# Ethernet

Ethernét (эзернет, от лат. aether — эфир) — пакетная технология компьютерных сетей, преимущественно локальных.

Стандарты Ethernet определяют проводные соединения и электрические сигналы на физическом уровне, формат кадров и протоколы управления доступом к среде — на канальном уровне модели OSI. Ethernet в основном описывается стандартами IEEE группы 802.3.

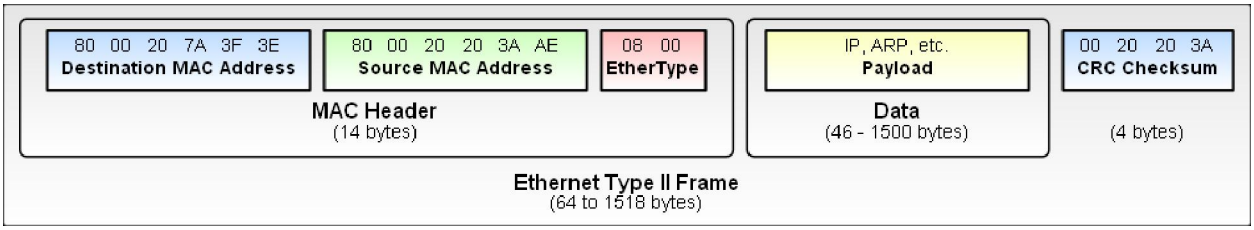
| Основные протоколы TCP/IP по уровням | |
|---|---|
| Прикладной | BGP • HTTP • DHCP • IRC • SNMP • DNS • NNTP • XMPP • SIP • BitTorrent • IPP • NTP • SNTP<br>Электронная почта SMTP • POP3 • IMAP4<br>Передача файлов FTP • TFTP • SFTP<br>Удалённый доступ rlogin • Telnet • SSH |
| Представления | XDR |
| Транспортный | TCP • UDP • SCTP • DCCP • RTP • RUDP |
| Сетевой уровень | IPv4 • IPv6 • ARP • RARP • ICMP • IGMP |
| Канальный | Ethernet • 802.11 WiFi • 802.16 WiMax • Token ring • Arcnet • FDDI • PPP • HDLC • SLIP • ATM • DTM • X.25 • Frame relay • SMDS |
| Физический | Ethernet • RS-232 • EIA-422 • RS-449 • RS-485 |

**Формат кадра**

Существует несколько форматов Ethernet-кадра.

Первоначальный Version I (больше не применяется).

Ethernet Version 2 или Ethernet-кадр II, ещё называемый DIX (аббревиатура первых букв фирм-разработчиков DEC, Intel, Xerox) — наиболее распространена и используется по сей день. Часто используется непосредственно протоколом интернет.




Наиболее распространенный формат кадра Ethernet II

Novell — внутренняя модификация IEEE 802.3 без LLC (Logical Link Control).

Кадр IEEE 802.2 LLC.

Кадр IEEE 802.2 LLC/SNAP.

Некоторые сетевые карты Ethernet, производимые компанией Hewlett-Packard использовали при работе кадр формата IEEE 802.12, соответствующий стандарту 100VG-AnyLAN.

В качестве дополнения Ethernet-кадр кадр может содержать тег IEEE 802.1Q для идентификации VLAN, к которой он адресован, и IEEE 802.1p для указания приоритетности.

Разные типы кадра имеют различный формат и значение MTU.

| Таблица 1.2. Коды протоколов Интернет | | |
|---|---|---|
| **код протокола Интернет** | **Сокращенное название протокола** | **описание** |
| 0 | - | Зарезервировано |
| 1 | ICMP | Протокол контрольных сообщений [rfc-792] |
| 2 | IGMP | Групповой протокол управления [rfc-1112] |
| 3 | GGP | Протокол маршрутизатор-маршрутизатор [RFC-823] |
| 4 | IP | IP поверх IP (инкапсуляция/туннели) |
| 5 | ST | Поток [rfc-1190] |
| 6 | TCP | Протокол управления передачей [RFC-793] |
| 7 | UCL | UCL |
| 8 | EGP | Протокол внешней маршрутизации [RFC-888] |
| 9 | IGP | Протокол внутренней маршрутизации |
| 10 | BBN-MON | BBN-RCC мониторирование |
| 11 | NVP-II | Сетевой протокол для голосовой связи [RFC-741] |
| 12 | PUP | PUP |
| 13 | ARGUS | Argus |
| 14 | Emcon | Emcon |
| 15 | Xnet | Перекрестный сетевой отладчик [IEN-158] |
| 16 | Chaos | Chaos |
| 17 | UDP | Протокол дейтаграмм пользователя [RFC-768] |
| 18 | MUX | Мультиплексирование [IEN-90] |
| 19 | DCN-MEAS | DCN измерительные субсистемы |
| 20 | HMP | Протокол мониторирования ЭВМ (host [RFC-869]) |
| 21 | PRM | Мониторирование при передаче пакетов по радио |
| 22 | XNS-IDP | Xerox NS IDP |
| 23 | Trunk-1 | Trunk-1 |
| 24 | Trank-2 | Trunk-2 |
| 25 | Leaf-1 | Leaf-1 |
| 26 | Leaf-2 | Leaf-2 |
| 27 | RDP | Протокол для надежной передачи данных [RFC-908] |
| 28 | IRTP | Надежный TP для Интернет [RFC-938] |
| 29 | ISO-TP4 | ISO транспортный класс 4 [RFC-905] |
| 30 | Netblt | Массовая передача данных [RFC-969] |
| 31 | MFE-NSP | Сетевая служба MFE |
| 32 | Merit-INP | Межузловой протокол Merit |
| 33 | SEP | Последовательный обмен |
| 34 | | не определен |
| 35 | IDRP | Междоменный протокол маршрутизации |
| 36 | XTP | Xpress транспортный протокол |
| 37 | DDP | Протокол доставки дейтаграмм |
| 38 | IDPR-CMTP | IDPR передача управляющих сообщений |
| 39 | TP++ | TP++ транспортный протокол |
| 40 | IL | IL-транспортный протокол |
| 41 | SIP | Простой Интернет-протокол |
| 42 | SDRP | Протокол маршрутных запросов для отправителя |
| 43 | SIP-SR | SIP исходный маршрут |
| 44 | SIP-Frag | SIP-фрагмент |
| 45 | IDRP | Интер-доменный маршрутный протокол |
| 46 | RSVP | Протокол резервирования ресурсов канала |
| 47 | GRE | Общая инкапсуляция маршрутов |

| | | |
|---|---|---|
| 49 | BNA | BNA |
| 50 | SIPP-ESP | SIPP ES3 |
| 52 | I-NLSP | Интегрированная система безопасности сетевого уровня |
| 53 | Swipe | IP с кодированием |
| 54 | NHRP | Nbma протокол определения следующего шага |
| 55-60 | | не определены |
| 61 | | Любой внутренний протокол ЭВМ |
| 62 | CFTP | CFTP |
| 63 | | Любая локальная сеть |
| 64 | Sat-Expak | Satnet и Expak |
| 65 | MIT-Subn | Поддержка субсетей MIT |
| 66 | RVD | Удаленный виртуальный диск MIT |
| 67 | IPPC | IPPC |
| 68 | | Любая распределенная файловая система |
| 69 | Sat-Mon | Мониторирование Satnet |
| 70 | | не определен |
| 71 | IPCV | Базовая пакетная утилита |
| 75 | PVP | Пакетный видео-протокол |
| 76 | BRsat-Mon | Резервное мониторирование Satnet |
| 78 | Wb-mon | Мониторирование Expak |
| 79 | Wb-expak | Широкополосная версия Expak |
| 80 | ISO-IP | ISO Интернет протокол |
| 88 | IGRP | IGRP (Cisco) — внутренний протокол маршрутизации |
| 89 | OSPFIGP | OSPFIGP — внутренний протокол маршрутизации |
| 92 | MTP | Транспортный протокол мультикастинга |
| 101-254 | | не определены |
| 255 | | Зарезервировано |

# ICMP

| **ICMP** | |
|---|---|
| **Название:** | *Internet Control Message Protocol* |
| **Уровень (по модели OSI):** | Сетевой |
| **Семейство:** | TCP/IP |
| **Спецификация:** | RFC 950 |

**ICMP** (англ. Internet Control Message Protocol — межсетевой протокол управляющих сообщений) — сетевой протокол, входящий в стек протоколов TCP/IP. В основном ICMP используется для передачи сообщений об ошибках и других исключительных ситуациях, возникших при передаче данных. Также на ICMP возлагаются некоторые сервисные функции.

Протокол ICMP описан в RFC 792 (с дополнениями в RFC 950) и является стандартом Интернета (входит в стандарт STD 5 вместе с IP). Хотя формально ICMP использует IP (ICMP пакеты инкапсулируются в IP пакеты), он является неотъемлемой частью IP и обязателен при реализации стека TCP/IP. Текущая версия ICMP для IPv4 называется ICMPv4. В IPv6 существует аналогичный протокол ICMPv6.

Протокол ICMP не делает протокол IP средством надёжной доставки сообщений. Для этих целей существует TCP.

ICMP сообщения (тип 12) генерируются при нахождении ошибок в заголовке IP пакета (за исключением самих ICMP пакетов, дабы не привести к бесконечно растущему потоку ICMP сообщений об ICMP сообщениях).

ICMP сообщения (тип 3) генерируются маршрутизатором при отсутствии маршрута к адресату.

Утилита ping, служащая для проверки возможности доставки IP пакетов использует ICMP сообщения с типом 8 (эхо-запрос) и 0 (эхо-ответ).

Утилита traceroute, отображающая путь следования IP пакетов, использует ICMP сообщения с типом 11.

ICMP сообщения с типом 5 используются маршрутизаторами для обновления записей в таблице маршрутизации отправителя.

ICMP сообщения с типом 4 используются получателем (или промежуточным маршрутизатором) для управления скоростью отправки сообщений отправителем.

## *Формат ICMP-пакета*

| **Формат ICMP-пакета** | | | |
|---|---|---|---|
| **Бит** | **0—7** | **8—15** | **16—31** |
| **0** | Тип | Код | Контрольная сумма |
| **32** | Содержание сообщения (зависит от значений полей «Код» и «Тип») | | |

### Типы ICMP пакетов (полный список)

- **0** — Эхо-ответ
- **1** — Зарезервировано
- **2** — Зарезервировано
- **3** — Адресат недоступен

код 0 — Сеть недостижима
код 1 — Хост недостижим
код 2 — Протокол недостижим
код 3 — Порт недостижим
код 4 — Необходима фрагментация, но установлен флаг ее запрета (DF)
код 5 — Неверный маршрут от источника
код 6 — Сеть назначения неизвестна
код 7 — Хост назначения неизвестен
код 8 — Хост источник изолирован
код 9 — Сеть административно запрещена
код 10 — Хост административно запрещен
код 11 — Сеть недоступна для TOS
код 12 — Хост недоступен для TOS
код 13 — Коммуникации административно запрещены

- **4** — Сдерживание источника (отключение источника при переполнении очереди)
- **5** — Перенаправление

код 0 — Перенаправление пакетов в сеть
Код 1 — Перенаправление пакетов к хосту

Код 2 — Перенаправление для каждого типа обслуживания (TOS)
Код 3 — Перенаправление пакета к хосту для каждого типа обслуживания

- **6 —** Альтернативный адрес хоста
- **7 —** Зарезервировано
- **8 —** Эхо-запрос
- **9 —** Объявление маршрутизатора (RFC-1256)
- **10 —** Запрос маршрутизатора (RFC-1256)
- **11 —** Превышение временного интервала (для дейтаграммы время жизни истекло)

код 0 — Время жизни пакета (TTL) истекло при транспортировке
код 1 — Время жизни пакета (время сборки фрагментов) истекло при дефрагментации

- **12 —** Неверный параметр (проблема с параметрами дейтаграммы: ошибка в IP-заголовке или отсутствует необходимая опция)

код 0 — Указатель говорит об ошибке
код 1 — Отсутствует требуемая опция
код 2 — Некорректная длина

- **13 —** Запрос метки времени
- **14 —** Ответ с меткой времени
- **15 —** Информационный запрос
- **16 —** Информационный ответ
- **17 —** Запрос адресной маски (RFC-950)
- **18 —** Отклик на запрос адресной маски (RFC-950)
- **19 —** Зарезервировано (для обеспечения безопасности)
- **20-29 —** Зарезервировано (для экспериментов на устойчивость к ошибкам)
- **30 —** Трассировка маршрута (RFC-1393)
- **31 —** Ошибка преобразования дейтаграммы (RFC-1475)
- **32 —** Перенаправление для мобильного хоста
- **33 —** IPv6 Where-Are-You (где вы находитесь)
- **34 —** IPv6 I-Am-Here (я здесь)
- **35 —** Запрос перенаправления для мобильного хоста
- **36 —** Отклик на запрос перенаправления для мобильного хоста
- **37 —** Запрос доменного имени (Domain Name Request)
- **38 —** Ответ на запрос доменного имени (Domain Name Reply)
- **39 —** SKIP
- **39 —** Photuris

код 0 — Зарезервировано
код 1 — Неизвестный индекс параметров безопасности (Unkown Security Parameters Index)
код 2 — Параметры безопасности верны, но произошла ошибка аутентификации (Valid Security Parameters, but Authentication Failed)
код 3 — Параметры безопасности верны, но произошел сбой при дешифровке (Valid Security Parameters, but Decryption Failed)

- **41-255 —** Зарезервировано

# Протокол передачи команд и сообщений об ошибках (ICMP). Протоколы DCCP и TFRC

Протокол передачи команд и сообщений об ошибках (ICMP - internet control message protocol, RFC-792, - 1256) выполняет различные и не только диагностические функции. При пересылке пакетов промежуточные узлы не информируются о возникших проблемах, поэтому ошибка в маршрутной таблице может восприниматься как неисправность в узле адресата и достоверно диагностироваться не будет. ICMP-протокол сообщает об ошибках в IP-дейтаграммах, но не дает информации об ошибках в самих ICMP-сообщениях. ICMP использует IP, а IP-протокол должен использовать ICMP. В случае ICMP-фрагментации сообщение об ошибке будет выдано только один раз на дейтаграмму, даже если ошибки были в нескольких фрагментах. Подводя итоги, можно сказать, что ICMP-протокол:

- осуществляет передачу запросов и откликов;
- контролирует время жизни дейтаграмм в системе;
- реализует переадресацию пакета;
- выдает сообщения о недостижимости адресата или о некорректности параметров;
- формирует и пересылает временные метки;
- выдает запросы и отклики для адресных масок и другую информацию.

ICMP-сообщения об ошибках никогда не выдаются:

- в ответ на ICMP-сообщение об ошибке;
- при мультикастинге или широковещательной адресации;
- для фрагмента дейтаграммы (кроме первого);
- для дейтаграмм, чей адрес отправителя является нулевым, широковещательным или мультикастинговым.

Эти правила призваны блокировать потоки дейтаграмм, посылаемые в отклик на мультикастинг или широковещательные ICMP-сообщения. Особенности протокола ICMPv6 описаны в документе RFC-2463. В IPv6 на протокол ICMP возложены функции управления группами (IGMP).

ICMP-сообщения имеют свой формат, а схема их вложения аналогична UDP или TCP и представлена на рис. 3.1.

Все ICMP пакеты начинаются с 8-битного поля типа ICMP и его кода (15 значений).

**Рис. 3.1.** Схема вложения ICMP-пакетов в Ethernet-кадр

По существу, значения полей типа и кода выполняют почти ту же функцию, что и порт в TCP и UDP-протоколах.

Код уточняет функцию ICMP-сообщения. Таблица 3.1 этих кодов приведена ниже (символом * помечены сообщения об ошибках, остальные являются запросами).

| Таблица 3.1. Типы и коды ICMP-сообщений | | |
|---|---|---|
| **ICMP-сообщение** | | **описание сообщения** |
| **Тип** | **код** | |
| 0 | | Эхо-ответ (ping-отклик) |
| 3 | | Адресат недостижим |
| | 0 | * Сеть недостижима |
| | 1 | * ЭВМ недостижима |
| | 2 | * Протокол недоступен |
| | 3 | * Порт недоступен |
| | 4 | * Необходима фрагментация сообщения (установлен флаг DF) |
| | 5 | * Маршрутизация отправителя нереализуема |
| | 6 | * Сеть места назначения неизвестна |
| | 7 | * ЭВМ места назначения неизвестна |
| | 8 | * Исходная ЭВМ изолирована (устарело) |
| | 9 | * Связь с сетью места назначения административно запрещена |
| | 10 | * Связь с ЭВМ места назначения административно запрещена |
| | 11 | * Сеть не доступна для данного вида сервиса (TOS) |
| | 12 | * ЭВМ не доступна для данного вида сервиса (TOS) |
| | 13 | * Связь административно запрещена с помощью фильтра |
| | 14 | * Нарушение старшинства ЭВМ |
| | 15 | * Дискриминация по старшинству |
| 4 | 0 | * Отключение источника при переполнении очереди (quench) |
| 5 | | Переадресовать (изменить маршрут) |
| | 0 | Переадресовать дейтаграмму в сеть (устарело) |
| | 1 | Переадресовать дейтаграмму на ЭВМ |
| | 2 | Переадресовать дейтаграмму для типа сервиса (ToS) и сети |
| | 3 | Переадресовать дейтаграмму для типа сервиса и ЭВМ |
| 8 | 0 | Эхо запроса (ping-запрос) |
| 9 | 0 | Объявление маршрутизатора |
| 10 | 0 | Запрос маршрутизатора |
| 11 | | Для дейтаграммы время жизни истекло (ttl=0): |
| | 0 | *при передаче |
| | 1 | * тайм-аут при сборке (случай фрагментации) |
| 12 | | * Проблема с параметрами дейтаграммы |
| | 0 | * Ошибка в IP-заголовке |
| | 1 | * Отсутствует необходимая опция |
| 13 | | Запрос временной метки |
| 14 | | Временная метка-отклик |
| 15 | | Запрос информации (устарел) |
| 16 | | Информационный отклик (устарел) |
| 17 | | Запрос адресной маски |
| 18 | | Отклик на запрос адресной маски |

Процедура "отключение источника" (quench, поле тип ICMP=4) позволяет управлять потоками данных в каналах Интернет. Не справляясь с обработкой дейтаграмм, ЭВМ-адресат (или транзитное сетевое устройство) может послать запрос "отключить источник" отправителю, который может сократить темп посылки пакетов, например, вдвое, или вовсе прервать их посылку. Специальной команды, отменяющей прежние запреты, не существует. Если сообщения об отключении прекращаются, источник может возобновить посылку пакетов или увеличить частоту их отправки. Ниже (на рис. 3.2) представлен формат эхо запроса (ping) и отклика для протокола ICMP.

Поля идентификатор (обычно это идентификатор процесса) и номер по порядку (увеличивается на 1 при посылке каждого пакета) служат для того, чтобы отправитель мог связать в пары запросы и отклики.

**Рис. 3.2.** Формат эхо-запроса и отклика ICMP

Поле тип определяет, является ли этот пакет запросом (8) или откликом (0). Поле контрольная сумма представляет собой 16-разрядное дополнение по модулю 1 контрольной суммы всего ICMP-сообщения, начиная с поля тип. Поле данные служит для записи информации, возвращаемой отправителю. При выполнении процедуры ping эхо-запрос с временной меткой в поле данные посылается адресату. Если адресат активен, он принимает IP-пакет, меняет адрес отправителя и получателя местами и посылает его обратно. ЭВМ-отправитель, восприняв этот отклик, может сравнить временную метку, записанную им в пакет, с текущим показанием внутренних часов и определить время распространения пакета (RTT - round trip time). Размер поля данные не регламентирован и определяется предельным размером IP-пакета.

**Рис. 3.2а.**

Так как в пакете ICMP нет поля порт, то при запуске нескольких процессов PING одновременно может возникнуть проблема с тем, какому из процессов следует передать тот или иной отклик. Для преодоления этой неопределенности следует использовать уникальные значения полей идентификатор.

Время распространения ICMP-запроса, вообще говоря, не равно времени распространения отклика (ситуация с очередями, например, в маршрутизаторах может быть разной). Это связано не только с возможными изменениями в канале. В общем случае маршруты их движения могут быть различными. Те же запросы используются при выполнении процедуры Traceroute.

Эта процедура позволяет не только диагностировать, но и оптимизировать маршрут. Например, команда traceroute cernvm.cern.ch, выданная в ЭВМ SUN (ИТЭФ), может отобразить на экране (в скобках указаны IP-адреса узлов и значения времени жизни дейтаграмм, значения RTT приводится в миллисекундах) (таблица 3.1.1.):

Таблица 3.1.1.

| **traceroute to crnvma cern ch** | | **(128 141 2 4) 30 hops max, 40 byte packets** |
|---|---|---|
| 1 | itep-fddi-bbone | (193.124.224.50) 3 ms 2 ms 3 ms |
| 2 | msu-tower.moscow.ru.radio-msu.net | (193.124.137.13) 3 ms 3 ms 3 ms |
| 3 | npi-msu.moscow.ru.radio-msu.net | (193.124.137.9) 27 ms 3 ms 9 ms |
| 4 | desy.hamburg.de.radio-msu.net | (193.124.137.6) 556 ms 535 ms 535 ms |
| 5 | * 188.1.133.56 | (188.1.133.56) 637ms 670ms |
| 6 | duesseldorf2.empb.net | (193.172.4.12) 740ms(ttl=59!) 839ms(ttl=59!) 2066ms(ttl=59!) |
| 7 | bern3.empb.net | (193.172.4.30) 2135ms (ttl=58!) 1644ms (ttl=58!) 1409ms (ttl=58!) |
| 8 | cernh3.euro-hep.net | (193.172.24.10) 1808ms 1508ms 1830ms |
| 9 | cgate1.cern.ch | (192.65.185.1) 1116ms 1270ms 1278ms |

| | | |
|---|---|---|
| 10 | crnvma.cern.ch | (128.141.2.4) 1132ms 1362ms 1524ms |

Отсюда видно, что наиболее узкими участками маршрута (на момент выполнения процедуры) являются Гамбург-Дюссельдорф-Берн-CERN. Следует иметь в виду, что канал МГУ-Гамбург был спутниковым и 500мс задержки здесь вносит удвоенное время распространения сигнала до спутника и обратно. Участок Гамбург-Дюссельдорф (X.25, квота 256кбит/с) был общим для потока данных из Германии в США. Передача IP поверх X.25 также снижает эффективную широкополосность канала. Остальные связи обладают недостаточной пропускной способностью. Программа ping показывает для данных участков в часы пик высокую долю потерянных пакетов. Таким образом, имея карту связей и используя указанные процедуры, вы можете успешно диагностировать ситуацию в сети. Продвинутые же программисты могут легко написать свои диагностические программы, базирующиеся на ICMP, как для локальной сети, так и для "окрестного" Интернет.

Когда маршрутизатор не может доставить дейтаграмму по месту назначения, он посылает отправителю сообщение "адресат недостижим" (destination unreachable). Формат такого сообщения показан ниже (на рис. 3.3).

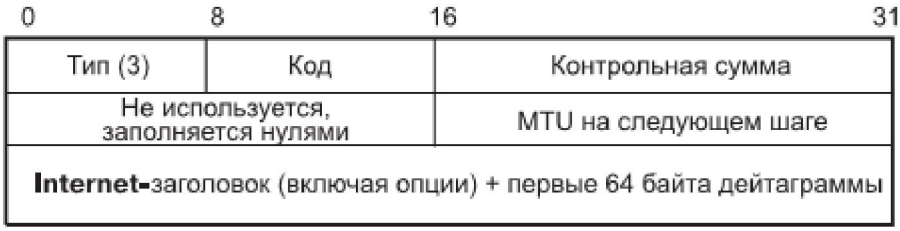



**Рис. 3.3.** Формат ICMP-сообщения "адресат недостижим"

Поле код содержит целое число, проясняющее положение дел. Перечень кодов таких сообщений помещен в таблице 3.1. Поле MTU на следующем этапе характеризует максимальную длину пакетов на очередном шаге пересылки. Эта информация (MTU) позволяет выбрать оптимальный размер пакета.

Так как в сообщении содержится Интернет-заголовок и первые 64-бита дейтаграммы, легко понять, какой адрес оказался недостижим. Этот тип ICMP-сообщения посылается и в случае, когда дейтаграмма имеет флаг DF=1 ("не фрагментировать"), а фрагментация необходима. В поле код в этом случае будет записано число 4.

Если буфер приема сообщения переполнен, ЭВМ посылает сообщение типа 4.

Так как собственные часы различных ЭВМ имеют свое представление о времени, протокол ICMP, среди прочего, служит для синхронизации работы различных узлов, если это требуется (запросы временных меток). Протокол синхронизации NTP (network time protocol) описан в RFC-1119.

Когда дейтаграммы поступают слишком часто и принимающая сторона не справляется с этим потоком, отправителю может быть послано сообщение с требованием резко сократить информационный поток (quench-запрос). Снижение потока должно продолжаться до тех пор, пока не прекратятся quench-запросы. Такое сообщение имеет формат, показанный на рис. 3.4.

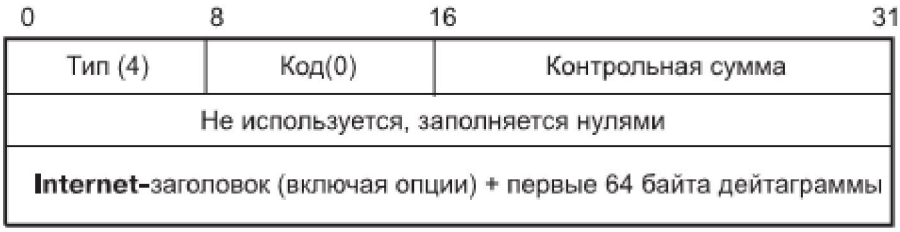



**Рис. 3.4.** Формат ICMP-запроса снижения загрузки

В Internet таблицы маршрутизации остаются без изменений достаточно долгое время, но иногда они все же меняются. Если маршрутизатор обнаружит, что ЭВМ использует неоптимальный маршрут, он может послать ей ICMP-запрос переадресации. При этом следует учитывать, что этой возможностью может воспользоваться и хакер. Формат такого сообщения отображен на рисунке 3.5.

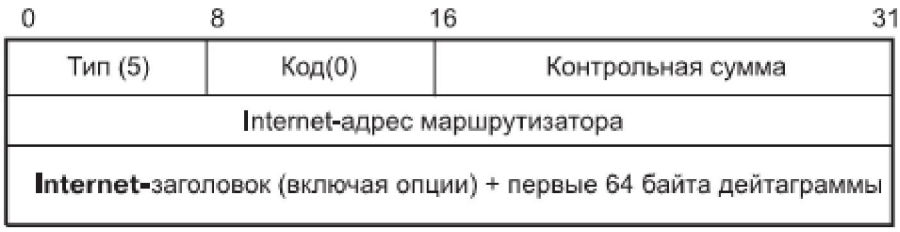



**Рис. 3.5.** Формат ICMP-запроса переадресации

Поле Internet-адрес маршрутизатора содержит адрес маршрутизатора, который ЭВМ должна использовать, чтобы посланная дейтаграмма достигла места назначения, указанного в ее заголовке. В поле internet-заголовок кроме самого заголовка лежит 64 первых бита дейтаграммы, вызвавшей это сообщение. Поле код специфицирует то, как нужно интерпретировать адрес места назначения (см. табл. 3.1).

Команды переадресации маршрутизатор посылает только ЭВМ и никогда другим маршрутизаторам. Рассмотрим конкретный пример. Пусть некоторая ЭВМ на основе своей маршрутной таблицы посылает пакет маршрутизатору М1. Он, просмотрев свою маршрутную таблицу, находит, что пакет следует переслать маршрутизатору М2. Причем выясняется, что пакет из М1 в М2 следует послать через тот же интерфейс, через который он попал в М1. Это означает, что М2 доступен и непосредственно для ЭВМ-отправителя. В такой ситуации М1 посылает ICMP-запрос переадресации ЭВМ-отправителю указанного пакета с тем, чтобы она внесла соответствующие коррективы в свою маршрутную таблицу (либо чтобы администратор поменял IP-адрес или маску ЭВМ).

Маршрутная таблица может загружаться из файла, формироваться менеджером сети, но может создаваться и в результате запросов и объявлений, посылаемых маршрутизаторами. После загрузки системы маршрутизатор посылает широковещательный запрос. Один или более маршрутизаторов посылают в ответ сообщения об имеющейся маршрутной информации. Кроме того, маршрутизаторы периодически (раз в 450600 сек.) широковещательно объявляют о своих маршрутах, что позволяет другим маршрутизаторам скорректировать свои маршрутные таблицы. В RFC-1256 описаны форматы ICMP-сообщений такого рода (см. рис. 3.6). Из-за своей уязвимости данная технология в последнее время практически не используется.

**Рис. 3.6.** Формат ICMP-сообщений об имеющихся маршрутах

Поле число адресов характеризует количество адресных записей в сообщении. Поле длина адреса содержит число 32-битных слов, необходимых для описания адреса маршрутизатора. Поле время жизни предназначено для записи продолжительности жизни объявленных маршрутов (адресов) в секундах. По умолчанию время жизни равно 30 мин. Поля уровень приоритета представляют собой меру приоритетности маршрута по отношению к другим маршрутам данной подсети. Чем больше

этот код, тем выше приоритет. Маршрут по умолчанию имеет уровень приоритета 0. Формат запроса маршрутной информации (8 октетов, рис. 3.7).

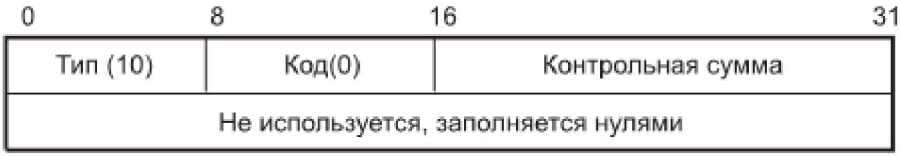



**Рис. 3.7.** Формат запроса маршрутной информации

Если вы работаете с ОС Windows, а конфигурационными сетевыми параметрами являются IP-адрес вашей ЭВМ, сетевая маска и адреса GW и DNS, то процедуры переадресации, скорее всего к вам отношения не имеют.

Так как следующий прогон (hop) дейтаграммы определяется на основании локальной маршрутной таблицы, ошибки в последней могут привести к зацикливанию пакетов. Причиной зацикливания могут быть и многие другие причины. Для подавления таких циркуляций используется контроль по времени жизни пакета (TTL). При ликвидации пакета по истечении TTL маршрутизатор посылает отправителю сообщение "время истекло", которое имеет формат (рис. 3.8):

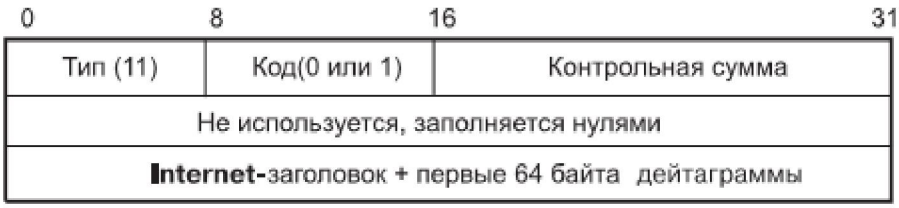



**Рис. 3.8.** Формат сообщения "время (TTL) истекло"

Значение поля код определяет природу таймаута (см. табл. 3.1).

Когда маршрутизатор или ЭВМ выявили какую-либо ошибку, не из числа описанных выше (например, нелегальный заголовок дейтаграммы), посылается сообщение "конфликт параметров". Это может произойти, например, при неверных параметрах опций. В этом случае посылается сообщение вида (рис. 3.9):

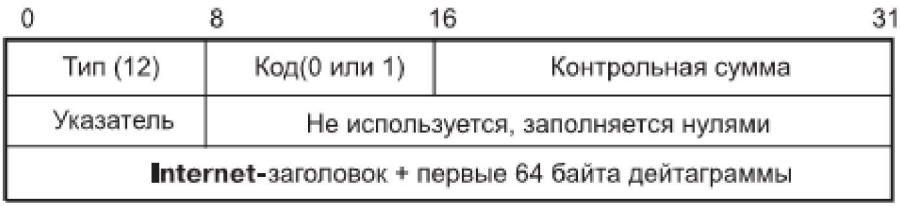



**Рис. 3.9.** Формат сообщения типа "конфликт параметров"

Поле указатель отмечает октет дейтаграммы, который создал проблему. Код=1 применяется для сообщения о том, что отсутствует требуемая опция (например, опция безопасности при конфиденциальных обменах); поле указатель при значении поля код=1 не используется.

В процессе трассировки маршрутов возникает проблема синхронизации работы часов в различных ЭВМ. К счастью, синхронизация внутренних часов ЭВМ требуется не так часто (например, при выполнении синхронных измерений), негативную роль здесь могут играть задержки в каналах связи. Для запроса временной метки другой ЭВМ используется сообщение запрос временной метки, которое вызывает отклик с форматом (рис. 3.10):

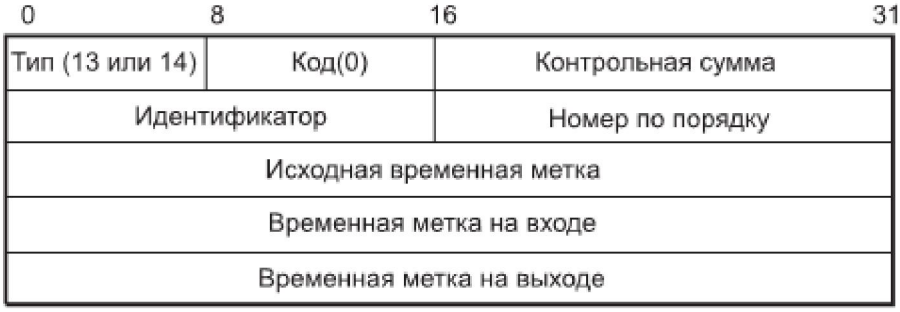



**Рис. 3.10.** Формат ICMP-запроса временной метки

Поле тип=13 указывает на то, что это запрос, а тип=14 - на то, что это отклик. Поле идентификатор и номер по порядку используются отправителем для связи запроса и отклика. Поле исходная временная метка заполняется отправителем непосредственно перед отправкой пакета. Поле временная метка на входе заполняется маршрутизатором при получении этого пакета, а временная метка на выходе - непосредственно перед его отправкой.

При работе с субсетью важно знать маску этой субсети. Как уже отмечалось выше, IP-адрес содержит секцию адреса ЭВМ и секцию адреса сети. Для получения маски субсети ЭВМ может послать "запрос маски" в маршрутизатор и получить отклик, содержащий эту маску. ЭВМ может это сделать непосредственно, если ей известен адрес маршрутизатора, либо прибегнуть к широковещательному запросу. Ниже описан формат таких запросов-откликов (рис. 3.11).

**Рис. 3.11.** Формат запроса (отклика) маски субсети

Поле тип здесь специфицирует модификацию сообщения, тип=17 - это запрос, а тип=18 - отклик. Поля идентификатор и номер по порядку, как обычно, обеспечивают взаимную привязку запроса и отклика, а поле адресная маска содержит 32-разрядную маску сети.

Процесс интеграции услуг в Интернет несколько лет назад перешел в практическую плоскость и вслед за IP-телефонией, настала очередь цифрового телевидения. Главным транспортным средством для мультимедиа-данных является протокол UDP (RTP/RTCP). Но в протоколе UDP нет встроенных средств для подавления перегрузки. Возможность использования ICMP(4) трудно рассматривать всерьез, так как этот механизм достаточно груб и включается тогда, когда перегрузка уже привела к потере пакетов. Решить проблему управления перегрузкой в потоках без гарантии доставки должен новый протокол DCCP.

## 3.1. Протокол управления перегрузкой для дейтаграмм DCCP

### *Введение*

Протокол DCCP (Datagram Congestion Control Protocol) является транспортным протоколом, который использует двунаправленные уникастные соединения с управлением перегрузкой для ненадежной доставки дейтаграмм. Протокол DCCP предназначен для приложений, которые реализуют поточную схему TCP, но имеют приоритет для своевременной доставки данных с сохранением порядка кадров или требуют надежности, или для приложений, которые требуют механизма подавления перегрузки, отличного от TCP. До настоящего времени такие приложения использовали либо TCP, чья надежность и гарантия упорядочения доставки давалась за счет неопределенно большой задержки, либо UDP и независимый механизм управления перегрузкой (или вообще с отсутствием подавления пере-

грузки). Протокол DCCP будет предоставлять стандартный способ управления перегрузкой и позволит применять механизм ECN.

Протокол DCCP предназначен для приложений, которые не требуют параметров SCTP [RFC-2960], таких, как упорядоченная доставка при нескольких потоках.

Протокол UDP исключает большие задержки, но приложения UDP, которые используют управление перегрузкой, вынуждены вносить задержки сами. Протокол DCCP имеет встроенную систему управления перегрузкой, включая поддержку ECN для ненадежных потоков дейтаграмм и исключая непредсказуемые задержки, характерные для TCP. Протокол DCCP обеспечивает надежное согласование параметров при установлении соединения.

Одной из целей DCCP было максимальное облегчение для UDP приложений перехода на DCCP, когда он будет внедрен. Чтобы облегчить это, DCCP был спроектирован с минимальной избыточностью как с точки зрения размера заголовка пакета, так и с позиции загрузки ЦПУ машин партнеров. В DCCP была включена минимальная функциональность, при сохранении возможности включения новых функций, таких, как FEC (forward error correction), псевдонадежность и множественные потоки, которые могут быть добавлены поверх DCCP, если потребуется.

Возможны разные механизмы управления перегрузкой для разных приложений. Например, игры реального времени могут требовать быстрого использования всей доступной полосы пропускания, в то время как потоковая среда может использовать компромисс между скоростью отклика и стабильной, менее импульсивной передачей. (Резкое изменение потока может вызвать неприемлемые сбои UI, такие, как слышимые паузы или щелчки). Протокол DCCP, таким образом, предлагает приложению выбор одного из нескольких механизмов управления перегрузкой. Одной из альтернатив является TCP-подобное управление перегрузкой, сокращение вдвое окна перегрузки в ответ на потерю пакета. Приложения, применяющие механизм управления перегрузкой, будут быстро реагировать на изменения доступной полосы, но должны выдерживать резкое изменение окна перегрузки, что типично для TCP. Вторую альтернативу представляет механизм управления скоростью передачи, дружественный по отношению TCP (TFRC) [RFC-3448]. Это алгоритм управления перегрузкой, базирующийся на ура внении, минимизирует резкие изменения скорости передачи.

Протокол DCCP позволяет также ненадежному трафику без проблем использовать технику ECN. Ядро UDP API не может позволить приложениям считать UDP пакеты адаптированными к ECN, так как API не может гарантировать, что приложение способно корректно детектировать или реагировать на перегрузку. Ядро DCCP API лишено такого недостатка, так как имеет встроенный механизм управления перегрузкой.

В DCCP было решено не применять менеджер управления перегрузкой [RFC-3124], который допускает несколько одновременных потоков между отправителем и получателем. Менеджер перегрузки может быть использован только приложениями, которые имеют свою собственную обратную связь в случае потери пакетов между конечными точками соединения, но этого нет во многих приложениях, работающих с UDP. Кроме того, сегодняшний вариант менеджера перегрузки, как правило, не поддерживает более одного механизма управления перегрузкой. DCCP должен быть способен использовать механизм управления перегрузкой там, где это требуется приложению.

Предполагается, что протокольные механизмы DCCP, будут адаптированы к любому приложению, требующему уникастных потоков с управлением перегрузки. Механизмы управления перегрузкой, встраиваемые в DCCP, описаны в отдельных ID профайлах управления [CCID 2 PROFILE, CCID 3 PROFILE], они могут, однако вызвать проблемы для некоторых приложений, включая широкополосное интерактивное видео. Эти приложения должны быть способны задействовать DCCP, когда будут стандартизованы подходящие ID профайлы управления перегрузкой.

Протокол DCCP имеет следующие характеристики:

- реализует поток дейтаграмм с подтверждением получения, но без повторной посылки;
- ненадежный диалог установления и разрыва соединения;
- надежное согласование параметров;
- выбор механизмов подавления перегрузки, дружественных по отношению к TCP, включая TCP-подобное управление перегрузкой (CCID 2) и управление потоком [RFC-3448] (CCID 3). CCID 2 использует разновидность TCP-механизма управления перегрузкой и приемлемо для потоков, которые стремятся воспользоваться преимуществами доступной полосы, CCID 3 пригодно для потоков, которые требуют более стабильной скорости передачи;
- опции, которые говорят отправителю с высокой надежностью, какие пакеты достигли получателя, были ли эти пакеты помечены ECN [RFC-3168 и RFC-3540], повреждены или отброшены во входном буфере получателя;
- управление перегрузкой, снабженное встроенной индикацией явной перегрузки ECN (Explicit Congestion Notification);
- механизмы, позволяющие серверу избежать поддержки состояний неподтвержденных попыток соединений;
- выявление MTU пути [RFC-1191].

### *Основные отличия DCCP от TCP*

Ниже рассмотрены наиболее существенные отличия DCCP от TCP.

- Поток пакетов. DCCP является протоколом для потоков пакетов, а не потоков байт.
- Ненадежность. DCCP никогда не пересылает дейтаграммы повторно.
- Порядковые номера пакетов. Порядковые номера относятся к пакетам, а не к байтам. Каждому пакету, посылаемому DCCP, присваивается новый порядковый номер, то же и с пакетами подтверждений. Это позволяет получателю пакетов DCCP детектировать потери подтверждений; смотри раздел "Sequence Number Validity" в [DCCP].
- Обширное пространство для опций (до 1020 байт).
- Согласование параметров. Это является базовым механизмом, с помощью которого партнеры согласуют значения параметров или свойства соединения.
- Выбор управления перегрузкой. Партнеры могут использовать разные механизмы управления перегрузкой. В соединении A<>B информационные пакеты, посланные от A>B, могут применять алгоритм CCID 2, а пакеты данных от B>A могут использовать CCID 3.
- Различные форматы подтверждений. CCID соединения определяет, какой объем данных должно быть передан в ack. В CCID 2 (TCP-подобном) посылается один ack на 2 пакета, а каждый ack должен оповещать, какие конкретно пакеты получены (опция Ack Vector); в CCID 3 (TFRC) посылается в среднем один ack за время RTT, а ack должны сообщать как минимум о длинах последних интервалов потерь.
- Отсутствие приемного окна. DCCP является протоколом управления перегрузкой, а не протоколом управления потоками.
- Разделение различных видов потерь. Опция потерянных данных (Data Dropped) позволяет одному из партнеров сообщить, что пакет был потерян из-за повреждения, переполнения входного буфера и т.д..
- Определение подтверждения. В TCP получение пакетов подтверждается, только когда они ставятся в очередь для передачи приложению. В протоколе DCCP это делается не так. Получение пакета подтверждается, когда обработаны поля его опций.
- Встроенная поддержка мобильности.

### *Основные понятия и термины*

Каждое DCCP соединение реализуется между двумя партнерами, которые будут называться DCCP A и DCCP B. Данные могут передаваться в обоих направлениях. В рамках протокола рассматриваются субнаборы соединений, называемых полусоединениями (halfconnection - HC). (Смотри рис. 3.12.) Эти полусоединения обеспечивают передачу информационных пакетов в одном направлении плюс соответствующих подтверждений в противоположном направлении. В контексте одного полусоединения HC-отправитель является партнером отправителем данных, в то время как HC-Receiver является партнером, посылающим подтверждения. Каждое полусоединение контролируется механизмом управления перегрузкой, специфицированным однобайтовым идентификатором, или CCID. Партне-

ры согласуют режим обмена на фазе формирования соединения. CCID для полусоединения описывает, как НСотправитель ограничивает поток информационных пакетов; как он поддерживает необходимые значения параметров, такие, как окна перегрузки; как НС-получатель уведомляет отправителя о перегрузке, посылая подтверждения; и как он поддерживает частоту подтверждений.

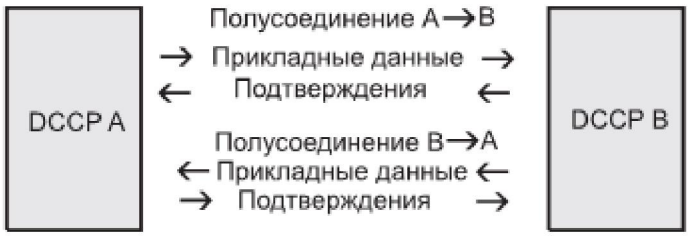



**Рис. 3.12.** Структура обменов в протоколе DCCP

**Инициация и разрыв соединения**

Любое соединение DCCP предполагает формирование DCCP сокетов в пассивном состоянии прослушивания. Активной стороной считается клиент, а роль пассивной стороны выполняет сервер.

**Характеристики**

DCCP имеет встроенный механизм согласования параметров соединения, такой, как активные CCID для обоих полусоединений (см. рис. 3.12). Опции Change, Prefer и Confirm служат для согласования параметров. Change посылается с целью изменения определенного параметра. Адресат может откликнуться, послав Prefer, где попросит партнера заменить значение параметра, или может изменить значение параметра и послать подтверждение Confirm. Надежность установления параметров обеспечивается повторной передачей, если это потребуется.

## *Пакеты DCCP*

Протокол DCCP использует девять различных типов пакетов: DCCP-Request, DCCP-Response, DCCP-Data, DCCP-Ack, DCCP-DataAck, DCCP-CloseReq, DCCP-Close, DCCP-Reset, и DCCP-Move. Длина пакета DCCP варьируется в пределах от 12 до 1020 байт. Алгоритм формирования и разрыва DCCP соединения подразумевает следующие обмены.

DCCP-Request - Посылается клиентом для инициализации соединения (первый шаг трехходовой процедуры).

DCCP-Response - Посылается сервером в ответ на запрос DCCPR-equest (второй этап трехшаговой процедуры инициации соединения).

DCCP-Data - Используется для передачи прикладных данных.

DCCP-Ack - Используется для передачи только подтверждений.

DCCP-DataAck - Используется для передачи прикладных данных в сочетании с подтверждениями.

DCCP-CloseReq - Посылается сервером для предложения клиенту закрыть соединение.

DCCP-Close - Используется клиентом или сервером для закрытия соединения; также для запуска процедуры сброса (DCCP-Reset).

DCCP-Reset - Используется для закрытия соединения, обычным или необычным образом.

DCCP-Sync, DCCP-SyncAck - Используется для ресинхронизации номеров пакетов после длительного периода потерь.

Процедура взаимодействия партнеров включает в себя следующие обмены.

1. Клиент посылает серверу пакет DCCP-Request, специфицируя номера портов клиента и сервера, запрашиваемый сервис, и любые параметры, подлежащие согласованию, включая CCID, который должен использовать сервер при работе с клиентом. Клиент может опционно подключать к пакету DCCP-Request некоторые данные.
2. Сервер посылает клиенту пакет-отклик, уведомляя его о своем желании осуществить обмен данными. В отклике указываются любые особенности и опции, которые предлагает сервер. Опционно отклик может содержать Init Cookie.
3. Клиент посылает серверу пакет DCCP-Ack, который подтверждает получение пакета DCCP-отклика.
4. Далее следуют обмен нулем или более подтверждениями DCCP-Ack, если требуется завершить согласование используемых параметров.
5. Сервер и клиент обмениваются пакетами DCCP-Data, DCCP-Ack. Если у клиента нет данных, тогда сервер посылает пакеты DCCP-Data и DCCP-DataAck, в то время как клиент будет посылать только DCCP-Acks.
6. Сервер посылает пакет DCCP-CloseReq, запрашивая закрытие соединения.
7. Клиент посылает пакет DCCP-Close, подтверждая закрытие.
8. Сервер посылает пакет DCCP-Reset, в котором поле причина содержит "Closed", при этом состояние соединения ликвидируется.
9. Клиент получает пакет DCCP-Reset и сохраняет свое состояние в течение некоторого времени для завершения происходящих обменов.

## Форматы пакетов

### *Базовый заголовок пакета*

Все DCCP пакеты имеют базовый заголовок с форматом, показанным на рис. 3.13.

**Рис. 3.13.** Базовый заголовок пакетов DCCP

Если X равно нулю, передаются только младшие (LSB) 24 бита порядкового номера, а базовый заголовок имеет длину 12 байт. Ниже поясняется назначение полей базового заголовка.

Порт отправителя и получателя (по 16 бит каждый)

Эти поля идентифицируют соединение, аналогично соответствующим полям TCP и UDP. Порт отправителя характеризует порт, через который послан данный пакет, а порт назначения характеризует процесс, которому должен быть этот пакет доставлен. Когда соединение формируется, клиент должен выбрать порт отправителя случайным образом, чтобы уменьшить вероятность атаки.

Прикладные интерфейсы DCCP должны воспринимать номера портов, так же, как это делается в случае TCP и UDP. Например, машины, которые различают привилегированные и непривилегированные порты для TCP и UDP, должны делать то же самое и для DCCP.

Смещение данных (8 бит)

Смещение от начала заголовка пакета DCCP первого октета прикладных данных (выражается в 32-битных словах). Получатель должен игнорировать пакеты, где значение поля Data Offset (смещение данных) меньше минимального размера заголовка для данного типа или больше размера самого пакета DCCP.

CCVal (4 бита)

Используется HCотправителем CCID. Например, CCID отправителя A-B, который активен в DCCP A, может послать 4 бита данных с каждым пакетом получателю, записав их в поле CCVal. Отправитель должен установить CCVal равным нулю, если только HCотправитель CCID не требует иного, а получатель должен игнорировать поле CCVal, если HCполучатель CCID не специфицирует обратного.

Checksum Coverage (CsCov - покрытие контрольным суммированием): 4 бита.

Поле CsCov определяет части пакета, которые покрываются полем контрольная сумма. Она всегда покрывает заголовок и опции DCCP, но некоторые приложения могут допускать исключения. Это может улучшить работу протокола в условиях каналов с высоким уровнем шума.

Контрольная сумма (16 бит)

Интернет контрольная сумма заголовка пакета DCCP (включая опции), псевдозаголовок сетевого уровня и, в зависимости от CsCov, все, некоторые или никакие поля данных приложений.

Зарезервировано (Res) (3 бита)

Отправитель должен записать в это поле все нули, а получатель должен это поле игнорировать.

Тип (4 бита)

Поле тип специфицирует тип пакета. Определены следующие типы пакетов:

Таблица 3.2. Типы пакетов DCCP

| **Тип** | **назначение** |
|---|---|
| 0 | DCCP-Request (запрос) |
| 1 | DCCP-Response (отклик) |
| 2 | DCCP-Data (данные) |
| 3 | DCCP-Ack (подтверждение) |
| 4 | DCCP-DataAck (подтверждение данных) |
| 5 | DCCP-CloseReq (запрос закрытия) |
| 6 | DCCP-Close (закрытие) |

| | |
|---|---|
| 7 | DCCP-Reset (сброс) |
| 8 | DCCP-Sync (установление соединения) |
| 9 | DCCP-SyncAck (подтверждение соединения) |
| 10-15 | Зарезервировано |

Адресаты должны игнорировать любые пакеты с зарезервированным типом.

Расширенные порядковые номера (X). 1 бит

Если X=1, в заголовке используются 48-разрядные порядковые номера. Пакеты DCCP-Data, DCCP-DataAck и DCCP-Ack могут иметь значение X, равное 0 или 1. Все пакеты DCCP-Request, DCCP-Response, DCCP-CloseReq, DCCP-Close, DCCP-Reset, DCCP-Sync и DCCP-SyncAck должны иметь X=1; партнеры должны игнорировать любые такие пакеты с X=0. При высокоскоростных соединениях следует использовать для всех пакетов X=1.

Порядковый номер. 48 или 24 бита

Идентифицирует пакет в последовательности. Номер по порядку увеличивается на 1 после посылки каждого пакета, включая пакеты DCCP-Ack, которые не несут в себе прикладных данных.

Каждому механизму управления перегрузкой, поддерживаемому DCCP, присвоен идентификатор, или CCID: номер в диапазоне от 0 до 255. Во время установления соединения (и опционно после этого) партнеры согласуют свои механизмы управления перегрузкой. ID управления перегрузкой имеет код 1. Величина CCID/A равна CCID, используемому для полусоединения $A \rightarrow B$. DCCP B посылает опцию "Change R(CCID, K)", чтобы DCCP A применял для своих информационных пакетов CCID K.

CCID является характеристикой сервера, при согласовании CCID-опций может использоваться список из нескольких приемлемых CCID, записанных в убывающем порядке по приоритету. Например, опция "Change R(CCID, 2 3 4)" просит получателя применять для своих пакетов CCID 2, хотя CCID 3 и 4 также являются приемлемыми. (Это соответствует байтам "35, 6, 1, 2, 3, 4": Change R опция <35>, длина опции <6>, ID код <1>, CCID (2, 3, 4)). Аналогично, "Confirm L(CCID, 1, 2 3 4)" говорит получателю, что отправитель использует для своих пакетов CCID 2, но что CCID 3 и 4 могут быть также приемлемыми. В настоящее время стандартизованы следующие значения CCID:

Таблица 3.3. Идентификаторы управления перегрузкой DCCP

| **CCID** | **значение** |
|---|---|
| 0-1 | Зарезервировано |
| 2 | TCP-подобное управление перегрузкой |
| 3 | Управление перегрузкой TFRC |
| 4-255 | Зарезервировано |

Все определенные в настоящее время типы пакетов, за исключением DCCP-запрос и DCCP-Data, содержат в себе субзаголовок номера подтверждения в четырех или восьми байтах, следующих за базовым заголовком. Когда X=1, он имеет формат, показанный на рис. 3.14.

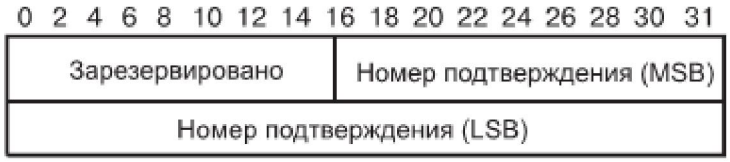



**Рис. 3.14.** Формат подзаголовка номера подтверждения

Клиент инициирует DCCP соединение, посылая пакет запроса. Эти пакеты могут содержать прикладные данные и должны иметь 48-битные порядковые номера (X=1).

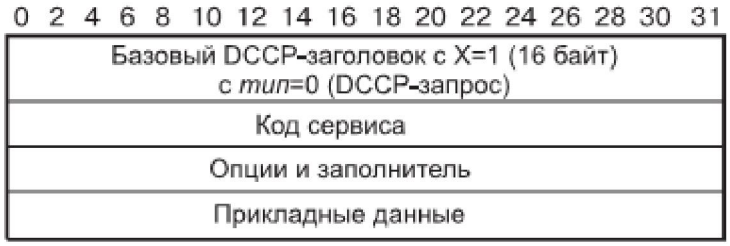



**Рис. 3.15.** Формат пакета DCCP-запроса

Код сервиса: 32 бита

Описывает прикладной уровень сервиса, к которому желает подключиться клиентское приложение. Поле код сервиса предназначено для индикации прикладного протокола, используемого для соединения.

Сервер откликается на корректный DCCP-запрос пакетом DCCP-отклика. Это является второй фазой трехходового диалога. Пакеты DCCP-отклика могут содержать прикладные данные и должны использовать 48-битные порядковые номера (X=1).

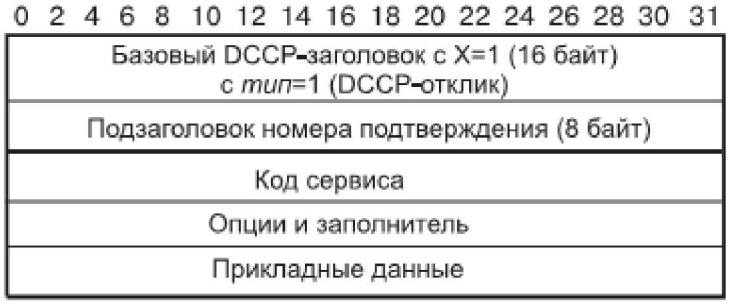



**Рис. 3.16.** Формат пакета DCCP-отклика

Номер подтверждения имеет 48 бит

Содержит GSR (Greatest Sequence Number Received - наибольший полученный порядковый номер). Так как DCCP-отклик посылается только во время установления соединения, он всегда равен порядковому номеру полученного DCCP-запроса.

Код сервиса: 32 бита

Должен равняться коду сервиса, соответствующему DCCP-запросу.

## Форматы пакетов DCCP-Data, DCCP-Ack и DCCP-DataAck

Поле данных DCCP соединения пересылается в пакетах DCCP-Data и DCCP-DataAck, в то время как пакеты DCCP-Ack используются для подтверждений, когда не требуется передавать какой-либо информации. Пакеты DCCP-Ack не содержат данных, но имеют номера подтверждений.

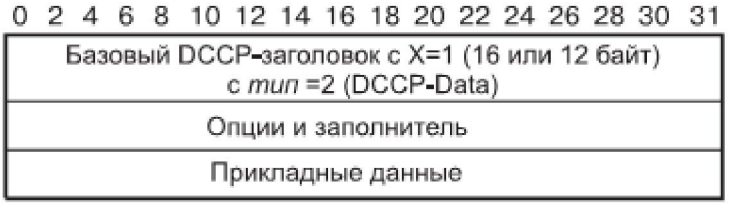



**Рис. 3.17.** Формат пакета DCCP-Data

Пакеты DCCP-Ack и DCCP-DataAck часто содержат дополнительно опции подтверждения, такие, как Ack вектор, которые требуются для используемого механизма управления перегрузкой.

### *Опции и характеристики*

Все пакеты DCCP могут содержать опции, которые располагаются в конце заголовка, их длина кратна 8 битам. Все опции всегда включаются в контрольное суммирование. Опция может начинаться с любого байта. В настоящее время определены следующие опции: таблица 3.4.

Таблица 3.4. Опции

| Тип | Длина [байт] | назначение |
|---|---|---|
| 0 | 1 | Заполнитель |
| 2 | 1 | Медленный получатель |
| 32 | 3-4 | Игнорируется |
| 33 | переменная | Изменение |
| 34 | переменная | Предпочтение |
| 35 | переменная | Подтверждение |
| 36 | переменная | Инициализация Cookie |
| 37 | переменная | Вектор Ack [в данный момент 0] |
| 38 | переменная | Вектор Ack [в данный момент 1] |
| 39 | переменная | Data Dropped |
| 40 | 6 | Временная метка |
| 41 | 6-10 | Отклик на временную метку |
| 42 | переменная | Идентификация |
| 44 | переменная | Вызов |

| | | |
|---|---|---|
| 45 | 4 | Контрольная сумма поля данных |
| 46 | 4-6 | Время работы |
| 128-255 | переменная | Специфические опции CCID |

### *Числовые характеристики*

Первый байт данных каждой опции Change, Prefer или Confirm является числовым параметром, определяющим тип согласуемого параметра. Остальные данные представляют собой одно или более значение параметра и интерпретируются в соответствии с особенностями данного параметра (характеристики). В настоящее время определены следующие параметры (таблица 3.5):

### *Подтверждения*

Управление перегрузкой требует, чтобы получатели передавали отправителям данные о потерях пакетов и метках ECN. Получатели DCCP должны сообщать обо всех замеченных перегрузках в соответствии с профайлом CCID.

### *Сервис Ack Ratio*

Ack Ratio предоставляет общий механизм, с помощью которого CCID, определяющий время подтверждения информационных пакетов, может выполнять упрощенное управление перегрузкой. CCID 2, TCP-подобное управление перегрузкой, использует Ack Ratio, чтобы ограничить частоту подтверждений. Некоторые CCID игнорируют Ack Ratio, выполняя управление перегрузкой для подтверждений каким--то другим способом.

Таблица 3.5.

| **Число** | **назначение** | **нач. знач** | **обязательность** |
|---|---|---|---|
| 0 | Зарезервировано | | |
| 1 | Управление перегрузкой (CCID) | 2 | Y |
| 2 | Возможность коротких номеров | 1 | Y |
| 3 | Окно последовательности | 100 | Y |
| 4 | Невозможность ECN | 0 | N |
| 5 | Ack Ratio | 2 | N |
| 6 | Послать вектор Ack | 0 | N |
| 7 | Послать число NDP | 0 | N |
| 8 | Минимальное покрытие контрольной суммой | 0 | N |
| 9 | Проверка контрольной суммы данных | 0 | N |
| 10-127 | Зарезервировано | | |
| 128-255 | CCID-Specific Features (специфические характеристики) | | |

### *Опция медленного получателя*

НСП-олучатель посылает опцию Slow Receiver (медленный получатель) своему отправителю, чтобы уведомить его о проблемах с обработкой его данных.

### *Опция отброшенных данных*

Опция Data Dropped (отброшенных данных) указывает, что некоторые пакеты, отмеченные как полученные, на самом деле были отброшены, прежде чем были переданы приложению. Механизм управления перегрузкой отправителя может реагировать на потерянные информационные пакеты.

### *Опция контрольной суммы поля данных*

Опция контрольной суммы поля данных содержит в себе 16-битное дополнение по модулю 1 суммы всех 16-битных слов, содержащихся в поле данных DCCP (информации, транспортируемой пакетами DCCP-Request, DCCP-Response, DCCP-Data, DCCP-DataAck или DCCP-Move). В сочетании с длиной контрольной суммы, меньшей 15, это позволяет различить случаи повреждения заголовка и поля данных пакета. Пакеты с поврежденным заголовком должны рассматриваться как отброшенные сетью, в то время как пакеты с поврежденным полем данных могут обрабатываться дифференцированно, например, реакция отправителя на повреждение может быть менее жесткой, чем на перегрузку.

### *Многодомность и мобильность*

Протокол DCCP осуществляет простую поддержку многодомности и мобильности за счет механизма переадресации партнеров. Мобильный партер должен согласовать поддержку мобильности заранее. Когда мобильный партнер получает новый адрес, он посылает пакет DCCP-Move с этого адреса стационарной станции. Стационарная станция изменяет состояние соединения с учетом изменившегося адреса партнера. Поддержка DCCP мобильности призвана решить лишь простейшие проблемы многодомности и мобильности. Например, DCCP не имеет поддержки одновременного перемещения партнеров. Приложениям, где требуется более сложная семантика мобильности или более сильные гарантии безопасности, следует использовать существующие решения, подобные Mobile IP или [SB00].

### *Мультиплексирование*

В противоположность TCP, DCCP не предлагает надежно упорядоченную доставку. Как следствие, в случае DCCP нет осложнений для слоев, размещенных выше DCCP, в частности, для мультиплексирования субпотоков в рамках одного DCCP соединения.

## DCCP и RTP

Транспортный протокол реального времени, RTP [RFC-1889], в настоящее время используется (поверх UDP) многими приложениями DCCP. Существует два потенциальных источника избыточности в комбинации RTP поверх DCCP: задублированные данные подтверждения и задублированные порядковые номера. Этот источник избыточности добавляет 4 байта на пакет по отношению к RTP поверх UDP. Однако конкретные CCID могут использовать место, занимаемое порядковым номером RTP.

## Соображения безопасности

Протокол DCCP не предоставляет криптографических гарантий безопасности. Приложения, требующие серьезной безопасности должны использовать IPsec или какую-то иную схему безопасности.

Несмотря ни на что, DCCP имеет возможности противостоять некоторым видам атак. Этому способствует используемая система нумерации пакетов.

Текущий список Интернет-проектов, в том числе по направлению DCCP, доступен по адресу http://www.ietf.org/ietf/1idabstracts.txt.

## 3.2. Протокол TFRC

### Введение

Актуальность проблем транспортировки мультимедийных данных по каналам Интернет с QoS=лучшее_возможное стимулирует разработки модификаций протоколов, управляющих перегрузкой (DCCP, TFRC и т.д.).

Протокол TFRC (TCP Friendly Rate Control - RFC-3448) предоставляет механизм управления перегрузкой для уникастных потоков в Интернет [3.2]. Ниже рассматривается способ управления перегрузкой, который может быть использован, например, в транспортном протоколе RTP [3.7], в приложении с контролем перегрузки в виртуальном канале точка-точка [3.1].

TFRC спроектирован так, чтобы обеспечивать справедливое распределение ресурсов при конкуренции с другими TCP потоками; под "справедливой" подразумевается ситуация, когда поток получает до двух раз большую полосу, чем TCP при тех же условиях. Однако TFRC имеет много меньшие вариации пропускной способности по сравнению с TCP, что делает его более удобным для таких приложений, как телефония или потоковое видео, где важна относительно постоянная скорость передачи.

Эти преимущества достигаются за счет того, что TFRC реагирует медленнее, чем TCP на изменения доступной полосы. Таким образом, TFRC следует использовать, только когда приложение требует почти постоянной пропускной способности, в частности, избегая двойного снижения пропускной способности при потере одного пакета, как это происходит в случае TCP. Для приложений, которым нужно передать как можно больше данных за ограниченное время, следует порекомендовать TCP, или, если надежность не требуется, можно применить схемы управления перегрузкой AdditiveIncrease, MultiplicativeDecrease (AIMD).

TFRC спроектирован для приложений, которые используют пакеты фиксированной длины, и варьируют скорость передачи пакетов в случае возникновения перегрузки. Некоторые аудиоприложения имеют фиксированный период следования пакетов и изменяют размер пакетов при возникновении перегрузки (TFRCPS - PacketSize). Эту версию предполагается разработать позднее.

Протокол TFRC реализует механизм, ориентированный на получателя, с вычислением характеристик перегрузки (т.е., частоты потери пакетов) на стороне получателя, а не отправителя. Это прекрасно подходит для приложений, где отправителем является большой сервер, обслуживающий большое число соединений одновременно, а получатель имеет достаточно памяти и ресурсов ЦПУ для вычислений. Кроме того, механизм, базирующийся на получателе, более удобен для реализации управления перегрузкой при мультикастинге.

### Принцип работы

В процессе реализации механизма управления перегрузкой TFRC непосредственно использует выражение для пропускной способности для обеспечения скорости передачи, как функции частоты потери пакетов и RTT. Для того чтобы успешно конкурировать с TCP, TFRC использует уравнение пропускной способности TCP, которое описывает скорость передачи TCP как функцию частоты потери пакетов, RTT и размера сегмента. Определим случай потери как потерю одного или более помеченных пакетов в пределах окна, где термин "помеченный пакет" относится к индикации пере-

грузки в рамках режима ECN (Explicit Congestion Notification) [3.6]. Механизм управления перегрузкой TFRC работает следующим образом.

- Получатель измеряет частоту потери пакетов и передает эту информацию отправителю.
- Отправитель использует это сообщение, кроме прочего, для измерения RTT.
- Частота потери пакетов и RTT подставляются в выражение пропускной способности для определения приемлемой величины скорости передачи.
- Отправитель подстраивает скорость передачи под рассчитанное значение.

Динамика работы TFRC чувствительна к тому, как выполняется и используется измерение.

## Уравнение пропускной способности TCP

Любое реалистическое выражение для пропускной способности TCP, как функции частоты потерь и RTT, приемлемо для реализации TFRC. Однако, замечено, что выражение для пропускной способности TCP должно отражать поведение таймаута повторной передачи, так как именно этот эффект доминирует при высокой вероятности потери сегмента.

Выражение для пропускной способности, рекомендуемое для TFRC, является несколько упрощенной версией формулы, используемой для Reno TCP [3.4]. В идеале предпочтительно выражение для пропускной способности, базирующееся на модели SACK TCP, но такого выражения пока не получено. Выражение для пропускной способности имеет вид:

$$X = \frac{s}{R \times SQRT(2 \times b \times p/3) + (t_RTO \times (3 \times SQRT(3 \times b \times p/8) \times p \times (1 + 32 \times p^2)))}$$

где X - скорость передачи в байтах в сек, s длина пакета в байтах, R = RTT в секундах, p - вероятность потери сегментов (между 0 и 1.0). t_RTO является значением времени таймаута повторной передачи в секундах, b - число пакетов, подтверждаемых одним TCP ACK.

При реализации протокола используется значение t_RTO = $4 \times R$. Возможно и более точное выражение для расчета t_RTO, но эксперименты показывают, что предложенное выражение позволяет достичь вполне приемлемых результатов [3.9]. Другой способ - использование выражения t_RTO = max(4R, 1сек), для того, чтобы обеспечить рекомендуемый минимум RTO, равный одной секунде [3.5].

Многие современные реализации TCP соединений используют задержанные подтверждения, посылая подтверждения для пары полученных пакетов, и, таким образом, используют b = 2. Однако в TCP разрешается посылать подтверждения для каждого информационного пакета, что соответствует b = 1. Так как большинство реализаций не используют задержанных подтверждений, рекомендуется работать с b = 1.

Параметры s (размер пакета), p (частота потерь) и R (RTT) должны измеряться или вычисляться программами, реализующими протокол TFRC.

## Содержимое пакетов

Рассмотрим сначала содержимое информационных пакетов, посылаемых отправителем, и пакетов обратной связи, отправляемых получателем. Так как TFRC будет использоваться совместно с транспортным протоколом, форматы пакетов не специфицируются.

### *Информационные пакеты*

Каждый информационный пакет, посланный отправителем, содержит следующие данные:

- номер по порядку. Это число инкрементируется на 1 после передачи очередного пакета. Поле должно быть достаточно велико, чтобы исключить возможность появления двух пакетов с идентичными номерами в пределах записанной истории пришедших пакетов у получателя;
- временную метку, отмечающую время отправки пакета. Обозначим временную метку для пакета с номером i как ts_i. Разрешающая способность для таких меток должна быть на уровне миллисекунд. Метки используются получателем для определения, принадлежит ли потеря одному и тому же событию потери. Также они применяются для оценки значения RTT. Альтернативой меткам, инкрементируемым каждую миллисекунду, могут быть метки, которые инкрементируются каждую четверть RTT. Оценку RTT, полученную по пакету i, обозначим R_i. Оценка RTT используется получателем совместно с временной меткой, для определения, принадлежат ли множественные потери одному и тому же событию.

### *Пакеты обратной связи*

Каждый пакет обратной связи, посылаемый получателем, содержит следующую информацию.

- Временная метка последнего полученного пакета. Обозначим ее как t_recvdata. Если последний полученный пакет имеет номер i, тогда t_recvdata = ts_i. Эта временная метка используется отправителем для оценки RTT.
- Время с момента прихода последнего информационного пакета на вход получателя до генерации сообщения обратной связи. Обозначим это время как t_delay.
- Частота, с которой получатель оценивает то, что данные были получены, с момента посылки последнего сообщения обратной связи. Обозначим эту величину X_recv.
- Текущая оценка получателем частоты потерь p.

## Протокол отправителя данных

Отправитель данных посылает поток пакетов данных получателю с контролируемой частотой. Когда получен пакет обратной связи от отправителя, отправитель данных изменяет скорость передачи, используя информацию, содержащуюся в сообщении обратной связи. Если получатель не получил сообщения обратной связи за время 2RTT, он должен понизить скорость передачи в два раза. Это делается с помощью таймера nofeedback.

Протокол отправителя включает в себя следующие шаги.

- Измерение средней длины посылаемых пакетов.
- Действия, выполняемые отправителем при получении пакета обратной связи.
- Действие отправителя, когда истекает время таймаута nofeedback.
- Предотвращение осцилляций (опционно).
- Диспетчеризация передачи для операционных систем, работающих не в режиме реального времени.

### *Измерение длины пакетов*

Параметр s (размер пакета) обычно известен приложению. Исключение составляют два случая:

- размер пакета варьируется в зависимости от передаваемых данных. Если вариация размера пакета не коррелирована с изменением скорости передачи, оценка среднего размера пакета s проблем не составляет;
- приложение для управления перегрузкой должно изменить размер пакетов, а не частоту их посылки. Так происходит в ситуации аудиоприложений: каждый пакет соответствует определенному временному интервалу.

### *Инициализация отправителя*

Для инициализации отправителя значение X делается равным 1 пакету/сек, а время таймера feedback устанавливается > 2 сек. Начальные значения для величин R (RTT) и t_RTO не присваиваются. Начальное значение tld (Time Last Doubled) во время медленного старта делается равным 1.

Поведение отправителя при получении пакета обратной связи

Отправитель знает свое текущее значение скорости передачи пакетов X и поддерживает его в течение RTT (R).

Когда отправителю в момент t_now приходит пакет обратной связи, должны быть осуществлены следующие действия:

1. вычисляется новое значение RTT;
2. обновляется оценка RTT.

   Если не получено обратной связи до этого, то R = R_sample.

   В противном случае

   R = q × R + (1q) × R_sample;

   Протокол TFRC не чувствителен к точности фильтрующей константы q, но рекомендуется значение по умолчанию q=0.9;

3. обновляется время таймаута:

   t_RTO = $4 \times R$;

4. обновляется скорость передачи согласно:

   Если (p > 0)

   На основе выражения для пропускной способности TCP вычисляем X_calc.

   X = max(min(X_calc, 2 × X_recv), s/t_mbi);

   В противном случае

   Если (t_now - tld >= R)

   X = max(min(2 × X, 2 × X_recv), s/R);

   tld = t_now;

   Заметим, что если p == 0, отправитель находится в состоянии медленного старта, когда он практически удваивает скорость передачи за время RTT. Значение s/R определяет минимальную скорость передачи в период медленного старта, равную одному пакету за RTT. Параметр t_mbi равен 64 секундам и характеризует максимум межпакетной задержки передачи в отсутствии обратной связи. Таким образом, когда p > 0, отправитель посылает как минимум по одному пакету каждые 64 секунды;

5. устанавливается таймер nofeedback на время после $max(4 \times R, 2 \times s/X)$ секунд.

***Истечение времени таймера nofeedback***

Если время таймера nofeedback истекает, отправитель должен выполнить следующие действия:

1. уменьшить скорость передачи вдвое. Если отправитель получил сообщение обратной связи от получателя, применяется модификация кэшированной копии X_recv. Так как скорость передачи ограничена 2 X_recv, модификация X_recv ограничивает текущее значение скорости передачи, позволяет отправителю начать медленный старт, удваивая скорость передачи для каждого RTT, если сообщения обратной связи подтверждают отсутствие потерь.

   Если (X_calc > 2 × X_recv)

   X_recv = max(X_recv/2, s/(2 × t_mbi));

   В противном случае

   X_recv = X_calc/4;

   Выражение s/(2 × t_mbi) ограничивает частоту одним пакетом каждые 64 секунды в случае постоянного отсутствия обратной связи.

2. значение X должно быть повторно вычислено, как это описано в пункте (4) выше.

   Если истекает время таймера nofeedback, когда отправитель еще не имеет реального значения RTT и не получил еще сообщения обратной связи от получателя, тогда шаг (1) может быть обойден, а скорость передачи сразу сокращена вдвое:

   X = max(X/2, s/t_mbi)

3. перезапускается таймер nofeedback на время $max(4 \times R, 2 \times s/X)$ секунд.

   Заметим, что когда отправитель прерывает передачу, получатель прекратит отправку сообщений обратной связи. Это вызовет запуск таймера nofeedback и уменьшение X_recv. Если отправитель позднее начинает передачу снова, X_recv ограничит скорость передачи и будет реализовываться обычный медленный старт, пока скорость передачи не достигнет X_calc.

   Если отправитель был пассивен с момента таймаута nofeedback, а X_recv меньше четырех пакетов за RTT, тогда X_recv не должно сокращаться вдвое в ответ на истечение времени таймаута. Это гарантирует, что разрешенная скорость передачи никогда не будет сокращена ниже двух пакетов за RTT после периода пассивности.

### *Предотвращение осцилляций*

Для предотвращения осцилляций полезно уменьшить скорость передачи отправителя - это приведет к сокращению длины очереди и, следовательно, RTT. Чтобы достичь этого, отправитель поддерживает оценку долгосрочного значения RTT и модифицирует скорость передачи в зависимости от того, на сколько отличается последнее значение RTT от этой оценки. Долгосрочное значение R_sqmean определяется следующим образом:

если до этого не получено сообщения обратной связи

R_sqmean = sqrt(R_sample);

в противном случае

R_sqmean = q2*R_sqmean + (1q2)*sqrt(R_sample);

Таким образом, R_sqmean представляет собой экспоненциально взвешенное среднее значение квадратного корня из RTT. Константа q2 должна быть установлена аналогично q, рекомендуемое значение равно 0.9.

Базовое значение скорости передачи X отправитель получает из функции пропускной способности. Далее вычисляется модифицированное мгновенное значение частоты X_inst:

X_inst = X * R_sqmean / sqrt(R_sample);

Когда sqrt(R_sample) больше, чем R_sqmean, тогда очередь обычно растет, а скорость передачи должна быть понижена.

### *Диспетчеризация отправки пакетов*

Так как TFRC базируется на скорости передачи, а операционные системы обычно не могут диспетчеризовать события достаточно точно, необходимо воспользоваться любой возможностью отправки пакетов, чтобы обеспечить нужную среднюю скорость передачи. Таким образом, обычный цикл отправки обеспечит правильный межпакетный интервал, t_ipi:

t_ipi = s/X_inst;

Когда отправитель начинает передачу в момент t_0, он вычисляет t_ipi и рассчитывает номинальное время посылки для первого пакета t_1 = t_0 + t_ipi. Когда приложение переходит в состояние простоя, оно проверяет текущее время, t_now и затем запрашивает повторную диспетчеризацию после (t_ipi - (t_now - t_0)) секунд. Когда приложение диспетчеризовано, оно снова проверяет текущее время, t_now. Если (t_now > t_1 - delta), посылается первый пакет.

Теперь может быть вычислено новое значение t_ipi, которое используется для вычисления номинального времени отправки t_2 пакета 2: t2 = t_1 + t_ipi. Далее процесс повторяется для каждого последующего пакета.

В некоторых случаях, когда номинальное время отправки следующего пакета t_i вычислено, может так случиться, что t_now > t_i - delta. В таком варианте пакет должен быть послан немедленно. Таким образом, если операционная система имеет таймер с низким разрешением, а скорость передачи высока, тогда TFRC может посылать короткие группы пакетов, которые разделены интервалами, соответствующими дискретам разрешения таймера ОС.

Параметр delta служит для того, чтобы придать некоторую гибкость при выборе момента отправки пакета. Если операционная система имеет разрешение таймера t_gran секунд, тогда значение delta вычисляется согласно:

delta = min(t_ipi/2, t_gran/2);

t_gran равно 10мсек для многих систем Unix. Если t_gran неизвестно, можно уверенно предполагать значение 10мсек.

## Вычисление частоты потери пакетов (p)

Получение точных и стабильных результатов измерений частоты потерь является крайне важным для протокола TFRC. Измерение частоты потерь выполняется получателем, основываясь на детектировании потерь помеченных пакетов. Делается это путем регистрации порядковых номеров полученных пакетов.

### *Детектирование потерянных помеченных пакетов*

TFRC предполагает, что все пакеты имеют порядковые номера, которые инкрементируются на 1 при отправке очередного пакета.

В рамках протокола TFRC предполагается, что, если потерянный пакет посылается повторно, он получает новый порядковый номер, а не номер потерянного пакета. Если транспортный протокол требует повторной передачи с первоначальным порядковым номером, разработчик протокола должен позаботиться о том, чтобы различать задержавшийся и повторно переданный пакеты и детектировать потери в случае повторной передачи.

Получатель поддерживает структуру данных, которая позволяет отслеживать, какие пакеты пришли, а какие нет. Предполагается, что структура данных состоит из списка номеров пакетов, которые пришли, и временных меток, указывающих на время прихода.

Потеря пакета детектируется при получении, по крайней мере, трех пакетов с номерами больше, чем у потерянного. Требование трех пакетов идентично регламентации протокола TCP и делает TFRC более устойчивым в случае перепутывания порядка следования пакетов. В противоположность TCP, если пакет приходит поздно (после прибытия 3 последующих пакетов) в TFRC, опоздавший пакет может заполнить пробел в рекорде приема TFRC и получатель может повторно вычислить частоту потери пакетов.

Для соединений ECN получение помеченного пакета воспринимается как сигнал перегрузки, не дожидаясь прихода последующих пакетов.

### *Работа с предысторией потерь пакетов*

TFRC требует, чтобы система была устойчива в отношении потери нескольких пакетов подряд. Здесь имеется аналогия с TCP, где обычно выполняется сокращение CWND только вдвое за время одного RTT. Таким образом, получатель должен отслеживать историю потери пакетов в рамках рекорда событий, где событием считается потеря одного или более пакетов за RTT. Чтобы выполнить это отслеживание, получатель должен знать, какое значение RTT использовать, - эти данные периодически предоставляет отправитель в виде управляющей информации, присоединенной к пакетам данных. TFRC не чувствителен к тому, как выполнены измерения RTT, но для этих целей рекомендуется применять RTT, измеренное отправителем.

Чтобы определить, является ли потерянный, помеченный пакет началом нового случая потери или его следует отнести к уже зарегистрированному случаю, нужно сравнить порядковый номер и временную метку пакета, доставленного получателю. Для помеченного пакета S_new его время приема T_new может быть записано непосредственно. Для потерянного пакета номинальное "время прибытия" можно получить методом интерполяции. Пусть:

S_loss равно порядковому номеру потерянного пакета;

S_before равно порядковому номеру последнего потерянного пакета, который должен был прибыть до S_loss;

S_after - порядковый номер первого пакета, который должен прийти после S_loss;

T_before является временем приема S_before;

T_after является временем прихода S_after.

Заметим, что T_before может оказаться до или после T_after изза смены порядка прихода пакетов. Для потерянного пакета S_loss можно интерполировать номинальное "время прибытия", зная времена S_before и S_after. Таким образом:

T_loss = T_before + ( (T_after - T_before)* (S_loss - S_before)/(S_after - S_before) );

Заметим, что если последовательность номеров оказалась разорвана между S_before и S_after, то порядковые номера должны быть модифицированы с учетом этого, прежде будут начаты вычисления. Если наибольший возможный порядковый номер равен S_max и S_before > S_after, то замена каждого порядкового номера S на S' = (S + (S_max + 1)/2) mod (S_max + 1) вполне достаточна.

Если потерянный пакет S_old был идентифицирован как начинающий предыдущее событие потери и S_new только что определено как потерянное, то мы интерполируем номинальные времена прибытия S_old и S_new как T_old и T_new соответственно.

Если T_old + R >= T_new, то S_new является частью существующего события потери. В противном случае S_new - первый пакет нового события потери.

### *Интервал между случаями потери пакетов*

Если интервал потерь A начинается с порядкового номера пакета S_A, а следующий интервал потерь B начинается с порядкового номера пакета S_B, то число пакетов в интервале потерь A равно (S_B - S_A).

### *Среднее время между потерями*

Чтобы вычислить частоту потерь p, нужно сначала определить средний период потерь. Это делается с помощью фильтра, который взвешивает n последних интервалов событий потери таким образом, что измеренная частота событий потерь меняется плавно.

Веса для w_0 - w_(n1) вычисляются следующим образом:

если (i < n/2)

w_i = 1;

в противном случае

w_i = 1 - (i - (n/2 - 1))/(n/2 + 1);

таким образом, если n=8, величины от w_0 до w_7 равны:

1.0, 1.0, 1.0, 1.0, 0.8, 0.6, 0.4, 0.2

Величина n для числа интервалов потери, используемая при вычислениях частоты потерь, определяет скорость реакции TFRC на изменения уровня перегрузки. Как это стандартизовано в настоящее время, TFRC не должен применяться для величин n значительно больше 8, для трафика, который может конкурировать в Интернет с TCP.

При вычислении среднего интервала потерь мы должны решить, следует ли включать самое последнее событие потерь. Мы это делаем только в случае, когда оно достаточно велико, чтобы увеличить средний интервал потерь.

Таким образом, если самыми последними интервалами потерь являются I_0 до I_n, где I_0 - интервал с момента самого последнего случая потерь, то мы вычисляем средний интервал потери I_mean как:

```
I_tot0 = 0;
I_tot1 = 0;
W_tot = 0;
```

```
for (i = 0 to n1) { I_tot0 = I_tot0 + (I_i * w_i);  W_tot = W_tot + w_i;  }
for (i = 1 to n) { I_tot1 = I_tot1 + (I_i * w_(i1)); }
I_tot = max(I_tot0, I_tot1);
I_mean = I_tot/W_tot;
```

Частота потери пакетов p равна:

p = 1 / I_mean;

### *Учет предыстории*

Как описано выше в разделе "Вычисление частоты потери пакетов", самому последнему интервалу потерь при оценке усредненного интервала потерь приписывается доля 1/(0.75 × n). В этом разделе рассмотрен опционный механизм обесценивания предыдущих событий (history discounting), подробнее обсуждаемый в [3.3] и [3.9]. Этот механизм позволяет TFRCполучателю настроить веса, акцентируя внимания на весах событий потерь, произошедших относительно недавно. В этой версии самые последние интервалы потерь имеют более чем вдвое больший вес при вычислении усредненной длительности интервала потерь.

Чтобы ввести в действие этот механизм, определяем фактор обесценивания (дискаунт фактор) DF_i для каждого интервала потерь L_i, (i > 0), где каждый фактор обесценивания является числом с плавающей запятой. Массив этих факторов содержит в себе историю обесценивания для каждого из интервалов потерь. В начале все значения DF_i массива приравниваются 1:

```
for (i = 1 to n) { DF_i = 1; }
```

Усредненный интервал потерь вычисляется с использованием n предшествующих интервалов потерь I_1, ..., I_n, и интервала I_0, который относится к пакетам, полученным после последнего события потери. Вычисление усредненного интервала потерь с помощью факторов обесценивания является простой модификацией процедуры, описанной в разделе "Вычисление частоты потери пакетов":

I_tot0 = I_0 * w_0

I_tot1 = 0;

W_tot0 = w_0

W_tot1 = 0;

for (i = 1 to n1) {

I_tot0 = I_tot0 + (I_i × w_i × DF_i × DF);

W_tot0 = W_tot0 + w_i × DF_i × DF; }

for (i = 1 to n) { I_tot1 = I_tot1 + (I_i × w_(i1) × DF_i); W_tot1 = W_tot1 + w_(i1) × DF_i; }

p = min(W_tot0/I_tot0, W_tot1/I_tot1);

Общий фактор обесценивания DF обновляется при получении каждого пакета. Сначала получатель вычисляет взвешенное среднее I_mean интервалов потерь I_1, ..., I_n:

I_tot = 0;

W_tot = 0;

for (i = 1 to n) { W_tot = W_tot + w_(i1) × DF_i; I_tot = I_tot + (I_i × w_(i1) × DF_i); }

I_mean = I_tot / W_tot;

Это взвешенное среднее I_mean сравнивается с I_0, - числом пакетов, полученных с момента последнего события потерь. Если I_0 больше чем вдвое I_mean, тогда новый интервал потерь значительно больше предыдущих и общий фактор обесценивания DF обновляется, чтобы понизить относительный вес старых интервалов:

if (I_0 > 2 × I_mean) {

DF = 2 × I_mean/I_0;

if (DF < THRESHOLD)

DF = THRESHOLD;

} else

DF = 1;

Ненулевое значение THRESHOLD гарантирует, что более старые интервалы потери, относящиеся к предшествующему периоду высокой перегрузки, не будут обесценены вовсе. Рекомендуется установить THRESHOLD равным 0.5. Заметим, что с приходом каждого нового пакета, I_0 будет увеличиваться, а фактор обесценивания DF - обновляться.

Когда происходит новое событие потери, текущий интервал I_0 сдвигается на место I_1, интервал потери I_i занимает место I_(i+1), а интервал потерь I_n забывается. Предшествующий дискаунтфактор DF должен быть введен в дискаунт массив. Так как DF_i несет в себе дискаунтфактор, ассоциированный с интервалом потерь I_i, массив DF_i должен быть также сдвинут. Это делается следующим образом:

for (i = 1 to n) { DF_i = DF × DF_i; }

for (i = n-1 to 0 step - 1) { DF_(i+1) = DF_i; }

I_0 = 1;

DF_0 = 1;

DF = 1;

## Протокол получателя данных

Получатель периодически посылает сообщения обратной связи отправителю. Пакеты обратной связи должны посылаться, по крайней мере, один раз за RTT, если только отправитель не передает меньше одного пакета за RTT, в этом случае сообщения обратной связи должны посылаться после каждого получения информационного пакета. Пакет обратной связи следует также посылать всякий раз, когда детектируется новое событие потери, не дожидаясь конца RTT, и всякий раз, когда фиксируется нарушение порядка следования пакетов.

Если отправитель работает с высокой частотой (посылает много пакетов за RTT), существуют некоторые преимущества посылки сообщений обратной связи периодически, чаще одного раза за RTT, что позволяет быстрее реагировать на изменения результатов измерения RTT и обеспечить большую

устойчивость к потере пакетов обратной связи. Однако, нет никакого смысла посылать много пакетов обратной связи за время RTT.

### *Поведение получателя при получении информационного пакета*

Когда приходит информационный пакет, получатель выполняет следующие действия.

1. Добавление пакета в историю доставки пакетов (packet history).
2. Пусть предыдущее значение p равно p_prev. Вычисляется новое значение p, как это описано в разделе "Протокол отправителя данных".
3. Если p > p_prev, происходит истечение времени таймера обратной связи и выполняются действия, описанные в разделе "Протокол отправителя данных".

Если p <= p_prev, не нужно предпринимать никаких действий. Однако при оптимизации может проверяться, происходит ли при получении пакета заполнение пробела в истории поступления, при котором два события потери сливаются в одно. Если произошло именно это, получатель может также послать сообщение обратной связи немедленно. Влияние такой оптимизации в нормальной ситуации является незначительным.

### *Истечение времени таймера обратной связи*

Когда время таймера обратной связи истекает, необходимые действия зависят от того, был ли получен информационный пакет со времени отправки последнего сообщения обратной связи.

Пусть максимальный порядковый номер пакета для получателя равен S_m, а значение RTT, измеренное с учетом S_m, равно R_m. Если были получены информационные пакеты с момента посылки предыдущего сообщения обратной связи, получатель выполняет следующие действия.

1. Вычисляется средняя частота событий потери.
2. Вычисляется частота поступления пакетов X_recv, на основе полученных пакетов за время предыдущих R_m секунд.
3. Формируется и отправляется пакет обратной связи, содержащий информацию, которая описана в разделе "Содержимое пакетов".
4. Перезапускается таймер обратной связи на время R_m секунд.

Если с момента посылки сообщения обратной связи не получено никаких информационных пакетов, то пакет обратной связи не посылается, а таймер обратной связи перезапускается на время R_m секунд.

### *Инициализация получателя*

Получатель инициализируется первым пакетом, который к нему приходит. Пусть порядковый номер пакета равен i.

Когда первый пакет получен:

- установим p=0;
- установим X_recv = 0;
- подготовим и пошлем пакет обратной связи;
- установим таймер feedback на время R_i секунд.

### *Инициализация истории потерь после первого потерянного сегмента*

Пакеты вплоть до первой потери не могут быть задействованы для вычисления скорости передачи, так как темп передачи в это время меняется слишком быстро. TFRC предполагает, что правильное значение скорости передачи данных после первой потери равно половине скорости передачи, при которой произошла потеря. TFRC аппроксимирует эту величину с помощью X_recv, частотой приема за последний период RTT. После первой потери TFRC получатель вычисляет интервал потери, который был бы необходим, чтобы обеспечить скорость передачи данных X_recv, и использует этот искусственный интервал потери для запуска механизма предыстории потерь.

В протоколе TFRC это делается путем нахождения некоторого значения p, для которого выражение пропускной способности из раздела "Уравнение пропускной способности TCP" дает величину пропускной способности в пределах 5% от X_recv, заданный текущий размер пакета s и RTT R. Первый интервал потерь устанавливается тогда равным 1/p.

## Варианты протокола, базирующиеся на отправителе

Вполне возможно реализовать вариант TFRC, базирующийся на модели отправителя, где получатель использует надежную доставку для посылки отправителю данных о потерях пакетов, а отправитель вычисляет частоту потерь и приемлемую скорость посылки пакетов. Однако эта модель здесь не рассматривается.

Главным преимуществом модели TFRC, базирующейся на отправителе, является то, что отправитель в этом случае не должен доверять расчетам потерь, выполненным получателем. Однако в условиях требований надежной доставки данных о потерях от получателя отправителю модель TFRC, базирующаяся на модели отправителя, предъявляет более жесткие ограничения к транспортному протоколу.

Напротив, модель TFRC, базирующаяся на получателе, устойчива к потере пакетов обратной связи и, следовательно, не требует надежной доставки пакетов обратной связи. Эта модель также лучше годится для таких приложений, как потоковое мультимедиа для WEBсерверов, где обычно желательно перенести часть нагрузки с сервера на клиента.

## Реализации

В данном разделе специфицируется реализации механизма управления перегрузкой TFRC, для приложений и транспортных протоколов.

Для t_RTO = 4 × R и b = 1 выражение пропускной способности в разделе "Уравнение пропускной способности TCP" может быть преобразовано к виду:

$$X = \frac{s}{R \times f(p)}$$

Для $f(p) = sqrt(2 \times p/3) + (12 \times sqrt(3 \times p/8) \times p \times (1 + 32 \times p^2))$.

Для вычисления функции f(p) могут использоваться lookup-таблицы.

Заметим, что опционный механизм подавления осцилляций для отправителя, описанный в разделе "Предотвращение осцилляций", использует вычисление квадратных корней.

Вычисление среднего значения интервала потерь в разделе "Интервал между случаями потери пакетов" включает в себя умножения на веса w_0 - w_(n-1), которые для n=8 равны:

1.0, 1.0, 1.0, 1.0, 0.8, 0.6, 0.4, 0.2.

С минимальной потерей гладкости можно использовать веса, которые являются степенью 2 или суммами степеней 2, напр.,

1.0, 1.0, 1.0, 1.0, 0.75, 0.5, 0.25, 0.25.

Опционный механизм "исторического дискаунтинга" используется для расчете усредненной частоты потерь. Механизм дискаунтинга включается, только когда приходится сталкиваться с достаточно протяженными периодами без потери пакетов. Для обеспечения более высокой эффективности работы дискаунтфакторы DF_i могут делаться равными целым степеням 2.

## Навязывание хосту ложного маршрута с использованием протокола ICMP

Общеизвестно, что маршрутизация в Сети играет важнейшую роль для обеспечения ее нормального функционирования. Маршрутизация в Internet осуществляется на сетевом уровне (IP-уровень). На программном уровне для ее обеспечения в памяти сетевой операционной системы каждого хоста существуют специальные таблицы, содержащие данные о возможных маршрутах. На аппаратном уровне каждый сегмент Сети подключен к глобальной сети как минимум через один маршрутизатор, а следовательно, все хосты и маршрутизатор должны физически располагаться в одном сегменте. Поэтому все сообщения, адресованные в другие сегменты Сети, направляются на маршрутизатор, который, в свою очередь, перенаправляет их далее по указанному в пакете IP-адресу, выбирая при этом оптимальный маршрут. Рассмотрим, что представляет собой таблица маршрутизации хоста. Она состоит из пяти колонок: сетевой адрес, сетевая маска, адрес маршрутизатора, интерфейс и метрика (см. рис. 4.9)

```
C:\>route print
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x100003 ...00 53 45 00 00 00 ...... WAN (PPP/SLIP) Interface
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.0.35    192.168.0.35       1
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
     192.168.0.35  255.255.255.255        127.0.0.1       127.0.0.1      50
     192.168.0.37  255.255.255.255     192.168.0.35    192.168.0.35       1
    192.168.0.255  255.255.255.255     192.168.0.35    192.168.0.35      50
        224.0.0.0        240.0.0.0     192.168.0.35    192.168.0.35       1
Default Gateway:      192.168.0.35
===========================================================================
Persistent Routes:
  None
```

*Рис. 4.9. Таблица маршрутизации хоста*

В каждой строке этой таблицы содержится описание соответствующего маршрута, которое включает IP-адрес конечной точки пути (сетевой адрес), IP-адрес соответствующего ближайшего маршрутизатора (адрес маршрутизатора), а также ряд других параметров, характеризующих этот путь. Обычно в системе существует так называемый маршрут по умолчанию (поле "сетевой адрес" содержит значение 0.0.0.0, заданное по умолчанию, а поле "адрес маршрутизатора" - IP-адрес маршрутизатора): все пакеты, посылаемые на IP-адрес вне пределов данной подсети, будут направляться по этому пути, то есть па маршрутизатор. IP-адресация, как адресация на сетевом уровне, была задумана именно для межсегментной передачи данных из одной точки глобальной сети в другую. Для обращения на канальном уровне используются аппаратные адреса сетевых карт. Если бы Сеть представляла собой всего один сегмент, то дополнительной IP-адресации не потребовалось бы, так как аппаратных адресов сетевых адаптеров было бы вполне достаточно для непосредственной пересылки пакетов. Сегодня Internet представляет собой совокупность сегментов, соединенных через маршрутизаторы, и в Сети мы вынуждены использовать систему двойной адресации (по аппаратным и сетевым адресам). Поэтому, когда пакет направляется в другой сегмент Сети, в заголовке сетевого уровня указывается IP-адрес назначения, а в заголовке канального уровня - Ethernet-адрес ближайшего маршрутизатора.

## Немного о ICMP

Как известно, в сети Internet существует управляющий протокол ICMP (Internet Control Message Protocol), одной из функций которого является удаленное управление таблицей маршрутизации на хостах внутри сегмента Сети. Динамическое удаленное управление маршрутизацией изначально задумывалось для предотвращения возможной передачи сообщений по неоптимальному маршруту, а также для повышения отказоустойчивости Сети в целом. Предполагалось, что сетевой сегмент может быть подключен к Internet не через один (как это обычно происходит), а через несколько маршрутизаторов. В этом случае адресоваться во внешнюю сеть можно через любой из ближайших узлов. Это довольно удобно как с точки зрения оптимальности маршрута (например, к хосту www.example.one кратчайший маршрут проходит через первый маршрутизатор, а к хосту www.example.two - через второй маршрутизатор), так и с точки зрения повышения надежности работы Сети: при выходе из строя одного из маршрутизаторов связь с внешним миром возможна через другой маршрутизатор. В обоих случаях - как при изменении оптимального маршрута, так и при выходе из строя маршрутизатора - необходимо изменение таблицы маршрутизации в памяти сетевой операционной системы. Одно из назначений протокола ICMP состоит именно в *динамическом* изменении таблицы маршрутизации оконечных сетевых систем.

Итак, в Internet удаленное управление маршрутизацией реализовано в виде передачи с маршрутизатора на хост управляющего ICMP-сообщения Redirect Message (Перенаправить сообщение). Заголовок сообщения представлен на рис. 4.10.

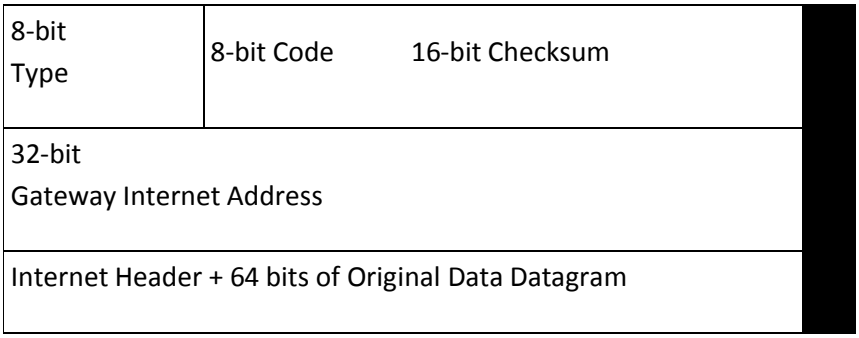



*Рис. 4.10. Заголовок сообщения ICMP Redirect Message*

Поля сообщения принимают следующие значения: поле Type - 5, поле Code - 0 = Redirect datagrams for the Network, 1 = Redirect datagrams for the Host, 2 = Redirect datagrams for the Type of Service and Network или 3 = Redirect datagrams for the Type of Service and Host.

Как следует из спецификации протокола ICMP, сообщение Redirect бывает двух видов. Первый вид сообщения (code 0) носит название *Redirect Datagrams for the Network* (Таблица данных перенаправления в сетях) и уведомляет хост о необходимости смены адреса маршрутизатора, заданного по умолчанию, или о смене маршрута к определенной подсети. Второй вид (code 1) - *Redirect Datagrams for the Host* (Таблица данных перенаправления для хоста) - информирует хост о том, что следует создать новый маршрут к отмеченному в сообщении объекту и внести его в таблицу маршрутизации, указывая IP-адрес хоста, для которого нужна смена маршрута (адрес будет занесен в поле Destination (Назначение) в пристыкованном IP-заголовке), и новый IP-адрес маршрутизатора, куда необходимо направлять пакеты для данного хоста (этот адрес заносится в поле Gateway). В результате исследования реакции различных сетевых систем на данное ICMP-сообщение выяснилось важное ограничение, накладываемое на IP-адрес нового маршрутизатора, - он должен быть в пределах адресов данной подсети.

Исследование публикаций, посвященных протоколу ICMP, показало, что сообщение Redirect Datagrams for the Network устарело и уже не используется современными сетевыми ОС. Это подтверждает и проведенный авторами анализ исходных текстов ОС Linux 1.2.8. Иначе дело обстоит с управляющим сообщением Redirect Datagrams for the Host: здесь нет такого единства в рядах разра-

ботчиков различных сетевых ОС. Наши исследования показали, что на практике многие из известных UNIX-систем игнорируют это сообщение. Например, если в версии Linux 1.2.8 еще была реакция на Redirect Datagrams for the Host, то уже в версии Linux 2.0.0 такое сообщение игнорировалось. Иначе обстоит дело с ОС фирмы Microsoft - Windows 95 и Windows NT. Опасность Redirect Datagrams for the Host не была принята во внимание, и обе системы реагируют на это сообщение, позволяя удаленно изменять таблицу маршрутизации на любом Windows-хосте.

Итак, мы подошли к основному вопросу: в чем же потенциальная опасность реакции сетевой ОС на ICMP-сообщение Redirect Datagrams for the Host? Ответ очевиден: существует возможность несанкционированного изменения маршрутизации на хосте. Что может помешать хакеру самому послать подобное сообщение и навязать системе ложный маршрут? К сожалению, ничего. Анализ механизма идентификации ICMP-сообщения Redirect показывает, что единственным параметром, идентифицирующем это сообщение, является IP-адрес отправителя, который должен совпадать с IP-адресом маршрутизатора, так как это сообщение может и должно передаваться только маршрутизатором. Особенность протокола ICMP состоит в том, что он не предусматривает никакой дополнительной аутентификации отправителей. То есть ICMP-сообщения передаются на хост маршрутизатором однонаправлено без создания виртуального соединения (в процессе создания такого соединения можно было бы динамически, например по схеме Диффи-Хелмана, выработать идентифицирующую информацию). А так как ни хост, ни маршрутизатор не имеют заранее определенной статической ключевой информации для идентификации ICMP-сообщения Redirect, то очевидно, что в данном случае у получателя такого сообщения нет возможности установить подлинность его отправителя. Следовательно, ничто не мешает атакующему самому послать ложное ICMP-сообщение Redirect о смене маршрута от имени ближайшего роутера.

Приведенные выше факты позволяют осуществить типовую удаленную атаку "внедрение в распределенную ВС ложного объекта путем навязывания ложного маршрута".

Для осуществления этой удаленной атаки необходимо подготовить ложное ICMP-сообщение Redirect Datagrams for the Host, где указать конечный IP-адрес маршрута (адрес хоста, маршрут к которому будет изменен) и IP-адрес ложного маршрутизатора. Затем такое сообщение передается на атакуемый хост от имени маршрутизатора, для чего в IP-заголовке в поле адреса отправителя указывается IP-адрес маршрутизатора. Можно предложить два варианта данной удаленной атаки.

В первом случае кракер находится в том же сегменте сети, что и объект атаки. Тогда, послав ложное ICMP-сообщение, он в качестве IP-адреса нового маршрутизатора может указать свой IP-адрес или любой из адресов данной подсети. Это даст атакующему возможность перемаршрутизировать сообщения, направляемые атакованным хостом на определенный IP-адрес, и получить контроль над трафиком между этим хостом и интересующим взломщика сервером. Далее атака перейдет во вторую стадию, связанную с приемом, анализом и передачей пакетов, получаемых от "обманутого" хоста. Рассмотрим функциональную схему осуществления этой атаки (рис. 4.11):

1. Передача на атакуемый хост ложного ICMP-сообщсния Redirect Datagrams for the Host.
2. Если пришел ARP-запрос от атакуемого хоста, то посылается ARP-ответ.
3. Если пришел пакет от атакуемого хоста, то он переправляется на настоящий маршрутизатор.
4. Если пришел пакет от маршрутизатора, то он переправляется на атакуемый хост.
5. При приеме пакета возможно воздействие на информацию по схеме "ложный объект РВС".

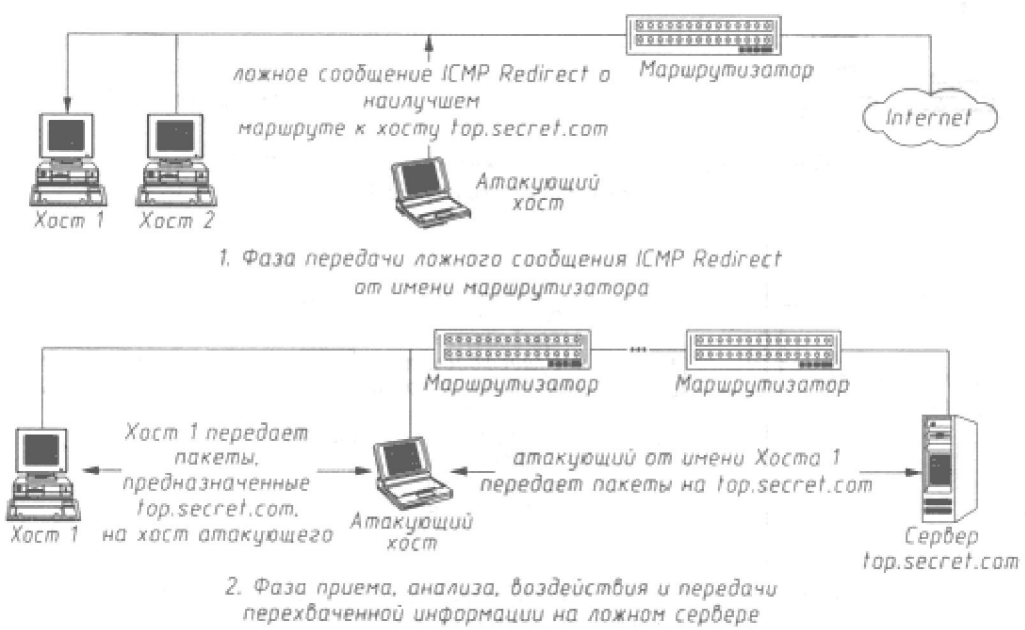



*Рис. 4.11. Внутрисегментное навязывание хосту ложного маршрута при использовании протокола ICMP*

При осуществлении второго варианта удаленной атаки кракер находится в другом сегменте относительно цели атаки. Тогда в случае передачи на атакуемый хост ложного ICMP-сообщения Redirect сам взломщик уже не может получить контроль над трафиком, так как адрес нового маршрутизатора должен находиться в пределах подсети атакуемого хоста. То есть использование данного варианта атаки не позволит атакующему получить доступ к передаваемой по каналу связи информации; в этом случае атака достигает другой цели - нарушения работоспособности хоста (рис. 4.12). Кракер с любого хоста в Internet может послать подобное сообщение на объект атаки, и если сетевая ОС на этом хосте не проигнорирует данное сообщение, то связь между хостом и указанным в ложном ICMP-сообщении сервером нарушится. Это произойдет потому, что все пакеты, направляемые хостом на этот сервер, будут отправлены на IP-адрес несуществующего маршрутизатора. Для того чтобы выполнить такую межсегментную атаку, атакующему необходимо, во-первых, знать внутренний IP-адрес маршрутизатора, к которому подключен хост, и, во-вторых, выбрать имя сервера, которому требуется изменить маршрутизацию.

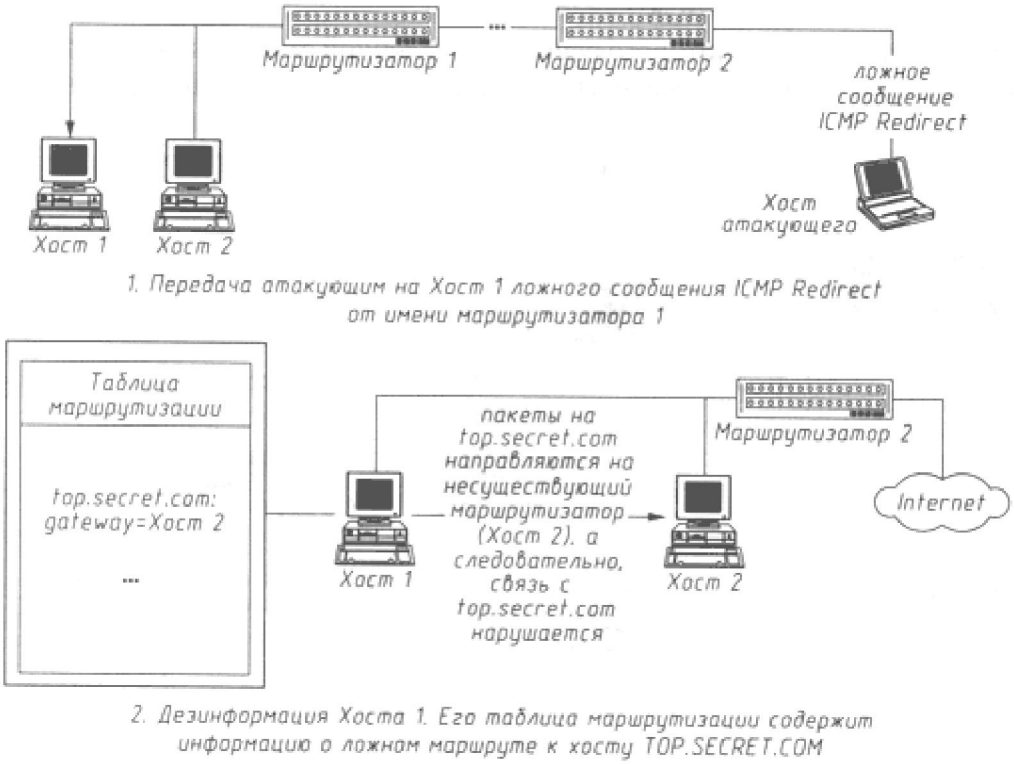



*Рис. 4.12. Межсегментное навязывание хосту ложного маршрута при использовании протокола ICMP, приводящее к отказу в обслуживании*

Первая проблема - внутренний IP-адрес маршрутизатора - решается только простым перебором, так как узнать этот адрес из внешней сети не представляется возможным (например, утилита traceroute даст только IP-адрес маршрутизатора во внешней сети). Так как большинство хостов в сети находится в подсетях класса С, то для осуществления этой атаки достаточно будет послать 256 ICMP-сообщений Redirect с различными IP-адресами отправителя. Например, IP-адрес хоста 194.85.96.197. Предположим, что этот хост находится в подсети класса С, тогда IP-адрес маршрутизатора, к которому он подключен, будет иметь те же первые три байта: 194.85.96.*, и взломщику достаточно будет перебрать только последний байт (послать 256 пакетов).

Вторая проблема для кракера - выбор имени (или IP-адреса) сервера, к которому будет изменена маршрутизация. Очевидно, что узнать, к какому серверу наиболее часто обращается пользователь данного хоста, для атакующего не представляется возможным. Однако взломщик может, посылая неограниченное число ICMP-сообщений Redirect, последовательно изменять маршрутизацию к наиболее известным или часто используемым серверам Internet (поисковые серверы, серверы известных фирм и т.д.). Дело сильно упростится в том случае, если кракер обладает некоторой информацией об атакуемом объекте (например, он является бывшим сотрудником данной фирмы).

Эксперимент показал, что оба варианта рассмотренной удаленной атаки удается осуществить как межсегментно, так и внутрисегментно на ОС Linux 1.2.8, ОС Windows 95 и ОС Windows NT 4.0. Остальные сетевые ОС, исследованные нами (Linux версии выше 2.0.0 и CX/LAN/SX, защищенная по классу Bl UNIX), игнорировали ICMP-сообщенис Redirect, что кажется вполне логичным с точки зрения обеспечения безопасности.

Защититься от этого воздействия можно фильтрацией проходящих ICMP-сообщений Redirect при помощи систем Firewall. Другой способ защиты заключается в изменении исходных текстов сетевого ядра операционных систем с дальнейшей его перекомпиляцией, чтобы запретить реакцию на ICMP-сообщение Redirect. Однако это возможно только в случае свободного распространения исходных текстов ОС (как в случае с ОС Linux или ОС FreeBSD).