслуживать также TCP-запросы. Сервер имен **может** ограничить ресурсы для обслуживания запросов TCP, но **не следует** отказываться от обработки таких запросов лишь потому, что они могли быть обслужены по протоколу UDP.

Недопустимо сохранять (кэшировать) усеченные отклики и использовать их впоследствии в качестве нормальных откликов.

Обсуждение

Протокол UDP является предпочтительным для передачи запросов, поскольку UDP порождает меньше пакетов и не так сильно загружает канал связи. Использование UDP играет очень важную роль для загруженных серверов (особенно для корневых). UDP также обеспечивает дополнительную устойчивость, поскольку преобразователь может сделать UDP-запросы к нескольким серверам по цене одного запроса TCP.

Ситуации с усекновением откликов DNS весьма редки в современной среде Internet, но все-таки реальны. Предсказать такие ситуации невозможно, поскольку их возникновение зависит от данных. Эта зависимость включает число записей RR в ответе, размер каждой записи RR и реализацию алгоритма сжатия имен. Обычно считается, что отсечения списков NS и MX не должно происходить для ответов, содержащих не более 15 записей RR.

Возможность использования усеченных ответов зависит от приложения. Почтовым программам недопустимо использовать усеченные записи MX, поскольку это может привести к возникновению почтовых петель.

Практика показала возможность использования UDP в большинстве случаев. Серверы имен должны использовать компрессию данных в откликах. Преобразователи должны отличать ответы с усеченным дополнительным разделом (Additional), который содержит только добавочную информацию, от случаев усекновения раздела Answer (ответ) - в случае отсечения записей MX ответы просто нельзя использовать в почтовых программах. Администраторы должны ограничиваться разумным числом первичных имен в списках серверов имен, вариантов MX и т. п.

Однако очевидно, что новые типы записей DNS, которые появятся определены в будущем, могут содержать объем информации, превышающий 512-байтовый предел для UDP, и, следовательно, потребуют использования протокола TCP. Таким образом, программы службы доменных имен должны сегодня поддерживать протокол TCP как резерв для UDP, понимая, что в будущем использование TCP неизбежно.

Серверы имен и преобразователи на основе частного соглашения **могут** применять TCP для всего трафика между собой. Для передачи зон **должен** использоваться протокол TCP.

Сервер DNS **должен** иметь достаточно внутренних ресурсов для продолжения обработки запросов UDP во время ожидания отклика или при переносе зоны через открытое соединение TCP [DNS:2].

Сервер **может** поддерживать запросы UDP, для доставки которых используются групповые или широковещательные адреса IP. Однако для запросов с групповым адресом **недопустимо** устанавливать бит RD[1] и запросы в групповых или широковещательных пакетах с установленным битом RD **должны** игнорироваться сервером имен. Хостам, передающим запросы DNS с использованием широковещательного или группового адреса **следует** передавать их таким способом только в редких случаях - поместив в кэш IP-адрес(а) из отклика, хост в дальнейшем может передавать нормальные запросы по этому адресу.

Обсуждение

Широковещательные и (особенно) групповые запросы могут обеспечивать способ поиска серверов имен в соседних сетях при отсутствии информации об адресах таких серверов. Однако частое использование таких запросов может привести к ненужной и чрезмерной загрузке как сети, так и серверов имен.

6.1.3.3 Эффективное использование ресурсов

Приведенные ниже требования к серверам имен и преобразователям очень важны для нормальной работы Internet в целом, особенно для ситуаций, когда серверы DNS постоянно вовлечены в работу автоматических серверов верхних уровней (например, почтовых).

(1) Преобразователь **должен** обеспечивать управление повторной передачей для того, чтобы не расходовать излишней полосы каналов связи; кроме того, **должно** ограничиваться количество ресурсов, потребляемых для отклика на один запрос (конкретные рекомендации можно найти на страницах 43-44 работы [DNS:2]).

(2) После того, как запрос был передан несколько раз без отклика, программа **должна** прекратить попытки и сообщить приложению о некритичной ошибке.

(3) Всем серверам DNS и преобразователям **следует** кэшировать временные неполадки с периодом ожидания в несколько минут.

Обсуждение

Это будет предотвращать избыточный трафик DNS от приложений, которые немедленно повторяют запрос при получении информации о некритичной ошибке в нарушение требований параграфа 2.2 настоящего документа.

(4) Всем серверам DNS и преобразователям **следует** кэшировать негативные отклики, которые говорят, что заданное имя не существует (в соответствии с требованиями [DNS:2]).

(5) При повторении серверами DNS и резольверами запросов UDP **следует** использовать экспоненциальный алгоритм изменения периода повторов, для которого **следует** задавать верхнюю и нижнюю границу.

Реализация

Следует использовать измеренные значения RTT[2] и вариаций (если возможно) для расчета начального периода повтора запросов. Если такая информация недоступна, следует использовать по умолчанию период повтора не менее 5 секунд. Реализации могут ограничивать интервал повторной передачи, но эта граница должна превышать удвоенное значение максимального времени жизни сегмента в Internet с учетом задержки при обработке на сервере имен.

(6) Когда сервер или резольвер получает ответ Source Quench для переданного запроса, **следует** приложить усилия для снижения частоты запросов к этому серверу в ближайшем будущем; Сервер **может** игнорировать отклики Source Quench, получаемые в результате передачи дейтаграмм-откликов.

Реализация

Рекомендуется для снижения частоты запросов к серверу попытаться использовать другой сервер, если таковые имеются. Другим вариантом является увеличение периода повторов для запросов к серверу.

6.1.3.4 Многодомные хосты

Когда при преобразовании имени в адрес функция встречает многодомный хост, **следует** ранжировать или сортировать его адреса, используя информацию о номерах непосредственно подключенных сетей и любые другие сведения о производительности, а также предысторию.

Обсуждение

[1]Recursion Desired - желательна рекурсия.
[2]Round-trip time - период кругового обхода. *Прим. перев.*

Различные адреса многодомного хоста обычно подразумевают различные пути Internet, из которых некоторые могут быть более предпочтительными с точки зрения производительности, надежности или административных ограничений. Общего алгоритма определения наилучшего пути не существует. Рекомендуется принимать такие решения на основе локальных конфигурационных параметров, установленных системным администратором.

Реализация

Предложенная ниже схема хорошо показала себя на практике:

(a) конфигурационные параметры хоста включают Network-Preference List - простой список сетей в порядке предпочтения. Этот список может быть пустым, если предпочтения отсутствуют.

(b) Когда имя хоста преобразуется в список адресов IP, эти адреса сортируются по номерам сетей в том же порядке, который задан в Network-Preference List. IP-адреса, отсутствующие в списке предпочтений, помещаются в конец сортированного списка.

6.1.3.5 Возможности расширения

Программы DNS **должны** поддерживать все общеизвестные, не зависящие от классов форматы [DNS:2]; **следует** при разработке программ минимизировать возможные издержки при введении новых общеизвестных типов и локальных экспериментах с нестандартными типами.

Обсуждение

Типы и классы данных, используемые DNS, постоянно изменяются - появляются новые типы, а старые удаляются или определяются заново. Введение новых типов должно зависеть только от правил компрессии доменных имен в сообщениях DNS и трансляции между печатным (master-файл) и внутренним форматом записей RR[1].

Сжатие основано на знании формата данных внутри отдельной записи RR. Поэтому компрессия должна использоваться только для содержимого общеизвестных и не зависящих от класса записей RR, но недопустима для зависящих от класса RR или записей, не относящихся к общеизвестным. Имя владельца RR всегда подходит для сжатия.

Сервер имен может получить (путем переноса зоны) записи RR, которые сервер не умеет преобразовывать в печатный формат. Подобную информацию преобразователь может получить в результате запроса. Для корректной работы такие данные нужно предварительно сохранить, поскольку программы DNS не могут использовать текстовые форматы для внутреннего хранения.

DNS определяет синтаксис доменных имен лишь в самом общем виде - как строку меток из 8-битовых символов длиной до 63 символов каждая с общей длиной строки не более 255 октетов. Частные приложения DNS могут дополнительно ограничивать синтаксис, хотя развертывание DNS привело к появлению приложений, разрешающих имена более общих типов. В частности, параграф 2.1 данного документа несколько либерализует синтаксис имен хостов по сравнению с требованиями RFC 952 [DNS:4].

6.1.3.6 Состояние типов RR

Сервер имен **должен** быть способен загружать все типы RR (за исключением MD и MF) из конфигурационных файлов. Типы MD и MF являются устаревшими и **недопустимы** для использования (в частности, **недопустимо** загружать эти типы из конфигурационных файлов).

Обсуждение

RR типов MB, MG, MR, NULL, MINFO и RP являются экспериментальными и приложения, использующие DNS, не должны ожидать поддержки этих типов в любом домене. Многие из этих типов могут быть переопределены.

Типы TXT и WKS редко используются сайтами Internet, поэтому для большинства доменов не следует надеяться на существование этих записей.

6.1.3.7 Устойчивость

Программам DNS иной раз приходиться работать в средах, где корневые серверы или иные важные серверы недоступны в результате проблем с сетевыми соединениями или по иным причинам. В этой ситуации серверы имен и преобразователи DNS **должны** продолжать предоставление сервиса для доступной части пространства имен, выдавая информацию о временной недоступности остальной части сети.

Обсуждение

Хотя DNS используется преимущественно в сетях, соединенных с Internet, должна обеспечиваться возможность использования и в изолированных системах. Следовательно, работоспособность программ не должна зависеть от возможности доступа к корневым серверам при обслуживании запросов для локальных имен.

6.1.3.8 Локальный список хостов

Обсуждение

Хост может использовать локальный список хостов в качестве резервирования или в дополнение к DNS. При таком варианте возникает вопрос очередности использования; наиболее гибким является выбор предпочтений с помощью конфигурационных опций.

Обычно содержимое списка задается локально для каждого хоста. Однако общедоступные списки хостов Internet поддерживаются Сетевым информационным центром DDN (DDN NIC) в формате, заданном [DNS:4]. Эти таблицы можно загрузить с DDN NIC, используя протокол, описанный в работе [DNS:5]. Следует отметить, что эти таблицы содержат лишь малую часть хостов Internet. Хостам, использующим протокол [DNS:5] для получения списков DDN NIC, следует применять команду VERSION для проверки наличия обновлений в таблице и только после этого

[1]Resource Record - запись о ресурсе.

запрашивать всю таблицу с помощью команды ALL. Идентификатор VERSION следует трактовать как произвольную строку и проверять ее только на совпадение, не пытаясь найти порядковый номер версии.

Таблица хостов DDN NIC включает административную информацию (например, сети и шлюзы), которая не требуется для работы хостов и, следовательно, не включается в базу данных DNS. Однако в будущем значительная часть этой информации может быть включена в DNS. И наоборот, DNS обеспечивает важные службы (в частности, записи MX), которые недоступны в таблице хостов DDN NIC.

6.1.4 Пользовательский интерфейс DNS

6.1.4.1 Администрирование DNS

Этот документ посвящен вопросам архитектуры и реализации программ для хостов и не связан с администрированием и поддержкой. Однако вопросы администрирования весьма важны в DNS, поскольку ошибки в отдельных сегментах большой распределенной базы данных могут повлиять на работу множества сайтов. Вопросы администрирования подробно рассматриваются в [DNS:6] и [DNS:7].

6.1.4.2 Интерфейс DNS - пользователь

Хост **должен** обеспечивать интерфейс с DNS для всех прикладных программ на хосте. Этот интерфейс обычно направляет все запросы системному процессу для выполнения функций преобразования имен [DNS:1, 6.1:2].

По минимуму базовый интерфейс **должен** поддерживать запросы для всей информации заданного типа и класса, связанной с указанным именем, и **должен** возвращать всю запрошенную информацию или код возникшей ошибки. При отсутствии ошибок базовый интерфейс возвращает полностью запрошенную информацию, не меняя ее, поэтому при появлении новых типов и классов базовый интерфейс не потребуется обновлять.

Обсуждение

Индикация некритичных ошибок является существенной частью интерфейса, поскольку не всегда возможно воспринять информацию непосредственно от DNS (см. параграф 6.1.3.3).

Хост **может** обеспечивать иные интерфейсы DNS, основанные на отдельных функциях, преобразовании неупорядоченных данных о домене в удобную форму и т. п. В частности, хост **должен** обеспечивать интерфейс DNS для выполнения преобразований имен хостов в адреса и наоборот.

6.1.4.3 Возможности сокращений

Пользовательский интерфейс **может** обеспечивать поддержку сокращенного ввода широко распространенных имен. Хотя определение такого метода выходит за пределы спецификации DNS, здесь сформулированы некоторые правила, которые позволят обеспечить доступ ко всему пространству имен DNS и предотвратят излишнюю загрузку ресурсов Internet. При использовании сокращений:

(a) **должно** быть некое соглашение для обозначения полностью введенных имен (в этом случае обработка сокращений отключается). Общепринятым методом является точка в конце полного имени.

(b) Преобразование сокращенных имен **должно** выполняться только один раз и в том контексте, в котором имя было введено.

Обсуждение

Например, при использовании сокращений в почтовой программе, сокращенное им должно преобразовываться в полное и сохраняться в почтовой с пометкой полноты. В противном случае, сокращенное им может быть преобразовано еще раз при поиске в списке почтовой системы или при многократной канонизации.

Существуют два общепринятых метода сокращений:

(1) Псевдонимы интерфейсного уровня

Псевдонимы интерфейсного уровня реализуются в виде списка пар псевдоним - доменное имя. Эти списки могут создаваться в масштабе хоста или отдельно для каждого пользователя и для каждой функции может быть создан свой список (например, один список для трансляции имен в адреса, а другой - для почтовых доменов). Когда пользователь вводит имя, интерфейс пытается найти в списке подходящий псевдоним и при удачном поиске меняет сокращенное им на полное.

Отметим, что псевдонимы интерфейсного уровня полностью отличаются от механизма CNAME - интерфейсные псевдонимы имеют лишь локальную значимость, CNAME обеспечивает псевдонимы в масштабе Internet, реализуемые на уровне серверов DNS.

(2) Списки поиска

Список поиска концептуально реализуется, как упорядоченный список доменных имен. Когда пользователь вводит имя, доменные имена из списка поиска подставляются как суффикс для заданного пользователем имени одно за другим, пока не будет найдено доменное имя с желаемыми ассоциированными данными или не закончится список. Списки поиска часто содержат имена родительского домена для хоста или других доменов-предков. Зачастую списки поиска создаются для отдельных пользователей или процессов.

Следует обеспечивать администратору возможность запрета поиска по спискам - такой запрет в некоторых случаях может потребоваться для предотвращения злоупотреблений с DNS.

Существует опасность, что механизм поиска по списку будет приводить к излишним запросам для корневых серверов в процессе поиска полного имени для заданного пользователем сокращения. Механизм поиска по списку **должен** обеспечивать маркировку полной формы имен и **следует** также реализовать оба предложенных ниже способа предотвращения излишних запросов:

(a) реализовать в локальном преобразователе/сервере имен кэширование негативных откликов (см. 6.1.3.3);

(b) средство расширения имен по списку должно обращаться к нелокальным серверам только при наличии одной или двух точек в сгенерированном доменном имени.

Обсуждение

Реализация этого требования позволяет избежать ненужных задержек при проверке списка и снизить число запросов к корневым серверам и серверам верхних уровней. Например, если пользователь ввел имя "X" и список поиска содержит в качестве компоненты корневой сервер, при поиске обращение к корневому серверу будет предшествовать переходу к следующему элементу списка. В результате число запросов к корневому серверу будет неоправданно возрастать.

Кеширование негативных откликов имеет ограниченный эффект при первом использовании имени. Правило внутренних точек проще в реализации, но может блокировать поиск для некоторых имен верхних уровней.

6.1.5 Требования к DNS

Функция	*Параграф*	*Требования*
Общие вопросы:		
Преобразование имени в адрес	6.1.1	Обязательно
Преобразование адреса в им	6.1.1	Обязательно
Поддержка преобразований с использованием таблицы хостов	6.1.1	Возможно
Корректна обработка RR с TTL=0	6.1.2.1	Обязательно
Необязательность использования QCLASS=*	6.1.2.2	Следует
Использование QCLASS=IN для Internet	6.1.2.2	Обязательно
Нулевые значения неиспользуемых полей	6.1.2.3	Обязательно
Использование сжатия в откликах	6.1.2.4	Обязательно
Включение конфигурационной информации в отклики	6.1.2.5	Недопустимо
Поддержка всех хорошо известных, независимых от класса типов	6.1.2.5	Обязательно
Легко расширяемый список типов	6.1.2.5	Следует
Загрузка всех типов RR (кроме MD и MF)	6.1.2.6	Обязательно
Загрузка типа MD или MF	6.1.2.6	Недопустимо
Работоспособность при недоступности корневого сервера и т. п.	6.1.2.7	Обязательно
Программа преобразования (resolver):		
Поддержка множества одновременных запросов	6.1.3.1	Следует
Полнофункциональный резольвер:	6.1.3.1	Возможно
локальное кэширование	6.1.3.1	Обязательно
старение данных в локальном кэше	6.1.3.1	Обязательно
настройка конфигурации при старте	6.1.3.1	Следует
Оконечный преобразователь (stub):	6.1.3.1	Возможно
использование резервных серверов имен (рекурсия)	6.1.3.1	Обязательно
локальное кэширование	6.1.3.1	Возможно
старение данных в локальном кэше	6.1.3.1	Обязательно
Поддержка многодомных удаленных хостов:		
сортировка адресов в порядке предпочтения	6.1.3.4	Следует
Транспортные протоколы:		
Поддержка запросов UDP	6.1.3.2	Обязательно
Поддержка запросов TCP	6.1.3.2	Следует
Передача запросов сначала с помощью UDP	6.1.3.2	Обязательно[1]
Использование TCP, если UDP-запросы отвергнуты	6.1.3.2	Следует
Сервер имен ограничивает ресурсы для запросов по TCP	6.1.3.2	Возможно
“Наказание” для неоправданных запросов TCP	6.1.3.2	Не следует
Использование “усеченных” данных, как нормальных	6.1.3.2	Недопустимо
Частное соглашение на использование только TCP	6.1.3.2	Возможно
Использование TCP для переноса зон	6.1.3.2	Обязательно
Использование TCP не блокирует запросов UDP	6.1.3.2	Обязательно
Поддержка групповых и широковещательных запросов	6.1.3.2	Возможно
Бит RD в запросе установлен	6.1.3.2	Недопустимо
Бит RD игнорируется сервером для групповых и широковещательных запросов	6.1.3.2	Обязательно
Редкая передача только для получения адресов серверов имен	6.1.3.2	Следует
Использование ресурсов:		
Управление передачей в соответствии с [DNS:2]	6.1.3.3	Обязательно
Конечные границы для запроса	6.1.3.3	Обязательно
Сообщение о некритичной ошибке после нескольких неудач	6.1.3.3	Обязательно
Кэширование временных отказов	6.1.3.3	Следует
Кэширование негативных откликов	6.1.3.3	Следует
Повторы с экспоненциальным периодом	6.1.3.3	Следует
Верхняя и нижняя граница	6.1.3.3	Следует
Клиент обрабатывает Source Quench	6.1.3.3	Следует
Сервер игнорирует Source Quench	6.1.3.3	Возможно

[1]Если не существует частного соглашения между сервером и резольвером на использование только TCP.

Функция	*Параграф*	*Требования*
Пользовательский интерфейс:		
Все программы имеют доступ к интерфейсу DNS	6.1.4.2	Обязательно
Возможность запросить всю информацию для данного имени	6.1.4.2	Обязательно
Возврат полной информации или сообщения об ошибке	6.1.4.2	Обязательно
Специальные интерфейсы	6.1.4.2	Возможно
Трансляция им <-> адрес	6.1.4.2	Обязательно
Возможности сокращений:	6.1.4.3	Возможно
Соглашение для полных имен	6.1.4.3	Обязательно
Однократное преобразование	6.1.4.3	Обязательно
Преобразование в приемлемом контексте	6.1.4.3	Обязательно
Список поиска:	6.1.4.3	Возможно
Администратор может запретить	6.1.4.3	Следует
Предотвращение излишних корневых запросов	6.1.4.3	Обязательно
Оба метода	6.1.4.3	Следует

6.2 Инициализация хоста

6.2.1 Введение

В этом разделе описана инициализация программ хоста через подключенную сеть или (в более общем случае) через Internet. Такие операции требуются для бездисковых станций, но могут использоваться и для хостов с дисками. Для бездисковых хостов процесс инициализации называется загрузкой из сети (network booting) и управляется программой загрузки (bootstrap), хранящейся в ПЗУ (boot ROM).

При инициализации бездисковых хостов через сеть выделяют две фазы:

(1) настройка уровня IP.

Бездисковые станции зачастую неспособны хранить параметры своей конфигурации, поэтому должна обеспечиваться возможность динамического получения таких параметров для поддержки остальных этапов процесса загрузки хоста. Конфигурационные параметры должны включать по крайней мере адреса IP для данного хоста и сервера загрузки. Для поддержки загрузки через маршрутизатор требуется также маска подсети и список используемых по умолчанию шлюзов.

(2) Загрузка системного кода для хоста.

На этапе загрузки операционной системы используется подходящий протокол передачи файлов для копирования программного кода через сеть с сервера загрузки.

Для хостов с дисками может использоваться первая фаза - динамическая настройка конфигурации. Это важно для микрокомпьютеров, наличие дисковода в которых позволяет ошибочно дублировать параметры конфигурации с других хостов. Кроме того, инсталляция новых хостов значительно упрощается за счет возможности получения конфигурационных параметров с центрального сервера - это экономит время администратора и избавляет от ненужных ошибок.

6.2.2 Требования

6.2.2.1 Динамическая настройка конфигурации

Для динамической настройки поддерживается целый ряд протоколов и служб.

- Сообщения ICMP Information Request/Reply

  Эта устаревшая пара сообщений предназначена для обеспечения хостам возможности определения номера сети. К сожалению, эти сообщения полезны только для хостов, которые уже знают свой номер (связанную с хостом часть адреса IP).

- Протокол обратного преобразования адресов RARP [BOOT:4]

  RARP является протоколом канального уровня для широковещательных сред, который позволяет определять адрес IP по адресу канального уровня. К сожалению, RARP не работает через шлюзы IP и, следовательно, требует наличия сервера RARP в каждой сети. Другой конфигурационной информации протокол RARP не обеспечивает.

- Сообщения ICMP Address Mask Request/Reply

  Эти сообщения ICMP позволяют хосту определить адресную маску для отдельного сетевого интерфейса.

- Протокол BOOTP [BOOT:2]

  Этот протокол позволяет хосту определить свой IP-адрес и адрес сервера загрузки непосредственно в процессе загрузки. Кроме того, дополнительно может передаваться маска подсети и список используемых по умолчанию шлюзов. Для нахождения сервера BOOTP хост передает широковещательные запросы с использованием протокола UDP. Для передачи широковещательных запросов BOOTP через маршрутизаторы используется специальное расширение, а в будущем IP Multicasting обеспечит стандартный механизм.

Для динамической настройки хостов предлагается использовать протокол BOOTP с расширением BOOTP Vendor Information Extensions, определенным в RFC 1084 [BOOT:3]. RFC 1084 определяет некоторые важные особенности такого расширения, не зависящие от реализации. В частности, это расширение позволяет протоколу BOOTP обеспечивать информацию о маске сети; **рекомендуется** передавать маски сетей в соответствии с этим документом.

Обсуждение

Исторически понятие подсетей появилось после определения IP и был разработан специальный механизм (сообщения ICMP Address Mask) для передачи хостам значения маски. Однако адресная маска IP и соответствующий IP-адрес образуют концептуальную пару и для упрощения работы их следует определять

одновременно с помощью одного механизма из конфигурационного файла или во время динамической настройки типа BOOTP.

Отметим, что протокол BOOTP не обеспечивает возможности настройки конфигурации для всех интерфейсов многодомного хоста. Такие хосты или используют BOOTP раздельно для каждого интерфейса или настраивают один интерфейс с помощью BOOTP для выполнения загрузки через сеть, а потом инициализируют остальные интерфейсы.

Предполагается, что конфигурационные параметры прикладных уровней извлекаются из файлов после загрузки операционной системы.

6.2.2.2 Фаза загрузки

На этапе загрузки предлагается использовать протокол TFTP [BOOT:1] на основе адреса IP, полученного по BOOTP.

Не следует использовать TFTP с широковещательным адресом, по причинам, рассмотренным выше (см. 4.2.3.4).

6.3 Удаленное управление

6.3.1 Введение

Сообщество Internet внесло значительный вклад в разработку протоколов сетевого управления [MGT:1, MGT:6] - в результате этого были разработаны протоколы SNMP[1] [MGT:4] и CMOT[2] [MGT:5].

Для управления с помощью протоколов SNMP или CMOT хост должен поддерживать соответствующий агент управления. В состав хостов Internet **следует** включать агент для протокола SNMP или CMOT.

Оба протокола SNMP и CMOT работают на основе MIB[3], определяющих набор объектов и значений для управления устройствами. Читая и меняя значения параметров, удаленное приложение может запрашивать и изменять состояние управляемой системы.

Стандарт MIB [MGT:3] был разработан для использования в обоих протоколах управления на основе типов данных, определенных SMI[4] [MGT:2]. Дополнительные переменные MIB могут включаться в ветви enterprises и experimental пространства имен MIB [MGT:2].

Для каждого протокольного модуля на хосте **следует** реализовать имеющие к нему отношение переменные MIB. Хосту **следует** использовать переменные MIB, определенные в последней версии стандарта MIB, и **можно** поддерживать другие переменные MIB, когда они приемлемы и полезны.

6.3.2 Общие вопросы

Базы MIB предназначены для хостов и шлюзов, хотя для этих случаев реализации MIB могут различаться в деталях. В этом параграфе рассмотрены вопросы интерпретации MIB для хостов. Очевидно, что последующие версии MIB будут включать дополнительные объекты для управления хостами.

Управляемый хост должен реализовать следующие группы определений объектов MIB: System, Interfaces, Address Translation, IP, ICMP, TCP, UDP.

Ниже перечислены конкретные интерпретации, применимые к хостам:

ipInHdrErrors

Отметим, что ошибка time-to-live exceeded[5] может происходить на хостах только при пересылке дейтаграмм isource-route.

ipOutNoRoutes

Эта переменная содержит счетчик дейтаграмм, отброшенных по причине отсутствия маршрута к хосту. Это может случиться, если все используемые по умолчанию шлюзы для данного хоста не работают.

ipFragOKs, ipFragFails, ipFragCreates

Хосты, не поддерживающие преднамеренной фрагментации (см. параграф «Фрагментация» в работе [INTRO:1]) **должны** возвращать нулевые значения для этих переменных.

icmpOutRedirects

Для хостов эта переменная **должна** всегда иметь нулевое значение, если хост не передает сообщений Redirect.

icmpOutAddrMaskReps

Для хостов эта переменная **должна** всегда иметь нулевое значение, если хост не уполномочен на передачу информации об адресных масках.

ipAddrTable

Для хостов объект IP Address Table представляет собой эффективную таблицу логических интерфейсов.

ipRoutingTable

Для хостов объект IP Routing Table представляет собой комбинацию маршрутного кэша и таблицы статических маршрутов, описанные в параграфе «Маршрутизация исходящих дейтаграмм» работы [INTRO:1].

В каждом объекте ipRouteEntry, записи ipRouteMetric1...4 обычно не имеют значения для хоста; **следует** задавать для них значения -1, если ipRouteType не имеет значения remote.

Если адресаты в подключенной сети не появляются в кэше маршрутов (см. параграф «Маршрутизация исходящих дейтаграмм» в работ [INTRO:1]), не будет записей со значением ipRouteType = direct.

Обсуждение

Текущая спецификация MIB не включает тип обслуживания TOS в записи ipRouteEntry, но в будущих реализациях этот параметр будет включен.

---

[1] Simple Network Management Protocol - простой протокол сетевого управления.
[2] Common Management Information Protocol over TCP - протокол передачи управляющей информации через TCP.
[3] Management Information Base - база информации для управления.
[4] Structure of Management Information - структура управляющей информации.
[5] Время жизни истекло.

Предполагается также расширение MIB для поддержки программ удаленного управления (например, удаленной настройки почтовых систем). Сетевые сервисные службы типа почтовых систем должны, следовательно, разрабатываться с учетом прерываний для удаленного управления.

6.3.3 Требования к функциям управления

Функция	*Параграф*	*Требования*
Поддержка агента SNMP или CMOT	6.3.1	Следует
Реализация указанных объектов в стандартной базе MIB	6.3.1	Следует

7. Литература

В этом разделе перечислены основные и дополнительные источники информации по рассмотренным в документе вопросам.

Общие вопросы

[INTRO:1] "Requirements for Internet Hosts -- Communication Layers," IETF Host Requirements Working Group, R. Braden, Ed., RFC 1122, October 1989.

[INTRO:2] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:3] "Official Internet Protocols," J. Reynolds and J. Postel, RFC 1011[1], May 1987.

[INTRO:4] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC 980, March 1986.

[INTRO:5] "Assigned Numbers," J. Reynolds and J. Postel, RFC 1010[2], May 1987.

TELNET

[TELNET:1] "Telnet Protocol Specification," J. Postel and J. Reynolds, RFC 854[3], May 1983.

[TELNET:2] "Telnet Option Specification," J. Postel and J. Reynolds, RFC 855, May 1983.

[TELNET:3] "Telnet Binary Transmission," J. Postel and J. Reynolds, RFC 856, May 1983.

[TELNET:4] "Telnet Echo Option," J. Postel and J. Reynolds, RFC 857, May 1983.

[TELNET:5] "Telnet Suppress Go Ahead Option," J. Postel and J. Reynolds, RFC 858, May 1983.

[TELNET:6] "Telnet Status Option," J. Postel and J. Reynolds, RFC 859, May 1983.

[TELNET:7] "Telnet Timing Mark Option," J. Postel and J. Reynolds, RFC 860, May 1983.

[TELNET:8] "Telnet Extended Options List," J. Postel and J. Reynolds, RFC 861, May 1983.

[TELNET:9] "Telnet End-Of-Record Option," J. Postel, RFC 885[4], December 1983.

[TELNET:10] "Telnet Terminal-Type Option," J. VanBokkelen, RFC 1091[5], February 1989.

[TELNET:11] "Telnet Window Size Option," D. Waitzman, RFC 1073, October 1988.

[TELNET:12] "Telnet Linemode Option," D. Borman, RFC 1116[6], August 1989.

[TELNET:13] "Telnet Terminal Speed Option," C. Hedrick, RFC 1079, December 1988.

[TELNET:14] "Telnet Remote Flow Control Option," C. Hedrick, RFC 1080[7], November 1988.

Дополнительная литература по TELNET

[TELNET:15] "Telnet Protocol," MIL-STD-1782[8], U.S. Department of Defense, May 1984.

[TELNET:16] "SUPDUP Protocol," M. Crispin, RFC 734, October 1977.

[TELNET:17] "Telnet SUPDUP Option," M. Crispin, RFC 736, October 1977.

[TELNET:18] "Data Entry Terminal Option," J. Day, RFC 732[9], June 1977.

[TELNET:19] "TELNET Data Entry Terminal option -- DODIIS Implementation," A. Yasuda and T. Thompson, RFC 1043, February 1988.

FTP

[FTP:1] "File Transfer Protocol," J. Postel and J. Reynolds, RFC 959[10], October 1985.

---

[1]Этот документ периодически обновляется. На сегодняшний день последняя версия содержится в RFC 5000. *Прим. перев.*

[2]В соответствии с RFC 3232 документ STD 2 утратил силу. Значения Assigned Numbers следует искать в базе данных, доступной на сайте www.iana.org/numbers.html. *Прим. перев.*

[3]В RFC 5198 содержится ряд обновлений для этого документа. *Прим. перев.*

[4]В исходном документе ошибочно указан документ RFC 855. *Прим. перев.*

[5]Этот документ заменяет RFC 930.

[6]В настоящее время этот документ заменен RFC 1184. *Прим. перев.*

[7]В настоящее время этот документ заменен RFC 1372. *Прим. перев.*

[8]Этот документ описывает тот же самый протокол, что и RFC 854. При возникновении разночтений RFC 854 имеет более высокий приоритет, а данный документ имеет преимущество над обоими.

[9]В RFC 1043 [TELNET:19] содержится ряд поправок к этой спецификации. *Прим. перев.*

[10]В RFC 2228 (безопасность), RFC 2640 (интернационализация), RFC 2773, RFC 3659, RFC 5797 содержится ряд дополнений и изменений к спецификации протокола. *Прим. перев.*

[FTP:2] "Document File Format Standards," J. Postel, RFC 678, December 1974.

[FTP:3] "File Transfer Protocol," MIL-STD-1780[1], U.S. Department of Defense, May 1984.

TFTP:

[TFTP:1] "The TFTP Protocol Revision 2," K. Sollins, RFC 783[2], June 1981.

Электронная почта:

[SMTP:1] "Simple Mail Transfer Protocol," J. Postel, RFC 821[3], August 1982.

[SMTP:2] "Standard For The Format of ARPA Internet Text Messages," D. Crocker, RFC 822[4], August 1982.

[SMTP:3] "Mail Routing and the Domain System," C. Partridge, RFC 974[3], January 1986.

[SMTP:4] "Duplicate Messages and SMTP," C. Partridge, RFC 1047, February 1988.

[SMTP:5a][5] "Mapping between X.400 and RFC 822," S. Kille, RFC 987[6], June 1986.

[SMTP:5b] "Addendum to RFC 987," S. Kille, RFC 1026[7], September 1987.

[SMTP:6] "Simple Mail Transfer Protocol," MIL-STD-1781[8], U.S. Department of Defense, May 1984.

[SMTP:7] "A Content-Type Field for Internet Messages," M. Sirbu, RFC 1049, March 1988.

Служба доменных имен:

[DNS:1] "Domain Names - Concepts and Facilities," P. Mockapetris, RFC 1034[9], November 1987.

[DNS:2] "Domain Names - Implementation and Specification," RFC 1035[10], P. Mockapetris, November 1987.

[DNS:3] "Mail Routing and the Domain System," C. Partridge, RFC 974[3], January 1986.

[DNS:4] "DoD Internet Host Table Specification," K. Harrenstein, RFC 952, M. Stahl, E. Feinler, October 1985.

Дополнительные ссылки по DNS:

[DNS:5] "Hostname Server," K. Harrenstein, M. Stahl, E. Feinler, RFC 953, October 1985.

[DNS:6] "Domain Administrators Guide," M. Stahl, RFC 1032, November 1987.

[DNS:7] "Domain Administrators Operations Guide," M. Lottor, RFC 1033, November 1987.

[DNS:8] "The Domain Name System Handbook," Vol. 4 of Internet Protocol Handbook, NIC 50007, SRI Network Information Center, August 1989.

Инициализация системы:

[BOOT:1] "Bootstrap Loading Using TFTP," R. Finlayson, RFC 906, June 1984.

[BOOT:2] "Bootstrap Protocol (BOOTP)," W. Croft and J. Gilmore, RFC 951[11], September 1985.

[BOOT:3] "BOOTP Vendor Information Extensions," J. Reynolds, RFC 1084[12], December 1988.

[BOOT:4] "A Reverse Address Resolution Protocol," R. Finlayson, T. Mann, J. Mogul, and M. Theimer, RFC 903, June 1984.

Управление:

[MGT:1] "IAB Recommendations for the Development of Internet Network Management Standards," V. Cerf, RFC 1052, April 1988.

[MGT:2] "Structure and Identification of Management Information for TCP/IP-based internets," M. Rose and K. McCloghrie, RFC 1065[13], August 1988.

---

[1]Этот документ основан на ранней версии спецификации FTP (RFC 765) и утратил свою силу.
[2]Действующий сегодня вариант спецификации TFTP описан в RFC 1350. *Прим. перев.*
[3]Этот документ признан устаревшим и заменен документом RFC 2821, который, в свою очередь, был заменен RFC 5321 и обновлен в RFC 5336. *Прим. перев.*
[4]Этот документ признан устаревшим и заменен RFC 2822, который, в свою очередь, был заменен RFC 5322 и обновлен в RFC 5335 и RFC 5336. *Прим. перев.*
[5]Документы [SMTP:5a] и [SMTP:5b] содержат описание стандарта, предложенного для организации шлюзов между Internet и средами X.400.
[6]В настоящее время этот документ устарел и заменен RFC 1327 и RFC 2156. *Прим. перев.*
[7]В исходном документе номер этого RFC не указан в результате ошибки. В настоящее время этот документ устарел и заменен RFC 1327 и RFC 2156. *Прим. перев.*
[8]Эта спецификация описывает тот же протокол, что RFC 821. Однако, стандарт MIL-STD-1781 не полон (в частности, в нем отсутствуют записи MX [SMTP:3]).
[9]RFC 1101, RFC 1183, RFC 1348, RFC 1876, RFC 1982, RFC 2065, RFC 2181, RFC 2308, RFC 2535, RFC 4035, RFC 4343, RFC 4592, RFC 5936 содержат ряд дополнений и поправок к этому стандарту. *Прим. перев.*
[10]В RFC 1101, RFC 1183, RFC 1348, RFC 1876, RFC 1982, RFC 1995, RFC 1996, RFC 2065, RFC 2136, RFC 2137, RFC 2181, RFC 2308, RFC 2535, RFC 2845, RFC3425, RFC 3658, RFC 4033, RFC 4034, RFC 4035, RFC 4343, RFC 5936 содержится ряд дополнений и поправок к этому стандарту. *Прим. перев.*
[11]В RFC 1395, RFC 1497, RFC 1532, RFC 1542, RFC 5494 содержится ряд дополнений и поправок к этому стандарту. *Прим. перев.*
[12]Этот документ отменен более поздними RFC 1395, RFC 1497, RFC 1533. *Прим. перев.*
[13]Этот документ отменен более поздним RFC 1155. *Прим. перев.*

[MGT:3] "Management Information Base for Network Management of TCP/IP-based internets," M. Rose and K. McCloghrie, RFC 1066[1], August 1988.

[MGT:4] "A Simple Network Management Protocol," J. Case, M. Fedor, M. Schoffstall, and C. Davin, RFC 1098[2], April 1989.

[MGT:5] "The Common Management Information Services and Protocol over TCP/IP," U. Warrier and L. Besaw, RFC 1095[3], April 1989.

[MGT:6] "Report of the Second Ad Hoc Network Management Review Group," V. Cerf, RFC 1109, August 1989.

Вопросы безопасности

Существует множество вопросов безопасности, связанных с приложениями и служебными программами, но подробное рассмотрение этих вопросов выходит за пределы данного RFC. Связанные с безопасностью темы рассмотрены в параграфах, посвященных протоколу TFTP (4.2.1, 4.2.3.4, 4.2.3.5), командам SMTP VRFY и EXPN (5.2.3), а также командам SMTP HELO (5.2.5), и SMTP DATA (Section 5.2.8).

Адрес автора

Robert Braden (Роберт Браден)

USC/Information Sciences Institute

4676 Admiralty Way

Marina del Rey, CA 90292-6695

телефон: (213) 822 1511

EMail: Braden@ISI.EDU

Перевод на русский язык

Николай Малых

nmalykh@protocols.ru

[1]Этот документ отменен более поздним RFC 1156. *Прим. перев.*
[2]Этот документ отменен более поздним RFC 1157. Перевод имеется на сайте http://www.protocols.ru. *Прим. перев.*
[3]Этот документ отменен более поздним RFC 1189. *Прим. перев.*