Network Working Group
Request for Comments: 1122

Internet Engineering Task Force
R. Braden, Editor
October 1989

Требования к хостам Internet - коммуникационные уровни

Requirements for Internet Hosts -- Communication Layers

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. Данный документ посвящен коммуникационным протоколам, а второй документ RFC 1123 - прикладным протоколам и протоколам поддержки.

Оглавление

1. Введение

Этот документ является первым из пары RFC, определяющих и обсуждающих требования к реализации хост-систем на базе стека протоколов Internet. Документ посвящен коммуникационным протоколам - канальный уровень, уровень IP (сетевой) и транспортный уровень. Второй документ пары - RFC 1123 Requirements for Internet Hosts -- Application and Support [INTRO:1] - посвящен протоколам прикладного уровня. С этой парой документов также тесно связан RFC 1009 - Requirements for Internet Gateways [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 не полон - 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 сложностям при взаимодействии, снижению производительности и иным проблемам.

В документе обсуждается и разъясняется множество требований и рекомендаций. Простой список требований может быть опасен по ряду причин:

- некоторые требуемые функции более важны, а ряд функций не является обязательным;
- 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 он связан.

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.

1.1 Архитектура Internet

Рассмотрению основ архитектуры Internet и поддерживаемых протоколов посвящена книга DDN Protocol Handbook [INTRO:3], основы архитектуры рассмотрены также в работах [INTRO:9], [INTRO:10], [INTRO:11]. В работе [INTRO:5] рассматриваются вопросы получения документов со стандартами протоколов Internet, а документ [INTRO:6] содержит список номеров (числовых идентификаторов), выделенных для протоколов Internet.

1.1.1 Хосты Internet

Хост-компьютер или попросту хост является конечным пользователем коммуникационного сервиса. На хостах в общем случае выполняются программы по запросам пользователей и/или поддерживаются коммуникационные службы Internet для обслуживания пользовательских запросов. Хосты Internet соответствуют концепции конечной системы (End-System) в стеке протоколов OSI [INTRO:13].

Коммуникационная система Internet содержит соединенные между собой сети передачи пакетов, в которых обмен информацией между хостами осуществляется на основе протоколов Internet. Сети подключаются к Internet или промежуточным системам OSI (Intermediate Systems, [INTRO:13]) с использованием компьютеров, обеспечивающих коммутацию пакетов - такие компьютеры называют шлюзами (gateway) или маршрутизаторами IP (IP router)[1]. В документе Requirements for Internet Gateways [INTRO:2] содержатся официальные спецификации для шлюзов (маршрутизаторов) Internet. RFC 1009 вместе с настоящим документом и RFC 1123 [INTRO:1] определяют правила для текущей реализации архитектуры Internet.

Хосты Internet сильно различаются по своим размерам, производительности и выполняемым функциям. В сети Internet используются самые разные компьютеры, включая настольные ПК, рабочие станции, мэйнфреймы и суперкомпьютеры. По выполняемым функциям хосты делятся на системы общего назначения (например, терминальные серверы) и мультисервисные системы (поддержка широкого спектра сетевых служб - remote login, FTP, электронная почта и т. п.).

[1] Во время подготовки этого RFC термин gateway (шлюз) использовался применительно к хостам, обеспечивающим пересылку пакетов между разными физическими сетями. Со временем функции межсетевой пересылки трафика стали реализоваться в основном на специализированных устройствах, называемых маршрутизаторами (router), а термин шлюз сейчас обычно относят в к системам, обеспечивающим преобразования форматов на более высоких, нежели IP, уровнях (например, шлюз электронной почты). В этом документе термины «шлюз» и «маршрутизатор» используются, как синонимы, если в тексте явно не означено иное. *Прим. перев.*

Для обозначения хостов с множеством интерфейсов в одну или несколько сетей используется термин multihomed (многодомный). Определение многодомного хоста приведено в параграфе 1.3.3 Терминология.

1.1.2 Архитектурные допущения

Современная архитектура Internet основана на группе базовых допущений о коммуникационной системе. Допущения, связанные с хостами, перечислены ниже:

(a) **Internet является «сетью сетей».**
Каждый хост напрямую подключен к какой-либо сети (сетям) - соединение с Internet является лишь концептуальным. Два хоста одной сети могут обмениваться между собой данными с использованием тех же протоколов, которые служат для связи с хостами удаленных сетей.

(b) **Шлюзы не хранят информацию о состоянии соединений.**
Для повышения устойчивости коммуникационных систем шлюзы разрабатываются таким образом, чтобы обеспечивалась пересылка дейтаграмм IP без организации явных соединений (stateless) и независимо от других дейтаграмм. В результате для обеспечения устойчивости могут использоваться избыточные (redundant) пути, которые активизируются при возникновении сбоев на основных путях.
Вся информация, требуемая для сквозного управления потоком данных (end-to-end flow control) через соединение, обеспечивается хостами с помощью программ транспортного или прикладного уровня. Таким образом, все данные для контроля соединения сосредоточены в конечных точках соединения и могут быть потеряны только при сбое на хостах.

(c) **Маршрутизация осуществляется в шлюзах.**
Маршрутизация является сложным процессом и должна выполняться шлюзами, а не хостами. Важной причиной такого допущения является избавление от необходимости смены или повторной настройки программ в результате изменений, вносимых при изменении архитектуры маршрутизации Internet.

(d) **Система должна быть совместима с различными вариантами сетей.**
Базовой предпосылкой архитектуры Internet является возможность изменения в широких пределах сетевых параметров (например, скорости каналов, задержки, числа теряемых пакетов, изменения порядка доставки пакетов, максимального размера пакетов и т. п.). Другим важным требованием является устойчивость к сбоям в отдельных сетях, шлюзах, хостах и сохранение возможности работы с доступной скоростью. Конечной целью взаимодействия открытых систем (open system interconnection) является обеспечение эффективной работы Internet и полной интероперабельности для всех типов хостов и сетей, соединенных через различные каналы Internet.
В некоторых случаях разработчики преследуют менее амбициозные цели. Например, среды ЛВС обычно менее требовательны к хостам по сравнению с Internet в целом - локальные сети имеют небольшие задержки, теряется только малая часть пакетов и порядок доставки пакетов обычно сохраняется. Некоторые компании в своих реализациях используют решения, приемлемые для ЛВС, но совершенно не подходящие для гетерогенных сред. Обычно такую продукцию преподносят в качестве дешевого решения для локальных сетей. Однако, изолированная ЛВС может перестать быть таковой и при организации взаимодействия с другими сетями и Internet использование таких упрощенных решений с неизбежностью породит проблемы. В конце концов ни пользователь, ни производитель могут оказаться не в состоянии решить проблемы, возникшие в результате применения не полностью соответствующих стандартам оборудования и программ.

Приведенные в документе требования относятся к полнофункциональным хостам Internet, обеспечивающим полную интероперабельность при использовании самых разных путей передачи данных в сети Internet.

1.1.3 Стек протоколов Internet

Для связи через Internet на хостах должен использоваться многоуровневый набор протоколов, соответствующий стеку протоколов Internet. Обычно на хостах реализован по крайней мере один протокол для каждого из уровней.

Уровни протоколов, используемые в архитектуре Internet, описаны в работе [INTRO:4]:

- **Прикладной уровень** (Application Layer)
  Прикладной уровень располагается в верхней части стека протоколов Internet. В стеке Internet прикладной уровень не разделен на подуровни, хотя некоторые из протоколов прикладного уровня Internet содержат внутренние подуровни. Прикладной уровень стека Internet объединяет в себе функции двух уровней (Presentation - уровень представления и Application - прикладной) эталонной модели OSI.
  Мы будем различать две категории протоколов прикладного уровня - пользовательские протоколы, которые предоставляют услуги непосредственно пользователю, и протоколы поддержки (служебные), обеспечивающие системные функции общего назначения. Требования к пользовательским и служебным протоколам рассмотрены в RFC 1123 [INTRO:1].
  Наиболее распространенными пользовательскими протоколами Internet являются:
  - Telnet (удаленный вход в систему);
  - FTP (передача файлов);
  - SMTP (доставка электронной почты).

  Существует также множество стандартизованных [INTRO:4] и фирменных пользовательских протоколов.
  Служебные протоколы используются для преобразования имен, загрузки ОС и управления - к числу таких протоколов относятся SNMP, BOOTP, RARP, DNS[1].

- **Транспортный уровень** (Transport Layer)
  Транспортный уровень обеспечивает сквозную связь (end-to-end) между приложениями через сеть. На транспортном уровне используются два основных протокола:

[1]Domain Name System - система доменных имен.

- Transmission Control Protocol (TCP) - протокол управления передачей;
- User Datagram Protocol (UDP) - протокол пользовательских дейтаграмм.

TCP представляет собой основанный на соединениях (connection-oriented) транспортный сервис с гарантированной доставкой пакетов, обеспечивающий надежную доставку с сохранением порядка следования пакетов и управлением потоком данных. Протокол UDP не использует явных соединений (connectionless) и передает данные в виде дейтаграмм (datagram) без гарантии доставки.
Исследовательскими организациями были разработаны и другие протоколы транспортного уровня, которые могут получить статус стандартных протоколов.
Более подробное описание протоколов транспортного уровня приведено в главе 4.

- **Уровень Internet**[1] (Internet Layer)
  Все транспортные протоколы используют протокол IP (Internet Protocol) для передачи данных от отправителя к получателю. IP представляет собой службу доставки дейтаграмм без организации соединений (connectionless), не обеспечивающую сквозной гарантии доставки. Таким образом, при доставке на хост получателя дейтаграммы IP могут оказаться поврежденными; кроме того, не гарантируется сохранение порядка их доставки, отдельные дейтаграммы могут быть потеряны, а некоторые - продублированы. Если требуются гарантии доставки, ответственность за такие гарантии должны брать на себя вышележащие уровни. Протокол IP отвечает за адресацию, обозначение типа сервиса, фрагментацию и сборку, а также безопасность.
  Передача данных без организации соединений лежит в основе протокола IP и является одной из основных характеристик архитектуры Internet. Протокол IP послужил моделью при разработке сетевого протокола OSI Connectionless Network Protocol [INTRO:12].
  Протокол управления ICMP является важной составной частью IP, хотя с точки зрения архитектуры он работает поверх IP (т. е., использует IP для передачи данных, подобно транспортным протоколам TCP и UDP). ICMP обеспечивает доставку сообщений об ошибках, насыщении сети и перенаправлении пакетов для первого маршрутизатора (first-hop).
  IGMP представляет собой протокол уровня Internet, используемый для организации динамических групп хостов с целью группового обмена информацией (IP multicasting).
  Протоколы уровня Internet (IP, ICMP и IGMP) более подробно рассмотрены в главе 3.

- **Канальный уровень** (Link Layer)
  Для связи с непосредственно подключенными к нему сетями хост должен поддерживать коммуникационный протокол, используемый для обмена данными с сетью. Мы будем называть такой протокол MAC-протоколом (media-access layer protocol[2]) или протоколом канального уровня (link layer).
  На канальном уровне может применяться множество протоколов в зависимости от используемой сетевой технологии. Протоколы канального уровня рассмотрены в главе 2.

1.1.4 Встроенная маршрутизация

Программы некоторых хостов Internet включают встроенный маршрутизатор - такие хосты могут пересылать пакеты, как это делают шлюзы, обеспечивая в то же время функции прикладного уровня, как обычные хосты.

Такие системы двойного назначения должны соответствовать требованиям RFC 1009 [INTRO:2] в части функций шлюза и требованиям настоящего документа, применительно к функциям хоста. Во всех перекрывающихся случаях эти две спецификации должны быть согласованы.

В сообществе Internet существует множество мнений о поддержке встроенных функций маршрутизации. Ниже приведены основные аргументы за и против таких систем[3]:

- **За** - такие системы могут быть приемлемы с точки зрения удобства и экономичности в локальных сетях или изолированных системах для использования существующих хостов в качестве маршрутизаторов.
  Существуют также аргументы в пользу встроенной маршрутизации с точки зрения архитектуры - компьютеров с несколькими сетевыми интерфейсами больше, нежели с одним интерфейсом, и поддержка функций маршрутизации позволяет более эффективно использовать такие хосты.

- **Против** - алгоритмы и протоколы маршрутизации постоянно изменяются с ростом Internet. Попытка реализации этих протоколов на хостах общего назначения будет требовать постоянного обновления программ на таких хостах. Наличие многочисленных реализаций кода маршрутизации дополнительно затрудняет внесение требуемых изменений. И, наконец, реализация уровня IP для шлюза сложнее, нежели для обычного хоста, что затрудняет работу с такими хостами.
  Кроме того, стиль работы некоторых хостов просто не приемлем для их использования в качестве стабильных и надежных маршрутизаторов.

Обе точки зрения имеют свои плюсы и минусы. Прежде, чем выбрать решение для себя, администратор должен принять во внимание реальную потребность в поддержке хостом функций маршрутизации. Более детальное рассмотрение этого вопроса приведено в параграфе 3.1.

1.2 Общие вопросы

Существуют два важнейших аспекта, которые должны принимать во внимание разработчики программ для хостов Internet.

1.2.1 Постоянное изменение 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

[1]Уровень Internet соответствует сетевому уровню (network layer) эталонной модели OSI. *Прим. перев.*
[2]Протокол уровня доступа к среде.
[3]В современной среде настольных ПК этот вопрос несколько утратил остроту, поскольку функции маршрутизации реализованы практически для всех операционных систем (встроенными средствами или с помощью дополнительных программ). *Прим. перев.*

документе. Изменения планируются и осуществляются под контролем и с участием производителей сетевой продукции и организаций, ответственных за работу сетей.

Обновление и постоянное совершенствование являются неотъемлемыми чертами современных сетевых протоколов и такая ситуация будет сохраняться еще достаточно долго. 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 между собой через эту сеть. Опыт и знания разработчиков, реализованные в их программах, за короткое время становятся достоянием технического сообщества Internet.

1.2.2 Принципы устойчивости

Для протоколов всех уровней существует общее правило, которому приложения должны следовать во избежание проблем с устойчивостью и интероперабельностью [IP:1]:

Предъявлять жесткие требования по отношению к себе, быть более мягким по отношению к другим[1].

Программы должны уметь обрабатывать все мыслимые ошибки; не имеет значения вероятность возникновения той или иной ошибки — рано или поздно 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 допущение 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[2]!

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 не полна. 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 и другой аспект - программы на других хостах могут содержать определения, которые могут подтолкнуть хост к неразумным, но допустимым с точки зрения протокола действиям. 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. [INTRO:1]);

(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, я 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. Это позволит реализовать весь стек в ПЗУ 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

[1] В оригинале «Be conservative in what you do, be liberal in what you accept from others». *Прим. перев.*
[2] Сегодня ситуация уже иная и целенаправленные действия злоумышленников наносят значительно больший вред. *Прим. перев.*

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 кодов (к сожалению, такое встречается в Internet достаточно часто). Для обеспечения корректного сосуществования с такими «сбойными» системами администраторам зачастую приходится «расстраивать» (mis-configure) 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 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 заключении 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

Деление протоколов на уровни, которое в общем случае используется как организационный принцип при реализации сетевых программ, служит также принципом организации данного документа. При описании правил мы предполагаем, что реализация достаточно точно отражает деление протоколов по уровням. Таким образом, документ поделен на три основных части - канальный уровень, уровень internet и транспортный уровень. Документ RFC 1123 [INTRO:1] содержит описание программ прикладного уровня. Такая организация документов представляется простой и наглядной.

Однако, жесткое деление на уровни не является совершенной моделью как для стека протоколов, так и для реализации программ. Протоколы различных уровней взаимодействуют в комплексе, иногда даже трудно отделить один протокол от другого, а отдельные функции могут включать несколько уровней. При разработке программных реализаций существует масса вариантов, многие из которых «творчески» подходят к вопросам деления на уровни. Мы рекомендуем всем разработчикам внимательно прочесть документы [INTRO:7] и [INTRO:8].

В этом документе описывается концептуальный интерфейс взаимодействия между уровнями, использующий функциональную нотацию (procedure call - вызов процедур), подобную той, что применяется в спецификации TCP [TCP:1]. Реализация хоста должна поддерживать логические потоки информации, применяемые при таких вызовах, но не требовать дословной реализации самих вызовов. Например, во многих реализациях связь между транспортным уровнем и IP отражается предоставлением совместного доступа к общим структурам данных.

В общем случае 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 Уровень требований

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

Требуется, должен (MUST)

Этот глагол (или причастие от него) используется 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** 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 данных[1], 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

При описании протоколов нижних уровней термин сообщение обозначает модуль, передаваемый на транспортном уровне. В частности, сегмент TCP является сообщением. Сообщение содержит заголовок транспортного уровня, за которым следуют данные. Для сквозной передачи через сеть Internet сообщение должно инкапсулироваться в дейтаграммы.

Дейтаграмма IP - IP Datagram

Дейтаграмма IP представляет собой модуль данных протокола IP. Дейтаграмма IP содержит заголовок IP, за которым следует информация транспортного уровня (т. е., сообщение).

При описании уровня Internet (глава 3) термин дейтаграмма обычно относится к дейтаграммам IP.

Пакет - Packet

Пакет представляет собой модуль данных, передаваемый через интерфейс между уровнем Internet и канальным уровнем. Пакет включает заголовок IP и данные. Пакет может быть полной дейтаграммой IP или фрагментом такой дейтаграммы.

Кадр - Frame

Кадр представляет собой модуль данных для передачи на канальном уровне и содержит заголовок канального уровня, за которым следует пакет.

Подключенная сеть - Connected Network

Сеть, к которой непосредственно подключен хост, часто называют локальной сетью (local network) или подсетью (subnetwork) применительно к данному хосту. Однако использование этих терминов может привести к путанице, поэтому мы будем пользоваться термином подключенная сеть (connected network).

Multihomed - многодомный

Хост называют многодомным если он имеет более одного адреса IP. Более подробное обсуждение таких хостов вы найдете в параграфе 3.3.4.

Физический интерфейс - Physical network interface

Физический интерфейс используется для подключения хоста к сети. Интерфейс имеет адрес канального уровня (возможно, уникальный). На хосте может использоваться множество физических интерфейсов с общим адресом канального уровня, но для каждого хоста одной физической сети должен обеспечиваться уникальный адрес на канальном уровне.

Логический интерфейс - Logical [network] interface

Определим логический [сетевой] интерфейс как логический путь в подключенную сеть. Для идентификации логического интерфейса служит уникальный адрес IP. Более подробное рассмотрение логических интерфейсов приведено в параграфе 3.3.4.

Specific-destination address

Эффективный адрес получателя дейтаграммы, даже если она является широковещательной (broadcast) или групповой (multicast). Дополнительная информация по этому вопросу содержится в параграфе 3.2.1.3.

Путь - Path

В каждый момент времени все дейтаграммы IP от определенного хоста-отправителя к определенному хосту-получателю обычно передаются через одинаковую последовательность маршрутизаторов. Мы будем использовать термин «путь» для обозначения таких последовательностей маршрутизаторов. Отметим, что путь является однонаправленным, т. е. данные между парой хостов могут передаваться в каждом направлении по своему пути.

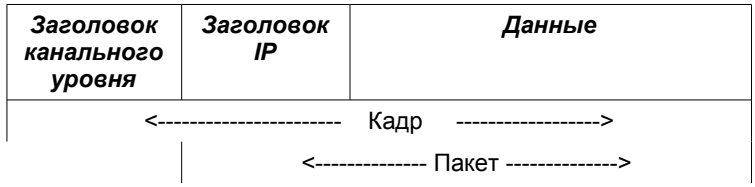
MTU[2]

Максимальный размер передаваемого модуля данных (т. е., размер наибольшего пакета, который может быть передан через сеть).

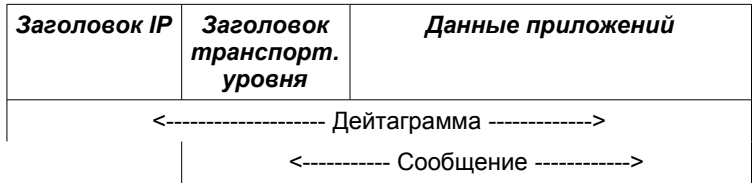
Термины кадр, пакет, дейтаграмма и сегмент дополнительно проиллюстрированы на приведенных ниже рисунках.

[1]Термином модуль будем обозначать неделимую при передаче на данном уровне совокупность данных. *Прим. перев.*
[2]Maximum transmission unit

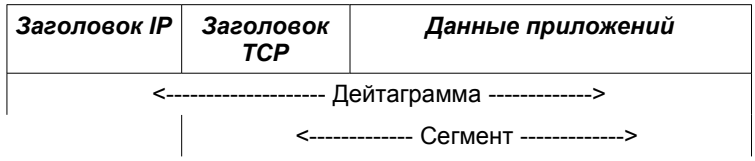
A) Передача в подключенную сеть



B) Перед фрагментацией IP или после сборки



то же самое для TCP



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.

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) и 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) и Rayan Zachariassen (Toronto).

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

2. Канальный уровень

2.1 Введение

Для всех систем Internet - хостов и маршрутизаторов - предъявляются одинаковые требования к протоколам канального уровня. Эти требования подробно рассмотрены в главе 3 документа «Requirements for Internet Gateways» [INTRO:2] и дополнены в настоящем документе.

2.2 Общие вопросы

Не рассматриваются.

2.3 Частные вопросы

2.3.1 Согласование трейлерного протокола

Для инкапсуляции на канальном уровне **может** применяться трейлерный протокол [LINK:1], но использование такого протокола должно быть согласовано обеими системами (хосты или маршрутизаторы), взаимодействующими на канальном уровне с применением трейлеров. Если системы не могут согласовать трейлерный протокол динамически (для каждого получателя), принятая по умолчанию конфигурация **должна** запрещать использование трейлеров.

Обсуждение

Трейлерный протокол представляет собой метод инкапсуляции на канальном уровне, обеспечивающий перекомпоновку пакетов, передаваемых в физическую сеть. В некоторых случаях трейлеры повышают производительность протоколов верхних уровней за счет снижения объема копируемых данных. Протоколы верхних уровней не знают об использовании трейлеров, но приемник и передатчик пакетов **должны** понимать, какой протокол используется для инкапсуляции.

Некорректное использование трейлеров может привести к серьезной путанице. С помощью трейлеров инкапсулируются только пакеты с заданным набором атрибутов и обычно этому требованию удовлетворяет лишь малая часть передаваемых пакетов. Таким образом, если одна система будет использовать трейлеры, а другая не будет, часть пакетов будет «улетать в черную дыру», а остальные будут доставляться нормально.

Реализация

В сетях Ethernet пакеты, инкапсулируемые с трейлерами, используют различные типы Ethernet [LINK:1] и согласование трейлера выполняется одновременно с использованием протокола ARP для определения адреса получателя. Обмен пакетами ARP для определения адреса осуществляется обычным способом, но хост, согласующий трейлер, будет передавать дополнительный отклик ARP (trailer ARP reply), указывающий тип трейлерного протокола инкапсуляции в формате обычного отклика ARP. Если хост, настроенный на использование трейлеров, получает отклик ARP с типом трейлера от удаленной машины, он может добавить ее в список систем, поддерживающих трейлеры (например, пометив соответствующую запись в кэше ARP).

Хосты, желающие использовать трейлерную инкапсуляцию, передают отклик trailer ARP всякий раз при завершении нормального обмена ARP для IP-адреса. Таким образом, хост, получивший запрос ARP для своего IP-адреса, будет передавать отклик trailer ARP в дополнение к нормальному отклику IP ARP; хост, передавший запрос IP ARP, будет передавать отклик trailer ARP при получении отклика на запрос IP ARP. При таком способе как запрашивающий, так и отвечающий хост в процессе обмена пакетами IP ARP может запросить использование трейлерной инкапсуляции.

Такая схема с использованием дополнительных пакетов trailer ARP reply вместо запросов ARP на указание типа трейлерного протокола была разработана для того, чтобы избежать непрерывного обмена пакетами ARP с некорректно работающими хостами, которые в ответ на запрос типа трейлера передают стандартный отклик ARP для IP-адреса. Проблема решается за счет передачи отклика trailer ARP в ответ на отклик IP ARP только в тех случаях, когда отклик IP ARP отвечает на невыполненный запрос (это верно для тех случаев, когда аппаратный адрес хоста остается неизвестным к моменту получения отклика IP ARP). Отклик trailer ARP всегда можно посылать вместе с откликом IP ARP, соответствующим запросу IP ARP.

2.3.2 Протокол преобразования адресов - ARP

2.3.2.1 Проверка кэша ARP

Реализация протокола преобразования адресов ARP[1] [LINK:2] **должна** обеспечивать механизм удаления устаревших записей из таблицы. Если поддерживаемый механизм использует тайм-аут, **должна** обеспечиваться возможность настройки времени ожидания.

Реализация **должна** обеспечивать механизм предотвращения лавинной рассылки (ARP flooding) в виде повторяющихся с высокой частотой запросов ARP Request для одного адреса IP. Рекомендуемая частота запросов не должна превышать для каждого адреса 1 запрос/сек.

Обсуждение

Спецификация ARP [LINK:2] предлагает, но не требует использовать механизм тайм-аутов для объявления неприемлемыми (invalidate) элементов кэша при смене хостом своего адреса Ethernet. Широкое распространение proxy ARP (см. параграф 2.4 в работе [INTRO:2]) существенно повышает вероятность того, что в кэше будут содержаться неприемлемые записи, следовательно требуется механизм удаления устаревших записей из кэша ARP. Даже в отсутствие proxy ARP использование тайм-аутов полезно для автоматической коррекции данных ARP, которые могли быть кэшированы.

Реализация

Для удаления устаревших записей из кэша используется 4 механизма (иногда в комбинации).

(1) **Timeout** - записи периодически удаляются из кэша, даже если они остаются актуальными[2]. При использовании ARP тайм-аут должен быть порядка 1 минуты.

(2) **Unicast Poll** - активный опрос удаленных хостов с помощью периодической отправки по их адресам пакетов ARP Request и удаление записей для тех адресов, от которых не пришло пакетов ARP Reply после N последовательных опросов. Тайм-аут должен по-прежнему составлять около 1 минуты, а типичное значение N = 2.

(3) **Link-Layer Advice** - если драйвер канального уровня обнаруживает проблемы с доставкой, соответствующая запись удаляется из кэша ARP.

(4) **Higher-layer Advice** - обеспечивается вызов на канальный уровень с уровня Internet для индикации проблем с доставкой. Эффект такого вызова заключается в том, что соответствующая запись удаляется из кэша. Такой вызов является аналогом вызова ADVISE_DELIVPROB() с транспортного уровня на уровень Internet (см. параграф 3.4) и подпрограмма ADVISE_DELIVPROB фактически может использоваться для вызова программы канального уровня, удаляющей запись из кэша ARP.

Варианты (1) и (2) используют тайм-аут порядка 1 минуты или меньше. При отсутствии ARP такой короткий период ожидания может породить заметный избыточный трафик в больших сетях Ethernet. Следовательно, может потребоваться увеличение тайм-аута ARP.

2.3.2.2 Очередь пакетов ARP

Канальный уровень **должен** сохранять (а не отбрасывать) по крайней мере один (последний) пакет из каждой группы пакетов для того или иного адреса IP и передавать сохраненный пакет, когда адрес будет преобразован (resolve).

Обсуждение

Отказ от выполнения приведенного выше требования приводит к потере первого пакета при каждом обмене. Хотя протоколы верхних уровней в общем случае способны заново отправлять пакеты при их потере, утрата пакетов будет снижать производительность системы. Например, потеря запроса на открытие сеанса TCP приведет к тому, что оценка времени кругового обхода (round-trip time) будет некорректной. Приложения на основе UDP (такие, как DNS) будут еще сильнее страдать от такого недостатка.

[1]Address Resolution Protocol.

[2]Отсчет времени должен начинаться заново всякий раз, когда запись в кэше обновляется. Точка отсчета определяется путем просмотра полей отправителя независимо от искомого адреса в широковещательных пакетах ARP.

2.3.3 Инкапсуляция Ethernet и IEEE 802

Инкапсуляция пакетов IP для сетей Ethernet описана в RFC 894 [LINK:3], а RFC 1042 [LINK:4] содержит описание инкапсуляции IP для сетей IEEE 802. RFC 1042 уточняет вопросы, рассмотренные в параграфе 3.4 документа [INTRO:2].

Для каждого хоста Internet, подключенного к сети Ethernet (10 Мбит/с) с помощью кабеля

- **требуется** поддержка приема и передачи пакетов с использованием инкапсуляции RFC 894;
- **рекомендуется** поддерживать прием пакетов RFC 1042 вперемешку с пакетами RFC 894;
- **допустимо** поддерживать передачу пакетов с использованием инкапсуляции RFC 1042.

Хосты Internet, реализующие передачу с использованием инкапсуляции обоих типов (RFC 894 и RFC 1042), **должны** поддерживать возможность настройки (конфигурационная опция) используемого типа инкапсуляции. По умолчанию **требуется** использовать инкапсуляцию RFC 894.

Отметим, что стандартная инкапсуляция IP в соответствии с RFC 1042 не использует значение идентификатора протокола (K1=6), зарезервированное IEEE для протокола IP, устанавливая вместо этого значение K1=170, указывающее на расширение (SNAP), которое может использоваться для поля EtherType. Для систем Internet **недопустима** передача пакетов IEEE 802 с использованием K1=6.

Трансляция адресов Internet в адреса канального уровня для сетей Ethernet и IEEE 802 **должна** обеспечиваться на основе протокола ARP.

Значение MTU для Ethernet составляет 1500, а для 802.3 - 1492[1] байта.

Обсуждение

Спецификация IEEE 802.3 обеспечивает работу по кабелям Ethernet 10 Мбит/с, что может вести к смешению в одной физической среде кадров Ethernet[2] и IEEE 802.3. Приемник может различать кадры Ethernet и 802.3 по значению поля длины (Length) для 802.3 - это 2-октетное поле совпадает[3] с полем EtherType в кадрах Ethernet. Значение поля длины для кадров 802.3 не должно превышать 1500, а все корректные значения EtherType превышают 1500.

Другая проблема совместимости связана с широковещательными кадрами на канальном уровне. Широковещательные кадры того или иного типа не будут видны хостам, способным принимать лишь другой тип кадров.

Основной задачей данного параграфа является обеспечение рекомендаций по взаимодействию сетей с инкапсуляцией RFC 894 и RFC 1042 в одной физической среде. Основной упор делается на доминирующие сейчас сети RFC 894, с учетом перехода в будущем на инкапсуляцию RFC 1042.

Отметим, что системы, поддерживающие только RFC 894, не могут напрямую взаимодействовать с системами RFC 1042. Если каждый тип инкапсуляции организовать в виде логической сети (на том же кабеле), то две логические сети можно будет связать через шлюз IP (маршрутизатор). Использование хостов, способных поддерживать оба формата, не всегда полезно (и не всегда возможно), поскольку трудно автоматически определить формат передачи кадров из-за проблем с широковещательными кадрами канального уровня.

2.4 Интерфейс между канальным уровнем и IP

Интерфейс приема пакетов между IP и канальным уровнем **должен** включать флаг, показывающий пакеты для широковещательной рассылки на канальном уровне.

Обсуждение

Хотя уровень IP в общем случае не знает адресов канального уровня (поскольку в каждой физической среде используется свой формат адресации), широковещательный адрес для поддерживающих широковещательную рассылку сред является в некотором роде исключением из общего правила. Более подробное рассмотрение этого вопроса приведено в параграфе 3.2.2.

Интерфейс передачи пакетов между IP и канальным уровнем **должен** включать 5-битовое поле TOS, описанное в параграфе 3.2.1.6.

Для канального уровня **недопустима** передача сообщений об ошибке Destination Unreachable на уровень IP по причине отсутствия записи с адресом получателя в кэше ARP.

2.5 Требования к канальному уровню

Функция	***Параграф***	*Требование*
Трейлерная инкапсуляция	2.3.1	Обязательно
Передавать трейлеры по умолчанию без согласования	2.3.1	Недопустимо
ARP	2.3.2	
Удаление устаревших записей	2.3.2.1	Обязательно
Предотвращение ARP-лавин	2.3.2.1	Обязательно
Настройка тайм-аута для кэша	2.3.2.1	Следует
Сохранение по крайней мере последнего пакета с непреобразованным адресом	2.3.2.2	Следует

[1] В соответствии с современной версией стандарта IEEE 802.3-2005 максимальный размер кадра Ethernet составляет 1500 октетов. *Прим. перев.*

[2] Речь идет о стандарте Ethernet DIX, который в настоящее время практически не используется. *Прим. перев.*

[3] По расположению в кадре. *Прим. перев.*

Функция	*Параграф*	*Требование*
Инкапсуляция Ethernet и IEEE 802	2.3.3	
Способность хоста:	2.3.3	
Использовать RFC 894 для приема и передачи	2.3.3	Обязательно
Использовать RFC 1042 для приема	2.3.3	Следует
Использовать RFC 1042 для передачи	2.3.3	Возможно
Для этого случая поддержка по умолчанию RFC 894	2.3.3	Обязательно
Использование инкапсуляции с K1=6	2.3.3	Недопустимо
Поддержка ARP для сетей Ethernet и IEEE 802	2.3.3	Обязательно
Канальный уровень сообщает о широковещательных кадрах уровню IP	2.4	Обязательно
Передача уровнем IP значений TOS на канальный уровень	2.4	Обязательно
Трактовка отсутствия адреса в кэше, как Destination Unreachable	2.4	Недопустимо

3. Протоколы уровня INTERNET

3.1 Введение

Принцип устойчивости - «быть либеральным на приеме и консервативным при передаче» особенно важен для уровня Internet, где некорректное поведение одного хоста может нарушить работу множества других хостов.

К числу протокольных стандартов уровня Internet относятся:

- **RFC 791** [IP:1] - определяет протокол IP и содержит введение в архитектуру Internet.
- **RFC 792** [IP:2] - определяет протокол ICMP, обеспечивающий диагностику и сообщения об ошибках для протокола IP. Хотя сообщения ICMP инкапсулируются в дейтаграммы IP, обработка ICMP рассматривается (и обычно реализована) как часть уровня IP. См. параграф 3.2.2.
- **RFC 950** [IP:3] - определяет обязательную поддержку подсетей для архитектуры адресации.
- **RFC 1112** [IP:4] определяет протокол IGMP (Internet Group Management Protocol - протокол управления группами Internet) как часть рекомендуемого расширения для хостов и интерфейсов хост - шлюз, обеспечивающего в масштабах Internet поддержку групповой адресации на уровне IP. См. параграф 3.2.3. Адресатами групповых (multicast) пакетов IP могут быть произвольные группы хостов Internet. Групповая адресация IP разработана, как естественное расширение групповой адресации на канальном уровне и обеспечивает стандартное толкование для локального доступа к multicasting-объектам.

Ссылки на другие источники информации приведены в главе 5.

Программы уровня Internet на каждом хосте **должны** поддерживать оба протокола - IP и ICMP. Требования в части поддержки IGMP рассмотрены в параграфе 3.3.7.

Уровень IP выполняет две основных функции: (1) выбирает следующий маршрутизатор или хост (next hop) для исходящих дейтаграмм IP и (2) обеспечивает сборку принимаемых дейтаграмм IP. Уровень IP также **может** (3) реализовать преднамеренную фрагментацию исходящих дейтаграмм. Наконец, уровень IP **должен** (4) поддерживать детектирование и обработку ошибок. Предполагается, что в будущем функциональность уровня IP может быть расширена путем разработки новых приложений Internet для поддержки и управления.

Для нормальных дейтаграмм осуществляется прямая (straightforward) обработка. Для входящих дейтаграмм уровень IP выполняет следующие операции:

(1) проверка корректности формата дейтаграммы;

(2) проверка того, что дейтаграмма адресована локальному хосту;

(3) обработка опций;

(4) при необходимости сборка дейтаграмм из фрагментов;

(5) передача инкапсулированного в дейтаграмме сообщения соответствующему модулю транспортного уровня.

Для исходящих дейтаграмм уровень IP выполняет следующие операции:

(1) установка всех полей, не заданных на транспортном уровне;

(2) выбор подходящего интерфейса подключенной сети (маршрутизация);

(3) фрагментация дейтаграммы, если это необходимо, или преднамеренная фрагментация при использовании таковой (см. параграф 3.3.3);

(4) передача пакетов соответствующему драйверу канального уровня.

Хост называют многодомным (multihomed), если он имеет несколько адресов IP. Поддержка множества адресов для хостов вносит в стек протоколов дополнительную сложность и возможность путаницы, В этой части архитектуры Internet требуется серьезная работа для решения всех проблем. С поддержкой множества адресов связаны, прежде всего, две аспекта проблемы:

(1) Local multihoming - рассматриваемый хост имеет множество адресов;

(2) Remote multihoming - локальному хосту приходится работать с удаленным хостом, использующим множество адресов.

В настоящее время обслуживание работы с удаленными многодомными хостами **должно** обеспечиваться на прикладном уровне, как описано в RFC 1123 [INTRO:1]. Работа с локальным хостом, поддерживающим множество адресов, рассмотрена ниже (параграф 3.3.4).

Любой хост, пересылающий дейтаграммы от других хостов, действует как маршрутизатор и **должен** удовлетворять спецификациям маршрутизаторов (шлюзов), рассматриваемым в RFC 1009 [INTRO:2]. Хост Internet, поддерживающий встроенный маршрутизатор, **должен** иметь конфигурационные опции, позволяющие отключить маршрутизацию (по умолчанию маршрутизация **должна** быть выключена). В таком режиме дейтаграмма, принятая через какой-либо интерфейс, не будет пересылаться другому хосту или шлюзу (если не используется маршрутизация, заданная отправителем - source-route), независимо от числа имеющихся в данном хосте интерфейсов. **Не допускается** автоматический переход хоста в режим маршрутизации при наличии на хосте нескольких интерфейсов, поскольку оператор хоста может не желать выполнять функции маршрутизации и быть некомпетентным в этом вопросе.

В таких случаях для принятой дейтаграммы зачастую используют операцию silently discard[1]. Это означает, что дейтаграмма отбрасывается без дальнейшей обработки и даже не передается сообщение ICMP об ошибке (см. параграф 3.2.2). Однако, для обеспечения диагностики хост **должен** обеспечивать возможность протоколирования таких ошибок (см. параграф 1.2.3), включая запись содержимого отбрасываемых дейтаграмм. Кроме того, количество отбрасываемых дейтаграмм **должно** учитываться счетчиком статистики.

Обсуждение:

Отбрасывание ошибочных дейтаграмм без уведомления в общем случае используется для предотвращения широковещательных штормов (broadcast storms).

3.2 Общие вопросы

3.2.1 Протокол Internet - IP

3.2.1.1 Номер версии - RFC 791, параграф 3.1

Дейтаграммы с номером версии, отличающимся от 4, **должны** отбрасываться без уведомления.

3.2.1.2 Контрольная сумма - RFC 791, параграф 3.1

Хост **должен** проверять контрольную сумму заголовка IP для каждой полученной дейтаграммы и отбрасывать без уведомления дейтаграммы с некорректной контрольной суммой.

3.2.1.3 Адресация - RFC 791, параграф 3.2

Существует пять классов IP-адресов - от A до E. Адреса класса D используются для групповой адресации IP [IP:4], а класс E зарезервирован для экспериментов[2].

Групповые адреса (класс D) представляют собой 28-битовые логические адреса, используемые для групп хостов, и могут быть постоянными или временными. Постоянные групповые адреса распределяет агентство IANA[3] [INTRO:6], а временные динамически выделяются для временных групп хостов. Принадлежность к группе определяется динамически на основе протокола IGMP [IP:4].

Рассмотрим более подробно IP-адреса классов A, B и C, используя обозначения:

```
{ <Номер сети>, <Номер хоста> }
```

или

```
{ <Номер сети>, <Номер подсети>, <Номер хоста> }
```

и -1 для обозначения полей, содержащих только единицы (1). Такая нотация не предполагает, что единицы в маске адреса должны быть непрерывными.

(a) { 0, 0 }

Данный хост в данной сети. Этот адрес **недопустимо** указывать в качестве адреса отправителя за исключением случаев передачи адреса отправителя в процессе инициализации, посредством которого хост узнает свой IP-адрес.

В параграфе 3.3.6 рассмотрены варианты нестандартного использования {0,0}.

(b) { 0, <Номер хоста> }

Указывает хост данной сети. Такие адреса **недопустимо** указывать в качестве адреса отправителя за исключением случаев использования, как адреса отправителя в процедурах инициализации, с помощью которых хост получает полный IP-адрес.

(c) { -1, -1 }

Широковещательный пакет ограниченного действия (Limited broadcast). Такой адрес **недопустимо** указывать в качестве адреса отправителя.

Дейтаграмма с таким адресом в поле получателя будет приниматься каждым хостом данной физической сети, но не будет выходить за пределы сети через маршрутизаторы.

(d) { <Номер сети>, -1 }

Широковещательный адрес для данной сети. Такой адрес **недопустимо** указывать в качестве адреса отправителя.

(e) { <Номер сети>, <Номер подсети>, -1 }

Направленное широковещание для заданной подсети[4]. Такой адрес **недопустимо** использовать в качестве адреса получателя за исключением случаев, когда отправитель является одной из двух конечных точек соединения «точка-точка» с 31-битовой маской.

(f) { <Номер сети>, -1, -1 }

[1]Отбросить без уведомления.
[2]В настоящее время принята парадигма бесклассовой междоменной маршрутизации CIDR (RFC 1517, RFC 1518, RFC 1519) и деление адресов IP на классы утратило актуальность. *Прим. перев.*
[3]Internet Assigned Number Authority.
[4]Текст этого пункта обновлен в соответствии с RFC 3021. Перевод текста исходного документа имел вид: «Широковещательный пакет для указанного маршрутизатора (конкретной подсети). Такой адрес **недопустимо** указывать в качестве адреса отправителя.». *Прим. перев.*

Широковещательный пакет для всех подсетей данной сети. Такой адрес **недопустимо** указывать в качестве адреса отправителя.

(g) { 127, <любой> }

Внутренний loopback-адрес хоста. Пакеты с таким адресом отправителя **недопустимо** передавать за пределы хоста.

(h) `{ <Номер сети>, <Номер подсети>, 0 }`

Номер подсети[1]. Этот адрес **не следует** использовать в качестве адреса отправителя за исключением ситуаций, когда отправитель является одной из двух конечных точек соединения «точка-точка» с 31-битовой маской. Для других типов каналов пакеты с таким адресом получателя **следует** отбрасывать без уведомления. Если такие пакеты не отбрасываются, они **должны** трактоваться, как широковещательные пакеты IP [RFC1812].

Номера сетей распределяются административно, чтобы каждая сеть в масштабе планеты имела уникальный номер.

В адресах IP **недопустимо** использование значений 0 и -1 в любом из полей <Номер хоста>, <Номер сети> или <Номер подсети>, за исключением специальных случаев, перечисленных выше. Это требование подразумевает, что каждое из полей должно иметь размер не менее 2 битов.

Более подробное рассмотрение широковещательных адресов дается в параграфе 3.3.6.

Хост **должен** поддерживать подсети IP [IP:3]. В соответствии с этим требованием каждый хост должен иметь маску адреса в форме {-1, -1, 0} (см. параграфы 3.2.2.9 и 3.3.1.1).

Когда хост передает дейтаграмму, в качестве IP-адреса отправителя **должен** указываться один из IP-адресов этого хоста, но не широковещательный или групповой адрес.

Хост **должен** отбрасывать без уведомления входящие дейтаграммы, не адресованные ему. Входящая дейтаграмма считается адресованной хосту, если в поле адреса получателя указан:

(1) IP-адрес этого хоста (один из имеющихся);

(2) широковещательный адрес IP, корректный для данной сети;

(3) адрес multicast-группы[2], в которую входит данный хост.

В большинстве случаев дейтаграммы, адресованные всем (broadcast) или группе (multicast) хостов, обрабатываются так, будто они направлены по одному из IP-адресов данного хоста. Мы будем использовать термин «конкретный адрес получателя» (specific-destination address) в качестве эквивалента локального IP-адреса хоста. Конкретный адрес получателя должен указываться в заголовках пакетов IP, если такие пакеты не являются групповыми или широковещательными. Конкретный адрес получателя является IP-адресом физического интерфейса, через который дейтаграмма принимается хостом.

Хост **должен** без уведомления отбрасывать дейтаграммы, содержащие адрес отправителя, противоречащий приведенным в этом параграфе правилам. Проверка корректности может осуществляться на уровне IP или любым протоколом транспортного уровня.

Обсуждение

Дейтаграммы с некорректными адресами могут появляться в результате широковещательной рассылки на канальном уровне дейтаграмм с индивидуальным (unicast) адресом или вследствие ошибок в настройках хостов и маршрутизаторов.

Архитектура хостов Internet должна поддерживать обработку IP-адресов, как 32-битовых целых чисел без каких-либо особенностей и избегать применения алгоритмов, требующих знания формата IP-адресов. Если этому правилу не следовать, любые изменения формата или интерпретации адресов в будущем потребуют внесения изменений в программы хостов. Однако, проверка корректности широковещательных и групповых адресов требует отказа от таких правил[3].

Разработчики программ должны знать, что приложения, использующие широковещательную адресацию во все подсети (рассмотренный выше вариант f), могут оказаться неработоспособными в некоторых сетях. Широковещание для всех подсетей поддерживается не всеми маршрутизаторами и даже при наличии такой поддержки отдельные администраторы запрещают ее при настройке маршрутизаторов.

3.2.1.4 Фрагментация и сборка - RFC 791, параграф 3.2

Модель Internet требует, чтобы каждый хост поддерживал сборку фрагментов (reassembly). Требования к фрагментации и сборке рассмотрены в параграфах 3.3.2 и 3.3.3.

3.2.1.5 Идентификация - RFC 791, параграф 3.2

При передаче идентичной копии ранее отправленной дейтаграммы хост **может** сохранять значение идентификационного поля для такой копии.

Обсуждение

Некоторые специалисты по протоколам Internet утверждают, что при передаче копии ранее отправленной дейтаграммы эта копия должна содержать такое же значение поля идентификации, как оригинал. Это обеспечивает два преимущества: (1) если дейтаграмма фрагментирована и некоторые фрагменты теряются, принимающая сторона может восстановить полную дейтаграмму из фрагментов оригинала и копии; (2) перегруженный маршрутизатор может использовать поле IP Identification (и поле Fragment Offset - смещение фрагмента) для удаления дубликатов дейтаграмм из очереди.

[1]Пункт (h) добавлен в перевод документа в соответствии с RFC 3021. *Прим. перев.*
[2]На принимающем физическом интерфейсе. *Прим. перев.*
[3]Существует еще ряд исключений, рассматриваемых в этом документе.

Однако наблюдения за реальным трафиком и потерей дейтаграмм в Internet показывают, что использование первого из перечисленных преимуществ маловероятно в силу существования других механизмов (например, перепаковка TCP перед повторной передачей), предотвращающих повторную передачу идентичных дейтаграмм [IP:9]. Следовательно, сохранение идентификационного поля при повторной передаче дейтаграмм может оказаться бесполезным. Кроме того, работающие без организации соединений протоколы (типа UDP) будут требовать взаимодействия с прикладными программами для сохранения значения идентификационного поля при повторной передаче.

3.2.1.6 Тип обслуживания - RFC 791, параграф 3.2

Байт Type-of-Service (тип обслуживания) в заголовке IP поделен на 2 части - поле Precedence (3 старших бита) и поле TOS (5 младших битов). В этом документе термин TOS всегда относится к одноименному полю (младшие 5 битов).

Поле Precedence предназначено для специальных приложений Министерства обороны США. Вопросы использования отличных от 0 значений этого поля выходит за пределы компетенции настоящего документа и стандартов IP. Производители продукции должны консультироваться с агентством DCA[1] для получения рекомендаций по использованию поля Precedence в своих реализациях протоколов других уровней[2]. Однако, разработчики должны понимать, что использование поля Precedence потребует передачи его значения между протоколами разных уровней, как это делается для поля TOS.

Уровень IP **должен** обеспечивать для транспортного уровня способ установки поля TOS в каждой передаваемой дейтаграмме (по умолчанию все биты имеют нулевые значения). **Рекомендуется** для уровня IP передавать значения TOS принятых из сети пакетов на транспортный уровень.

Отображения TOS на канальном уровне, определенные в RFC 795, **не рекомендуется** применять.

Обсуждение

Хотя поле TOS мало использовалось в прошлом, планируется возрастание роли этого поля в ближайшем будущем. Предполагается, что TOS будет использоваться для управления двумя аспектами работы шлюзов - маршрутизацией и алгоритмами использования очередей. В параграфе 2 работы [INTRO:1] рассматриваются требования к прикладным программам для работы с полем TOS.

Значение TOS должно также отображаться на селекторы сервиса канального уровня. Такое решение, например, применяется для организации эффективного совместного использования последовательных каналов разными классами трафика TCP. Однако отображение, предложенное в RFC 795 для сетей, которые входили в состав Internet в 1981 году, сейчас явно устарело.

3.2.1.7 Время жизни - RFC 791, параграф 3.2

Для хостов **недопустима** передача дейтаграмм с нулевым значением времени жизни (поле TTL).

Для хостов **недопустимо** отбрасывание дейтаграмм лишь потому, что они приняты со значением поля TTL < 2.

Уровень IP **должен** обеспечивать для транспортного уровня способ установки поля TTL в каждой передаваемой дейтаграмме. При использовании фиксированного значения TTL **требуется** обеспечить возможность настройки этого значения. Рекомендуемые значения времени жизни указаны в документе Assigned Numbers[3].

Обсуждение:

Поле TTL обеспечивает две функции - ограничение времени жизни сегментов TCP (RFC 793 [TCP:1], стр. 28) и предотвращение петель в маршрутизации Internet. Хотя TTL задает время в секундах, это поле используется также в качестве счетчика интервалов (hop-count), поскольку каждый маршрутизатор должен уменьшать значение поля TTL на 1.

Смысл поля TTL состоит в том, что при уменьшении значения этого поля до 0 дейтаграмма отбрасывается маршрутизатором (но не конечным хостом). Хосты, выполняющие функции маршрутизации, должны следовать этому правилу для TTL, как маршрутизаторы.

Протокол вышележащего уровня может устанавливать значение TTL при реализации поиска в расширенной области для некоторых ресурсов Internet. Такой ход используется также некоторыми средствами диагностики и может оказаться полезным в целом ряде случаев (например, для обнаружения «ближайшего» сервера данного класса с использованием групповой адресации IP). Конкретный протокол транспортного уровня может задать свою границу TTL для максимального времени жизни дейтаграмм.

При использовании фиксированного значения времени жизни оно должно быть достаточно велико (по крайней мере, больше «диаметра» Internet - самого длинного из возможных путей). Разумно устанавливать для времени жизни двойной «диаметр» с учетом продолжающегося расширения Internet.

3.2.1.8 Опции - RFC 791, параграф 3.2

Для транспортного уровня **должен** обеспечиваться способ задания опций IP, которые будут включаться во все передаваемые дейтаграммы IP (см. параграф 3.4).

Все опции IP (за исключением NOP и END-OF-LIST) из принимаемых дейтаграмм **должны** передаваться на транспортный уровень или системе обработки ICMP (если дейтаграмма является сообщением ICMP). Оба уровня (транспортный и IP) **должны** интерпретировать понятные им опции IP, игнорируя остальные.

Ниже в этом документе рассматриваются вопросы поддержки специфических опций IP, требуемой протоколами ICMP, TCP и UDP.

Обсуждение

[1]Defense Communication Agency.
[2]В настоящее время использование поля Precedence оговорено в ряде новых RFC (1812, 2460, 2474, 2873). *Прим. перев.*
[3]Последний вариант этого документа находится в RFC 1700. В настоящее время эта информация представлена в базе данных на сайте www.iana.org. *Прим. перев.*

Передача всех принятых опций IP на транспортный уровень является преднамеренным «нарушением жесткого деления на уровни», которое предназначено для упрощения ввода в будущем новых опций IP, относящихся к транспортному уровню. Каждый уровень должен выбрать все имеющие к нему отношение опции для внутренней обработки и игнорировать остальные опции. Для этого каждая опция IP (кроме NOP и END-OF-LIST) указывает свой размер.

В этом документе не определяется порядок обработки опций каждого заголовка IP. Хосты, использующие множество опций при передаче, должны понимать, что результаты могут зависеть от порядка обработки опций.

Реализация

Программы IP-уровня должны быть устойчивы к сообщениям с опциями, размер которых выходит за допустимые пределы. Известны примеры реализаций, которые входят в бесконечный цикл при получении пакетов с полем опций некорректного размера.

Ниже перечислены требования к отдельным опциям IP:

(a) **Security** (безопасность)
Некоторые среды требуют наличия опции Security в каждой дейтаграмме - такие требования выходят за пределы настоящего документа и стандартов IP. Отметим, что опции безопасности, определенные в RFC 791 и RFC 1038, утратили силу. Для приложений DoD[1] разработчикам следует обращаться к документу [IP:8].

(b) **Stream Identifier** (идентификатор потока)
Эта опция устарела – ее **недопустимо** передавать в пакетах, а на приемной стороне она **должна** игнорироваться.

(c) **Source Route** (маршрутизация, заданная отправителем)
Хост **должен** поддерживать маршрутизацию, заданную отправителем, в исходящих дейтаграммах и **должен** поддерживать себя, как конечного адресата при маршрутизации отправителем.

Если хост получает дейтаграмму, содержащую завершенный маршрут от отправителя[2] (completed source route), это говорит о доставке дейтаграммы конечному адресату. Принятые опции (записанный маршрут) **должны** передаваться на транспортный уровень (или системе обработки сообщений ICMP). Записанные маршруты будут инвертироваться и использоваться для передачи отклика на дейтаграмму (см. Опции IP в главе 4). Когда маршрут возврата построен, **требуется** корректно сформировать его даже при использовании записи маршрута от хоста-отправителя (см. пункт (B) ниже).

Заголовки IP, содержащие более одной опции Source Route, передавать **недопустимо**, поскольку маршрутизация в таких случаях будет зависеть от реализации.

В параграфе 3.3.5 приведены правила для хостов, выступающих в качестве промежуточных (intermediate hop) для source route (т. е. пересылающих дейтаграммы с заданной отправителем маршрутизацией).

Обсуждение

При фрагментировании дейтаграмм source-route каждый фрагмент будет содержать копию заданного отправителем маршрута. Поскольку обработка опций IP (включая source route) должна предшествовать сборке фрагментов, исходная дейтаграмма не может быть собрана до тех пор, пока она не будет доставлена конечному адресату.

Предположим, что такая дейтаграмма передается от хоста S на хост D через шлюзы G1, G2, ... Gn. В спецификации IP существуют неоднозначности, позволяющие задавать для опций source route дейтаграмм, передаваемых хостом S, вариант (A) или (B):

A) {>>G2, G3, ... Gn, D} <--- корректно

B) {S, >>G2, G3, ... Gn, D} <---- ошибка

(>> представляет указатель). При использовании варианта (A) дейтаграмма, принятая хостом D, будет содержать опции {G1, G2, ... Gn >>}, с IP-адресами хостов S и D, как отправителя и получателя. Если использован вариант (B), принятая хостом D дейтаграмма будет по-прежнему содержать адреса S и D, как отправителя и получателя, но опции будут иметь вид {S, G1, ...Gn >>}, т. е; хост-отправитель исходной дейтаграммы оказывается первым в маршруте возврата.

(d) **Record Route** (запись маршрута)
Реализация установки и обработки опции Record Route является **необязательной**.

(e) **Timestamp** (временная метка)
Реализация установки и обработки опции Timestamp является **необязательной**. При реализации этой опции должны выполняться следующие правила:

- хост-отправитель **должен** записывать временную метку в поле Timestamp опций, для которых поле адреса еще не указано или содержит адрес интерфейса данного хоста;
- принимающий хост **должен** (если это возможно) добавить текущее время в опцию Timestamp перед передачей опции для обработки на транспортный уровень или ICMP;
- значение временной метки **должно** соответствовать требованиям, приведенным в параграфе 3.2.2.8 для сообщений ICMP Timestamp.

3.2.2 Протокол управляющих сообщений Internet - ICMP

Сообщения ICMP делятся на два класса[3].

- ICMP-сообщения об ошибках:
  Destination Unreachable - адресат недоступен (см. параграф 3.2.2.1)
  Redirect - перенаправление (см. параграф 3.2.2.2)
  Source Quench - «заткнуть рот отправителю» (см. параграф 3.2.2.3)

[1]Министерство обороны США. *Прим. перев.*
[2]Указатель за пределы последнего поля.
[3]С момента разработки этого документа число типов ICMP было расширено. *Прим. перев.*

Time Exceeded - время жизни истекло (см. параграф 3.2.2.4)
Parameter Problem - проблема с параметрами (см. параграф 3.2.2.5)

- Запросы ICMP:
  Echo - эхо (см. параграф 3.2.2.6)
  Information - информация (см. параграф 3.2.2.7)
  Timestamp - временная метка (см. параграф 3.2.2.8)
  Address Mask - маска адреса (см. параграф 3.2.2.9)

При получении сообщений ICMP неизвестного типа такие сообщения **должны** отбрасываться без уведомления.

Каждое сообщение ICMP об ошибке включает заголовок IP и по крайней мере первые 8 октетов дейтаграммы, с которой связана ошибка. Заголовок и данные **должны** в точности соответствовать исходной дейтаграмме, связанной с ошибкой, **возможно** включение более 8 октетов.

В тех случаях, когда уровень IP должен передавать сообщения ICMP на транспортный уровень, из исходного сообщения **должен** извлекаться номер протокола IP, используемый для выбора соответствующего объекта транспортного уровня, обеспечивающего обработку ошибок.

Сообщения ICMP об ошибках **должны** передаваться с нормальными (т. е., 0) значениями битов TOS.

Не допускается передача сообщений ICMP об ошибках в результате приема следующих пакетов[1]:

- сообщение ICMP об ошибке;
- дейтаграммы с групповым или широковещательным адресом IP;
- дейтаграммы, переданные, как широковещательные на канальном уровне;
- фрагмент, не являющийся первым;
- дейтаграммы, для которых отправитель не определен, как один хост (например, нулевой адрес, loopback-адрес, широковещательный или групповой адрес, адрес класса E).

Обсуждение

Эти правила будут предотвращать возникновение «широковещательных штормов» при получении хостом ICMP-сообщения об ошибке в ответ на широковещательную дейтаграмму. Например, широковещательный сегмент UDP, адресованный в несуществующий порт, может инициировать лавину дейтаграмм ICMP Destination Unreachable от всех машин, которые не обслуживают указанный в дейтаграмме порт. В большой сети Ethernet такая лавина может привести к остановке сети на несколько секунд в результате возникновения коллизий.

Каждая дейтаграмма, передаваемая в широковещательном режиме в подключенную сеть, должна содержать корректный широковещательный адрес IP для своих получателей (см. параграф 3.3.6). Однако, некоторые хосты не соблюдают это правило, поэтому каждый хост должен проверять широковещательные адреса канального уровня и широковещательные адреса IP.

Реализация

Канальный уровень должен информировать уровень IP о получении широковещательных для канального уровня дейтаграмм (см. параграф 2.4).

3.2.2.1 Destination Unreachable - RFC 792

Для сообщений этого типа определены дополнительные коды:

6 - неизвестна сеть адресата
7 - неизвестен хост-адресат
8 - изолированный хост-отправитель
9 - связь с сетью адресата административно запрещена
10 - связь с хостом-адресатом административно запрещена
11 - сеть недоступна для заданного типа обслуживания
12 - хост недоступен заданного типа обслуживания

Рекомендуется для хостов генерировать сообщения Destination Unreachable с кодами:

2 (Protocol Unreachable - протокол недоступен) - указанный транспортный протокол не поддерживается

3 (Port Unreachable - порт недоступен) - указанный транспортный протокол (например, UDP) не может демультиплексировать дейтаграмму и нет механизма передачи отправителю уведомления.

Принятые сообщения Destination Unreachable **должны** передаваться на транспортный уровень. Транспортный уровень **должен** использовать полученную информацию подобающим образом (см. примеры в параграфах 4.1.3.3, 4.2.3.9, 4.2.4). Транспортному протоколу, обеспечивающему собственный механизм уведомления отправителя о недоступности портов (например, TCP при передаче сегментов RST), **недопустимо** воспринимать сообщения ICMP Port Unreachable для таких же целей.

Сообщение Destination Unreachable, принятое с кодом 0 (сеть), 1 (хост) или 5 (некорректный маршрут от отправителя), может приходить от транзитного маршрутизатора и **должно** интерпретироваться, как намек (не доказательство) на то, что адресат может быть недоступен [IP:11]. В частности, такие сообщения **не могут** служить доказательством неработоспособности маршрутизатора (см. параграф 3.3.1).

3.2.2.2 Redirect - RFC 792

[1]Приведенные здесь требования имеют превосходство над всеми остальными требованиями, указанными в этом документе для передачи сообщений ICMP.

Хостам **не рекомендуется** передавать сообщений ICMP Redirect, поскольку эти сообщения являются прерогативой маршрутизаторов. Хост, получивший сообщение Redirect, **должен** соответственно скорректировать свою маршрутную информацию. Каждый хост **должен** быть готов к приему сообщений Host Redirect и Network Redirect для их обработки в соответствии с рекомендациями параграфа 3.3.1.2.

Сообщения Redirect **рекомендуется** отбрасывать без уведомления, если указанный в них новый шлюз не находится в той же сети, через которую было доставлено сообщение Redirect [INTRO:2, Приложение A], или сообщения Redirect приходят от маршрутизатора, который не указан, как first-hop (первый маршрутизатор) для данного получателя (см. параграф 3.3.1).

3.2.2.3 Source Quench - RFC 792

Хост **может** передавать сообщения Source Quench в тех случаях, когда он находится или приближается к состоянию необходимости отбрасывания входящих дейтаграмм в результате переполнения буферов сборки или нехватки других ресурсов. Более подробную информацию по этому вопросу вы сможете найти в параграфе 2.2.3 работы [INTRO:2].

При получении сообщения Source Quench уровень IP **должен** сообщить об этом транспортному уровню (или системе обработки сообщений ICMP). В общем случае транспортный или прикладной уровень **должен** обеспечивать механизм реакции на сообщения Source Quench для всех протоколов, которые могут передавать последовательность дейтаграмм одному адресату и способны поддерживать достаточно информации о состоянии, чтобы сделать возможной реакцию на такие сообщения. Обработка сообщений Source Quench протоколами TCP и UDP описана в главе 4.

Обсуждение

Сообщения Source Quench могут генерироваться хостами-получателями или некоторыми шлюзами по пути передачи дейтаграмм. Хосту, получившему сообщение Source Quench, следует снизить уровень трафика для данного получателя на некоторое время и потом постепенно восстанавливать его. Механизм реакции на сообщения Source Quench может быть реализован на транспортном (для протоколов на основе соединений типа TCP) или прикладном (для протоколов, работающих на базе UDP) уровне. В работе [IP:14] предложен механизм для уровня IP, позволяющий непосредственно реагировать на сообщения Source Quench за счет управления скоростью передачи дейтаграмм, однако это предложение пока является только экспериментальным и не может быть рекомендовано для использования.

3.2.2.4 Time Exceeded - RFC 792

Принимаемые сообщения Time Exceeded **должны** передаваться на транспортный уровень.

Обсуждение

Маршрутизаторы передают сообщение Time Exceeded с кодом 0 (в процессе передачи) при получении дейтаграмм с нулевым значением TTL. Такая ситуация может говорить о наличии петель в маршрутизации или слишком малом значении TTL при генерации дейтаграммы.

Хост может получать сообщения Time Exceeded с кодом 1 (тайм-аут при сборке) от хоста-адресата, который не смог в заданное время получить все фрагменты и собрать дейтаграмму[1]. В будущем такие сообщения могут стать частью некоторых процедур MTU discovery, используемых для определения максимального размера дейтаграмм, которые можно передать без фрагментации.

3.2.2.5 Parameter Problem - RFC 792

Для хостов **рекомендуется** генерировать сообщения Parameter Problem. Принимаемые сообщения Parameter Problem **должны** передаваться на транспортный уровень и, кроме того, информация о таких сообщениях **может** передаваться пользователю.

Обсуждение

Сообщения ICMP Parameter Problem передаются хосту-отправителю при обнаружении любых проблем, для которых нет специализированных сообщений ICMP. Появление сообщений Parameter Problem обычно служит сигналом о наличии ошибок в работе протоколов на локальном или удаленном хосте.

Ниже определяется новое значение кода для сообщений Parameter Problem:

1 = отсутствует обязательный параметр.

Обсуждение

Этот код уже используется в военных приложениях при отсутствии опций безопасности.

3.2.2.6 Echo Request/Reply - RFC 792

Каждый хост **должен** поддерживать функции сервера ICMP Echo, обеспечивающие прием запросов Echo Request и передачу в ответ на них сообщений Echo Reply. **Рекомендуется** также реализовать интерфейс прикладного уровня для передачи сообщений Echo Request и получения откликов Echo Reply с диагностическими целями.

Сообщения ICMP Echo Request, полученные с групповым или широковещательным адресом IP, **можно** отбрасывать без уведомления.

Обсуждение

Здесь приводятся беспристрастные результаты дебатов между теми, кто считает, что отклики на широковещательные запросы ICMP Echo обеспечивают эффективные возможности для диагностики, и теми, кто утверждает, что с помощью таких откликов очень легко создать пакетные бури.

IP-адрес отправителя в откликах ICMP Echo Reply **должен** совпадать с указанным адресом получателя (см. определение в 3.2.1.3) из соответствующего сообщения ICMP Echo Request.

Данные, получаемые в запросе ICMP Echo Request, **должны** полностью включаться в результирующий отклик Echo Reply. Однако, если передача Echo Reply требует преднамеренной фрагментации, которая не реализована, дейтаграмма **должна** быть усечена до размера MTU (см. параграф 3.3.3).

[1]Такие дейтаграммы отбрасываются - см. параграф 3.3.2

Сообщения Echo Reply **должны** передаваться на пользовательский интерфейс ICMP, если соответствующий запрос Echo Request не направлен на уровень IP.

Если в запросе ICMP Echo Request присутствует опция Record Route и/или TimeStamp, эта опция(и) **должна** быть обновлена с включением в маршрут текущего хоста и вставлена в заголовок IP отклика Echo Reply без «усекновения». Таким образом сохраняется записанный маршрут для замкнутого кольца.

Если в запросе ICMP Echo Request присутствует опция Source Route, маршрут возврата **должен** быть инвертирован и вставлен в поле Source Route сообщения Echo Reply.

3.2.2.7 Information Request/Reply - RFC 792

Для хостов **рекомендуется** не реализовать эти сообщения.

Обсуждение

Пара сообщений Information Request/Reply была предназначена для поддержки самонастраиваемых систем типа бездисковых станций, чтобы позволить им получать IP-адреса в процессе загрузки. Однако протоколы RARP и BOOTP обеспечивают более эффективные механизмы получения хостами IP-адресов.

3.2.2.8 Timestamp/Timestamp Reply - RFC 792

Хост **может** поддерживать сообщения Timestamp и Timestamp Reply. Если такая поддержка реализована, **должно** выполняться приведенное ниже правило.

- Функция сервера ICMP Timestamp возвращает сообщение Timestamp Reply в ответ на каждое принятое сообщение Timestamp. Если эта функция реализована, она **должна** обеспечивать минимальные вариации задержки (т. е. функция должна быть включена в ядро, чтобы избежать вариаций задержки, связанных с пользовательскими процессами).

В перечисленных ниже случаях обработка Timestamp должна соответствовать правилам для ICMP Echo.

- Сообщения Timestamp Request с групповыми и широковещательными адресами IP **можно** отбрасывать без уведомления.
- IP-адрес отправителя в сообщении Timestamp Reply **должен** совпадать с адресом получателя, указанным в соответствующем запросе Timestamp.
- При получении опции Source-route в запросе Timestamp, путь возврата **должен** инвертироваться при создании опции Source Route для отклика Timestamp Reply.
- При наличии опции Record Route и/или Timestamp в запросе Timestamp Request, эти опции **рекомендуется** обновлять с включением текущего хоста и помещать обновленное значение в поле заголовка IP для сообщения Timestamp Reply.
- Входящие сообщения Timestamp Reply должны передаваться пользовательскому интерфейсу ICMP.
  Предпочтительной формой временных меток («стандартное значение») является число миллисекунд с полуночи по стандартному времени (Universal Time). Однако временные метки с таким разрешением могут быть слишком сложны в реализации. Например, во многих системах используются часы, синхронизируемые от электросети (50 или 60 Гц), следовательно, такие системы не могут обеспечить требуемого разрешения. Поэтому допускается отклонение от стандартного значения:

  (a) «Стандартное значение» **должно** обновляться не менее 15 раз в секунду (т. е. не менее 6 младших битов должны быть неопределенными).

  (b) Точность «стандартного значения» должна приближаться к точности таймера CPU.

3.2.2.9 Address Mask Request/Reply - RFC 950

Хост **должен** поддерживать первый и **может** реализовать все три из перечисленных ниже методов определения адресных масок, соответствующих IP-адресам этого хоста.

(1) статические конфигурационные параметры;

(2) динамическое определение масок в процессе инициализации системы (см. [INTRO:1]);

(3) передача сообщений ICMP Address Mask Request и прием откликов ICMP Address Mask Reply.

Для каждого хоста **требуется** обеспечить выбор используемого метода.

При использовании метода (3) можно применять Address Mask и должны выполняться следующие условия:

(a) При инициализации хост **должен** передать с широковещательным адресом сообщение Address Mask Request в подключенную сеть. Если ответ Address Mask Reply не приходит незамедлительно, **должно** использоваться минимальное число повторов.

(b) До получения отклика Address Mask Reply хосту **рекомендуется** предполагать, что маска соответствует классу адреса данного хоста (т. е. подключенная сеть не поделена на подсети).

(c) Первое принятое сообщение Address Mask Reply **должно** использоваться для установки адресной маски, соответствующей частному локальному адресу IP. Это верно даже в тех случаях, когда первое сообщение Address Mask Reply не было запрошено (unsolicited) - в таких случаях оно будет широковещательным и может прийти после того, как хост перестал передавать повторные запросы Address Mask Request. После того, как маска была установлена с использованием Address Mask Reply, последующие сообщения Address Mask Reply **должны** игнорироваться.

Если сообщения Address Mask запрещены, запросы ICMP Address Mask Request не будут передаваться и все принятые отклики ICMP Address Mask Reply для локального IP-адреса **должны** игнорироваться.

Для хостов **рекомендуется** выполнять некоторые разумные проверки устанавливаемых адресных масок (см. «Реализация» ниже).

Недопустима передача системой откликов Address Mask Reply, если эта система не является уполномоченным агентом для адресных масок. Уполномоченный агент может быть хостом или шлюзом, но он **должен** быть явно настроен, как агент для адресных масок. Получение адресных масок с помощью Address Mask Reply не дает получателю таких прав и **не может** использоваться, как основание для передачи откликов Address Mask Reply.

При статически заданных адресных масках **рекомендуется** использовать дополнительный конфигурационный флаг, определяющий для хоста возможность выступать в качестве уполномоченного агента для масок (отвечать на запросы Address Mask Request с использованием маски).

Настроенный в качестве агента хост **должен** передать в широковещательном режиме сообщение Address Mask Reply для маски инициализируемого интерфейса.

Дополнительные сведения об использовании сообщений Address Mask Request/Reply приведены в параграфе «Инициализация хоста» работы [INTRO:1].

Обсуждение

Хосты, передающие сообщения Address Mask Reply, часто могут порождать серьезные проблемы. Для предотвращения этого отклики Address Mask Reply должны передаваться только уполномоченными агентами, явно указанными администратором сети.

Когда уполномоченный агент получает сообщение Address Mask Request, он будет передавать отклик Address Mask Reply по IP-адресу отправителя запроса. Если сетевая часть этого адреса имеет нулевое значение (см. (a) и (b) в параграфе 3.2.1.3), отклик передается в широковещательном режиме.

Не получив отклика на сообщения Address Mask Request, хост будет предполагать отсутствие агентов или подсетей, но причиной отсутствия отклика может быть временная недоступность агента. Агент будет передавать в широковещательном режиме незапрошенные сообщения Address Mask Reply при своей инициализации для обновления масок на всех хостах, которые были инициализированы в период недоступности агента.

Реализация

Предлагается выполнять следующую проверку адресной маски - в маске должны содержаться не только единицы (1) и старшие 8 битов должны быть установлены в 1 или вся маска должна быть нулевой.

3.2.3 Протокол IGMP (Internet Group Management Protocol)

Протокол IGMP [IP:4] используется хостами и маршрутизаторами одной сети для организации и поддержки членства хостов в multicast-группах. Шлюзы используют этот протокол вместе с протоколом групповой маршрутизации для поддержки группового трафика IP через Internet.

В настоящее время реализация IGMP является **необязательной** (см. параграф 3.3.7). Без использования IGMP хосты могут участвовать в multicast-группах подключенной сети.

3.3 Частные вопросы

3.3.1 Маршрутизация исходящих дейтаграмм

Уровень IP выбирает следующий маршрутизатор для каждой передаваемой дейтаграммы. Если получатель находится в подключенной сети, дейтаграмма передается на этот хост напрямую, в остальных случаях дейтаграмма направляется маршрутизатору подключенной сети.

3.3.1.1 Выбор Local/Remote

Для определения принадлежности хоста к подключенной сети **должен** использоваться следующий алгоритм [см. IP:3]:

(a) Адресная маска (применительно к локальному IP-адресу для многодомного хоста) представляет собой 32-битовое значение, позволяющее выбрать поля номеров сети и подсети в адресах IP.

(b) Если биты IP-адреса получателя, извлеченные с помощью маски[1], совпадают с битами IP-адреса отправителя, полученными с такой же маской, это говорит о принадлежности хоста к подключенной сети и дейтаграмма передается напрямую хосту-получателю.

(c) Если условие (b) не выполняется, для доставки дейтаграммы должен использоваться шлюз, определяемый в соответствии с требованиями параграфа 3.3.1.2.

Для некоторых специальных случаев используется иной алгоритм:

- для групповых и широковещательных адресов ограниченного действия дейтаграммы просто передаются на канальный уровень через соответствующий интерфейс;
- Для широковещательных дейтаграмм, адресованных всей сети или подсети, могут использоваться стандартные алгоритмы маршрутизации.

IP-уровень хоста **должен** корректно работать в минимальной сетевой среде (в частности, при отсутствии маршрутизаторов). Если IP-уровень хоста упорно ищет хотя бы один шлюз при инициализации, такой хост не сможет работать в изолированной сети.

3.3.1.2 Выбор шлюза

Для эффективной маршрутизации группы дейтаграмм одному получателю хост-отправитель **должен** сохранять кэш маршрутов. Хост использует описанный ниже алгоритм для маршрутизации дейтаграмм с использованием кэша (алгоритм предназначен прежде всего для переноса основного бремени маршрутизации на шлюзы) [IP:11].

[1]Операция AND. *Прим. перев.*

- Если кэш маршрутов не содержит информации для интересующего адреса, хост выбирает принятый по умолчанию шлюз и передает дейтаграмму этому маршрутизатору. Соответствующая запись вносится в кэш маршрутов.

- Если шлюз не является лучшим путем к адресату, он должен будет переслать дейтаграмму наиболее подходящему маршрутизатору и вернуть хосту-отправителю сообщение ICMP Redirect.

- Получив сообщение Redirect, хост обновляет шлюз в соответствующей записи кэша для того, чтобы последующие дейтаграммы доставлялись по более эффективному пути.

Поскольку маска сети для адреса получателя в общем случае не известна, сообщения Network Redirect **должны** трактоваться аналогично сообщениям Host Redirect, т.е. запись в кэше для хоста-получателя (и только для него) будет обновляться (или создаваться, если ранее ее не было) с учетом нового шлюза.

Обсуждение

Эта рекомендация предназначена для защиты от шлюзов, передающих сообщения Network Redirect в сеть с подсетями в нарушение требований к маршрутизаторам [INTRO:2].

При отсутствии в кэше записи для адреса получателя (если адресат не находится в подключенной сети) уровень IP **должен** выбрать маршрутизатор из своего списка принятых по умолчанию шлюзов. Уровень IP **должен** поддерживать множество используемых по умолчанию шлюзов.

В качестве дополнительной возможности уровень IP хоста **может** реализовать таблицу статических маршрутов. Каждый статический маршрут **может** включать флаг, определяющий возможность переписывания этого маршрута с помощью ICMP Redirect.

Обсуждение

Для начала работы хосту требуется знать по крайней мере один используемый по умолчанию шлюз. Эту информацию можно получить из конфигурационного файла или загрузочного сценария (например, BOOTP [INTRO:1]).

Предполагается, что хост может расширять список используемых по умолчанию шлюзов, добавляя в него маршрутизаторы по мере их обнаружения. Например, хост может записывать каждый шлюз, на который пересылает пакеты. Такое решение может оказаться очень эффективным для некоторых случаев, но в иных ситуациях (не все маршрутизаторы одинаковы) оно может порождать проблемы и, по этой причине, не рекомендуется.

Статический маршрут представляет собой отображение хоста или сети на тот или иной следующий маршрутизатор; маршрут может также зависеть от типа обслуживания (ToS), рассматриваемого в следующем параграфе. Статические маршруты устанавливаются администратором сети взамен обычных механизмов автоматической маршрутизации и для обслуживания исключительных ситуаций. Однако следует помнить, что статические маршруты являются потенциальным источником ошибок при изменении конфигурации или повреждениях оборудования.

3.3.1.3 Кэш маршрутов

Каждая запись в кэше маршрутов должна содержать следующие поля:

(1) локальный IP-адрес (для многодомных хостов);

(2) IP-адрес получателя;

(3) тип(ы) обслуживания ToS;

(4) IP-адрес следующего маршрутизатора.

Поле (2) **может** содержать полный адрес получателя или только адрес сети, в которую тот входит. Поле TOS (3) **должно** присутствовать в записи.

Процедуры работы с кэшем маршрутов описаны в параграфе 3.3.4.2.

Обсуждение

Включение поля TOS в кэш маршрутов и его рассмотрение в алгоритме маршрутизации будет обеспечивать механизм для применения в будущем маршрутизации по типу обслуживания в сети Internet (см. параграф 3.2.1.6).

Каждая запись в кэше определяет конечную точку пути через Internet. Хотя маршрут между двумя точками может динамически изменяться, характеристики пути передачи остаются почти неизменными в течение продолжительного времени для каждого транспортного соединения между парой хостов.

Следовательно, запись в кэше маршрутов является естественным местом для хранения информации о свойствах пути. Примером такого свойства может служить максимальный размер нефрагментируемой дейтаграммы (см. параграф 3.3.3) или средняя задержка на круговом пути, измеренная транспортным протоколом.

Эти данные в общем случае собираются и используются протоколами вышележащих уровней (например, TCP) или приложениями, использующими протокол UDP. В настоящее время ведутся эксперименты по кэшированию свойств путей описанным здесь способом.

Существуют разногласия по вопросу использования ключей для кэша - только адрес получателя или оба адреса (получателя и отправителя). Сторонники использования только адресов получателей приводят следующие аргументы:

(1) в соответствии с требованиями параграфа 3.3.1.2 сообщения Redirect будут порождать записи, ключами к которым являются адреса получателей; простейшая и наиболее общая схема всегда будет использовать адреса хостов;

(2) уровень IP не всегда может знать адресную маску для сети получателя в сложной среде с подсетями;

(3) использование только адресов хостов-получателей позволит применять полные 32-битовые адреса, обеспечивая восприимчивость к изменению архитектуры Internet.

Сторонники использования в качестве ключей адресов отправителей и получателей также приводят свои аргументы:

(1) экономия памяти;

(2) упрощение структуры данных, простота объединения с таблицами принятых по умолчанию и статических маршрутов (см. ниже);

(3) обеспечивается больше полезного места для хранения информации о свойствах пути, как было упомянуто выше.

Реализация

Кэш должен быть достаточно велик и обеспечивать возможность включения записей для максимального числа хостов-получателей, которое может использоваться в каждый момент времени.

Маршрутная запись в кэше может также включать управляющую информацию, используемую для выбора заменяемой записи. Это может быть реализовано, например, в форме бита recently used (недавно использована) или временной метки для последнего обращения. В целях диагностики рекомендуется сохранять время последнего изменения записи.

При реализации может возникнуть желание снизить издержки на сканирование кэша маршрутов для каждой передаваемой дейтаграммы. Это можно реализовать с помощью хэш-таблицы для ускорения просмотра или путем предоставления ориентированными на соединения протоколами транспортного уровня «советов» или временных указателей на подходящую запись в кэше уровню IP с каждой последовательной дейтаграммой.

Хотя мы рассматривали здесь кэш маршрутов и список используемых по умолчанию шлюзов по-отдельности, на практике их часто объединяют в одну структуру данных - таблицу маршрутизации.

3.3.1.4 Обнаружение «мертвых» шлюзов

Уровень IP **должен** обеспечивать возможность обнаружения неработающих маршрутизаторов на следующем интервале, включенных в кэш маршрутов, и выбора других маршрутов взамен поврежденных (см. параграф 3.3.1.5).

Процесс обнаружения сбойных маршрутизаторов детально рассмотрен в RFC 816 [IP:11]. До сегодняшнего дня не разработано полного алгоритма, обеспечивающего эффективное обнаружение сбойных маршрутов, хотя предложен целый ряд методик.

- **Не рекомендуется** использовать конкретный маршрутизатор при отсутствии признаков его работоспособности.
- Активные средства проверки типа ping (т.е., использование сообщений ICMP Echo Request/Reply) слишком накладны и не обеспечивают требуемого масштабирования. В частности, следует отметить, что **недопустимо** проверять состояние первого маршрутизатора путем непрерывной передачи по его адресу запросов ICMP.
- Даже при отсутствии других способов проверки состояния маршрутизатора ping **следует** применять только в случаях отсутствия подтверждений работы маршрутизатора при передаче ему реального трафика, что позволяет усомниться в работоспособности маршрутизатора.
- Чтобы избежать использования ping уровни выше и ниже IP **должны** обеспечивать возможность получения сведений о состоянии пути в кэше маршрутов за счет использования позитивной (маршрутизатор работает) или негативной информации о состоянии шлюза.

Обсуждение

Если реализация не включает адекватного механизма обнаружения неработающих маршрутизаторов, сбой в маршрутизаторе будет приводить к пропаданию пакетов в «черной дыре». Такие ситуации вызывают массу нареканий со стороны пользователей и весьма сложны для обнаружения.

Механизм обнаружения «мертвых» маршрутизаторов не должен создавать неприемлемой нагрузки на хост, подключенную сеть или соседние маршрутизаторы. Продолжительность детектирования и приемлемый уровень нагрузки в некоторой степени зависят от характера использования хоста, но в общем случае требуется достаточно быстро находить повреждение в ближайшем маршрутизаторе, чтобы не нарушалась работа приложений транспортного уровня, не использующих прямых соединений, за время детектирования сбоя и поиска альтернативного пути.

Передача информации от соседних уровней стека протоколов усложняет интерфейс между уровнями, но может существенно упростить обнаружение сбойных маршрутизаторов. Информация может приходить почти от всех элементов архитектуры IP/TCP, но наиболее важны сведения от транспортного и канального уровней. Ниже приведены примеры такой информации.

- TCP или другой протокол на основе соединений должен обеспечивать возможность получения негативной информации (например, слишком большое число повторных передач).
- TCP может давать позитивную информацию после получения (нового) подтверждения о доставке данных. Даже при асимметричном маршруте такое подтверждение свидетельствует об успешной передаче.
- Сообщения ICMP Redirect от конкретных маршрутизаторов должны использоваться, как позитивные сведения о работе шлюза.
- Информация канального уровня, который способен эффективно детектировать сбои и сообщать о них (например, с помощью сообщений ARPANET Destination Dead), должна использоваться, как негативные сведения.
- Сбои в работе ARP или при проверке отображений (преобразований) ARP могут использоваться, как негативная информация для соответствующих адресов IP.
- Поступление пакетов от конкретного адреса канального уровня является подтверждением работы соответствующего устройства. Однако включение таких данных в сведения о работоспособности шлюзов требует отображения адресов канального уровня на адреса IP и последующей проверки принадлежности этих адресов указанным в кэше маршрутов шлюзам. Такое решение может оказаться недостаточно эффективным.

Отметим, что использование позитивных сведений от каждой полученной дейтаграммы может привести к чрезмерной загрузке системы.

Сведения могут передаваться с использованием требуемых аргументов во все интерфейсы с IP-уровнем, однако некоторые транспортные и прикладные протоколы не способны генерировать корректные анонсы. Такие

интерфейсы, следовательно, должны поддерживать нейтральные сведения, поскольку передача анонса с неверным знаком (позитив - негатив) может привести к некорректной работе системы.

Существуют (и широко распространен) иной метод определения «мертвых» маршрутизаторов, но его использование не рекомендуется. Этот метод основан на прослушивании хостом (в пассивном режиме) дейтаграмм протокола IGP[1], которыми маршрутизаторы обмениваются между собой с использованием широковещательных адресов. Такое решение имеет существенный недостаток - хосты должны распознавать все протоколы внутренних шлюзов, которые маршрутизатор может использовать (см. [INTRO:2]). Кроме того, такой вариант будет работать только в широковещательной сети.

В настоящее время ping (т. е., использование сообщений ICMP Echo) используется, как механизм проверки работоспособности маршрутизаторов только при возникновении абсолютной необходимости. Отклики на ping гарантируют работоспособность проверяемого интерфейса и связанной с ним машины, но они не гарантируют возможности передачи дейтаграмм в другие интерфейсы маршрутизатора. При наличии сообщений Redirect или иных явных признаков того, что машина является шлюзом, отклики на ping будут говорить о том что маршрутизатор успешно выполняет свои функции. Однако отбрасывание хостом без уведомления пакетов, которые маршрутизатор должен пересылать или перенаправлять, может приводить к тому, что такое предположение перестанет быть верным. Чтобы избежать подобных проблем, разрабатывается новый тип сообщений «are you a gateway?» (это маршрутизатор?).

Реализация

Для обнаружения «мертвых» маршрутизаторов предлагается следующий алгоритм:

- Связать таймер «повторной маршрутизации» с каждым шлюзом, на который указывает запись в кэше маршрутов. При инициализации таймера устанавливается значение Tr, которое должно быть достаточно мало, чтобы можно было обнаружить неработающий маршрутизатор до того, как транспортное соединение будет разорвано по тайм-ауту.
- Позитивные сведения будут сбрасывать таймер в Tr, а негативные - обнулять таймер.
- Всякий раз, когда уровень IP используется для маршрутизации дейтаграммы, должно проверяться состояние таймера. Если таймер содержит нулевое значение, уровень IP будет использовать ping для данного шлюза.
- Пакеты ping (ICMP Echo) можно при необходимости повторять до N раз. Если за N попыток не было получено ни одного отклика, делается вывод о неработоспособности маршрутизатора и в кэше должен указываться новый шлюз для всех записей, указывающих на сбойный маршрутизатор.

Отметим, что значение Tr обратно пропорционально числу возможных анонсов. Значение Tr должно быть достаточно мало, чтобы обеспечить следующие условия:

- пакеты ping должны составлять достаточно малую часть (например, <10%) от всех пакетов, передаваемых маршрутизатору с данного хоста;
- пакеты ping должны передаваться достаточно редко (например, каждые 3 минуты).

Поскольку описанный алгоритм имеет дело с маршрутизаторами, на которые указывают записи в кэше маршрутов, а не с самими записями, для реализации этого алгоритма желательно использовать двухуровневую структуру кэша (например, по типу кэша ARP).

3.3.1.5 Выбор нового шлюза

Если прекративший работать шлюз не является в настоящий момент используемым по умолчанию, уровень IP может незамедлительно переключиться на работу с принятым по умолчанию маршрутизатором. При сбое в работе установленного по умолчанию шлюза уровень IP **должен** выбрать другой маршрутизатор для использования по умолчанию (предполагается, что может существовать несколько таких шлюзов) на прекратившем работу пути и организовать новые маршруты.

Обсуждение

В случае прекращения работы маршрутизатора другие шлюзы подключенной сети узнают об этом с помощью некоторых протоколов обмена информацией между маршрутизаторами. Однако, это не происходит моментально, поскольку протоколы маршрутизации имеют время обмена порядка 30-60 секунд. Если хост переключится на другой маршрут до того, как выбранный маршрутизатор узнает о сбое, этот маршрутизатор может попытаться переслать дейтаграмму вышедшему из строя шлюзу и послать отправителю дейтаграммы сообщение Redirect, указывающее на «мертвый» маршрутизатор (!). В результате этого может начаться процесс автогенерации обновлений кэша маршрутов до того, как маршрутизатор узнает о прекращении работы другого шлюза. Для предотвращения таких ситуаций логика обнаружения «мертвых» маршрутизаторов должна обеспечивать некоторый гистерезис. Однако на практике упомянутые случаи автогенерации обновлений случаются редко, поскольку обслуживание хоста не может быть восстановлено до тех пор, пока шлюз не организует маршрутную информацию.

Реализация

Одним из вариантов реализации выбора нового маршрута по умолчанию является простой циклический перебор хостом своего списка используемых по умолчанию шлюзов. Другим вариантом является ранжирование маршрутизаторов по уровню приоритета и использование по умолчанию шлюза с максимальным приоритетом. Если в данный момент текущим является маршрутизатор не с высшим приоритетом, хост использует ping (достаточно редкие запросы), чтобы обнаружить восстановление работы маршрутизатора с высшим приоритетом. Частота запросов может быть очень низкой, порядка одного запроса в 3 минуты.

3.3.1.6 Инициализация

Должна обеспечиваться возможность настройки следующих параметров:

(1) IP-адреса;

(2) маски адресов;

[1]Interior Gateway Protocol - протокол внутренней маршрутизации.

(3) список используемых по умолчанию шлюзов с уровнем приоритета для каждого.

Должна обеспечиваться возможность ручной настройки перечисленных параметров и, кроме того, могут использоваться различные методы динамического определения параметров (см. параграф «Инициализация хоста» в [INTRO:1]).

Обсуждение

Некоторые реализации хостов прослушивают протоколы маршрутизации в широковещательной сети для обнаружения шлюзов. Стандартный метод определения используемых по умолчанию шлюзов находится в стадии разработки[1].

3.3.2 Сборка фрагментов

Уровень IP **должен** обеспечивать сборку фрагментов дейтаграмм IP.

Будем обозначать максимальный размер дейтаграммы, которая может быть собрана, как EMTU_R[2]; иногда используется термин «размер буфера сборки». Значение EMTU_R **должно** быть не менее 576 (октетов) и **рекомендуется** обеспечивать возможность настройки этого значения или использования неограниченного буфера сборки. Кроме того, это значение **рекомендуется** делать не меньше, чем значение MTU для подключенных сетей.

Обсуждение

Фиксированное значение предела EMTU_R не должно встраиваться в программный код, поскольку некоторые протоколы прикладного уровня требуют использования EMTU_R > 576.

Реализация

При реализации можно использовать непрерывный буфер сборки для каждой дейтаграммы или применять более сложные структуры данных, позволяющие собирать дейтаграммы неопределенно большого размера; в последнем случае говорят о неограниченном EMTU_R.

Логически сборка представляет собой просто копирование каждого фрагмента в буфер с использованием указанного смещения. Отметим, что фрагменты дейтаграмм могут перекрываться в результате повторной передачи после нового фрагментирования с сохранением идентификатора.

Некоторую хитрость при сборке представляет учет для определения момента, когда собраны все фрагменты дейтаграммы. Мы рекомендуем алгоритм Кларка [IP:10], не требующий дополнительного пространства памяти для учета. Однако следует отметить, что в отличие от сказанного в [IP:10], заголовок первого фрагмента должен быть сохранен для включения в возможное сообщение ICMP Time Exceeded (тайм-аут при сборке).

Должен обеспечиваться механизм, посредством которого транспортный уровень будет определять значение MMS_R - максимальный размер сообщения, которое может быть принято и собрано в дейтаграмму IP (см. функцию GET_MAXSIZES в параграфе 3.4). Если значение EMTU_R можно определить, величина MMS_R определяется, как MMS_R = EMTU_R - 20[3].

Для сборки **должно** задаваться максимальное время (тайм-аут). Значение тайм-аута **рекомендуется** делать фиксированным, а не привязывать его к оставшемуся времени жизни (TTL). Рекомендуется устанавливать тайм-аут в диапазоне 60 - 120 секунд. По истечении заданного времени частично собранные дейтаграммы **должны** отбрасываться с передачей сообщений ICMP Time Exceeded хосту-отправителю (если получен начальный фрагмент).

Обсуждение

Спецификация IP говорит, что тайм-аут для сборки должен быть равен оставшемуся времени жизни дейтаграммы (TTL) из заголовка IP, но такое решение не работает должным образом, поскольку маршрутизаторы в общем случае трактуют TTL просто как счетчик интервалов, а не время доставки. Если тайм-аут для сборки слишком мал, дейтаграммы будут отбрасываться без нужды, что приведет к излишней загрузке коммуникационных каналов. Значение тайм-аута должно быть не меньше среднего времени доставки пакетов через Internet. Реальная оценка минимального значения тайм-аута для сборки фрагментов составляет 60 секунд.

Предлагается сохранять в кэше значения времени на передачу дейтаграмм туда и обратно, измеряемые протоколами транспортного уровня, и использовать эти значения для определения величины тайм-аута при сборке. Однако для применения таких методов на практике требуются дополнительные исследования.

При установке слишком большого значения тайм-аута на принимающем хосте может не хватить выделенного для буферов пространства и время жизни сегмента MSL[4] [TCP:1] станет слишком велико. Значение MSL управляет максимальной скоростью, с которой могут передаваться фрагменты дейтаграмм при использовании различных значений 16-битового поля идентификации (увеличение MSL снижает максимальную скорость). Спецификация TCP [TCP:1] предполагает для MSL значение 2 минуты. Эта величина устанавливает верхний предел для тайм-аута при сборке.

3.3.3 Фрагментация

Уровень IP **может** реализовать механизм преднамеренной фрагментации дейтаграмм.

Будем обозначать максимальный размер передаваемой дейтаграммы IP для конкретной комбинации отправитель - получатель (и возможно TOS), как EMTU_S[5].

Хост **должен** реализовать механизм, позволяющий транспортному уровню выяснять значение MMS_S (максимальный размер сообщения транспортного уровня, которое может быть передано) для данной комбинации {отправитель, получатель, TOS} (см. функцию GET_MAXSIZES в параграфе 3.4). Если локальной фрагментации не производится, должно выполняться условие:

[1]См. RFC 1256 ICMP Router Discovery messages. *Прим. перев.*
[2]Effective MTU to receive - эффективное значение MTU для приема.
[3]Минимальный размер заголовка IP.
[4]Maximum Segment Lifetime.
[5]Effective MTU for sending - эффективное значение MTU для передачи.

```
MMS_S = EMTU_S - <размер заголовка IP>
```

и значение EMTU_S не должно быть больше MTU для сетевого интерфейса, соответствующего адресу отправителя дейтаграммы. Отметим, что значение <размер заголовка IP> в этом выражении будет равно 20, если IP не резервирует пространство для вставки опций IP в дополнение к опциям, устанавливаемым на транспортном уровне.

Хост, не реализующий локальную фрагментацию, **должен** обеспечивать получение транспортным (для TCP) или прикладным (для UDP) уровнем значения MMS_S от уровня IP и передачу дейтаграмм, размер которых не превышает MMS_S.

В общем случае желательно избегать локальной фрагментации и выбирать значение EMTU_S достаточно небольшим для того, чтобы избежать фрагментации на любом из шлюзов по пути доставки. При отсутствии актуальной информации о минимальном значении MTU для пути уровень IP **должен** использовать значение EMTU_S <= 576, если получатель не находится в подключенной сети (для этого случая используется принятое в сети значение MTU).

Значение MTU для каждого физического интерфейса **должно** быть настраиваемым.

Реализация уровня IP может использовать флаг конфигурации All-Subnets-MTU (MTU для всех подсетей), показывающий, что значение MTU в подключенной сети будет использоваться и для других подсетей этой сети (но не для других сетей). Таким образом, этот флаг заставляет использовать маску сети взамен маски подсети для выбора значения EMTU_S. Для многодомных хостов флаг All-Subnets-MTU требуется для каждого сетевого интерфейса.

Обсуждение

Выбор корректного размера дейтаграмм для передачи данных является сложной задачей [IP:9].

(a) В общем случае от хоста не требуется прием дейтаграмм IP размером более 576 байтов (включая заголовок) и хост не должен передавать дейтаграмм большего размера без предварительного явного согласования с хостом-получателем. Таким образом, MMS_S задает только верхнюю границу размера дейтаграмм, которые может передавать транспортный уровень. Даже при MMS_S > 556 транспортный уровень должен ограничивать свои сообщения размером 556 байтов при отсутствии информации от хоста-получателя о возможности принимать более крупные сообщения.

(b) Некоторые транспортные протоколы (например, TCP) обеспечивают возможность явного уведомления отправителя о максимальном размере дейтаграмм, которые другая сторона может принимать и собирать [IP:7]. На уровне IP подобного механизма не существует.
Транспортный протокол, предполагающий EMTU_R > 576 (см. параграф 3.3.2), может передавать дейтаграммы большого размера другим хостам, поддерживающим такой же протокол.

(c) В идеальном случае хост должен ограничивать свое значение EMTU_S для данного получателя до минимального значения MTU во всех сетях на пути к получателю во избежание фрагментации. Фрагментация IP, будучи формально корректной, может существенно снижать производительность протокола транспортного уровня, поскольку потеря одного фрагмента будет требовать повторной передачи всех фрагментов сообщения [IP:9].

Поскольку практически все сети в среде Internet поддерживают MTU=>576, настоятельно рекомендуется использовать значение 576 для дейтаграмм, передаваемых за пределы локальной сети.

Для определения MTU на данном пути предлагается передавать фрагмент дейтаграммы с нулевым смещением и ожидать отклика ICMP Time Exceeded в результате тайм-аута при сборке (остальные фрагменты не передаются). Такое сообщение будет содержать заголовок самого большого из оставшихся фрагментов в своем теле. Ведутся эксперименты и с более прямыми механизмами, но они еще не адаптированы (см. например RFC 1063).

3.3.4 Локальные многодомные хосты

3.3.4.1 Введение

Многодомный хост имеет множество адресов IP, которые можно рассматривать, как логические интерфейсы. Эти интерфейсы могут быть связаны с одним или различными физическими интерфейсами, а последние могут быть соединены с одной или разными сетями.

Различается несколько вариантов многодомных хостов:

(a) *Множество логических сетей*
Создатели архитектуры Internet предполагали, что каждая физическая сеть будет иметь уникальный номер IP-сети (или подсети). Однако администраторы локальных сетей иногда считают полезным отказ от такого допущения и организуют в одной физической ЛВС множество логических сетей.
Если хост, подключенный к такой физической сети, настроен на обслуживание трафика каждой из N различных логических сетей, этот хост будет иметь N логических интерфейсов. Эти интерфейсы могут использовать один или множество физических интерфейсов для подключения к одной физической сети.

(b) *Множество логических хостов*
Когда хост имеет множество адресов IP с одинаковой сетевой частью (или одинаковым номером подсети), используется понятие логического хоста. Логические интерфейсы хоста могут использовать один или множество физических интерфейсов.

(c) *Простой вариант*
В этом случае каждый логический интерфейс отображается на отдельный физический интерфейс, подключенный к своей физической сети. Термин «многодомный» изначально относился только к таким хостам, но сейчас толкование термина существенно расширилось.
Хост со встроенными функциями маршрутизации обычно представляет собой простой вариант многодомного хоста. Отметим, однако, что многодомный хост не обязан поддерживать функций маршрутизации (т. е. он может не реализовать пересылку пакетов из одной подключенной сети в другую).
Простой многодомный хост представляет наиболее серьезную проблему маршрутизации. Выбор интерфейса (т.е., сети first-hop) может существенно влиять на производительность и даже на доступность удаленных частей Internet.

В заключение отметим также обратный случай (не многодомный хост), когда один логический интерфейс объединяет несколько физических интерфейсов (например, в целях повышения надежности или пропускной способности за счет организации параллельных каналов передачи данных между двумя точками сети две системы могут быть соединены множеством каналов «точка-точка»). Такой вариант мы будем называть мультиплексированием на канальном уровне. При таком мультиплексировании протоколы, расположенные выше канального уровня, просто не знают о наличии множества физических интерфейсов - драйвер устройства на канальном уровне отвечает за мультиплексирование и маршрутизацию пакетов через разные физические интерфейсы.

В архитектуре протоколов Internet элемент протокола транспортного уровня не имеет собственного адреса, используя взамен общий адрес IP. Это задача уровня IP - обеспечивать интерфейс для транспортного и прикладного уровней. В частности, прикладная программа может знать о множестве адресов IP многодомного хоста и выбирать один из них для своего использования.

3.3.4.2 Требования для многодомных хостов

Приведенные здесь правила применяются при выборе IP-адреса отправителя для передачи дейтаграмм многодомными хостами.

(1) Если дейтаграмма передается в ответ на принятую дейтаграмму, адрес отправителя **должен** совпадать с адресом получателя в принятой дейтаграмме. Более подробное описание требования для вышележащих уровней приведено в параграфах 4.1.3.5 и 4.2.3.7, а также в параграфе «Общие вопросы» [INTRO:1].

В остальных случаях адрес отправителя можно выбирать.

(2) Приложение **должно** быть способно явно указывать адрес отправителя при организации соединения или запросе.

(3) При отсутствии такой спецификации сетевые программы **должны** выбирать адрес отправителя в соответствии с приведенными ниже правилами.

С многодомными хостами связаны два ключевых вопроса:

A) Хост **может** отбрасывать без уведомления входящие дейтаграммы, в которых адрес получателя не соответствует физическому интерфейсу, принявшему дейтаграмму.

B) Хост **может** ограничить себя передачей дейтаграмм IP (не source-route) только через физический интерфейс, соответствующий IP-адресу отправителя в дейтаграмме.

Обсуждение

Разработчики программ для хостов Internet используют две различных по концепции модели многодомных хостов, кратко рассмотренные ниже. В этом документе не отдается преимущества какой-либо модели - обе имеют свои плюсы и минусы. Различия между этими моделями, рассмотренные в (A) и (B), являются опциональными.

- *Модель Strong ES*

  Модель Strong ES[1] придает важное значение различиям между хостами и маршрутизаторами (ES/IS) и применительно к ней следует изменить **может** на **должен** в пунктах (A) и (B), рассмотренных выше. В этой модели многодомный хост представляется как множество логических хостов в одном физическом компьютере.
  Применительно к (A) сторонники модели Strong ES отмечают, что механизм автоматической маршрутизации Internet не обеспечивает маршрутизации дейтаграмм в физические интерфейсы, которые не соответствуют адресу получателя.
  В модели Strong ES расчет маршрута для исходящих дейтаграмм представляет собой отображение:

  ```
  маршрут(IP-адрес отправителя, IP-адрес получателя, TOS) -> шлюз
  ```

  Адрес отправителя включен, как параметр для того, чтобы выбрать шлюз, напрямую доступный через соответствующий физический интерфейс. Отметим, что эта модель логически требует по крайней мере одного принятого по умолчанию шлюза и предпочитает иметь множество таких шлюзов для каждого IP-адреса отправителя.

- *Модель Weak ES*

  В этой модели различия между хостами и маршрутизаторами не считаются существенными и следует использовать **недопустимо** вместо **может** для приведенных выше пунктов (A) и (B). Эта модель может быть более естественной для хостов, прослушивающих протоколы маршрутизации, и просто необходима для хостов с поддержкой встроенной маршрутизации.

  Модель Weak ES может приводить к сбоям в работе механизма Redirect. Если дейтаграмма передана физическому интерфейсу, который не соответствует адресу получателя, первый маршрутизатор не сможет понять, когда ему нужно отправить сообщение Redirect. С другой стороны, хост поддерживающий функции маршрутизации, может получать маршрутную информацию без использования сообщений Redirect.

  В модели Weak ES расчет маршрута для исходящих дейтаграмм представляет собой отображение:

  ```
  маршрут(IP-адрес получателя, TOS) -> шлюз, интерфейс
  ```

3.3.4.3 Выбор адреса отправителя

Обсуждение

При передаче начального запроса соединения (например, сегмент TCP SYN) или дейтаграммы запроса обслуживания (например, UDP-запрос) транспортный уровень многодомного хоста должен знать, какой адрес отправителя нужно использовать. Если приложение не задало этот адрес, транспортный уровень должен запросить у IP-уровня концептуальное отображение:

```
GET_SRCADDR(удаленный IP-адрес, TOS) -> локальный IP-адрес
```

Значение TOS задает тип обслуживания (см. 3.2.1.6) и результатом является нужный адрес отправителя. Для реализации такого отображения предлагаются следующие правила:

[1]End System - конечная система, т.е., хост

(a) когда удаленный адрес Internet относится к одной из (под)сетей, с которыми хост непосредственно соединен, может быть выбран соответствующий адрес отправителя, если нужный интерфейс находится в рабочем состоянии;

(b) можно просмотреть кэш маршрутов для поиска активного маршрута в интересующую сеть через любой сетевой интерфейс; если такой маршрут найден, можно выбрать локальный адрес IP, соответствующий интерфейсу;

(c) аналогичным образом можно использовать и таблицу статических маршрутов (см. 3.3.1.2);

(d) можно просмотреть список используемых по умолчанию шлюзов - если такой шлюз связан с другими интерфейсами, можно выбрать шлюз с максимальным приоритетом.

В будущем для многодомных хостов может быть определен способ передачи маршрутизаторам всех подключенных сетей запросов для определения сети, наиболее подходящей для данного получателя.

Реализация

Отметим, что описанный процесс по сути совпадает с маршрутизацией дейтаграмм (см. 3.3.1) и, следовательно, хост может объединять реализацию этих функций.

3.3.5 Пересылка Source Route

С учетом приведенных ниже ограничений хост **может** выступать в качестве промежуточного интервала в маршруте source route, пересылая маршрутизируемые отправителем дейтаграммы на следующий указанный хост.

Однако, при выполнении таких функций квази-маршрутизации хост **должен** соответствовать всем требованиям, предъявляемым к шлюзам при пересылке дейтаграмм source route [INTRO:2]. Эти требования имеют более высокий приоритет, нежели рассмотренные выше требования к хостам.

A) TTL (см. параграф 3.2.1.7)

Значение поля TTL **должно** декрементироваться и дейтаграмма может быть отброшена в соответствии с требованиями к шлюзам [INTRO:2].

B) ICMP Destination Unreachable (см. параграф 3.2.2.1)

Хост **должен** быть способен генерировать сообщения Destination Unreachable со следующими кодами:

4 (Fragmentation Required but DF Set), если дейтаграмма source route не может быть фрагментирована в соответствии с требованиями сети получателя;

5 (Source Route Failed), если дейтаграмму source route невозможно переслать (например, из-за проблем с маршрутизацией или в связи с тем, что следующий интервал при строгом задании маршрута - strict source route - не находится в подключенной сети).

C) IP-адрес отправителя (см. параграф 3.2.1.3)

Маршрутизируемые отправителем дейтаграммы при их пересылке **могут** иметь (в нормальных условиях имеют) адрес отправителя, который не является одним из IP-адресов пересылающего хоста.

D) Опция Record Route (см. параграф 3.2.1.8d)

Хост, пересылающий дейтаграммы source route, которые содержат опцию Record Route, **должен** обновлять значение этого поля, вписывая туда информацию о себе.

E) Опция Timestamp (см. параграф 3.2.1.8e)

Хост, пересылающий дейтаграммы source route, которые содержат опцию Timestamp, **должен** добавлять в нее текущую временную метку в соответствии с правилами для этой опции.

Для определения правил, регулирующих работу хостов при пересылке дейтаграмм source route, мы будем использовать термин «локальная обработка», если следующий шлюз доступен через тот же физический интерфейс, который принял дейтаграмму; в остальных случаях будет использоваться термин «нелокальная обработка».

- Хост может выполнять локальную обработку без каких-либо ограничений.
- Хост, поддерживающий нелокальную обработку, **должен** иметь конфигурационную опцию, позволяющую запретить пересылку и по умолчанию пересылка должна быть отключена.
- Хост **должен** соответствовать всем требованиям к шлюзам в части настройки политики фильтрации ([INTRO:2]), ограничивающей нелокальную обработку.

Когда хост получает дейтаграмму с незавершенным маршрутом source route, но по тем или иным причинам не пересылает ее, он **должен** послать сообщение ICMP Destination Unreachable (код 5, Source Route Failed) отправителю дейтаграммы, если сама дейтаграмма не является сообщением ICMP об ошибке.

3.3.6 Широковещание

В параграфе 3.2.1.3 определены 4 стандартных формы широковещательных адресов IP:

Limited Broadcast - ограниченная область: {-1, -1}

Directed Broadcast - широковещание для сети: {<Номер сети>,-1}

Subnet Directed Broadcast - широковещание для подсети: {<Номер сети>,<Номер подсети>,-1}

All-Subnets Directed Broadcast - широковещание для всех подсетей: {<Номер сети>,-1,-1}

Хост **должен** распознавать все эти форматы в поле получателя принимаемых дейтаграмм.

Существует класс хостов[1], использующих нестандартный формат широковещательных адресов (0 взамен -1). Для всех хостов **рекомендуется** распознавать и принимать такие нестандартные форматы в полях адреса получателя для

[1]4.2BSD Unix и производные, но не 4.3BSD.

входящих дейтаграмм. Хост **может** использовать конфигурационную опцию для выбора формата (0 или -1) на каждом физическом интерфейсе, но по умолчанию должна применяться стандартная форма (-1).

Когда хост отправляет дейтаграмму по широковещательному адресу канального уровня, IP-адрес получателя **должен** быть корректным широковещательным или групповым адресом IP.

Для хостов **рекомендуется** отбрасывать без уведомления дейтаграммы, полученные в широковещательных кадрах канального уровня (см. 2.4), если в них не указан широковещательный или групповой IP-адрес получателя.

Для рассылки широковещательных сообщений в подключенные сети **рекомендуется** использовать адреса формата Limited Broadcast.

Обсуждение

Использование формата Limited Broadcast взамен Directed Broadcast может повысить устойчивость системы. Проблемы часто бывают вызваны машинами, которые не понимают до конца природы широковещательных адресов (см. 3.2.1.3) или используют собственные идеи о применении таких адресов. Типичным примером из прошлого являются машины, которые не понимают выделение подсетей, но подключены к сети, содержащей последние. Передача сообщений с адресом Subnet Broadcast для подключенной сети будет приводить к тому, что такие машины воспримут эти сообщения как адресованные другому хосту (не им).

Существует также вопрос о возможности передачи дейтаграмм с адресом формата Limited Broadcast через все интерфейсы многодомного хоста, однако этот вопрос выходит за пределы документа.

3.3.7 IP Multicasting

Хост **должен** поддерживать локальное использование групповой адресации для всех подключенных сетей, на которых возможно отображение IP-адресов класса D на адреса канального уровня (см. ниже). Локальная поддержка групповой адресации включает передачу multicast-дейтаграмм, присоединение к multicast-группам, прием multicast-дейтаграмм и выход из multicast-групп. Сюда включается поддержка всех расширений [IP:4], за исключением протокола IGMP, поддержка которого **не является обязательной**.

Обсуждение

Протокол IGMP обеспечивает шлюзы, поддерживающие маршрутизацию групповых адресов, информацией, требуемой для обеспечения групповой адресации IP через множество сетей. В настоящее время multicast-маршрутизация находится в стадии экспериментов и доступна не везде. Для хостов, не подключенных к сетям с multicast-шлюзами, или в тех случаях, когда не нужно принимать multicast-дейтаграммы из других сетей, протокол IGMP не используется, поэтому его реализация не является обязательной. Однако, другие расширения [IP:4] в настоящее время рекомендованы в целях обеспечения доступа на уровне IP с локальной групповой адресацией, как альтернатива локальной широковещательной адресации. Предполагается, что реализация IGMP тоже станет обязательной в будущем, когда multicast-маршрутизация получит более широкое распространение.

Если поддержка IGMP не реализована, для хостов **рекомендуется** сохранять принадлежность к группе all-hosts (все хосты) с адресом 224.0.0.1 при инициализации уровня IP и сохранять принадлежность к этой группе в течение всего периода активности уровня IP.

Обсуждение

Включение в группу all-hosts будет обеспечивать локальную поддержку групповой адресации (например, протокол обнаружения шлюзов) даже без поддержки IGMP.

Отображение IP-адресов класса D на локальные адреса в настоящее время стандартизовано для следующих типов сетей:

- Ethernet/IEEE 802.3 в соответствии с [IP:4];
- всех сетей, поддерживающих широковещательную, но не групповую адресацию (IP-адреса класса D отображаются на локальный широковещательный адрес);
- всех типов каналов «точка-точка» (например, SLIP или HDLC) - в этом случае отображения адресов не требуется. Все групповые дейтаграммы IP передаются в исходном виде, включая локальное кадрирование.

Отображение для сетей других типов будет стандартизовано в будущем[1].

Хостам **рекомендуется** обеспечивать для протоколов верхних уровней или приложений способ определения поддержки групповой адресации IP хостами подключенных сетей.

3.3.8 Сообщения об ошибках

На практике хосты **должны** возвращать сообщения ICMP об ошибках при обнаружении последних, за исключением тех случаев, когда возврат сообщений ICMP об ошибках специально запрещен.

Обсуждение

Общим явлением в сетях на базе дейтаграмм является «болезнь черных дыр» - дейтаграммы передаются адресату, а от того не приходит никаких откликов. При отсутствии откликов обнаружение проблем становится весьма затруднительным.

3.4 Интерфейс между IP и транспортным уровнем

Интерфейс между IP и транспортным уровнем **должен** обеспечивать полный доступ ко всем механизмам уровня IP, включая опции, тип обслуживания TOS и время жизни (TTL). Транспортный уровень **должен** иметь механизм установки интерфейсных параметров или/и обеспечивать передачу таких параметров от приложений.

Обсуждение

[1]См. RFC 1469 (Token Ring), RFC 2022 (ATM). *Прим. перев.*

Для приложений очень важна возможность использования таких параметров, даже если соответствующие механизмы еще недостаточно эффективны в Internet (например, TOS). Реализация поддержки этих механизмов обеспечивает возможность работы с ними без серьезного изменения программных настроек хоста по мере реализации механизмов в Internet.

Опишем концептуальный интерфейс между транспортным уровнем и IP, как набор процедурных вызовов. Приведенное здесь описание является расширением RFC 791 [IP:1] (параграф 3.3).

- Передача дейтаграмм

  `SEND(src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt => result)`
  параметры процедуры описаны в RFC 791. Передача параметра Id необязательна (см. 3.2.1.5).

- Прием дейтаграмм

  `RECV(BufPTR, prot => result, src, dst, SpecDest, TOS, len, opt)`
  Все параметры определены в RFC 791, за исключением SpecDest = указанный адрес получателя дейтаграммы (см. 3.2.1.3)

  Полученный в результате параметр dst содержит адрес получателя дейтаграммы. Поскольку этот адрес может быть широковещательным, **должен** передаваться параметр SpecDest, не определенный в RFC 791. Параметр opt содержит все опции IP для дейтаграммы и также **должен** передаваться на транспортный уровень.

- Выбор адреса отправителя

  `GET_SRCADDR(remote, TOS)  -> local remote = remote IP address`
  TOS = тип обслуживания (Type-of-Service);

  local = локальный адрес IP.

  См. параграф 3.3.4.3.

- Определение максимального размера дейтаграмм

  `GET_MAXSIZES(local, remote, TOS) -> MMS_R, MMS_S`
  MMS_R = максимальный размер принимаемого сообщения транспортного уровня;

  MMS_S = максимальный размер передаваемого сообщения транспортного уровня;

  (local, remote, TOS определены выше).

  См. параграфы 3.3.2 и 3.3.3.

- Сообщение об успешной доставке

  `ADVISE_DELIVPROB(sense, local, remote, TOS)`
  Параметр sense представляет собой 1-битовый флаг, показывающий тип анонса (позитивный или негативный), описанного в параграфе 3.3.1.4. Остальные параметры были определены выше.

- Передача сообщения ICMP

  `SEND_ICMP(src, dst, TOS, TTL, BufPTR, len, Id, DF, opt) -> result`
  (Параметры определены в RFC 791).

  Передача параметра Id не является обязательной (см. параграф 3.2.1.5). Транспортный уровень **должен** быть способен передавать некоторые сообщения ICMP Port Unreachable или любой из запросов. Эта функция может рассматриваться как частный случай вызова SEND() и отдельное рассмотрение ее диктуется лишь задачами упрощения.

- Прием сообщения ICMP

  `RECV_ICMP(BufPTR ) -> result, src, dst, len, opt`
  (Параметры определены в RFC 791).

  Уровень IP **должен** передавать некоторые сообщения ICMP соответствующим программам транспортного уровня. Эта функция может рассматриваться как частный случай вызова RECV() и указана отдельно лишь для упрощения.

Для сообщений ICMP об ошибках, передаваемых на верхние уровни, данные **должны** включать исходный заголовок IP и все октеты исходной дейтаграммы, включенные в сообщение ICMP. Эти данные будут использоваться транспортным уровнем для поиска информации о состоянии соединения.

На верхний уровень передаются, в частности, следующие сообщения:

- Destination Unreachable;
- Source Quench;
- Echo Reply (пользовательскому интерфейсу ICMP, если Echo Request передан не с уровня IP);
- Timestamp Reply (пользовательскому интерфейсу ICMP);
- Time Exceeded.

Обсуждение

В будущем интерфейс обмена данными между транспортным уровнем и IP может быть расширен (см. параграф 3.3.1.3) для передачи информации о пути.

3.5 Требования к уровню INTERNET

Функция	*Параграф*	*Требование*
Реализация IP и ICMP	3.1	Обязательно
Обработка remote multihoming на прикладном уровне	3.1	Обязательно
Поддержка local multihoming	3.1	Возможно
Выполнение требований к шлюзам при рассылке дейтаграмм	3.1	Обязательно
Настройка конфигурации для встроенного маршрутизатора	3.1	Обязательно[1]
Режим маршрутизатора по умолчанию выключен[1]	3.1	Обязательно
Автоматическая настройка по числу интерфейсов[1]	3.1	Недопустимо
Возможность протоколирования отброшенных дейтаграмм	3.1	Следует
Корректировка счетчиков статистики при отбрасывании	3.1	Следует
Отбрасывание без уведомления пакетов других (не 4) версий	3.2.1.1	Обязательно
Проверка контрольной сумм и отбрасывание без уведомления при ошибках	3.2.1.2	Обязательно
Адресация:		
Адресация подсетей (RFC 950)	3.2.1.3	Обязательно
Адресом отправителя должен быть собственный адрес хоста	3.2.1.3	Обязательно
Отбрасывание без уведомления дейтаграмм с некорректным адресом получателя	3.2.1.3	Обязательно
Отбрасывание без уведомления дейтаграмм с некорректным адресом отправителя	3.2.1.3	Обязательно
Поддержка сборки фрагментов	3.2.1.4	Обязательно
Сохранение идентификатора для копии дейтаграммы	3.2.1.5	Возможно
TOS:		
Возможность установки TOS на транспортном уровне	3.2.1.6	Обязательно
Передача принятых значений TOS на транспортный уровень	3.2.1.6	Следует
Использование на канальном уровне отображения RFC 795	3.2.1.6	Не следует
TTL:		
Передача пакетов с TTL=0	3.2.1.7	Не допускается
Отбрасывание принятых дейтаграмм с TTL < 2	3.2.1.7	Не допускается
Возможность установки TTL на транспортном уровне	3.2.1.7	Обязательно
Возможность установки фиксированного значения TTL	3.2.1.7	Обязательно
Опции IP:		
Возможность транспортного уровня передавать опции	3.2.1.8	Обязательно
Передача всех принятых опций на вышележащий уровень	3.2.1.8	Обязательно
Игнорирование неизвестных опций на уровне IP	3.2.1.8	Обязательно
Опции безопасности	3.2.1.8a	Возможно
Передача идентификатора потока	3.2.1.8b	Не следует
Игнорирование опции идентификатора потока	3.2.1.8c	Обязательно
Запись маршрутов	3.2.1.8d	Возможно
Временная метка	3.2.1.8e	Возможно
Опция Source Route:		
Инициирование и завершение опции Source Route	3.2.1.8c	Обязательно
Дейтаграммы с заполненным SR передаются на транспортный уровень	3.2.1.8c	Обязательно
Построение корректного (без избыточности) пути возврата	3.2.1.8c	Обязательно
Передача множества опции SR в заголовке	3.2.1.8c	Недопустимо
ICMP:		
Отбрасывание без уведомления неизвестных типов ICMP	3.2.2	Обязательно
Включение более 8 октетов исходной дейтаграммы	3.2.2	Возможно
Включение полученных октетов	3.2.2	Обязательно
Передача сообщений ICMP Error транспортному ровню	3.2.2	Обязательно
Передача сообщений ICMP с TOS=0	3.2.2	Следует
Передача сообщений ICMP об ошибках для:		
- сообщений ICMP об ошибках	3.2.2	Недопустимо
- широковещательных и групповых пакетов IP	3.2.2	Недопустимо
- широковещательных пакетов канального уровня	3.2.2	Недопустимо
- фрагментов, не являющихся первыми	3.2.2	Недопустимо
- дейтаграмм с неуникальным адресом отправителя	3.2.2	Недопустимо
Возврат сообщений ICMP об ошибках (если не запрещено)	3.3.8	Обязательно
Получатель недоступен (destination unreachable):		
Генерация destination unreachable (код 2 и 3)	3.2.2.1	Следует
Передача destination unreachable на верхние уровни	3.2.2.1	Обязательно
Действия верхних уровней для destination unreachable	3.2.2.1	Следует
Интерпретация dest. unreachable лишь как совета	3.2.2.1	Обязательно
Redirect:		
Хост посылает Redirect	3.2.2.2	Не следует
Обновление кэша маршрутов при получении Redirect	3.2.2.2	Обязательно
Обслуживание Redirect для хоста и сети	3.2.2.2	Обязательно
Отбрасывание некорректных сообщений Redirect	3.2.2.2	Следует
Source Quench:		
Передача Source Quench при нехватке буферов	3.2.2.3	Возможно
Передача Source Quench на верхний уовень	3.2.2.3	Обязательно
Воздейтсвие верхнего уровня на Source Quench	3.2.2.3	Следует
Передача Time Exceeded на верхний уровень	3.2.2.4	Обязательно
Проблемы с параметрами:		
Передача сообщений Parameter Problem	3.2.2.5	Следует
Передача Parameter Problem на верхний уровень	3.2.2.5	Обязательно
Передача сообщений Parameter Problem пользователю	3.2.2.5	Возможно

[1]Только при реализации этой опции

Функция	*Параграф*	*Требование*
ICMP Echo Request/Reply:		
Эхо-сервер и эхо-клиент	3.2.2.6	Обязательно
Эхо-клиент	3.2.2.6	Следует
Отбрасывание Echo Request для широковещательных адресов	3.2.2.6	Возможно
Отбрасывание Echo Request для групповых адресов	3.2.2.6	Возможно
Использование указанного адреса отправителя в Echo Reply	3.2.2.6	Обязательно
Сохранение данных в Echo Reply	3.2.2.6	Обязательно
Передача Echo Reply на верхний уровень	3.2.2.6	Обязательно
Отражение опций Record Route, Time Stamp	3.2.2.6	Следует
Инверсия и отражение опции Source Route	3.2.2.6	Обязательно
ICMP Information Request/Reply	3.2.2.7	Не следует
ICMP Timestamp/Timestamp Reply:		Возможно
Минимизация вариаций задержки	3.2.2.8	Следует[1]
Отбрасывание без уведомления широковещательных пакетов Timestamp	3.2.2.8	Возможно[1]
Отбрасывание без уведомления групповых Timestamp	3.2.2.8	Возможно[1]
Использование указанного адреса как отправителя TS Reply	3.2.2.8	Обязательно[1]
Отражение опций Record Route и Timestamp	3.2.2.8	Следует[1]
Обращение и отражение опции Source Route	3.2.2.8	Обязательно[1]
Передача на верхний уровень	3.2.2.8	Обязательно[1]
Выполнение правил для «стандартных значений»	3.2.2.8	Обязательно[1]
ICMP Address Mask Request/Reply:		
Настройка источника получения Address Mask	3.2.2.9	Обязательно
Поддержка статической конфигурации адресных масок	3.2.2.9	Обязательно
Динамическое получение маски при загрузке	3.2.2.9	Возможно
Получение маски с помощью Addr. Mask Request/Reply	3.2.2.9	Возможно
Повторная передача запроса при отсутствии отклика	3.2.2.9	Обязательно[2]
Использование маски по умолчанию при отсутствии отклика	3.2.2.9	Следует2
Обновление маски после получения первого отклика	3.2.2.9	Обязательно[2]
Разумная проверка маски	3.2.2.9	Следует
Неуполномоченная передача откликов Address Mask Reply	3.2.2.9	Недопустимо
Явное указание уполномоченных агентов	3.2.2.9	Обязательно
Статическая конфигурация => флаг Addr-Mask-Authoritative	3.2.2.9	Следует
Широковещательная передача Addr. Mask Reply при инициализации	3.2.2.9	Обязательно[2]
Маршрутизация исходящих дейтаграмм:		
Использование маски при выборе локальный/удаленный	3.3.1.1	Обязательно
Работа с подключенной сетью без шлюза	3.3.1.1	Обязательно
Поддержка кэша для ближайших (next-hop) шлюзов	3.3.1.2	Обязательно
Одинаковая трактовка Host/Net Redirect	3.3.1.2	Следует
Использование шлюза по умолчанию при отсутствии записи в кэше	3.3.1.2	Обязательно
Поддержка множества “шлюзов по умолчанию”	3.3.1.2	Обязательно
Поддержка таблицы статических маршрутов	3.3.1.2	Возможно
Флаг: возможность переписывания маршрута путем Redirect	3.3.1.2	Возможно
Использование в качестве ключа адреса хоста, а не сети	3.3.1.3	Возможно
Включение TOS в кэш маршрутов	3.3.1.3	Следует
Возможность обнаружения сбоев в следующем шлюзе	3.3.1.4	Обязательно
Предположение о постоянной доступности маршрута	3.3.1.4	Не следует
Непрерывное использование ping для шлюзов	3.3.1.4	Недопустимо
ping используется только при передаче трафика	3.3.1.4	Обязательно
ping используется только при отсутствии позитивных анонсов	3.3.1.4	Обязательно
Анонсы от верхних и нижних уровней	3.3.1.4	Следует
Смена “умершего” шлюза по умолчанию	3.3.1.5	Обязательно
Возможность ввода конфигурационных параметров вручную	3.3.1.6	Обязательно

[1]Только при реализации опции
[2]Если опция реализована и включена.

Функция	*Параграф*	*Требование*
Фрагментация и сборка		
Возможность сборки входящих дейтаграмм	3.3.2	Обязательно
По крайней мере дейтаграммы размером 576 байтов	3.3.2	Обязательно
Настраиваемое или неограниченное значение EMTU_R	3.3.2	Следует
Возможность транспортного уровня выяснять MMS_R	3.3.2	Обязательно
Передача ICMP Time Exceed при тайм-ауте во время сборки	3.3.2	Обязательно
Фиксированный тайм-аут для сборки	3.3.2	Следует
Передача MSS_S на верхние уровни	3.3.3	Обязательно
Локальная фрагментация исходящих пакетов	3.3.3	Возможно
или отказ от передачи пакетов > MSS_S	3.3.3	Обязательно
Передача не более 576 байтов за пределы сети	3.3.3	Следует
Конфигурационный флаг All-Subnets-MTU	3.3.3	Возможно
Многодомные хосты:		
Ответ в таким же адресом, как указанный адрес получателя	3.3.4.2	Следует
Возможность для приложений выбирать локальные адреса IP	3.3.4.2	Обязательно
Отбрасывание дейтаграмм на “некорректном” интерфейсе	3.3.4.2	Возможно
Передача дейтаграмм только через “правильный” интерфейс	3.3.4.2	Возможно[1]
Пересылка SOURCE-ROUTE:		
Пересылка дейтаграмм с опцией Source Route	3.3.5	Возможно[1]
Выполнение соответствующих правил для шлюзов	3.3.5	Обязательно[1]
Обновление TTL с помощью правил шлюза	3.3.5	Обязательно[1]
Возможность генерации кодов ошибок ICMP 4, 5	3.3.5	Обязательно[1]
IP-адрес отправителя не указывает на локальный хост	3.3.5	Возможно[1]
Обновление опций Timestamp, Record Route	3.3.5	Обязательно[1]
Настраиваемый переключатель для нелокальных SRing	3.3.5	Обязательно[1]
По умолчанию отключено	3.3.5	Обязательно[1]
Выполнение правил доступа к шлюзу для нелокальных SRing	3.3.5	Обязательно[1]
Если не пересылаем, использовать опцию Dest Unreach (5)	3.3.5	Следует[2]
Широковещание:		
Широковещательный IP-адрес отправителя	3.2.1.3	Недопустимо
Нормальный прием широковещательных дейтаграмм форм. 0 и -1	3.3.6	Следует
Опция передачи широковещ. дейтаграмм форм. 0 и -1	3.3.6	Возможно
По умолчанию формат -1	3.3.6	Следует
Распознавание всех форматов широковещательных адресов	3.3.6	Обязательно
Использ. групповых/широковещательных адресов IP в широковещ. канального уровня	3.3.6	Обязательно
Отбрасывание дейтаграмм только с широковещательным адресом канального уровня	3.3.6	Следует
Использование адресов Limited Broadcasct для подключенных сетей	3.3.6	Следует
Групповая адресация:		
Поддержка локальной групповой адресации IP (RFC 1112)	3.3.7	Следует
Поддержка IGMP (RFC 1112)	3.3.7	Возможно
Присоединение группы all-hosts при старте	3.3.7	Следует
Поддержка интерфейса с верхним уровнем	3.3.7	Следует
Интерфейс:		
Возможность использовать все механизмы IP на транспортном уровне	3.4	Обязательно
Передача идентификатора интерфейса на транспортный уровень	3.4	Обязательно
Передача всех опций IP на транспортный уровень	3.4	Обязательно
Транспортный уровень может передав. некоторые сообщения ICMP	3.4	Обязательно
Передача заданных сообщений ICMP на транспортный уровень	3.4	Обязательно
Включение заголовка IP + 8 или более октетов	3.4	Обязательно

4. Транспортные протоколы

4.1 Протокол пользовательских дейтаграмм UDP

4.1.1 Введение

Протокол пользовательских дейтаграмм UDP[3] [UDP:1] обеспечивает лишь минимальный транспортный сервис - негарантированную доставку дейтаграмм - и предоставляет приложениям прямой доступ к службе дейтаграмм уровня IP. UDP используется приложениями, которым не нужен высокий уровень сервиса TCP, или требуемые приложению коммуникационные службы не поддерживаются протоколом TCP (например, групповая или широковещательная адресация).

Протокол UDP не добавляет почти ничего к службам уровня IP - лишь поддержку контрольных сумм и мультиплексирование по номерам портов. Следовательно, прикладные программы, использующие UDP должны сами решать все задачи сквозного обмена данными, которые обычно обслуживаются протоколами, основанными на соединениях (т. е., обеспечивать повторную передачу при неудачах, управлять потоком данных, контролировать насыщение и т. п). Сложность взаимодействия между IP и TCP заменяется на сложное взаимодействие между UDP и приложениями, использующими протокол UDP.

4.1.2 Общие вопросы

В спецификации UDP ошибки отсутствуют.

[1]Если не используются встроенные функции маршрутизатора или source routed.
[2]Это требование изменяется для дейтаграмм, содержащих сообщений ICMP об ошибках.
[3]User Datagram Protocol.

4.1.3 Частные вопросы

4.1.3.1 Порты

Общеизвестные порты UDP подчиняются тем же правилам, что и общеизвестные порты для протокола TCP (см. параграф 4.2.2.1).

Если принятая дейтаграмма адресована порту UDP, для которого нет прослушивания, протоколу UDP **следует** передать отправителю сообщение ICMP Port Unreachable.

4.1.3.2 Опции IP

Протокол UDP **должен** передавать прикладным программам все опции IP, полученные от уровня IP.

Приложениям **должна** предоставляться возможность установки опций IP для передаваемых дейтаграмм UDP и протокол UDP **должен** передавать эти опции уровню IP.

Обсуждение

В настоящее время через UDP передаются только опции Source Route, Record Route и Time Stamp. Однако в будущем число таких опций может быть расширено (например, за счет опций безопасности IP) и реализации протокола UDP, поэтому, не должны делать каких-либо предположений о формате или содержимом передаваемых опций.

Приложениям на основе UDP может потребоваться получение информации source route из дейтаграмм с запросами и установка обратного маршрута в передаваемых откликах.

4.1.3.3 Сообщения ICMP

Протокол UDP **должен** передавать на уровень приложений все сообщения ICMP об ошибках, полученные от уровня IP. Для реализации этого может использоваться вызов процедуры ERROR_REPORT (см. 4.2.4.1).

Обсуждение

Отметим, что сообщения ICMP об ошибках в результате передачи дейтаграмм UDP принимаются асинхронно. Приложения на базе протокола UDP, желающие получать сообщения ICMP об ошибках, сами отвечают за поддержку состояния, которое обеспечит демультиплексирование принятых сообщений (например, программа может сохранять операцию приема незавершенной). Приложения также отвечают за предотвращение конфликтов в результате задержки сообщений ICMP об ошибках по причине занятости портов.

4.1.3.4 Контрольные суммы UDP

Хост **должен** поддерживать механизмы генерации и проверки контрольных сумм UDP. Приложения **могут** управлять процессом генерации контрольных сумм UDP, но по умолчанию протокол **должен** самостоятельно работать с контрольными суммами.

Если принятая дейтаграмма UDP содержит отличную от нуля и некорректную контрольную сумму, протокол UDP **должен** отбросить такую дейтаграмму без уведомления. Приложения **могут** управлять решением вопроса об отбрасывании дейтаграмм без контрольной суммы.

Обсуждение

Некоторые приложения, применяемые преимущественно в локальных сетях, могут отключать использование контрольных сумм UDP в целях повышения производительности. В результате этого природа множества ошибок, которые могут возникать, не будет определена. Эффективность работы с отключенным механизмом контрольных сумм UDP весьма спорна.

Реализация

В реализации контрольных сумм UDP часто допускают одну и ту же ошибку. В отличие от контрольных сумм TCP контрольные суммы UDP не являются обязательными - нулевое значение поля контрольной суммы в заголовке UDP говорит об отсутствии контрольной суммы. Если отправитель получает для контрольной суммы реальной дейтаграммы UDP нулевое значение, он должен установить в поле контрольной суммы значение 65535 (все биты имеют значение 1). От получателя не требуется в таких случаях каких-либо дополнительных действий, поскольку значения 0 и 65535 эквивалентны с точки зрения арифметики с дополнением до 1.

4.1.3.5 Многодомные хосты UDP

При получении дейтаграммы UDP указанный адрес получателя **должен** передаваться на уровень приложений. Прикладные программы **должны** иметь возможность указывать IP-адрес отправителя при передаче дейтаграмм UDP или оставлять поле адреса пустым (в этих случаях сетевые программы должны указать подходящий адрес отправителя). **Рекомендуется** обеспечивать способ передачи выбранного адреса отправителя на прикладной уровень, чтобы программа могла позднее получать отклик на дейтаграмму только от соответствующего интерфейса.

Обсуждение

Приложения на основе запросов/откликов, использующие протокол UDP, должны устанавливать для откликов в качестве адреса отправителя тот же адрес, который был задан для получателя запроса (см. параграф «Общие вопросы» в [INTRO:1]).

4.1.3.6 Некорректная адресация

Дейтаграммы UDP, принятые с некорректным IP-адресом отправителя (например, широковещательный или групповой адрес) **должны** отбрасываться на уровне UDP или IP (см. 3.2.1.3).

Когда хост передает дейтаграмму UDP, в качестве адреса отправителя **должен** использоваться IP-адрес (один из возможных) этого хоста.

4.1.4 Интерфейс между уровнем UDP и прикладным уровнем

Интерфейс между приложениями и UDP **должен** полностью обеспечивать службы, описанные в параграфе 3.4. Таким образом, приложениям на основе UDP требуются функции GET_SRCADDR(), GET_MAXSIZES(), ADVISE_DELIVPROB() и RECV_ICMP(), описанные в 3.4. Например, функция GET_MAXSIZES() может использоваться для определения эффективного значения максимального размера дейтаграмм UDP для конкретной триады {интерфейс, удаленный хост, TOS}.

Программы прикладного уровня **должны** иметь возможность установки значений TTL и TOS, а также опций IP при передаче дейтаграмм UDP и установленные значения **должны** прозрачно передаваться на уровень IP. UDP **может** передавать полученные значения TOS на уровень приложений.

4.1.5 Требования к протоколу UDP

Функция	*Параграф*	*Требование*
UDP передает сообщения Port Unreachable	4.1.3.1	Следует
Опции IP в UDP		
Передача полученных опций на уровень приложений	4.1.3.2	Обязательно
Приложения могут устанавливать опции при передаче	4.1.3.2	Обязательно
UDP передает опции на уровень IP	4.1.3.2	Обязательно
Передача сообщений ICMP на прикладной уровень	4.1.3.3	Обязательно
Контрольные суммы UDP:		
Генерация и проверка контрольных сумм	4.1.3.4	Обязательно
Отбрасывание дейтаграмм с ошибкой в контрольной сумме	4.1.3.4	Обязательно
Передача дейтаграмм без контрольной суммы	4.1.3.4	Возможно
По умолчанию контрольная сумма используется	4.1.3.4	Обязательно
Получатель может требовать контрольную сумму	4.1.3.4	Возможно
Многодомные хосты UDP:		
Передача указанного адреса получателя приложениям	4.1.3.5	Обязательно
Возможность задания локального адреса отправителя на прикладном уровне	4.1.3.5	Обязательно
Возможность задания шаблона локального адреса отправителя на прикладном уровне	4.1.3.5	Обязательно
Уведомление приклад. уровня об используемом локальном адресе	4.1.3.5	Возможно
Дейтаграммы с некорректным IP-адресом отправителя отбрасываются UDP/IP	4.1.3.6	Обязательно
При передаче дейтаграмм используется только корректный. адрес IP	4.1.3.6	Обязательно
Службы интерфейса UDP с приложениями:		
Полный интерфейс IP (см. 3.4) для приложений	4.1.4	Обязательно
Возможность задания TTL, TOS и опций IP при передаче	4.1.4	Обязательно
Передача принятого TOS на уровень приложений	4.1.4	Возможно

4.2 Протокол управления передачей - TCP

4.2.1 Введение

Протокол управления передачей - TCP[1] [TCP:1] представляет собой транспортный протокол стека Internet для работы с виртуальными соединениями. TCP обеспечивает гарантированную доставку с сохранением порядка для полнодуплексных потоков данных (октеты или байты). Протокол TCP используется теми приложениями, которым нужен ориентированный на соединения транспортный сервис с гарантией доставки (например, электронная почта SMTP, передача файлов по протоколу FTP, служба виртуальных терминалов Telnet); требования к таким протоколам прикладного уровня описаны в работе [INTRO:1].

4.2.2 Общие вопросы

4.2.2.1 Общеизвестные порты: RFC 793, параграф 2.7

Обсуждение

TCP резервирует номера от 0 до 255 для общеизвестных портов, которые служат для использования стандартных служб через Internet. Остальные номера портов могут свободно распределяться между прикладными процессами. Текущий список общеизвестных портов можно найти в документе Assigned Numbers [INTRO:6]. Предпосылкой задания новых общеизвестный номеров является подготовка для новой службы RFC, достаточно детально описывающего сервис для обеспечения возможности его реализации.

Некоторые системы выделяют еще одну область портов TCP - зарезервированные порты, которые обычно используются для системных задач. Например, зарезервированные порты могут занимать номера от 256 до некоторого значения, принятого в данной системе. Некоторые системы используют защиту общеизвестных и зарезервированных портов, позволяя лишь привилегированным пользователям открывать соединения TCP с использованием этих портов. Такая мера весьма разумна, если хост не делает предположений, что другие хосты используют порты с младшими номерами портов аналогичным образом.

4.2.2.2 Использование флага Push: RFC 793, параграф 2.8

Когда приложение использует серию вызовов SEND без установки флага PUSH, TCP **может** объединять (агрегировать) данные внутренними средствами, не передавая их. Подобно этому, при получении серии сегментов без бита PSH, TCP **может** помещать данные во внутреннюю очередь без передачи их принимающему приложению.

Бит PSH не является маркером записи и не связан с границами сегментов. Отправителю **рекомендуется** удалять последовательные биты PSH при пакетировании данных для передачи сегментов максимального размера.

TCP может реализовать флаги PUSH при вызовах функции SEND. Если флаги PUSH не реализованы, требуется выполнение двух условий при передаче TCP: (1) буфер данных не должен быть бесконечным и (2) **должен** устанавливаться бит PSH в последнем буферизованном сегменте (т. е., при отсутствии в очереди данных для передачи).

[1]Transmission Control Protocol.

В RFC 793 на страницах 48, 50 и 74 ошибочно указано, что принятый бит PSH должен передаваться прикладному уровню - передача прикладному уровню полученного флага PSH не является обязательной.

От прикладных программ требуется установка флага PUSH при вызове SEND, если требуется форсировать доставку во избежание простоя соединения. Однако протоколу TCP **рекомендуется** по возможности передавать сегменты максимального размера в целях повышения производительности (см. 4.2.3.4).

Обсуждение

Когда флаг PUSH не реализован для вызовов функции SEND (т. е., интерфейс между прикладным уровнем и TCP использует потоковую модель в чистом виде), ответственность за объединение небольших фрагментов данных в сегменты разумного размера частично ложится на уровень приложений.

В общем случае интерактивный прикладной протокол должен устанавливать флаг PUSH по крайней мере при последнем вызове SEND для каждого набора команд или откликов. Протоколы, передающие большие объемы данных (типа FTP), должны устанавливать флаг PUSH для последнего сегмента файла или при возникновении необходимости предотвратить «застой» буфера.

На приемной стороне бит PSH форсирует доставку буферизованных данных прикладной программе (даже при неполном использовании буфера). Отсутствие бита PSH может использоваться для предотвращения ненужных обращений к прикладным процессам, что может существенно повысить производительность (особенно важно это для больших хостов с разделением времени). Передача бита PSH принимающей программе позволяет провести аналогичную оптимизацию на прикладном уровне.

4.2.2.3 Размер окна: RFC 793, параграф 3.1

Размер окна **должен** трактоваться как беззнаковое целое - если большое окно будет представляться, как окно с отрицательным размером, TCP просто не будет работать. Рекомендуется использовать 32-битовые поля для размера окна передачи и приема в записи с параметрами соединения.

Обсуждение

Известно, что поле размера окна в заголовке TCP слишком мало для высоких скоростей и путей с большой задержкой. Для расширения окна TCP определены экспериментальные опции (см. например, [TCP:11]). С учетом такого расширения при реализации TCP следует рассчитывать на 32-битовые значения размера окна.

4.2.2.4 Указатель срочности: RFC 793, параграф 3.1

Второе предложение этого параграфа содержит ошибку - urgent pointer указывает на порядковый номер **последнего** (а не следующего за ним) октета в последовательности срочных данных. Описание на стр. 56 (последнее предложение) корректно.

Протокол TCP **должен** поддерживать последовательности срочных данных любой длины.

Протокол TCP **должен** информировать прикладной уровень в асинхронном режиме всякий раз, когда указатель Urgent получен при отсутствии ожидающих обработки срочных данных или Urgent упреждает срочные данные в потоке. Для приложений **должен** обеспечиваться способ определения количества остающихся срочных данных, которые будут прочитаны из соединения, или, по крайней мере, возможность узнать о наличии таких данных.

Обсуждение

Хотя механизм Urgent может использоваться для любых приложений, обычно он применяется для передачи «прерываний» типа команд Telnet (см. параграф Using Telnet Synch Sequence в [INTRO:1]).

Асинхронное уведомление по обдельному каналу (out-of-band) будет позволять приложениям перейти в режим urgent при чтении данных из соединения TCP. Это позволяет контролировать команды, передаваемые приложению, нормальный буфер которого заполнен необработанными данными.

Реализация

Базовый механизм ERROR-REPORT(), описанный в параграфе 4.2.4.1, может использоваться для информирования приложений о приходе важных данных.

4.2.2.5 Опции TCP: RFC 793, параграф 3.1

Протокол TCP **должен** обеспечивать возможность получения опций TCP в любом сегменте. Протокол TCP **должен** игнорировать без генерации ошибки любые опции TCP, которые в нем не реализованы, предполагая, что опции содержат поле размера (все определяемые в будущем опции TCP будут также содержать поле размера). Протокол TCP **должен** быть готов к обработке опций некорректного (например, нулевого) размера без возникновения серьезных проблем - одним из вариантов такой обработки является сброс соединения и протоколирование причины.

4.2.2.6 Максимальный размер сегмента: RFC 793, параграф 3.1

Протокол TCP **должен** поддерживать опции максимального размера сегмента - MSS[1] для приема и передачи [TCP:4].

Для TCP **рекомендуется** передавать опцию MSS в каждом сегменте SYN при получении MSS, отличного от принятого по умолчанию значения 536; **можно** передавать эту опцию во всех случаях.

Если опция MSS не получена при организации соединения, протокол TCP **должен** использовать принятое по умолчанию значение MSS = 536 (576-40) [TCP:4].

Максимальный размер сегмента, реально передаваемого TCP - эффективное значение MSS для передачи, - **должен** быть меньше MSS для передачи (отражает доступный размер буфера сборки на удаленном хосте) и максимального размера, поддерживаемого уровнем IP:

```
Eff.snd.MSS = min(SendMSS+20, MMS_S) - TCPhdrsize - Ipoptionsize,
```

где:

[1]Maximum Segment Size

- SendMSS - значение MSS, полученное от удаленного хоста или принятое по умолчанию значение 536, если опция MSS не была получена;
- MMS_S - максимальный размер сообщений транспортного уровня, которые может передавать TCP;
- TCPhdrsize - размер заголовка TCP (обычно 20, но может быть больше за счет опций TCP);
- IPoptionsize - размер всех опций IP, которые TCP будет передавать на уровень IP с этим сообщением.

Значение MSS, которое будет передано в опции MSS, не должно быть больше

```
MMS_R - 20
```

где MMS_R - максимальный размер для сообщений транспортного уровня, которые могут быть приняты (и собраны); TCP получает значения MMS_R и MMS_S от уровня IP (см. описание функции GET_MAXSIZES в 3.4).

Обсуждение

Размер сегмента TCP оказывает существенное влияние на производительность. Увеличение сегментов повышает пропускную способность за счет снижения объема заголовков в общем потоке данных и уменьшения времени на обработку этих заголовков. Однако, если пакеты столь велики, что требуется фрагментация IP, эффективность существенно снижается при потере любого фрагмента [IP:9].

Некоторые реализации TCP передают опцию MSS только в тех случаях, когда хост-получатель относится к подключенной сети. Однако в общем случае уровень TCP может не иметь достаточно информации для определения такой принадлежности, поэтому разумно передать уровню IP решение вопроса определения приемлемого значения MTU для пути Internet. Рекомендуется использовать опцию MSS во всех случаях, когда значение отличается от 536 (тогда уровень IP определяет MMS_R в соответствии с параграфами 3.3.3 и 3.4). Предлагаемые для уровня IP механизмы определения MTU в этом случае не будут оказывать влияния на TCP.

4.2.2.7 Контрольная сумма TCP: RFC 793, параграф 3.1

В отличие от контрольных сумм UDP (см. 4.1.3.4), контрольные суммы TCP использовать обязательно. Отправитель **должен** генерировать контрольную сумму, а получатель **должен** ее проверять.

4.2.2.8 Диаграмма состояния соединения TCP: RFC 793 параграф 3.2, стр. 23

С диаграммами состояния связано несколько проблем:

(a) Стрелка от SYN-SENT к SYN-RCVD должна помечаться snd SYN,ACK, в соответствии с текстом на стр. 68 и рисунком 8.

(b) Возможна стрелка от состояния SYN-RCVD к состоянию LISTEN при условии получения RST после пассивного открытия (см. стр 70 в RFC 793).

(c) Возможно прямо перейти из FIN-WAIT-1 в состояние TIME-WAIT (см. стр. 75 в RFC 793).

4.2.2.9 Выбор начального порядкового номера: RFC 793, параграф 3.3, стр. 27

Протокол TCP **должен** использовать начальный порядковый номер, генерируемый в соответствии с текущим временем.

4.2.2.10 Число последовательных попыток открытия: RFC 793, параграф 3.4, стр. 32

На рисунке 8 допущена ошибка - пакет в строке 7 должен быть идентичен пакету в строке 5.

Протокол TCP **должен** поддерживать одновременные попытки организации соединения.

Обсуждение

Разработчики иногда удивляются, увидев, что при попытке двух приложений одновременно соединиться друг с другом организуется только одно соединение. Это не ошибка, а предусмотренное поведение, поэтому не пытайтесь «исправить» ситуацию.

4.2.2.11 Восстановление по старым дубликатам SYN: RFC 793, параграф 3.4, стр. 33

Отметим, что реализации TCP **должны** сохранять информацию о соединениях, достигших состояния SYN_RCVD в результате пассивного или активного использования OPEN.

4.2.2.12 Сегмент RST: RFC 793, параграф 3.4

Для протокола TCP **следует** допускать прием сегментов RST, содержащих данные.

Обсуждение

Предлагается включать в сегменты RST текст в формате ASCII, содержащий код и объяснение причины RST. Стандарта для представления таких данных еще не разработано.

4.2.2.13 Завершение соединений: RFC 793, параграф 3.5

Соединения TCP могут завершаться двумя способами: (1) нормальная процедура завершения TCP с использованием FIN и (2) разрыв в результате передачи одного или нескольких сегментов RST, приводящих к немедленному закрытию соединения. Если соединение TCP закрыто удаленной стороной, локальное приложение **должно** получить информацию об использованном варианте завершения (1 или 2).

При нормальном завершении соединений TCP обеспечивается гарантированная доставка данных в обоих направлениях. Поскольку оба направления соединений TCP закрываются независимо, возможно существование «полузакрытых» соединений, когда передача данных в одном направлении завершена (и невозможна), а в другом данные продолжают передаваться.

Хост может реализовать «полудуплексные» последовательности закрытия TCP так, что приложение, вызвавшее функцию CLOSE, не сможет после этого читать данные из соединения. Если такой хост вызывает CLOSE в процессе

приема данных через соединение TCP или новые данные получены уже после вызова CLOSE, протоколу TCP **следует** передать сегмент RST для информирования о потере данных.

При активном закрытии соединения оно **должно** сохраняться в состоянии TIME-WAIT (ожидание) в течение времени 2 x MSL[1]. Однако **возможно восприятие** новых вызовов SYN от удаленного TCP для повторного открытия соединения непосредственно из состояния TIME-WAIT, если выполняются следующие условия:

(1) начальный порядковый номер для нового соединения больше максимального порядкового номера, использованного в предыдущем соединении;

(2) происходит возврат в состояние TIME-WAIT, если SYN оказывается дубликатом старого вызова.

Обсуждение

Полнодуплексное завершение соединений TCP для сохранения данных не включено в аналогичный транспортный протокол TP4 стека ISO.

Некоторые системы не поддерживают полузакрытых соединений, поскольку такие соединения не согласуются с моделью ввода-вывода используемой операционной системы. В таких системах приложение после вызова CLOSE уже не сможет читать данные из соединения - такая ситуация называется полудуплексным завершением сеанса TCP.

Изящный алгоритм завершения TCP требует, чтобы активное состояние сохранялось (по крайней мере) на одной из сторон в течение времени 2 x MSL (4 минуты). В течение этого периода пара (удаленный сокет, локальный сокет), определяющая соединение, остается занятой и не может быть повторно использована. Для сокращения периода занятости данной пары портов некоторые реализации TCP позволяют воспринимать новые вызовы SYN в состоянии TIME-WAIT.

4.2.2.14 Передача данных: RFC 793, параграф 3.7, стр. 40

С момента выхода RFC 793 алгоритмы TCP постоянно совершенствовались для повышения эффективности обмена данными. В последующих параграфах этого документа описаны обязательные и рекомендуемые алгоритмы TCP, позволяющие определить, когда следует передавать данные (4.2.3.4) и подтверждения (4.2.3.2) или обновлять окно (4.2.3.3).

Обсуждение

Одним из важных вопросов производительности является «синдром глупого окна» (SWS[2]), описанный в [TCP:5] - стабильное небольшое увеличение окна, в результате которого происходит существенное снижение производительности TCP. Алгоритмы предотвращения SWS описаны ниже как для передающей (4.2.3.4), так и для приемной стороны (4.2.3.3).

Кратко говоря, причиной SWS является получение информации о расширении окна вправо всякий раз, когда есть буферное пространство, доступное для приема данных, и отправитель использует увеличение окна (невзирая на его незначительность) для передачи большего объема данных [TCP:5]. Результатом этого может стать непрерывная передача крошечных сегментов даже при наличии больших буферов на приемной и передающей стороне. SWS может возникать только при передаче больших объемов данных; если соединение стабильно (передается постоянный поток данных), проблема исчезает. Причиной проблемы является типичная реализация прямого управления окном - ниже описаны алгоритмы для получателя и отправителя, позволяющие решить эту проблему.

Другим важным вопросом производительности TCP является то, что некоторые приложения (особенно системы удаленного входа) на хостах с посимвольным вводом имеют тенденцию передавать потоки однооктетных сегментов данных. Во избежание застоя каждый вызов TCP SEND от таких приложений должен выталкиваться приложением явно или в неявной форме с помощью TCP. В результате может возникнуть поток сегментов TCP, содержащих лишь по одному октету, что ведет к снижению эффективности использования Internet и вносит существенный вклад в насыщение. Алгоритм Nagle, описанный в параграфе 4.2.3.4, обеспечивает простое и эффективное решение этой проблемы. Алгоритм обеспечивает группировку символов для соединений Telnet (это может удивить пользователей, ожидающих эхо после каждого символа, но восприятие пользователей не является проблемой).

Отметим, что алгоритм Nagle и алгоритм предотвращения SWS дополняют друг друга в деле повышения производительности. Алгоритм Nagle предотвращает передачу крошечных сегментов, которая может приводить к незначительному постоянному расширению окна, а алгоритм предотвращения SWS блокирует появление мелких сегментов в результате незначительных сдвигов правого края окна в сторону расширения.

В осторожных реализациях может передаваться два или более подтверждений для каждого принятого сегмента. Предположим для примера, что получатель незамедлительно подтверждает прием каждого сегмента. Когда прикладная программа «потребит» данные и снова увеличит доступное пространство в буфере приема, получатель может передать второе подтверждение отправителю для обновления окна. Экстремальная ситуация возникает при 1-октетных сегментах в соединениях TCP, использующих протокол Telnet для удаленного входа в систему. Встречаются реализации, где на каждое нажатие клавиши пользователем генерируется три передаваемых в ответ сегмента: (1) подтверждение, (2) однобайтовое расширение окна и (3) эхо-символ.

4.2.2.15 Тайм-аут повторной передачи: RFC 793, параграф 3.7, стр. 41

Сейчас известно, что предложенный в RFC 793 алгоритм расчета тайм-аута для повторной передачи не является адекватным (см параграф 4.2.3.1).

Недавняя работа Якобсона [TCP:7] по вопросам насыщения Internet и стабильности повторной передачи TCP предлагает новый алгоритм, объединяющий механизмы slow start (замедленный старт) и congestion avoidance (предотвращение насыщения). Протокол TCP **должен** реализовать эти алгоритмы.

Если повторный пакет идентичен исходному (сохранены не только границы данных, но также окно и поля подтверждения в заголовке), **можно** использовать прежнее значение поля идентификации IP (см. 3.2.1.5).

Реализация

[1]Maximum Segment Lifetime - максимальное время жизни сегмента.
[2]Silly Window Syndrome.

Некоторые разработчики TCP выбирают «пакетированный» поток данных, т. е., указывают границы сегмента при передаче в очередь и держат эти границы в очереди на повторную передачу, пока не будет получено подтверждение. В других вариантах (они могут быть проще) пакетирование не происходит до завершения передачи (или повтора), поэтому очередь на повторную передачу в таких реализациях не используется.

В варианте с очередью на повторную передачу производительность TCP может быть повышена за счет повторного пакетирования для сегментов, ожидающих подтверждения при первом тайм-ауте повторной передачи. Оставшиеся сегменты будут объединяться в один сегмент максимального размера с новым идентификатором IP. После этого TCP будет держать новый сегмент в очереди на передачу, пока не будет получено подтверждение. Однако, если первые два сегмента в очереди на повторную передачу превышают максимальный размер сегмента, TCP будет повторять передачу только для первого сегмента из очереди с сохранением идентификационного поля IP.

4.2.2.16 Управление окном: RFC 793, параграф 3.7, стр. 41

Получателям TCP **не рекомендуется** отсекать часть окна (т. е., перемещать правый край окна влево). Однако отправитель TCP **должен** быть устойчивым к уменьшению окна, при котором значение «useable window» (см. 4.2.3.4) становится отрицательным.

Если такое происходит, отправителю **не следует** передавать новые данные, но следует повторить передачу неподтвержденных данных между SND.UNA и SND.UNA+SND.WND. Отправитель также **может** повторить передачу данных за пределами SND.UNA+SND.WND, но **не следует** прерывать соединение по тайм-ауту, если доставка данных за пределами правого края окна не подтверждена. Если окно сокращается до нуля, протокол TCP **должен** проверить его стандартным способом (см. ниже).

Обсуждение

Во многих реализациях TCP возникают проблемы при сокращении окна справа после передачи данных в окно большего размера. Отметим, что TCP использует эвристические методы для выбора последнего обновления окна, вопреки возможному изменению порядка дейтаграмм. В результате может быть проигнорировано обновление окна на меньшее, которое было предложено ранее, если не увеличивается ни порядковый номер, ни номер подтверждения.

4.2.2.17 Проверка нулевого окна: RFC 793, параграф 3.7, стр. 42

Должна поддерживаться проверка нулевого (предлагаемого) окна.

TCP **может** сохранять свои предлагаемые приемные окна закрытыми в течение неопределенного времени. До тех пор, пока TCP на приемной стороне продолжает слать подтверждения в ответ на проверочные сегменты, протокол TCP на передающей стороне **должен** сохранять соединение открытым.

Обсуждение

Очень важно помнить, что сегменты ACK, которые не содержат данных, передаются протоколом TCP без гарантий. Если проверка нулевого окна не поддерживается, соединение может быть разорвано при потере сегмента ACK, заново открывающего окно.

Задержка при открытии нулевого окна обычно происходит в тех случаях, когда принимающее приложение останавливает обмен данными с со своим TCP. Например, демон печати может остановить работу при отсутствии в принтере бумаги.

Передающему хосту **рекомендуется** посылать первую пробу нулевого окна, когда такое окно существовало в течение периода, равного тайм-ауту для передачи (см. 4.2.2.15); интервалы передачи последующих проб **рекомендуется** экспоненциально увеличивать.

Обсуждение

Эта процедура минимизирует задержку, если условие нулевого окна возникает в результате потери сегмента ACK, содержащего обновление открытия окна. Рекомендуется экспоненциальное увеличение интервала (возможно с заданием максимального значения). Эта процедура напоминает алгоритм повтора передачи и реализацию этих процессов можно объединить.

4.2.2.18 Пассивные вызовы OPEN: RFC 793, параграф 3.8

Каждый пассивный вызов OPEN создает новую запись о соединении в состоянии LISTEN или возвращает ошибку - это **не должно** оказывать влияния на любые ранее созданные записи соединений.

Реализации TCP, поддерживающие множество пользователей одновременно, **должны** обеспечивать вызовы OPEN, функционально позволяющие приложениям прослушивать (LISTEN) порт, пока не будет блокировано соединение с таким же номером локального порта в состоянии SYN-SENT или SYN-RECEIVED.

Обсуждение

Некоторые приложения (например, серверы SMTP) могут требовать обслуживания множества одновременных попыток организации соединения. Вероятность отказа при таких попытках может быть снижена за счет предоставления программе возможности прослушивания соединения одновременно с попыткой согласования параметров для ранее организованного соединения.

Реализация

Приемлемые реализации одновременных попыток соединения могут разрешать множество открытых пассивных вызовов OPEN или «клонирование» соединений в состоянии LISTEN из одного пассивного вызова OPEN.

4.2.2.19 Время жизни: RFC 793, параграф 3.9, стр. 51

В RFC 793 указано, что TCP будет запрашивать у IP-уровня передачу сегментов TCP со значением TTL = 60. Это требование устарело и значение TTL для передачи сегментов TCP **должно** быть настраиваемым (см. 3.2.1.7).

4.2.2.20 Обработка событий: RFC 793, параграф 3.9

Хоть это и не является обязательным, **следует** поддерживать для TCP очереди с нарушением порядка сегментов TCP (в RFC 793 на стр. 70 сказано **возможно**).

Обсуждение

Некоторые реализации для небольших хостов не используют очереди сегментов по причине ограниченности буферного пространства. Такое решение может привести к существенному снижению производительности TCP, поскольку потеря единственного сегмента приведет к тому, что все последующие сегменты будут доставляться с нарушением порядка.

В общем случае обработка принятых сегментов **должна** быть построена так, чтобы сегменты ACK по возможности объединялись. Например, если TCP обрабатывает группу сегментов из очереди, передача сегмента ACK **должна** происходить только после обработки всех таких сегментов.

Ниже приведены некоторые поправки к параграфу «Обработка событий» в RFC 793.

(a) Вызов CLOSE, состояние CLOSE-WAIT (стр. 61): следует читать LAST-ACK взамен CLOSING.

(b) Состояние LISTEN, проверка SYN (стр. 65, 66): При наличии бита SYN передача сбрасывается, если для сегмента некорректны значения security/compartment или precedence. В документе допущена ошибка. Правильная команда сброса показана ниже:

```
<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>
```

(c) Состояние SYN-SENT, проверка SYN (стр. 68): При переходе соединения в состояние ESTABLISHED должны быть установлены следующие переменные:

```
SND.WND <- SEG.WND
SND.WL1 <- SEG.SEQ
SND.WL2 <- SEG.ACK
```

(d) Проверка защиты и предпочтений (стр. 71): Первый заголовок ESTABLISHED STATE реально должен быть списком всех состояний, кроме SYN-RECEIVED - ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT.

(e) Вместо «check the SYN bit» (стр. 71) следует читать «In SYN-RECEIVED state and if the connection was initiated with a passive OPEN, then return this connection to the LISTEN state and return. Otherwise check the SYN bit[1]».

(f) Проверка поля ACK, состояние SYN-RECEIVED (стр. 72: При переходе соединения в состояние ESTABLISHED должны быть установлены переменные, указанные в п. (c).

(g) Проверка поля ACK, состояние ESTABLISHED (стр. 72): ACK является дубликатом, если SEG.ACK =< SND.UNA (знак = пропущен). Аналогичный пропуск в условии обновления окна - должно быть: SND.UNA =<SEG.ACK =< SND.NXT.

(h) USER TIMEOUT (стр. 77):

Лучше будет уведомлять приложение о тайм-ауте, а не о разрешении TCP закрыть соединение (см. также параграф 4.2.3.5).

4.2.2.21 Подтверждение для сегментов из очереди: RFC 793, параграф 3.9

TCP **может** передавать сегмент ACK, подтверждающий RCV.NXT, когда приходит корректный сегмент, который находится в окне, но не на его левой границе.

Обсуждение

В RFC 793 (стр. 74) нет ясности по вопросу передачи сегмента ACK при получении сегментов с нарушением порядка (т. е., SEG.SEQ не равно RCV.NXT).

Одной из причин передачи подтверждений для сегментов с нарушением порядка доставки может быть поддержка экспериментального алгоритма, названного fast retransmit (ускоренный повтор передачи). При использовании этого алгоритма отправитель передает избыточные подтверждения ACK для указания потери сегмента до истечения тайм-аута повторной передачи. Подсчитывается число полученных подтверждений ACK с одинаковым значением SEG.ACK и одинаковой правой границей окна. При получении большего числа ACK, нежели заданное пороговое значение, предполагается потеря сегмента, начинающегося с SEG.ACK, и выполняется повторная передача без ожидания тайм-аута. Пороговое значение выбирается таким образом, чтобы компенсировать максимальное разупорядочивание сегментов в Internet. Использование этого алгоритма пока слишком непродолжительно, чтобы сделать выводы о его полезности.

4.2.3 Частные вопросы

4.2.3.1 Расчет тайм-аута для повторной передачи

Хост TCP **должен** поддерживать алгоритмы Karn и Jacobson для расчета тайм-аута повторной передачи RTO.

- алгоритм Jacobson для расчета взвешенного времени кругового обхода (RTT) включает простое измерение вариаций [TCP:7];
- алгоритм Karn для выбора способа измерения RTT гарантирует, что случайные значения времени кругового обхода не будут уничтожать результат расчета взвешенного времени обхода [TCP:6].

Такая реализация **должна** также включать экспоненциальный рост последовательных значений RTO для одного сегмента. Для повторной передачи сегментов SYN **рекомендуется** использовать такой же алгоритм, как для сегментов данных.

Обсуждение

[1]В состоянии SYN-RECEIVED, если соединение было инициировано пассивным вызовом OPEN, это соединение должно переводиться обратно с состояние LISTEN с возвратом управления. В остальных случаях следует проверить флаг SYN.

Известны две проблемы, связанные с расчетом RTO, описанным в RFC 793. Во-первых, при наличии повторов точное измерение RTT становится затруднительным. Во-вторых, алгоритм расчета взвешенного времени кругового обхода неадекватен [TCP:7], поскольку использует некорректное предположение о малости и постоянстве вариаций RTT. Для решения этих проблем используются алгоритмы Karn и Jacobson, соответственно.

Повышение производительности в результате использования этих алгоритмов может колебаться от незначительного до гигантского. Алгоритм Jacobson для включения вариаций измеренных значений RTT особенно важен для низкоскоростных каналов, где естественные вариации размера пакетов приводят к значительным вариациям RTT. Один из разработчиков отметил рост эффективности использования канала 9.6 кбит/с с 10% до 90% в результате реализации алгоритма Jacobson для TCP.

Для инициализации оценочных параметров новых соединений **рекомендуется** использовать значения:

(a) RTT = 0 секунд;

(b) RTO = 3 секунды (взвешенные вариации инициализируются значением, которое будет влиять на RTO).

Известно, что рекомендованные значения верхней и нижней границ RTO неадекватны для больших сетей. **Рекомендуется** задавать нижнюю границу в долях секунды (для работы со скоростными ЛВС), а в качестве верхней границы использовать значение 2*MSL (240 секунд).

Обсуждение

Опыт показывает, что приведенные выше параметры инициализации имеют разумные значения, а использование алгоритмов Karn и Jacobson делает поведение TCP разумным, независимо от выбора начальных параметров.

4.2.3.2 Когда передавать сегмент ACK

Хост, получающий поток сегментов данных TCP, может повысить эффективность работы (хостов и Internet) за счет передачи меньшего числа подтверждений ACK для принятых сегментов. Такой метод называется задержкой подтверждений [TCP:5].

Для TCP **рекомендуется** реализовать задержку ACK, но такая задержка не должна быть слишком большой. В частности, задержка **должна** быть меньше 0.5 сек. и в потоке сегментов полного размера подтверждения должны передаваться по крайней мере для каждого второго сегмента.

Обсуждение

Задержка ACK позволяет приложениям обновить окно и, возможно, передать незамедлительный отклик. В частности, для систем удаленного доступа с посимвольным вводом задержка ACK может снижать число сегментов, передаваемых сервером, втрое (ACK, обновление окна и эхо-символ передаются в одном сегменте).

В дополнение к этому на некоторых крупных, многопользовательских хостах задержка ACK может существенно снизить затраты на обработку заголовков за счет уменьшения общего числа обрабатываемых пакетов [TCP:5]. Однако, чрезмерная задержка ACK может нарушать определение времени обхода и работу алгоритмов «тактирования» пакетов [TCP:7].

4.2.3.3 Когда передавать Window Update

Протокол TCP **должен** включать алгоритм предотвращения SWS на приемной стороне [TCP:5].

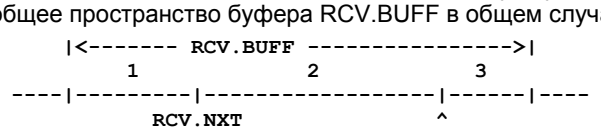
Реализация

Алгоритм предотвращения SWS на приемной стороне определяет, когда может быть анонсировано расширение правого края окна (обновление окна). Этот алгоритм используют совместно с задержкой подтверждений ACK (см. 4.2.3.2) для определения момента передачи получателю сегмента ACK, содержащего текущее окно. При описании мы будем использовать обозначения, принятые в RFC 793 (см. рис. 4 и 5 в этом документе).

Решением для приемной стороны является предотвращение анонсов смещения правого края окна RCV.NXT+RCV.WND с небольшим инкрементом даже при получении из сети мелких сегментов.

Предположим, что на приемной стороне имеется буфер RCV.BUFF. В любой момент времени RCV.USER октетов из этого буфера может быть связано с данными, которые уже получены и подтверждены, но еще не восприняты пользовательским процессом. Когда соединение находится в статичном состоянии, RCV.WND = RCV.BUFF и RCV.USER = 0.

Сохранение правого края окна неподвижным при получении и подтверждении данных требует чтобы получатель анонсировал пространство, которое меньше его реального буфера, т. е., получатель должен задавать значение RCV.WND, которое позволит сохранить постоянной сумму RCV.NXT+RCV.WND при возрастании RCV.NXT. Таким образом, общее пространство буфера RCV.BUFF в общем случае делится на три части:



(фиксированная часть)

1. RCV.USER - полученные, но не воспринятые приложением данные;
2. RCV.WND - пространство, анонсируемое отправителю;
3. Reduction - доступное, но еще не анонсированное пространство.

Предлагаемый алгоритм предотвращения SWS для приемной стороны сохраняет фиксированное значение RCV.NXT+RCV.WND до тех пор, пока выполняется условие:

```
RCV.BUFF - RCV.USER - RCV.WND  >= min(Fr * RCV.BUFF, Eff.snd.MSS)
```

где Fr - часть, рекомендуемое значение которой составляет 1/2, а Eff.snd.MSS - эффективное значение MSS для передачи в данном соединении (см. 4.2.2.6). При выполнении условий неравенства устанавливается RCV.WND = RCV.BUFF-RCV.USER.

Отметим, что общим эффектом этого алгоритма является анонсирование RCV.WND с инкрементом Eff.snd.MSS (для разумных буферов на приемной стороне Eff.snd.MSS < RCV.BUFF/2). Отметим также, что приемная сторона

должна использовать свое значение Eff.snd.MSS, предполагая, что оно совпадает со значением для передающей стороны.

4.2.3.4 Когда передавать данные

Протокол TCP **должен** поддерживать механизм предотвращения SWS на приемной стороне.

Для протокола TCP **рекомендуется** реализация алгоритма Nagle [TCP:9], позволяющего объединять короткие сегменты. Однако, приложениям **должен** обеспечиваться способ запрета алгоритма Nagle для отдельных соединений. Во всех случаях для передачи данных действуют также ограничения, вносимые алгоритмом Slow Start (см. 4.2.2.15).

Обсуждение

Алгоритм Nagle в общем случае действует следующим образом:

При наличии неподтвержденных данных (т. е., SND.NXT > SND.UNA) в буферы TCP передаются все пользовательские данные (не принимая во внимание бит PSH), пока остающиеся данные не будут подтверждены или пока TCP не сможет передать сегмент полного размера (Eff.snd.MSS байтов; см. 4.2.2.6).

Некоторые приложения (например, отображение окон в реальном масштабе времени) требуют запрета алгоритма Nagle, чтобы небольшие сегменты данных передавались с максимальной скоростью.

Реализация

Алгоритм предотвращения SWS на передающей стороне реализуется сложней, нежели на приемной, поскольку отправитель не знает (явно) размер буферного пространства RCV.BUFF на приемной стороне. Проверенным вариантом является расчет отправителем значения максимального окна передачи для соединения Max(SND.WND) и использование полученного значения для оценки RCV.BUFF. К сожалению, возможна только оценка буфера, поскольку приемная сторона может время от времени менять значение RCV.BUFF. Чтобы избежать застоя соединений в результате этого, необходимо использовать значение тайм-аута для форсирования передачи данных, имеющего преимущество перед алгоритмом предотвращения SWS. На практике форсирование передачи по тайм-ауту должно происходить редко.

Доступное окно имеет размер ([TCP:5]):

```
U = SND.UNA + SND.WND - SND.NXT
```

т. е., предлагаемое окно меньше, чем размер переданных, но не подтвержденных данных. Если D указывает количество данных в очереди на передачу TCP, рекомендуется использовать следующий набор правил для передачи данных:

(1) при достижении максимального размера передаваемого сегмента, т. е.:

```
min(D,U) >= Eff.snd.MSS;
```

(2) или установлен флаг push и все данные из очереди могут быть переданы, т. е.:

```
[SND.NXT = SND.UNA and] PUSHED and D <= U
```

(условие в квадратных скобках вносится алгоритмом Nagle);

(3) или по крайней мере Fs-ая часть максимального окна может быть передана, т. е.:

```
[SND.NXT = SND.UNA and] min(D.U) >= Fs * Max(SND.WND);
```

(4) или установлен флаг PUSH и достигнут тайм-аут.

Fs представляет собой часть, рекомендуемое значение которой составляет 1/2. Значение тайм-аута должно составлять 0.1 - 1.0 сек. Может оказаться удобным объединение этого таймера с таймером проверки нулевого окна, описанным в параграфе 4.2.2.17.

В заключение отметим, что использование алгоритма предотвращения SWS рекомендуется взамен алгоритма sender-side, описанного в работе [TCP:5].

4.2.3.5 Сбои в соединениях TCP

Многократные повторы передачи одного сегмента TCP свидетельствуют о наличии сбоев на удаленном хосте или пути через Internet. Эти сбои могут быть кратковременными или продолжительными. При возникновении таких ситуаций хост **должен** использовать перечисленные ниже процедуры [IP:11]:

(a) Существуют два пороговых значения R1 и R2 для измерения числа повторов передачи одного сегмента. Для задания R1 и R2 могут использоваться единицы времени или число повторов передачи.

(b) При достижении числа повторов R1 передается негативный анонс (см. 3.3.1.4) на уровень IP для включения диагностики работоспособности шлюзов.

(c) При достижении порога R2 (превышает R1) соединение закрывается.

(d) Приложениям **должна** обеспечиваться возможность установки порога R2 для отдельного соединения. Например, интерактивные соединения могут устанавливать для R2 «бесконечное» значение, предоставляя пользователю самостоятельно решать вопрос о разрыве соединения.

(e) Протоколу TCP **следует** информировать приложения о проблемах с доставкой (если это не запрещено приложением; см. 4.2.4.1), при достижении порога R1, но до порога R2. Такая информация позволяет программам удаленного доступа (типа Telnet) информировать пользователя о проблемах.

Следует устанавливать для R1 значение, соответствующее по крайней мере 3 повторам при текущем значении RTO. Для R2 **следует** задавать значение не менее 100 секунд.

При попытке создать соединение TCP может наблюдаться сбой с многократным повтором сегмента SYN, получением сегмента RST или сообщения ICMP Port Unreachable. Повторные передачи SYN **должны** обрабатываться обычным способом (как для данных), включая уведомление прикладного уровня.

Однако, значения R1 и R2 для сегментов SYN могут отличаться от значения для сегментов данных. В частности, значение R2 для SYN **должно** быть достаточно велико, чтобы обеспечивать повторные передачи по крайней мере в течение 3 минут. Приложение может закрыть соединение и раньше (например, задав число попыток).

Обсуждение

Для некоторых путей Internet время организации соединения достаточно велико и число таких путей может существенно возрасти в будущем.

4.2.3.6 TCP Keep-Alive

Разработчики **могут** включать в свои реализации TCP пакеты keep-alive, хотя такая практика не принята повсеместно. При включении keep-alive приложениям **должна** обеспечиваться возможность запрета таких пакетов для отдельных соединений TCP и по умолчанию такие пакеты **должны** быть запрещены.

Пакеты keep-alive **должны** передаваться только при отсутствии пакетов данных или подтверждений в течение заданного интервала. Значение этого интервала **должно** быть настраиваемым и по умолчанию **должно** составлять не менее 2 часов.

Очень важно помнить, что для сегментов ACK, не содержащих данных, протокол TCP не обеспечивает гарантии доставки. Следовательно, при реализации механизма keep-alive сбои в ответ на специфические проверки **не должны** интерпретироваться как «умирание» соединения.

Рекомендуется передавать сегменты keep-alive без данных, однако **можно** настраивать протокол на передачу сегментов keep-alive, содержащих один произвольный октет для совместимости с ошибочными реализациями TCP.

Обсуждение

Механизм keep-alive периодически проверяет удаленную сторону при простое соединения, даже если отсутствуют неподтвержденные данные. Спецификация TCP не включает механизма keep-alive, поскольку он: (1) может обрывать нормальные соединения в результате транзитных сбоев Internet, (2) приводит к ненужному расходу полосы и (3) повышает расходы для соединений Internet с платным трафиком.

Некоторые реализации TCP все же включают механизм keep-alive. Для подтверждения работоспособности бездействующего соединения такие реализации передают пробные сегменты, предназначенные для получения отклика удаленной стороны. Такие сегменты в общем случае включают поле SEG.SEQ=SND.NXT-1 и могут также содержать один произвольный октет данных. Отметим, что для бездействующих соединений SND.NXT=RCV.NXT, поэтому значение SEG.SEQ будет выходить за пределы окна. Следовательно, пробный сегмент будет заставлять приемник вернуть сегмент подтверждения, говорящий о том, что соединение сохраняет работоспособность. Если удаленная сторона разорвала соединение, она будет возвращать RST вместо подтверждающего сегмента.

К несчастью, в некоторых не вполне корректных реализациях TCP возникают сбои при получении сегмента с SEG.SEQ = SND.NXT-1, если такой сегмент не содержит данных. Приложение может дополнительно проверить, способна ли удаленная сторона корректно отвечать на пакеты keep-alive без вставленного в них произвольного октета данных.

Механизм TCP keep-alive следует использовать только в серверных приложениях, которые могут неограниченно долго сохранять соединения, потребляя без нужды сетевые ресурсы, даже если клиент по тем или иным причинам разорвал или потерял соединение.

4.2.3.7 Многодомные хосты TCP

Если приложение на многодомном хосте не указывает локальный IP-адрес при активной организации соединения TCP, протокол TCP **должен** запрашивать уровень IP для получения локального IP-адреса до (первой) передачи SYN. Более подробные сведения приведены в параграфе 3.4 - функция GET_SRCADDR().

Во всех остальных случаях через это соединение был уже передан или принят сегмент и протокол TCP **должен** использовать тот же локальный адрес, который применялся для предыдущего сегмента.

4.2.3.8 Опции IP

При передаче опций на уровень TCP со стороны IP, протокол TCP **должен** игнорировать непонятные опции.

TCP **может** поддерживать опции Time Stamp и Record Route.

Приложениям **должна** обеспечиваться возможность задания source route при активной организации соединения TCP и этот маршрут должен иметь преимущество перед source route из принятой дейтаграммы.

При пассивной организации соединения TCP и достижении пакетом конечной точки, указанной опцией IP Source Route (содержащей путь возврата), протокол TCP **должен** сохранить маршрут возврата и использовать его для всех сегментов, передаваемых через это соединение. Если в последующих сегментах приходит иной маршрут source route, новый маршрут **следует** использовать взамен прежнего.

4.2.3.9 Сообщения ICMP

Протокол TCP **должен** передавать сообщения ICMP об ошибках, полученные с уровня IP, соединениям, с которыми связаны ошибки. Демультиплексирование осуществляется на основе заголовков IP, содержащих сообщения ICMP.

- Source Quench
  Протокол TCP **должен** реагировать на сообщения Source Quench замедлением передачи через соединение. Для реализации этого **следует** использовать процедуру как при тайм-ауте повторной передачи.

- Destination Unreachable - коды 0, 1, 5
  Поскольку такие сообщения говорят о кратковременных ошибках, протокол TCP **не должен** прерывать соединение; **рекомендуется** передать эту информацию приложению.

Обсуждение

Протокол TCP может сообщать о некритичных ошибках непосредственно прикладному уровню с помощью процедуры ERROR_REPORT; допускается также информировать приложения только при возникновении тайм-аута для соединения TCP.

- Destination Unreachable - коды 2-4
  Эти сообщения говорят о серьезных ошибках, поэтому **следует** разрывать соединения TCP.

- Time Exceeded - коды 0, 1
  Эти ошибки трактуются аналогично Destination Unreachable с кодами 0, 1, 5 (см. выше).

- Parameter Problem
  Эти ошибки трактуются аналогично Destination Unreachable с кодами 0, 1, 5 (см. выше).

4.2.3.10 Проверка корректности удаленного адреса

Реализация TCP **должна** отбрасывать (как ошибочные) локальные вызовы OPEN с некорректным IP-адресом удаленной стороны (например, широковещательный или групповой адрес).

Входящие запросы SYN с некорректным адресом отправителя должны игнорироваться уровнем TCP или IP (см. 3.2.1.3).

Реализация TCP **должна** без уведомления отбрасывать все входящие сегменты SYN с широковещательными или групповыми адресами.

4.2.3.11 Картины трафика TCP

Реализация

Спецификация протокола TCP [TCP:1] предоставляет разработчикам свободу выбора алгоритма для контроля за потоком сообщений через соединение - пакетирование, управление окном, передача подтверждений и т. д. Выбрать алгоритм непросто, поскольку протокол TCP должен адаптироваться к различным картинам трафика. Опыт показывает, что разработчикам TCP требуется проверять реализацию для двух экстремальных вариантов трафика:

- **Односимвольные сегменты**
  Даже при использовании передающей стороной алгоритма Nagle соединения TCP для удаленного входа в систему через ЛВС с малой задержкой будут порождать поток односимвольных сегментов. Если на удаленном терминале включен режим эхо-символов, принимающая сторона будет в общем случае генерировать отклик на каждый принятый символ.

- **Передача больших объемов данных**
  При использовании TCP для передачи большого объема данных поток почти полностью будет состоять из сегментов размером MSS (эффективное значение). Хотя TCP использует пространство порядковых номеров с байтовой (октетной) гранулярностью, при передаче больших объемов данных должны учитываться только сегменты.

Опыт показывает что эффективные и корректные реализации TCP хорошо работают в обоих экстремальных случаях.

Наиболее важным средством для проверки новых реализаций TCP являются программы трассировки пакетов. Многолетний опыт показывает важность трассировки для различных картин трафика при использовании с другими реализациями TCP для поиска ошибок.

4.2.3.12 Эффективность

Реализация

На основании накопленного опыта выработаны рекомендации для разработчиков TCP.

(a) **Не копируйте данные**
При передаче больших объемов данных наибольшую нагрузку на процессор создает копирование данных из одного места в другое для определения контрольной суммы. Важно минимизировать число копий данных TCP. Поскольку передача данных через шину памяти может существенно ограничивать скорость, полезно объединять копирование данных с вычислением контрольных сумм.

(b) **Внимательно относитесь к вычислению контрольных сумм**
Хорошие программы вычисления контрольных сумм TCP обычно в 2-5 раз быстрее, по сравнению с простой реализацией определений CRC. Для эффективного вычисления контрольных сумм требуется программирование высокого класса (см. [TCP:10]).

(c) **Код общего назначения**
Обработка протокола TCP может быть сложной, но для большинства сегментов используется лишь несколько простых решений. Посегментная обработка существенно ускоряется за счет эффективного кодирования основной линии с минимизацией числа принимаемых решений для наиболее вероятных ситуаций.

4.2.4 Интерфейс между TCP и прикладным уровнем

4.2.4.1 Асинхронные отчеты

Должен обеспечиваться механизм информирования приложений о некритичных ошибках TCP. В общем случае это реализуется с помощью прикладной процедуры ERROR_REPORT, которая может асинхронно [INTRO:7] вызываться с транспортного уровня:

```
ERROR_REPORT(local connection name, reason, subreason)
```

Кодирование причин ошибок не рассматривается здесь, однако сообщения, асинхронно передаваемые приложениям, **должны** включать:

- полученные сообщения ICMP об ошибках (см. 4.2.3.9);

- информацию о многократных повторах передачи (см. 4.2.3.5);

- анонсы указателей срочности (см. 4.2.2.4).

Для программ, которые не хотят получать информации об ошибках ERROR_REPORT, **следует** отключить вызов процедуры.

Обсуждение

Сообщения в общем случае включают сведения о некритичных ошибках, которые можно игнорировать без вреда для большинства приложений. Предполагается, что по умолчанию передача отчетов об ошибках запрещена, но это не обязательно.

4.2.4.2 Тип обслуживания

Прикладному уровню **должна** быть обеспечена возможность задавать тип обслуживания TOS для сегментов, передаваемых через соединение. **Следует** также обеспечивать приложениям возможность изменения TOS в процессе использования соединения. Протоколу TCP **следует** передавать текущее значение TOS без изменений на уровень IP при передаче сегмента в соединение.

Значение TOS задается независимо для каждого направления в соединении, поэтому принимающее приложение будет задавать TOS для сегментов ACK.

TCP **может** передавать приложениям последнее использованное значение TOS из принятых сегментов.

Обсуждение

Некоторые приложения (например, SMTP) меняют тип обмена данными во время соединения и, следовательно, могут пожелать изменить значение TOS.

Отметим также, что вызов OPEN в соответствии с RFC 793 включает параметр options, в котором могут быть заданы опции IP (source route, record route, timestamp).

4.2.4.3 Вызов Flush

Некоторые реализации TCP включают вызовы FLUSH, которые очищают очередь передачи TCP от всех данных, помещенных в нее с помощью вызовов SEND из прикладных программ, но сохраняет данные, остающиеся в правой части текущего окна передачи. Таким образом, эта функция удаляет из очереди как можно больше данных без потери синхронизации порядковых номеров. Это полезно для реализации функций типа abort output в Telnet.

4.2.4.4 Многодомные хосты

Пользовательский интерфейс, описанный в параграфах 2.7 и 3.8 RFC 793, требует расширения для многодомных хостов. Функция OPEN **должна** поддерживать необязательный параметр с локальным адресом:

```
OPEN( ... [local IP address,] ... )
```

Обсуждение

Некоторые приложения на базе TCP (например, FTP) требуют указывать локальный IP-адрес, который используется для организации соединения.

Реализация

Пассивный вызов OPEN с заданным локальным адресом IP будет ждать входящего запроса на соединение с этим адресом. Если параметр не указан, пассивный вызов OPEN будет ждать запроса на входящее соединение по любому локальному адресу IP и потом связывать локальный IP-адрес с этим соединением.

Для активных вызовов OPEN указанный локальный IP-адрес будет использоваться для организации соединения. Если параметр не задан, сетевая программа будет выбирать подходящий локальный адрес IP (см. 3.3.4.2) для организации соединения.

4.2.5 Требования к протоколу TCP

Функция	*Параграф*	*Требование*
Флаг Push		
Объединение или очередь при отсутствии флага Push	4.2.2.2	Возможно
Передающая сторона удаляет последовательные флаги Push	4.2.2.2	Следует
При вызове функции SEND можно установить Push	4.2.2.2	Возможно
При отсутствии Push бесконечный буфер передачи	4.2.2.2	Недопустимо
При отсутствии Push установка PSH для последнего сегмента	4.2.2.2	Обязательно
Уведомление принимающей программы о PSH	4.2.2.2	Возможно
Передача по возможности сегментов максимального размера	4.2.2.2	Следует
Окно		
Размер трактуется как беззнаковое целое	4.2.2.3	Обязательно
Поддержка 32-битового поля размера	4.2.2.3	Следует
Сокращение окна справа	4.2.2.16	Не следует
Устойчивость к сокращению окна	4.2.2.16	Обязательно
Неопределенное закрытие окна приемником	4.2.2.17	Возможно
Отправитель проверяет нулевое окно	4.2.2.17	Обязательно
Первая проверка после RTO	4.2.2.17	Следует
Экспоненциальное увеличение интервала проверки	4.2.2.17	Следует
Возможность неопределенного обнуления окна	4.2.2.17	Обязательно
Тайм-аут для нормального соединения с нулевым окном	4.2.2.17	Недопустимо
Срочные данные		
Указатель на последний октет	4.2.2.4	Обязательно
Последовательности срочных данных произвольной длины	4.2.2.4	Обязательно
Асинхронное уведомление приложений о срочных данных	4.2.2.4	Обязательно
Приложение может узнавать о наличии срочных данных	4.2.2.4	Обязательно

Функция	*Параграф*	*Требование*
Опции TCP		
Получение опций в любом сегменте	4.2.2.5	Обязательно
Игнорировать неподдерживаемые опции	4.2.2.5	Обязательно
Устойчивость к опциям некорректного размера	4.2.2.5	Обязательно
Реализация приема и передачи опции MSS	4.2.2.6	Обязательно
Передача опции MSS, если максимальный размер не равен 536	4.2.2.6	Следует
Передача опции MSS во всех случаях	4.2.2.6	Возможно
Значение MSS для передачи по умолчанию равно 536	4.2.2.6	Обязательно
Расчет эффективного размера сегмента передачи	4.2.2.6	Обязательно
Контрольные суммы TCP		
Отправитель рассчитывает контрольную сумму	4.2.2.7	Обязательно
Получатель проверяет контрольную сумму	4.2.2.7	Обязательно
Установка начального номера по текущему времени	4.2.2.9	Обязательно
Организация соединений		
Поддержка одновременных попыток	4.2.2.10	Обязательно
SYN-RCVD помнит последнее состояние	4.2.2.11	Обязательно
Пассивные вызовы CALL могут мешать друг другу	4.2.2.18	Недопустимо
Функция одновременного прослушивания для одного порта	4.2.2.18	Обязательно
Запрос адреса отправителя на уровне IP при необходимости	4.2.3.7	Обязательно
В противном случае использовать локальные адреса соединения	4.2.3.7	Обязательно
OPEN для групповых и широковещательных IP-адресов	4.2.2.14	Недопустимо
Отбрасывание сегментов для групповых/широковещательных адресов	4.2.2.14	Обязательно
Завершение соединений		
Сегмент RST может содержать данные	4.2.2.12	Следует
Информирование приложений о разрыве соединения	4.2.2.13	Обязательно
Полудуплексное закрытие соединений	4.2.2.13	Возможно
Передача RST для индикации потери данных	4.2.2.13	Следует
Сохранять состояние TIME-WAIT в течение 2 x MSL	4.2.2.13	Обязательно
Восприятие новых SYN во время TIME-WAIT	4.2.2.13	Возможно
Повторная передача		
Алгоритм Jacobson Slow Start	4.2.2.15	Обязательно
Алгоритм Jacobson Congestion-Avoidance	4.2.2.15	Обязательно
Повторная передача с сохранением идентификации IP	4.2.2.15	Возможно
Алгоритм Karn	4.2.3.1	Обязательно
Алгоритм Якобсона для оценки RTO	4.2.3.1	Обязательно
Экспоненциальное увеличение тайм-аута	4.2.3.1	Обязательно
Расчет SYN RTO как для данных	4.2.3.1	Следует
Рекомендуемые начальные значения и границы	4.2.3.1	Следует
Генерация подтверждений (ACK)		
Очередь для сегментов с нарушением порядка	4.2.2.20	Следует
Обработка всей очереди до передачи подтверждения	4.2.2.20	Обязательно
Передача ACK для сегментов с нарушением порядка	4.2.2.21	Возможно
Отложенные подтверждения	4.2.3.2	Следует
Задержка < 0.5 сек	4.2.3.2	Обязательно
Подтверждается каждый 2-ой сегмент полного размера	4.2.3.2	Обязательно
Алгоритм предотвращения SWS на приемной стороне	4.2.3.3	Обязательно
Передача данных		
Настраиваемое значение TTL	4.2.2.19	Обязательно
Алгоритм предотвращения SWS на передающей стороне	4.2.3.4	Обязательно
Алгоритм Nagle	4.2.3.4	Следует
Приложение может отключить алгоритм Nagle	4.2.3.4	Обязательно
Сбои в соединениях		
Негативный анонс для IP при достижении R1	4.2.3.5	Обязательно
Закрытие соединения при достижении R2	4.2.3.5	Обязательно
Приложения могут устанавливать R2	4.2.3.5	Обязательно
Информировать прикладной уровень после R1, но до R2	4.2.3.5	Следует
Рекомендуемые значения для R1 и R2	4.2.3.5	Следует
Поддержка такого же механизма для SYN	4.2.3.5	Обязательно
Значение R2 для SYN не менее 3 минут	4.2.3.5	Обязательно
Передача пакетов Keep-Alive	4.2.3.6	Возможно
Приложение может передавать запросы	4.2.3.6	Обязательно
По умолчанию механизм отключен	4.2.3.6	Обязательно
Возможность передачи только во время бездействия	4.2.3.6	Обязательно
Возможность настройки интервала	4.2.3.6	Обязательно
По умолчанию интервал не менее 2 часов	4.2.3.6	Обязательно
Устойчивость к потере подтверждений	4.2.3.6	Обязательно
Опции IP		
Игнорировать опции, не понятные TCP	4.2.3.8	Обязательно
Поддержка временных меток	4.2.3.8	Возможно
Поддержка записи маршрута	4.2.3.8	Возможно
Source Route		
Возможность задать из приложения	4.2.3.8	Обязательно
Переписывание Source Route в дейтаграммах	4.2.3.8	Обязательно
Построение маршрута возврата по исходному	4.2.3.8	Обязательно
Изменение Source Route дял новых сегментов	4.2.3.8	Следует

Функция	*Параграф*	*Требование*
Прием сообщений ICMP от уровня IP	4.2.3.9	Обязательно
Destination Unreach (0,1,5) => информировать приложение	4.2.3.9	Следует
Destination Unreach (0,1,5) => разорвать соединение	4.2.3.9	Недопустимо
Destination Unreach (2-4) => разорвать соединение	4.2.3.9	Следует
Source Quench => замедленный старт	4.2.3.9	Следует
Time Exceeded => информировать приложение без разрыва соединения	4.2.3.9	Следует
Param Problem => информировать приложение без разрыва соединения	4.2.3.9	Следует
Проверка адресов		
Отказ для вызовов CALL с неверным адресом IP	4.2.3.10	Обязательно
Отказ для SYN от некорректных адресов IP	4.2.3.10	Обязательно
Отбрасывание без уведомления SYN с широковещательными/групповыми адресами	4.2.3.10	Обязательно
Интерфейс между TCP и приложениями		
Механизм информирования об ошибках	4.2.4.1	Обязательно
Приложение может отключать информирование об ошибках	4.2.4.1	Следует
Приложение может задавать TOS для передачи	4.2.4.2	Обязательно
Передача без изменений на уровень IP	4.2.4.2	Следует
Приложение может менять TOS для действующего соединения	4.2.4.2	Следует
Передача приложению полученного TOS	4.2.4.2	Возможно
Вызов FLUSH	4.2.4.3	Возможно
Адрес IP как необязательный параметр OPEN	4.2.4.4	Обязательно

5. Литература

Введение

[INTRO:1] "Requirements for Internet Hosts -- Application and Support[1]," IETF Host Requirements Working Group, R. Braden, Ed., RFC 1123, October 1989.

[INTRO:2] "Requirements for Internet Gateways," R. Braden and J. Postel, RFC 1009, June 1987.

[INTRO:3] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:4] "Official Internet Protocols,"[2] J. Reynolds and J. Postel, RFC 1011, May 1987.

[INTRO:5] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC 980, March 1986.

[INTRO:6] "Assigned Numbers," J. Reynolds and J. Postel, RFC 1010[3], May 1987.

[INTRO:7] "Modularity and Efficiency in Protocol Implementations," D. Clark, RFC 817, July 1982.

[INTRO:8] "The Structuring of Systems Using Upcalls," D. Clark, 10th ACM SOSP, Orcas Island, Washington, December 1985.

Дополнительная информация

[INTRO:9] "A Protocol for Packet Network Intercommunication," V. Cerf and R. Kahn, IEEE Transactions on Communication, May 1974.

[INTRO:10] "The ARPA Internet Protocol," J. Postel, C. Sunshine, and D. Cohen, Computer Networks, Vol. 5, No. 4, July 1981.

[INTRO:11] "The DARPA Internet Protocol Suite," B. Leiner, J. Postel, R. Cole and D. Mills, Proceedings INFOCOM 85[4], IEEE, Washington DC, March 1985.

[INTRO:12] "Final Text of DIS8473, Protocol for Providing the Connectionless Mode Network Service,"[5] ANSI, March 1986.

[INTRO:13] "End System to Intermediate System Routing Exchange Protocol," ANSI X3S3.3[6], April 1986.

Канальный уровень

[LINK:1] "Trailer Encapsulations," S. Leffler and M. Karels, RFC 893, April 1984.

[LINK:2] "An Ethernet Address Resolution Protocol," D. Plummer, RFC 826, November 1982.

[LINK:3] "A Standard for the Transmission of IP Datagrams over Ethernet Networks," C. Hornig, RFC 894, April 1984.

[LINK:4] "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks," J. Postel and J. Reynolds, RFC 1042, February 1988.

Уровень IP

[IP:1] "Internet Protocol (IP)," J. Postel, RFC 791, September 1981.

[IP:2] "Internet Control Message Protocol (ICMP)," J. Postel, RFC 792, September 1981.

[IP:3] "Internet Standard Subnetting Procedure," J. Mogul and J. Postel, RFC 950, August 1985.

[IP:4] "Host Extensions for IP Multicasting," S. Deering, RFC 1112[7], August 1989.

[1] В RFC 2181 внесены уточнения и изменения для этого документа. *Прим. перев.*
[2] Последний вариант этого документа опубликован в RFC 5000. *Прим. перев.*
[3] В соответствии с RFC 3232 этот документ отменен и выделенные значения перенесены в базу данных IANA. *Прим. перев.*
[4] Этот документ также опубликован, как ISI-RS-85-153 и статья в IEEE Communications Magazine, March 1985.
[5] Этот документ также опубликован, как RFC 994.
[6] Этот документ также опубликован, как RFC 995.
[7] В RFC 2236 включены добавления к RFC 1112 в части протокола IGMP. *Прим. перев.*

[IP:5] "Military Standard Internet Protocol," MIL-STD-1777[8], Department of Defense, August 1983.

[IP:6] "Some Problems with the Specification of the Military Standard Internet Protocol," D. Sidhu, RFC 963, November 1985.

[IP:7] "The TCP Maximum Segment Size and Related Topics,"[9] J. Postel, RFC 879, November 1983.

[IP:8] "Internet Protocol Security Options," B. Schofield, RFC 1108, October 1989.

[IP:9] "Fragmentation Considered Harmful,"[10] C. Kent and J. Mogul, ACM SIGCOMM-87, August 1987. Published as ACM Comp Comm Review, Vol. 17, no. 5.

[IP:10] "IP Datagram Reassembly Algorithms,"[11] D. Clark, RFC 815, July 1982.

[IP:11] "Fault Isolation and Recovery," D. Clark, RFC 816, July 1982.

Дополнительная информация

[IP:12] "Broadcasting Internet Datagrams in the Presence of Subnets," J. Mogul, RFC 922, October 1984.

[IP:13] "Name, Addresses, Ports, and Routes," D. Clark, RFC 814, July 1982.

[IP:14] "Something a Host Could Do with Source Quench: The Source Quench Introduced Delay (SQUID)[12]," W. Prue and J. Postel, RFC 1016, July 1987.

Протокол UDP

[UDP:1] "User Datagram Protocol," J. Postel, RFC 768, August 1980.

Протокол TCP

[TCP:1] "Transmission Control Protocol," J. Postel, RFC 793, September1981.

[TCP:2] "Transmission Control Protocol," MIL-STD-1778[13], US Department of Defense, August 1984.

[TCP:3] "Some Problems with the Specification of the Military Standard Transmission Control Protocol," D. Sidhu and T. Blumer, RFC 964, November 1985.

[TCP:4] "The TCP Maximum Segment Size and Related Topics," J. Postel, RFC 879, November 1983.

[TCP:5] "Window and Acknowledgment Strategy in TCP," D. Clark, RFC 813, July 1982.

[TCP:6] "Round Trip Time Estimation," P. Karn & C. Partridge, ACM SIGCOMM-87, August 1987.

[TCP:7] "Congestion Avoidance and Control," V. Jacobson, ACM SIGCOMM-88, August 1988.

Дополнительная информация

[TCP:8] "Modularity and Efficiency in Protocol Implementation," D. Clark, RFC 817, July 1982.

[TCP:9] "Congestion Control in IP/TCP," J. Nagle, RFC 896, January 1984.

[TCP:10] "Computing the Internet Checksum," R. Braden, D. Borman, and C. Partridge, RFC 1071, September 1988.

[TCP:11] "TCP Extensions for Long-Delay Paths," V. Jacobson & R. Braden, RFC 1072[14], October 1988.

Вопросы безопасности

С программами различных коммуникационных уровней хостов связано множество вопросов безопасности, но их обсуждение выходит за рамки данного RFC.

Архитектура Internet в общем случае обеспечивает весьма слабую защиту против подмены IP-адресов отправителя, поэтому любой механизм обеспечения безопасности на основе IP-адресов отправителей должен применяться с осторожностью. Однако в ограниченной среде некоторая проверка адресов отправителей становится вполне возможной. Например, можно создать безопасную ЛВС, входной маршрутизатор которой будет отбрасывать любые дейтаграммы, где в качестве отправителя указан внутренний адрес локальной сети. В этом случае хост ЛВС может различать внешние и внутренние хосты по адресу отправителя. Проблема усложняется при задании маршрута отправителем и существуют предложения запретить хостам рассылку дейтаграмм source-route из соображений безопасности (см. 3.3.5).

Вопросы безопасности рассматриваются в параграфах, связанных с опцией IP Security (см. 3.2.1.8), сообщениями ICMP Parameter Problem (см. 3.2.2.5), опциями IP в дейтаграммах UDP (см. 4.1.3.2) и резервированием портов TCP (см. 4.2.2.1).

Адрес автора

Robert Braden

[8]Эта спецификация, как указано в RFC 963, предназначена для описания протокола IP, но в ней пропущены важные моменты (например, обязательная поддержка подсетей [IP:3] и дополнительная поддержка групповой адресации [IP:4]). Кроме того, документ достаточно устарел. При возникновении конфликтов с этим документов преимущество следует отдавать RFC 791, RFC 792 и RFC 950, а настоящий документ имеет преимущество над всеми. *Прим. перев.*

[9]Обсуждаются соотношения между TCP Maximum Segment Size и размером дейтаграмм IP.

[10]В этой полезной статье обсуждаются вопросы фрагментации и представлен ряд дополнительных решений проблемы.

[11]Этот и следующий документ должен прочесть каждый разработчик.

[12]В документе впервые описана направленная широковещательная адресация. Однако большая часть этого RFC посвящена маршрутизаторам и хостам.

[13]Эта спецификация, как указано в RFC 964, описывает тот же протокол, что и RFC 793 [TCP:1]. При возникновении конфликтов преимущество должно отдаваться RFC 793, а данный документ имеет преимущество по отношению к обоим.

[14]Этот документ признан устаревшим и заменен RFC 1323, RFC 2018 и RFC 6247. *Прим. перев.*

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@protokols.ru